Some Examples Using Blaise Component Pack

Gina-Qian Y. Cheung
Survey Reseach Center
University of Michigan

Introduction:

Blaise Component Pack, commonly called BCP, is a COM library. Blaise uses a proprietary database format that is accessible by the Blaise System only. We can use this library to directly access the Blaise database without the need for any intermediate interface or application. As with any other component this provides properties and methods accessible to programmers.

Using this BCP we can do many things with Blaise Database. Some of the main tasks which can be accomplished using BCP are:

- Fetch all the field information (like blockname, field name, field type, display text,primary key, etc.) by reading either Blaise Database(*.bdb) or Blaise Meta Information file (*.bmi).

- Fetch all the data from the database.

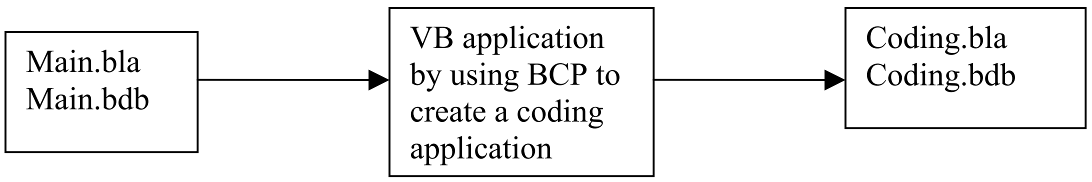
- Navigate through all the FIELDS and RULES (using Rules Navigator) of a data model.

- There is no need to write manipula or cameleon scripts. With BCP you can perform a variety of programming and scripting in environments like Borland® Delphi™, Microsoft® Visual Basic®, Microsoft® Visual Basic for Applications (VBA), Microsoft® Visual C++®, Microsoft® Visual J++®, and Sybase®PowerBuilder.

Some examples we had:

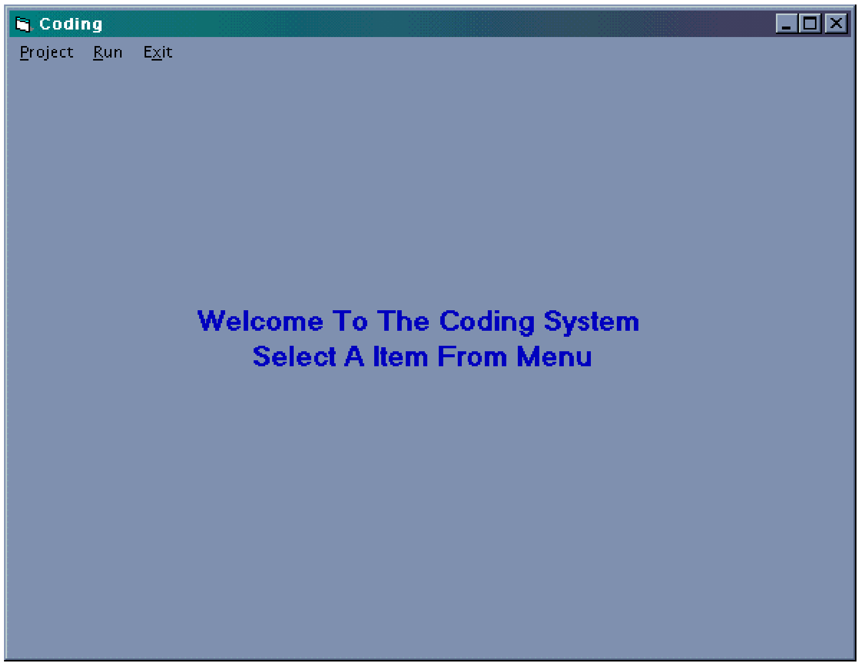
Example 1 : Creating a Blaise Coding Application

Created an application in Visual Basic that creates a Blaise coding program from the main Blaise application. In this application, it lists all the OPEN/STRING fields to the supervisor so that they can select all the fields needed to be coded. Then the Coding Application is created and related data fields are populated from main data set to the coding application data set.



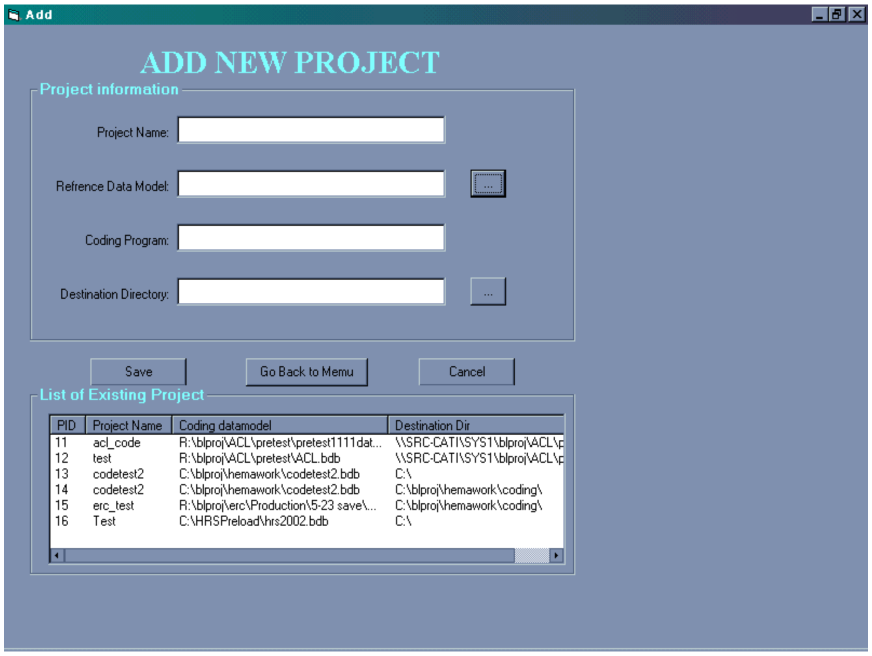
This application creates a coding program after reading a database/data model.

Run the application



Welcome To The Coding System
Select A Item From Menu

Create a new project (Project -> Add)



ADD NEW PROJECT

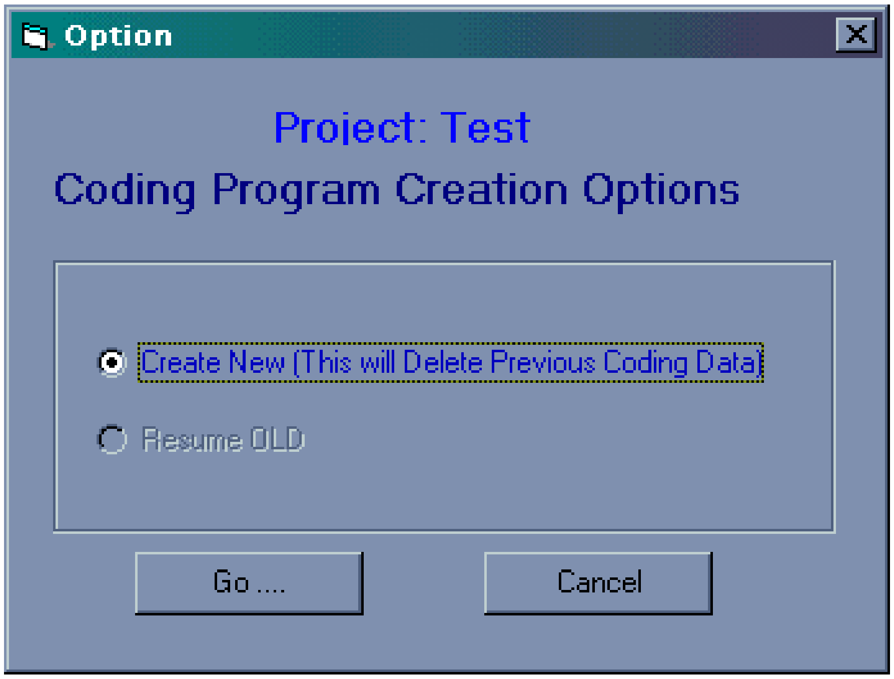
Project information

Project Name:	
Refrence Data Model:	
Coding Program:	
Destination Directory:	

List of Existing Project

PID	Project Name	Coding datamodel	Destination Dir
11	acl_code	R:\blproj\ACL\pretest\pretest1111dat...	\\SRC-CATI\SYS1\blproj\ACL\p
12	test	R:\blproj\ACL\pretest\ACL.bdb	\\SRC-CATI\SYS1\blproj\ACL\p
13	codetest2	C:\blproj\hemawork\codetest2.bdb	C:\
14	codetest2	C:\blproj\hemawork\codetest2.bdb	C:\blproj\hemawork\coding\
15	erc_test	R:\blproj\erc\Production\5-23 save\...	C:\blproj\hemawork\coding\
16	Test	C:\HRSPreload\hrs2002.bdb	C:\

After this click on Project - > OPEN from the main menu


Option

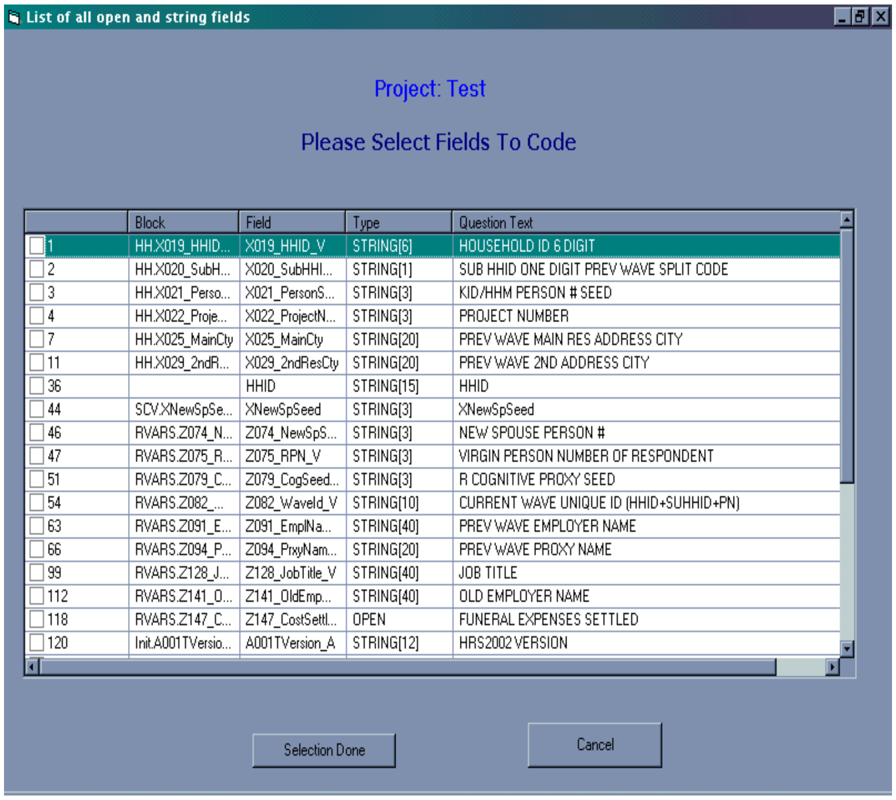
Project: Test

Coding Program Creation Options

[✓ Create New (This will Delete Previous Coding Data)]

Resume OLD

Click on "Go" button
This displays the list of all open & string fields to code

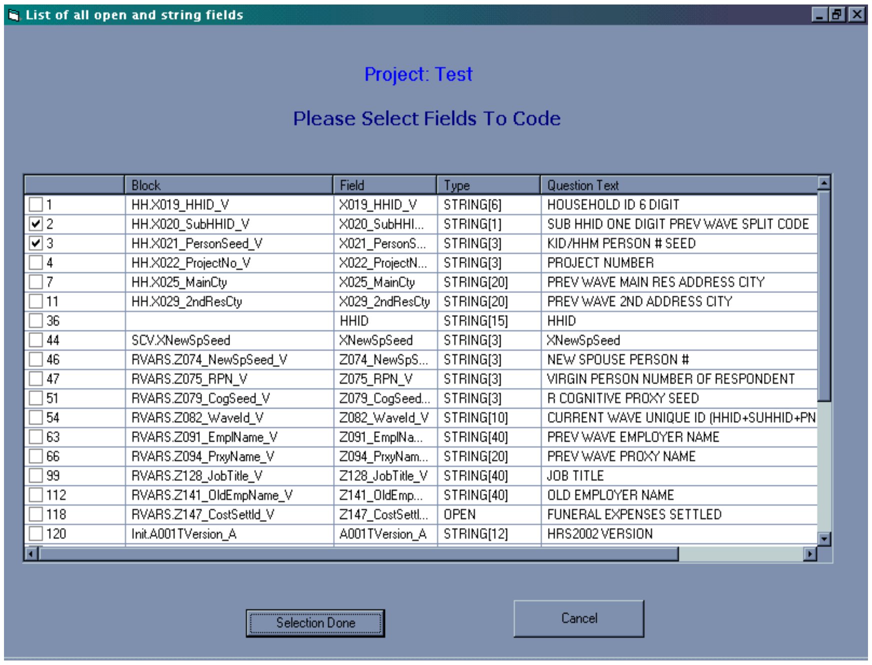

List of all open and string fields

Project: Test

Please Select Fields To Code

	Block	Field	Type	Question Text
☐ 1	HH.X019_HHID...	X019_HHID_V	STRING[6]	HOUSEHOLD ID 6 DIGIT
☐ 2	HH.X020_SubH...	X020_SubHHI...	STRING[1]	SUB HHID ONE DIGIT PREV WAVE SPLIT CODE
☐ 3	HH.X021_Perso...	X021_PersonS...	STRING[3]	KID/HHM PERSON # SEED
☐ 4	HH.X022_Proje...	X022_ProjectN...	STRING[3]	PROJECT NUMBER
☐ 7	HH.X025_MainCty	X025_MainCty	STRING[20]	PREV WAVE MAIN RES ADDRESS CITY
☐ 11	HH.X029_2ndR...	X029_2ndResCty	STRING[20]	PREV WAVE 2ND ADDRESS CITY
☐ 36		HHID	STRING[15]	HHID
☐ 44	SCV.XNewSpSe...	XNewSpSeed	STRING[3]	XNewSpSeed
☐ 46	RVARS.Z074_N...	Z074_NewSpS...	STRING[3]	NEW SPOUSE PERSON #
☐ 47	RVARS.Z075_R...	Z075_RPN_V	STRING[3]	VIRGIN PERSON NUMBER OF RESPONDENT
☐ 51	RVARS.Z079_C...	Z079_CogSeed...	STRING[3]	R COGNITIVE PROXY SEED
☐ 54	RVARS.Z082_...	Z082_WaveId_V	STRING[10]	CURRENT WAVE UNIQUE ID (HHID+SUHHID+PN)
☐ 63	RVARS.Z091_E...	Z091_EmplNa...	STRING[40]	PREV WAVE EMPLOYER NAME
☐ 66	RVARS.Z094_P...	Z094_PrxyNam...	STRING[20]	PREV WAVE PROXY NAME
☐ 99	RVARS.Z128_J...	Z128_JobTitle_V	STRING[40]	JOB TITLE
☐ 112	RVARS.Z141_O...	Z141_OldEmp...	STRING[40]	OLD EMPLOYER NAME
☐ 118	RVARS.Z147_C...	Z147_CostSettl...	OPEN	FUNERAL EXPENSES SETTLED
☐ 120	Init.A001TVersio...	A001TVersion_A	STRING[12]	HRS2002 VERSION

Select the fields to be coded :

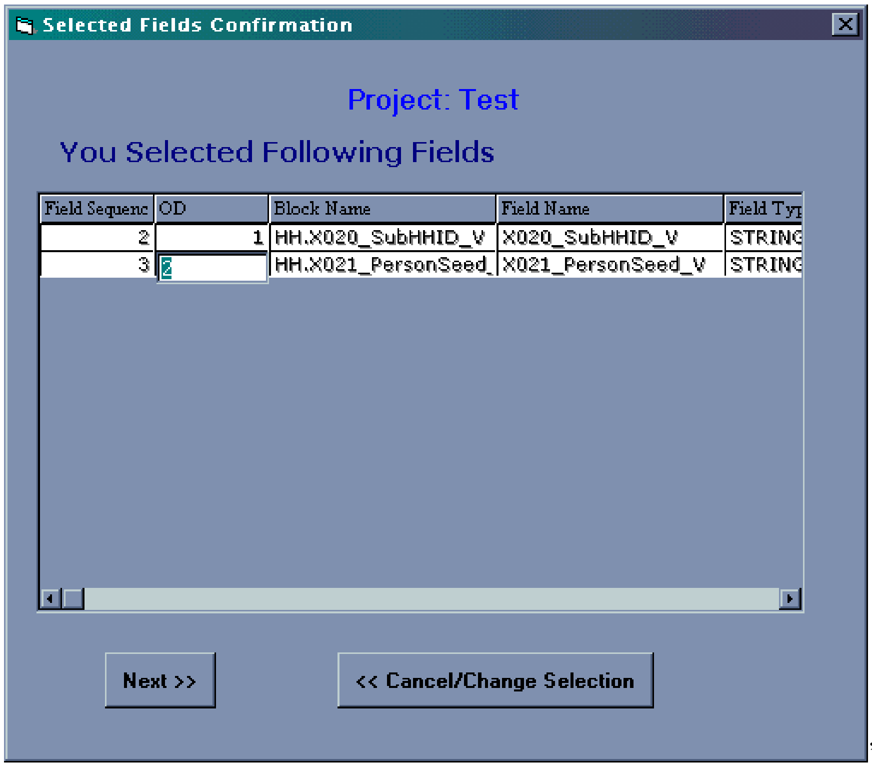


Project: Test

Please Select Fields To Code

	Block	Field	Type	Question Text
☐ 1	HH.X019_HHID_V	X019_HHID_V	STRING[6]	HOUSEHOLD ID 6 DIGIT
✓ 2	HH.X020_SubHHID_V	X020_SubHHI...	STRING[1]	SUB HHID ONE DIGIT PREV WAVE SPLIT CODE
✓ 3	HH.X021_PersonSeed_V	X021_PersonS...	STRING[3]	KID/HHM PERSON # SEED
☐ 4	HH.X022_ProjectNo_V	X022_ProjectN...	STRING[3]	PROJECT NUMBER
☐ 7	HH.X025_MainCty	X025_MainCty	STRING[20]	PREV WAVE MAIN RES ADDRESS CITY
☐ 11	HH.X029_2ndResCty	X029_2ndResCty	STRING[20]	PREV WAVE 2ND ADDRESS CITY
☐ 36		HHID	STRING[15]	HHID
☐ 44	SCV.XNewSpSeed	XNewSpSeed	STRING[3]	XNewSpSeed
☐ 46	RVARS.Z074_NewSpSeed_V	Z074_NewSpS...	STRING[3]	NEW SPOUSE PERSON #
☐ 47	RVARS.Z075_RPN_V	Z075_RPN_V	STRING[3]	VIRGIN PERSON NUMBER OF RESPONDENT
☐ 51	RVARS.Z079_CogSeed_V	Z079_CogSeed...	STRING[3]	R COGNITIVE PROXY SEED
☐ 54	RVARS.Z082_WaveId_V	Z082_WaveId_V	STRING[10]	CURRENT WAVE UNIQUE ID (HHID+SUHHID+PN
☐ 63	RVARS.Z091_EmplName_V	Z091_EmplNa...	STRING[40]	PREV WAVE EMPLOYER NAME
☐ 66	RVARS.Z094_PrxyName_V	Z094_PrxyNam...	STRING[20]	PREV WAVE PROXY NAME
☐ 99	RVARS.Z128_JobTitle_V	Z128_JobTitle_V	STRING[40]	JOB TITLE
☐ 112	RVARS.Z141_OldEmpName_V	Z141_OldEmp...	STRING[40]	OLD EMPLOYER NAME
☐ 118	RVARS.Z147_CostSettld_V	Z147_CostSettl...	OPEN	FUNERAL EXPENSES SETTLED
☐ 120	Init.A001TVersion_A	A001TVersion_A	STRING[12]	HRS2002 VERSION

After selecting the fields to Code, click on “Selection Done” button



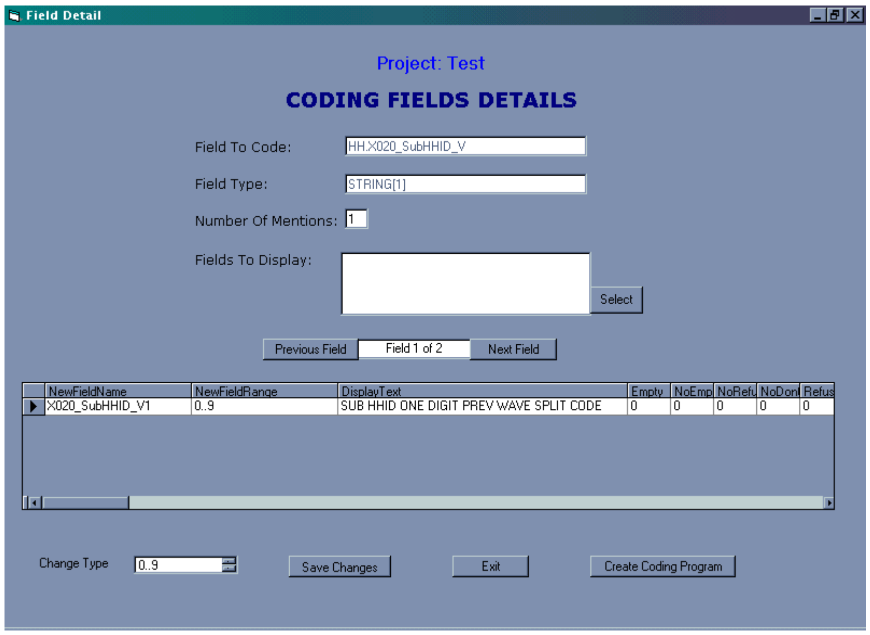
Project: Test

You Selected Following Fields

Field Sequenc	OD	Block Name	Field Name	Field Typ
2	1	HH.X020_SubHHID_V	X020_SubHHID_V	STRING
3	2	HH.X021_PersonSeed_	X021_PersonSeed_V	STRING

,

For each & every field selected, the following screen will come up where you can specify the number of mentions for each field and select the fields to display while coding.


Field Detail

Project: Test

CODING FIELDS DETAILS

Field To Code: HH.X020_SubHHID_V

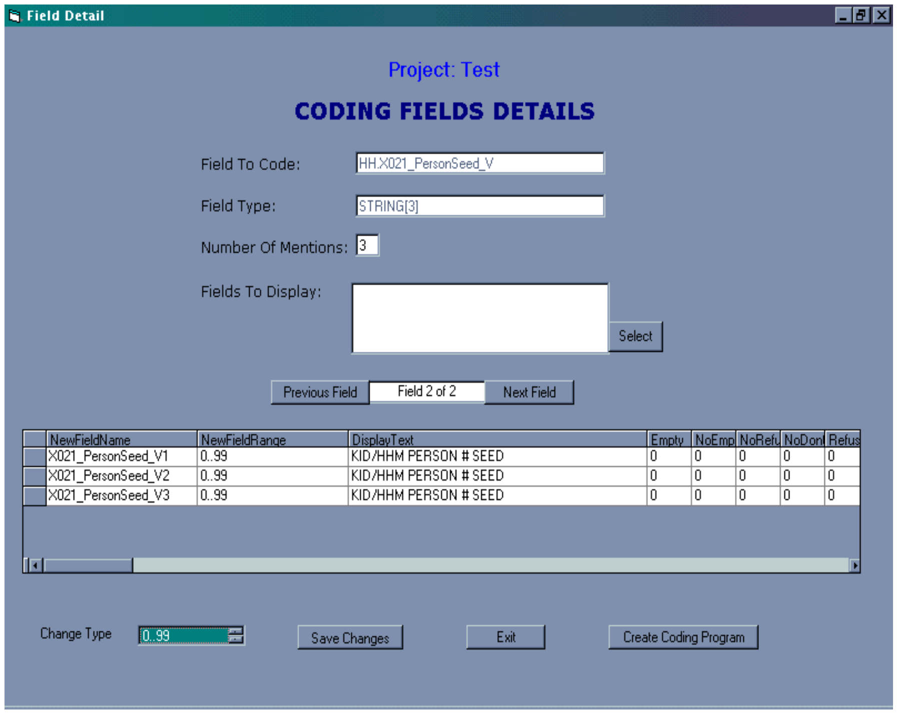
Field Type: STRING[1]

Number Of Mentions: 1

Fields To Display:

NewFieldName	NewFieldRange	DisplayText	Empty	NoEmp	NoRefu	NoDonI	Refus
X020_SubHHID_V1	0..9	SUB HHID ONE DIGIT PREV WAVE SPLIT CODE	0	0	0	0	0

Change Type 0..9


Field Detail

Project: Test

CODING FIELDS DETAILS

Field To Code: HH.X021_PersonSeed_V

Field Type: STRING[3]

Number Of Mentions: 3

Fields To Display:

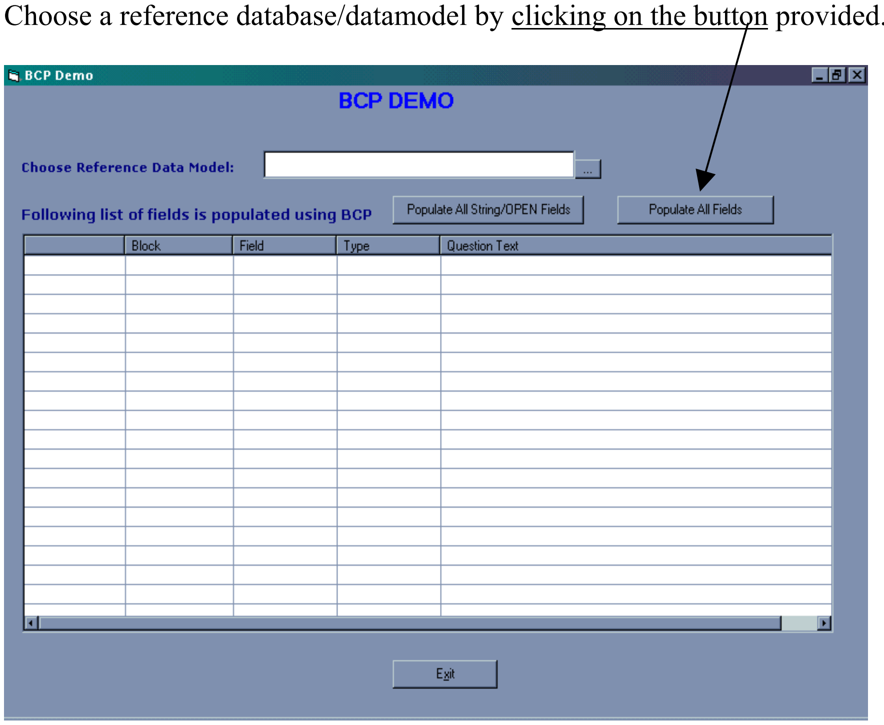
NewFieldName	NewFieldRange	DisplayText	Empty	NoEmp	NoRefu	NoDonI	Refus
X021_PersonSeed_V1	0..99	KID/HHM PERSON # SEED	0	0	0	0	0
X021_PersonSeed_V2	0..99	KID/HHM PERSON # SEED	0	0	0	0	0
X021_PersonSeed_V3	0..99	KID/HHM PERSON # SEED	0	0	0	0	0

Change Type 0..99

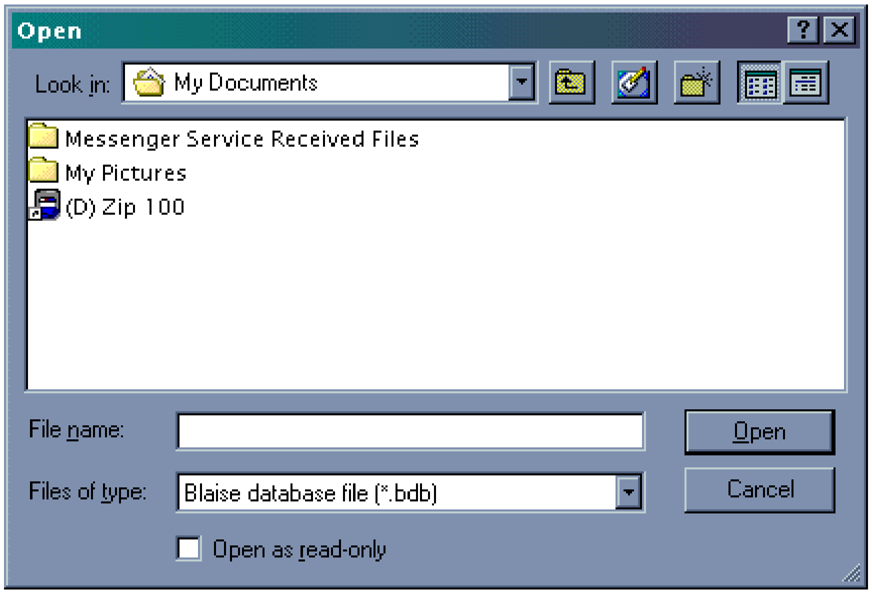
<u>Example 2:</u> View and edit Blaise data

Created a form that displays some selected fields data for all the cases. This data can be edited from within VB interface (you need to choose a field to edit as there are hundreds of fields in the database).

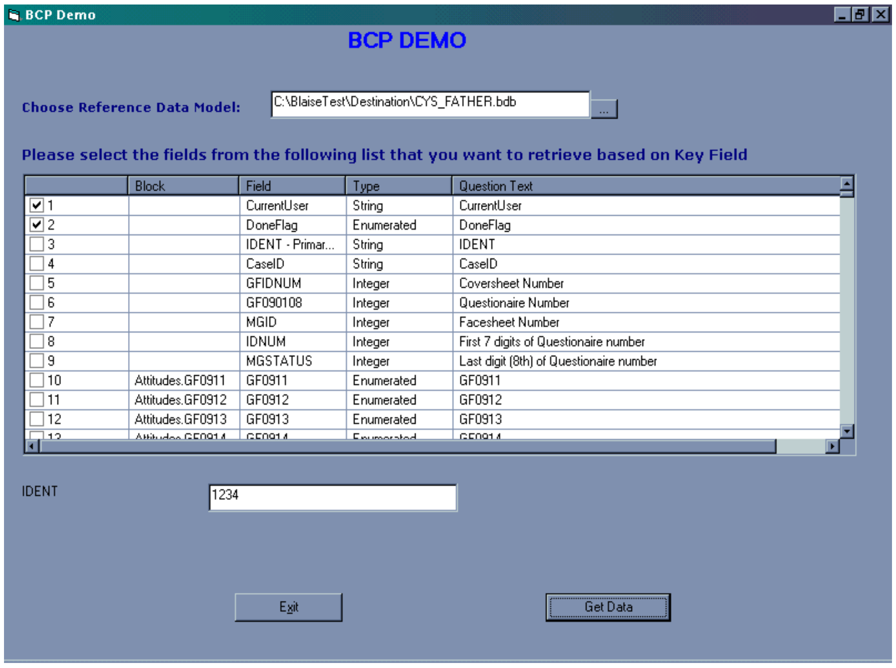
Choose a reference database/datamodel by clicking on the button provided.


BCP Demo
BCP DEMO
Choose Reference Data Model:
...
Following list of fields is populated using BCP
Populate All String/OPEN Fields
Populate All Fields
Block
Field
Type
Question Text
Exit

Clicking on that button will bring up the following window :



After selecting the database enter the value of the keyfield for which you want to edit the data. Select the fields you want to edit – For now it is restricted to two fields.


BCP Demo

BCP DEMO

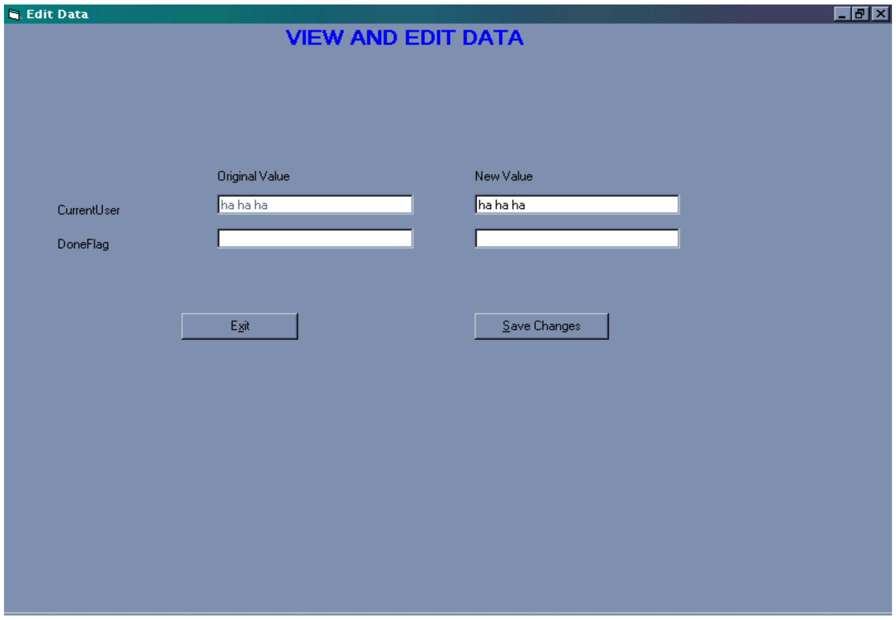
Choose Reference Data Model: C:\BlaiseTest\Destination\CYS_FATHER.bdb

Please select the fields from the following list that you want to retrieve based on Key Field

	Block	Field	Type	Question Text
✓ 1		CurrentUser	String	CurrentUser
✓ 2		DoneFlag	Enumerated	DoneFlag
3		IDENT - Primar...	String	IDENT
4		CaseID	String	CaseID
5		GFIDNUM	Integer	Coversheet Number
6		GF090108	Integer	Questionaire Number
7		MGID	Integer	Facesheet Number
8		IDNUM	Integer	First 7 digits of Questionaire number
9		MGSTATUS	Integer	Last digit (8th) of Questionaire number
10	Attitudes.GF0911	GF0911	Enumerated	GF0911
11	Attitudes.GF0912	GF0912	Enumerated	GF0912
12	Attitudes.GF0913	GF0913	Enumerated	GF0913

IDENT 1234

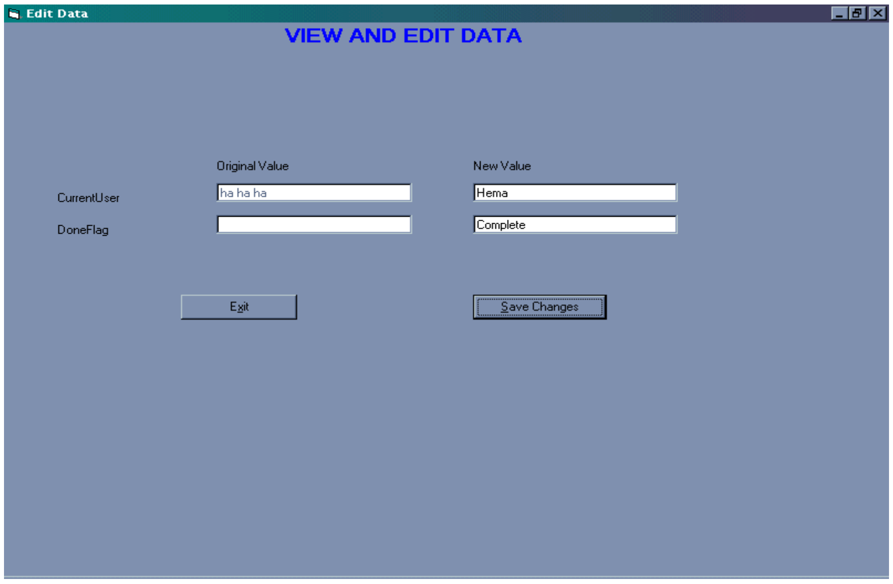
After doing all this click on Get data.


Edit Data

VIEW AND EDIT DATA

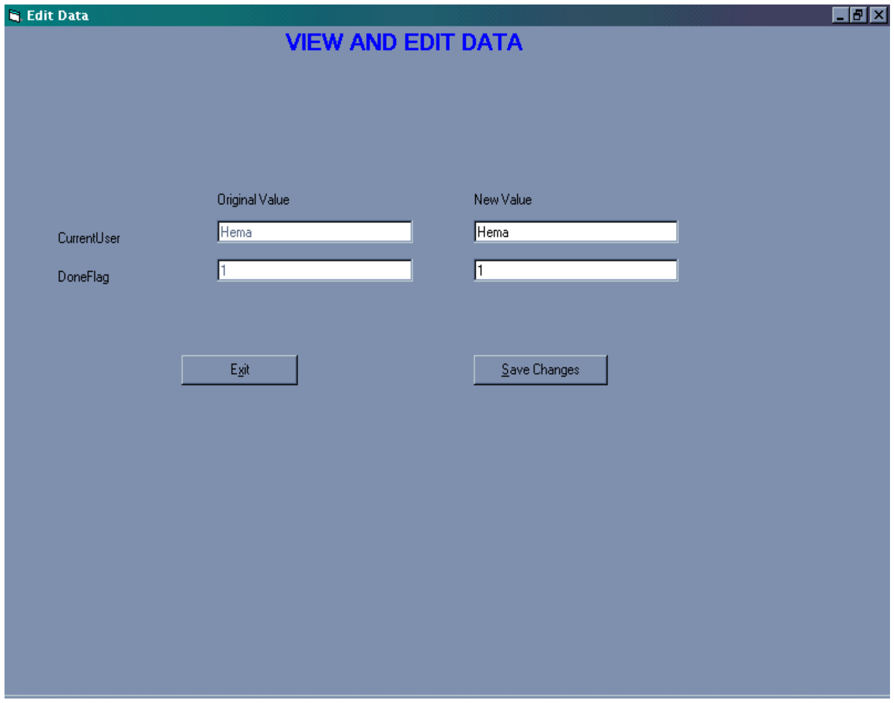
	Original Value	New Value
CurrentUser	ha ha ha	ha ha ha
DoneFlag		

Here you can view the existing data from these fields and edit them.. Cllick on “Save Changes,” this will update the data in database if the data is valid as per rules written in Blaise program.



ha ha ha
Hema
Complete

Viewing Data again to see the result :



	Original Value	New Value
CurrentUser	Hema	Hema
DoneFlag	1	1

It's showing 1 in Done Flag as it is enumerated data type. "Complete" is assigned as "1" in UDT(User Defined Data Type). We can display the category text also if it is required.

EXAMPLE 3. BCP Usage and Data/Metadata-out

Purpose: The conditions table, along with the identification of the condition in the variables table, is the identification of the total set of conditions under which the field data is updated in the entire data model. This kind of information is useful in describing or analyzing a questionnaire without the use of a specific programming language such as Blaise. By examining the conditions under which the field is updated, it is possible to verify that the intentions of the designers are being correctly implemented. In addition, by searching the condition table, it is also possible to isolate which fields are dependant on another field being answered. This would be particularly useful when analyzing frequency tables on a dataset. By examining the conditions for a field, you can identify primary info that influence whether or not that question is asked.

The program we wrote, Universe.exe, was written in Visual Basic 6.0 and used the COM object library to analyze a prepared datamodel. We made calls to various BCP functions and methods to gather the information we needed. The program made use of recursive-style calls very much in the manner that Cameleon does. In particular, we used the Rules Navigator functions to traverse the Rules in the datamodel and get information about the Current Conditions (another Rules Navigator function) for each field.

These were the Rules Navigator functions used:

```
MoveToFirstStatement
MoveToThenStatement
MoveToChildStatement
MoveToNextStatement
MoveToParentStatement
MoveToElseStatement
```

In the process of completing our project to produce tables to output Blaise data to SAS in the form of relational tables we used the BCP to produce our set of conditions under which fields in the datamodel were assigned or asked.

These are some of the functions used:

```
 ' Determine the Rules statement type (IF, ELSE, Assignment, ASK, FOR loop, Block Reference, etc.)
   StatementType

  ' Determine the set of Fields involved in the condition
   InvolvedFields.Item(1)

  '  Determine the number of nested conditions which apply to the field method
   CurrentConditions.Count

  ' Set the condition based on the above count used as an index
   CurrentConditions.Item(nCount)
```

Several other functions were used from the BCP to determine certain field attributes such as the field type and the field method (ASK, KEEP, SHOW, or assignment).

```
' Get an empty database reference
Set xmDatabaseManager = New BlAPI4A.DatabaseManager
Set xmDatabase = xmDatabaseManager.OpenDatabase("")

' Set the datamodel file name
xmDatabase.DictionaryFileName = xsMetaDataFile

' Determine Field, Auxfield or Local
nFieldKind = xmDatabase.Field(sFieldName).FieldKind

' Determine the base fieldname such as Name instead of HHL.Person.Name
sFieldName = xmDatabase.Field(sFieldName).LocalName
```

Here is the section of code we used for extracting a set of conditions once we have opened the database (datamodel) and navigated to a particular field for which we want the conditions:

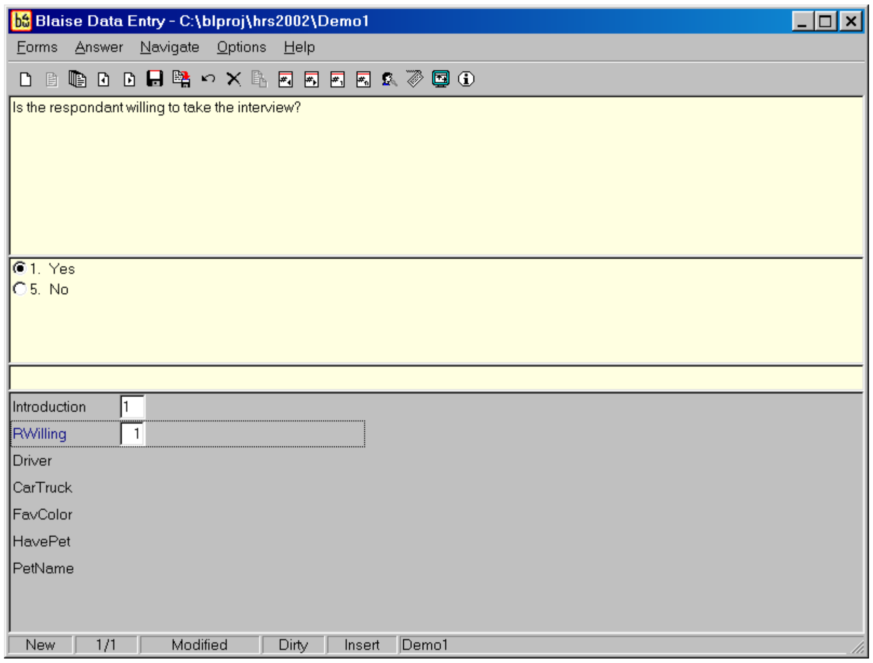
```
' Concatenate conditions
sCondition = ""
nConditionCount = xmDatabase.RulesNavigator.CurrentConditions.Count
If nConditionCount > 0 Then
   nCount = 0
   nTerm = 0
   For nLevel = 1 To nStmtLevel
      If nStmtType(nLevel) = blstCondition Or _
         nStmtType(nLevel) = blstElse Or _
         nStmtType(nLevel) = blstForLoop Then
         nCount = nCount + 1
         If nCount > nConditionCount Then
            MsgBox ("Count exceeds ConditionCount")
         End If
      End If
      If nStmtType(nLevel) = blstCondition Or _
         nStmtType(nLevel) = blstElse Then
         sConditionItem = xmDatabase.RulesNavigator.CurrentConditions.Item(nCount)
         If nStmtType(nLevel) = blstElse Then
            sConditionItem = "NOT(" & sConditionItem & ")"
         End If
         nTerm = nTerm + 1
         If nTerm = 1 Then
            sCondition = sConditionItem
         Else
            If nTerm = 2 Then
               sCondition = "(" & sCondition
            End If
            sCondition = sCondition & ") AND (" & _
               sConditionItem
         End If
      End If
   Next
```

```
          If nTerm > 1 Then
             sCondition = sCondition & ")"
          End If
       End If
```

This is a section of a listing of the output that we get from this program as a part of the description of what the variables in the datamodel look like by field name:

```
DATAMODEL Demo1

Type
   TYesNo = (Yes (1), No (5))

Locals   {These will not be output in conditions}
   I : INTEGER
AuxFields   {These will not be output in conditions}
   Introduction "This is the introductory text"
      : STRING[1]
BLOCK block1
   FIELDS {For Block 1}
      Driver "Do you drive a car or truck?"
         : TYesNo
      CarTruck "Car or truck?"
         : (Car (1), Truck (2))
   RULES
      Driver
      If Driver = Yes Then
         CarTruck
      Endif
ENDBLOCK

   FIELDS {For Main Block}
     RWilling "Is the respondant willing to take the interview?"
      : TYesNo
     Field1 : Block1  {A Block of Fields}
     FavColor "What is your favorite color?"
      : STRING[20]
     HavePet "Do you have a pet?"
      : TYesNo
     PetName "What is your Pet's Name?"
      : STRING[20]

RULES
 IF Sysdate > Todate(2000,01,01) THEN
    Introduction
    RWilling
     IF RWilling = Yes OR RWilling = DK THEN
      Field1
      FavColor
      HavePet
     Endif
```

```
  ENDIF

  IF HavePet = Yes THEN
    PetName
  ENDIF
ENDMODEL
```

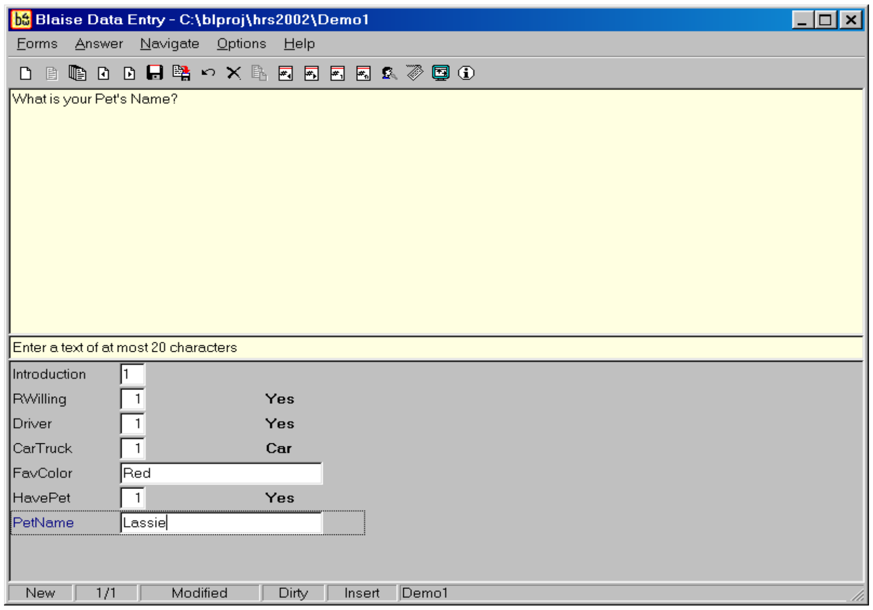

Blaise Data Entry - C:\blproj\hrs2002\Demo1
Forms Answer Navigate Options Help

Is the respondant willing to take the interview?

[✓ 1. Yes]
5. No

Introduction	1
RWilling	1
Driver	
CarTruck	
FavColor	
HavePet	
PetName	

New
1/1
Modified
Dirty
Insert
Demo1


Blaise Data Entry - C:\blproj\hrs2002\Demo1
Forms Answer Navigate Options Help

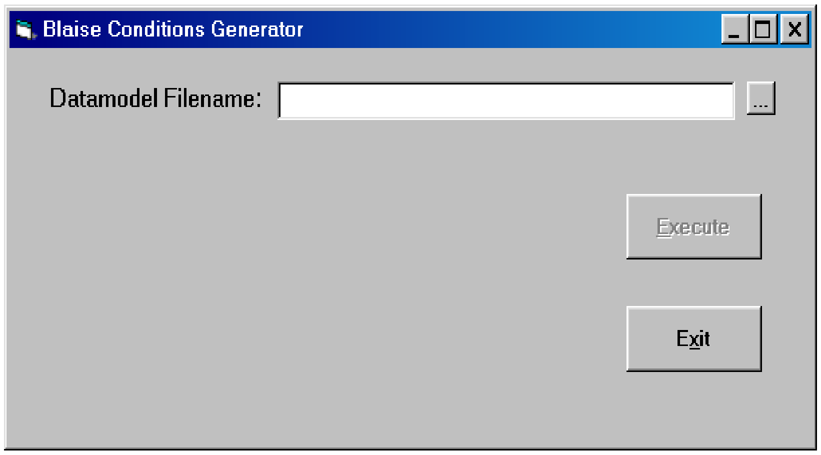
What is your Pet's Name?

Enter a text of at most 20 characters

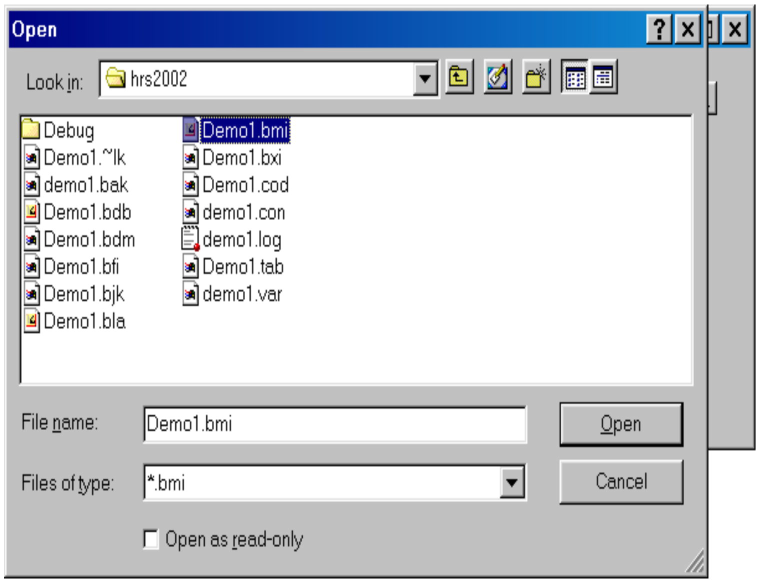
Introduction	1	
RWilling	1	Yes
Driver	1	Yes
CarTruck	1	Car
FavColor	Red	
HavePet	1	Yes
PetName	Lassie	

New
1/1
Modified
Dirty
Insert
Demo1

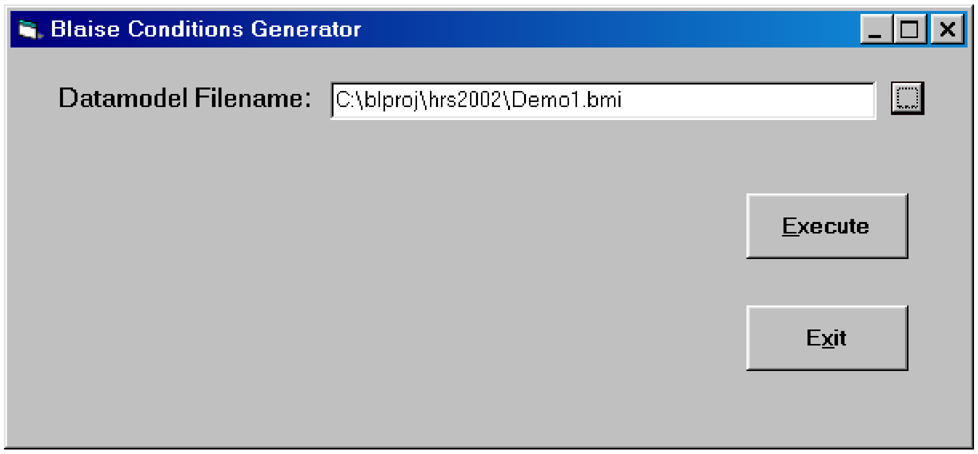
This is the screen first seen when running Universe.exe. You can type in a full path and datamodel name or use the select button to find your datamodel.



The Common Dialog box is opened and you can then select your file.



When a valid datamodel name is filled into the name field, all that is left to do is press the ‘Execute’ button and the condition file will be created. The variables table previously created by a Cameleon run will also be updated to reflect which fields are affected by what conditions.



C:\blproj\hrs2002\Demo1.bmi

This is the content of the condition file. The first item is the condition name (based on the first occurrence of a condition using field name) and the third item is the condition statements.

Name~Label~Condition
RWilling~~SYSDATE > TODATE (2000, 1, 1)
Driver~~(SYSDATE > TODATE (2000, 1, 1)) AND ((RWilling = Yes) OR (RWilling = DONTKNOW))
CarTruck~~(SYSDATE > TODATE (2000, 1, 1)) AND ((RWilling = Yes) OR (RWilling = DONTKNOW)) AND (Driver = Yes)
PetName~~HavePet = Yes

Note that only the unique conditions are listed.

The following is the contents of the variable file.

Name~Label~Object~Table~Condition~Location~Length~Type~Frame~Responses~Minimum~Maximum~Decimals~Open?~DK?~Ref?~Empty?~Text
BlaiseKey~Blaise primary key~Demo1~Demo1~~1~0~Char~~1~~~~N~N~N~N~
Demo1Instance~Block Demo1 instance number~Demo1~Demo1~~1~5~Numeric~~1~~~0~N~N~N~N~
RWilling~Is the respondant willing to take the interview?~Demo1~Demo1~RWilling~6~1~Numeric~TYesNo~1~1~5~0~N~N~N~N~Is the respondant willing to take the interview?
FavColor~What is your favorite color?~Demo1~Demo1~Driver~12~20~Char~~1~~~~N~N~N~N~What is your favorite color?
HavePet~Do you have a pet?~Demo1~Demo1~Driver~32~1~Numeric~TYesNo~1~1~5~0~N~N~N~N~Do you have a pet?
PetName~What is your Pet's Name?~Demo1~Demo1~PetName~33~20~Char~~1~~~~N~N~N~N~What is your Pet's Name?
BlaiseKey~Blaise primary key~block1~block1~~1~0~Char~~1~~~~N~N~N~N~
block1Instance~Block block1 instance number~block1~block1~~1~5~Numeric~~1~~~0~N~N~N~N~
Driver~Do you drive a car or truck?~block1~block1~Driver~6~1~Numeric~TYesNo~1~1~5~0~N~N~N~N~Do you drive a car or truck?
CarTruck~Car or truck?~block1~block1~CarTruck~7~1~Numeric~CarTruck~1~1~2~0~N~N~N~N~Car or truck?

Future Plans

In the future, we plan to use more functions out of the BCP to replace the various pieces of information that we get from Cameleon currently such as field type, enumerations, loop levels, block names as well as all of the other attributes of the datamodel.