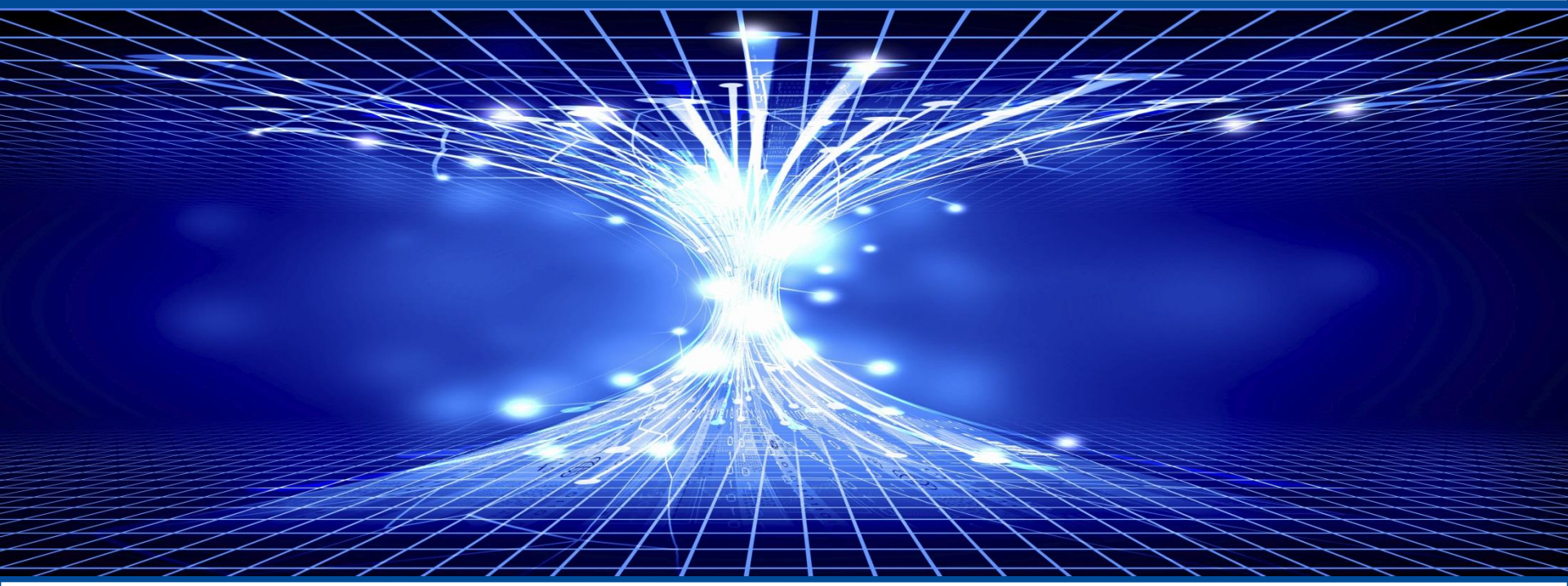
UNISYS | Securing Your Tomorrow™



Development for Performance

Objective

By the end of this module, you'll be able to—

- Identify best practices and guidelines that can help you maintain your model better and achieve better runtime performance

Module Topics

- Database Structures
- Best Practices – Runtime Performance
- Database Access Commands
  - Best Practices – Read and Write Operations

Introduction

- The module discusses best practices and guidelines to achieve better performance for both AB Suite Runtime for MCP and AB Suite Runtime for Windows
- Some guidelines are applicable to both platforms while some are applicable to individual runtimes
- Slide Legend:

MCP	for MCP related guidelines
Windows	for Windows related guidelines

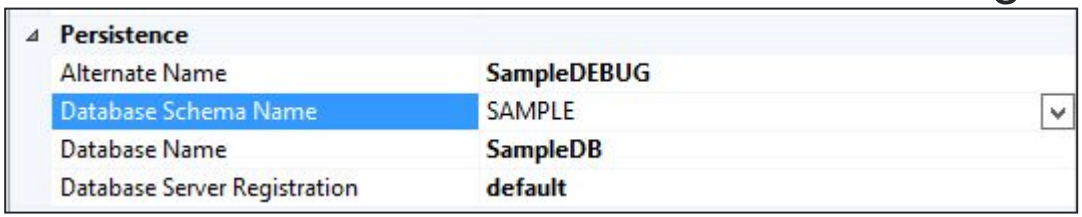
UNISYS | Securing Your Tomorrow™

Database Structures

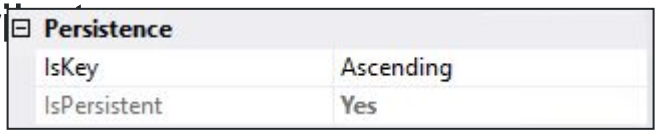
Windows

Database Structures—Windows Runtime

- A SQL Server database table is created for each persistent class.
- Persistent attribute is a column in the database table.
- Database schema name is the *Database Schema Name* configuration property.



- Keys are set with the *IsKey* property of an attri



- Non-conditional profiles are indexes over the class table.
- Conditional profiles are indexed views.
- Database table is created for each event set. AB Suite also allows multiple event sets.

MCP

Database Structures—MCP Runtime

- A DMS II dataset is created for each class containing persistent attributes.
- A persistent attribute becomes an item in the dataset.
- Collection of items in a dataset comprise a record. (For example, CUST)
- Ordinates are set with the *IsKey* property of an attribute.
- Non-conditional profiles are sets.
- Conditional profiles are subsets.
- A dataset is created for each event set. AB Suite supports multiple event sets.
- All datasets are of type STANDARD by default. You might select a DIRECT type for special cases. For example, ascending contiguous numeric key.

Database Structures—MCP Runtime

- All Profiles (sets or Subsets) are INDEX SEQUENTIAL.
- The DMS II database schema is described in Data And Structure Definition Language (DASDL).

Multiple Event Sets

- Multiple event sets allow unrelated transactions to be split into multiple event sets.
- For example, you can create separate event sets for credit card transactions, loan transactions, and savings account transactions.

CustID	TrType	amount	Balance	Ref No	Date
c001	Cr	1000	14000	719800	02/02/2012
c003	D	20000	45000	998870	02/02/2012
c018	D	20000	45000	998871	02/02/2012

Credit Card Transactions

CustID	Date	Tran type	Amount	Balance
c003	02/02/2012	I	200	200000
c004	02/02/2012	W	10000	50000
c035	02/02/2012	D	-3980.09	70000

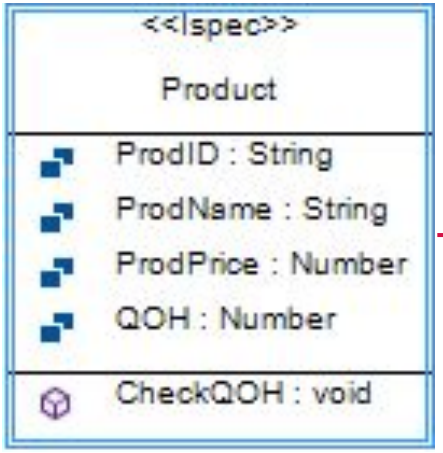
Account Transactions

CustID	Date	Tran Details	Debit	Credit	Balance
c025	02/02/2012	Repayment	-2152.42		-337, 906.55
c028	02/02/2012	Account fee	-30		-257,999,02
c035	02/02/2012	Repayment	-3980.09		-456,000.76

Loan Transactions

Persistent Structures

- Persistent structures are created as a table in the database.
- For example, an ispec called Product will be stored in a table called Product in the database, and each persistent attribute will be stored in a column in that table.

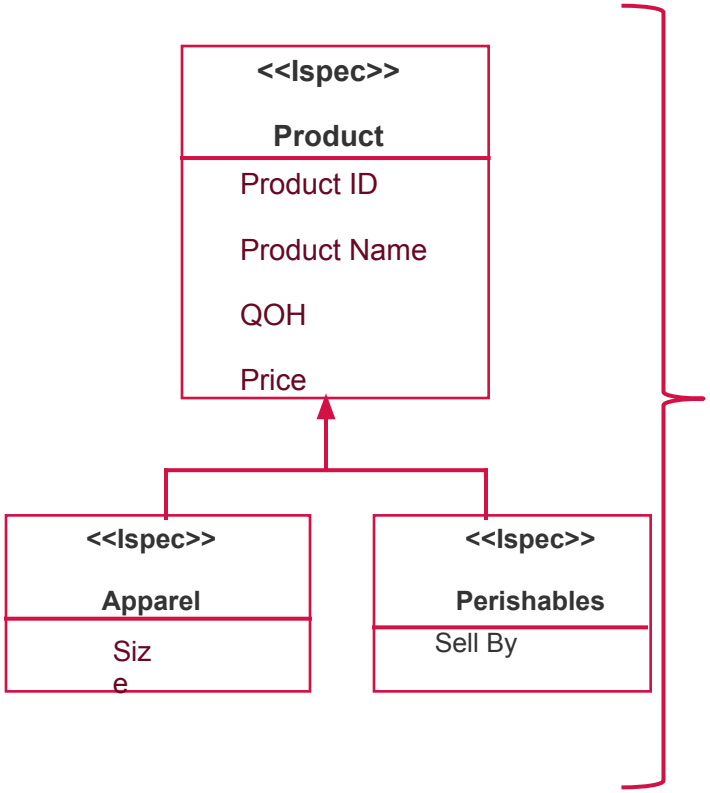


Product Table

Prod ID	ProdName	ProdPrice	QOH
P012	Product_Line2	150	40
P034	Produc_Line1	100	150

Polymorphic Persistent Structures

When a persistent class inherits from a superclass, the database table acts as one data source containing the persistent elements of the superclass and the subclass.



Class

Apparel

Perishables

Apparel

Product

Perishables

Product

Prod ID	ProdName	Prod Price	QOH	SellBy	Size
P001	Product_App_A	75	30		5
P002	Product_Peri_Type1	30	100	03/09/2013	
P005	Product_App_B	72	100		10
P012	Product_Line2	150	40		
P031	Product_Peri_Type3	100	50	03/08/2012	
P034	Produc_Line1	100	150		

UNISYS | Securing Your Tomorrow™

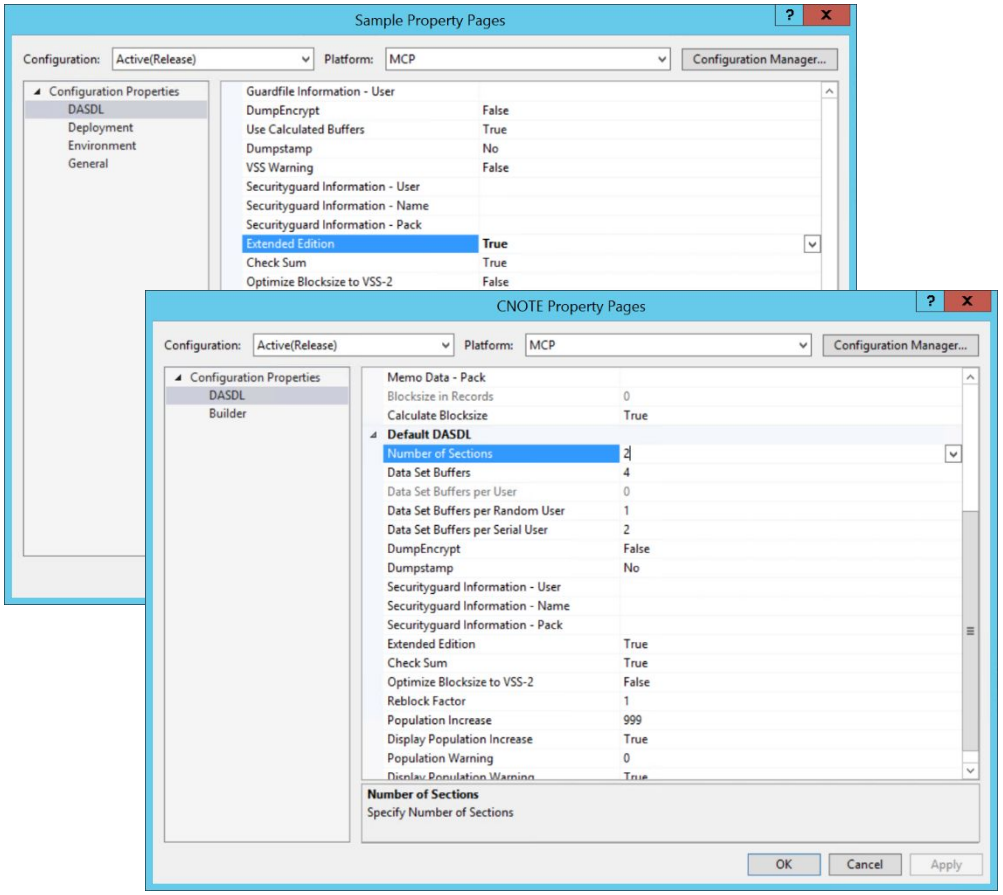
Better Runtime Performance: Best Practices

Minimizing Deadlocks and Performance Degradation

MCP

Minimizing Deadlocks:

- For event or persistent inheritance structures, consider applying sections.



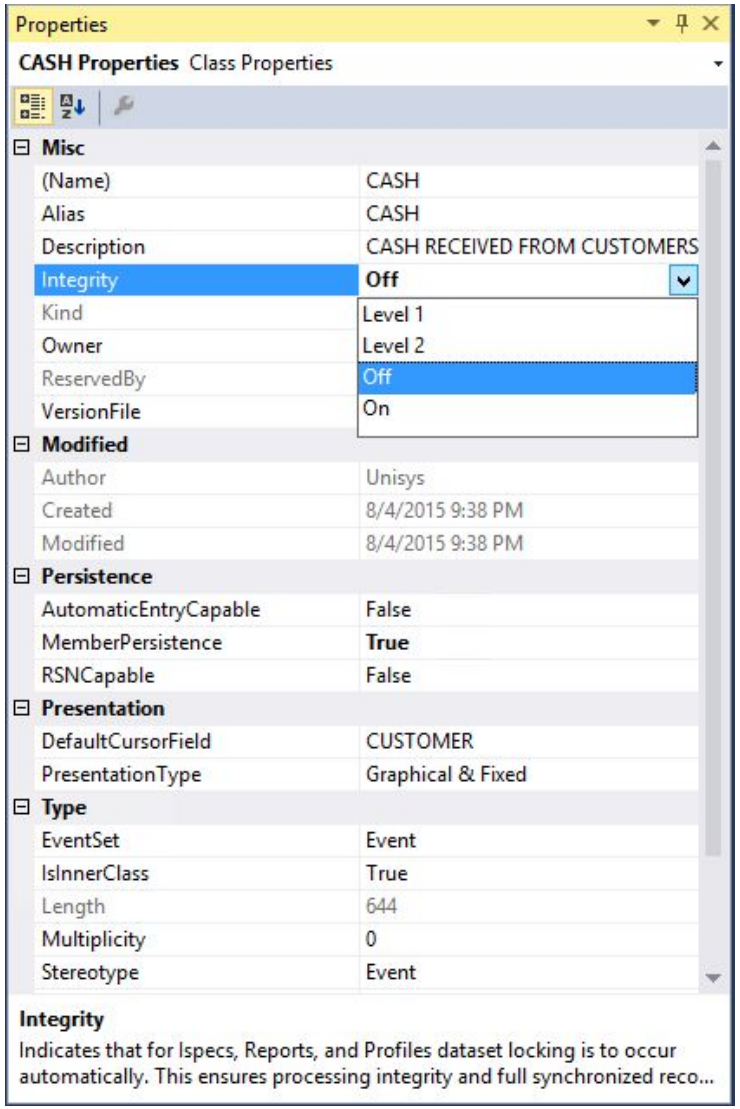
To set the number of sections:

1. Enable the *Extended Edition* DASDL configuration property for the segment.
2. Specify a value in the *Number of Sections* DASDL configuration property for the persistent structure.

Minimizing Deadlocks and Performance Degradation

MCP

- Excessive use of the integrity settings can result in deadlocks and performance degradation.
- For cases where only specific structures require data protection, use the SECURE qualification on ForEach, DT, and LU as an alternative.
- You can set the Integrity level for an element such as ispec, reports, and profiles by setting the *Integrity* property.



Minimizing Deadlocks and Performance Degradation

MCP

- For example, an application might have a control class (CNTRL) that is updated by every ispec. Each ispec updates the CNTRL record by reading the record and then flagging back an updated counter value.
- Following is an example of LDL+ logic:

```
 ..
DT EVERY P_CNTRL (MYISPEC)  SECURE   :lock the record so that it is not
                        compromised
    BREAK
END
ADD 1  CNTRL.COUNTER GIVING SD_COUNTER
----- :more logic here
----- :more logic here
FLAG SD_COUNTER   CNTRL.COUNTER
```

The SECURE will ensure no one else can compromise your update by preventing other users from updating this record while you are processing it.

Minimizing Deadlocks and Performance Degradation

MCP

- For AB Suite applications, a valid restart point/syncpoint is created for every report execution at the time of the first database update, thus creating a control record to which the database can be rolled back in the event of a problem.
- If reports are waiting for the syncpoint to occur, there will be a performance degradation. Syncpoints can only be taken at a quiet point in database activity. For this reason, syncpoints are not taken in co-routines.
- To manage this performance degradation, it is necessary to:
  - Minimize the time that an ispec or report spends in transaction state
  - Consider reports that are run from an ispec being run before the ispec goes into transaction state

Minimizing Deadlocks and Performance Degradation

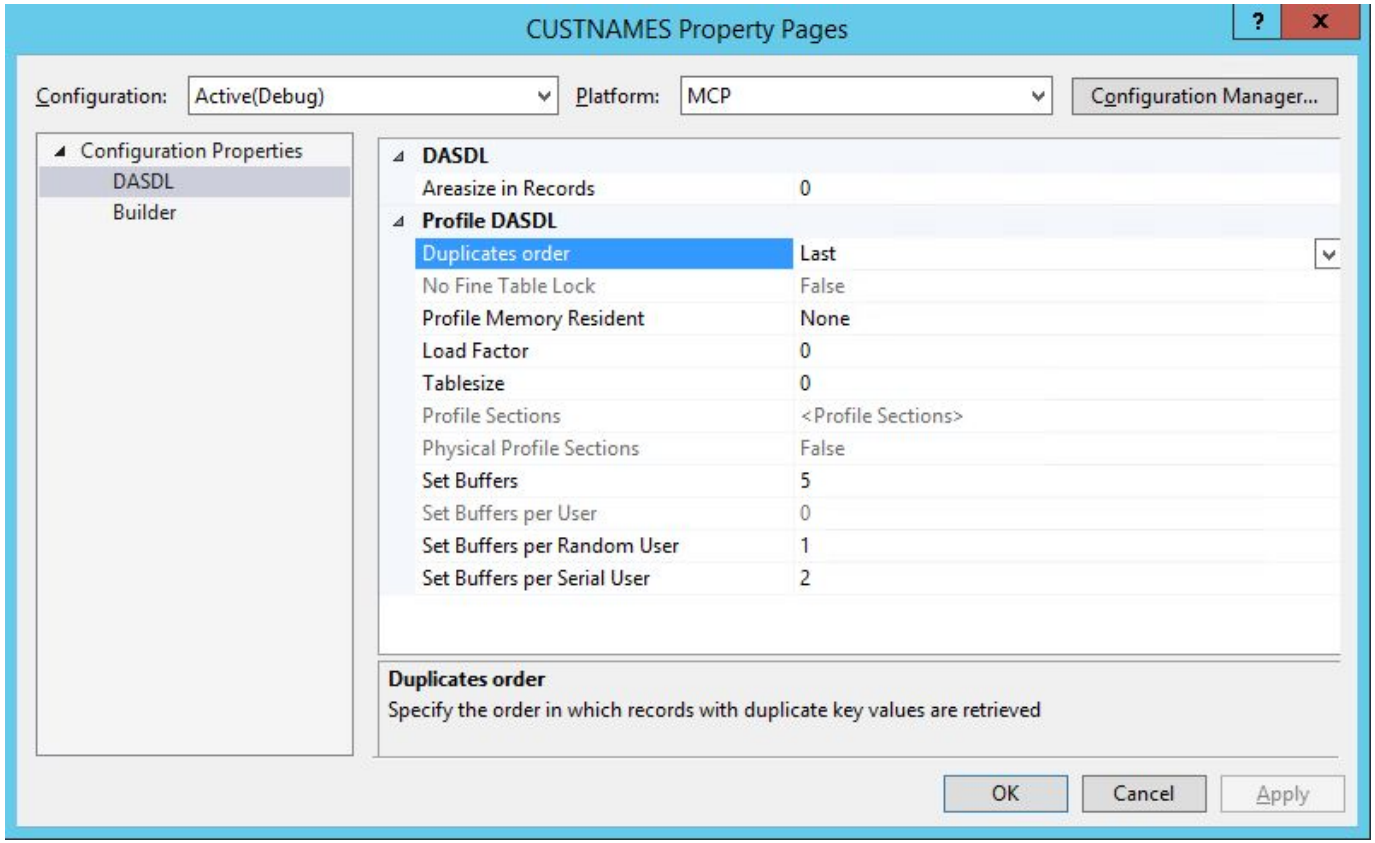
MCP

- When heavy processing is done while the ispec or report is in transaction state, it can have a significant impact to the overall performance of the system.
- Avoid extended ispec and report processing while in transaction state.
- Consider organizing the processes so that database updates are the last thing done in the ispec cycle.

Minimizing Deadlocks and Performance Degradation

MCP

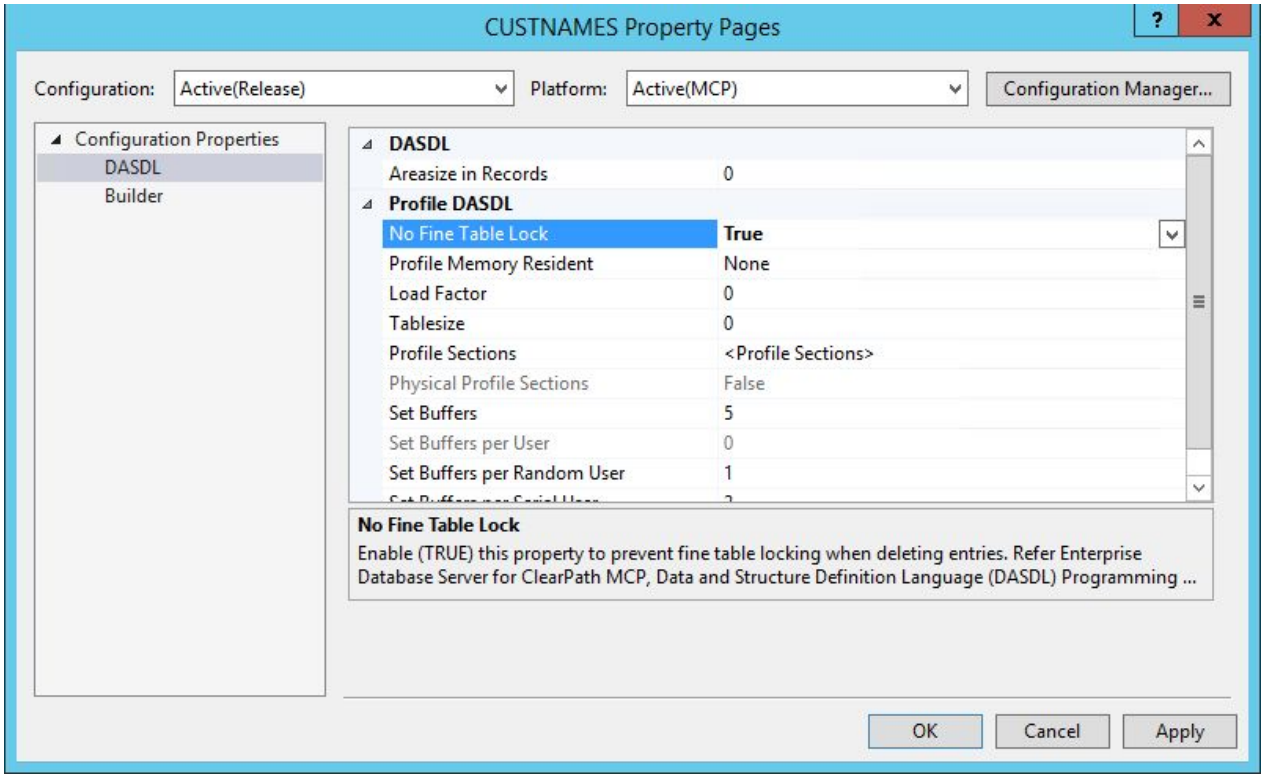
For profiles where *Duplicates Allowed* property is set to True and records are deleted while processing, consider setting the *Duplicates order* property of such profiles to Last or First



Minimizing Deadlocks and Performance Degradation

MCP

For very high activity datasets, where *Extended Edition* segment configuration property is set to True, the profiles have *Duplicates Allowed* property set to False, and records are deleted while processing, consider enabling *No Fine Table Lock* for the profiles.



Minimizing Deadlocks and Performance Degradation

MCP

- The KEYONLY command option increases the efficiency of LU and DT commands
  - Limits data retrieval to keys only for LU commands
  - Limits data retrieval to keys and attributes declared in the profile description for DT commands
  - Eliminates unnecessary I/O operations when other attributes are not required.
- You can use the KEYONLY command option in conjunction with the Multi and Serial command options. It cannot be used with the Secure option.
- You cannot use the KEYONLY command option if the segment's *Integrity* property set to true or if an individual ispec has its *Integrity* property set to true.

Minimizing Deadlocks and Performance Degradation

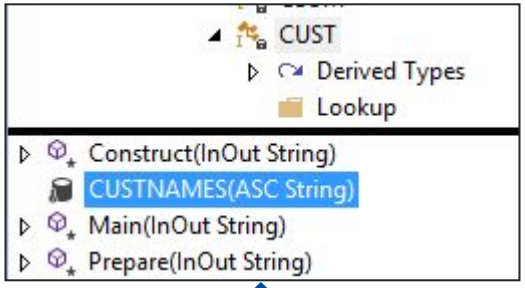
MCP

- The KEYONLY command option, when used with the Determine commands, retrieves data stored in keys and attributes declared in the profile description.
  - Attributes in a profile description are also known as **Profile Data** elements
- Using Profile Data elements eliminates unnecessary I/O operations when other attributes are not required.
- Trade off: A Profile Data element is physically stored in two places—in the profile as well as in its associated ispec class.

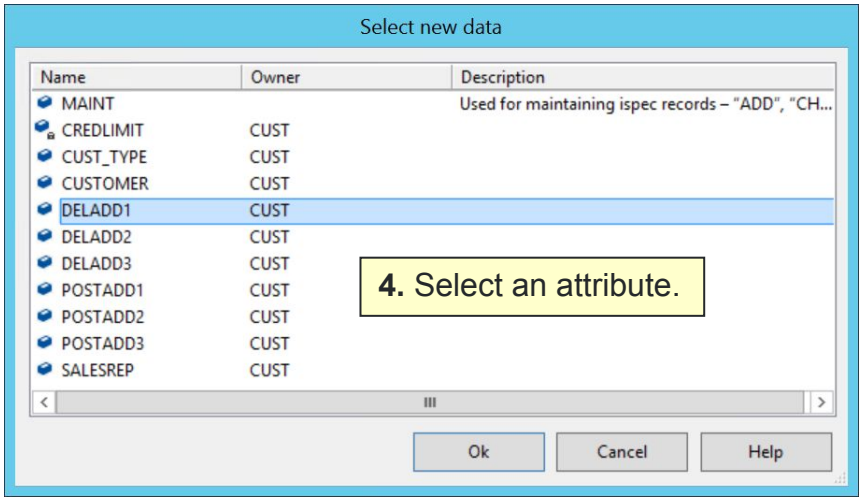
Minimizing Deadlocks and Performance Degradation

MCP

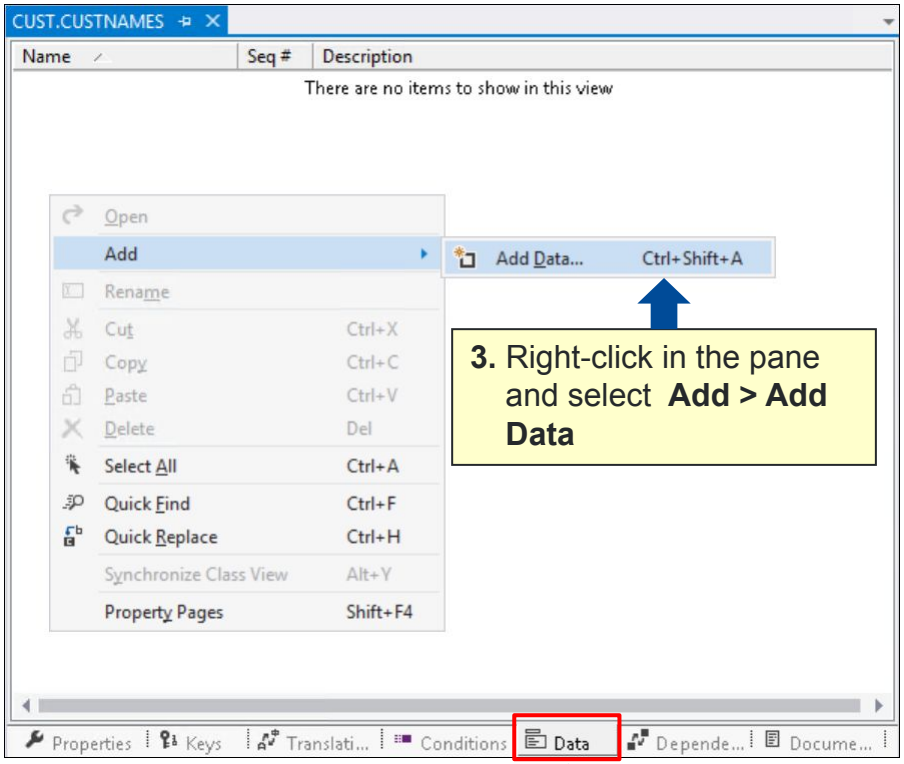
You can add profile data elements using the profile's Data tab.



1. Double-click profile.



4. Select an attribute.



3. Right-click in the pane and select **Add > Add Data**

2. Select Data tab.

Guidelines for Optimum Transaction Throughput

MCP

You can use the following guidelines for optimum transaction throughput on AB Suite MCP applications:

- Database Structure Sectioning
- Database Options/Settings
- COMS Settings
- AB Suite Developer Application Configuration

Guidelines for Optimum Transaction Throughput

MCP

Database Structure Sectioning - Identifying candidates

- Enable LOCKSTATISTICS via the Visible DBS (VDBS) command
- Obtain database statistics after significant amount of processing
- The ratio of the “Number of times lock held” versus “Number of lock waits” must be a non-trivial value
- The Total Lock Wait Time must be more than just a few seconds
- The DMSII Audit Trail may also benefit from larger Blocksize (maximum 1048575 segments), Buffers (maximum 255) and Areasize as well as Sectioning.

Guidelines for Optimum Transaction Throughput

MCP

Database Options/Settings

1. Set the *Extended Edition* property to True.
2. Set REAPPLYCOMPLETED/INDEPENDENTTRANS configuration properties to True.
3. Provide as much ALLOWEDCORE as practical for your environment.
4. Set all the database buffers to a value 64000 +1 or 2.
5. Set the SYNCPOINT property value to 4095.
6. Increase the CONTROLPOINT value for better performance.
7. Set the SYNCWAIT property value to 2.
8. Set OVERLAYGOAL property value to 0.001.
9. Set the Statistics property to false.
10. For database structures that are frequently processed serially (for example: batch processing) increase the REBLOCKFACTOR property value which, in conjunction with the DMSII REBLOCKFACTOR property, might provide better database record processing performance.

Guidelines for Optimum Transaction Throughput

MCP

- COMS Settings
  1. Set the Global “Input Queue Allowedcore” to a large value via Utility
  2. Set the Program “Input-Queue Memory Size” to a large value via Utility
- AB Suite Developer Application Configuration
  1. Set *Log Activities* and *Log Transactions* properties to False
  2. Set *Protected Input* property to False
  3. Disable POF by entering None in the *POF Name* property
  4. Set *Preserve Session Data* property to False

Using SLEEP and CRITICAL POINT

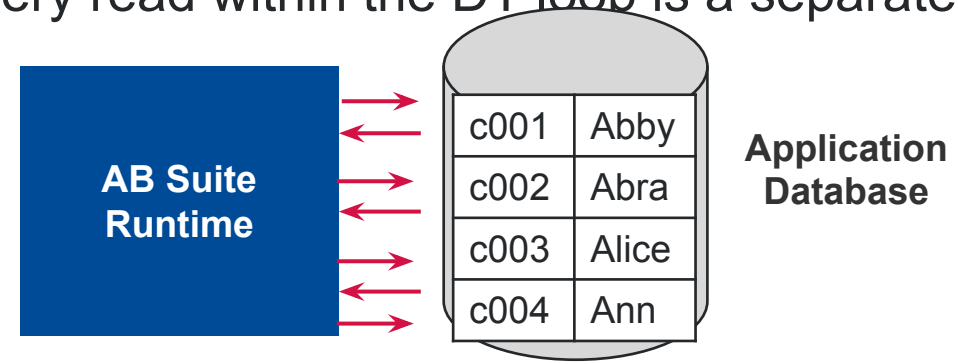
Using SLEEP and CRITICAL POINT (CP)

- Reports should be structured to avoid long database transaction states as this might impact other users of the database resulting in database structure deadlocks.
- Use Sleep 0 to end transaction without any forced delay time
- Balance the use of SLEEP keeping in mind:
  - Consistent (atomic) transactions
  - Not too frequent, since there is a small overhead in the database performing an end transaction operation
  - Not too infrequent , since it might result in overall performance degradation (due to records being locked for an extended period of time).
- Use CP for reports requiring restart capability
- SLEEP 0 is a better alternative if a report does not require restart functionality but there is a need to break up transaction states

Performance Optimization in Windows

- The default technique used by AB Suite Windows Runtime to read records from the database is **Server-Side Cursors**. With Cursors, the database retrieves one record at a time.
- In the following example, every read within the DT loop is a separate read on the database:

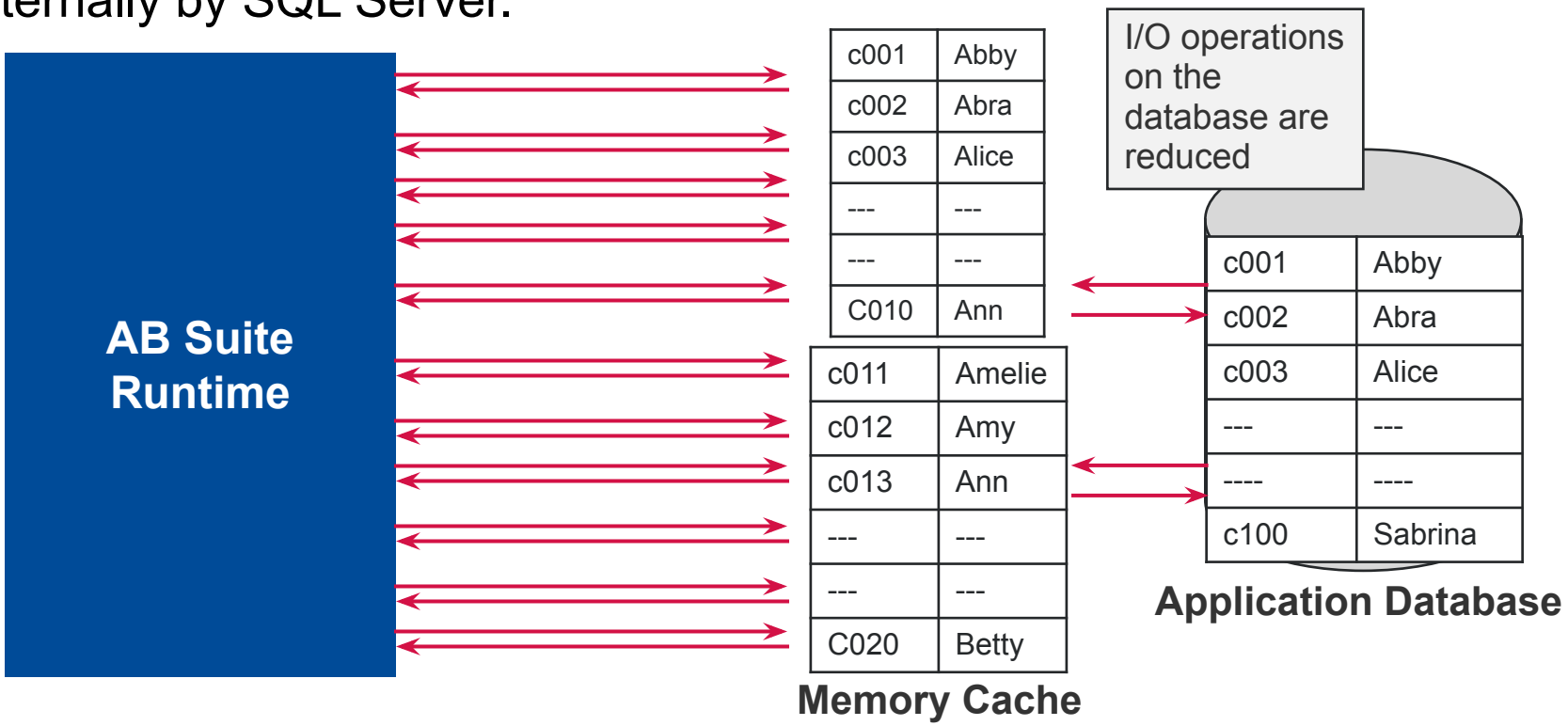
```
DT from CUST
   CUSTNAME:=CUST.CUSTNAME
   ---
   CUST.STORE()
   ----
End
```



- To achieve better Runtime performance you can use the following settings to change the default behavior:
  - Using DataReader
  - Using the MULTI behavior
  - Setting Transaction Isolation Level
  - Using Non-Phased SQL

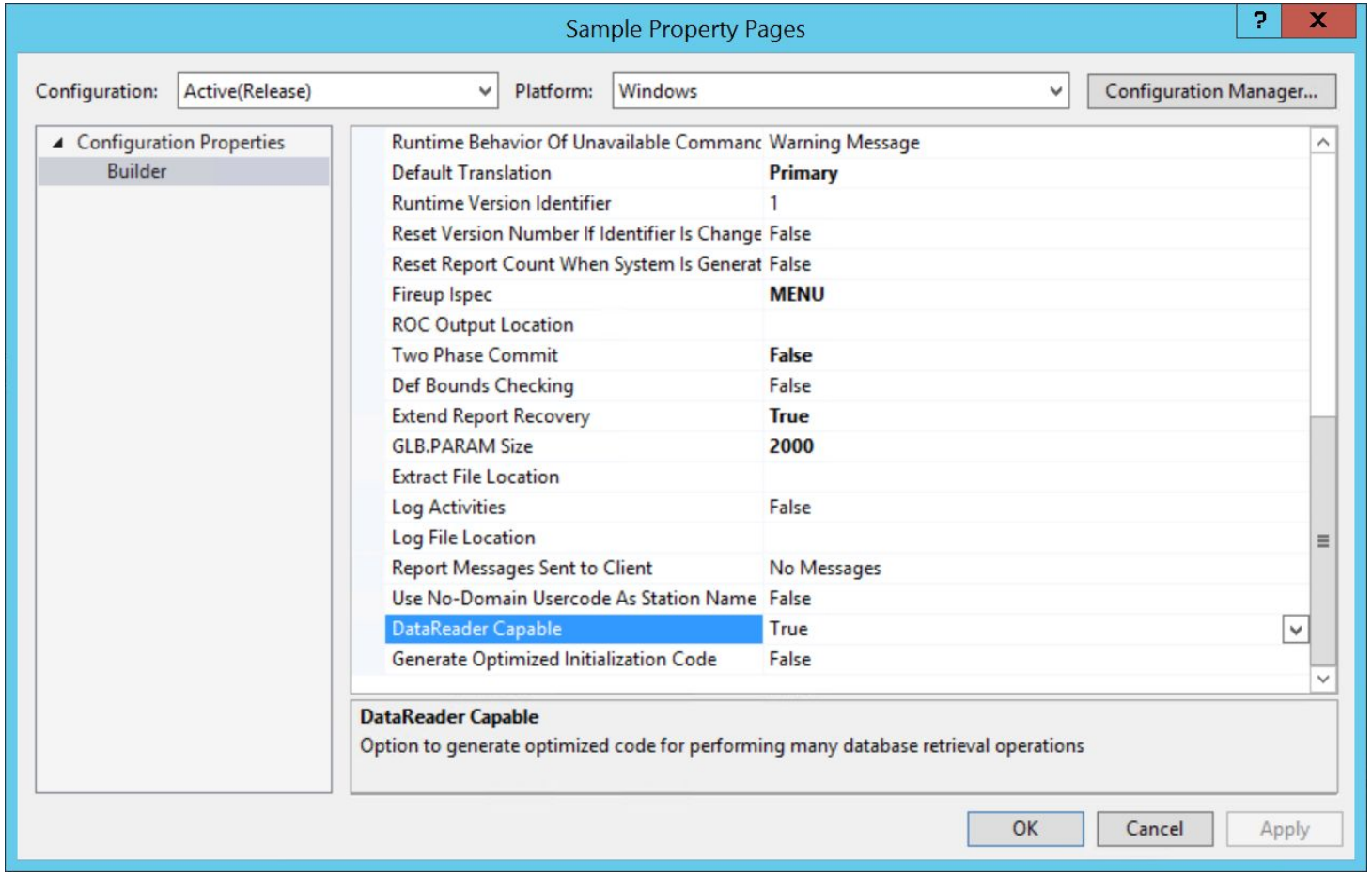
Using DataReader

Using DataReader, records are retrieved into an internal memory cache with many records being retrieved in blocks. As the Determine code requests the next record during each loop iteration, the record is returned from the memory cache rather than the physical database. This is handled internally by SQL Server.



Using DataReader

- By default the *DataReader Capable* configuration property is set to True.



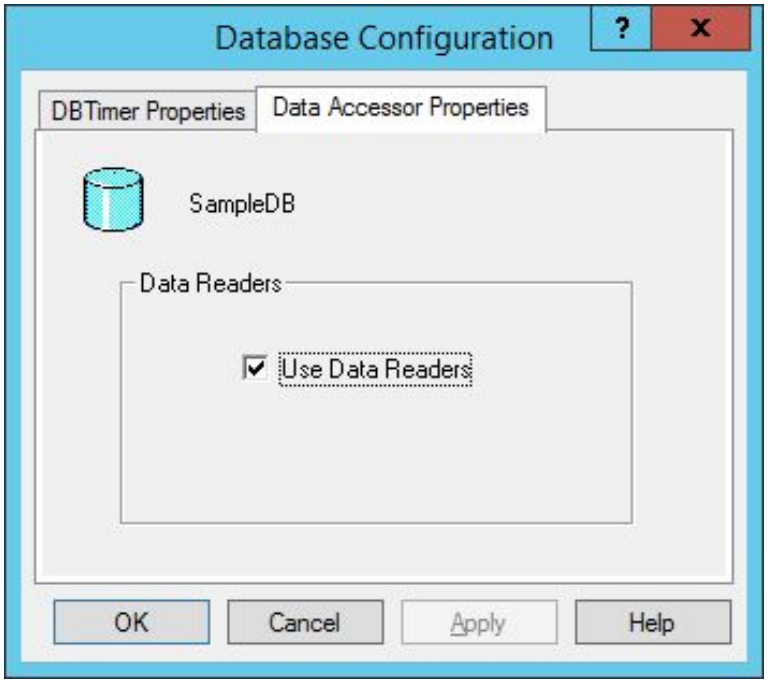
Using DataReader

- DataReader can cause some Determine command behavior differences compared to Cursors.
- The following table summarizes the difference between the cursor behavior and the way DataReader works:

Cursors Behavior	Using DataReader
– Cursors reads one record at a time, with the next record being retrieved from the database at the beginning of each iteration of a Determine loop. – If a table being read is updated during the read loop such that the content or set of records yet to be returned changes, then these changes will be visible to the later iterations. – For example, if a new record is stored in the table such that it is included on the Profile being read, then this record will be returned by later iterations of the read loop.	– With the DataReader, at the beginning of a Determine command, a separate thread will be started to read the complete result set of records into an intermediate memory cache. – This result set within the cache will not be updated by any changes to the actual records that happens during the read loop. – No synchronization between the data in the cache and the physical – In this situation, you can use “Use Database Reader” option to disable the DataReader for this part of the code.

Using DataReader

- Apart from setting the *DataReader Capable* configuration property, it is possible to enable or disable DataReader at Runtime for the system as a whole or for individual ispecs/reports.
- To enable the DataReader option:
  1. In the Administration tool, right-click the database node and select **All Tasks/Configure Database Parameters**
  2. Select the **Data Accessor Properties** tab.
  3. Select the **Use Data Readers** option



Using DataReader

Configure with XML file

- An XML configuration file called DBConfig.xml can be used to enable or disable DataReader for individual ispecs or reports.

  It is assumed that the system is generated with the *DataReader Capable* property set to true.

- The XML file needs to be created within the AB Suite **Data\Public** folder for example, C:\AB Suite 6.1\Data\Public

- In the <system> node the attribute DataReaderOnline is used to set a default DataReader setting for the online system i.e. all Ispecs. The attribute DataReaderReports is used to set a default DataReader setting for all reports

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
  <database name="SAMPLEDB">
    <system name="Sample" DataReaderOnline="false" DataReaderReports="false">
      <ispecs>
        <ispec name="CUST" datareader="true" />
        <ispec name="SALE" datareader="false" />
        <ispec name="INGDS" datareader="true" />
        <ispec name="SINQ" datareader="false" />
      </ispecs>
      <reports>
        <report name="CONSOLIDAT" datareader="false" />
        <report name="CUSTLIST" datareader="true" />
        <report name="PRODSTATS" datareader="true" />
      </reports>
    </system>
  </database>
</Configuration>
```

Windows

Using MULTI Behavior

- The MULTI setting can be defined within a configuration file called DBConfig.xml.
- Different internal mechanisms used:
  - DataReader cannot be used for a read loop that contains a database commit, in which case server-side cursors are used automatically
  - Conceptually, MULTI is similar to the way DataReader functions but comes into play in the case when the database is being read using server-side cursors
  - You can specify the number of records to retrieve in each block. Recommended values would be between 10 and 50. Based on experience you can arrive at an optimum value.
  - The granularity at which MULTI can be defined is at an ispec or a report
- Trade off: You will not see a significant difference with this option when there is a small number of records being read

Using MULTI Behavior

- With MULTI defined, when a read loop is first entered, a block containing multiple records is retrieved into an intermediate memory buffer. Then as the code iterates through the read loop the records are returned from the memory buffer.
- MULTI can be specified for the whole on-line system, all reports, or for individual ispec and reports.
- It is recommended to define MULTI for individual reports that already have DataReader defined.

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
  <database name="SAMPLEDB">
    <system name="Sample" DataReaderOnline="false" DataReaderReports="false">
      <ispecs>
        <ispec name="CUST" datareader="true" />
        <ispec name="SALE" datareader="false" />
        <ispec name="INGDS" datareader="true" />
        <ispec name="SINQ" datareader="false" />
      </ispecs>
      <reports>
        <report name="CONSOLIDAT" datareader="false" />
        <report name="CUSTLIST" datareader="true" Multi="20"/>
        <report name="PRODSTATS" datareader="true" Multi="20"/>
      </reports>
    </system>
  </database>
</Configuration>
```

Setting Transaction Isolation Level

- Transaction Isolation Level is a COM+ component setting that indicates how records are locked and impacts SQL Server database transactions.
- For example, the Transaction Isolation Level setting can be used to avoid the records being locked by SQL Server when a report reads through the records in an entire database table via a Determine loop.

Setting Transaction Isolation Level

- The Transaction Isolation Level setting can be defined within a configuration file called DBConfig.xml.
- The definition of Transaction Isolation Level is an extension of the DataReader settings.
- Transaction Isolation Level can be specified for the whole on-line system, all reports, or for individual ispec and reports

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
  <database name="SAMPLEDB">
    <system name="Sample" DataReaderOnline="false" DataReaderReports="false"
                IsolationlevelOnline="Repeatable Read"
                IsolationlevelReports="Repeatable Read">
      <ispecs>
        <ispec name="CUST" datareader="true" />
        <ispec name="SALE" datareader="false" />
        <ispec name="INGDS" datareader="true"
                                  isolationlevel="serializable" />
        <ispec name="SINQ" datareader="false" />
      </ispecs>
      <reports>
        <report name="CONSOLIDAT" datareader="false"
                                  isolationlevel="serializable" />
        <report name="CUSTLIST" datareader="true"/>
        <report name="PRODSTATS" datareader="true"
                                  Isolationlevel="READ UNCOMMITTED" />
      </reports>
    </system>
  </database>
</Configuration>
```

Using Non-Phased SQL

- During the execution of various Determine commands using multi-key profiles, SQL SELECT statements are constructed and submitted to SQL Server to retrieve the required set of records.
- **Phased SQL** and **Non-Phased SQL** are two different techniques of retrieving records from a database.
- The behavior of Phased SQL and Non-Phased SQL in terms of the records that are returned is identical.

Using Non-Phased SQL

The following table summarizes the difference between the Phased SQL and Non-Phased SQL:

Phased SQL	Non-Phased SQL
– The retrieval of records is broken up into a series of "phases" with one SELECT statement being submitted for each phase.	– The retrieval of records is *not* broken up into "phases"
– The number of phases (that is number of SELECT statements that will be submitted) to retrieve all records for the Determine command is equal to the number of **keys** on the Profile being used and the type of Determine command.	– A single SELECT statement is constructed to define the retrieval of all records.
– For example, with Determine From and if there are four keys on the Profile, then there will be four phases (that is four SELECT statements).	– For example, with Determine From and if there are four keys on the Profile, then there will be only one phase (that is only one SELECT statement).
– Multiple SELECT statements with a simple WHERE clause.	– A single SELECT statement with a complex WHERE clause.

Using Non-Phased SQL

- The Phased SQL setting can be defined within a configuration file called DBConfig.xml.
- Non-Phased SQL behavior can be specified for the whole on-line system, all reports, or for individual ispecs and reports.
- Note that long-running reports that have many nested Determine commands can show substantial improvement with Non-Phased SQL.

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
  <database name="SAMPLEDB">
    <system name="Sample" DataReaderOnline="false" DataReaderReports="false">
      <ispecs>
        <ispec name="CUST" datareader="true" />
        <ispec name="SALE" datareader="false" />
        <ispec name="INGDS" datareader="true" />
        <ispec name="SINQ" datareader="false" PhasedSQL="false"/>
      </ispecs>
      <reports>
        <report name="CONSOLIDAT" datareader="false" />
        <report name="CUSTLIST" datareader="true" PhasedSQL="false"/>
        <report name="PRODSTATS" datareader="true" PhasedSQL ="false"/>
      </reports>
    </system>
  </database>
</Configuration>
```

Using Non-Phased SQL

- Why use Non-Phased SQL?
  - To reduce the number round trips to SQL Server and back to the application.
  - In a profile with six keys, if the key values are such that all six phased SQL statements need to be executed to complete the required processing.
  - The population of the Ispec table or conditional profile is tens of thousands of records then the single Non-Phased SQL statement may be quicker.

- How to determine which SQL mechanism to use?
  - Consider the distribution of the record key values in the profile, and the type of processing of these records.
  - Consider the population of the ispec table or conditional profile and the number of the profile keys.

UNISYS | Securing Your Tomorrow™

Database Access Commands

Profile Structures

- Analyse the profiles defined in your model. More profiles result in additional database updates when records are inserted or modified.
  - **Duplicate profiles :** Profiles with extra keys might be redundant or duplicate the same information. For example, Profile1 with keys Customer, Name, and CredLimit is the same as Profile2 with keys Customer and Name. Ideally, between the two profiles, you can choose to use the profile with most keys. In this case choose Profile1 instead of Profile2.
  - **Profile with the same key but in different orders:** Profile1 with Customer as key in ascending order is essentially the same as Profile2 with Customer as key in descending order.
  - **One-off profiles:** A profile defined for just one report can be unnecessary. Alternatively, the report can retrieve the record in some other order, then extract, sort, and read the data. Depending on the priority of the report, one-off profiles can be avoided by making alternate design decisions.
  - **Conditional profiles:** Profiles with similar conditions can be avoided.
- Limit the number of profile structures in your model. More profiles result in a lot of database updates when records are inserted or modified.

Best Practices – Read and Write Operations

- Avoid excessive reads and writes
- Use a Group clause with a LU or DT clause to reduce the range of records selected on a read. For example:

```
determine group Event.INVENTORY (PRODUCT) from (GLB.ZEROS, GLB.SPACES, GLB.ZEROS)
        add Event.QUANTITY STOCKBAL
end
```

- Break out of loops (Determine, LookUp, ForEach) at the appropriate points.
- Do not use a profile if the order of retrieval is not important – when you use the command `Determine Actual CUST`, there is no need to use a profile

Best Practices – Read and Write Operations

As you read a record into multiple instances of a class, avoid excessive read and writes into the database. For example, in the following method another read into CUST at a later point in the logic is not needed.

```
Checkout (P001. CCNo. CUSTID)
DT from PROD (p001) : Select product
      PROD1 :=  PROD
end
DT from CUST (CUSTID) :
      CUST1:= CUST
end
if CheckCredLIM(CCNo) >= PROD1.PRICE
      GENERATEBILL()
else
      --------    :Additional logic here
      --------    :Additional logic here
      --------    :Additional logic here
      DT from CUST (CUSTID) : This read is not needed again
      message " Card number +CUST.Ccno has invalid limit"
end
```

Best Practices – Read and Write Operations

- Use instances of a class definition represented by an attribute to read records into, within a DT Loop.
- For example:

```
  :Browse CUST for a particular customer
determine from CUST1.CustNames ("Smith")
        break
end
determine from CUST2.CustNames ("Jones")
        break
end

  :If frequency of visit is higher then set loyalty group to A else B
if CUST1.Frequency > CUST2.Frequency
        CUST1.Loyalty := A
        CUST2.Loyalty := B
else
        CUST2.Loyalty := A
        CUST1.Loyalty := B
end
```

Best Practices – Read and Write Operations

- You can use the ForEach Polymorphic database command to retrieve records for multiple object types.
- For example, a method to sort goods, based on QOH and the SellBy date and then decide whether to keep the stock or discard, using ForEach you can use just one query to retrieve even the inherited structures.

Prod ID	ProdName	Prod Price	QOH	SellBy	Size
P001	Product_App_A	75	30		5
P002	Product_Peri_Type1	30	100	03/09/2013	
P005	Product_App_B	72	100		10
P012	Product_Line2	150	40		
P031	Product_Peri_Type3	100	50	03/08/2012	

Best Practices – Read and Write Operations

```
foreach Product1 in Product Polymorphic
    if Product.QOH > 100
        if Product.SellBy < CurrentDate()
            KeepProduct() : Generate report for products to stay in stock
        else
            DiscardProduct() : Generate report for products to discard
                ------- : Do some task
                ------- : Do some task
        end
    end
end
```

Summary

In this module, you have learned about the:

- Best practices and guidelines that can help you maintain your model better and achieve better runtime performance