Database and Application Migration

DBMS Migration for ADABAS/Natural

ADABAS to RDBMS
Natural to Java, C#, or COBOL

White Paper
## Contents

- Executive Summary ................................................................. 1
- Global Assessment ................................................................. 4
- Database Model Mapping ......................................................... 6
- Code and JCL Migration ........................................................... 8
  - COBOL Migration ............................................................... 8
  - Natural Migration ............................................................... 9
  - JCL Migration ................................................................. 10
- Data Migration ........................................................................ 11
- Testing ..................................................................................... 12
- Code Refresh .......................................................................... 13
- Conclusion ............................................................................. 14
- About BluePhoenix ............................................................... 15
Executive Summary

This document describes how you can use BluePhoenix DBMSMigrator for ADABAS/Natural as a means to migrate your total ADABAS/Natural environment including migrating your data and JCL. The purpose of this document is to provide an overview of how BluePhoenix Solutions migration technology, methodology, and migration experts can help you migrate your Natural and COBOL applications and environments away from ADABAS/Natural to more mainstream environments such as Java, C#, and/or COBOL, and RDBMSs.

Manual migration and consolidation of applications and databases from legacy computing environments to modern platforms and technologies often present an IT organization with the challenge of complex and costly projects. Such projects must confront common risks such as inconsistent results, introduction of new errors, system and application availability, and unreliable software functionality.

BluePhoenix migration projects follow a strict and proven methodology that is designed to migrate successfully a source application from ADABAS to any relational database while assuring high quality and timely results. This methodology can be carried out either by BluePhoenix personnel or by certified BluePhoenix system integrators.

![Diagram of ADABAS/Natural Migration Overview]

*Figure 1. ADABAS/Natural Migration Overview*
1. Global Assessment and Planning

Using BluePhoenix IT Discovery, an automated inventory and analysis process ensures that all components needed for the migration are available and identifies all external interfaces to ADABAS/Natural applications, 3rd party products used, data interfaces, and interfaces to external functionality such as security. This phase is performed at the customer site in conjunction with the customer either by BluePhoenix or a BluePhoenix certified systems integrator.

2. Detailed Analysis and Database Model Mapping

This phase includes determining the mapping between ADABAS and/or VSAM (for VSAM/Natural and for ADABAS/VSAM/COBOL) to the desired relational database. Since both of these environments support concepts such as repeating fields and do not contain types such as date and time, a mapping from the legacy data structure to the new table structure and relational model is required including how descriptors, super-descriptors, and sub-descriptors are to be handled. A number of options are provided for handling normalization to balance relational theory with performance.

This phase is done in conjunction with the customer and can be done either by BluePhoenix or by a BluePhoenix certified systems integrator.

3. Code Migration

For direct calls and ADAPREP calls—based on the data-model mapping rules in the DBMSMigrator repository and the information that was added—DBMSMigrator creates I/O “objects” for use by any COBOL program that previously accessed ADABAS. This is done using optimized SQL code, in a manner that does not require the modification of the current programs. For ADASQL calls, the ADASQL is replaced directly by in-line SQL.

Using the same data-model mapping information, Natural programs are migrated to either Java/J2EE or C#/ .NET using the same I/O layer.

4. Platform Migration and JCL Support

When remaining on the mainframe, all JCL can be upgraded to replace the various ADABAS and Natural utility calls with the associated RDBMS utilities.

If a change of platforms is involved, JCL can be migrated to Script on Unix or Windows. Mainframe utilities such as SORT and FTP are also supported and are replaced by equivalent utilities on the target platform.
5. Data-Migration Program Generation

The automated generation of data-migration programs enables smooth data transfer from ADABAS and/or VSAM to the new environment. These programs are used to extract the data from the ADABAS database or VSAM files into sequential files, transform the ADABAS/VSAM format files to files that resemble the new tables, and prepare the final files for the appropriate load utility. This generation of the data-migration programs is done at the BluePhoenix factory, while the actual migration is performed at the customer site.

6. Testing

Testing for DBMSMigrator is made easy by the high degree of automation involved. Testing is carried out in two phases: migration testing and acceptance testing. Migration testing is carried out at the BluePhoenix factory using a small amount of test data provided by the customer to verify that the migration was carried out successfully. The second phase of testing is acceptance testing, which verifies that the application is functioning correctly in its new production environment and is typically carried out at the customer site by either BluePhoenix, a systems integrator, or the customer.

7. Code Refresh

During the migration project, daily maintenance and enhancements to the original ADABAS/Natural application can continue. Only a short “freeze” period is required immediately before moving the new application into production. During this “freeze” period, all changes made to the original applications are incorporated into the newly migrated application by means of a code “refresh.”
Global Assessment

The Global Assessment is performed using IT Discovery, which provides a complete assessment of the usage of a customer ADABAS/Natural environment as a precursor to carrying out the migration itself.

![IT Discovery Global Assessment](image)

IT Discovery is a complete Application Portfolio Management System that supports cross-technology analysis of both the static structure and operational characteristics of an IT environment. IT Discovery is an open product that can analyze many languages, databases, TP monitors, and scripting environments; however, when used in conjunction with an ADABAS/Natural environment migration, the analysis is focused more on those items that relate to applications developed in either Natural or COBOL using ADABAS or other file systems accessible by Natural.

In performing this analysis, the following information is extracted:

**From Natural:**

- All Natural libraries, programs, subprograms, subroutines, maps, copy code, help routines, and how they interrelate.
- All local, global, and parameter data areas as well as “+” style global variables.
- All DDMs including all fields.
- All use of other products such as Natural Connection, Natural Process, Natural Construct, etc.
- All access to 3rd party, outside systems via calls, either another language or operating system.
From ADABAS:

- All databases, files, and associated fields.
- All usage of password security.

If VSAM/Natural is used:

- VSAM file and field information from the Natural DDM.

If DB2/Natural is used:

- DB2 table and column information from the Natural DDM.

From Natural Security:

- All usages of Natural Security.

From PREDICT:

- All uses of PREDICT names for the purpose of use within ADAPREP or ADASQL.

IT Discovery exception reports are analyzed and resolved. Missing components must be found and added; duplicate entities should be removed.

Upon completion of this step, a high-level migration plan is prepared and libraries are “clustered.” A migration cluster is a logical subset of a large client system. A cluster includes one or more libraries or program groups that share programs, data, and related components, while requiring relatively little interaction with other migration clusters. These clusters may be efficiently migrated, tested, and implemented as a single unit. Working with clusters reduces the complexity of migrating large, complicated systems, and allows work to be done on multiple clusters in parallel.
In order to migrate both the code and data involved in an ADABAS/Natural migration, it is necessary to create a mapping between the current model using ADABAS and/or VSAM into the new environment based on DB2, Oracle, or SQL Server.

Database model mapping comprises two basic steps:

1. **Mapping**: Map the source data model to the destination relational database.

2. **Relational Data Model Generation**: Generate the DDL for UDB, Oracle, or SQL Server.

1. **Mapping**

The mapping is carried out via the DBMSMigrator Model Mapping Workbench which provides numerous mapping options.

Figure 3. DBMSMigrator Model Mapping Workbench
The following are examples of the mapping options:

- **Split Files**: An ADABAS or VSAM file can be split into more than one table. For VSAM, multiple views of the same data file can also be split based on a field data value.

- **Fields**: Field names and formats are changed to conform to the RDBMS standards.

- **ADABAS Periodic Groups, Multi-valued Fields, and VSAM Occurs fields**: These repeating fields may be either normalized to a new table, or flattened into the same table. If a new table is created, a primary key based on the primary key of the owner table plus an index column is also created.

- **Descriptors, Sub-descriptors, and Super-descriptors**: These are transformed into indices. Any necessary extra fields and triggers needed to emulate the behavior of ADABAS null fields with super-descriptors are automatically created.

- **Use of ISN**: ISN can be maintained if it is used as part of the application or dropped.

### 2. Relational Data Model Generation

With a single mouse click, DBMSMigrator generates:

- A relational database based on the mapping

- The corresponding DDL file

- Data migration programs to unload the data from ADABAS and load it into the new RDBMS
Code and JCL Migration

The DBMSMigrator tools migrate source COBOL that uses ADABAS to COBOL that uses an RDBMS, and translates source Natural to Java, C#, or COBOL that uses either a new or the same RDBMS.

COBOL Migration

Overview

A migrated COBOL application supports an RDBMS instead of ADABAS. A BluePhoenix migration provides a framework that emulates ADABAS behavior.

Whenever a COBOL program interfaces with ADABAS it uses the "CALL ADABAS" command. After migration, this command is redirected to call a new I/O routine layer that emulates the ADABAS behavior. Upon receiving the ADABAS buffers, the I/O layer performs the following:

1. Generates SQL commands based on the buffers content
2. Performs the SQL commands to the RDBMS
3. Receives the results from the RDBMS
4. Transforms the results to ADABAS buffers
5. Returns the results to the COBOL program

The COBOL program remains the same and is not modified by the migration process. The access to the I/O routines instead of ADABAS is transparent to the COBOL program.

I/O Routine Layer

The I/O layer is responsible for emulating the ADABAS behavior with an RDBMS. It handles analyzing the ADABAS buffers and transforming them to dynamic SQL calls.

The I/O layer is written in Java or C#, and includes a class for each DDM that encapsulates the physical to logical behavior of that DDM.

This layer also uses a mapping table to map between ADABAS file numbers and the corresponding DDM classes, and to map between ADABAS short name fields and the DDM long field names.

The I/O layer has a new "CALL ADABAS" routine that replaces the original ADABAS routine.

RDBMS response codes are mapped to ADABAS errors; therefore, the COBOL program can continue working with the same error handling code.
Natural Migration

Overview

Natural programs can be transformed into Java, C#, or COBOL. The migrated code is based on a special framework that emulates the behavior of the various Natural statements and data types. Since most of the Natural statements and data types do not exist in the Java/C# languages, and since there is no equivalent statement in Java/C# to replace them, a special library is used that supplies exactly the same functionality as the Natural statements.

This framework is an object-oriented library that is inherited by all the migrated Natural programs. It includes numerous support classes that handle various aspects of the Natural language including memory management, database management, Natural variable types, Natural system variables, sorting, printing, etc.

Mapping Natural Programming Objects to Java Objects

- Each Natural program, subprogram, subroutine, local data area, global data area and map containing processing rules becomes a single Java/C# source file (class).

- Internal subroutines become methods in the generated program.

- The Natural library containing the application becomes a single Java/C# class. The Global Data Area (GDA) becomes a class within the Natural library class.

- Each Natural INPUT statement in the application—whether in a MAP or in-line—corresponds to a single JSP (Java Server Page) or ASPX.

- Each Natural DDM in the application generates a single Java/C# source program. The Java/C# code encapsulates the logical-to-physical mapping and is the single point for changes in the way data is physically stored.
All JCL members, PROCs, and control statements that are identified as being related to the target application and that need to be changed, are modified to operate with the new subsystem(s).

If OS/390 JCL will continue to be used—either because the migrated batch application will run on a mainframe or because environments such as Micro Focus MTO, Fujitsu zBATCH, or Sun MTP are being used to execute JCL on non-mainframe platforms—global changes are made to replace the ADABAS batch related information with the new information. Based on specified defaults, ADABAS file DD statements are deleted, a TMP program name with the required parameters is inserted into the EXEC statements, a new DD statement referring to the RDBMS plan and program names is inserted, and more. Also, changes to job statements and other JCL information (file names, print class, etc.) can be made at this time.

If the destination platform is Unix or Windows, the JCL can also be converted to KShell or VB.NET scripts.
The automated generation of data-migration programs by DBMSMigrator enables smooth data transfer from the ADABAS environment to the relational environment. This program generation process is based on source data models and the RDBMS data definitions and mapping rules that were created during data model mapping. This process also allows the migration programmer to add rules for data cleansing and modification that go beyond those defined by the mapping rules.

These programs extract the data from the ADABAS database into sequential files, and transform the ADABAS or VSAM format data to the destination relational table in accordance with the model mapping rules.

At the end of this process, load files are produced depending on the destination RDBMS.

A standard load utility is used to read the files created during the transformation phase and load the data into the relational database.
Testing

An important aspect of any migration is testing. No matter how automated the migration process, every application is different and testing is still required.

However, due to the nature of automated migration, testing can be carried out in a fashion that requires less work than would be necessary if an application were undergoing a hand-coded new release.

The first phase of testing is migration testing:

This testing is carried out to insure that the migration process is functioning correctly. The testing still involves executing test cases; however, the test cases are focused on testing differently functional areas of the application as they relate to the nature of the source Natural or COBOL programs rather than to the functional nature of the application itself.

Due to the nature of the automated migration, functional concepts in the source environment (e.g., Natural or COBOL) will always be migrated the same way, so migration testing is more focused on the nature of the migration rather than the nature of the application. Migration testing is typically carried out at the BluePhoenix factory, since any errors discovered during this process will typically require fixes to DBMSMigrator rather than to the application itself.

The second phase of testing is acceptance testing:

This testing verifies that the application is functioning correctly and as before. Since the goal of automated migration is to not change the application functionality at all, any test scripts against the source applications that are already in existence or that were developed while migration was underway, can be used again to test the migrated application. Errors revealed during acceptance testing may or may not be fixed in the tools depending on the nature of the problem and when the problem occurs during the test cycle.
Code Refresh

When migrating Natural to Java, C#, or COBOL, the destination is modified and therefore it is necessary to deal with the issue of what to do with ongoing maintenance changes made during the migration process.

Although migration is typically not a long task, it still takes time largely due to testing. In order to aid carrying application changes forward, DBMSMigrator provides a “refresh” option whereby a new copy of the source code can be taken to enable application changes that have been made since the last source code snapshot was taken. Because of this refresh capability, maintenance and development work can continue on the original Natural programs in parallel with the migration activities. The refresh capability does allow for changes in both the source data models and source programs; but of course it goes without saying that the fewer changes the better in order to minimize any impact on previous testing.

Refresh typically takes place:

- After migration testing and before acceptance testing. Since the goal of migration testing is really testing the migration process and not the application, once any problems associated with the migration have been resolved and any custom migration requests have been implemented, it is a good time to take a “fresh” source copy.

- After acceptance testing. In order to incorporate applications, a fresh copy of the source can be taken and run through again. At this stage, a comparison of the code generated is compared to the code generated the previous time in order to determine which new programs have changed. Based on this information, only needed changes can be accepted thus minimizing any retesting impact. Of course, any ongoing application changes can also be reapplied to the new code manually if this is seen to be less effort.
Conclusion

Migrating away from ADABAS and Natural is not a simple task. Organizations have typically used these products to write mission-critical applications that are core to their business. Yet in today’s world, these applications are no longer mainstream, instead requiring special skills, hard to find skill sets, and proprietary environments.

Using DBMSMigrator for ADABAS/Natural, an organization can migrate away from ADABAS and Natural to more mainstream environments such as Java, C#, and/or COBOL that access an RDBMS. DBMSMigrator is highly tuned to deal with all of the issues involved in migrating applications written using ADABAS and Natural by providing not only the ability to migrate but the ability to consider issues such as performance in the process.

Whether you wish BluePhoenix to handle the complete migration for you or you choose to work through a BluePhoenix certified systems integrator, BluePhoenix experience and technology solutions greatly lower the risk involved and make the migration a smooth and easy process.
About BluePhoenix

Reviving Yesterday’s Systems for Tomorrow’s Business

BluePhoenix Solutions (NASDAQ: BPHX) is a leading provider of value-driven legacy modernization solutions that enable companies to leverage their existing IT assets for increased business value and improved competitive advantage.

BluePhoenix offers mainframe (System z) and midrange (System i) customers a comprehensive suite of tools, services, and domain expertise. These offerings range from global IT assessment and impact analysis to automated database migration (ADABAS, IDMS, IMS, VSAM, etc.), application migration (COBOL, Natural, PowerBuilder, COOL:Gen, etc.), application remediation, platform rehosting, and architecture renewal.

BluePhoenix provides CIOs and senior management with assistance in coming to terms with the broad range of complex modernization issues that they face. We are particularly well suited to advise CIOs of best practice modernization strategies that provide rapid ROI, improved productivity and efficiency, while managing operational risk. BluePhoenix helps customers define their modernization strategy, which aligns business with technology investment decision making and architecture standards. The results are improved business agility and efficiency and reduced total cost of ownership (TCO).

We deliver confidence—the confidence that your modernization project will succeed without fail, on time and on budget, with full data integrity and application logic retention, minimal downtime, and maximum end-user transparency.

Tools and Services Summary

- Global assessment and modernization roadmap
- Automated application and language migration
- Automated database migration and consolidation
- Architecture renewal and platform rehosting
- SOA enablement
- Application remediation
- Outsourced offshore development
- Compliance solutions
Background and Statistics

BluePhoenix was established in 1987 and went public on NASDAQ in 1997. For over 20 years, BluePhoenix has successfully leveraged its best-practice expertise in small, midsize, and large-scale modernization projects. The company works closely with its customers to determine which of their assets should be migrated, redeveloped, or wrapped for reuse as services (SOA) or business processes. This approach both protects and increases the value of business applications and legacy systems, with minimized risk and downtime.

The company has over 750 highly experienced professionals, 15 offices around the world, and 7 full-service onshore and offshore delivery centers, including locations in the USA, UK, Denmark, Germany, Italy, France, The Netherlands, Romania, Russia, Cyprus, South Korea, Australia, and Israel.

Customers

BluePhoenix provides modernization solutions to companies from diverse industries and vertical markets such as automotive, banking and financial services, insurance, manufacturing, and retail.

Our list of prestigious customers includes companies such as Aflac, Aegon, Bell Canada, BMW, Charles Schwab, Citigroup, DaimlerChrysler, Danish Commerce and Companies Agency, Desjardin, Deutsche Bank, Ford, Fortis Bank, LA County Employees Retirement Association, Lawson Products, LBS, Merrill Lynch, Pratt & Whitney, Rabobank, Rural Servicios Informáticos, Safeway, SDC Udvikling, and TEMENOS.

Partners

In a world of converging technologies and services, BluePhoenix is committed to establishing strong partnerships with product vendors, service providers, system integrators, and consultants. We develop long-term business alliances with our Solution and Technology Partners and fully integrate our tools and products to ensure best of breed solutions that are cost-effective and easy to deploy.

Our partner base includes leading organizations such as IBM, Oracle, Microsoft, LogicaCMG, Capgemini, EDS, DMR (Fujitsu), HP, Micro Focus, and Sun.
BluePhoenix Offices

USA
usa@bphx.com

Australia
australia@bphx.com

Cyprus
cysec@bphx.com

Denmark
nordic@bphx.com

France
france@bphx.com

Germany
germany@bphx.com

Israel
israel@bphx.com

Italy
italy@bphx.com

Romania
romania@bphx.com

Russia
russia@bphx.com

South Korea
south.korea@bphx.com

Spain
spain@bphx.com

The Netherlands
netherlands@bphx.com

United Kingdom
uk@bphx.com

www.bphx.com