IBM Redbook Solution Guide Insert

Database Modernization – The key to long-term value
Adsero Optima™ (AO) Foundation

Your database structures – the key to your future

As discussed earlier in the AO Roadmap solution guide, a fundamental assertion of AO is that most of the value in a heritage application is encapsulated in the use of the underlying database structures and components, the coded relationship constructs and the validation rules that are sprinkled throughout the application logic. Because of this, the first step in a long-term modernization approach is to achieve the immediate migration to DDL, and to then use this platform as the foundation for ALL other modernization going forward.

AO Foundation aids in the migration to DDL. The migration is completely transparent to the application (even if the source code is not available). The application then uses the new DDL defined database as a foundation to gradually modernize the heritage applications. The result will be a significant improvement to the application. This improves the fundamental database design as it allows the implementation of proper relational structures, referential constraints and database-enforced validations. Entity relationships are gradually moved out of the code and into the DB2 database engine. All validation rules are also gradually moved into the new DDL defined database.

Based on significant achievements and experience in this area, AO Foundation uses the following high-level database modernization process:

1. In phase 1 AOF converts DDS to native DDL defined constructs **without changing ANY LVLID**'s on the new database definitions. This ensures absolute transparency to the application logic. AOF allows the conversion of a single database file, multiple selected files, one library, multiple selected libraries or an entire system at a time to DDL. ***(It is important to recognize that in order to maintain LVLID's, NONE of the logical files will be converted in this phase. On completion, all constructs will be completely new, and with significantly improved performance characteristics.)***
2. AOF supports both native and surrogate migrations, with the preference for a native migration. The reason being is that the objective is to gradually use the newly defined DDL constructs as foundation to move entity relationships and data validations into the new DDL defined database. Surrogates make this (extraction of relationships and validations from code, moving it into DB2) problematic in our experience. Surrogates are used, in general, only where technical requirements indicate their use.
3. We then suggest that a three way parallel process be followed:
    a. Extract all metadata from the old database into a new consolidated data dictionary and start a process of removing all duplicates and inconsistencies in the metadata. This (duplicates and inconsistencies) is a factor of the age of our systems (most applications have been in existence for upward of two decades) and the maintenance processes used. It is not a reflection on developer resources. The result is a significantly improved and consistent use of metadata attributes throughout the system, with considerable integrity improvements.

   b. Use AOF to identify unique keys in the database constructs and to roll this back into the new DDL defined tables. This will allow the implementation of a TRUE relational database model, which is necessary in order to implement constraints. In heritage applications and historic coding practices, unique keys were normally enforced using logical files (In early S/38 days keys on physical files was against recommendations), or through application logic. It is important to note that AOF assists in driving this into DB2, with no change to LVLID's!
   c. Dependent on coding practices and installation standards, it is recommended that field and file descriptions are used as initial SQL "aliases" for long file (table) and field (column) descriptions. This exposes more sensible, descriptive table and column names to end-users. This step alone will improve the "legacy" perception of your application. The database will present itself as being modern to executives and end-users with more descriptive names compared to the usual 6 – 10 character abbreviated "techno-speak" commonly used. AOF can automate this process completely and transparently, with no impact to LVLID's. Alternatively, manual editing of the descriptions can be enabled, to include descriptions that are more meaningful as part of a metadata enrichment process, which AOF facilitates.
4. Some of the above actions may have an impact on LVLID's, but AO provides a comprehensive repository function with extensive cross-reference capabilities, down to metadata element level, to enable full management and implementation capability.
5. Once AOF is used to provide a true relational database model and having removed all duplication and conflicts, the new database model is then used as a foundation to start driving additional inferred relationships and rules into the database. This includes extracting validation rules from the application logic and implementing those as triggers. In addition, the introduction of new IO Servers (which deliver results set to high level application functions, gradually phasing out record level access in favor of result sets produced for consumption system wide), and implementing system wide "Enterprise Services".

It should be recognized that the entire focus and philosophy of the AO Roadmap, technology and product portfolio is designed to facilitate a modernization process that is:

- ✓ Low risk
- ✓ Gradual
- ✓ Non-disruptive
- ✓ Transparent
- ✓ Iterative

And which assists software developers in modernizing their systems from the inside out. Most of this can be achieved during normal routine maintenance, and after (or during) the initial database modernization process is completed or performed.

In our experience, most "big bang" modernization projects are doomed to failure.

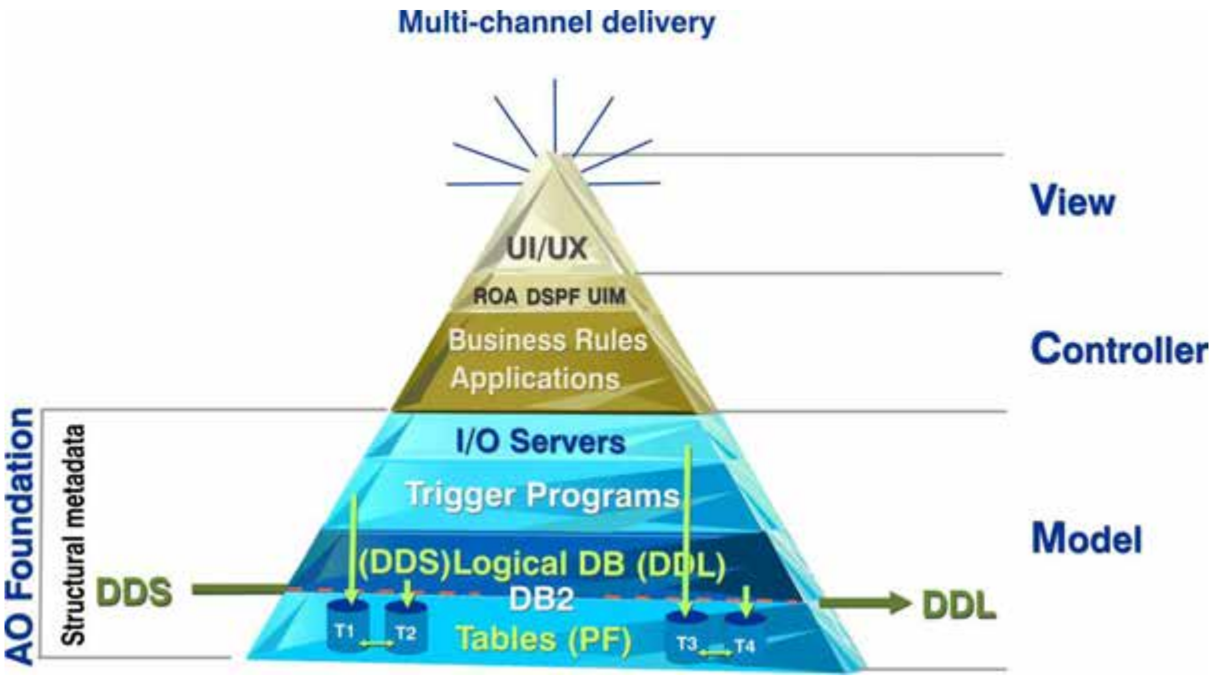
The AO roadmap and products do not force any specific sequence, allowing the choice of best practices and to achieve rapid results for any business.

Kindly review the following animated video, which describes the process succinctly:
http://www.adsero-optima.com/nutshell-msp

Modern Application Architecture

It is an essential consideration in understanding the AO approach, to acknowledge that IBM i applications have been “broken” from an application architecture perspective, ever since the introduction of the ILE programming model and the massive advances in the database engine. This is the reason why any long-term modernization has to start at the database engine layer.

As part of any modernization project, we need to implement modern multi-tier application architecture, with separation between database, the UI and business unique logic. Most programmers refer to this as the MVC model, which depicts Model-View-Controller, as per the graphic below:

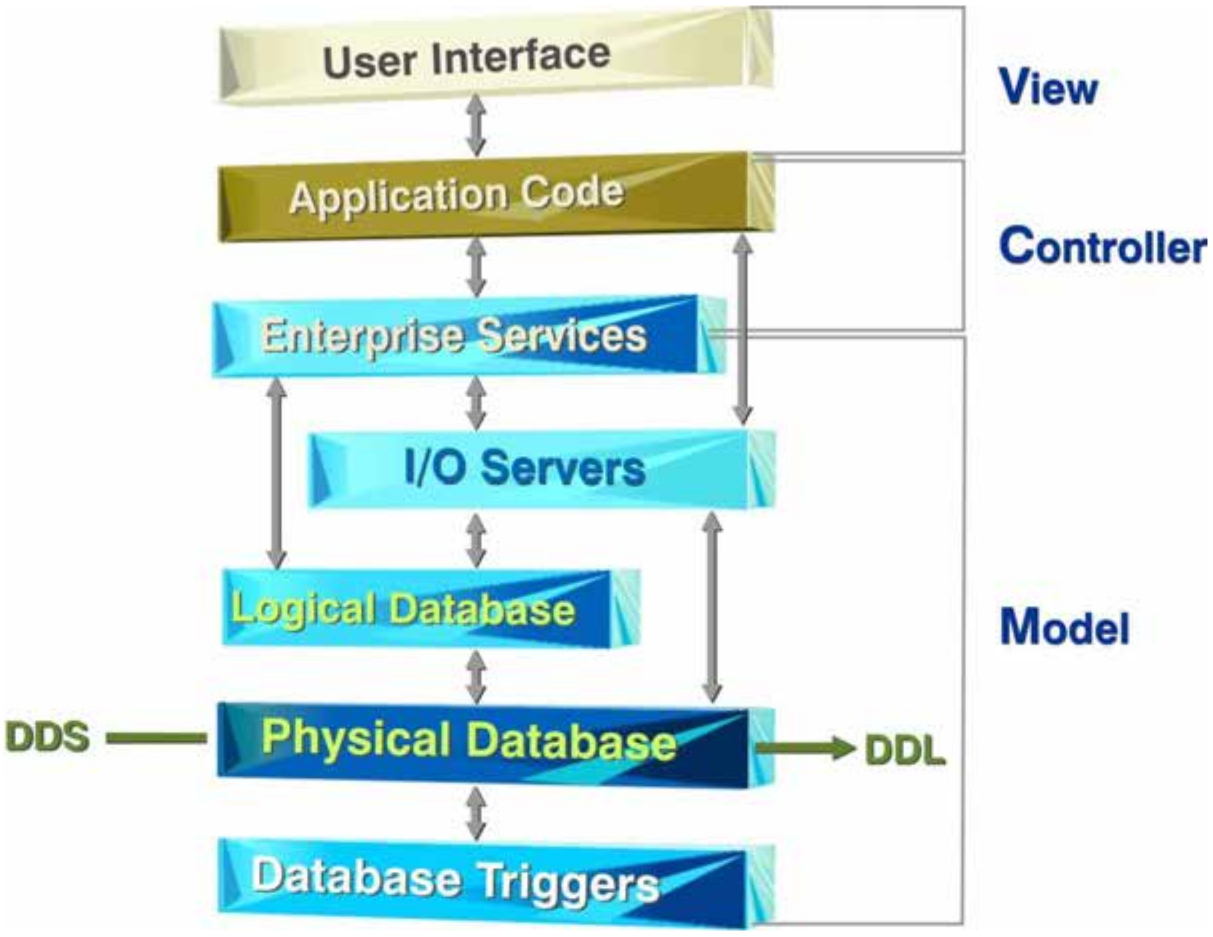


As most of our heritage applications remain large monolithic constructs, with database, application logic and UI (display file) interspersed (Spaghetti ala RPG), maintenance has become excessively burdensome and error prone over the years. It is this fact that causes most agility constraints, as changes quite often “break” other functions elsewhere in the application.

These monoliths have to be re-engineered in such a way, as to facilitate the separation of the database, UI and business logic, in a low risk, gradual, non-disruptive process, to remove this fundamental constraint. **<u>The database is the absolute key in achieving this.</u>**

The objective of any modernization project is to eventually deliver a modern application architecture, which will provide clear separation of function and recovery of the fundamental business value in your heritage application and especially the competitive advantage that is currently hidden in your application code. Based on remarkable results and proof, we have demonstrated that this is an achievable objective.

A truly modern IBM i based application should have the following architectural components/structures:



Implementing this architecture provides structures and components that will achieve dramatic improvements in application integrity and flexibility, and produce dramatic increases in agility. It will provide mechanisms for recovering the true competitive advantage encapsulated within the heritage code.

Of particular importance is the gradual transition away from RLA (record level access) or Native IO, in favor of introducing IO Servers that produce result sets for consumption/processing by your RPG or other HLL programs. The long-term objective should be to get the database engine to perform as much dataset pre-processing as possible, as it delivers result sets far more efficiently and effectively than can be achieved in any HLL program.

Another aim should be delivering system and application wide generic services as *SRVPGMs, to be available as components anywhere in the system, similar to the ESB concept in the SOA world,

Using AO Foundation to achieve long-term modernization

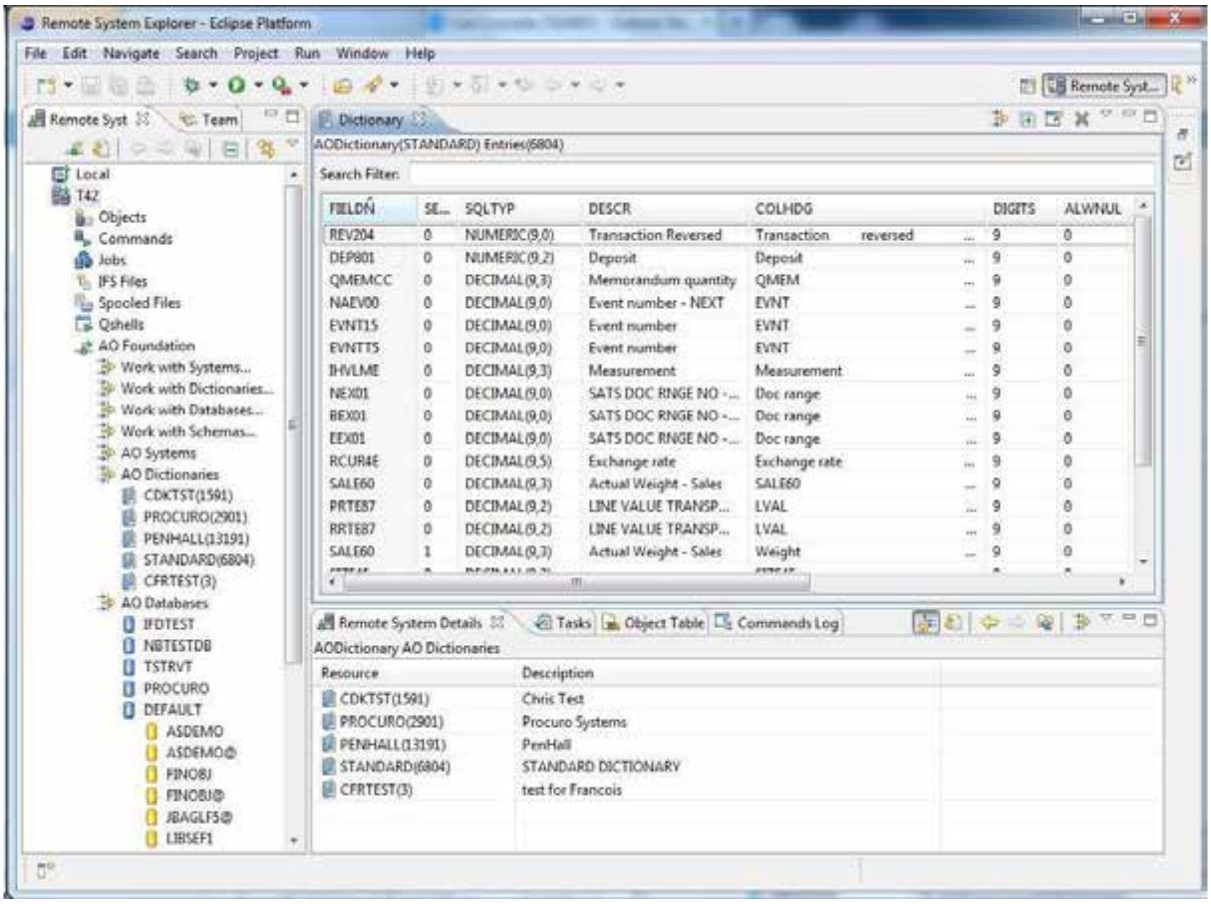
To achieve the highlighted application architecture, the usual process to be followed includes the following steps:

1. Migrating the base database definitions from DDS to DDL with no LVLID changes. During this initial step, problematic or non-supported constructs are usually excluded. The power of

DB2 on IBM i allows the use of any combination of old and new constructs, with little to no negative consequences.

2. Automatic synchronization (AO Replay) of your database content between your DDS production database and your new DDL database. This allows for seamless, non-disruptive switching between old DDS and new DDL production databases.
3. The generation of a new data dictionary, that will become your new central repository of all metadata and structural metadata. This dictionary will be used as “mother lode” to consolidate and sanitize your metadata, removing especially inconsistent attributes and removing all duplication. This new data dictionary will also become the repository of all enrichments activities on the metadata (standard fonts, colors, icons, URL’s, other behavior), which can be automatically consumed in future iterations of UI’s, printing functions, etc. This new repository will eventually replace your FRF (Field Reference Files).
4. The implementation of unique keys on tables, using existing logical database constructs, to generate a true relational database at the DB2 layer. This allows the generation of “automatic” constraints between entities, improving database structures and integrity significantly.
5. The exposure of long file and field names (table and column names) as aliases at a DB2 layer level, which immediately improves the perceptions and experience of your users, in the way the database presents itself to BI and other analytical tools.
6. Using these new DDL definitions to gradually drive data validations down into the database engine.
7. Separation of all UI, by leveraging Rational Open Access for RPG (ROA) and a few of the top UI tool vendors. AO supports a number of these. By using ROA, many of the constraints imposed by the 5250 datastream effectively disappear, providing a significantly improved UI/UX of your heritage application.
8. Using the above constructs to serve as foundation to now gradually recover and encapsulate those components in your systems causing agility constraints or that represents competitive advantage “hidden” in your business logic.

Below is a screen shot of the current beta version of our Eclipse based RSE (Remote System Explorer) plug-in, which ensures that AO can be used by any of your RDP or RDi enabled programmers. By the time this document is published, this feature of AO will be generally available. Additionally it will also be integrated with IBM DataStudio, to improve graphical database modeling:



AODictionary(STANDARD) Entries(6804)

FIELDN	SE...	SQLTYP	DESCR	COLHDG		DIGITS	ALWNUL
REV204	0	NUMERIC(9,0)	Transaction Reversed	Transaction reversed	...	9	0
DEP801	0	NUMERIC(9,2)	Deposit	Deposit	...	9	0
QMEMCC	0	DECIMAL(9,3)	Memorandum quantity	QMEM	...	9	0
NAEV00	0	DECIMAL(9,0)	Event number - NEXT	EVNT	...	9	0
EVNT15	0	DECIMAL(9,0)	Event number	EVNT	...	9	0
EVNTT5	0	DECIMAL(9,0)	Event number	EVNT	...	9	0
IHVLME	0	DECIMAL(9,3)	Measurement	Measurement	...	9	0
NEX01	0	DECIMAL(9,0)	SATS DOC RNGE NO -...	Doc range	...	9	0
BEX01	0	DECIMAL(9,0)	SATS DOC RNGE NO -...	Doc range	...	9	0
EEX01	0	DECIMAL(9,0)	SATS DOC RNGE NO -...	Doc range	...	9	0
RCUR4E	0	DECIMAL(9,5)	Exchange rate	Exchange rate	...	9	0
SALE60	0	DECIMAL(9,3)	Actual Weight - Sales	SALE60	...	9	0
PRTE87	0	DECIMAL(9,2)	LINE VALUE TRANSP...	LVAL	...	9	0
RRTE87	0	DECIMAL(9,2)	LINE VALUE TRANSP...	LVAL	...	9	0
SALE60	1	DECIMAL(9,3)	Actual Weight - Sales	Weight	...	9	0

AODictionary AO Dictionaries

Resource	Description
CDKTST(1591)	Chris Test
PROCURO(2901)	Procuro Systems
PENHALL(13191)	PenHall
STANDARD(6804)	STANDARD DICTIONARY
CFRTEST(3)	test for Francois

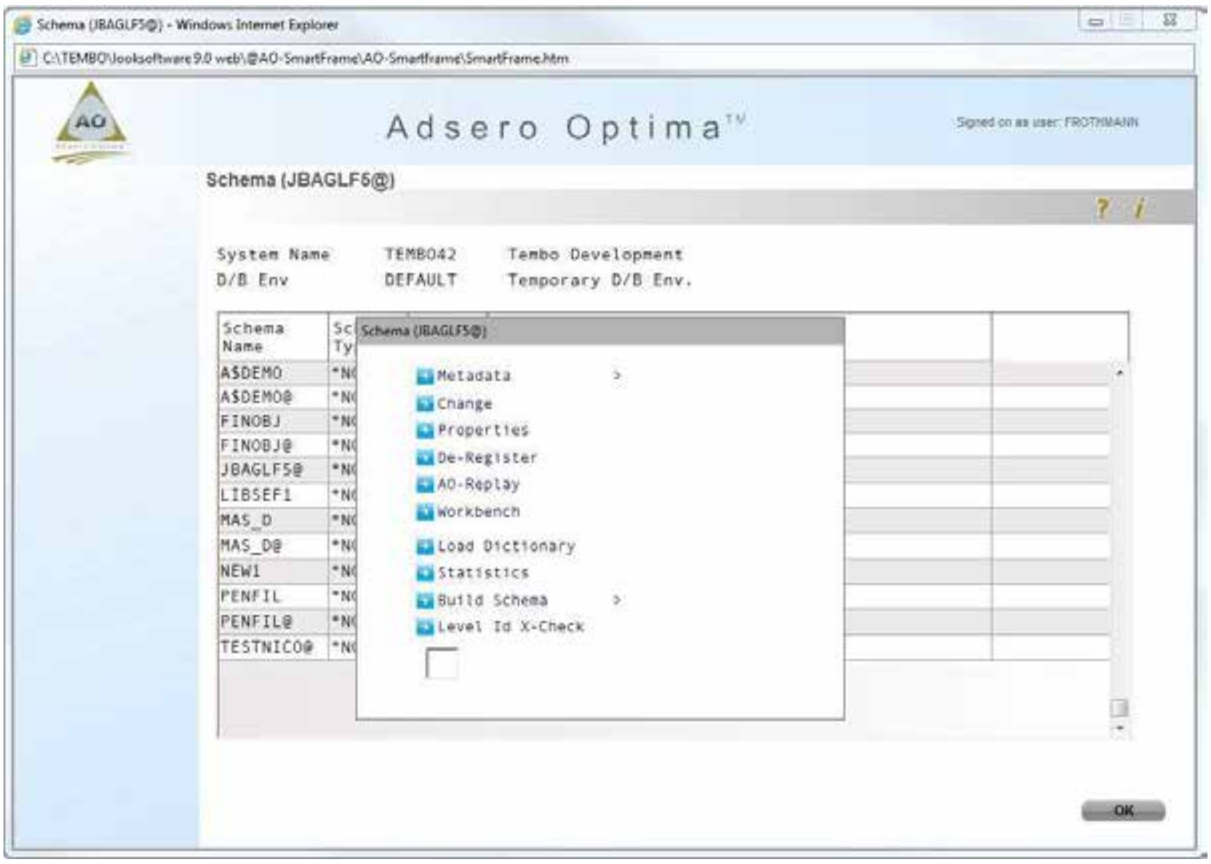
Due to the immense function-rich Eclipse/RDP/RDi interface, the balance of the screen shots in this document, will use AO UIM screens. UIM (same architectural construct used for most of the IBM i operating system UI) based screens are used to highlight only those functions relevant for this solution guide.

The same application architecture has been used that all customers and ISV's should aspire to. This means AO uses IO Servers, Enterprise Services and *SRVPGM's that provide the result sets that are processed by both our UIM panels and the JAVA based Eclipse interface.

Based on experience, this is probably the area where most customers experience significant pain, as they re-code entity relationships and validation rules that **ALREADY EXIST in their RPG programs**, in their new C#, JAVA, PHP and other HLL environments, for far reaching (painful) consequences. It would have been significantly easier (and with dramatic integrity improvements) for them to expose their existing relationships and validations, by rolling this back into DB2. By using this properly, all heritage and new applications should use the same enterprise services and IO Servers, to consume result sets from the database.

AO Foundation DDS to DDL Migration

As indicated in the “AO in a Nutshell” animation, the structural metadata is extracted directly from the internal DB2 object structures, which means that the migration can be performed from DDS to DDL without source code. All actions are carried out on AO internal definitions and NEVER against any DDS production versions. See http://www.adsero-optima.com/nutshell-msp



Schema (JBAGLF5@)

```
System Name    TEMBO42    Tembo Development
D/B Env        DEFAULT    Temporary D/B Env.
```

Schema Name	Sc Ty
A$DEMO	*N(
A$DEMO@	*N(
FINOBJ	*N(
FINOBJ@	*N(
JBAGLF5@	*N(
LIBSEF1	*N(
MAS_D	*N(
MAS_D@	*N(
NEW1	*N(
PENFIL	*N(
PENFIL@	*N(
TESTNICO@	*N(

One of the fundamental design considerations of AO is that it should provide one single management interface for ALL development and maintenance functions and to facilitate the introduction of a true database scientist (DBS), database engineer (DBE) or database architect (DBA) type function, dependent upon the scale of the environment involved. Based on our experience, 80% of your application functions in the average online commercial transaction processing application today, will be (or should be) implemented within the database engine layer.

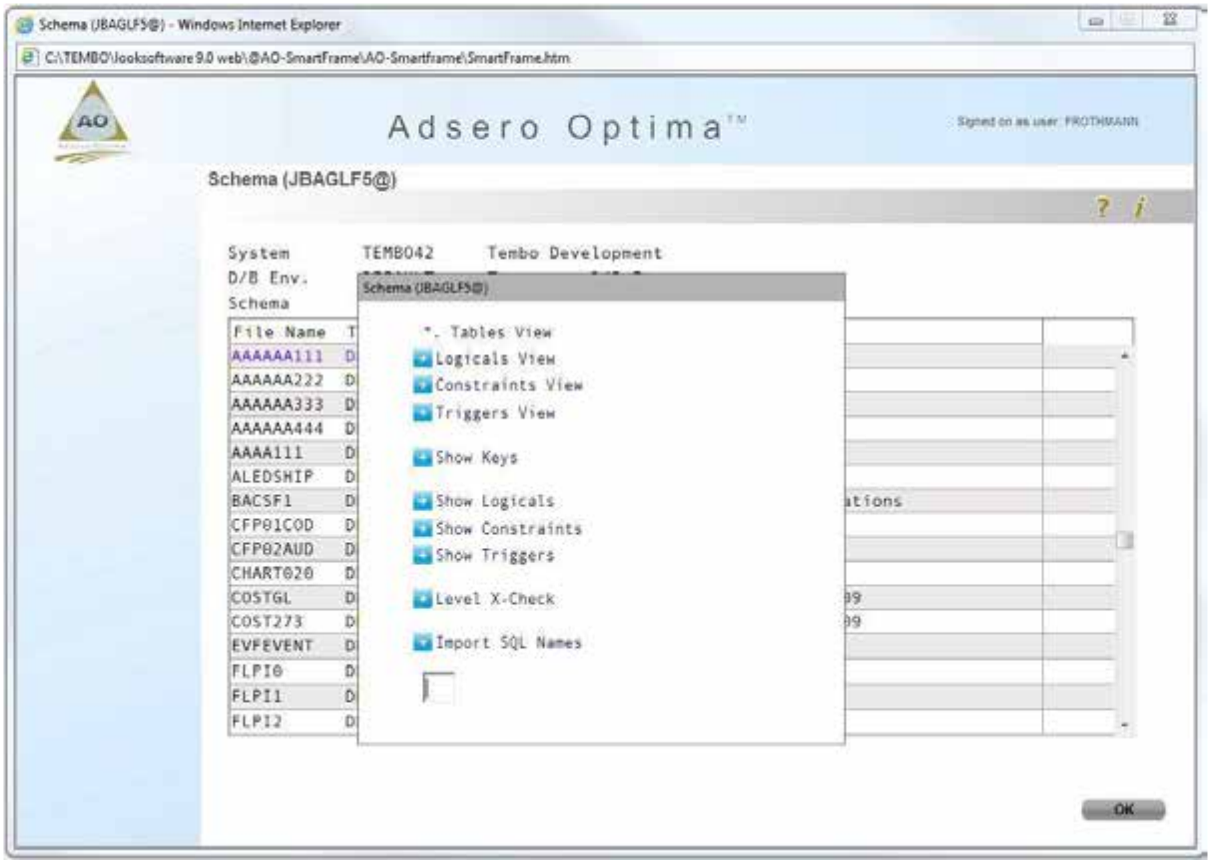
AOF continuously tracks the database “health” and statistics, to allow the DBS/DBE/DBA to focus on the areas/components that will deliver most significant improvements as you modernize your heritage application portfolio. The most significant functions above include AO Replay that facilitates the transparent migration of the old (DDS) production database to the new DDL database. This can be achieved while running in production, as AO ensures that the databases are synchronized.

The development team and the developer resources use the AOF Database Workbench function as they continue to modernize and improve heritage applications. This is a powerful and function rich feature within AOF.

The Statistics and Level ID Cross Check functions allows your DB staff to continuously measure progress of the modernization project and what the potential impact is.

AO Foundation – Database Workbench

The focus of the AOF Database Workbench function is to provide a single view of every component of the database, including a logical perspective (all LF, views, and indexes), a separate perspective for constraints, keys, and triggers as well as functions to display all database components on a single view.



Schema (JBAGLF5@) - Windows Internet Explorer

C:\TEMBO\looksoftware 9.0 web\@AO-SmartFrame\AO-Smartframe\SmartFrame.htm

Adsero Optima™

Signed on as user: PROTHMANN

Schema (JBAGLF5@)

System TEMB042 Tembo Development

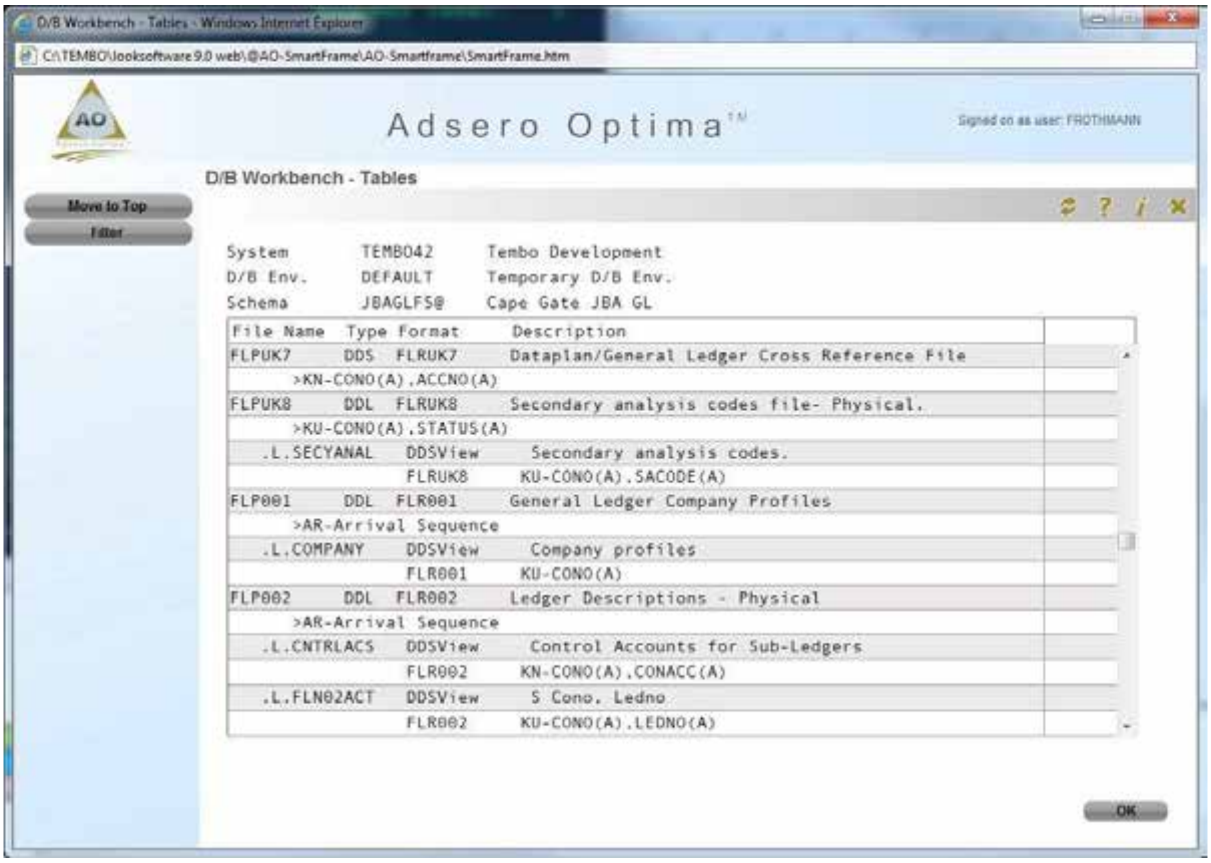
D/B Env.

Schema

File Name	T
AAAAAA111	D
AAAAAA222	D
AAAAAA333	D
AAAAAA444	D
AAAA111	D
ALEDSHIP	D
BACSF1	D
CFP01COD	D
CFP02AUD	D
CHART020	D
COSTGL	D
COST273	D
EVFEVENT	D
FLPI0	D
FLPI1	D
FLPI2	D

The next example shows how this function can be used to display all logical database constructs. This can be used to identify logical file definitions to determine the most suitable candidate to use as a unique key. The key can be implemented at the new DDL Table level for use in generating constraints and relationship definitions.

AOF has very powerful tooling focused on making the development team more efficient and accurate when working with the heritage database constructs and components. The objective is to provide all the required information within a few keystrokes/mouse clicks, to help make an informed, scientific decision and then implement this back into your new database with maximum efficiency. AOF also highlights the impact of that change throughout the database.

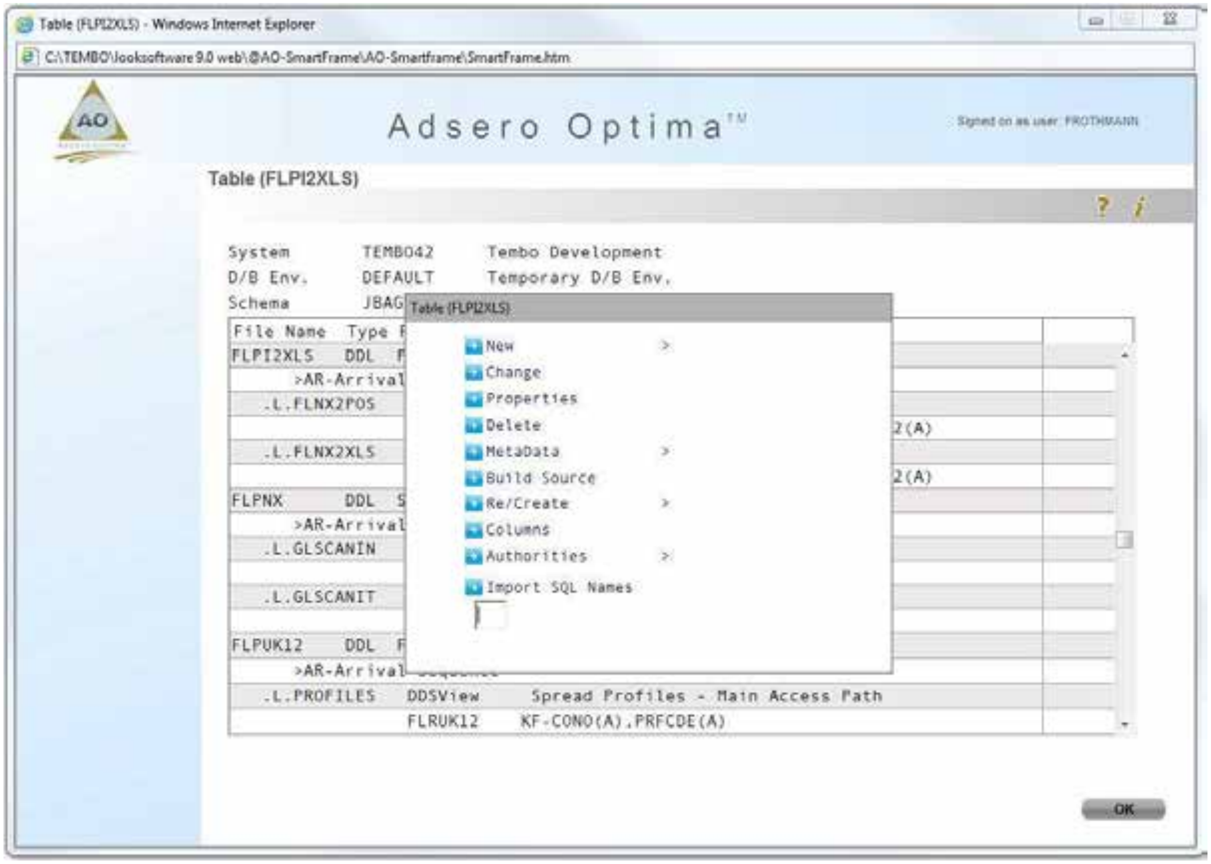

D/B Workbench - Tables - Windows Internet Explorer
C:\TEMBO\looksoftware 9.0 web\@AO-SmartFrame\AO-Smartframe\SmartFrame.htm
Adsero Optima™
Signed on as user: FROTHMANN
D/B Workbench - Tables
Move to Top
Filter
OK

```
System       TEMBO42      Tembo Development
D/B Env.     DEFAULT      Temporary D/B Env.
Schema       JBAGLFS@     Cape Gate JBA GL

File Name   Type Format      Description
FLPUK7      DDS  FLRUK7      Dataplan/General Ledger Cross Reference File
        >KN-CONO(A).ACCNO(A)
FLPUK8      DDL  FLRUK8      Secondary analysis codes file- Physical.
        >KU-CONO(A).STATUS(A)
   .L.SECYANAL    DDSView      Secondary analysis codes.
                  FLRUK8     KU-CONO(A).SACODE(A)
FLP001      DDL  FLR001      General Ledger Company Profiles
        >AR-Arrival Sequence
   .L.COMPANY     DDSView      Company profiles
                  FLR001     KU-CONO(A)
FLP002      DDL  FLR002      Ledger Descriptions - Physical
        >AR-Arrival Sequence
   .L.CNTRLACS    DDSView      Control Accounts for Sub-Ledgers
                  FLR002     KN-CONO(A).CONACC(A)
   .L.FLN02ACT    DDSView      S Cono. Ledno
                  FLR002     KU-CONO(A).LEDNO(A)
```

The above is a typical view of a heritage application database that is midway through a database modernization project. The database shown has been in production since April 2012 with absolutely no disruption or downtime to the users of this system (which is a 24 x 7 manufacturing installation). The remaining DDS physical file on this panel is completely regenerated, which, functions similar to the DDL generated objects. This PF was completely regenerated from the extracted metadata and structural metadata with, in this case, a page size of 64k (this is a standard parameterized setting in AO). AO retained it as DDS, while allowing the customer development staff to research the best option to implement a unique key on this file.

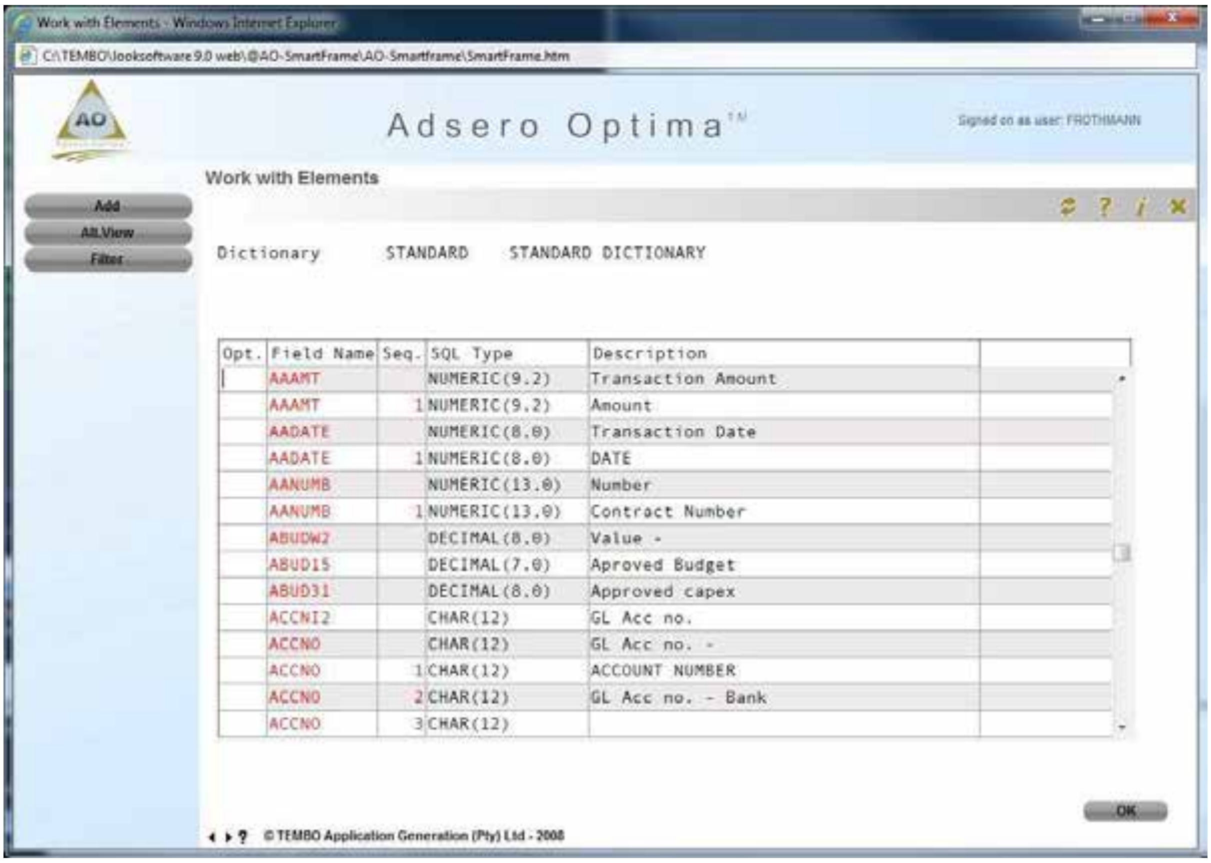
As indicated in the next panel, one of the objectives of AOF is to provide a single management and maintenance platform for your entire database, with a particular focus to assist in bringing heritage database constructs and components into the new millennium. The product will generate any source, which means it can also be used to educate heritage staff in the use and syntax of DDL, or for generating source code for integration into change and configuration management products to move these changes into production.



```
System        TEMBO42      Tembo Development
D/B Env.      DEFAULT      Temporary D/B Env.
Schema        JBAG
File Name  Type F
FLPI2XLS    DDL   F
     >AR-Arrival
  .L.FLNX2POS
                                                 2(A)
  .L.FLNX2XLS
                                                 2(A)
FLPNX       DDL   S
     >AR-Arrival
  .L.GLSCANIN

  .L.GLSCANIT

FLPUK12     DDL   F
     >AR-Arrival
  .L.PROFILES    DDSView     Spread Profiles - Main Access Path
                 FLRUK12    KF-CONO(A),PRFCDE(A)
```


Work with Elements - Windows Internet Explorer
C:\TEMBO\looksoftware 9.0 web\@AO-SmartFrame\AO-Smartframe\SmartFrame.htm
AO
Adsero Optima™
Signed on as user: FROTHMANN
Work with Elements
Add
All.View
Filter
OK
© TEMBO Application Generation (Pty) Ltd - 2008

Dictionary   STANDARD   STANDARD DICTIONARY

Opt.	Field Name	Seq.	SQL Type	Description	
	AAAMT		NUMERIC(9.2)	Transaction Amount	
	AAAMT	1	NUMERIC(9.2)	Amount	
	AADATE		NUMERIC(8.0)	Transaction Date	
	AADATE	1	NUMERIC(8.0)	DATE	
	AANUMB		NUMERIC(13.0)	Number	
	AANUMB	1	NUMERIC(13.0)	Contract Number	
	ABUDW2		DECIMAL(8.0)	Value -	
	ABUD15		DECIMAL(7.0)	Aproved Budget	
	ABUD31		DECIMAL(8.0)	Approved capex	
	ACCNI2		CHAR(12)	GL Acc no.	
	ACCNO		CHAR(12)	GL Acc no. -	
	ACCNO	1	CHAR(12)	ACCOUNT NUMBER	
	ACCNO	2	CHAR(12)	GL Acc no. - Bank	
	ACCNO	3	CHAR(12)		

The previous panel provided a glimpse of the functionality available in the new consolidated data dictionary. This functionality and tooling provides the capability to remove all metadata duplication, by highlighting the representative master definition of that element. This can then be used to consistently roll this master definition back into the production database, removing all duplication and inconsistencies in the metadata. The amount of duplication and inconsistencies in your metadata may be surprising, but it is due to the age of the systems and the methodologies used to create them...

For example, we have found more than fifty instances of a particular metadata element in a customer database, with quite a few different attributes. This is in no way unique, as many systems are 20 years or older with many programmers having maintained the system during the elapsed period.

As indicated earlier in this document, we also believe that during review and consolidation of your metadata and data dictionary, you should start the process of enriching your metadata, as this will improve the quality of your system and what can be achieved with your metadata significantly.

Conclusion

Based on extensive experience, it is clear that massive value remains in heritage applications and that compelling business justification exists to modernize heritage application assets. This process has to start at the database layer, as approximately 80% of our heritage functions belong in the database engine in the modern commercial application development paradigm.

Adsero Optima™ achieves this by:

1. Facilitating the (mostly automated) migration from DDS defined database structures, to native DDL defined database structures as an immediate initial step. This becomes the absolute foundation for ANY and ALL subsequent modernization.
2. Consultation to determine a modernization strategy and plan, and to highlight future development potential.
3. Consolidating and sanitizing metadata and structural metadata, as an immense amount of duplication and inconsistencies have been integrated within your applications and the underlying database due to decades of neglect.
4. Introducing referential constraints into the database, with fundamental knowledge recovered from your applications. This immediately starts to improve data integrity by orders of magnitude.
5. Removing user interface constraints (strategic partnerships), to facilitate multi-channel delivery of application functionality to any current or new UI delivery channel.
6. Recovering validation rules from the code base, facilitating the introduction of these validation rules as triggers directly into the database. This provides a process that will facilitate consolidation of all validation rules, improve data integrity and agility and achieve single instances (as opposed to multiple instances) of the validation rules. Additionally these rules will be implemented and consumed consistently throughout the system, regardless from where this is accessed.
7. Providing a gradual recovery and modernization of true business unique logic, this will represent the underlying competitive advantage encapsulated by your heritage application.

Supported platforms

This solution is supported by platforms with the following features:

- The AO Roadmap can be applied on ANY release of IBM i, or earlier releases of OS/400 and i5/OS, although some manual work will have to be performed.
- AO Foundation requires IBM i V6R1M0 as minimum.