# SoftwareMining

## VSAM Handling



**Intended Audience:**

- Translation resource (responsible for preparing and performing translation)
- Java / C# Programmers and Architects

Version: 2.9



## Overview

During the translation process, all VSAM File Definitions will be translated to SQL. The translated Java application uses a Relational Database for its data not VSAM file or operating system files.

## Contents

# 1 Overview: Translating VSAM File Definitions.

VSAM structures are defined in COBOL File-Definitions. The definition contains column names, types and sizes.

SoftwareMining CORECT uses these File Definitions to produce the following artifacts from VSAM File Definitions:

- XML Representation of the File Definition
- Java Data Access layer.
- SQL data schemas (Oracle, DB2, MySQL). (For KSDS/Indexed Files).

## 1.1 XML Representation of the File Definition

VSAM FD structures are first translated to an XML file containing same structural information.

The XML file is used internally to generate SQL Data-Schemas and the Java Data-Access layers. This allows some configuration parameters to be changed, and the SQL/Java code to be regenerated.

## 1.2 Java /C# - Persistent Data Access Objects

One of the characteristics of the Java/C# Data-Access layer is its ability to work with COBOL like data-structures.

It is these classes that handle COBOLs REDEFINES, COMP values, as well as the providing equivalent of COBOL "Group" access methods.

The can also write the data in each record to a character-string, or more importantly – it can **read** fixed length character string from a **COBOL Line-SEQUENTIAL file** and correctly **populate the underlying SQL tables**.

The generated Java /C# Persistent Data-Access Objects and SQL schemas form the basis of SoftwareMining Data Migration strategy.

## 1.3 SQL Schemas

During the translation, the hierarchical structures of COBOL File-Definitions will be flattened out, and the REDEFINES and COMP values will be taken out and a simple SQL data schemas generated.

# 2 VSAM Sequential Files

## 2.1 Record Sequential vs Line Sequential

Line-Sequential / Variable length files: COBOL's (and the translated system) will use a CR-LF at end of each record. This eases identification of each record easier. They are specially useful when the structure may be of varied length (ie containing OCCURS-DEPENDING).

Record Sequential / Fixed length files: COBOL's (and the translated system) concatenates the records defined in this format. Hence, for a record defined to contain 100 characters, a data file containing 5 records will have 500 characters with no CR-LF used as a delimiter. Record-Sequential files are especially useful if/when the data itself to contain CR-LF sequences – which may happen in representation of Packed-Decimal fields.

The (translated) Cobol code often specified if a records is in Fixed or Variable length, and the translator would mark the Data-Access-Object appropriately.

If/when the information is absent from the translated code – the system will default to using VARIABLE length (line sequential) files.

This default can be overwritten using the following configuration:

C# / SoftwareMining.config:

```
<corectSettings>
  <add key="SEQUENTIAL_RECORDS_ARE_VARIABLE_LENGTH" value="false" />
```

Java/ softwaremining.properties (SoftwareMining.config in C#):

```
SEQUENTIAL_RECORDS_ARE_VARIABLE_LENGTH=false
```

## 2.2 What happens to Packed-Decimal fields in Sequential Files

These are treated in same manner as COBOL.

Note the representation of some packed-decimal values may contain Carriage-Return characters. Such datafiles cannot use CR-LF as record delimiters, and have to be marked as RECORD-SEQUENTIAL to allow the system to read write using "record length" information instead of CR-LF delimiters. This can be achieved by setting the default and/or using the API described in the above section.

Also, at a later stage (after testing phase) – the individual sequential files can be configured to read/write from Comma-Separated-Values (CSV) file format. This obviously can significantly improve the integration prospects for application.

In such cases, the numeric value of the COMP fields will be written out.

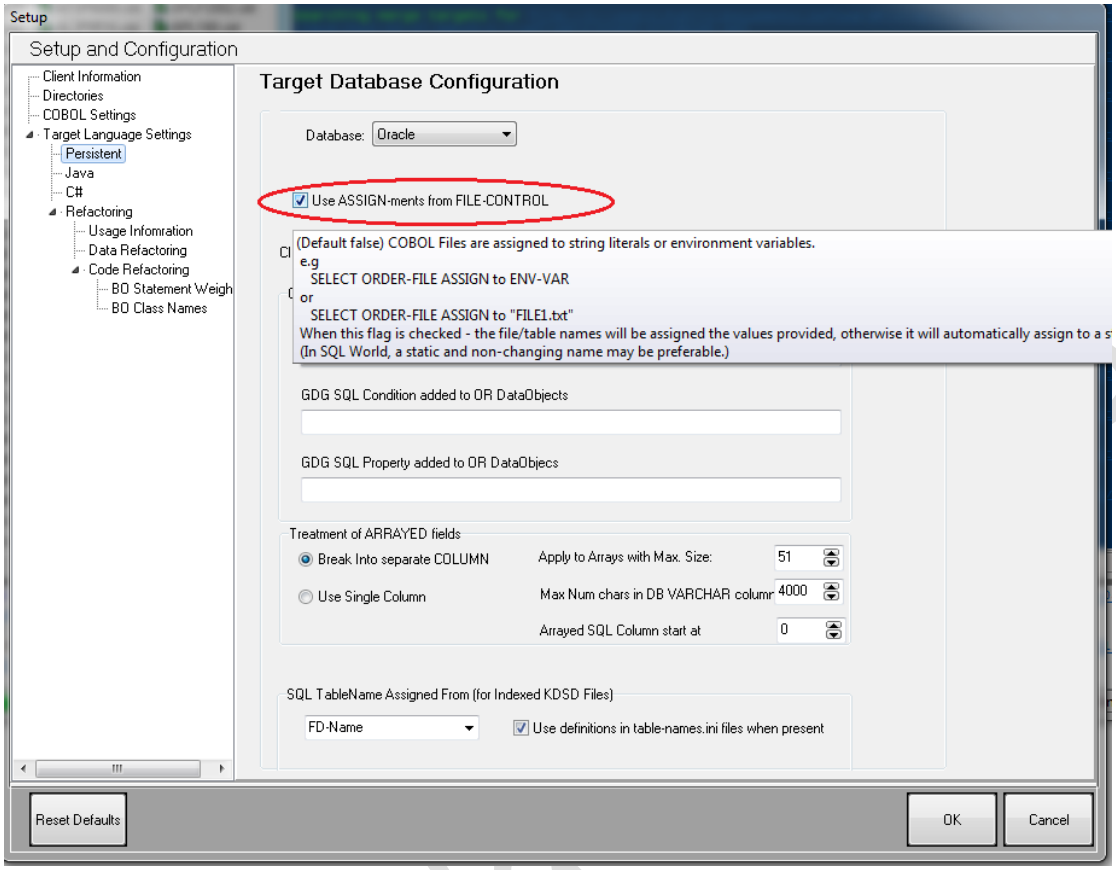
## 2.3 Assigning File-names from environment variables.

In many Cobol applications, the name of a data-file is passed JCL via to the application FILE-CONTROL section.

```
SELECT Client-Record assign to filename-variable ….
```

The translation can use a similar approach – the environment variable filename-variable must be created and assigned the appropriate value in a shell-scripts replacing the JCL. The program will then use look for this environment-variable.

This is configured in the following screen, and applies to all VSAM files – indexed and sequential.




## 2.4 Migrating Sequential Files to Comma-Separated-Values

Changing the super-class of such classes from SequentialFile to AbstractCSVFile will generate the file in new format. The main benefit if CSV format is:

- packed-decimal fields: CSV file will contain the field-value instead of binary representation of the field-value.
- Isolation/Separation of every fields (in Sequential format – everything is joined together)
- Integration with other applications, e.g. Microsoft Excel

# 3 VSAM KSDS – Indexed files

## 3.1 Different FD structures accessing the same data files.

Unlike an SQL Table, COBOL data-files do not contain any meta-data or column information. This allows a data-file to be accessed by different structures defined in programs. Once we move to SQL – the structure of the table has to be defined, and cannot be changed. For example, a table with columns

```
FIRST-NAME VARCHAR(20),
LAST-NAME VARCHAR(20)
```

Cannot dynamically change to contain information

```
NAME VARCHAR (40)
```

Where possible the translator will use a mixture of record-size & structural equivalence to determine whether two structures can be merged/can map to the same database table.

In doing this, the following issues need further consideration:

- Organizing Translation Order.
- Accessing same table with different DAO's (see use of DELEGATE classes)
- Informing translator the structures point to same table (see use of table-names.ini)

### 3.1.1 Organizing Translation Order (generating CICSVSAM.cbl) for CICS programs

CICS READ/WRITE statements refer to DATASETS. Often data-sets with different names refer to the same data (SQL table). The aim of the section is to create a single COBOL program (CICSVSAM.cbl) containing the primary structures to be used for generation of the SQL tables and the associated Data-Access-Objects.

The translation of this file will generate SQL and the primary DAO's (to be used as *delegates*). All other programs should either use these directly or use structures which Delegate the SQL read/write to these.

For example, lets say the project only contains 2 type of data-file (CLIENT and ITEM), each defined in 2 different type of structure. Hence the project contains the following copybooks:

CLIENT-DETAILED.CPY:

```
       01 CLIENT-RECORD.
         05 FIRST-NAME PIC X(20).
         05 LAST-NAME  PIC X(20).
```

CLIENT-SUMMARY.CPY:

```
       01 CLIENT-RECORD.
```

```
    05 NAME PIC X(30).
    05 FILLER PIC X(10).
```

ITEM-DETAILED.CPY

```
01 ITEM-RECORD.
  05 ITEM-ID    PIC X(20).
  05 ITEM-NAME  PIC X(20).
  05 ITEM-DESC  PIC X(20).
```

ITEM- SUMMARY.CPY

```
01 ITEM-RECORD.
  05 FILLER   PIC X(60).
```

Lets say the project comprises of 2 programs :

programA references:

CLIENT-DETAILED.cpy

ITEM-SUMMARY.cpy

programB

CLIENT-SUMMARY.cpy

ITEM-DETAILED.cpy

The solution is to manually create a new program referencing the DETAILED copy books, and then to translate this new program first. The translator will then be aware of the DETAILED versions for CLIENT and ITEM, and can automatically switch to using the DETAILED information when translating programs A and B.

The following are template for the CICSVSAM.cbl programs

### *CICS VSAM Template*

```
IDENTIFICATION DIVISION.
PROGRAM-ID.    CICSVSAM.
ENVIRONMENT DIVISION.
DATA DIVISION.
```

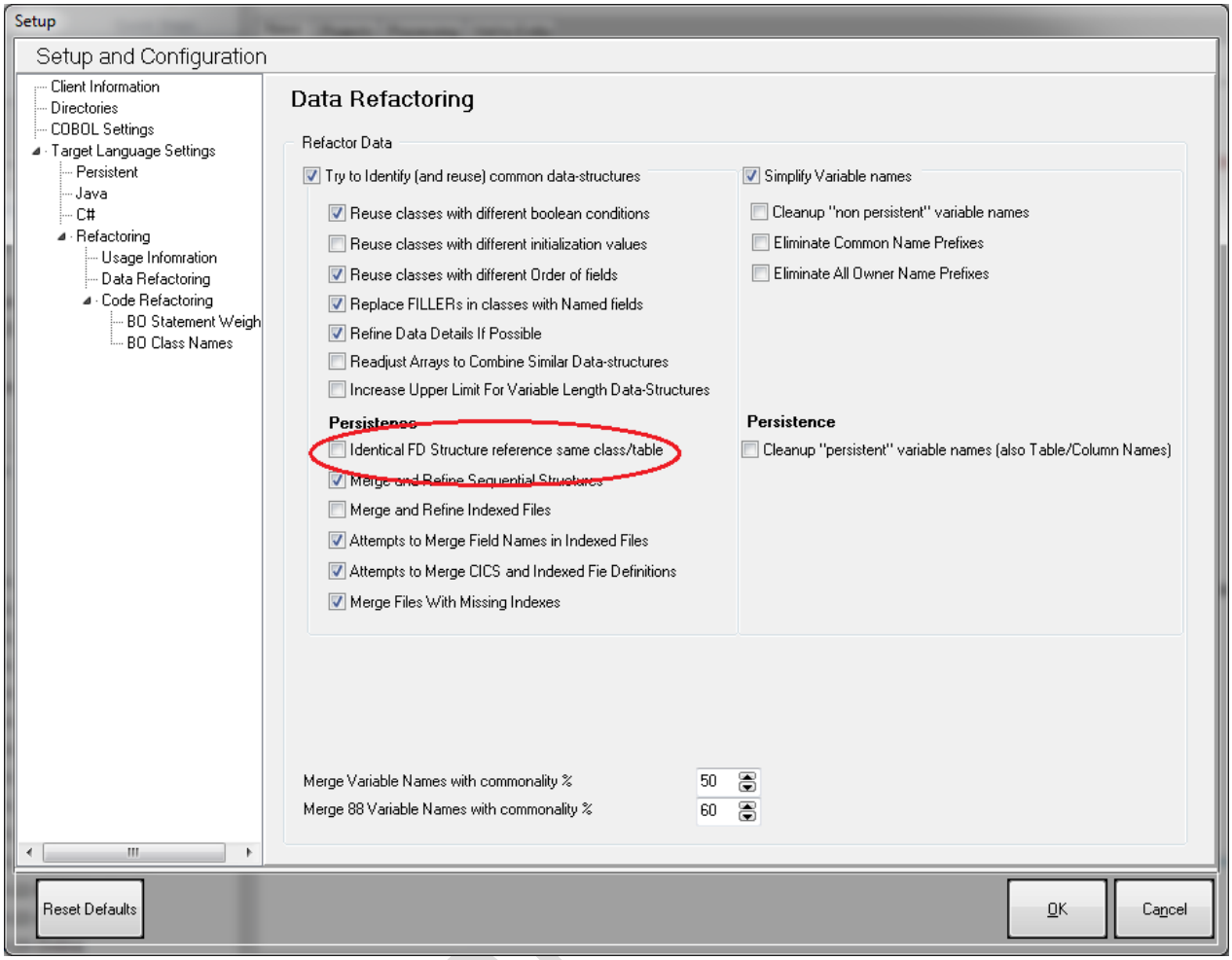
```
WORKING-STORAGE SECTION.

01  WS-FIELDS.
    05  WS-RECORD-KEY              PIC X.
    05  WS-RESPONSE                PIC X.
    05  WS-LENGTH                  PIC 9.

 COPY   CLIENT-DETAILED.
 COPY   ITEM-DETAILED.


PROCEDURE DIVISION.
0000-MAIN.

    EXEC CICS READ
              DATASET('CLIENT')
              INTO(CLIENT-RECORD)
              RIDFLD(WS-RECORD-KEY)
              LENGTH(WS-LENGTH)
              RESP(WS-RESPONSE)
    END-EXEC.

    EXEC CICS READ
              DATASET('ITEM')
              INTO(ITEM-RECORD)
              RIDFLD(WS-RECORD-KEY)
              LENGTH(WS-LENGTH)
              RESP(WS-RESPONSE)
    END-EXEC.
```

***Question: Where are there fewer SQL table generated than the number of FD's in CICSVSAM.cbl?***

This by caused by system assuming two IDENTICAL (names & structures) File-Definitions are the same one.

In most cases, this is a benefit. For example the same structure (lets call it CLIENT-RECORD) in 2 different files may be assigned to different files – but if system sees they are identical – then it knows these things should be referencing the same database table.

Also make sure the following is un-ticked for the *CICSVSAM.cbl* file – you would probably want to have it checked when processing other programs.


Setup
Setup and Configuration
Client Information
Directories
COBOL Settings
Target Language Settings
Persistent
Java
C#
Refactoring
Usage Infomration
Data Refactoring
Code Refactoring
BO Statement Weigh
BO Class Names
Data Refactoring
Refactor Data
Try to Identify (and reuse) common data-structures
Reuse classes with different boolean conditions
Reuse classes with different initialization values
Reuse classes with different Order of fields
Replace FILLERs in classes with Named fields
Refine Data Details If Possible
Readjust Arrays to Combine Similar Data-structures
Increase Upper Limit For Variable Length Data-Structures
Persistence
Identical FD Structure reference same class/table
Merge and Refine Sequential Structures
Merge and Refine Indexed Files
Attempts to Merge Field Names in Indexed Files
Attempts to Merge CICS and Indexed Fie Definitions
Merge Files With Missing Indexes
Simplify Variable names
Cleanup "non persistent" variable names
Eliminate Common Name Prefixes
Eliminate All Owner Name Prefixes
Persistence
Cleanup "persistent" variable names (also Table/Column Names)
Merge Variable Names with commonality %
50
Merge 88 Variable Names with commonality %
60
Reset Defaults
OK
Cancel

### 3.1.2 Use of Table-Names.ini (Cross Reference Table) in VSAM Files

The aim of this file is to help the translator in determining which delegates to use.

For example consider the following program:

```
IDENTIFICATION DIVISION.
    PROGRAM-ID. PROGRAM-1.
   …
    FILE-CONTROL.
        SELECT  CLIENT-FILE
         ASSIGN TO CLNT
         ORGANIZATION IS INDEXED.

    DATA DIVISION.
    FILE SECTION.
     FD CLIENT-FILE.
        01 CLIENT-REC.

          05 FIRST-NAME PIC X(20).

          05 LAST-NAME  PIC X(20).

  …
```

```
IDENTIFICATION DIVISION.
    PROGRAM-ID. PROGRAM-2.
   …
    FILE-CONTROL.
        SELECT  CLIENT-FILE2
         ASSIGN TO CLNT-2
         ORGANIZATION IS INDEXED.

    DATA DIVISION.
    FILE SECTION.
     FD CLIENT-FILE2.
        01 CLIENT-RECORD.

         05 ID PIC X(30).

  …
```

Assuming **CLNT** and **CLNT-2** refer to the same data-set (client data), then both programs will need to access the same SQL table. However, since the system will be unable to detect the two files refer to the same table as data formats is different in each program (ie:

Program-1 defines:

```
        01 CLIENT-REC.

          05 FIRST-NAME PIC X(20).

          05 LAST-NAME  PIC X(20).
```

And program-2 defines a totally different structure.

```
01 CLIENT-RECORD.
  05 ID PIC X(30).
```

The translation of the two programs will result in generation of the following two data-access-objects (DAO):

- ClientFile(for program1) : Accessing Table **CLNT**
- ClientFile**2**(for program2) : Accessing Table **CLNT-2**

You can tell the translator to treat the two files as the same table by the following:

- Create a file called "table-names.ini" in the [SoftwareMining-installation-dir]\bin
- Please the following information in this file:

```
[PROGRAM-1]
CLIENT-FILE=CLIENT-SQL-TABLE

[PROGRAM-2]
CLIENT-FILE-2=CLIENT-SQL-TABLE
```

With the introduction of table-names.ini, the translation of the two programs will result in generation of the following two data-access-objects (DAO):

- ClientFile *DAO* (for program1) : Accessing Table CLIENT-SQL-TABLE
- ClientFile**2**DAO(for program2) : **delegating** the access to *ClientFile DAO* – which will read the data from CLIENT-SQL-TABLE.

In this case, the following two classes will be generated:

```
com.companyxxx.applicationxxx.persistent.ClientFile
```

and

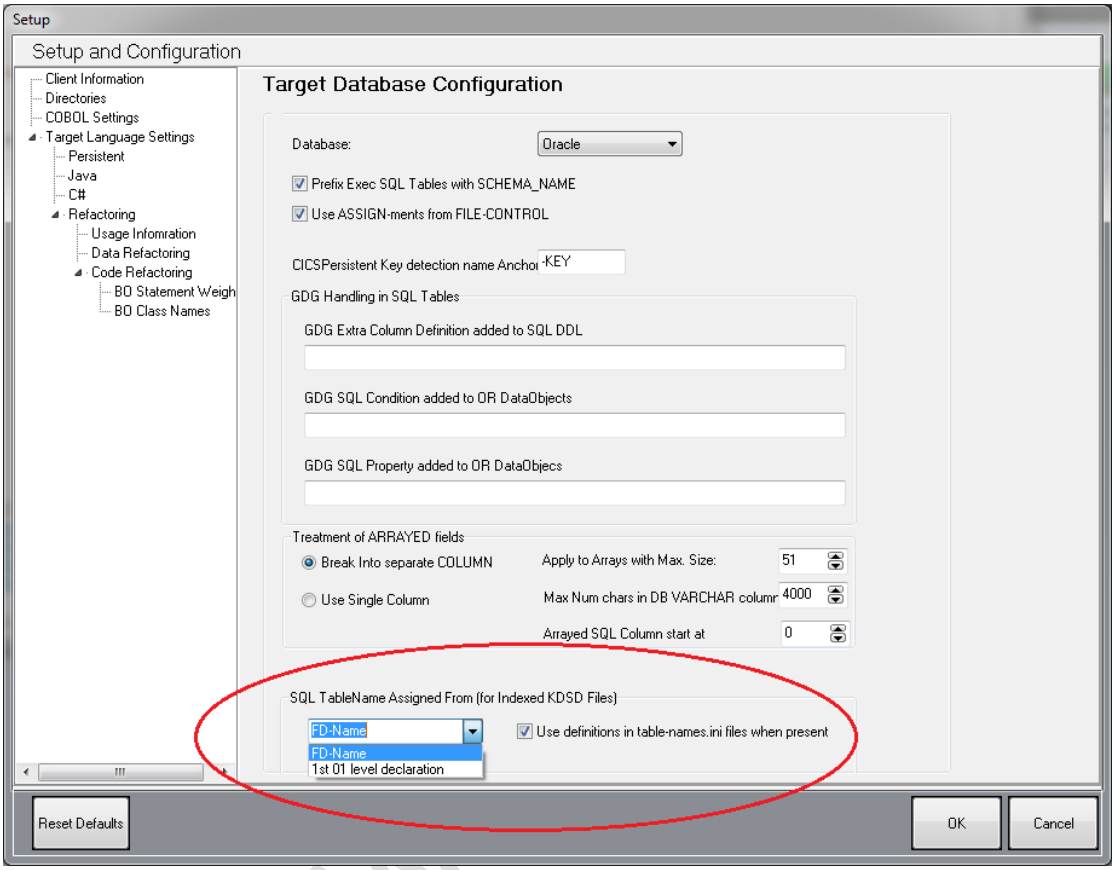
```
com.companyxxx.applicationxxx.persistent.program2.ClientFile2
```

The constructor for program2. ClientFile2will ***delegate*** the database access to “ClientFile” class by a statement such as:

```
public ClientFile2() {
  super(owner);
  assignDelegateDAO(new ClientFile(owner));
   ...
}
```

This will cause all calls to clientFile**2** CRUD calls to be redirected to clientFile, and the data then transferred from clientFile to clientFile**2** (similar to a MOVE statement).

The following flag can be used for telling the system table-names.ini uses File-Definition or 01-level name for each record:



### 3.1.3 Use of Table-Names.ini (Cross Reference Table) in CICS Applications

The aim of this file is to help the translator in determining which structures represent the same files.

Lets say the CICS program-3 contains:

Prog-3:

```
EXEC CICS READ
          DATASET('CLIENT')
          INTO(CLIENT-RECORD)
          RIDFLD(WS-RECORD-KEY)
          LENGTH(WS-LENGTH)
          RESP(WS-RESPONSE)
END-EXEC.
```

But CICS programs 3 and 4 contain :

Prog-4:

```
EXEC CICS READ
          DATASET('CLIENT-4')
          INTO(CLINT-4)
          RIDFLD(WS-RECORD-KEY)
          LENGTH(WS-LENGTH)
          RESP(WS-RESPONSE)
END-EXEC.
```

Prog-5:

```
EXEC CICS READ
          DATASET('CLIENT-5')
          INTO(WS-CLIENT)
          RIDFLD(WS-RECORD-KEY)
          LENGTH(WS-LENGTH)
          RESP(WS-RESPONSE)
END-EXEC.
```

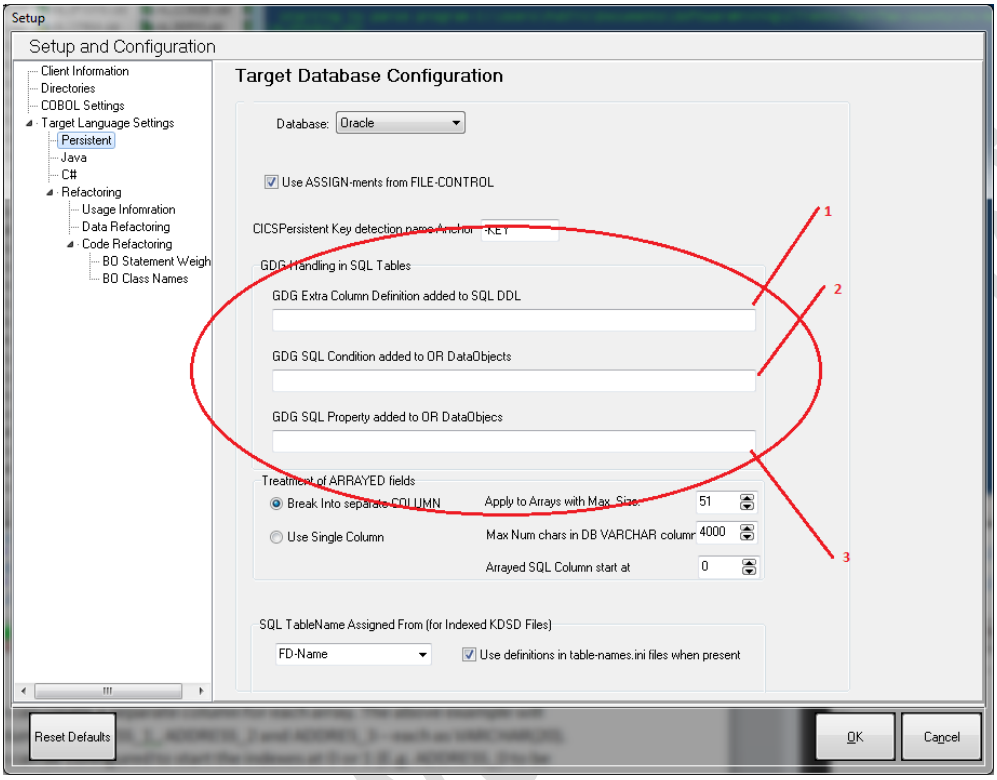
Assuming the data-sets **'CLIENT'** , **'CLIENT-4'** and **'CLIENT-5'** correspond to the same target SQL table, then the translator will need the following in table-names.ini file:

```
[PROGRAM-3]
CLIENT=CLIENT-SQL-TABLE

[PROGRAM-4]
CLIENT-4=CLIENT-SQL-TABLE

[PROGRAM-5]
CLIENT-5=CLIENT-SQL-TABLE
```

## 3.2 GDG handling in migrated KSDS to SQL

**Q: How GDG Support for KSDS Files:**

The system can incorporate an extra column to represent GDG within the tables/Classes which will require GDG handling. However, the translator will have to be told about the tables requiring extra columns.




In the setup window :

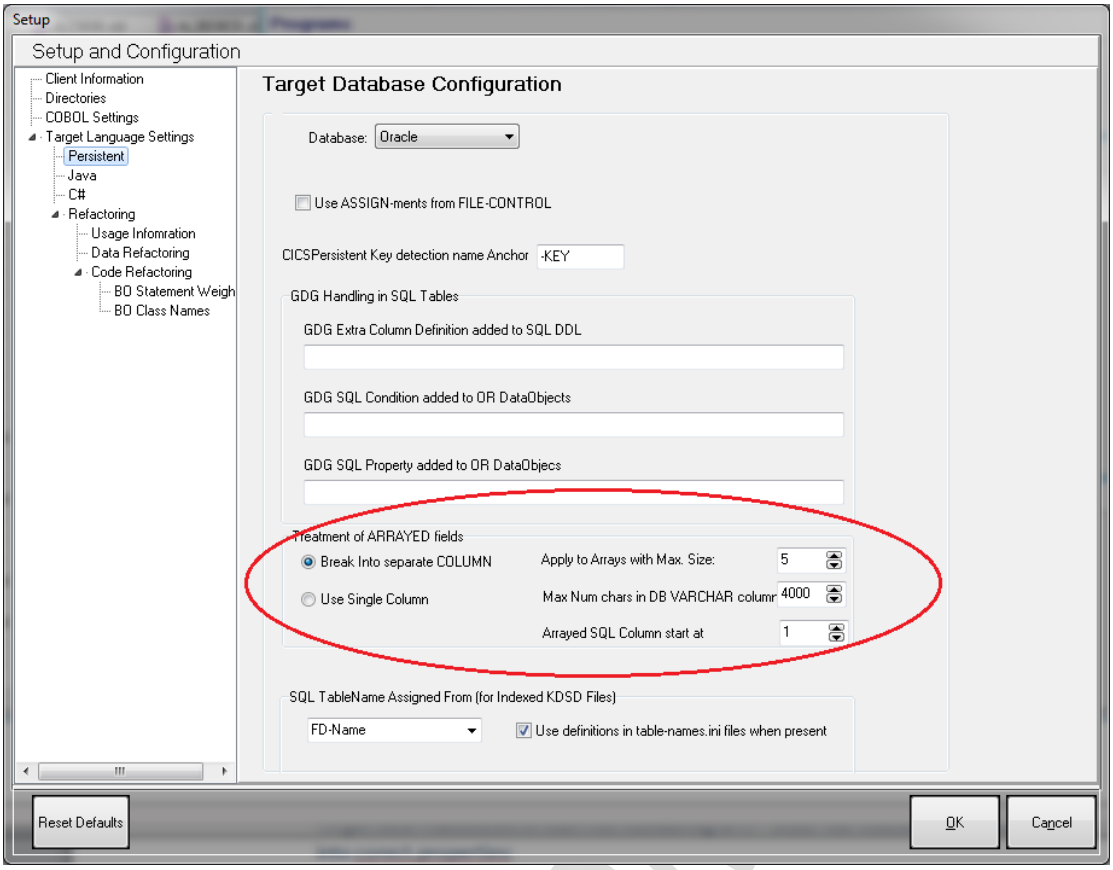
1. Field-1 facilitates defining a new COLUMN for each database table
2. Field-2 Allows adding SQL condition into each Object-Relation DAO
3. Field-3 Allows adding parameter to the Object-Relation DAO files. For example, define a parameter GDG-1, and then insert the following into gdg.properties (which needs to be accessible at runtime by the application):

   GDG1=(YEAR=2015)

## 3.3 How are VSAM Array's translated to SQL

A KSDS File-Definition may contain arrayed fields. The system can provide different mechanisms of migrating the arrayed fields into a SQL tables.

The following setup screen caters for the different configurations.




For example, consider the definition:

```
05 ADDRESS PIC X(20) .
```

- Option-1

  The system can create a separate column for each array. The above example will create 3 columns: ADDRESS_1 , ADDRESS_2 and ADDRES_3 – each as VARCHAR(20). The system can be configured to start the indexes at 0 or 1 (E.g. ADDRESS_0 to be the first column).

  *To generate SQL whose column names start at 0 - enter the following line under* `[Database]` *section of corect.ini (We will expose this in the setup screen in next build).*

  ```
  Arrayed SQL Column start at=0
  ```

  To get Java-framework to start the numbering at 0 – enter the following property into softwaremining.properties for Java translations, (SoftwareMining.config in C#):

  ```
  OR_ARRAY_INDEX_BASE=0
  ```

  - Option-2 The approach to handling arrayed items (OCCURS ) is to generate a single CHAR or BLOB/CLOB/ (IMAGE in SQLServer). The above example will result in a single column ADDRESS CHAR(60).

## 3.4 VSAM KSDS - FAQ

**Wild-Cards in SQL:** *Will the system use wild-cards in dynamically generated SQL?*

When converted code make a call to Database, it will use explicit column names instead of wildcards (SELECT * FROM ..) .

**Data-Types:** *Does Java code used for OLTP and/or batch processing convert data queried from Oracle to COBOL data type? For example number (9,2) to comp-3 to be presented, displayed or passed to other programs on Unix/Java environment.*

The column will be defined as NUMBER in Oracle. The necessary conversion to COMP will be done by SoftwareMining framework.

**Logging:** *How do I see the SQL generated for Object-Relational classes.*

These are dynamically generated at runtime. To see the generated SQL at runtime – please add the following from log4j.property file.

```
log4j.logger.com.softwaremining.sql=TRACE
```

**Object-Relational Mapping:** *How is the libraries pick up information for the SQL generation?*

The libraries generate the SQL at runtime the information provided by each class in "com.company.app.persistent" package

The TABLE-NAME is taken from the top-most group definition. The translator may over-ride this using the statement:

```
persistentClass.assignTableName("YYYY");
```

The Field-names are taken from names associated with each Field.

**Field/Column sizes.** *Does conversion tool have any record limitation that Oracle record must match the same order and fixed size of COBOL copybook? For example, if copybook record size is fixed 100, then does the data saved and retrieved from Oracle have to be 100 fixed as well?*

For character type field, the answer is yes. If there is a requirement to change the size, then the size in database as well as the Java class has to change.

For numeric fields, are different – for example the oracle's INTEGER datatype will be used for representation of PIC 9 and PIC 9(4)

**Redefines:** *What happens to redefined fields?*

REDEFINEd fields do not end up as columns in database, as their inclusion would introduce normalization issues.

When a redefined field has to appear in the database, the solution would be to reverse the REDEFINEs in COBOL – before translating the program. (ie the group containing the KEY should be the main group, and the other groups redefining it).

Other solution would be to manually reverse the REDEFINES in the generated XML, and then regenerate – both SQL and DataAccessObject will be regenerated.

Lets say we have a structure with 2 groups, and group-2 redefines Group-1.

We can only put fields from one of the groups (group-1) in the SQL – otherwise there will be duplication and normalization problems.

**Redefined Index fields**

Since the fields from REDEFINEs group are not allowed, then they cannot form indexes either.

Another solution would be to reorganize the KEY – so it belong to main group – instead of the redefinition.

The quick solution would be to reverse the REDEFINES – (ie group-1 redefines group-2).

**How to Split an VSAM Application (e.g. Batch and Online) – Translating into 2 separate directories**

The main challenge would be to enable re-use the Persistent classes generated in first translation (e.g. Online) in the 2nd (e.g. batch)

To enable this, please copy the folder xml/persistence from the first application, into the target output directory of 2nd application. The XML/persistence contains all the information regarding previously translated part of the program, and allows the translator to reuse the existing classes.

*Just before translation* - copy online/xml/persistent directory to batch/xml/persistent. The system will become aware of all previously translated persistent classes (in package online.persistence), and will reuse them. Some of them *maybe* re-generated, but the regenerated code would be compatible with the original set. It would be prudent to back up the original set in order to do a comparison (diff) to see what has changed.

At the end of the batch translation stage, assuming the translation was done to a totally brand new directory, with no previous Java classes (specifically no persistent java files), you will end up with the following directories / packages:

- online/persistent (package online.persistence) : the generated duplicates – you can compare these with the original set to make sure nothing has changed. You can then create a new jar file for persistence layer – this would contain the online.persistent classes from both sets, and remove the set from batch and online.
- Batch/persistent- This should only contain SEQUENTIAL files or KSDS files for which no merges were found.

## 3.5 Data Migration and EBCDIC migration

SoftwareMining will translate COBOL FD definitions to a java-bean format – the bean contains all necessary information regarding every field's format (COMPUTATIONAL, String, 9(nn) …) and starting position. The format and position information can be used by another SoftwareMining utility to read a data file, populate the individual fields in the java-bean, and then write the information to the appropriate database table.

### 3.5.1 Packed Decimal aware Data-Access-Objects (DAO)

For each VSAM structure, SoftwareMining will produce an Object-Relational Data-Access-Object and an SQL schema. For example, a CLIENT-RECORD will result in a CLIENT-TABLE (db schema) and ClientClass.java (DAO). The DAO will be aware of the original COBOL format for each field, e.g. which field is a Packed-decimal, which field is a String, what are the fields sizes, what is the associated table-name, the name of every field, … .

In migration of VSAM KSDS Files to SQL-Tables, packed decimal fields will be represented in numerical columns in SQL tables. E.g. The a field called AGE will be represented in a INTEGER column in SQL. This column will contain the numerical value of the field (e.g. 30) and not the binary representation used in Packed-Decimal fields.

### 3.5.2 EBCDIC To ASCII migration steps

In EBCDIC to ASCII conversion, **the characters representing Packed-Decimal fields must be excluded from the conversion process**. Mainframe FTP utility provides EBCDIC data to ASCII, but the utility will not know which characters correspond to Packed-Decimal fields.

During transfer of files to/from Mainframe, the FTP mode must hence be set to BINARY (to keep all original characters). The actual conversion can be performed using Packed Decimal aware Data-Access-Objects (DAO) generated by SoftwareMining using EbcdicConverter utility.

Examples:

```
java com.softwaremining.tools.datamigration.EbcdicConvertor e2a
com.company.app1.persistence.Client EBCDIC.input.datafile ASCII.output.datafile
```

Where

`e2a` – Ebcdic to Ascii (alternatively use a2e for asci to Ebcdic).

`com.company.app1.persistence.Client` is one of the translated DAO classes.

### 3.5.3 Limitations on Data Migration of complex structures with REDEFINED Packed-Decimals

When a structure (or DAO) contains complex redefinitions of fields mapped to Packed-Decimal (PD) fields, the EbcdicConvertor will be unable to establish whether to EBCDIC to Ascii migration on the position will/will-not be required. The tool may make an incorrect assumption. Therefore character conversion on structures with complex redefinitions will need manually verification, and may require manual correction.

The EBCDIC conversion utility will perform character conversion on data corresponding to ALL fields except:

1. Packed-Decimal fields. PD data have same binary representation on translated system, mainframe or rehosted systems (e.g. using Microfocus or Fujitsu Cobol). Therefore PD fields should not undergo character conversion.
2. REDEFINED Fields.

For example when dealing with the translation of the structure:

```
01 RECORD1.
     05 FOO    PIC  S9(5) COMP-3.
     05 BAR-R  PIC  X(3)  REDEFINES FOO.
     05 EFG    PIC  X(10).
```

Fields **FOO** and **BAR-R** both are at beginning of structure **RECORD1**- ie they start at position 0. They are both represented by to 3 characters. The EBCDIC character conversion utility will iterate through all fields to do the following:

- Field **FOO**, position 0 – length 3 : Ignore due to rule 1 above (ignore Packed-Decimal fields)
- Field **BAR-R**, position 0 – length 3 : Ignore due to rule 1 above (ignore redefined fields)
- Field **EFG**, position 3 – length 10 : Perform character conversion on residing between positions 3 and 13

However, consider what happens if the structure is changed to:

```
01 RECORD1.
     05 BAR    PIC  X(3)  .
     05 FOO-R  PIC  S9(5) COMP-3 REDEFINES BAR.
     05 EFG    PIC  X(10).
```

- Field **BAR**, position 0 – length 3 : is neither Packed-DEC or REDEFINES. Perform character conversion on residing between positions 1 and 3
- Field **FOO-R**, position 0 – length 3 : Ignore due to rule 1 and rule-2 above
- Field **EFG**, position 3 – length 10 : Perform character conversion on residing between positions 3 and 13

### 3.5.4 Populating the databases (KSDS Files)

The system uses the SEQUENTIAL part of the COBOL KSDS data-files (i.e. just the data, and not the index information). It is the Sequential Data which has to be FTP'ed across and used in population of generated SQL Schemas.

The data migration strategy is based on a utility which can feed data from the KSDS Sequential data file into SoftwareMining generate DAO. It is the DAO which inserts the data into the SQL table. The utility is also able to perform EBCDIC to ASCI conversion on the supplied data-file.

For more information please see PopulateDbTableFromSequentialFile .

### 3.5.5 Sample Scripts

Download Sample runtime scripts (batch programs) for EBCDICConvertor, data migration and more.

## 3.6 Performance Optimization techniques

The Object-Relational classes will dynamically generate the SQL necessary to access database at runtime. The performance of system can be greatly affected by

- amount of the data available in database,
- number of indexes defined on the tables,
- the number of KEYS used for the search (the issued SQL will have to ORDER the results using the keys),
- types of search (an Equality search is faster than Equals-or-Greater-Than .

The following section looks at the optimization API available in the SoftwareMining framework.

### 3.6.1 Using SQL *LIKE* instead of Greater-or-Equal (>=)

**Method**: void assignChangeGEConditonToLIKE

**JavaDoc**:
http://www.softwaremining.com/javadoc/com/softwaremining/sql/ISQLWrapper.html#assignChangeGEConditonToLIKE(boolean)

**Description:** In certain conditions replacing GE (>=) searches in an SQL statement with LIKE can potentially offer a huge performance improvement on the database. For Example:

```
SELECT ... from TABLE WHERE KEY_1 >= ? AND KEY_2 >= ?.
```

If each key is 5 chars, and the supplied values are "A " and "B " - the above translates to:

```
SELECT ... from TABLE WHERE KEY_1 >= 'A    ' AND KEY_2 >= 'B    '
```

Converting it to LIKE will become

```
SELECT ... from TABLE WHERE KEY_1 LIKE 'A%' AND KEY_2 LIKE 'B%'
```

Note in this case the values 'A ' and 'b ' are trimmed (spaces removed) before the % is added. Ie when the value passed to key1 is 'ABCDE' – then the system will generate:

```
SELECT ... from TABLE WHERE KEY_1 >= 'ABCDE' AND KEY_2 LIKE 'B%'
```

**Usage Example:**

```
Company company = new Company(this);
…
company.assignChangeGEConditonToLIKE (true);
company.seek(company.getGroupKey()   CONDITION_TYPE_GREATER_OR_EQUAL);
```

**Assigning Application Wide Default Values**

To define the above for ALL generated SQL statements – add the following to db.properties configuration file:

```
OR_CHANGE_GE_CONDITION_TO_LIKE__DEFAULT=true
```

### 3.6.2 Fine Tuning the LIKE statements

**Method**: void assignChangeGTEQ2EQForLIKECondition

**JavaDoc**:
http://www.softwaremining.com/javadoc/com/softwaremining/sql/ISQLWrapper.html#assignChangeGTEQ2EQForLIKECondition(boolean)

**Description:**

Only applies when Using SQL LIKE instead of Greater-or-Equal (e.g. defining the following in db.properties file:

```
OR_CHANGE_GE_CONDITION_TO_LIKE__DEFAULT=true
```

E.g. Consider generating

```
SELECT ... from TABLE WHERE KEY_1 >= ? AND KEY_2 >= ?.
```

If each key is 5 chars, and the supplied values are "ABCDE" and "B " - the above translates to:

```
SELECT ... from TABLE WHERE KEY_1 >= 'ABCDE' AND KEY_2 >= 'B    '
```

Converting it to LIKE will become

```
SELECT ... from TABLE WHERE KEY_1 >= 'ABCDE' AND KEY_2 LIKE 'B%'
```

When this flag is set, if the char-count of a field is equal to that of the supplied value, then the generated SQL will become

```
SELECT ... from TABLE WHERE KEY_1 = 'ABCDE' AND KEY_2 LIKE 'B%'
```

**Usage Example:**

```
Company company = new Company(this);
…
company.assignChangeGEConditonToLIKE (true);
company.assignChangeGTEQ2EQForLIKECondition (true);
```

**Assigning Application Wide Default Values**

To define the above for ALL generated SQL statements – add the following to db.properties configuration file:

```
OR_CHANGE_GE_CONDITION_TO_LIKE__DEFAULT=true
OR_CHANGE_GTEQ_TO_EQ_FOR_LIKE_CONDITIONS__DEFAULT=true
```

### 3.6.3 Limiting the number of rows returned by SQL Statement

**Methods**: `assignTopRowSelectStrategyCount` and `assignTopRowSelectStrategyRange(int startRowNumber, int count);`

**JavaDoc**:

http://www.softwaremining.com/javadoc/com/softwaremining/sql/ISQLWrapper.html#assignTopRowSelectStrategyCount(int)

http://www.softwaremining.com/javadoc/com/softwaremining/sql/ISQLWrapper.html#assignTopRowSelectStrategyRange(int, int)

**Description:** This feature is usually applicable to online applications. Consider a scenario where there are over hundred thousand records in a table (eg. Zip-code table), and an online screen offers the user to scroll through them from a user defined point - (e.g. from letter "B")– at a rate of 20 records per-page.

The default system behaviour would be to issue a SQL select statement which may return hundred thousand records, e.g.

```
SELECT ... from TABLE WHERE KEY_1 >= ?
```

For an online application – returning too much data may not be necessary – the user is unlikely to scroll through thousands of pages. This API will allow the system to limit the number of rows ORDERED and returned by the database.

**Usage Example:**

```
Company company = new Company(this);

…

company. assignTopRowSelectStrategyCount (200);
```

By using this API the system will issue the following SQL statement for ORACLE database. (the SQL is slightly different for DB2, MS-SQL and MySQL)

```
SELECT ... from TABLE WHERE KEY_1 >= ?
     ROWNUM  < OR_200
```

**Assigning Application Wide Default Values**

To define the above for ALL generated SQL statements – add the following to db.properties configuration file:

```
OR_TOP_ROW_SELECT_STRATEGY_COUNT__DEFAULT=200
```

### 3.6.4 Restricting the number of Keys used in SQL

**Methods**: `assignLowerLimitForExpandGTEQSearches` and

`assignUpperLimitForExpandGTEQSearches`

**JavaDoc**:

http://www.softwaremining.com/javadoc/com/softwaremining/sql/ISQLWrapper.html#assignLowerLimitForExpandGTEQSearches(int)

http://www.softwaremining.com/javadoc/com/softwaremining/sql/ISQLWrapper.html#assignUpperLimitForExpandGTEQSearches(int)

**Description:** This feature is usually applicable to searches on ORM classes which contain many KEYS.

Lets say there are 3 key fields involved in a search with condition GTEQ. In default settings, the following SQL condition will be generated

```
SELECT … FROM … WEHRE
    (A > ?)
or (A = ? and B > ?)
or (A = ? and B = ? and C >= ?))
```

by declaring `dao.assignLowerLimitForExpandGTEQSearches(3)` - the conditions for key below position 3 are ignored, and only the red areas below will be generated.

```
SELECT … FROM … WHERE
    (A > ?)
or (A = ? and B > ?)
or (A = ? and B = ? and C >= ?))
```

Similarly by using `dao.assignUpperLimitForExpandGTEQSearches(1)` - the conditions for key above position 1 are ignored, and only the red areas below will be generated.

```
SELECT … FROM … WHERE
    (A > ?)
or (A = ? and B > ?)
or (A = ? and B = ? and C >= ?))
```

**Usage Example:**

For a Object relational Data-Access-Object DAO with 5 keys

```
DAO dao = new DAO(this);

…

dao. assignUpperLimitForExpandGTEQSearches (2);

dao. assignLowerLimitForExpandGTEQSearches (4);
```

When using the above API the system will issue the black (SELECT) , blue (`UpperLimit`), and red (LowerLimit) of the following SQL statement (the grey parts will not be generated).

```
SELECT … FROM … WHERE
    (A > ?)
or (A = ? and B > ?)
or (A = ? and B = ? and C >= ?))
or (A = ? and B = ? and C = ? and D >= ?))
or (A = ? and B = ? and C = ? and D = ? and E = ?))
```

**Assigning Application Wide Default Values**

This value needs to be specific to each Data-Access – and cannot have an application wide setting.

### 3.6.5 Adding new SQL conditions to a search

**Methods**: `assignSQLConditions` and

`assignAdditionalSQLConditions`

**JavaDoc**:

http://www.softwaremining.com/javadoc/com/softwaremining/sql/ISQLWrapper.html#assignSQLConditions(java.lang.String)

http://www.softwaremining.com/javadoc/com/softwaremining/sql/ISQLWrapper.html#assignAdditionalSQLConditions(java.lang.String)

**Description:** These APIs allow customization of the WHERE conditions generated for each DAO.

| | |
|---|---|
| `assignSQLConditions` | Prohibits auto-generation of default conditions (using the keys declared on DAO). |
| `assignAdditionalSQLConditions` | Appends the new condition to the end of the ones generated by system (using the keys declared on DAO). |

Also see:

| | |
|---|---|
| `assignSQLConditions` | Prohibits auto-generation of default conditions (using the keys declared on DAO). |
| `assignAdditionalSQLConditions` | Appends the new condition to the end of the ones generated by system (using the keys declared on DAO). |

Also see

| | |
|---|---|
| `assignAdditionalSQLOrderBy` | Insert additional ORDER-BY statement to end of generated SQL |

**Usage Example:**

```
Company company = new Company(this);
…
company. assignAdditionalSQLConditions( "(COLUMN_2 = 12)");
company.seek(company.getGroupKey() , CONDITION_TYPE_GREATER_OR_EQUAL);
```

Generates:

```
SELECT ... from TABLE WHERE KEY_1 >= ? AND (COLUMN_2 = 12)
```

### 3.6.6 Reviewing the runtime generated SQL

Add the following to logging configuration file:

| | |
|---|---|
| Logj4 | log4j.logger.com.softwaremining.sql=TRACE |
| Log4j2 | <Logger name="com.softwaremining.sql" level="TRACE"/> |
| Log4net | <logger name="SoftwareMining.sql"><level value="TRACE" /></logger> |

## 3.7 Working with Mainframe EBCDIC files

Usually the translated system will be running on Unix or MS Windows machine and will be using UTF character sets for everything including screen displays, database access, sorting data files and read/write from Sequential and indexed files. These operations can be further fine-tuned using the following settings into the file softwaremining.properties for Java translations, (SoftwareMining.config in C#).

```
# APP_CHARACTER_ENCODING
# Default to ISO-8859-1 (mainly used in construction of Strings)
# Same options as DB_CHARACTER_ENCODING
#
APP_CHARACTER_ENCODING=ISO-8859-1


# Character encoding for database access (Translatio of VSAM to ORM, as well as
EXEC SQL).
# one of "UTF8" , "UTF16" , "ISO-8859-1" , "US-ASCII" , or "8859_1" (EBCDIC), ...
#  Defaults to APP_CHARACTER_ENCODING
#
#DB_CHARACTER_ENCODING=ISO-8859-1


# The following setting controls read/write of files from file system.
# setting it to IBM-1047 will allow it read mainframe Ebcdic files
#  Defaults to APP_CHARACTER_ENCODING
#
# DATAFILE_CHARACTER_ENCODING=IBM-1047
```

But sometime it is necessary to run parts of the system on mainframe (COBOL), and part in Java/C#. In such cases there will be a need for the translated system to access the EBCDIC data files generated on mainframe, or to generate EBCDIC files for use on the mainframe.

The following two approaches cater for this requirements.

### 3.7.1 Running Java in ASCI (UTF) , but reading EBCDIC VSAM datafiles

The following softwaremining.properties for Java translations, (SoftwareMining.config in C#) allows reading/writing ALL data files in EBCDIC character-sets.

```
# Allows running java in default c\haracter-set (as assigned on the machine -
usually set to -8859-1 )
# but reading and writing EBCDIC data files for compatibility with original
mainframe.
#   Defaults to false
# DATAFILES_ARE_EBCDIC=false

# The following setting controls read/write of files from file system.
# setting it to IBM-1047 will allow it read mainframe Ebcdic files
#  Defaults to APP_CHARACTER_ENCODING
#
# DATAFILE_CHARACTER_ENCODING=IBM-1047



# Character encoding used on Mainframe.
# this is only used for EBCDIC to ASCII conversion (if any), and when
#   USE_SM_EBCDIC_DECODER=false
# Defaults to IBM-1047
#
# ORIG_EBCDIC_CHARACTER_ENCODING=IBM-1047

# Only used for data-migration or working with EBCDIC files.
# When set - The system will use SoftwareMining's built-in utilities for
conversion from EBCDIC to ASCI and back.
# Otherwise, it will use JAVA's string encoding/decoding
#  When false (ie using java's character encodings) the following parameters
need to be set:
#     APP_CHARACTER_ENCODING (typically set to default "ISO-8859-1")
#     DATAFILE_CHARACTER_ENCODING (Set to  "IBM-1047" for Ebcdic data files).
# When switched off, it will use SoftwareMining's inbuilt conversion.
# USE_SM_EBCDIC_DECODER=true
```

Note limitations remain on complex structures with REDEFINES statements.

### 3.7.2 Running Java in EBCDIC

Java runtime configuration allows it to run with a character encoding different to the standard defined on the machine. E.g. a batch application can be started using:

```
java -Dfile.encoding=IBM-1047 com.softwaremining.SMTextAppLauncher com.company.app1.Prog1
```

Please also use the appropriate configurations for character settings in softwaremining.properties for Java translations, (SoftwareMining.config in C#).