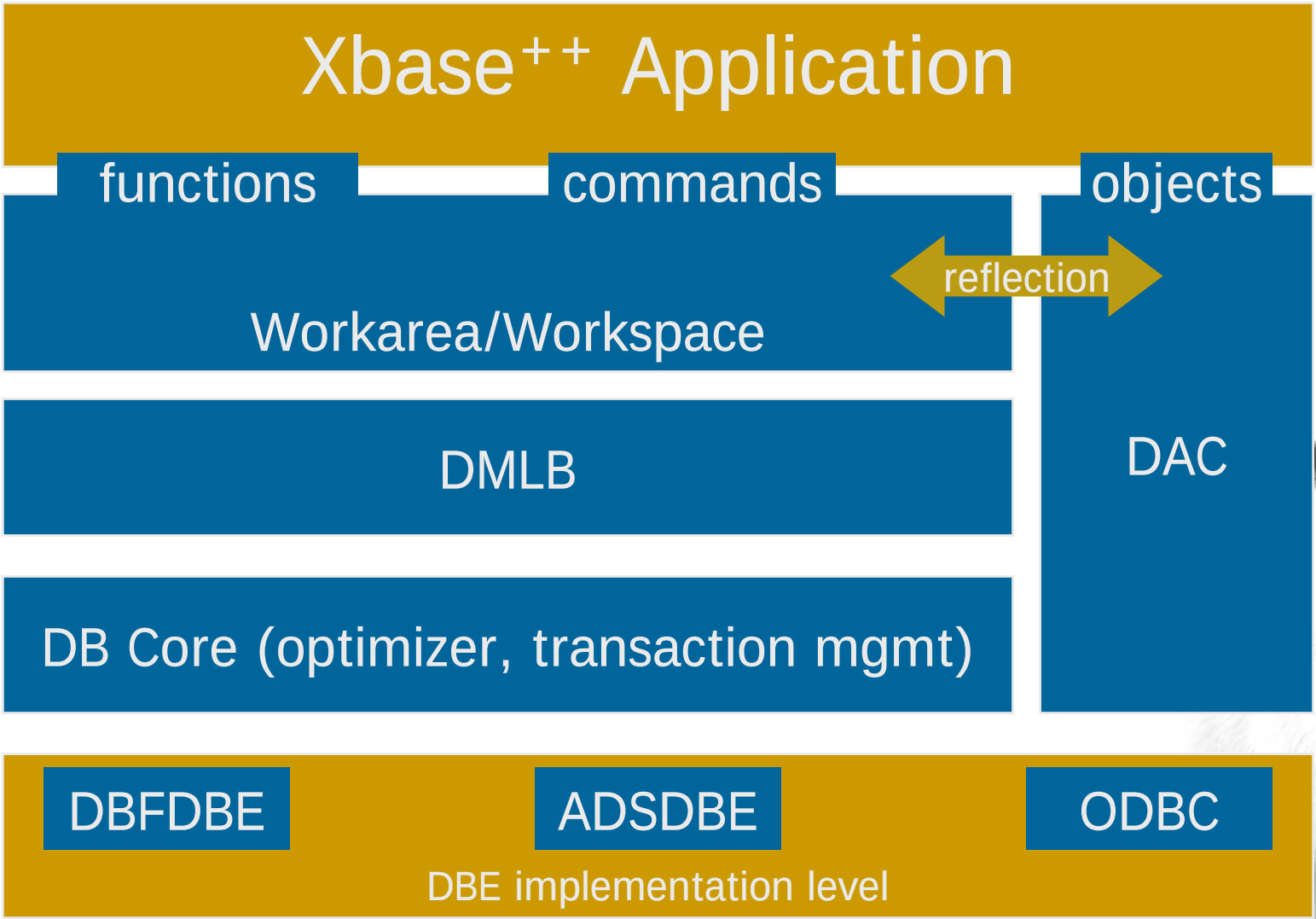

# The Advantage Database Server DatabaseEngine for Xbase++

# Agenda

- Xbase++ Architecture
- DatabaseEngine concept
- The ADSDBE
- Usage
- Migration
- Considerations
- Extended features
- Summary

# Xbase++ Data-Access-Architecture



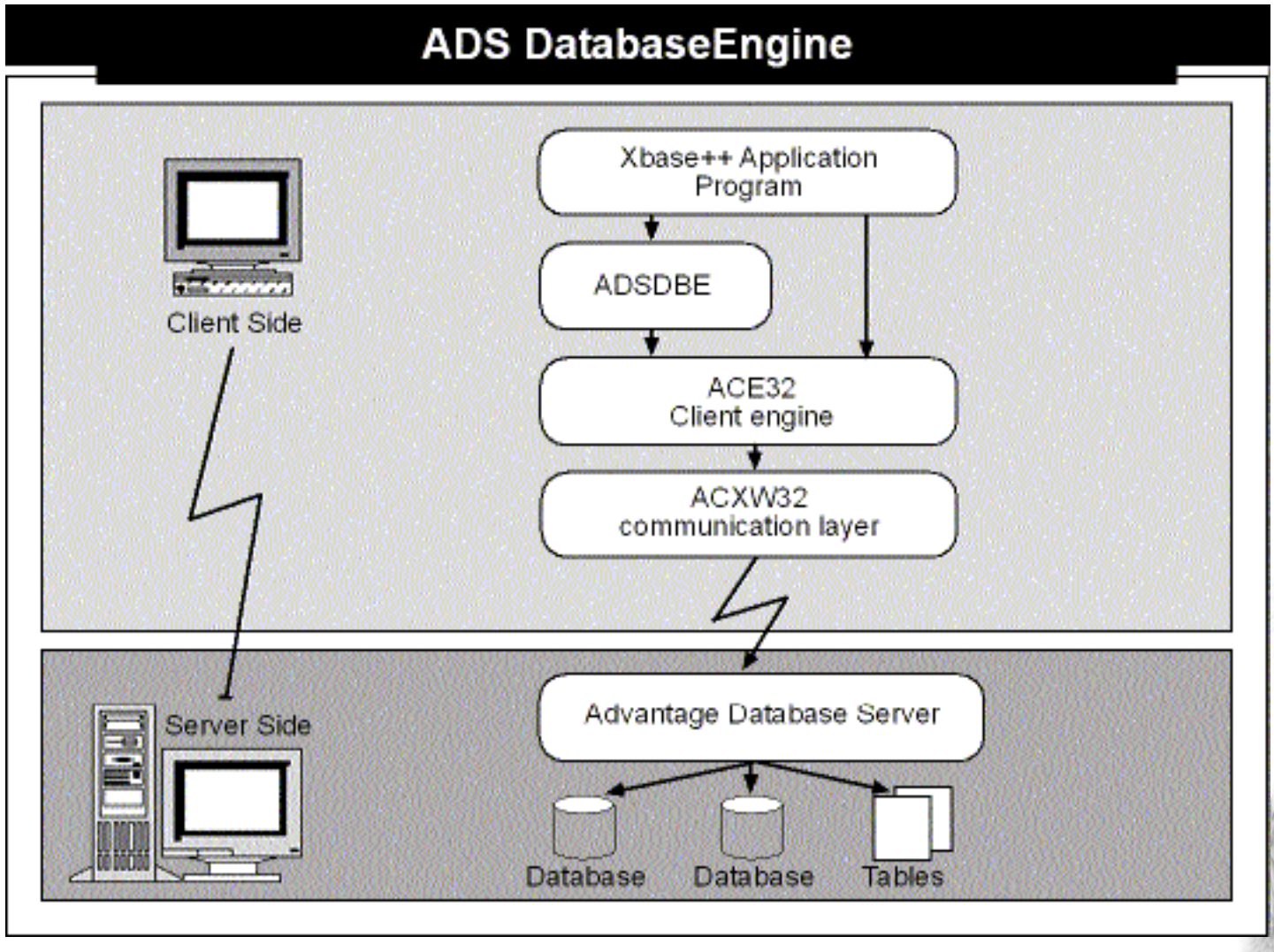

# DatabaseEngine concept

- DatabaseEngines are dynamic loadable at runtime (see dbesys.prg for default behaviour)
- A DatabaseEngines supports at least one out of four domains. (Storage, Order, Relation and/or Dictionary)
- Any DatabaseEngine can be configured for different behaviour. (DbeInfo())
- You can build your own logical DatabaseEngine by mixing capabilities of different physical engines

# The ADSDBE

- DatabaseEngine supporting Storage, Order and Relation capabilities
- With Version 1.9 Dictionary capabilities are supported too
- Can be used stand alone or mixed with any other DatabaseEngine
- Supports Advantage Database Server Version 5.7 and higher.
- Support IPX/SPX, TCP/IP and Netbios connections
- Supports Netware, Windows and Linux as the Server operating system

# The ADSDBE architecture




# Usage scenario

- Simple create/use table pattern
- Now using the ADS Server

# Different connection methods

- Generic
  - DacSession():New("DBE=ADSDBE;SERVER=your location”)
- Mapped drive
  - SERVER=F:
- UNC path
  - SERVER=\\ALASKA01\SOMEWHERE
- Dictionary
  - SERVER=\\ALASKA01\SOMEWHERE\Product Group.add
- IP:Port
  - SERVER=\\ads70.alaska-software.com:6262\ProductGroup
  - SERVER=\\192.168.2.1:6262\ProductGroup

# Best practice re. connections

- Use UNC path-names in your windows applications. Don’t relay on specific drives mapped.
- Don’t mix up drive names and unc pathes. This is horrible to debug
- In case you are using Linux, use lower case path and filenames only
- Always use SET DEFAULT TO with the path/location used in your connection string

# Considerations (1/2)

- OEM/ANSI
  - The ACE32 layer is ANSI only. Therefore you can not safely store any OEM data in character fields. Use binary fields instead or change your application runtime to ANSI (SET CHARSET TO ANSI)
- Collation tables
  - ADS supports collation tables based on the server installation. Different servers with different collation tables are problematic. Multi-national applications must select collation tables carefully at server installation.

# Considerations (2/2)

- Expressions
  - In general expression are limited to the predefined set of functions and operators supported by the ADS Server.
  - UDFs are not supported
- Xbase++ Database-Kernel automatically detects Filter expressions which can not evaluated by the ADS Server. These type of Filter expressions get evaluated locally. This however does decrease performance. (check DbInfo(ADSDBE,ADSDBO_LOCALFILTER) == .T.)
- Index expressions are limited to ADS functions/operators only. No local expression evaluation possible.

# ADT special considerations (1/2)

- New set of datatypes, select carefully
- NUMERIC if used is automatically mapped to double! (Exact numeric is now approximate numeric type)
- Empty date is not valid anymore “ . . “, ADT uses NULL/NIL to signal empty-date.
- By default all ADT fields are NULLABLE, therefore SET NULLVALUE OFF for compatibility.
- Default value of ADT fields is NULL, use dictionary if possible to setup a default value
- ADT is ANSI only! There is no fixed size binary type. Therefore you have no way to workaround the ANSI only limitation.

# ADT special considerations (2/2)

- DbDelete() does delete the record. There is no Recall() for ADT tables. (beware of implicit Navigation)
- Be reminded, ADS Server uses two different expression engines which are not compatible. One for Xbase expressions in filters, another engine for SQL expressions.
- Use IsNull() instead of IIF() to check for NULL/NIL value in your expressions

# Using the ACE32 API

- The ADSDBE provides you access to the system level handles
  - Connection handle oSession:GetConnectionHandle()
  - Table handle DbInfo(ADSDBO_TABLE_HANDLE)
  - Index/Order handle OrdInfo(ADSORD_INDEX_HANDLE)
- Use DllCall() or DLLFUNCTION to declare API.

# SQL with the ADSDBE

- The ADS DatabaseEngine accepts cursor handles with the DbUseArea() function.
- We can retrieve the connection handle with oSession:GetProperty()
- We can execute a ACE32 API using DLLFUNCTION

# Extended features:

- Enable properietary locking, does increase performance.
- Use transactions, this does increase performance too and will make your applications more reliable. Specifically in credit/debit scenarios.


# Thanks for your attention !

Please visit us at

www.alaska-software.com