


Datamigratie tools

Edlib
an AXIELL product

Axiell ALM Netherlands BV

Inhoud

1 Import.exe

De Adlib-tool *import.exe* werd destijds geïntroduceerd om een snellere manier van importeren te bieden dan het toenmalige DBSETUP. DBSETUP is vervangen door Adlib Designer, waarin importeren even snel gaat als met import.exe. Welk programma u wilt gebruiken voor importeren is dus een kwestie van persoonlijke voorkeur.
Import.exe (6.0 en hoger) herkent DOS, ANSI, UTF-8 en Unicode (zowel big- als little endian) uitwisselbestanden en importeert de gegevens daaruit, ongeacht het type van de database. Voor ANSI en DOS databases geldt hierbij: als er niet-importeerbare Unicode tekens in het uitwisselbestand staan dan zullen die vervangen worden door "?" (een vraagteken). Niet-UTF-8 uitwisselbestanden worden geinterpreteerd als zijnde in ANSI, behalve als het databasetype DOS is.

Maak een back-up
Het importeren van data in bestaande databases kan een ingrijpende procedure zijn. Maak daarom voor de veiligheid van al uw databases een back-up voordat u met importeren begint. Zo kunt u eventuele fouten altijd weer repareren. Zie de Installatiegids voor Museum, Bibliotheek en Archief voor meer informatie over het maken van back-ups.

U moet uw importjob in Designer definiëren, maar u kunt hem uitvoeren met import.exe. Alles wat u nodig hebt zijn deze executable en een Adlib importjob (.imp-bestand).

Start import.exe via de opdrachtregel in een DOS-venster. De syntaxis is als volgt:

```
IMPORT [-?][-a:<adapl>][-i:<jobnaam>]
[-r:<recovery-bestand>][-s][-u:<gebruikersnaam>]
```

Alles tussen vierkante haken [ ] is optioneel, en alles tussen scherpe haken <> moet worden vervangen door een daadwerkelijke naam. En laat bij het invullen alle haakjes weg.

Het volgende:

```
IMPORT -i:..\data\MyImportJob
```

voert bijvoorbeeld alleen maar de importjob genaamd `MyImportJob` in de directory `..\data`. uit.

Maar er zijn nog meer optionele argumenten die u in willekeurige volgorde achter `IMPORT` kunt opsommen. Hieronder vindt u alle mogelijke opties beschreven:

-?	`IMPORT -?` toont Help vergelijkbaar met dit document.
-a:<adapl>	Specificeer een adapl die in plaats van een mogelijk in de importjob ingestelde import-adapl moet worden uitgevoerd.
-b:<uitwisselbestand>	Via deze optie kunt u een importjob uitvoeren met een ander uitwisselbestand (*Invoerbestand*) dan gedefinieerd in die importjob. Zo kunt u één en dezelfde importjob gebruiken om snel meerdere uitwisselbestanden na elkaar in een database te importeren. Bijvoorbeeld: `IMPORT -i:..\data\MyImportJob -b:externedataset2` Let er op de optie `-s` niet te gebruiken wanneer u `-b` gebruikt, anders wordt de vorige import (gedeeltelijk) overschreven.
-c:<cross reference>	Deze optie creëert een adlib.lst-bestand met daarin een lijst van velden en tags van alle databases in de betreffende map (de data-map). Verder wordt een lijst gemaakt van alle gekoppelde velden.
-d:<database> <indextag>	Maak hiermee een dump van een indexbestand zoals dat ook in DBSETUP met **F9** kon, voor een index in een CBF-database (SQL wordt niet ondersteund). De dump komt in een tekstbestand met de naam van het veld en de extensie *.dmp* in de huidige map. Start de opdracht vanuit de Adlib *\data*-map. Voorbeeld van aanroep: `..\executables\import -d:thesau te` Een speciaal geval is de *wordlist.idx*-index: in deze index worden alle unieke woorden uit alle lange-tekstvelden in alle CBF-databases verzameld en van een uniek nummer voorzien. De vrije-

	tekstindexen die lange-tekstvelden (zoals *Titel* en *Beschrijving*) direct indexeren, verwijzen slechts naar het woordnummer uit de wordlist. Om de wordlist te dumpen naar een bestand, gebruikt u de volgende speciale syntaxis: `import –m –d:document wordlist.idx` Verwijs naar een database die vrije-tekst-indexen heeft, bijv. `document` of `collect`. In een Adlib SQL-database kunt u alle indexen direct inzien via SQL Server Management Studio en is er geen noodzaak om indexen via import.exe naar een bestand te dumpen.
-exclusive	Exclusieve toegang. Deze optie specificeert dat de import-executable exclusieve toegang heeft tot de database. Locking en dergelijke acties kunnen nu niet voorkomen en de programmacode die controleert op aanwezige bestands- en record-locks wordt bovendien overgeslagen, waardoor het importeren veel sneller verloopt. De optie kan alleen worden gebruikt wanneer u een importjob uitvoert (`-i:jobnaam`), en voor herindexeren (`-x`).
-fix	Repareer het CBF-bestand terwijl de primaire index wordt geherindexeerd. Gebruik `-fix` alleen na `-x:database %0`. Als er meerdere records met dezelfde priref in een database zitten (wat overigens zelden voorkomt), dan indexeert `-fix` het meest recente record daarvan. De oudere versie(s) van het record worden verplaatst naar een bestand genaamd *records.log*. Dit maakt het voor de databasebeheerder mogelijk om de verwijderde records te controleren en deze, als dat nodig is, opnieuw te importeren. (In *records.log* worden records in het Adlib tagged-formaat bewaard.)

-i:<jobnaam>	Opent de importjob en voert die uit.
-l	Deze optie wordt niet meer gebruikt.
-o	(Dit is de letter o, niet het cijfer nul.) Hiermee laat u alle te importeren records vastleggen in het logbestand dat moet zijn ingesteld voor de database waarin geïmporteerd wordt. Merk op dat normaal gesproken importeren niet wordt gelogd. Alleen voor *import.exe* kunt u logging dus expliciet inschakelen; in Designer kan dit nog niet.
-q:<databasenaam>	Met deze optie leegt u de database waarvan u de naam achter `-q:` opgeeft, voordat de rest van de importjob wordt uitgevoerd. Alle inhoud wordt verwijderd! Een databasenaam is gelijk aan de naam van het gerelateerde *.inf*-bestand maar dan zonder de extensie. Als het betreffende *.inf*-bestand zich in een andere map bevindt dan import.exe, dan moet u het pad naar dat bestand aan de databasenaam vooraf laten gaan. Voorbeeld: `import -q:..\data\collect`
-r:<recovery file>	Begin met het herstel middels het gespecificeerde logbestand. Zie de Designer Help en de *Installatiegids Museum, Bibliotheek en Archief*, voor informatie over logbestanden.
-s	Forceer automatische recordnummering; het eerste record zal met nummer 1 beginnen. Gebruik –s niet in combinatie met –b.
-u:<gebruikersnaam>	Hiermee kunt u als een andere gebruiker de importjob uitvoeren. Dat is handig als u bijvoorbeeld een importjob op een server wilt uitvoeren, en ingelogd bent als administrator; dan kunt u met bijvoorbeeld `-u:erik` (gebruik hier natuurlijk uw eigen naam) de import uitvoeren onder uw eigen naam. Of u gebruikt deze optie om de records te

	markeren als zijnde gecreëerd tijdens een importjob. Gebruik dan: `-u:import`. In de records wordt de *Invoernaam* nu *import* in plaats van uw eigen naam.
-v	Met de opdrachtregel-optie `-v` (vóór 6.1 was dit `-f`) geeft u aan dat tijdens het importeren *Use/Used for*-relaties moeten worden genegeerd. Dit is alleen van belang als u een thesaurusbestand importeert (of een ander bestand met zulke relaties), met de importjob-optie *Process links* gemarkeerd en in het doelbestand alle interne koppelingen op linkreferentie zijn. *Process links* vervangt namelijk niet alleen automatisch niet-voorkeurstermen door voorkeurstermen in externe koppelingen, maar ook in interne koppelingen op linkreferentie. Een niet-voorkeursvervanging is handig voor een import in een catalogus maar niet in een thesaurus, want daar worden beide soorten termen juist gedefinieerd. Als koppelingen dus wel verwerkt moeten worden, maar niet de *Use/Used for*-relaties, gebruik dan `-v` bij de aanroep van import.exe.
-x:<database> <index tag>	Herindexeer een specifieke index voor de database, bijvoorbeeld: `import -x:thesau te.` Het argument -x herindexeert de index: `<index-tag>` in de database: `<databasenaam>`. Als u ook het argument `-exclusive` toevoegt, wordt het herindexeren erg snel gedaan. Voer deze opdracht uiteraard in de *data*-submap van uw applicatie uit. Wanneer u een unieke index herindexeert, en er worden duplicaten gevonden, dan gaat het herindexeren gewoon door totdat de complete database is verwerkt; het voordeel hiervan is dat -x altijd een complete index creëert. Meldingen van duplicate termen worden in het .err-bestand geplaatst.

	In een Adlib SQL-database kunt u alleen reeds bestaande indexen herindexeren, terwijl u voor Adlib CBF-databases met deze optie ook automatisch nieuwe indexen kunt creëren.
De opties **-i** en **-r** sluiten elkaar overigens uit, en kunnen dus niet tezamen worden gebruikt.	

2 Export.exe

De Adlib-tool *export.exe* werd destijds geïntroduceerd om een snellere manier van exporteren te bieden dan het toenmalige DBSETUP. DBSETUP is vervangen door Adlib Designer, waarin exporteren even snel gaat als met export.exe. Welk programma u wilt gebruiken voor exporteren is dus kwestie van persoonlijke voorkeur.
Export.exe (6.0 en hoger) exporteert data uit DOS, ANSI of in UTF-8 gecodeerde Unicode databases naar uitwisselbestanden in UTF-8. Maar u moet de exportjob zelf in Designer definiëren. Beide programma's hebben dus een .exp-bestand nodig om uit te voeren. Start export.exe via de opdrachtregel in een "DOS"-venster. De syntaxis is als volgt:

```
EXPORT [-?][-e:<jobnaam>][-l][-s][-u:<gebruikersnaam>]
```

Alles tussen vierkante haken [ ] is optioneel, en alles tussen scherpe haken <> moet worden vervangen door een daadwerkelijke naam. En laat bij het invullen alle haakjes weg.

Het volgende:

```
EXPORT -e:..\data\MyExportJob
```

voert bijvoorbeeld alleen maar de exportjob genaamd `MyExportJob` in de directory `..\data`. uit.

U kunt de (optionele) opdrachtregel-argumenten in willekeurige volgorde achter `EXPORT` opsommen. Hieronder vindt u alle mogelijke opties beschreven:

-?	`EXPORT -?` toont Help vergelijkbaar met dit document.
-e:<jobnaam>	Opent de gegeven exportjob en voert die uit.
-l	Exporteer inclusief koppelingen.
-s	Forceer automatische recordnummering; het eerste record zal met nummer 1 beginnen.
-u:<gebruikersnaam>	Hiermee kunt u als een andere gebruiker de exportjob uitvoeren. Dat is handig als u bijvoorbeeld een exportjob op een server wilt uitvoeren, en ingelogd bent als administrator; dan kunt u met bijvoorbeeld `-u:erik` (gebruik hier natuurlijk uw eigen naam) de export uitvoeren onder uw eigen naam.

3 Adcopy

3.1 Introduction

The main purpose of the command-line Adlib Adcopy tool is to copy and convert an Adlib SQL database in which record data is still stored in binary format (pre-SQL Server 2005) to an Adlib SQL database in which record data is stored in XML format. The XML format is a requirement for the Adlib API in order to work with the thesaurus search operators, such as the *generic* search operator.
During this conversion the tool copies and changes the .inf files of your database (even pre-6.4 data structures), it adjusts the SQL data tables to reflect the current (6.6.0) Adlib SQL database structure and then copies the data tables into another Adlib SQL database where it also rebuilds all indexes. The contents of pointer files are also copied along in this conversion.

The tool can also be used to:

- migrate data from an Adlib Oracle database to an Adlib SQL Server database. (Adcopy can read both binary and XML data columns, but will always write the result as XML columns.)
- migrate data from one server to another. Normally you would use a backup and restore to move data from one server to another, but by using Adcopy you make sure that only tables which are actually in use will be migrated. Since the Adcopy program also rebuilds all indexes you are sure that after the copying process all indexes are up-to-date. Note that Adcopy is much slower at this than using a backup/restore process: Adcopy typically does about 100 catalogue records per minute.
- place data "in the cloud". Adcopy can be used to upload Adlib data to a Microsoft Azure SQL server in the cloud.
- create a stripped (public) version of your data. The Adcopy program can remove fields from the data whilst copying. This can be used to remove sensitive information from an SQL database before it leaves the premises, for instance to be hosted by a third party.

This manual currently only describes the SQL binary to XML conversion type, and the procedure to create a (stripped) copy of your database. Because of some overlap, the next chapter explains how to achieve either goal in a single procedure. Basically this procedure comes down to the following:

1. Adcopy needs a target data folder other than the one in your production environment to copy altered .inf files (database structure files) to. If you just want to convert binary data in your SQL database to XML data, you can create a temporary, differently named data folder for this purpose or you can copy your entire application (typically the Adlib software folder including its subfolders), and use that *\data* folder if you want to have a test environment ready for the converted data. Instead, you can also use an existing *\data* folder as the target folder, if you already have a target environment set up (like a web application or Adlib test application). See steps 1-4 in the next chapter for details.

2. Adcopy also needs a target SQL database to copy your source database to. This can be a temporary, new SQL database (which you'll have to create) if you are doing a binary-to-XML conversion or if you want to create a separate database for testing purposes before restoring it elsewhere later. However, the target database can be a pre-existing SQL database as well: this will overwrite any existing data in that database. The advantage of this last option is that everything is already in place to start working with the new data immediately; the disadvantage is that if the target environment is live, you skip the important testing phase of the new database. See step 5 in the next chapter for details.

3. Execute Adcopy to actually copy the database and possibly apply some changes to the copy, like the (automatic) binary-to-XML conversion and/or the removal of some sensitive data from it. See step 6 in the next chapter for details.

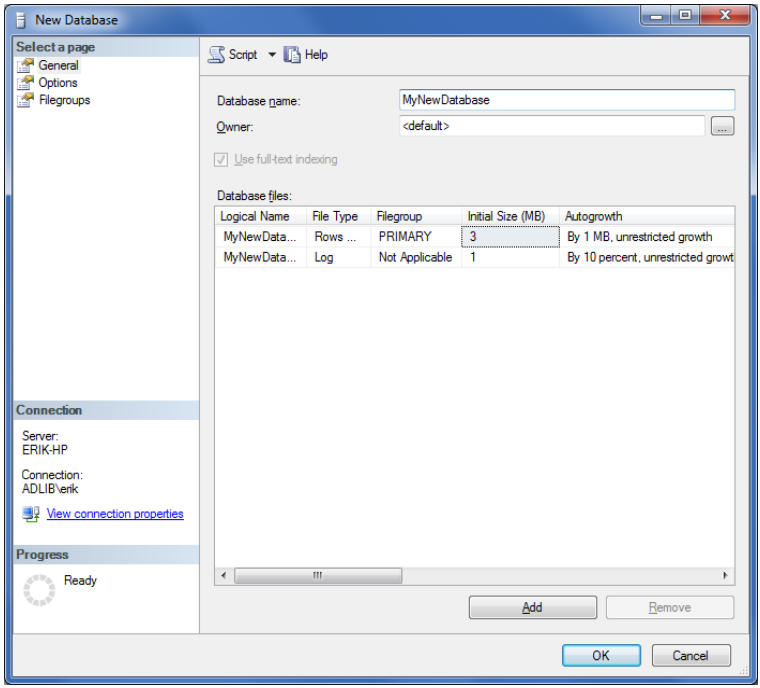
4. If you have copied your database to another pre-existing database, you are done now, except maybe for some testing. If, on the other hand, you have copied your database to a new SQL database, you'll probably need to do some renaming of files and folders to turn the copy into a production/live database; before that, you could stil test the copy if you didn't just create a new data folder but copied your entire application to create a test environment.
   See steps 7-13 in the next chapter for details.

Note that an Adlib database definition is not the same as a SQL database definition. An Adlib database or index file are both tables in a SQL database, for example. Also, there is no separate priref table, but in SQL the priref does get indexed. See the *SQL Server and Oracle* document (**Ctrl**+click to go to the download page on our website), for more information about how Adlib databases reside in a SQL or Oracle database. If you first want to know more about how Adlib databases

are structured in Adlib's proprietary CBF format, see the *Adlib software functionality profile* document, which can be downloaded as well.

3.2 Copy-converting your SQL database

1. For safety reasons, create a backup of your Adlib application and database. See the Installation guide for Museum, Library and Archive for information about making backups (and restoring them).
2. Copy the Adcopy files to a temporary folder on the local hard drive.
3. If you are about to convert binary data in your SQL database to XML data, or if you are about to copy your current SQL database to a new SQL database for other reasons, then in Windows Explorer look up the Adlib Software folder which has a *\data* subfolder. This folder contains your database structure files (.inf). Underneath the Adlib Software folder, next to the *\data* subfolder, create a new folder and name it e.g. *\data.new*.
   However, if you are about to insert a database copy into an existing database, then you don't need to create a new *\data* subfolder: there will be an existing target *\data* subfolder you can use.
4. If you created a new *\data* folder in step 3, then copy all files from *\data* into *\data.new*. (The *.inf* files in the new folder will be changed later on.)
   In all other cases you can skip this step.
5. From this step you can go two ways: (a) create a copy in a new SQL database, which is recommended for a binary-to-XML conversion, or (b) create a copy in another existing SQL database and overwrite any existing data in it.
   a. Open Microsoft SQL Server Management Studio (or a similar tool), and create a new database: right-click the *Database* node in the *Object Explorer* and in the pop-up menu choose *New database*… The name you enter for the *Database name* is not really important if you are doing the binary-to-XML conversion, because you'll change it later anyway. In the example below we named it *MyNewDatabase*. The credentials that Adcopy will use when you execute it next, need to have *datareader*, *datawriter* and *ddladmin* access rights to this database. Make any other desired settings before clicking *OK*.



b. Open Microsoft SQL Server Management Studio (or a similar tool), and open the existing target database into which you want to create a copy. Make sure that the credentials that Adcopy will use when you execute it next, have *datareader*, *datawriter* and *ddladmin* access rights to this database.
Any existing data in this target database will be overwritten by Adcopy! If the target database is operational, this also means that your new copy will go live without having tested it. This always holds some risk, so if you take this path, make sure you have a backup of the target database ready to restore if the copy you create with Adcopy doesn't turn out the way you wanted it. Also be aware that while Adcopy is overwriting your live target database, it shouldn't be accessible to users at all, to prevent errors of all sorts: it's no use to bring the SQL database offline, because then Adcopy won't be able to write to it anymore either, so you would have to ask co-workers to stop working in the target database and/or present a temporary page on your website, for example.

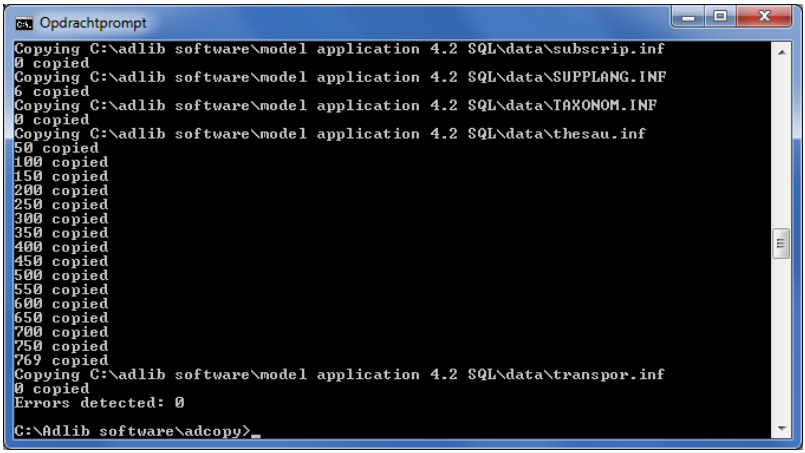
6. Adcopy can be controlled by command-line parameters, or by a parameter file in the same folder as the executable file, with the name *adcopy.xml*. If command-line parameters are provided then these will be used, otherwise the program will try to open the *adcopy.xml* parameter file.
   To provide command-line parameters, open a command line window and execute Adcopy using the following syntax:

   ```
   <(path to) adcopy> <path to the old data folder> <path to the
   new/target data folder> <the new/target database name> [the
   target SQL Server name] [user id for new/target database]
   [password for new/target database]
   ```

   Everything between [] is optional. If the new or existing target *data* folder is not empty, any existing .inf files in there will be overwritten if copied files have the same name. If you are copying your database to another existing database, you can use the existing *data* folder of the latter as the target data folder. The destination SQL Server name is optional: by default it is the (local) server, so only if your new database is destined for a different SQL Server, you will have to provide it explicitly. The name of the destination server will be stored in the copied .inf files. The user id is the user name with which Adcopy connects to the SQL server; supplying the user id in this way is only possible if the SQL server uses the SQL server authentication mechanism. This user name will also be stored in the copied .inf files. If a user id is provided then a password is also required, which will be stored in the copied .inf files as well. Further, the user id and password will be copied from the original .inf files if no new user id and password are provided. Example:

   ```
   adcopy "C:\adlib software\model application 4.2 SQL\data"
   "C:\adlib software\model application 4.2 SQL\data.new"
   MyNewDatabase SQLserver2008\Adlib
   ```

   This will take a while to complete. When it is finished, any errors will be reported in the command prompt window.



As mentioned, when no command line parameters are supplied, Adcopy will try to find and read *adcopy.xml.* This file provides the same parameters as the command line, but it offers a few additional options. Here is an example of the content of the *adcopy.xml* parameter file:

```
<?xml version="1.0" encoding="utf-8"?>
<AdCopyJob xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SourceFolder>C:\adlib4.2-SQLserver\data</SourceFolder>
  <!-- folder to copy data from -->
  <DestinationFolder>C:\adlib4.2-SQLserver\data2</DestinationFolder>
  <!-- folder to copy data to -->
  <DestinationDatabase>model42t</DestinationDatabase>
  <!-- name of the destination database -->
  <DestinationServer>(local)</DestinationServer>
  <!-- name of the destination server -->
  <UserId></UserId>
  <!-- userid to be used and placed in the destination .inf file -->
  <Password></Password>
  <!-- password to be used and placed in the destination inf file-->
  <MileStone>50</MileStone>
  <!-- milestone value -->

  <!-- a list of databases with fields that need to be removed -->
  <DatabaseList>
    <Database Name="collect">
      <!--entry for each database of which fields must be removed-->
      <Exclude>
        <!-- list of Adlib tags that need to be removed -->
        <string>ls</string>
        <!-- one string entry per tag that needs to be removed -->
        <string>VY</string>
      </Exclude>
```

```
    </Database>
  </DatabaseList>
</AdCopyJob>
```

The additional options in the parameter file are:

- the milestone value, to indicate progress during the conversion;
- an optional list of one or more Adlib databases, each containing a list of tags which need to be stripped from the copy. Of linked fields to strip, you only need to provide the link reference tag. This functionality operates on the unresolved data as it is stored in the record, which means without data in the linked field itself or in any merged-in fields. So, removing the link reference tag from the database copy will leave out the link reference and the relevant linked data including merged-in data. To strip tags from more than one database you must repeat the `<Database>` element with another name and accompanying list of tags.
  This functionality is handy when you want to create a copy of your database without any sensitive information in it, to be applied as the database for your public web application for instance, or for use by third parties. Typically you would not use this option if you just want to convert binary data in your SQL database to XML data.

7. If you have copied your database to another existing database, you are done now. Just check the target environment to see if everyting is as it should be. Skip steps 8 and further.
   If you haven't been copying your database to another pre-existing database and if you created a new subfolder for the .inf files in step 3, then rename the old *\data* folder in Windows Explorer, for instance to *\data.old*, and subsequently rename the new subfolder (e.g. *\data.new*) to *\data*. The changed .inf files now reside in your current *\data* folder.

8. In Microsoft SQL Server Management Studio, rename the old database (let's say this was called *AdlibDB*), for instance to *MyOldDatabase*. Also rename the new database (in our example *MyNewDatabase*) to the original name of the *MyOldDatabase* database: *AdlibDB* in this example. So the old *AdlibDB* has now been exchanged for the new *AdlibDB* which was created by Adcopy.
   (To rename a database, right-click it and choose *Rename* in the pop-up menu. You can only rename a database if it is not in use.)

9. If the old database, *AdlibDB* in our example, had specific access rights settings, then apply the same settings to the new database now.
10. Start Adlib Designer, open your Adlib application in the *Application browser* and select one of the Adlib database structure files (.inf). On the *Database properties* tab, change the adjusted *Data Source Name* (*MyNewDatabase* in our example) to the original name (like *AdlibDB* in this example) and leave the field. Adlib Designer will ask you if you want to apply the change to all databases: click *OK*. Then click the *Test* button to the right of this option, to test the connection. Only if the test succeeded, click the *Save all modified files* button and save all changed .inf files.

11. Start your Adlib application and check if all is well.
12. If you tested your new database succesfully, you may delete the *data.old* folder from your system; make sure you do not remove the *data* folder!
13. In Microsoft SQL Server Management Studio you can also delete the old database by right-clicking it (*MyOldDatabase* in our example) and choosing *Delete* in the pop-up menu. You can only delete a database if it is not in use.)

By the way: Adcopy is backwards compatible, so an older adlwin.exe can be used to open the database again after being processed by Adcopy.

4 Adsearch.exe

Adsearch.exe is een Adlib opdrachtregelprogramma waarmee u vanuit een DOS-venster zoekopdrachten in een Adlib CBF-database kunt uitvoeren (geen support voor SQL: gebruik dan een API) waarna het resultaat in een pointerfile wordt opgeslagen. Die pointerfile kunt u binnen Adlib op de normale manier gebruiken. De syntaxis is als volgt:

```
[directory\]adsearch -l <[directory+]databasenaam> -p <pointerfile-
nummer> [-t "<pointerfiletitel>"] <[directory\]bestandsnaam>
```

De mogelijke opties zijn:

-h	`adsearch -h` toont Help vergelijkbaar met dit document.
-l <[directory+]data-basenaam>	(Dit is de kleine letter L.) Vul na een spatie de naam van de te doorzoeken Adlib-database in, eventueel voorafgegaan door het pad ernaartoe gevolgd door een plusteken. Deze optie is verplicht.
-p <pointerfilenummer>	Geef na `-p` en een spatie het nummer op van de pointerfile waarnaar het resultaat moet worden weggeschreven. Als u `-p 0` opgeeft, dan kent adsearch zelf een pointerfilenummer toe dat nog niet in gebruik is. Deze optie is verplicht.
-t "<pointerfiletitel>"	Geef na `-t` de naam van de doelpointerfile op, omsloten door dubbele aanhalingstekens.
<bestandsnaam>	De zoekvraag moet u met de syntaxis van de Adlib selectietaal in een teksttbestand opstellen en opslaan met de extensie *.src*. Gebruik tags of Engelstalige veldnamen. Vanaf versie 6.4 kunt u dit tekstbestand in UTF-8 codering opslaan, maar in oudere versies moet u ANSI-codering gebruiken. Aan het eind van de adsearch-opdracht geeft u de naam van dit bestand op, zonder extensie. Omsluit de naam door dubbele aanhalingstekens als er spaties in voorkomen. Deze optie is verplicht.

Als u in het DOS-venster in de directory bent waarin *adsearch.exe* staat, bijvoorbeeld de directory *\executables* in uw Adlib-directory, dan kunt u bijvoorbeeld de volgende opdracht geven:

```
adsearch -l ..\data+Collect -p 1 -t "Object is ansichtkaart" ob-
jectnaamzoekvraag
```

U moet wel al het bestand *objectnaamzoekvraag.src* (in de huidige map) hebben aangemaakt. Dit bevat bijvoorbeeld de volgende zoekvraag:

```
object_name = ansichtkaart
```