CA 2E

Tutorial

Version 8.6.00

This documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2011 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA Technologies product documentation, complete our short customer survey, which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: About this Tutorial 19

Introduction 19
- Chapter 2 Defining Requirements 19
- Chapter 3 Data Modeling 19
- Chapter 4 Designing Functions 20
- Chapter 5 Generating, Compiling, and Executing 20
- Chapter 6 Maintaining Your Application 20
- Chapter 7 Advanced Functions 20
- Chapter 8 Report Functions 20

Related Information 20
- Getting Started 20
- Defining a Data Model 21
- Building Access Paths 21
- Building Applications 21
- Generating and Implementing Applications 21

CA 2E and the Application Development Life Cycle 22

Prerequisites for Using CA 2E 23

Chapter 2: Defining Requirements 31

Introduction to Defining Requirements 31
- Requirements Definition 31

Chapter 3: Data Modeling 35

Data Model 35
- Objectives 36
- Identify Application Entities 36
- Identify Business Relationships Between Entities 37
- Identify Attributes and Unique Identifiers for the Entities 37
- Identify the Attributes for Each Entity 38
- Select Each Entity's Unique Identifier 39
- Normalize the Entities 41
- Generalize the Entities 44
- CA 2E Data Model Diagram 45
- CA 2E Relations 46
- Objectives 46
- Overview of CA 2E Relations 47

The Edit Database Relations Panel 47
Entering Relation Statements 48
Validation of Relation Statements 49
Defining Objects 50
Specifying Object Attributes 52
File Entries 53
File Entries for COURSE 53
File Entries for RACE 54
Adding More Relations 55
Declaring More Files 56
Defining Objects 56
File Attributes 57
Deleting Relations 57
Documenting Relations 58
Field Details and Conditions 58
Objectives 58
Overview of Field Details 59
Overview of Field Conditions 59
Field Details 59
Field Detail Display for Horse Gender 59
Field Conditions 60
Adding a Field Condition 61
Field Condition Details 62
Adding Field Condition Detail 62
Adding Another Condition 63
Viewing Field Conditions 64
Specifying a Check Condition 65
Exercises 66
Displaying Selected Relations 67
Extending Relations 68
Objectives 68
Overview of Relation Extension 69
Involution and the Horse Pedigrees 69
Adding More Relations 70
Sequence of Relations 71
Extending Relations 72
Extended Relations for HORSE File 72
Example Showing Use of the Sharing Field 73
Adding Details of Extended Relations 73
Field Entries for HORSE 74
Assigning Unique Names to Field Entries 75
Showing and Hiding Relation Extension Lines 75

Virtual Fields....76
Overview of Virtual Fields....76
Specifying Virtual Fields....77
Virtual Fields for Details of a Horse's Dam and Sire....77
Sequence of Relations and Virtual Fields....78
Adding Virtual Fields....78
Virtual Field Entries....79
Selecting Virtual Fields for Dam....79
Confirming Virtual Fields for Dam....79
Adding More Virtual Fields....80
Selecting Virtual Fields for Sire....80
Confirming Virtual Fields for Sire....81
File Entries....81
Renaming Fields....82
Field Details....83
Entering a New Field Name....83
Field Details After Renaming....84
Renaming Other Fields....84
Exercise....84
Displaying File Entries Again....85
Adding Virtual Fields for the Race and Race Entry Files....85
Adding More Virtual Entries....86
Exercise....87
CA 2E Access Paths....88
Objectives....88
Overview of Access Paths....89
Default Access Paths....89
View Default Access Paths for COURSE....90
Access Paths for the HORSE File....91
Adding Virtual Fields to the Retrieval Access Path....92
Adding New Access Paths for HORSE....95
Confirming Addition of Access Paths....96
Access Path Details....96
Select/Omit Sets....97
Static and Dynamic Selection....97
Selection Conditions for the Mares Access Path....98
Select/Omit Set for Mares Access Path....99
Specifying the Conditions for the Select/Omit Set....100
Selection Access Path for Stallions....101
Access Path Details....101
Naming the Select/Omit Set....102
Specifying Conditions for the Access Path....102

Access Path Relations....103
Modifying the Default RTV Access Path for HORSE....103
Displaying Access Path Relations....104
Access Path Relations for the Retrieval Index....104
Involution and Access Path Relations....105
Changing a Referenced Access Path....105
Selecting an Access Path for Mares....106
Selecting an Access Path for Stallions....106

Chapter 4: Designing Functions 109

Introduction to Functions....109
Objectives....110
Overview of Functions....111
Default Functions....112
Default External Functions....112
Default Internal Functions....112
Defining Edit Course and Select Course....113
Displaying Default Functions For HORSE....114
Adding Functions....115
Select Mares and Select Stallions Functions....115
Creating New Functions....115
Understanding Function Details....116
Displaying Function Details....116
Device Designs....117
Overview of Device Designs....118
Displaying the Device Design for Edit Horse....118
Default Device Design Formats....119
Design Standards and Subfile Selector Options....120
Subfile Control Default....120
Subfile Record Default....120
Displaying the Rest of the Device Design....121
Editing the Default Device Design....122
Reducing the Width of the Device Design Layout....124
Folding the Device Design Layout Again....125
Panel Format Details....125
Edit Screen Format Details....126
Editing the Panel Format....127
Effect of Screen Format Changes on Device Design....128
Shortening Field Labels....129
Shortening Field Label for Dam Horse Code....130
Shortening Field Label for Sire Horse Code....130

Removing Field Labels 131
Removing Field Label for Dam Name 132
Removing Field Label for Sire Name 132
Centering a Field with Respect to its Label 134
Moving Fields to Right and Left 135
Using Function Keys 135
Using the Edit Screen Entry Details Panel 136
Adjusting the Label Spacing of the Dam Field 137
Modified Panel 138
Optional Exercise 138
Panel Format Relations 139
Editing Panel Format Relations 140
Completed Device Design 141
Exiting the Device Design 141
Exit Function Definition 142
Function Confirmation 142
Window Device Design 142
Invoking the Device Design 143
Default Device Design for Select Horse 144
Hiding Fields in the Subfile Record Format 145
Modified Subfile Control Format 146
Exercise 146
Window Options Editor 147
Changing Window Dimensions 148
Completed Device Design 148
Exiting the Device Design 149
Exercises 150
Action Diagrams 151
Overview of Action Diagrams 152
Default Action Diagram 153
Hidden Constructs and User Points 154
Edit Horse Action Diagram 154
Displaying Hidden Constructs in a Action Diagram 155
Adding Extra Validation to Edit Horse 155
The Process Response Construct 156
The Process Screen Construct 157
The Validate Subfile Record Construct 158
Adding a Validation Procedure 159
Subfile Record Relations 159
Editing the Action Diagram 160
Editing the Edit Horse Action Diagram 162
Inserting a Condition 162

Inserting a Single Action 163
Adding Another Condition by Copying 164
Specifying Details of Conditions 165
The Edit Action Condition Window 166
Contexts 166
Adding Details of the First Condition 167
Adding the Second Condition 168
Adding Details of the Second Condition 168
Adding Actions 169
Exiting the Action Diagram 169
Message Functions 171
Displaying the Message Functions 172
Adding a New Message Function 172
Defining Parameters 173
Specifying Parameters for the First Message Function 174
Message Function Details 175
Adding Text to the First Message 176
Returning to the Edit Horse Action Diagram 177
Displaying User Points for Edit Horse 178
Add Action for First Condition 179
The Edit Action Function Name Panel 179
Message Functions 180
Displaying the Message Functions 180
Selecting a Message Function 181
Function Details for the First Action 182
Action Diagram with the First Action Defined 182
Defining the Second Action 183
Adding the Second Message Function 184
Details of the Second Message Function 184
Defining Parameters for the Second Message Function 185
Message Function Details 186
Adding Message Function Text 186
Selecting the Second Message Function 187
Returning to the Action Diagram 187
Completed Action Diagram 188
Exiting the Action Diagram 188
Function Options 189
Objectives 189
Accessing the Function Options Panel 190
Default Function Options 191
Changing Function Options 191
Linking Functions 192

Objectives....193
Steps Required to Link Functions....194
Modifying the Edit Horse Action Diagram....194
Obtaining the Action Diagram User Points....195
Adding a New CASE Construct....195
Inserting a Condition....196
Entering the Condition....196
Entering the Condition Details....197
Selecting the Zoom#1 Condition....197
Specifying a Function as the Action....198
Naming the Function....199
Selecting a Function....199
Creating the New Function and Access Path....200
Selecting an Access Path....200
Creating a New Access Path....201
Specifying the Access Path Details....202
Defining the Access Path Key....203
Selecting the Access Path....204
Function Parameters....204
Objectives....205
Understanding Parameter Usage and Role....205
Specifying Function Parameters....206
Defining Function Parameters....207
Specifying Parameters Using an Access Path....207
Defining Parameter Details....208
Completing the Parameter Details....209
Selecting the New Function....209
Returning to the Edit Action - Function Details Window....210
Suppressing the Confirm Prompt and File Update....211
Inserting *MOVE as an Action....212
Specifying Parameters for the *MOVE Function....212
Specifying Defer Confirm....213
Reload Subfile....214
Specifying Reload Subfile....215
The Completed Action Diagram....215
Saving the Action Diagram....216
Updating the Edit Horse Function's Action Bar....216
Updating the Edit Horse Panel's Zoom Action Text....217
Working with the Selector Choice....217
Modifying an Action....218
Changing the Action Text....218
Showing the New Action Text....219

Exiting the Action Bar Editor 220
Exiting the Modified Panel 220
Saving the Modified Panel 220
Edit Database Relations Panel 221
Exercise 222

Chapter 5: Generating, Compiling, and Executing 223

Implementing Access Paths and Functions 223
Objectives 223
Overview of Implementation 224
Source Generation and Compilation of Access Paths and Functions 224
Display all Access Paths for Selection 225
Selecting All Access Paths for Batch Generation and Compilation 225
Completing the Request 226
Display all Functions for Selection 227
Selecting All External Functions for Batch Generation and Compilation 228
Completing the Request 229
Submit Batch Generation and Creation 230
Generating and Creating Objects 230
List of Objects to be Generated and Created 231
Confirming the Job List 232
Successful Submit for Generation and Compilation 233
Examining the Job List 233
Converting Condition Values to a Database File 235
YCVTCNDVAL Command Prompt 236
Confirming Conversion of Condition Values 237
Exiting the CA 2E Design Model 237
Re-synchronizing an CA 2E Design Model 238
Executing and Testing Compiled Programs 238
Calling the Program 239
The Edit Horse Function Panel 240
Adding Data to the HORSE File 241
Confirming Data Entries for the Horse File 242
Switching from New to Open Mode 243
Exiting the Edit Horse Program 244
Exercises 244

Chapter 6: Maintaining Your Application 245

Application Maintenance 245
Animating an Interactive Device Design 246
Overview of CA 2E Animation 246

Animate the Edit Horse Device Design....247
Animate Functions Panel....248
Animating Edit Horse....249
Entering Sample Data for Edit Horse....250
Editing and Maintaining Multiple Functions....251
Open Functions Panel....252
Animating the Display Racing Results Function....253
Converting Command Key Navigation....253
Working with Toolkit Panel Designs....254
Toolkit Command Key Navigation....255
Reassigning a Command Key in Toolkit....256
Setting Up Action Bar Navigation for Edit Horse....257
Assigning Action Bar Navigation....258
Testing the Function Link....261
Animating Edit Horse....262
Activate the Action Bar....262
Display Racing Results Panel Design....263
Exiting Both Functions....265
Working with Model Object Lists....266
Objectives....266
Overview of Model Objects and Model Object Lists....266
Editing Your Session List....268
Edit Model Object List Panel....269
Viewing Model Object Types....269
List Entry Differs from Model Object....270
Positioning a Model Object List....271
Returning to the Top of the List....273
Viewing a Subset of a Model Object List....274
Opening Multiple Functions at One Time....275
Working with Open Functions....276
Displaying the Unsubsetted Model List....277
Editing Model Objects....277
Exercise....278
Editing Conditions for Entry Status....278
Exercise....279
Displaying the Unsubsetted Model List....279
Editing Relations and Creating Objects....279
Deleting a Model List Entry....281
All Objects List....282
Accessing the All Objects List....283
How *ALLOBJ and Model Lists Differ....283
Displaying Alternate Views of Detail Information....284

Exercise 284
Restoring the Deleted Session List Entry 285
Accessing Other Model Lists 285
Returning to Your Session List 286
Working with Model Object Lists 286
Function Versioning 288
Objectives 288
Overview of Versions 288
Using Versions to Update an Existing Function 289
Accessing the Session List 290
Positioning to the Edit Horse 291
Creating a Version for Edit Horse 292
Working with Versions 293
Naming the Version 294
Implementation Name for the New Version 294
Editing the New Function 295
Inserting a Message Function 296
Specifying a Parameter for the Message Function 297
Editing the Message Text 298
Using a Substitution Variable 299
Selecting the New Message 299
Checking the Default Parameters 301
Returning to the Action Diagram 301
Submitting the Function Version for Generation 302
Viewing Job List Commands 303
Displaying Alternate Views 304
Testing Edit Horse - Version 1 304
Comparing Two Versions of a Function 306
Entering Parameters for YCMPMDLOBJ 306
Viewing Differences between Versions 307
Making the Edit Horse - Version 1 Current 308
Naming the New Current Version 309
Regenerating the New Current Version 310
Exercise 310
Viewing Model Object Information for the Versions 311
Refreshing List Entries for the Versions 312
Exercise 313
Model Object Cross References 313
Objectives 313
Overview of Model Object Cross References 313
Model Object Usages 314
Accessing Your Session List 314

Display Model Usages Panel 316
Usage Reason 317
Exercise 317
Using Usage Levels 318
Exiting Model Usages 318
Model Object References 318
Accessing the *ALLOBJ List 318
Positioning *ALLOBJ to an Implementation Name 319
Displaying References for Edit Horse 320
Display Model References Panel 321
Displaying Only External Functions 321
Creating a Model List of the References 322
Naming the List of References 323
Using the List of References 323
Exiting Display Model References 323
Redisplay Your Session List 324
Impact Analysis 324
Objectives 324
Overview of Impact Analysis 325
Change Type 325
Simulating a Change to a Model Object 325
Positioning the List to Course Code 326
Positioning the Usages to Course Code 327
Simulating a *PUBLIC Change 327
Interpreting the Results 328
Converting the Simulation Usages to a List 329
Optional Exercise 329
Component Change Processing 330
Accessing Edit Database Relations 330

Chapter 7: Advanced Functions 331

Span Access Path and Edit Transaction Function 331
Objectives 331
Overview 332
Overview of Processing Steps 332
Entering Your Design Model 333
Creating a Span Access Path 333
Defining the Span Access Path 334
Entering the Access Path Formats 335
Selecting the First Access Path Format 336
Confirming Selection of the First Format 336

Specifying the Second Format....337
Adding Virtual Fields to the Access Path Formats....338
Details for the First Access Path Format....340
Specifying the Key for the First Access Path Format....341
Details for the Second Access Path Format....342
Requesting Batch Generation of the SPN Access Path....343
The Edit Transaction Function....343
Objectives....343
Defining the Edit Transaction Function....344
The Default Device Design....345
The Modified Device Design....346
Defining Optional Entry Fields....347
Introduction to Function Fields....349
Objectives....349
Overview of Function Fields....349
Adding Function Fields....350
Naming the Function Field....351
Displaying Existing Fields....351
Defining Two New Function Fields....352
Overriding the Field Defaults....353
Displaying the Function Fields....353
Defining Function Field Parameters....354
Selecting the Function Field....355
Accepting the Parameters....356
The Modified Device Design....356
Defining a Function Field Action Diagram....357
Adding a CASE Construct....359
Adding an Action....360
Adding a Second Condition....360
Adding a Second Action....361
Defining the Conditions and Actions....361
Specifying the First Condition....362
Defining Field Conditions for the Status Field....362
Specifying the First Action....363
Specifying the Second Condition....364
Defining the Second Action....365
Complete Action Diagram....366
Returning to the Device Design....368
Adding the Second Function Field....368
Specifying the Second Function Field....369
Defining Parameters for the No. of finishers Function Field....370
The Added Function Field....370

Readjusting the Device Design 371
Exiting the Device Design 372
Submitting the Function for Generation 372
Exercises 374

Chapter 8: Report Functions 375

Introduction to the Print File Function 375
Objectives 376
Defining the Print File Function 376
Specifying a New Access Path 377
Adding Virtual Entries to the Access Path 378
Edit Access Path Details 379
Specifying Access Path Details 379
The New Key Order 380
Generating and Compiling the New Access Path 380
Selecting the Access Path 381
The Report Design 381
The Default Report Layout 382
Displaying the Report Formats 383
Dropping Formats from a Report Design 384
Adding Function Fields 384
Displaying Existing Function Fields 385
Defining New Function Fields 387
Specifying Function Parameters 388
The Default Parameters 389
Selecting the Function Fields 390
Confirming the Function Details 391
Adding the Total Number of Horses Function Field 391
Confirming the Parameters 392
Exercise 393
The Report Design with Function Fields 393
Completing the Report 393
The Modified Report Device Design 394
The Completed Report Layout 395
Saving the Report Device Design 396
Generating and Compiling the Function 396
Running Your Program 397
Introduction to the Print Object Function 397
Objectives 398
Defining the Print Object Function 398
Adding a Print Race Entries Function 399

Specifying a Query Access Path....400
Displaying the Access Path Format Entries....401
Specifying an Alternative Key....402
Compiling the QRY Access Path....402
Selecting the Access Path for the Function....403
Displaying the Report Device Design....403
The Default Report Device Design....404
Exercise - Updating the Report Layout....405
Exiting the Device Design....405
Combining the Report Functions....406
Embedding the Print Race Entries Function....407
Selecting the PRTOBJ Function....408
The Modified Device Structure....408
Saving the Device Structure....409
Displaying the Report Device Design....409
The New Report Device Design....410
View the Rest of the Report Device Design....410
Modifying the Combined Report Device Design....411
Specifying Parameters....412
Editing the Action Diagram....413
The Print Details Construct....413
Editing the Print Race Entries Function....414
The Action Diagram Editor Subfile Selector Values....414
Editing the Function....415
Displaying the Function Parameters....415
Editing the Function Parameters....416
Specifying a Restrictor Parameter....416
Confirming Parameter Details....417
Exit Action Diagram....418
Generating and Compiling the Functions....418
Testing the Program....419

Glossary **421**

Index **433**

Chapter 1: About this Tutorial

This tutorial provides a navigational guide to the new CA 2E user. It will guide you through the steps required to develop an application using CA 2E. After completing this tutorial, you will have designed a CA 2E working application. The tutorial includes all of the basic facilities of CA 2E as well as some advanced features.

This section contains the following topics:

Introduction (see page 19)
Related Information (see page 20)
CA 2E and the Application Development Life Cycle (see page 22)
Prerequisites for Using CA 2E (see page 23)

Introduction

This chapter describes the organization of the tutorial, sources for additional information about CA 2E, conventions used in the tutorial, the application development life cycle, guidelines for getting started with the tutorial, and a glossary of terms introduced in the tutorial.

Note: Be sure to read this chapter before beginning the tutorial.

The following sections summarize each chapter of the tutorial.

Chapter 2 Defining Requirements

Determines the business requirements of the application to be developed. Describes how to determine the data required from the business requirements. Applies the rules of normalization and generalization to the list of data elements to develop an entity relationship diagram.

Chapter 3 Data Modeling

Describes how to translate the entity relationship diagram into a CA 2E data model diagram. Describes how to enter files and field details, field validation, and default access paths created by CA 2E. Introduces involution. Describes how to create alternate access paths that use selection criteria and how to join files together.

Chapter 4 Designing Functions

Introduces CA 2E functions (Edit File, Select Record) and how to design these in CA 2E.

Chapter 5 Generating, Compiling, and Executing

Describes how to generate source to implement the access paths and functions created in the first four chapters and how to compile the source to produce executable i OS objects - files and programs.

Chapter 6 Maintaining Your Application

Presents tools to help you maintain a CA 2E generated application, including prototyping, working with model object lists, and creating function versions.

Chapter 7 Advanced Functions

Describes the use of the Edit Transaction function using the Span access path.

Chapter 8 Report Functions

Describes how to produce a basic report program from the CA 2E design model.

Related Information

This module provides only the information about CA 2E that is necessary to complete the Tutorial. Additional information can be found in the following list of CA 2E manuals.

Note: CA 2E panels provide extensive online help; simply press the Help key on your keyboard to display more information for the current panel. Be sure to use this facility as you do the tutorial.

Getting Started

This module describes the libraries that make up CA 2E and the development environment. This module also outlines the installation and upgrade process.

Defining a Data Model

This module provides instructions on using CA 2E to design and maintain a data model.

Building Access Paths

This module provides instruction on how to build, modify, delete and document access paths and how to create arrays.

Building Applications

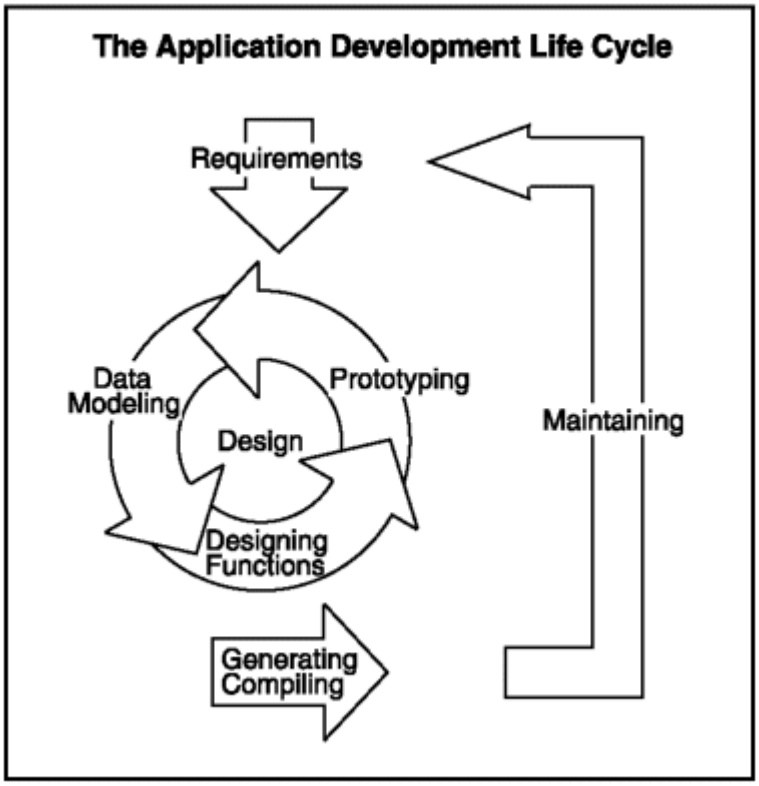
This module provides instructions on building functions in CA 2E. The module tells you how to set up system default values, build functions, edit device designs and action diagrams, and how to generate and compile functions.

Generating and Implementing Applications

This module describes model objects, model object lists, and impact analysis and provides instructions and recommendations to successfully generate, compile and implement a CA 2E application.

CA 2E and the Application Development Life Cycle

CA 2E supports the use of a top down, structured, data-driven application development life cycle. The application development life cycle is illustrated in the following diagram.



The application development life cycle provides direction during the design and generation of a CA 2E application. This tutorial provides instructions to complete each of the stages listed in the diagram.

1. Determine the application requirements.
2. Describe data in the form of a CA 2E data model. The data model is the basis for everything designed within CA 2E.
3. Design the functions that will operate on your data.
4. Prototype your functions to test your design.
5. Return to your design model to specify additional functionality.
6. Generate and compile your application.

Prerequisites for Using CA 2E

Before starting this tutorial, make sure the following conditions are satisfied:

- Installation

  CA 2E must be installed on your IBM i. This manual assumes that installation has been successfully completed. If not, please refer to the *Installation Guide* for additional details.

- Authorization

  You must be signed on an IBM i with a user profile that is authorized to use CA 2E. You must also be authorized to carry out programming operations such as creating files and RPG or COBOL programs. If you do not have the required profile or authorizations, consult your security officer. Refer to the *Installation Guide* for more details. To begin, you will require access to either the Command entry (QCMD), the Programmer's menu (QPGMMENU), or the CA 2E Main menu.

- Library List

  Your library list must include certain libraries before you can create a CA 2E design model; for example, the CA 2E and Toolkit product libraries. Refer to the worksheet that was filled out when CA 2E was installed at your company for a list of the required libraries. The worksheet is in the *Installation Guide*. You can add any missing libraries using the i OS Add Library List Entry (ADDLIBLE) command. For example,

```
ADDLIBLE Y2SY
ADDLIBLE Y1SY
```

- Creating a CA 2E Design Model

  You must have a CA 2E design model library in which to store your CA 2E design model. If you do not already have a design model you may create a new one using the CA 2E Create Model Library (YCRTMDLLIB) command as follows:

```
YCRTMDLLIB MDLLIB(MYMDL) OBJPFX(MY)+
 SYSTEXT('My Model') DSNSTD(*CUATEXT)
```

  This command allows you to create your model interactively. You can also do this as a batch process. In this example the design standard, *CUATEXT, has been specified so that all functions that include interactive panel displays will be presented with either Action Bars or Windows. The normal default for the design standard is *CUAENTRY.

  Note: If someone else creates your model, be sure they read the above paragraph and set the DSNSTD parameter to *CUATEXT; otherwise, the tutorial will not work properly.

  The message "Model library MYMDL created" should appear following successful completion of the command. The use of this command is only necessary when you are creating a new CA 2E design model. The next topic discusses how to access an existing CA 2E design model.

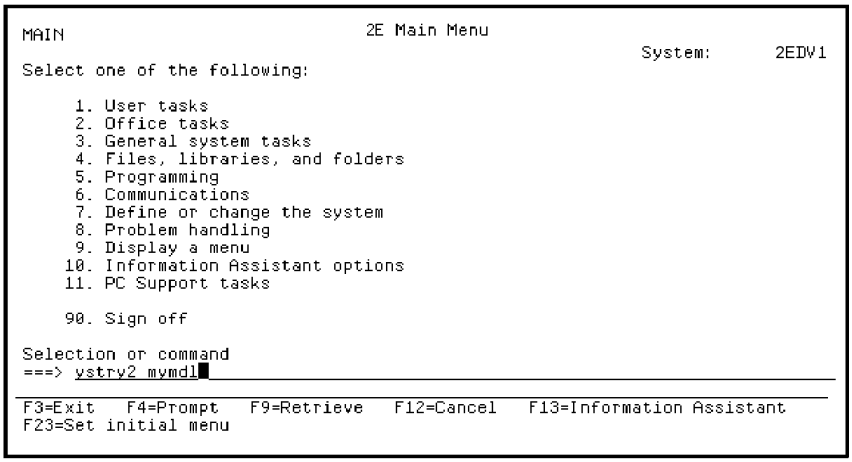
The YCRTMDLLIB command also creates a library to contain the source and application objects that you will generate from your CA 2E design model. The name of this library is given by the GENLIB parameter, which in this case defaults to the name MYGEN. Furthermore it creates a Toolkit Library list with the same name as the CA 2E design model library.

- Entering the CA 2E Design Model

  To enter CA 2E, type the Start CA 2E (YSTRY2) command, specifying the name of your CA 2E design model as the LIBLST parameter from any i OS Command Entry line:

YSTRY2 MYMDL

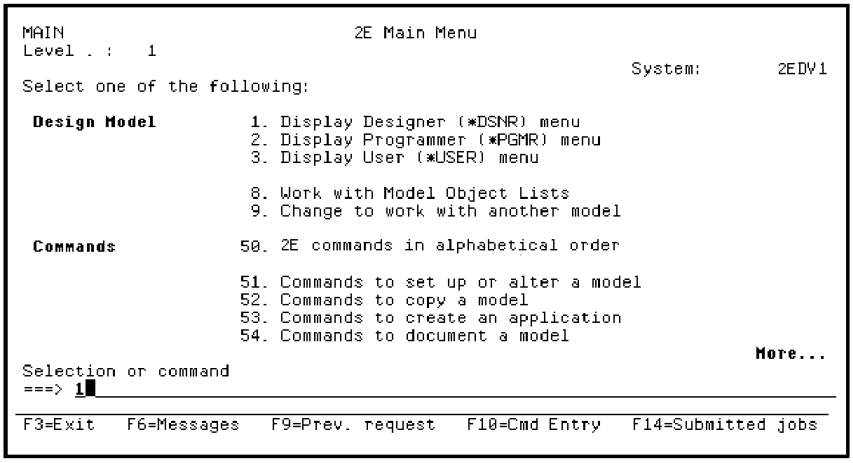
Type the command as shown.



Press Enter.

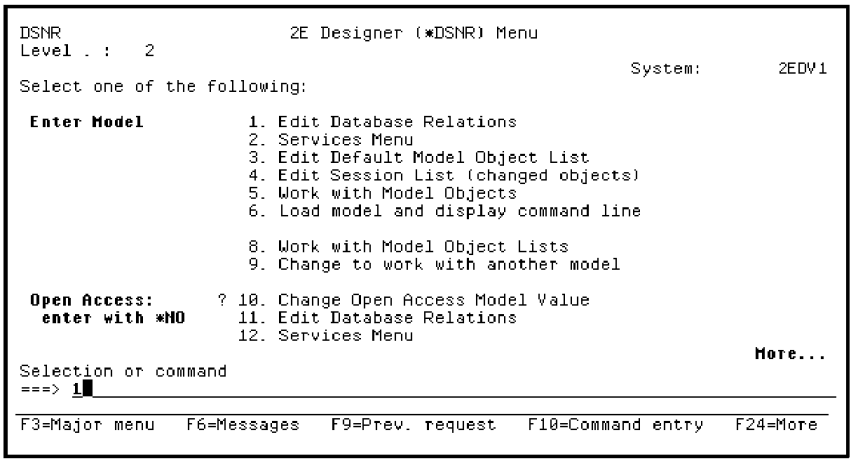
The CA 2E Main menu appears. The first panel of the CA 2E Main Menu provides a set of Design Model options and access to CA 2E commands grouped according to function.

Type **1** in the command entry line as shown to display the CA 2E Designer (*DSNR) Menu.



Press Enter.

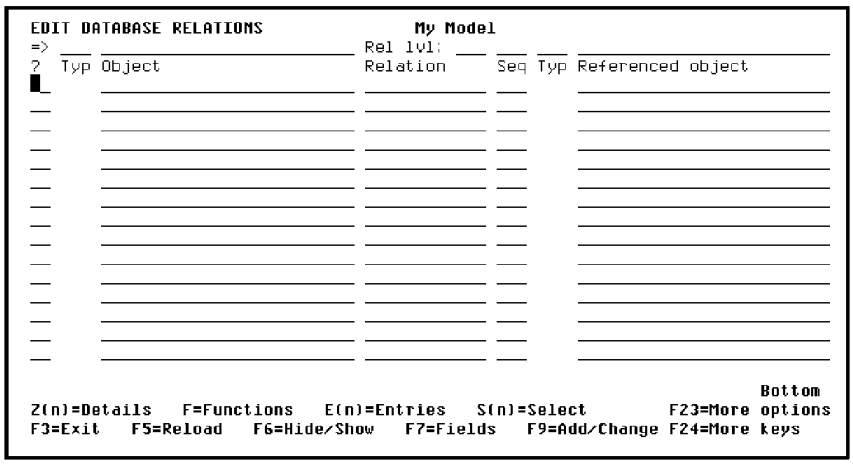
The first panel of the CA 2E Designer (*DSNR) Menu displays. This menu provides a list of tasks available to users with *DSNR authority. Type **1** in the command entry line as shown to enter your CA 2E design model for editing.



Press Enter.

Your CA 2E design model will now be loaded. A CA 2E window with the message Starting CA 2E Session appears. The load process may take several seconds.

When it is finished the Edit Database Relations panel will appear.



In general, you can begin working with your model at this point. However, if this is the first time you entered your tutorial model, read and complete the following instructions before continuing.

- Other Methods of Entering Your Design Model

  Following are two alternative methods of entering your design model.

  - Go directly to the CA 2E Designer (*DSNR) Menu using the Menu parameter on the Start CA 2E (YSTRY2) command from any i OS Command Entry line as follows:

YSTRY2 MYMDL MENU(DSNR) Bypass the Designer (*DSNR) Menu and directly enter your model for editing using the Toolkit Change Library List (YCHGLIBL) command and the Edit Model (YEDTMDL) command from any i OS Command Entry line.

YCHGLIBL MYMDL

YEDTMDL

The Y2 command is a short form of the YEDTMDL command.

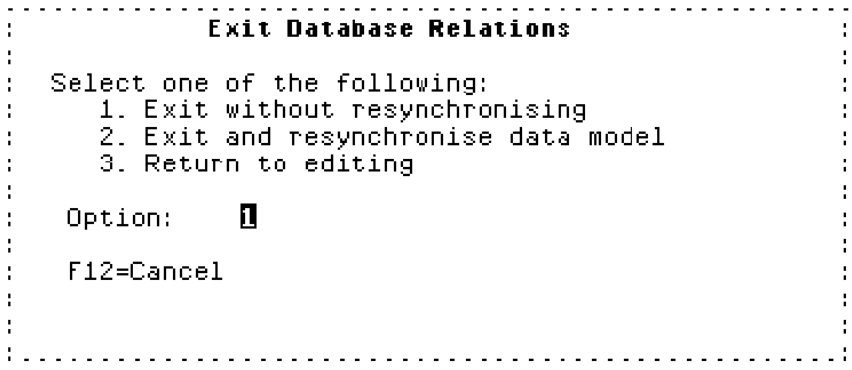
YCHGLIBL MYMDL

Y2

- Leaving the CA 2E Design Model

  It is unlikely that you will complete the tutorial in a single session. If at any time you want to leave your CA 2E design model, you can do so by pressing F3 repeatedly until you return to the Edit Database Relations panel

  At that point, press F3 again to display the Exit menu. Press F3 now for practice.



Normally when you exit your design model, you will accept the default option and press Enter. However, if you plan to continue with the tutorial, type **3** instead of accepting the default to continue your editing session and press Enter. This returns you to the Edit Database Relations panel.

- Setting Your Model Profile to Log Changes to Your Model

  Your model profile establishes the basic working environment for your interactive session. CA 2E automatically creates a model profile for each user that is granted access to a model. Before beginning work on your CA 2E model, you need to check, and possibly override, the default settings in your model profile.

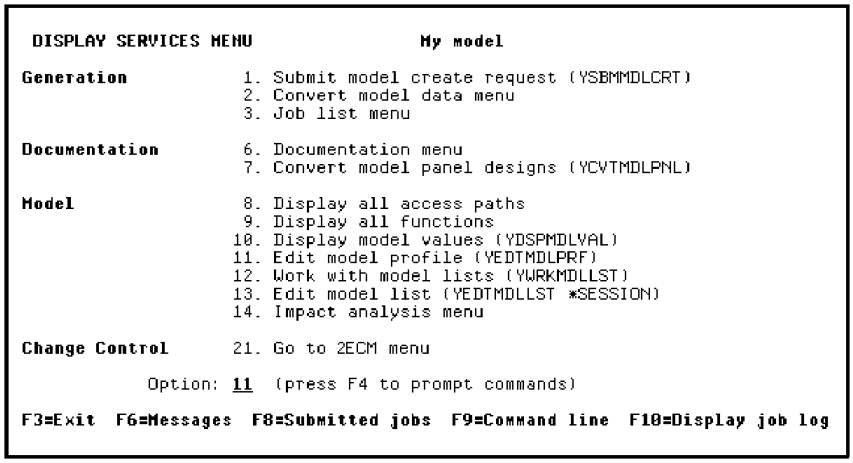
  Note: You need to do this step before you enter any information for your tutorial design model and you only need to do it once.

  This process assumes that the Edit Database Relations panel is displayed on your screen. If not, re-enter your model as explained previously using the Start CA 2E (YSTRY2) command.

- Accessing the Display Services Menu

  One way to access your model profile is by using the Display Services Menu. This menu is an important CA 2E tool that gives you access to many CA 2E support functions while you are working on a model. From the Edit Database Relations panel, press F17 to access the Display Services Menu This function key is available from many CA 2E panels.

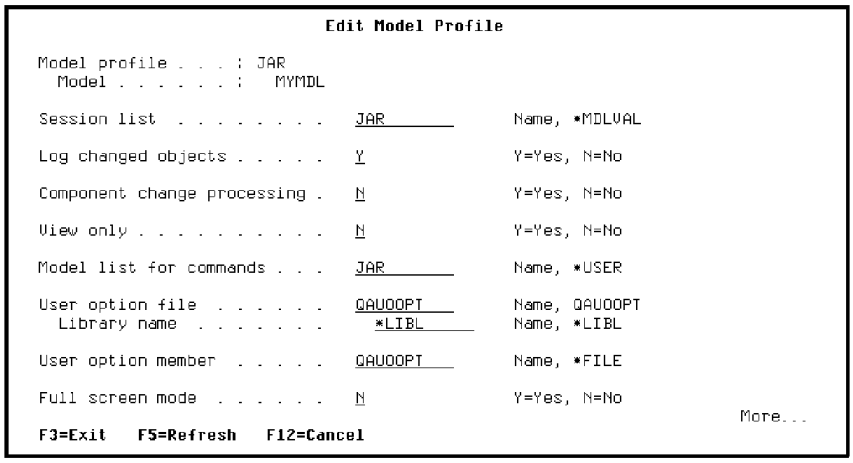
  From the Display Services Menu, type **11** in the Option field to Edit your model profile.



- Editing Your Model Profile

  Press Enter to display the first screen of the Edit Model Profile panel.

  The model profile lets you define defaults for various processes and file specifications for an interactive session. Notice that the default value for the Session list field is your user profile name. Also, check that the value of the Log changed objects field is **Y**. If it is not, type **Y**.



Press Enter.

Press F3 to return to the Display Services Menu.

- Understanding a Session List

  The basic components of a CA 2E model are called *model objects*. CA 2E provides a model object list tool that lets you manipulate logical groups of model objects. One example of a model object list is the *session list*. When the Log changed objects field in your model profile is Y, CA 2E automatically keeps track of each change you make to your CA 2E model and adds the changed model object to your session list. The session list is cumulative and therefore it persists across your interactive sessions.

  Notice the Edit model list (YEDTMDLLST *SESSION), option on the Display Services Menu. You can use this option at any time to view and work with the contents of your session list; in other words, you can use this option to view the model objects you changed in your CA 2E model.

  You will use the session list again later in the *Maintaining Your Application* chapter of this tutorial.

- Exiting the Display Services Menu

  When you finish using the tasks on the Display Services Menu, exit by pressing F3 to return to the Edit Database Relations panel.

- Deleting User-defined Data from your CA 2E Design Model

  This step applies only if you wish to delete the data you entered from your CA 2E design model. To do so use the CA 2E Clear Model (YCLRMDL) command as follows:

```
YCLRMDL MDLLIB(MYMDL) GENLIB(*GENLIB)
```

If you run this command, your CA 2E design model will still exist, but everything you have added to the design model will be deleted.

Chapter 2: Defining Requirements

This chapter gives the business requirements and describes the data and functions required to support the Horse Racing application.

This section contains the following topics:

Introduction to Defining Requirements (see page 31)

Introduction to Defining Requirements

This tutorial demonstrates how to use CA 2E to address all phases of the life cycle in the development and maintenance of a horse race application. A sample horse racing form is illustrated below.

NEWMARKET **11Aug 1995**

1:30 NEWMARKET STEEPLECHASE
Going: Heavy 2m 5f
Prize of $6000 for five year olds and older that have never won a race
Weight: 135 lbs

			Age	Lbs	
1	500-02	FAITHFUL DOBBIN (11) (Mrs. Monica Hackett)			
		W Harney	7	130	—C F Swan
2	3523/2	PEGASUS (16) (D V Wakefield)			
		Gerard Stack	6	130	—H Rogers
3	2313/OF	LEFT BANK (3) (Mrs. S Lait)			
		Noele Meade	7	130	—D H O'Connor
4	3211-03	BONFIRE (Mr. A Mishie)			
		A Mishie	7	132	—P D Hickey
5	000-02	COOPER TRIP (Mrs. W E Fletcher)			
		W E Fletcher	6	134	—E F Hutton
6	40- 0	LARKSPUR LANDING (Ms Greenbaum)			
		A C Smith	6	128	—C J Reynolds
7	6/3 P	BLUSWAD SHUZ (E A Presley)			
		P Presley	7	135	—A A Memphis

Seven runners
Last year: Gorky Park, 5-11-7, C F Swan, 5-1 (B Hughes) , 11 ran

Requirements Definition

Initial requirements for the application illustrated in this tutorial include the form from the preceding page. Management requires a system that tracks the horses and jockeys entered in races. In addition, they want to track the parentage and racing results of each horse and require certain reports.

Business Requirements

After analysis of the form and further discussion with management, it is determined that the application requires the following:

- The business objects or entities: Course, Horse, Jockey, and Race. These will become files.
- A record of the Dam (mother) for each horse to ensure that it is female and older than the horse.
- A record of the Sire (father) for each horse to ensure that it is male and older than the horse.
- A record of horses entered in each race.
- The following two reports:
  - A list of all horses including a count and total value
  - A list of the races each horse has entered

Data Requirements

Having identified the business requirements, it is necessary to examine them and determine the data needed to meet the requirements.

Analyze the form carefully to make a list of all the data needed to create the form. Do not be concerned about defining the data as a grouping of data (entity/file) or a property of the group (attribute/field). Some examples are shown here:

- Jockey
- Horse name
- Horse code
- Entry number
- Finishing position
- Race name
- Race distance
- Courses
- Course name
- Race time
- Going conditions
- Prize money
- Race date
- Dam (Horse's mother)
- Sire (Horse's father)
- Horse's date of birth

Having specified the requirements of this application, you can now begin to build the CA 2E data model. This data model will serve as the foundation for the application design in CA 2E. The data modeling process is illustrated in the *Data Modeling* chapter.

Chapter 3: Data Modeling

This chapter introduces the following topics:

- Data Model
- CA 2E Relations
- Field Details and Conditions
- Extending Relations
- Virtual Fields
- Access Paths

This section contains the following topics:

Data Model (see page 35)
Field Details and Conditions (see page 58)
Extending Relations (see page 68)
Virtual Fields (see page 76)
CA 2E Access Paths (see page 88)

Data Model

In this topic you will build the CA 2E data model that supports the requirements of the horse race application.

New terms introduced:

- Entity
- Relationship
- Entity Relationship Diagram
- Attribute
- Unique Identifier
- Primary key
- Foreign key
- Normalization
- Generalization
- CA 2E Relation
- CA 2E Data Model Diagram

Objectives

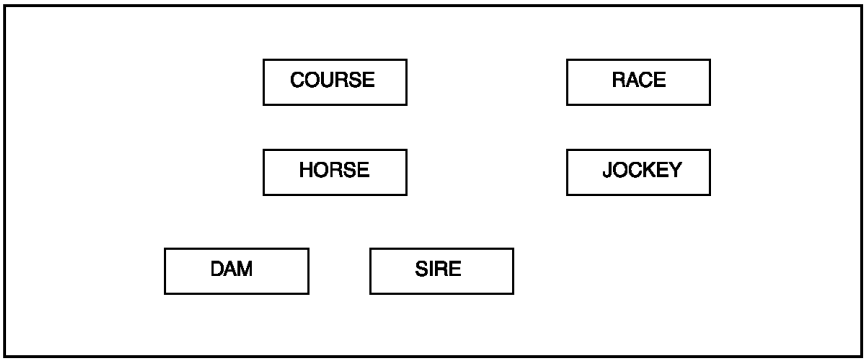
Use an Entity Relationship Diagram (ERD) to design the horse race application data model. Normalize and generalize the entities to refine the ERD. Translate the ERD into CA 2E relations. The following steps explain how to achieve these objectives.

Identify Application Entities

The first step in entity relationship modeling is to identify the entities (files) that are present in your application using the data elements you identified during your requirements analysis. An *entity* is a thing or object of significance for which you need to gather and store information.

Each entity must be uniquely identifiable. This means that each instance (occurrence) of an entity must be separate and distinctly identifiable from all other instances of that entity. For example, for the entity, Horse, each individual horse (instance) must be uniquely identifiable in the data model.

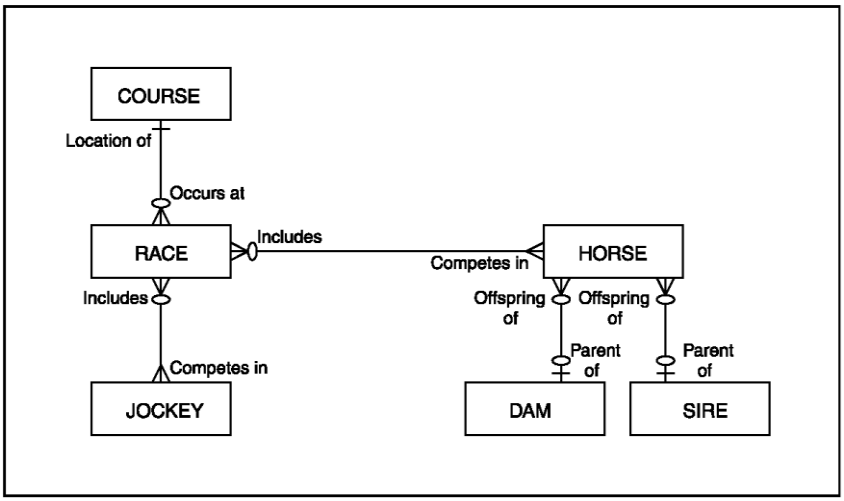
Each entity can be represented in a diagram by a box containing the name of the entity. The name is in singular and shown in all uppercase letters. This tutorial includes the following entities:



Identify Business Relationships Between Entities

The second step in entity relationship modeling is to identify the business relationships that exist between the entities. A business relationship (or *relationship* for short) is a named, significant association between two entities.

The relationships present in the CA 2E data model at this point can be represented as shown.



Identify Attributes and Unique Identifiers for the Entities

The third step of entity relationship modeling involves the following:

- Identifying the attributes (fields) that belong to each entity
- Selecting the attributes that serve as a unique identifier for each entity
- This unique identifier is also known as the primary key for the entity.
- Normalizing the entities
- Generalizing the entities

The following sections discuss each of these in more detail.

Identify the Attributes for Each Entity

An *attribute* is any detail that serves to qualify, identify, classify, quantify, or express the state of an entity. An attribute can also be any description of a thing of significance. The attributes must describe the entity against which they are shown.

Reviewing the data elements you identified during requirements analysis, and taking into consideration the entities you have already identified, it is possible to identify the attributes. List the attributes beside the appropriate entity as in the following table:

Entity	Attribute
COURSE	Course code Course name
HORSE	Horse code Horse name Horse gender Date of birth
JOCKEY	Jockey code Jockey name Jockey gender
SIRE	Sire code Sire name Sire Date of birth
DAM	Dam code Dam name Dam Date of birth
RACE	Course code Course name Race date Race time Race name Going conditions Distance Prize money Horse code Horse name Horse gender Date of birth Jockey code Jockey name Finishing position Handicap Entry status

Select Each Entity's Unique Identifier

Each entity must be uniquely identifiable so that each instance of the entity is separate and distinctly recognizable from all other instances of that entity. The *unique identifier*, also known as the *primary key*, can be any of the following:

- An attribute
- A combination of attributes
- A combination of relationships
- A combination of attributes and relationships

A primary key should consist of the fewest number of attributes necessary to make every instance of the entity unique. Review the attributes of each entity and select the primary key for each entity. Primary keys are identified in the following table:

Entity	Attribute	Primary Key
COURSE	Course code Course name	Course code
HORSE	Horse code Horse name Horse gender Date of birth	Horse code
JOCKEY	Jockey code Jockey name Jockey gender	Jockey code
SIRE	Sire code Sire name Sire Date of birth	Sire code
DAM	Dam code Dam name Dam Date of birth	Dam code

RACE	Course code Course name Race date Race time Race name Going conditions Distance Prize money Horse code Horse name Horse gender Date of birth Jockey code Jockey name Finishing position Handicap Entry status	Course code Race date Race time

Identify Foreign Keys

Note that the RACE entity contains the primary keys of the HORSE (Horse code) and JOCKEY (Jockey code) entities. The Horse code and Jockey code are needed as attributes in the RACE entity to identify the horses and jockeys that competed in each race. However, neither the Horse code nor the Jockey code are needed to identify the RACE entity; in other words, neither Horse code nor Jockey code are primary keys of RACE. Such non-key attributes of an entity that are primary keys of another entity are known as *foreign keys*. In the following table, foreign keys are identified.

Entity	Attribute	Primary Key	Foreign Key
COURSE	Course code Course name	Course code	N/A
HORSE	Horse code Horse name Horse gender Date of birth	Horse code	N/A
JOCKEY	Jockey code Jockey name Jockey gender	Jockey code	N/A
SIRE	Sire code Sire name Sire Date of birth	Sire code	N/A
DAM	Dam code Dam name Dam Date of birth	Dam code	N/A

RACE	Course code Course name Race date Race time Race name Going conditions Distance Prize money Horse code Horse name Horse gender Date of birth Jockey code Jockey name Finishing position Handicap Entry status	Course code Race date Race time	Horse code Jockey code

Normalize the Entities

Normalization of data is a procedure that ensures that a data model conforms to some useful standards.

For data and entity relationship models, the following standards have been defined to minimize duplication of data, to provide the flexibility necessary to support different functional requirements, and to enable the data model to be mapped onto a relational database design effectively.

- Eliminate repeating data groups
- Eliminate partial key dependencies
- Eliminate non-key dependencies (attributes that depend upon another non-key attribute in the entity)

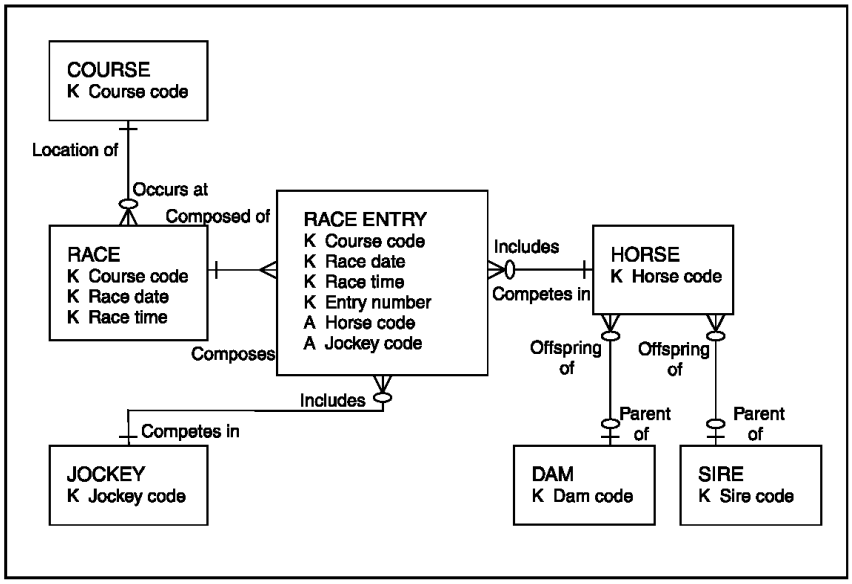
Although the steps are listed and briefly discussed here, you may wish to consult additional sources for a more detailed description.

Applying these steps causes the entities and attributes to be redefined as follows:

Entity	Attribute	Primary Key	Foreign Key
COURSE	Course code Course name	Course code	N/A

HORSE	Horse code Horse name Horse gender Date of birth	Horse code	N/A
JOCKEY	Jockey code Jockey name Jockey gender	Jockey code	N/A
SIRE	Sire code Sire name Sire Date of birth	Sire code	N/A
DAM	Dam code Dam name Dam Date of birth	Dam code	N/A
RACE	Course code Race date Race time Race name Going conditions Distance Prize money	Course code Race date Race time	N/A
RACE ENTRY	Course code Race date Race time Entry number Horse code Jockey code Finishing position Handicap Entry status	Course code Race date Entry number	Horse code Jockey code

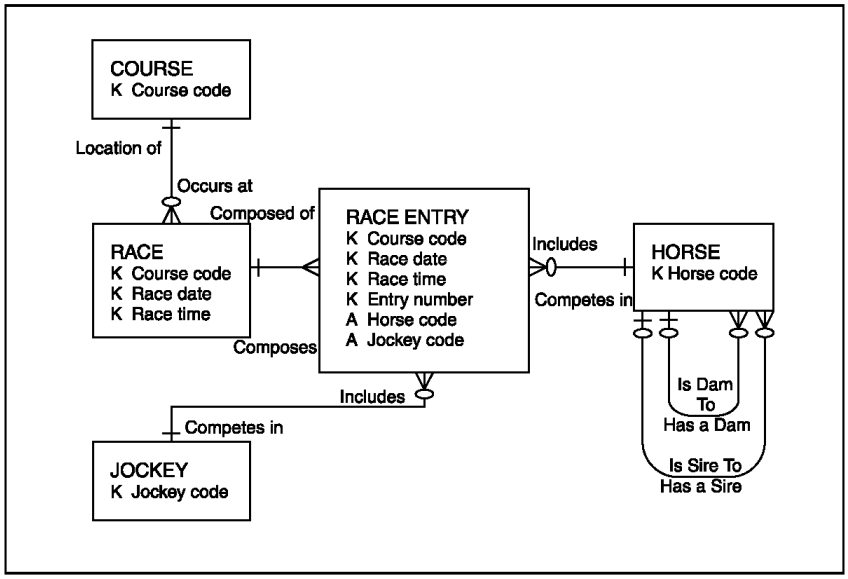
This process has resulted in the creation of another entity named RACE ENTRY. Now our Entity Relationship Diagram looks like this. Note that *K* indicates a key attribute and *A* indicates a foreign key (non-key attribute).



Generalize the Entities

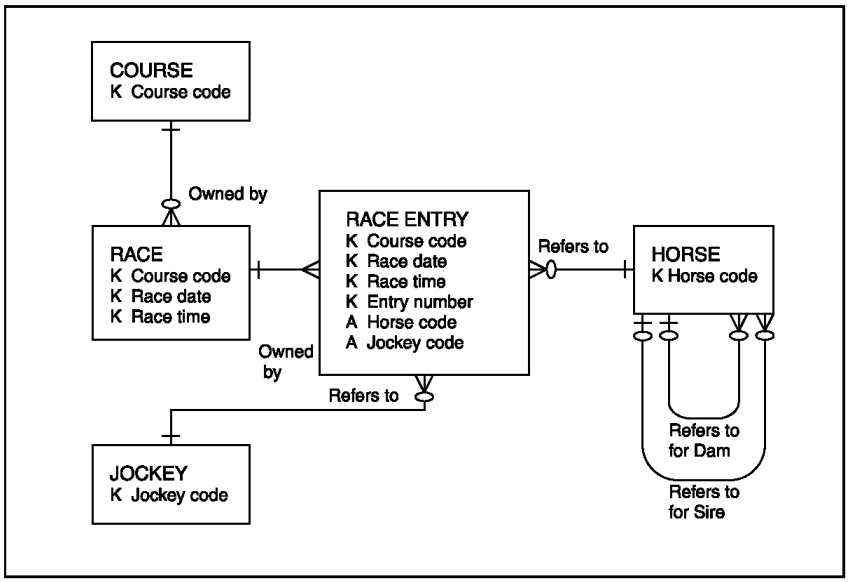
At this point, there is another source of redundant data, namely, the HORSE, DAM, and SIRE entities. A horse could appear in two of these three entities since a horse could both compete in races and also be a Dam or Sire. This situation can be resolved by combining these three entities into just one: HORSE. This process is called *entity generalization*.

The entity relationship diagram now looks like this:



CA 2E Data Model Diagram

The relationships that are identified in the entity relationship diagram can be translated to CA 2E relations. There are eight different CA 2E relations, four of which are commonly required. The purpose of the basic relations will be explained in the course of this tutorial, and more details may be obtained from the CA 2E manuals. The CA 2E Data Model Diagram for this horse race application looks like this:



Having identified the entities (files), attributes (fields), and relationships (relations), we are ready to specify and define them within the data model. This process is illustrated in the following topics.

CA 2E Relations

In this topic you will enter simple relations, define new CA 2E objects for use in the relations, and display the CA 2E entries arising from the relations.

New terms introduced:

- CA 2E relation
- Has relation
- Known by relation
- Owned by relation
- Refers to relation
- CA 2E model object
- File, file attribute
- Field, field attribute (data type)
- File entry

New panels are introduced:

- Edit Database Relations
- Define Objects
- Edit File Entries

Objectives

You will specify relations to define the HORSE, COURSE, JOCKEY, RACE, and RACE ENTRY entities.

Overview of CA 2E Relations

The first step in using CA 2E is to define your basic data model. Do this by typing a number of relation statements to define the entities in your data model.

Relation statements have the form:

Object + Relation + Object

where:

Object is a CA 2E model object. Model objects are used to represent real world entities, such as COURSE, HORSE, or RACE. There are several different types of CA 2E model objects. Initially we need only consider two types: fields (items of data) and files. In general, the entities we identified will become files and the attributes we identified will become fields.

Relation is a CA 2E relation type. The CA 2E relations establish the relationships between fields and files, and files and other files.

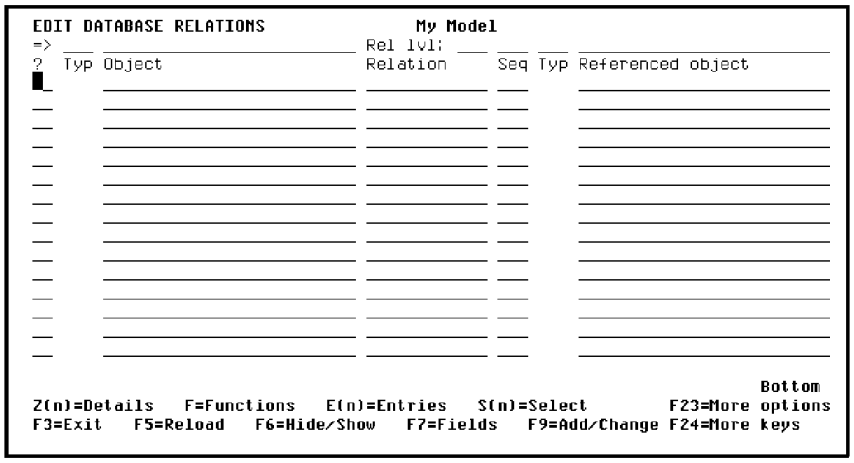
The Edit Database Relations Panel

When you enter your data model, the first panel you see is the Edit Database Relations panel. This panel is the top-level panel in CA 2E.

Note: If you have not already entered your model you can do so now by entering the following command on any i OS Command Entry line.

```
YSTRY2      MYMDL
```

Using the Edit Database Relations panel, you may enter relation statements that declare the entities in your data model. The panel has three main columns in which the relations are entered. These columns are labeled Object, Relation, and Referenced object.



Note: The Subfile selector column on the left lets you enter line selection options. The line of input-capable fields along the top of the panel lets you select which model objects within your CA 2E design model to display. Both will be explained later in this tutorial.

Entering Relation Statements

Working top down, define the most basic entities in your data model: COURSE and RACE.

To do this you will use three different CA 2E relations:

Known by

Declares a field that is part of the primary key that uniquely identifies a file

Has

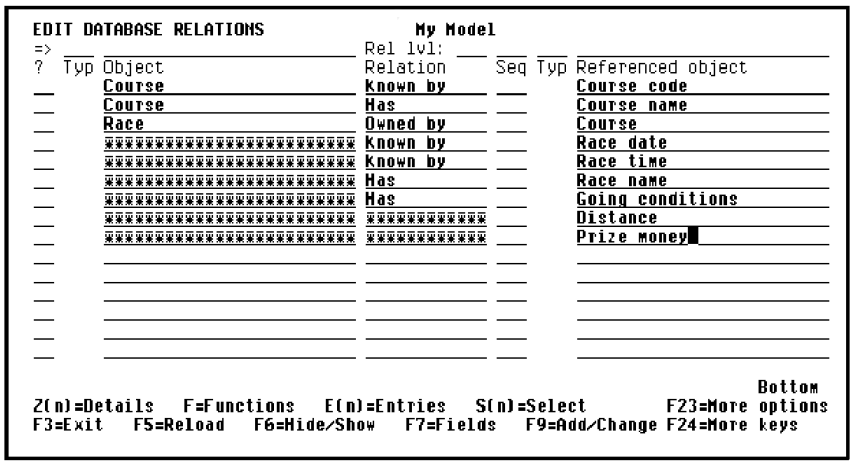
Declares a non-key field (attribute) on a file

Owned by

Specifies a relationship between two files; the primary key of the owning file becomes part of the primary key of the owned file

For example, in the tutorial model, RACE is Owned by COURSE. COURSE is uniquely identified by a Course code. Since RACE is Owned by COURSE, each RACE is uniquely identified by a combination of the COURSE at which it is run, a date, and a time. As a result, the primary key for RACE consists of Course code, Race Date, and Race Time.

Type in relations as shown and use the Tab key or the Field Exit key to advance to the next field or line.



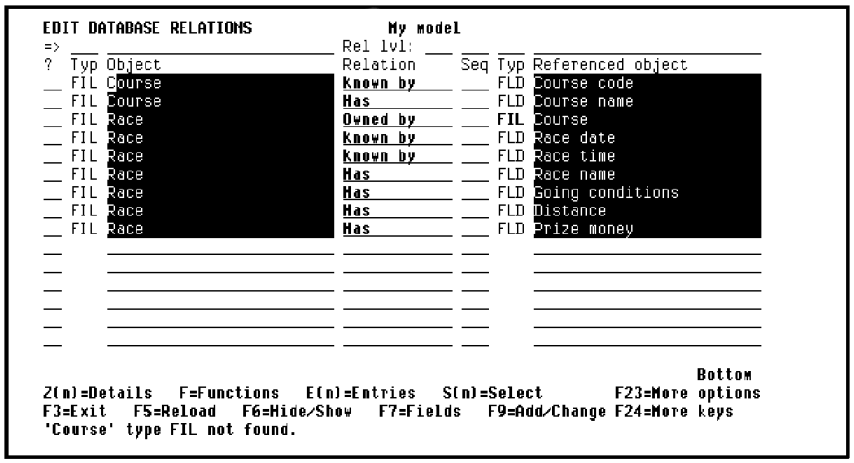
Note: Following are two shortcuts for entering relations.

- You can use the DUP key (indicated by the string of *'s in the above panel) to reduce the amount of typing. When the cursor is in a field and you press the DUP key, values from the previous line will be duplicated on the line below when you press Enter.
- You only need to type the first letter of a relation; for example, you can type **K** or **O** in place of Known by or Owned by, respectively.

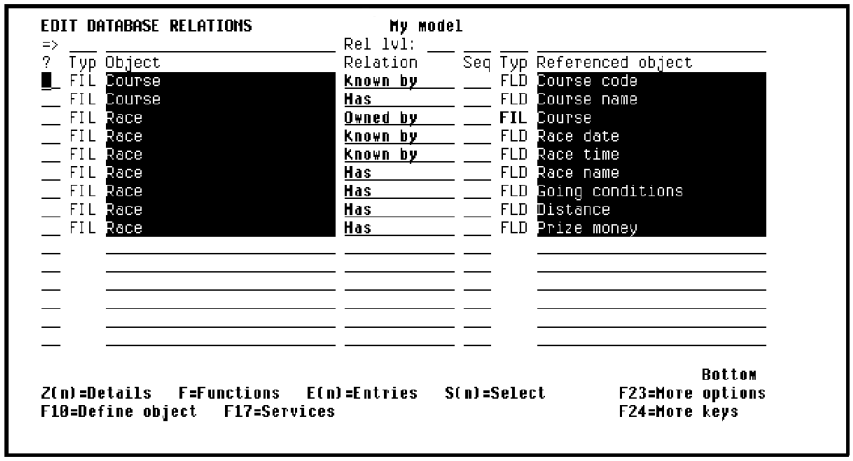
Press Enter after entering all the relations shown.

Validation of Relation Statements

After pressing Enter, the relation statements you have entered are redisplayed, with any use of the DUP key resolved. All new objects, for example, files and fields, are highlighted in reverse image to indicate that you have not yet defined them. An error message for the first undefined item "'Course' type FIL not found" is displayed at the bottom of the panel.



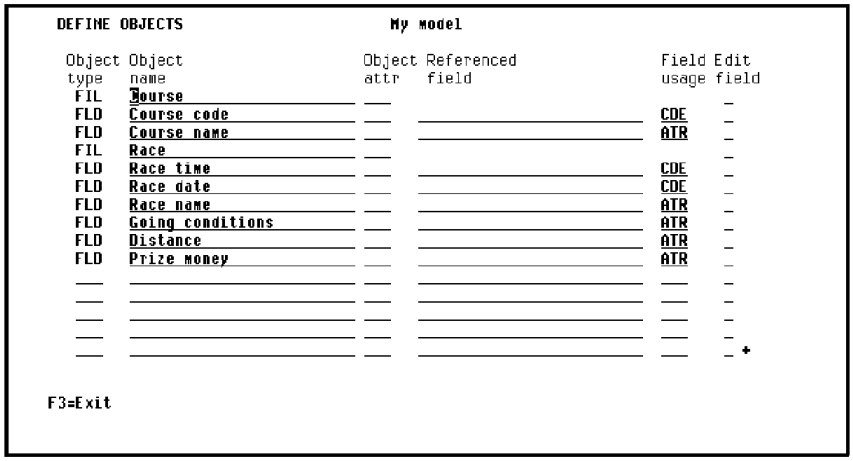
You will use the F10 key to define the objects you just entered. First, press F24 to display more function keys.



Press F10 to display the Define Objects panel where you will enter the details needed to define the objects.

Defining Objects

The Define Objects panel shows each of the new objects you entered on the Edit Database Relations panel.



Based on the relation statements you entered, each new object is automatically assigned one of the following types:

- FIL (for files)
- FLD (for fields)

In addition, fields are assigned a usage as follows:

- CDE (key fields)
- ATR (non-key fields)

Specifying Object Attributes

Using the Define Objects panel, you must specify an object attribute for each new object. You can view a list of the available object attributes by typing **?** in the object attribute field. Object attributes enable CA 2E to take design defaults.

- For files, the object attribute specifies a file type, which will be one of the following in this tutorial:
  - Reference file (REF) for relatively static master files
  - Capture file (CPT) for transaction files

  In the example, COURSE and RACE are Reference (REF) files.
- For fields, the object attribute specifies a data type. (Another term for data type is field type.) For example, Course code is of type CDE, and Course name is of type TXT.

  The data type causes CA 2E to automatically assign default characteristics to a field; for example, length and display format. You can override these defaults later.

  In this tutorial you will use the following data types:

  CDE—Alphanumeric code field

  TXT—Descriptive text

  VAL—Monetary value

  DT#—ISO date

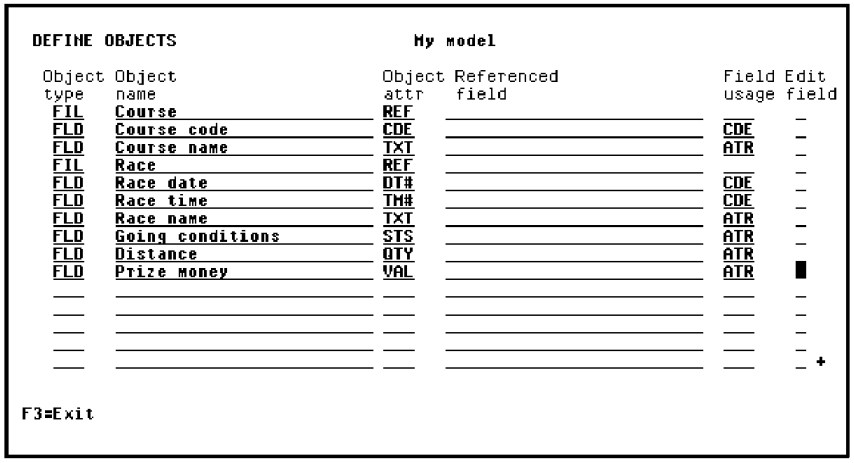
  TM#—ISO time

  STS—Status

  QTY—Quantity

  NBR—Pure numeric field value

  Type the object attributes as shown. Use the Tab key or Field Exit key to advance to another field or line.



Press Enter when you have typed all object attributes.

If all the object attributes you specified were valid,CA 2E automatically exits the Define Objects panel and returns to the Edit Database Relations panel.

File Entries

CA 2E automatically resolves each relation you entered on the Edit Database Relations panel into one or more file entries. A *file entry* indicates the presence of a field on a file. In other words, each CA 2E relation causes one or more fields to be created on a file.

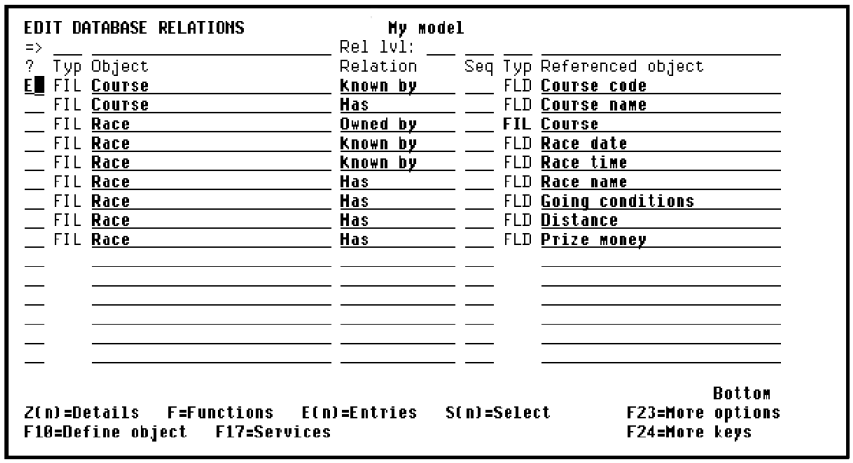
Some examples are:

- The **Known by** relation causes a key field to be created on the named file
- The **Has** relation causes an attribute to be created on the named file
- The **Owned by** relation causes one or more key fields to be created on the owned file.

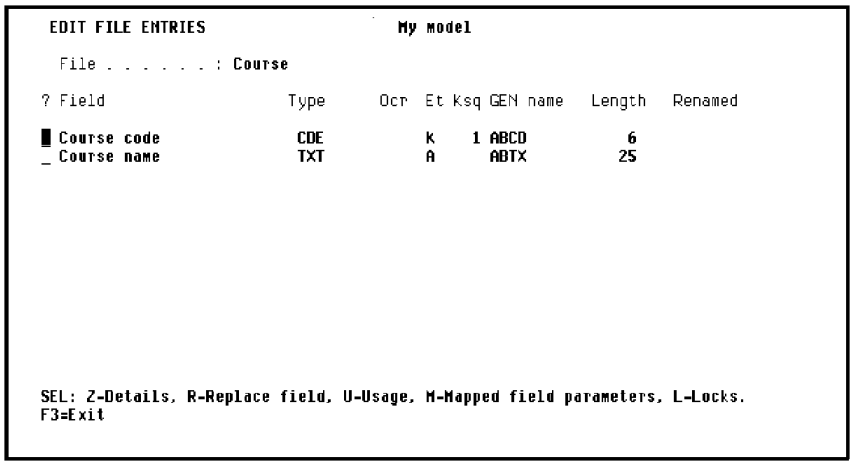
You can view how CA 2E has resolved your relations by displaying the entries for the file.

File Entries for COURSE

You can view the entries on the COURSE file using the Edit File Entries panel. From the Edit Database Relations panel, type **E** next to any of the relations for the COURSE file.



Press Enter to display the Edit File Entries panel. This panel shows the file entries CA 2E automatically created for the COURSE file based on the relations you entered for the file.



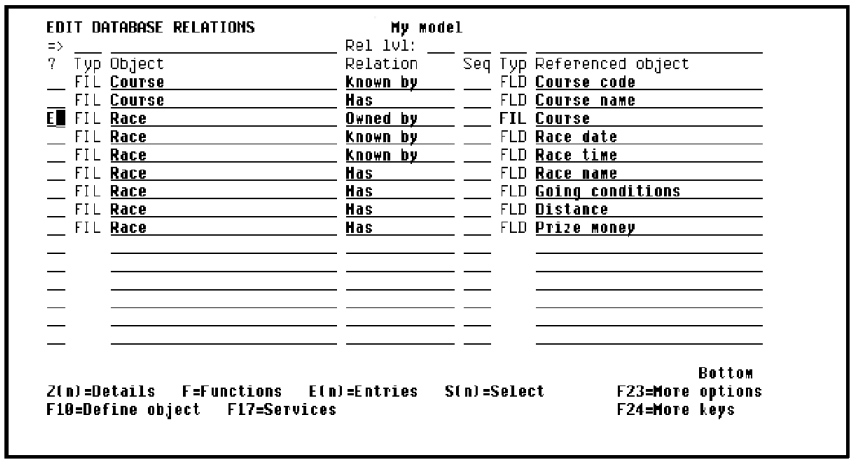
For each file entry, this panel displays the field names, the field type, whether the field is a key field (K) or an attribute (A), the CA 2E generated implementation name, and the field length. Note that CA 2E assigned the implementation name and default length based on the data type you specified; for example, for fields of type CDE,CA 2E assigns a 4-character implementation name ending in CD and a default length of 6.

Note: You cannot update fields on this panel.

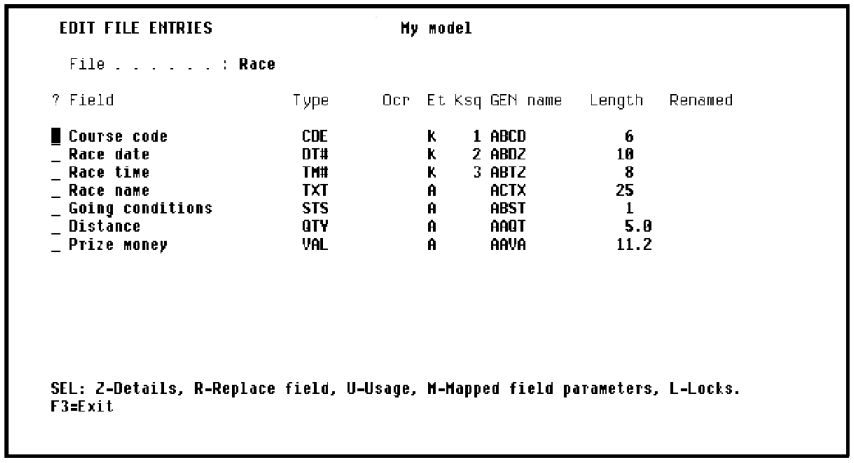
Press F3 to return to the Edit Database Relations panel.

File Entries for RACE

You can examine the entries in the RACE file as you did for the COURSE file. From the Edit Database Relations panel, type **E** against the RACE file.



Press Enter to display file entries for the RACE file.

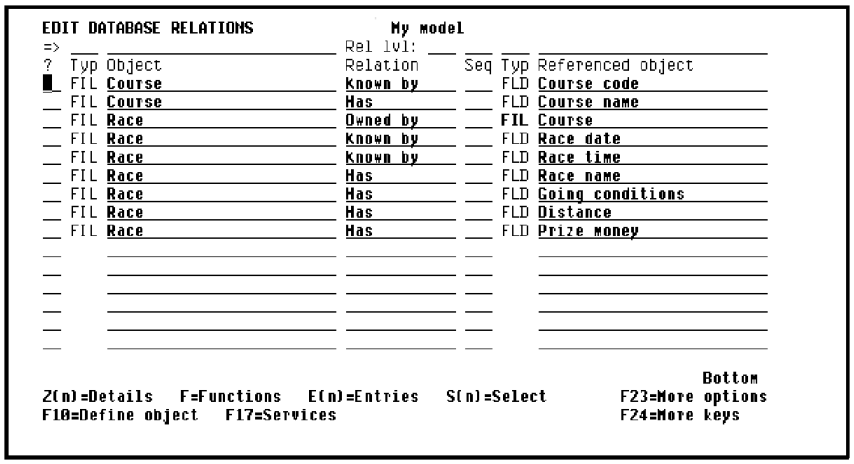


The RACE file has three key field entries resulting from the resolution of the Owned by and Known by relations. The key order is recorded in the Ksq (key sequence) column. The major key is Course code, since RACE is Owned by COURSE.

Press F3 to return to the Edit Database Relations panel.

Adding More Relations

You may now add other files to the data model.



Display a new panel by pressing Roll Up.

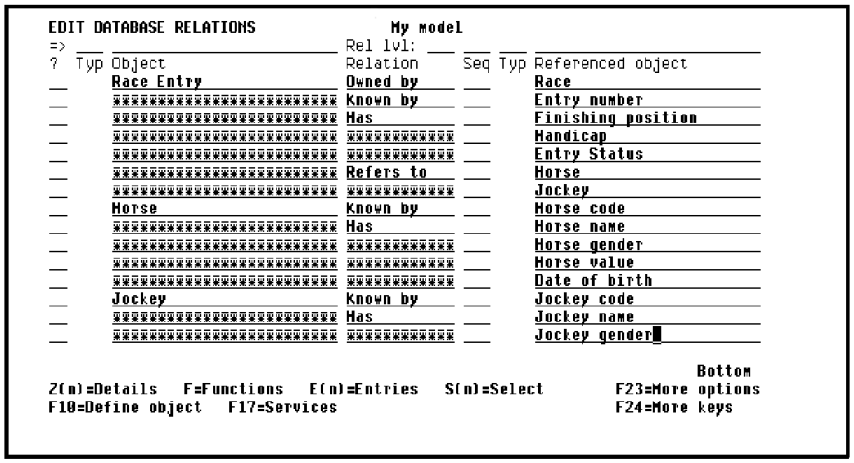
Declaring More Files

You can define the RACE ENTRY, HORSE, and JOCKEY files using the Known by, Has, and Owned by relations as you did for the COURSE and RACE files.

In addition, since RACE ENTRY requires references to the HORSE and JOCKEY files, you will also use the Refers to relation. The Refers to relation specifies a relationship between two files. It causes the key entries of the referenced file (HORSE and JOCKEY) to become foreign keys of the referencing file (RACE ENTRY).

Note that a file may reference another file more than once, as shown later in this tutorial.

Type the **RACE ENTRY**, **HORSE**, and **JOCKEY** relations.



Press F10 to define the new objects.

Defining Objects

The Define Objects panel displays with the object attribute column blank.

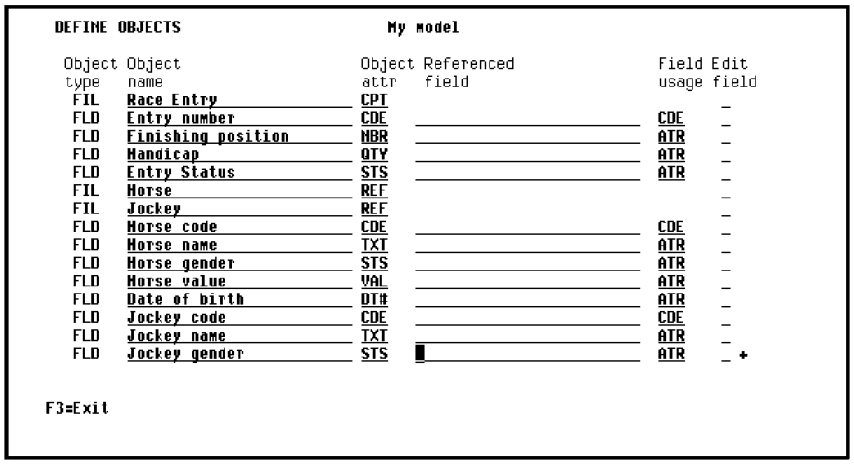
File Attributes

Specify the appropriate object attributes, in order to define the new objects to CA 2E. This requires the following two additional attributes:

CPT—Database capture file

NBR—Pure numeric field value

Type the object attributes.



Press Enter.

If all the object attributes you specified were valid,CA 2E automatically exits the Define Objects panel and returns to the Edit Database Relations panel. You have now finished the top-level definition of your data model.

Deleting Relations

You can delete a relation from the data model by typing **D** against the relation. Once deleted, retyping them can reinstate relations.

Documenting Relations

To obtain a listing of the relations that you have entered, use the Document Model Relations (YDOCMDLREL) command. First press F17 from the Edit Database Relations panel to display the Display Services Menu. Then do either of the following.

- Press F9 to display a command entry line, type the **YDOCMDLREL** command, and press Enter
- Select the Documentation menu option to display a list of documentation commands and select Document model relations

When you finish, press F3 until you return to the Edit Database Relations panel.

Field Details and Conditions

Having entered relations to define your data model, you must now add more detailed information about the fields. This includes defining simple validation rules and specifying any required overrides to the defaults CA 2E assigned for field properties; for example, field length.

This topic describes the field details and how to add field condition information to your data model.

New terms introduced:

- Field condition
- VAL condition type
- LST condition
- Check condition
- Selection line

New panels introduced:

- Edit Field Details
- Edit Field Conditions
- Edit Field Condition Details

Objectives

You will define conditions to specify the allowed values for the Horse gender field.

Overview of Field Details

Field details specify the properties of a given field, such as length, text, validation, and implementation name.

Overview of Field Conditions

Field conditions define allowed values for fields. Conditions record both the values that a field may take and the meaning that the values represent.

Conditions can be used in a number of ways. The most common uses of conditions are:

- Validating the entry of data
- Specifying selection or omission of data from a logical view of the data
- Specifying the processing conditions in a program that operates on the data

Field Details

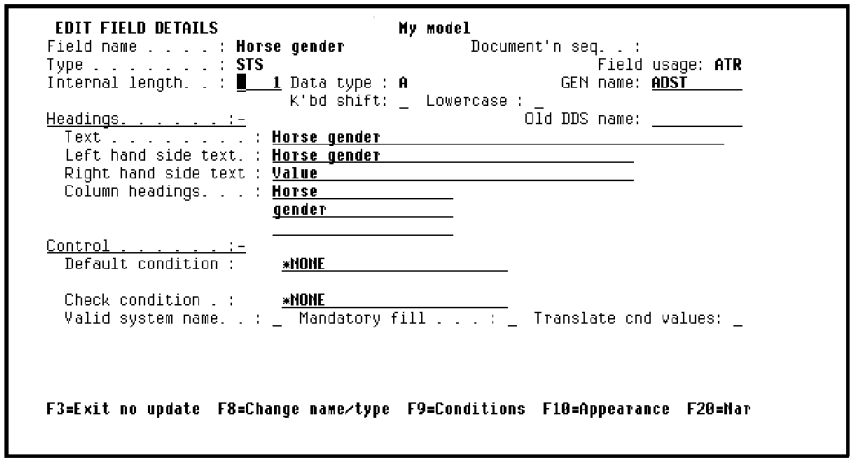
To obtain the field details for a field, you will select the appropriate relation statement line on the Edit Database Relations panel and type **Z2** against this object. The details of the referenced object will be displayed. Note that the Subfile selector options, Z1 and Z, both give details of the first object in the relation.

Field Detail Display for Horse Gender

In this case, the Horse gender field of the HORSE file will be used to illustrate conditions. Type **Z2** against the HORSE Has Horse gender relation.

```
EDIT DATABASE RELATIONS                  My model
=> ___ ____________________________ Rel lvl: ___ ___ ___ ____________________________
?  Typ Object                       Relation     Seq Typ Referenced object
__ FIL Course                       known by     ___ FLD Course code
__ FIL Course                       Has          ___ FLD Course name
__ FIL Horse                        known by     ___ FLD Horse code
__ FIL Horse                        Has          ___ FLD Horse name
Z2 FIL Horse                        Has          ___ FLD Horse gender
__ FIL Horse                        Has          ___ FLD Horse value
__ FIL Horse                        Has          ___ FLD Date of birth
__ FIL Jockey                       known by     ___ FLD Jockey code
__ FIL Jockey                       Has          ___ FLD Jockey name
__ FIL Jockey                       Has          ___ FLD Jockey gender
__ FIL Race                         Owned by     ___ FIL Course
__ FIL Race                         known by     ___ FLD Race date
__ FIL Race                         known by     ___ FLD Race time
__ FIL Race                         Has          ___ FLD Race name
__ FIL Race                         Has          ___ FLD Going conditions

                                                                    More...
Z(n)=Details   F=Functions   E(n)=Entries   S(n)=Select        F23=More options
F10=Define object   F17=Services                              F24=More keys
```

Press Enter to display the Edit Field Details panel for Horse gender.



The Edit Field Details panel shows information for the field. Most of the details, for example, field length, default appropriately according to the field's data type.

It is possible to override the default values if required. We will discuss how to change the field name at a later stage in this tutorial.

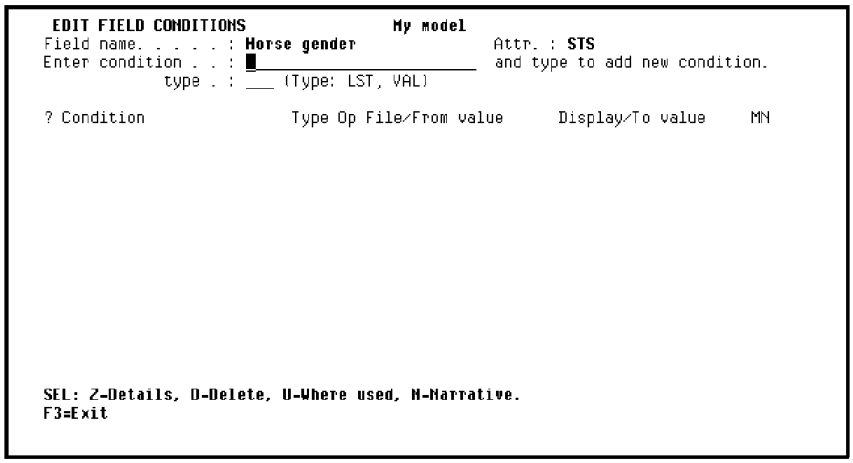
Field Conditions

One use of field conditions is to check that the value entered for a field is valid during data entry. This is done by specifying a Check condition. By default, no field validation takes place. This is indicated by *NONE for the Check condition field on the Edit Field Details panel.

A horse's gender can only be one of two values: male or female. Conditions to check for these values can be attached to the Horse gender field of the HORSE file as follows.

Adding a Field Condition

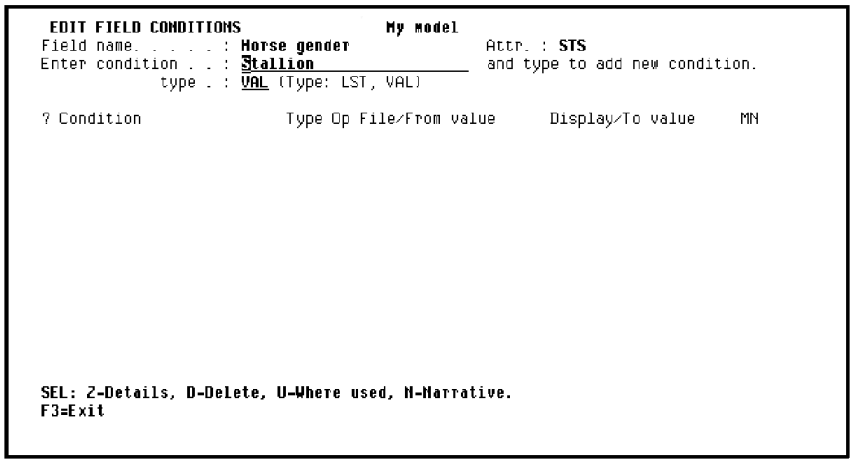
From the Edit Field Details panel for the Horse gender field, press F9 to display the Edit Field Conditions panel. This panel shows the available conditions for the field. Initially, there are none.



To add a field condition, type the name and type of the condition. In this tutorial, one value the Horse gender field can take is Stallion.

You defined Horse gender as a status field (field type STS). Status fields may only take discrete values, so the only condition types allowed for this type of field are single values (VAL) or a list of values (LST). Stallion is a single value so it is of type VAL.

Type the condition name **Stallion** with a type of **VAL**.



Press Enter to display the Edit Field Condition Details panel.

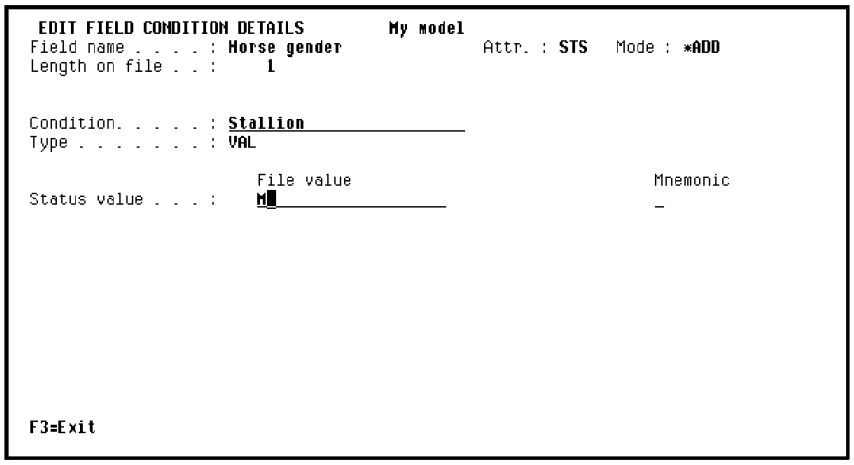
Field Condition Details

The Edit Field Condition Details lets you specify the actual value that is to be stored on the database file to represent a particular condition. For example, you can distinguish between the definition of the condition (Stallion) and the value used to represent the condition on file (M).

Adding Field Condition Detail

The condition value (or Status value) for Stallion is to be represented by an M in this tutorial. Note that condition values must be unique for each field. For example, M could not represent two different gender conditions.

Type the condition value **M**.



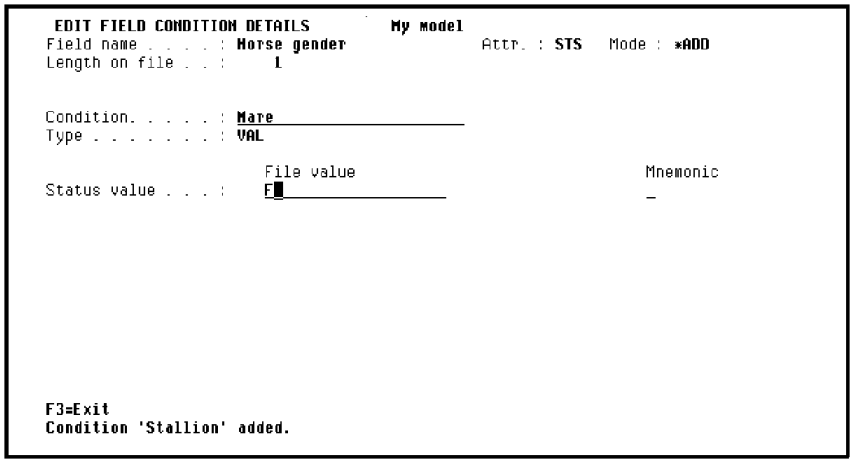
Press Enter.

After pressing Enter, the panel is redisplayed, with the message "Condition ‘Stallion’ added" to indicate that the condition has been successfully added.

Adding Another Condition

In this step you will add another condition value for mares. You can overwrite the displayed values in order to enter this additional field condition.

In this case, you can overwrite Stallion with Mare, and M with F. Type the details of the next condition.

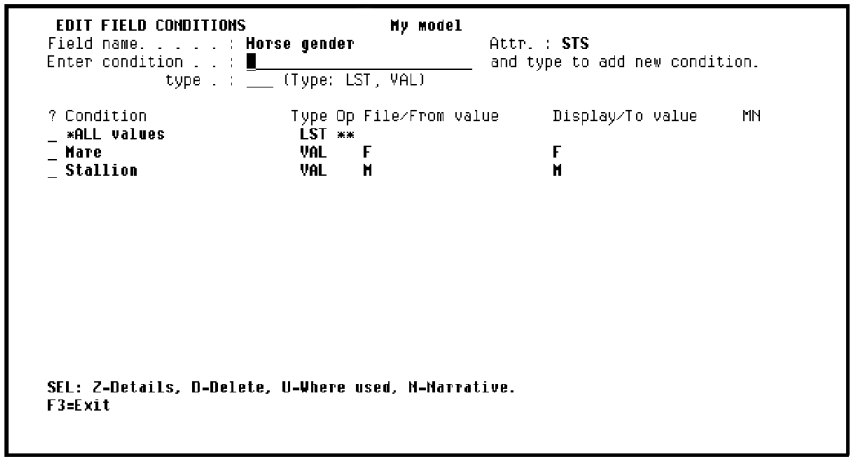


Press Enter to redisplay the panel with the message "Condition ‘Mare’ added."

Press F3 to exit and return to the Edit Field Conditions panel.

Viewing Field Conditions

On the Edit Field Conditions panel for the Horse gender field, you should now see a list of the conditions that you have just added. Note that a special condition has been added automatically. This is the condition called *ALL values, shown at the top of the list of conditions. The *ALL values condition can be used when you wish to specify that a field may take any one of the individual values listed.



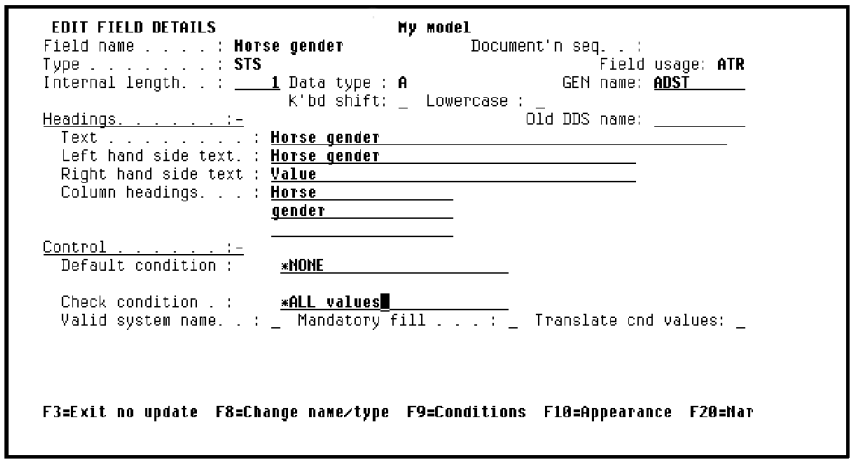
Press F3 to return to the Edit Field Details panel.

Specifying a Check Condition

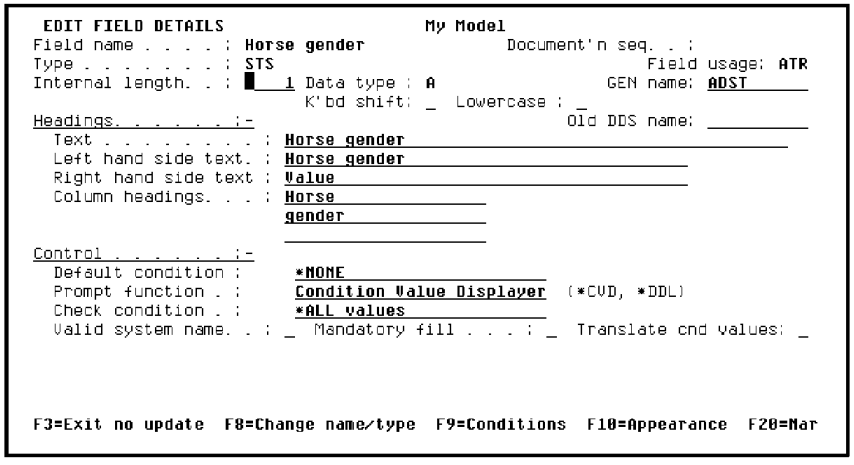
Notice the Check condition field on the Edit Field Details panel. This field indicates the values that are allowed for the Horse gender field. Initially the Check condition is *NONE, indicating that any value is allowed for the field. If you specify the list condition *ALL values, it indicates that the field must satisfy one of the particular value conditions that you have defined; in other words, it provides validation during data entry. If you specify one of the conditions that you have just defined, it imposes specific validation on the field.

You can type **?** in the Check condition field, to view a selection list of existing conditions. In this tutorial, all horses must be either mares or stallions. The check condition *ALL Values is appropriate.

Type ***ALL values** for the Check condition field.



Press Enter.



Note that a Prompt function field now displays. As a result, when the end user of the compiled application positions the cursor on the Horse gender field and either enters **?** or presses F4, a selection list of the conditions you defined will be displayed.

Press F3 to return to the Edit Database Relations panel.

Exercises

Before continuing with the rest of the tutorial, complete the following exercises using the same process you used to define conditions for Horse gender.

1. Add conditions to define the allowed values for the Jockey gender field.
2. Add conditions to define the allowed values for the Going conditions field on the RACE file; for example, Firm, Good, and Soft.
3. Add conditions to define the allowed values for the Entry Status field on the RACE ENTRY file; for example, Not yet run, Finished, Scratched, and Disqualified.

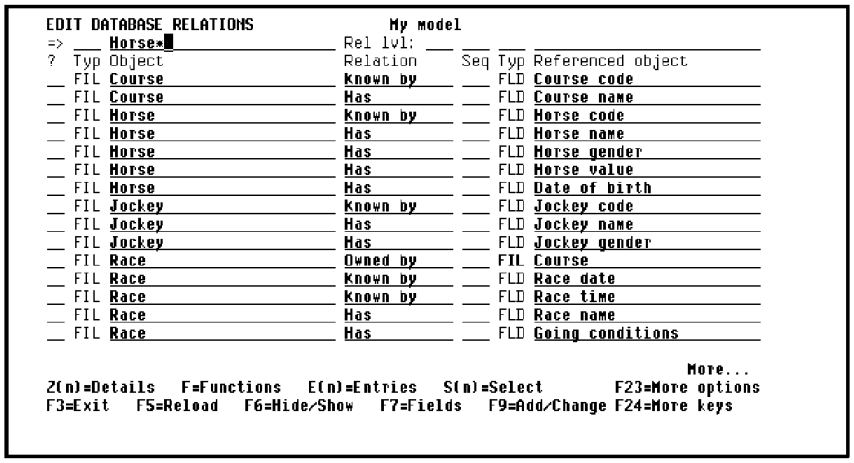
Displaying Selected Relations

Now that you have specified the relations, you can use the features of the Edit Database Relations panel to display selected objects.

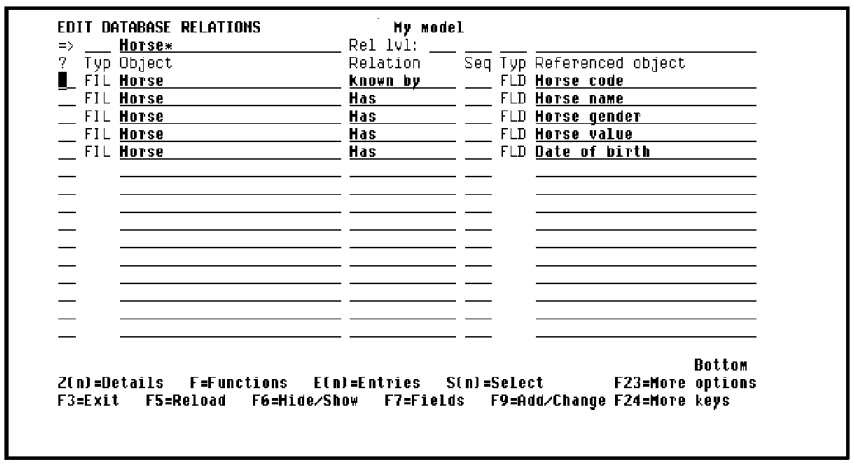
On the top of the Edit Database Relations panel are fields that let you position the display at the part of the model you want to view, and let you control which CA 2E relations are displayed. The line that contains these fields is referred to as the *selection line*. Type **?** in any section of the line to display the values available for selection. If you enter a letter (or string of letters) followed by an asterisk (*) in the Object or Referenced object columns, only those CA 2E objects whose names begin with that letter (or letters) will be selected and displayed.

In the next topic, you will work with the HORSE file. You can simplify the display by selecting and displaying only those relations that define the HORSE file. Do this by typing the file name followed by an asterisk on the selection line. You will use this technique often in this tutorial.

Type **Horse*** in the Object column of the selection line.



Press Enter.CA 2E displays only those relations that define the HORSE file.



Note: If the selection line is blank, you are returned to the beginning of the CA 2E design model when the panel is reloaded, for example, if you press F5 or Enter.

Extending Relations

This topic further refines the CA 2E data modeling language and introduces relation extension.

New terms introduced:

- Extended relations
- Sharing
- For text
- Involution (self-referencing relation)
- Relation sequence

Objectives

You will add relations to the HORSE file to specify each horse's parents (pedigree).

Overview of Relation Extension

As discussed previously, one file can be associated with another by means of the Refers to relation. For example, our data model contains the RACE ENTRY Refers to HORSE, and RACE ENTRY Refers to JOCKEY relations. Note that since the referenced file in these two relations are different, there is no difficulty in distinguishing between them.

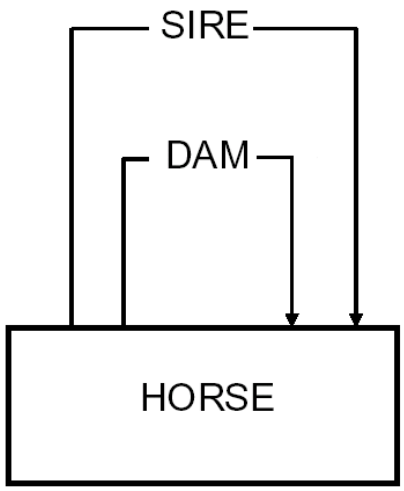
A file can also reference another file more than once. In order to distinguish among multiple references to the same file,CA 2E provides an extended form of the Refers to relation that lets you specify differences among the relations using For text.

In addition, a file can refer to itself. This is known as *involution* or a *self-referencing relation*.

Involution and the Horse Pedigrees

One of the purposes of this application is to record details of each horse's parents (Dam and Sire). To do so you will need to use both the extended form of the Refers to relation and involution.

Since a horse's parents are also horses, to record the horse's parents requires two HORSE Refers to HORSE relations, one for each parent. This is shown in the following diagram.

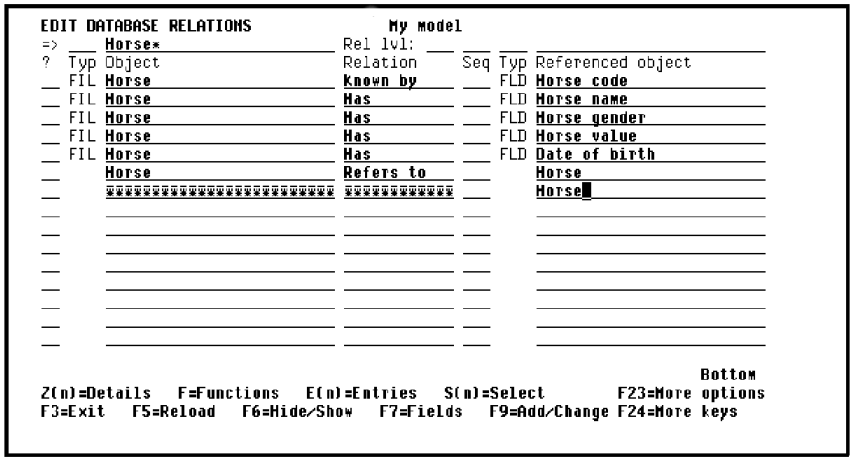


Because the HORSE file refers to itself more than once, you will need to use the extended form of the Refers to relation in order to distinguish between the two relations.

Adding More Relations

If you have not already done so, type **Horse*** on the selection line on the Edit Database Relations panel to display only the relations that define the HORSE file. Press Enter.

You are now ready to add the relations that define the horse's parents. To add references to a horse's parents type the Horse Refers to Horse relation twice; once for the Dam and once for the Sire. Type the HORSE Refers to HORSE relations.



Press Enter.

Sequence of Relations

Before extending the Refers to relations for the HORSE file, you need to ensure that these two relations remain at the end of the relations for the HORSE file. The importance of this will become apparent later in the discussion of virtual fields.

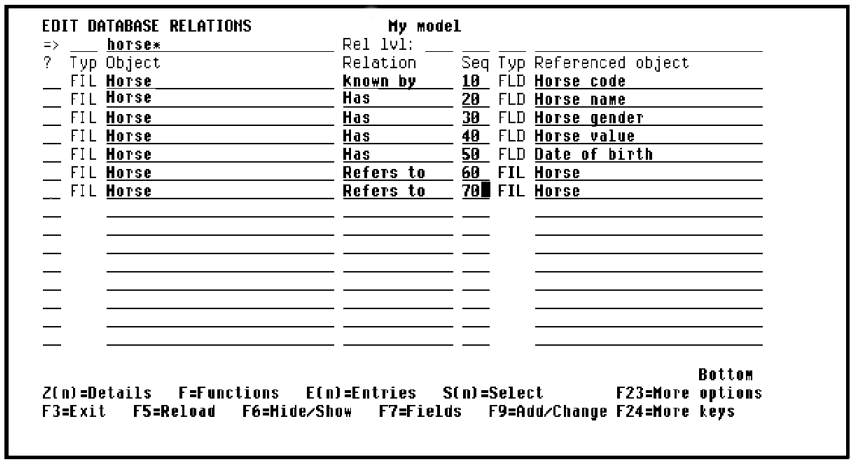
Whenever the data model is reloaded, for example, when you press F5,CA 2E redisplays the relations in the following order:

- First, alphabetically by file name
- Then within each file in relation order, for example, Owned by before Known by before Refers to before Has
- Then by referenced objects in the order in which you entered them

You can override this order by entering sequence numbers to the relations in the Seq column on the Edit Database Relations panel.

You will type sequence numbers 10 to 70 so that the Refers to relations remain at the end of the list when the panel is reloaded. If you do not do this the two Refers to relations will be placed after the Known by relation instead.

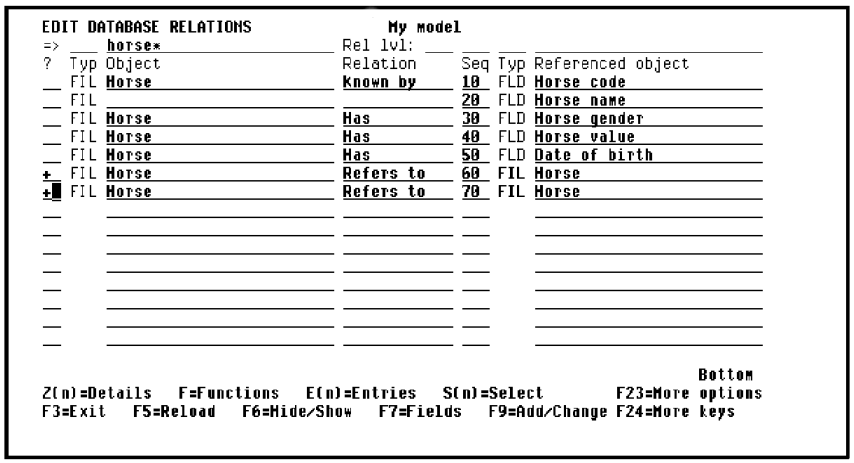
Type the sequence numbers.



Press Enter.

Extending Relations

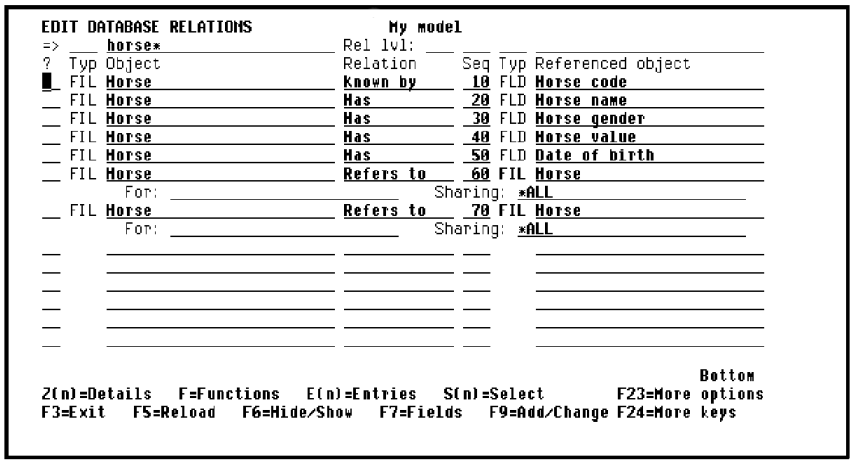
The two HORSE Refers to HORSE relations can be distinguished by extending the Refers to relations and defining For text. To do this, type **+** next to each of those relations.



Press F5.

Extended Relations for HORSE File

The two HORSE Refers to HORSE relations are now shown in an extended form that allows you to specify two extra terms on the relation. Any Refers to or Owned by relation can be extended in this way.



Extending a relation introduces the two following fields:

- For field

  The For field lets you specify the reason for each relation. In this tutorial, you will use this field to distinguish the two HORSE Refers to HORSE relations; in other words, HORSE Refers to HORSE for Dam, and HORSE Refers to HORSE for Sire.

- Sharing field

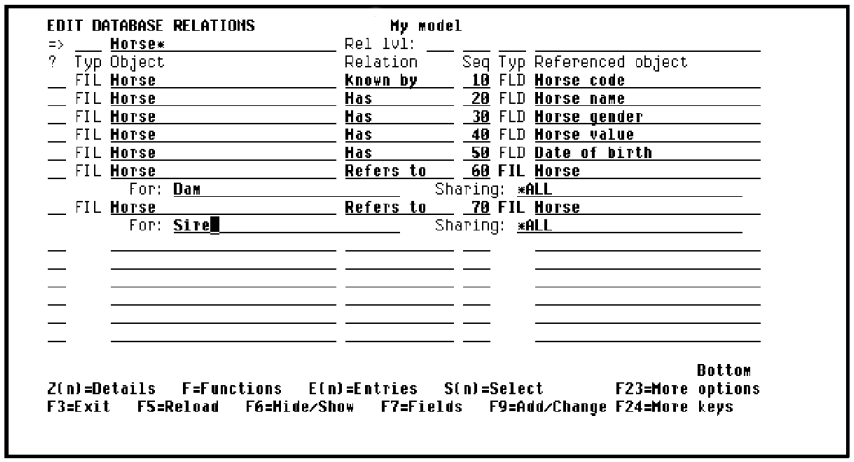
  The entry in the Sharing field indicates whether the common keys of the two relations are to be shared. *ALL is the default. This field applies only when the referenced file has more than one key field entry. In this tutorial, since the HORSE file has only one key field entry (Horse code), you can take the default.

Example Showing Use of the Sharing Field

Following is a brief description of a situation where you would need to enter a value for the Sharing field. Suppose your model includes a BREED file that is Known by Breed code and that the HORSE file is Owned by BREED. The Owned by relation causes Breed code to become a key entry on HORSE giving HORSE two keys, Breed code and Horse code. To indicate that a horse can be bred only with another horse of the same breed (same Breed code), enter ***ALL** for the Sharing field. To indicate that a horse can be bred with a horse of a different breed, enter ***NONE** for the Sharing field.

Adding Details of Extended Relations

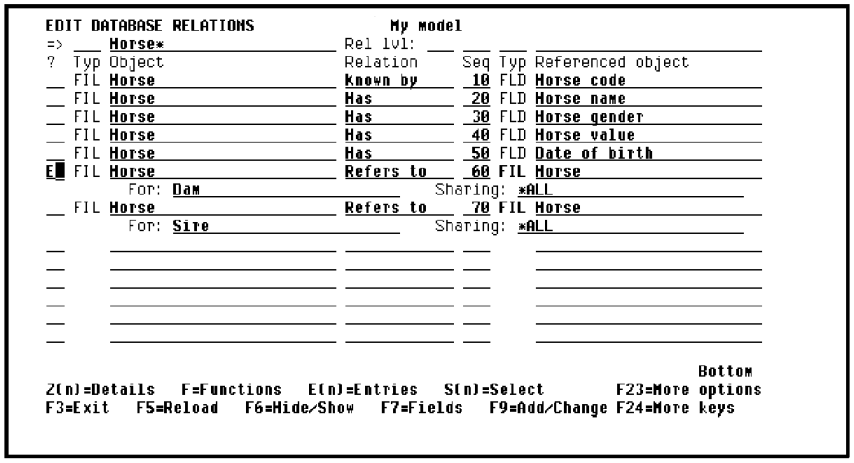
You can now specify text to describe the extended relations: Dam for the first relation, and Sire for the second. Type the extended relation details, **Dam** and **Sire**.



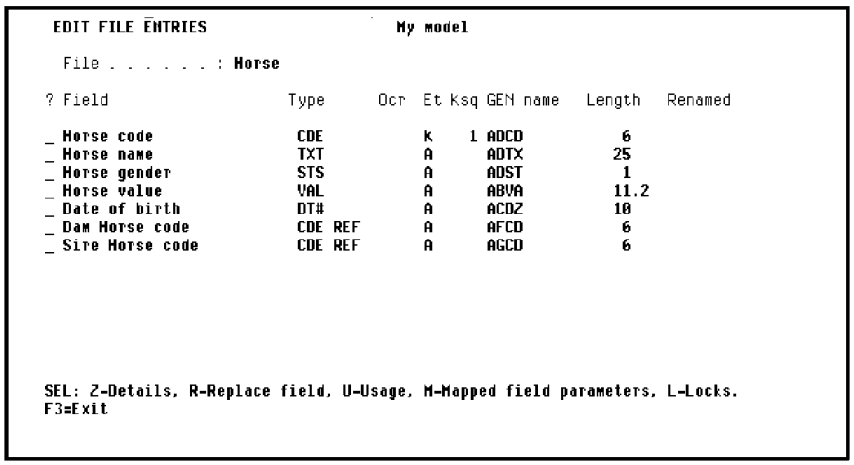
Press Enter.

Field Entries for HORSE

The entries on the HORSE file are automatically updated by CA 2E to include the two new Refers to relations. Examine the entries on the HORSE file by typing **E** in the Subfile selector.



Press Enter to display the Edit File Entries panel.



Note that the resolution of the two Refers to relations added two foreign key entries to the HORSE file, Dam Horse code and Sire Horse code.CA 2E based the definition of these fields on that of the Horse code field, and as a result, assigned the attribute CDE. REF following the attribute indicates that the field entry was resolved from a Refers to relation.

Assigning Unique Names to Field Entries

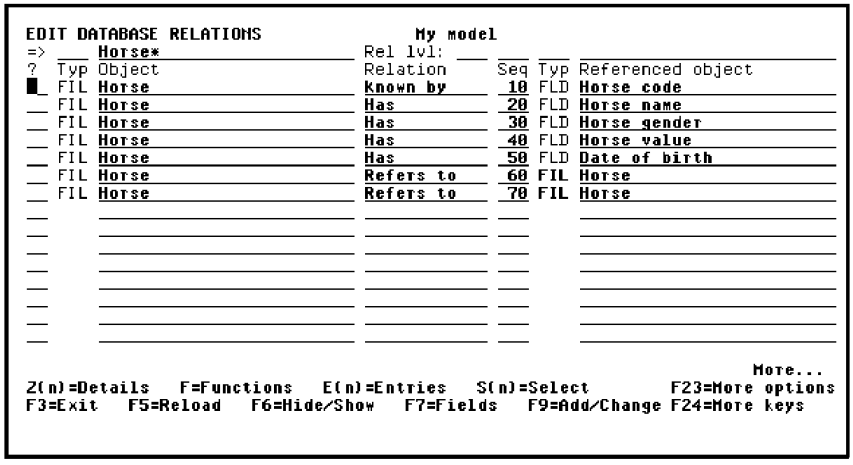
Normally the name of a foreign key is the same as the name of the original key field; in this case Horse code. However,CA 2E files cannot contain duplicate field names. To assign unique names to the two new foreign key entries,CA 2E prefixed Horse code with the For text you entered. If you had not entered For text,CA 2E would have added a 5-digit number to Horse code to create a unique name within the HORSE file; for example, Horse code 25642.

Press F3 to return to the Edit Database Relations panel.

Showing and Hiding Relation Extension Lines

You can use the F6 command key to toggle between showing and hiding relation extension lines. Note that the default extension lines are always hidden.

Press F6 to hide the relation extension lines for the two HORSE Refers to HORSE relations.



Press F6 again to show the relation extension lines.

Virtual Fields

This topic discusses the purpose of virtual fields and how to specify them within CA 2E.

New terms introduced:

- Virtual field
- Virtual field entry

New panel introduced:

- Edit Virtual Field Entries

Overview of Virtual Fields

A good database design eliminates redundancy; in other words, each item of data should be stored in only one place, rather than having multiple copies on different files. The i OS join logical file is a mechanism for building views that assemble data from different files.

In CA 2E you will use virtual fields to design such views. A *virtual field* is a field that is logically, but not physically, present on a file.CA 2E virtual fields are implemented through i OS join logical files.

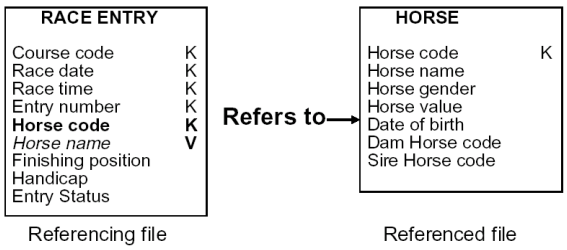
Note that there are some limitations as to how virtual fields may be used in functions.

Specifying Virtual Fields

When a file is referenced by another file by means of a Refers to relation, entries are automatically created on the referencing file for the key fields of the referenced file. These are known as foreign keys and are shown in bold in the figure below.

To include a non-key field from the referenced file in the referencing file, you need to specify it as a virtual field. This makes the field available for use in the functions that operate upon the referencing file.

For example, to include the name of the horse when displaying the RACE ENTRY file, specify Horse name as a virtual field on the RACE ENTRY file. As a result, Horse name is physically present on the HORSE file, and is logically, but not physically, present on the RACE ENTRY file. This is shown in italics in the following figure.



Note that this same capability is available when two files are associated with each other by means of an Owned by relation. In other words, non-key fields from the owning file may be included in the owned file as virtual fields.

Virtual Fields for Details of a Horse's Dam and Sire

The HORSE file now includes Dam Horse code and Sire Horse code as a result of the Refers to relations. In this step you will specify Horse name and Date of birth as virtual fields on the HORSE file so you can refer to the names and dates of birth for a horse's Dam and Sire.

This information is already present in the database. To retrieve this data onto the HORSE file, you need to specify the additional fields to be virtual fields on the appropriate Refers to relations.

Sequence of Relations and Virtual Fields

With a self-referencing file such as HORSE, the sequence numbers you entered previously against the database relations are significant when specifying virtual fields.

CA 2E uses a single-pass algorithm to resolve entries on a file. When CA 2E looks for virtual fields, it resolves all the relations for the referenced file in the order that they appear on the Edit Database Relations panel. The following figure shows both the default relation order for the HORSE file and the order determined by the sequence numbers you entered earlier in this tutorial.

Default Relation Order
(no sequence numbers)

Known by	Horse code
Refers to	Horse For Dam
Refers to	Horse For Sire
Has	Horse name
Has	Horse gender
Has	Horse value
Has	Date of birth

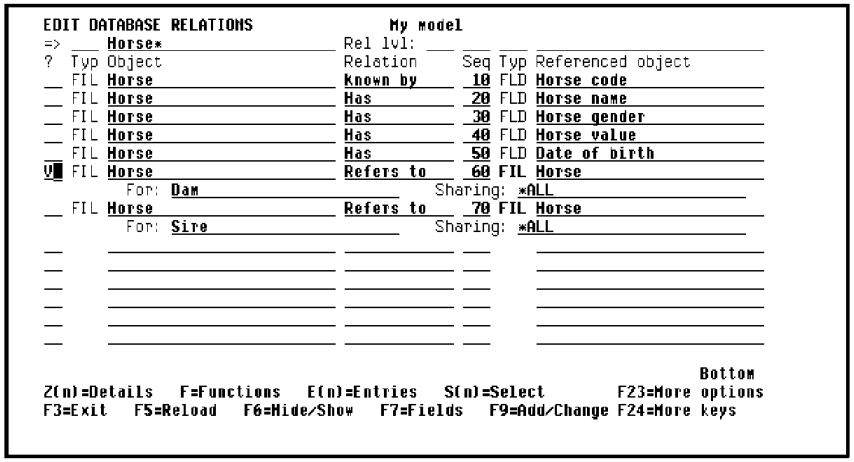
User-defined Relation Order
(with sequence numbers)

Known by	Horse code
Has	Horse name
Has	Horse gender
Has	Horse value
Has	Date of birth
Refers to	Horse For Dam
Refers to	Horse For Sire

Because the HORSE file refers to itself, any relations that occur after the Refers to HORSE relations will not have been resolved at the time CA 2E attempts to resolve the virtual fields for the Refers to relations. In other words, if you had not inserted sequence numbers for the HORSE file relations, the Has relations would not have been resolved at the time CA 2E needed to resolve the Refers to relations. As a result,CA 2E could not have created the virtual entries for the Horse name and Date of birth fields.

Adding Virtual Fields

You can now add the Dam's name and date of birth to the list of fields for the HORSE file by specifying virtual fields. Type **V** next to the HORSE Refers to HORSE For Dam relation.



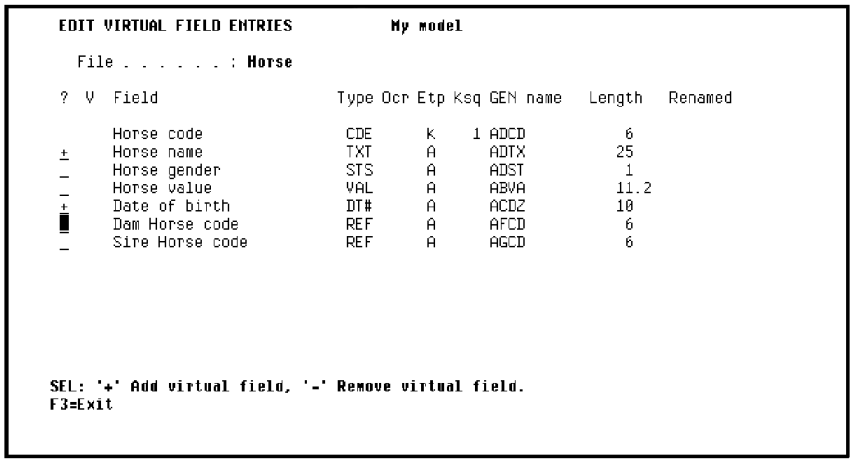
Press Enter to display the Edit Virtual Field Entries panel.

Virtual Field Entries

The Edit Virtual Field Entries panel shows all the fields on the referenced file. Since the HORSE file refers to itself, these are the same as the fields on the referencing file. From this list, you can select any field as a virtual field.

Selecting Virtual Fields for Dam

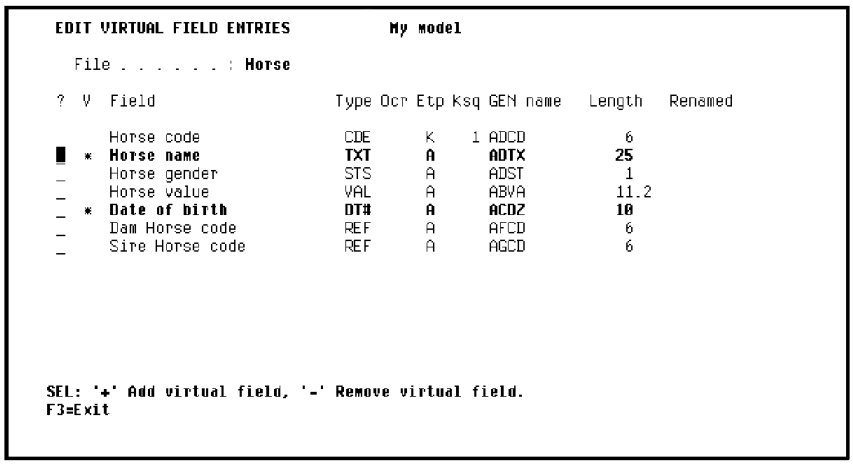
The Horse name and Date of birth fields will become virtual fields for the Refers to For Dam relation on the HORSE file. Type **+** against Horse name and Date of birth.



Press Enter.

Confirming Virtual Fields for Dam

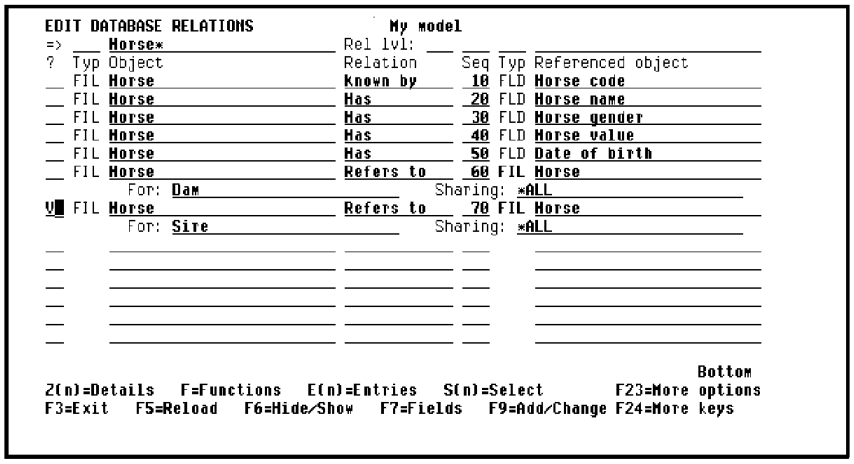
The virtual fields are indicated by an asterisk (*) in the V column of the panel confirming that CA 2E has created the virtual fields.



Press F3 to return to the Edit Database Relations panel.

Adding More Virtual Fields

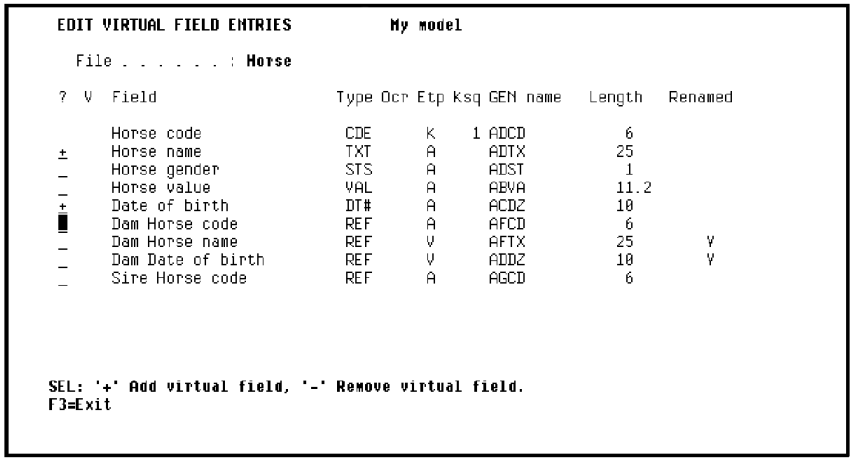
The procedure for adding virtual fields for the horse's Sire is the same as for the horse's Dam. Type **V** against the HORSE Refers to HORSE For Sire relation.



Press Enter to display the Edit Virtual Field Entries panel. This panel now includes the entries Dam Horse name and Dam Date of birth you specified earlier. Note that to assign unique names for these entries,CA 2E prefixed the original field names with the For text you specified in the Refers to relation.

Selecting Virtual Fields for Sire

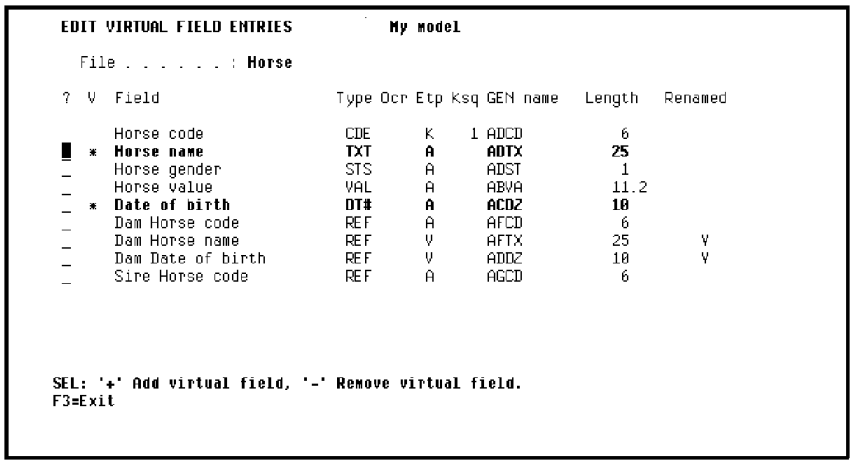
Select Horse name and Date of birth as virtual fields for the Sire. Type **+** to select Horse name and Date of birth.



Press Enter.

Confirming Virtual Fields for Sire

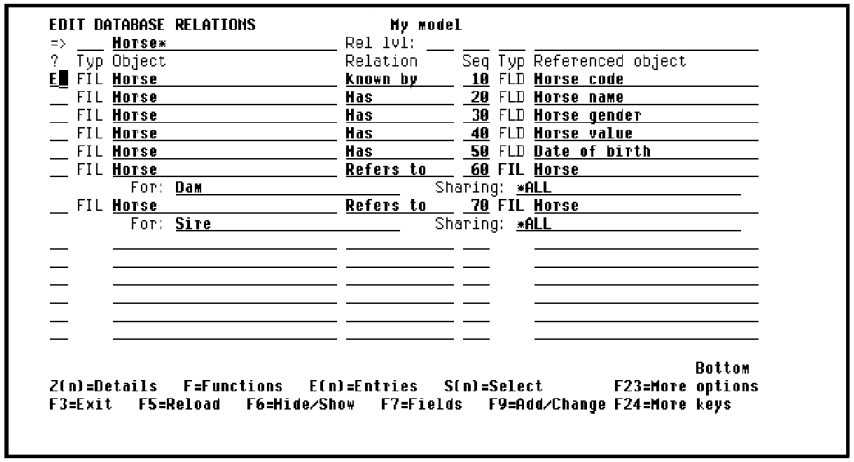
CA 2E has confirmed your selection of virtual fields, as indicated by the asterisks.



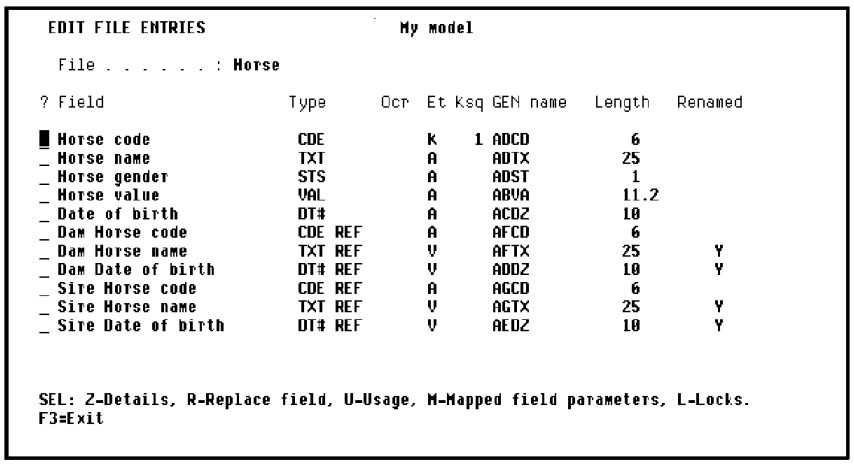
Press F3 to return to the Edit Database Relations panel.

File Entries

Check the file entries for the HORSE file. To display the file entries, type **E** next to one of the HORSE file relations.



Press Enter to display the Edit File Entries panel.

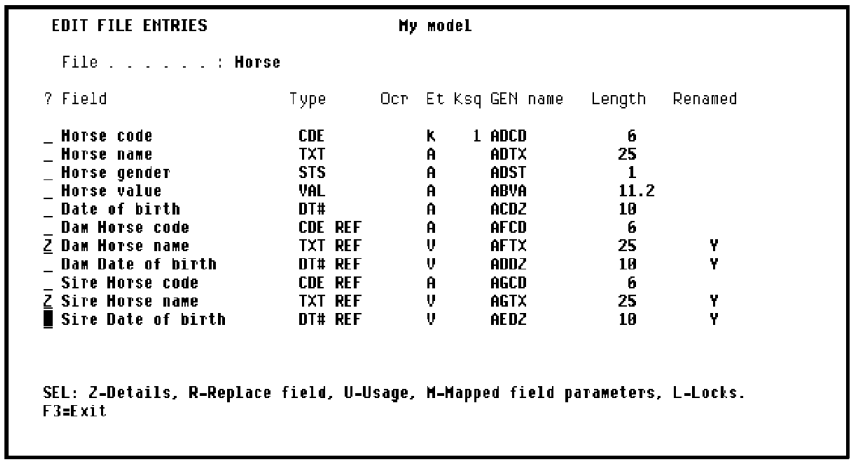


Note that the virtual fields you just specified are included as indicated by a V in the Entry type (Et) column.CA 2E has defined them by reference to the original fields and has named them by prefixing the For text to the original name, for example, Dam Horse name.

Renaming Fields

You can change the names that CA 2E has given the virtual fields to make them more explicit or compact. For example, you could change Dam Horse name to Dam name, and Sire Horse name to Sire name.

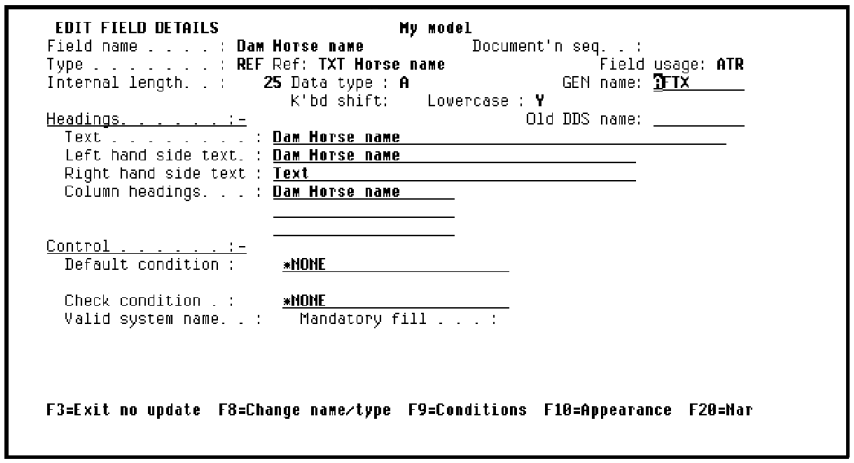
To select the fields to be renamed you will zoom against the relevant fields on the Edit File Entries panel. Type **Z** to select the fields to be renamed.



Press Enter to display the Edit Field Details panel for the first field you selected.

Field Details

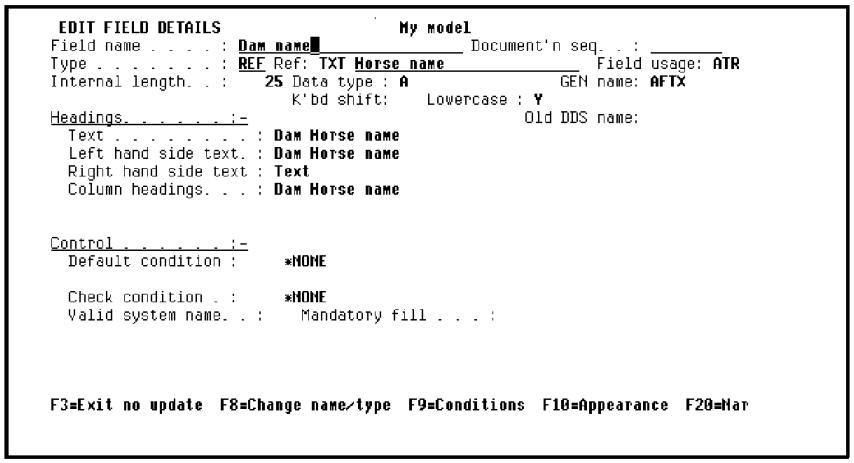
The Edit Field Details panel lets you view details for the field, Dam Horse name. Some of the attributes of the field, such as the name, are initially protected. The F8 key unprotects the field name so it can be changed.



Press F8 to change the field name.

Entering a New Field Name

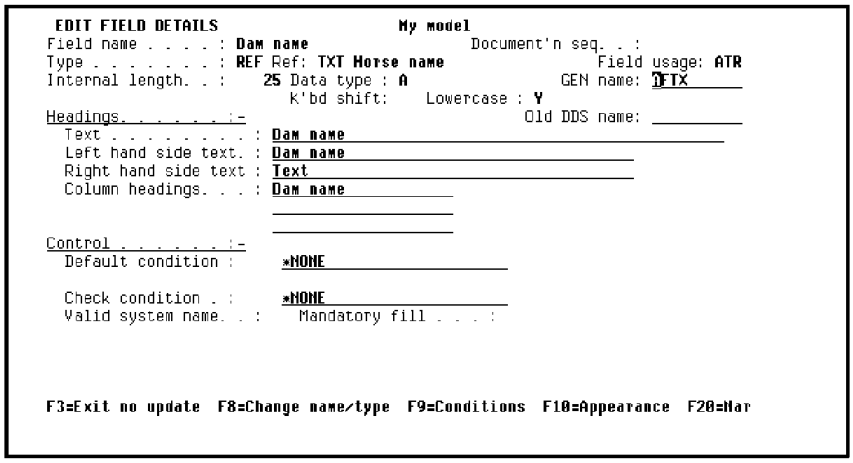
The cursor will now be on the Field name entry on the Edit Field Details panel. Rename the field by typing the new name over the existing name. Type **Dam name** over Dam Horse name.



Press Enter.

Field Details After Renaming

The Edit Field Details panel will be displayed with the new field name inserted throughout. You can rename any field in this way, and the new name will be reflected throughout your design model.



Renaming Other Fields

Press Enter to redisplay the Edit Field Details panel for the Sire horse name field. Recall that this was the second field you selected for renaming on the Edit File Entries panel.

Exercise

Change Sire Horse name to **Sire name** following the same steps as shown previously.

Displaying File Entries Again

When you have renamed the selected fields, press Enter to return to the Edit File Entries panel. The panel has been updated to show the new field names.

```
 EDIT FILE ENTRIES                    My model

  File . . . . . . : Horse

? Field                    Type      Ocr  Et Ksq GEN name   Length   Renamed

█ Horse code                CDE           k    1 ADCD          6
_ Horse name                TXT           A      ADTX         25
_ Horse gender              STS           A      ADST          1
_ Horse value               VAL           A      ABVA         11.2
_ Date of birth             DT#           A      ACDZ         10
_ Dam Horse code            CDE REF       A      AFCD          6
_ Dam name                  TXT REF       V      AFTX         25        Y
_ Dam Date of birth         DT# REF       V      ADDZ         10        Y
_ Sire Horse code           CDE REF       A      AGCD          6
_ Sire name                 TXT REF       V      AGTX         25        Y
_ Sire Date of birth        DT# REF       V      AEDZ         10        Y


SEL: Z-Details, R-Replace field, U-Usage, M-Mapped field parameters, L-Locks.
F3=Exit
```

Press F3 to return to the Edit Database Relations panel.

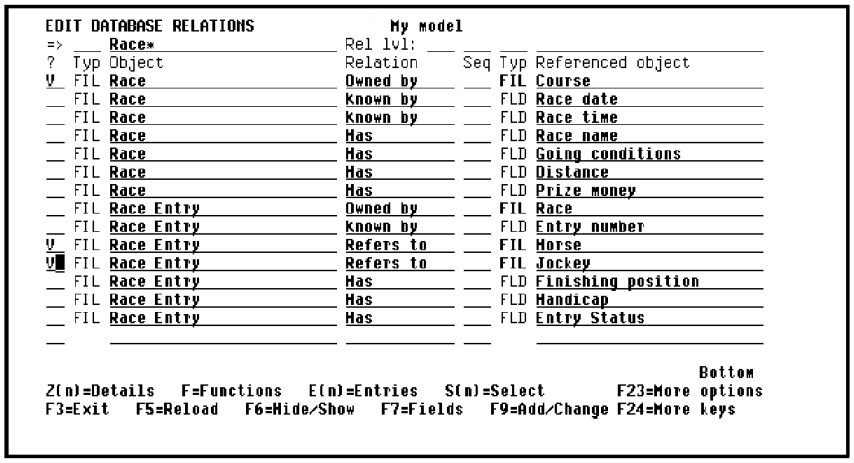
Adding Virtual Fields for the Race and Race Entry Files

Having added virtual fields to the HORSE file, you may specify virtual fields on other file-to-file relations; for example, on the RACE and RACE ENTRY files. Type **Race*** on the selection line.

```
EDIT DATABASE RELATIONS                My model
=> ___ Race*                    Rel lvl: ___ ___ ___
?  Typ Object                   Relation     Seq Typ Referenced object
█  FIL Horse                    Known by      10 FLD Horse code
__ FIL Horse                    Has           20 FLD Horse name
__ FIL Horse                    Has           30 FLD Horse gender
__ FIL Horse                    Has           40 FLD Horse value
__ FIL Horse                    Has           50 FLD Date of birth
__ FIL Horse                    Refers to     60 FIL Horse
        For: Dam                        Sharing: *ALL
__ FIL Horse                    Refers to     70 FIL Horse
        For: Sire                       Sharing: *ALL
__     ____________________ ____________ ___     ____________________
__     ____________________ ____________ ___     ____________________
__     ____________________ ____________ ___     ____________________
__     ____________________ ____________ ___     ____________________
__     ____________________ ____________ ___     ____________________
__     ____________________ ____________ ___     ____________________

                                                                  Bottom
Z(n)=Details   F=Functions   E(n)=Entries   S(n)=Select       F23=More options
F3=Exit   F5=Reload   F6=Hide/Show   F7=Fields   F9=Add/Change F24=More keys
```

Press Enter to display relations for the RACE and RACE ENTRY files.

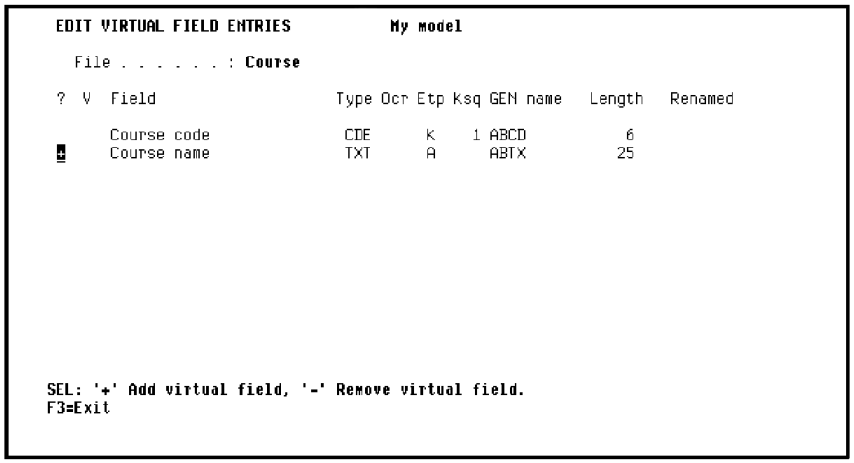
In this step you will define Course name as a virtual field on the RACE file and Horse name and Jockey name as virtual fields on the RACE ENTRY file. Type **V** against the three relations.



Press Enter.

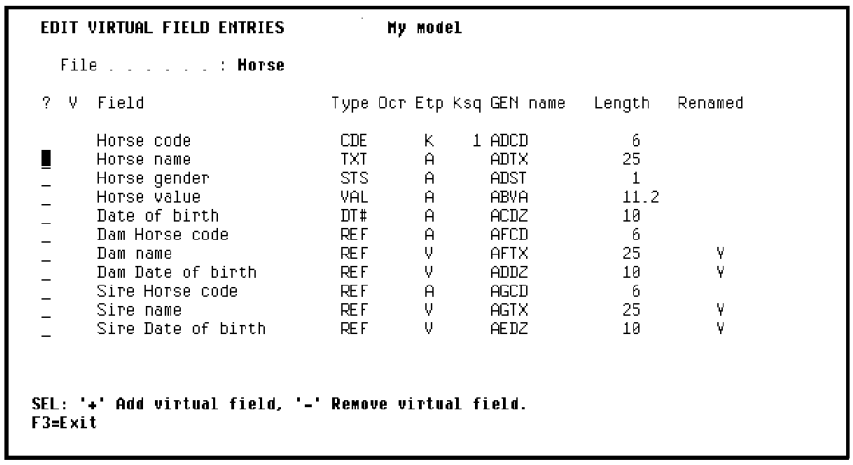
Adding More Virtual Entries

The allowed virtual field entries from the COURSE file are displayed. Type **+** against Course name to add it as a virtual entry to the RACE file.



Press Enter. The Edit Virtual Field Entries panel redisplays to confirm the creation of the virtual field as indicated by the '*' in the V column.

Press F3.CA 2E displays the Edit Virtual Field Entries panel for the HORSE file. Recall that you typed **V** against the RACE ENTRY Refers to HORSE relation on the Edit Database Relations panel.



```
EDIT VIRTUAL FIELD ENTRIES          My model

  File . . . . . : Horse

? V  Field                  Type Dcr Etp Ksq GEN name   Length   Renamed

     Horse code             CDE      K    1 ADCD          6
_    Horse name             TXT      A      ADTX         25
_    Horse gender           STS      A      ADST          1
_    Horse value            VAL      A      ABVA         11.2
_    Date of birth          DT#      A      ACDZ         10
_    Dam Horse code         REF      A      AFCD          6
_    Dam name               REF      V      AFTX         25        V
_    Dam Date of birth      REF      V      ADDZ         10        V
_    Sire Horse code        REF      A      AGCD          6
_    Sire name              REF      V      AGTX         25        V
_    Sire Date of birth     REF      V      AEDZ         10        V


SEL: '+' Add virtual field, '-' Remove virtual field.
F3=Exit
```

Exercise

Add Horse name and Jockey name as virtual fields to the RACE ENTRY file. Use the same method you just used to add Course name as a virtual field to the RACE file. When you finish, press F3 to return to the Edit Database Relations panel.

CA 2E Access Paths

This topic introduces CA 2E access paths and presents some of their features such as access path selection and access path relations. You are also shown how to generate source code for the database files to implement CA 2E access paths.

New terms introduced:

- Access path
- Retrieval (RTV) access path
- Based-on file
- Access path selection
- Select/omit set
- Access path relation
- Referenced access path

New panels introduced:

- Edit File Details
- Edit Access Path Details
- Edit Access Path Select/Omit
- Edit Access Path Conditions
- Edit Access Path Relations
- Display File Access Paths
- Virtualize Access Path

Objectives

In this topic you will:

- Accept the default access paths for the COURSE file.
- Add virtual fields to the default Retrieval access path for the HORSE file.
- Add two access paths to the HORSE file to select either female horses (Mares) or male horses (Stallions).
- Modify the default Retrieval access path for the HORSE file to ensure that Dams are female and Sires are male.

Overview of Access Paths

The next stage in defining your design model is to define the required access paths. Access paths specify views of the files in the data model. The views define how data is to be presented to the functions. Functions specify the processes that operate on the data, see the chapter "Designing Functions" for more information.

Access paths control three different aspects of how data is presented to a function. An access path determines:

- Which fields will be available to the function
- Which records will be selected or omitted from the file (select/omit)
- The order in which records are presented (key sequence) to the function

Default Access Paths

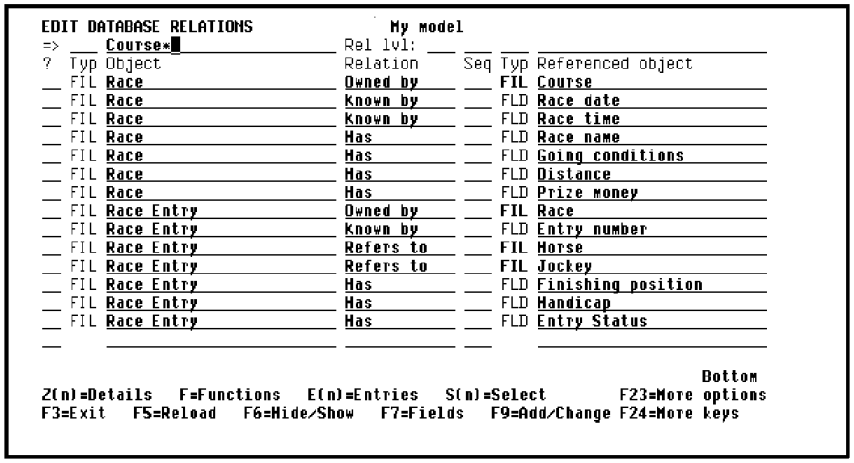
CA 2E uses the relations in the data model to create a default set of access paths for each file. In many cases these will be sufficient. You may override the defaults or define additional access paths.

Three default access paths are automatically created for every file:

- A physical (PHY) access path (un-keyed). It contains the address of all data stored physically within a file and stored in the order in which the data was written to the file.
- An update (UPD) access path (logical view). It is used to update the file and contains all fields defined for the file. It cannot be altered.
- A retrieval (RTV) access path (logical view). It specifies a view of the data that CA 2E generated programs use to retrieve records from a file. Each file has at least one.

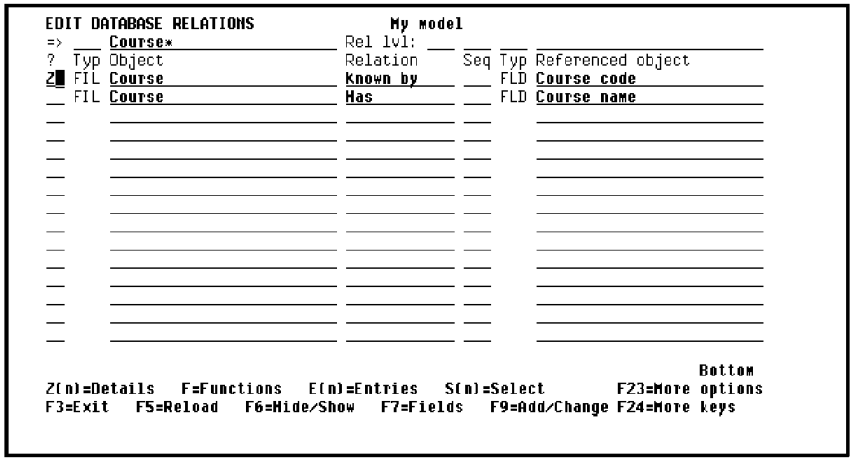
View Default Access Paths for COURSE

Type **Course*** on the selection line to display only relations for the COURSE file.

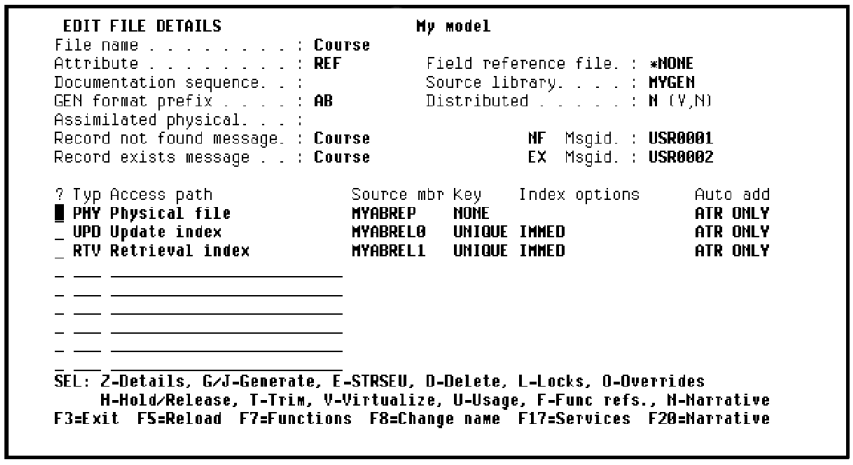


Press Enter.

The access paths for a file are recorded on the Edit File Details panel. To obtain this panel for the COURSE file, you will zoom against one of the relations for the file on the Edit Database Relations panel. Type **Z** against one of the relations for the COURSE file.



Press Enter to display the Edit File Details panel.

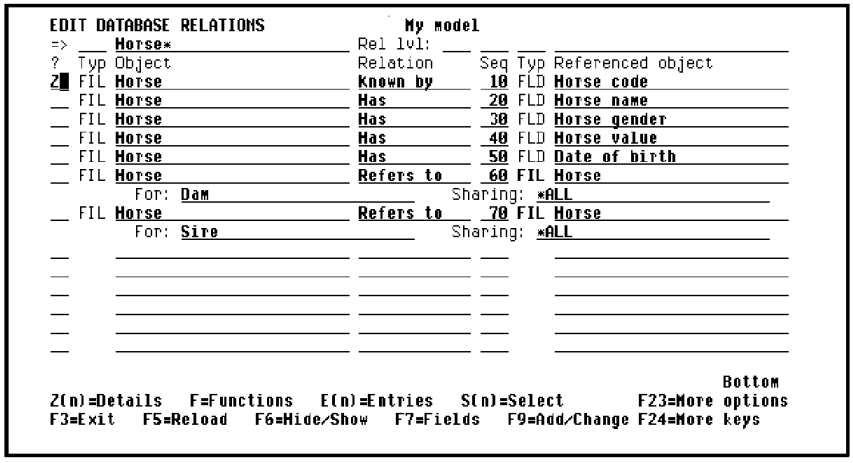


This panel shows the three default access paths that CA 2E automatically created for the COURSE file.

Press F3 to return to the Edit Database Relations panel.

Access Paths for the HORSE File

Display the relations for the HORSE file from the Edit Database Relations panel by typing **Horse*** on the selection line and pressing Enter. To display the access paths for the HORSE file, zoom into one of the relations for the file on the Edit Database Relations panel. Type **Z** against one of the relations.



Press Enter to display the default access paths for the HORSE file.

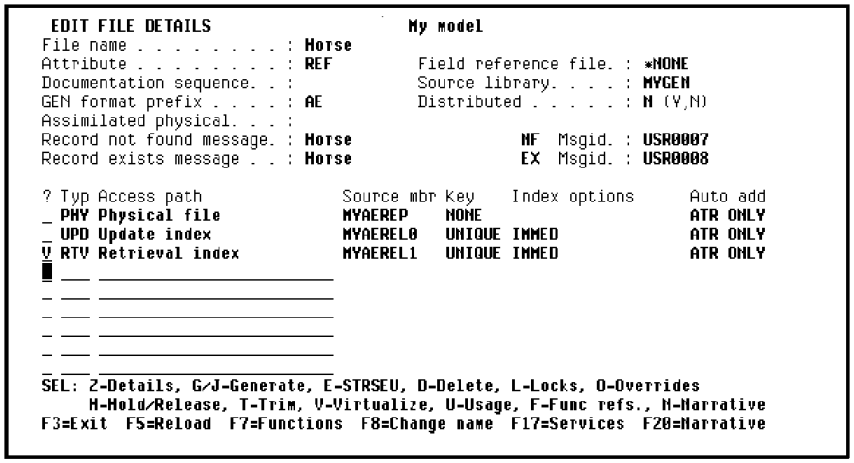
Adding Virtual Fields to the Retrieval Access Path

Each access path initially contains all of the relations for the file on which it is based but none of the virtuals. When you add a new relation to a file, the effect on the access path is controlled by the value in the Auto add column of the Edit File Details panel. The default is ATR ONLY, which automatically adds only attributes to the access path. As a result, if you want virtual fields included on an access path, you need to add them explicitly.

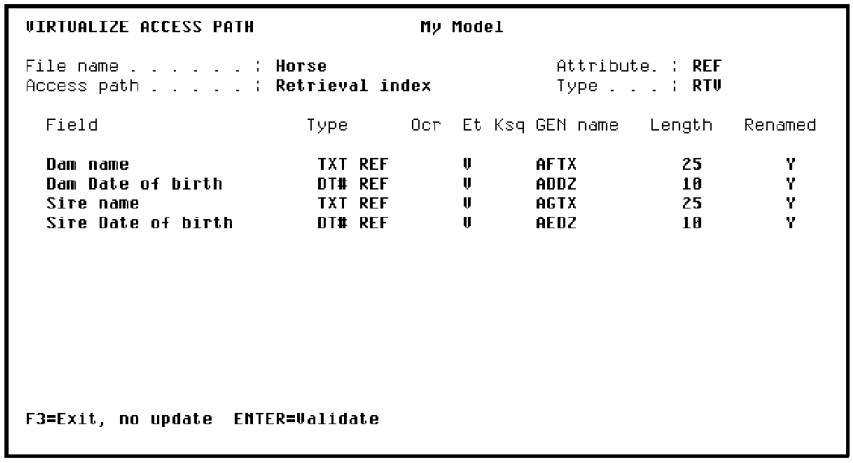
In this step you will add the virtual fields you just defined for the HORSE file to the default Retrieval access path.

Note: Adding virtual fields to an access path increases processing overhead. As a result, in a real-world model you would not add virtual fields to the default retrieval access path. Instead, create a new RTV access path and add the virtual fields to it. Assign a name to the new access path that indicates it contains virtual fields; for example, Retrieval with virtuals. You will learn how to add a new access path later in this topic.

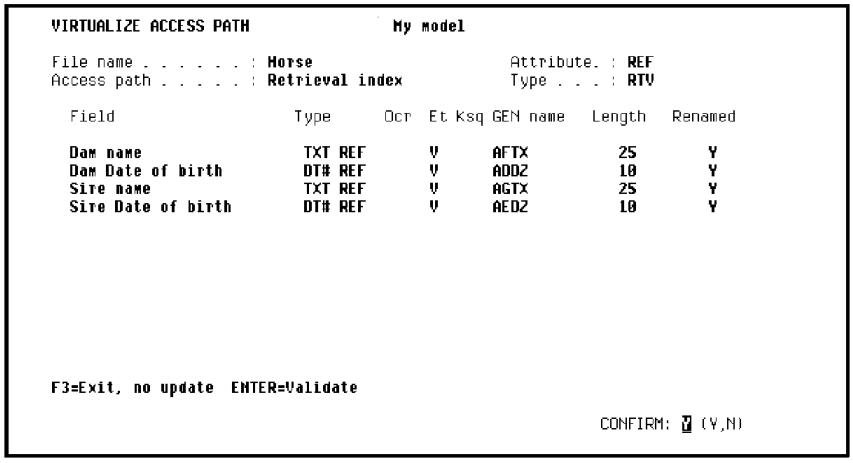
Type **V** in the Subfile selector for the Retrieval index access path.



Press Enter to view virtual entries for the HORSE file.

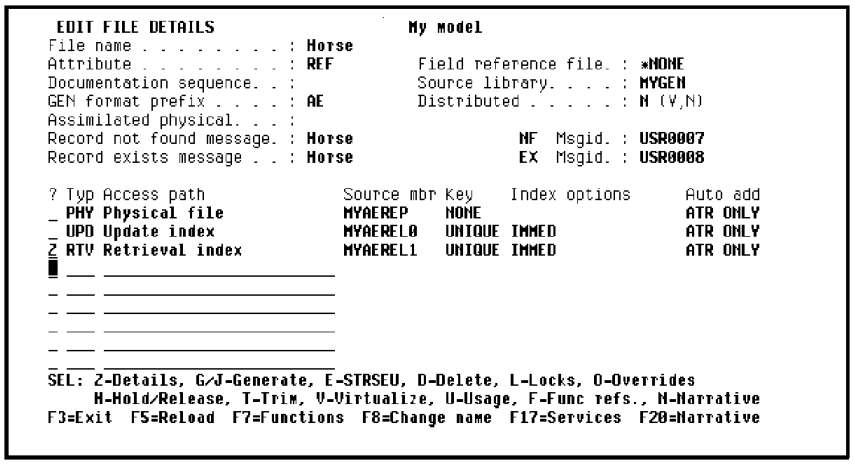


Press Enter to validate the addition of the virtual fields to the Retrieval index access path for the HORSE file.



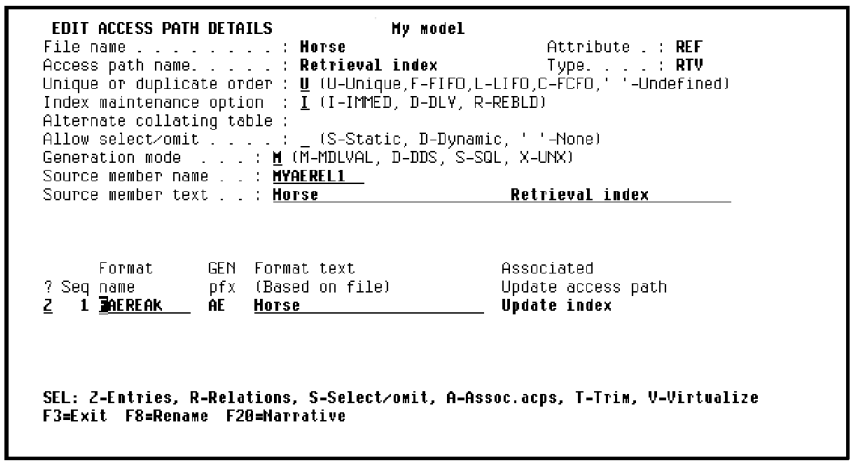
Press Enter again to respond to the Confirm prompt in the lower right-hand corner of your screen and return to the Edit File Details panel.

To view the entries for the default Retrieval access path, type **Z** in the Subfile selector for the Retrieval index.

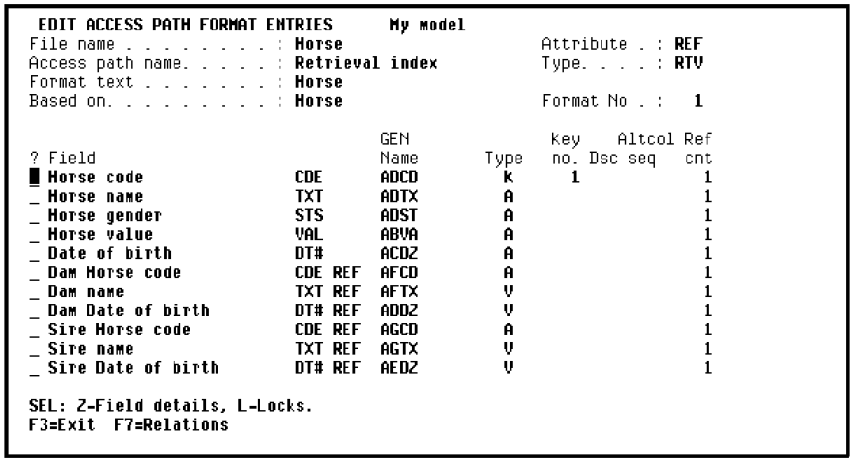


Press Enter to display the Edit Access Path Details panel.

Type **Z** again as shown to view entries for the Retrieval index access path.



Press Enter to display the Edit Access Path Format Entries panel.



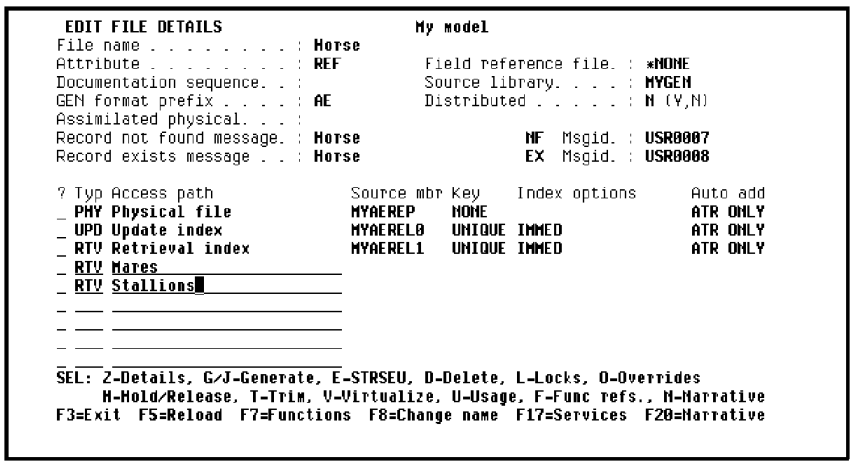
Verify that the virtual fields you specified are listed. Virtual fields are indicated by a **V** in the Type column. Press F13 to return to the Edit File Details panel.

Adding New Access Paths for HORSE

In this step you will add some new access paths as well as update the default Retrieval (RTV) access path. This will demonstrate some of the more sophisticated capabilities of access paths.

For example, you can define an access path to select only certain relations from the file or to include selection criteria based on conditions specified for certain fields. In this tutorial, you will specify new access paths that select only mares or only stallions from the HORSE file. Call these new RTV access paths Mares and Stallions.

Type the names of the new RTV access paths, **Mares** and **Stallions**.



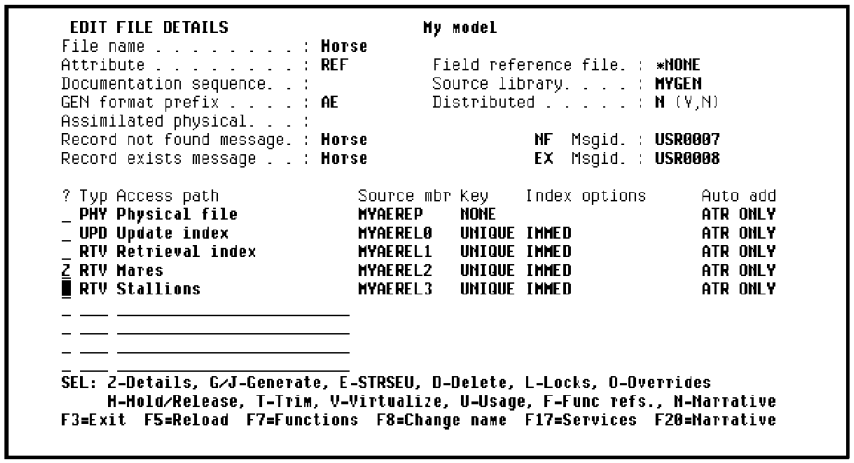
Press Enter.

Confirming Addition of Access Paths

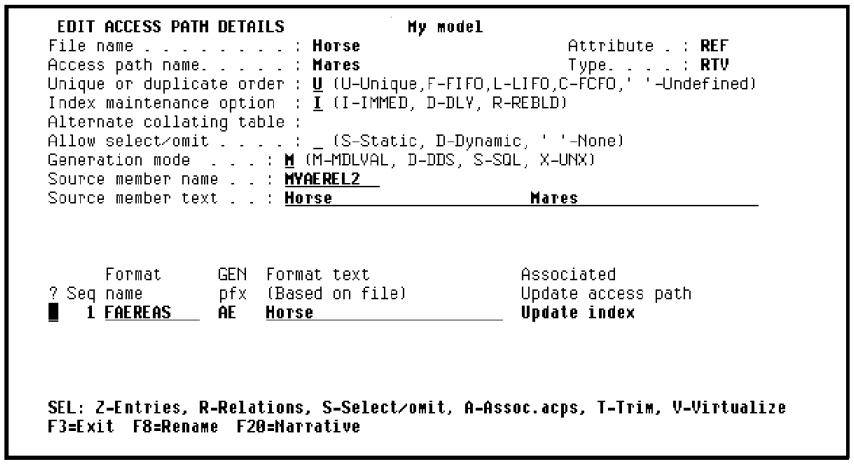
You have now added the two new access paths.CA 2E immediately assigns a source member name and default properties to the new access paths.

Access Path Details

The RTV type access path for Mares must select only female horses from the HORSE file. To specify this condition, type **Z** against the Mares Retrieval access path.



Press Enter to display the Edit Access Path Details panel.



This panel supplies the parameters that control the i OS options for that access path. In this tutorial, we will discuss only those options you need to define conditions; namely, Select/Omit Sets and Static and Dynamic Selection. See your IBM manuals for details about the other options.

Select/Omit Sets

A *select/omit set* lets you select or omit records from a view based on data values in specified fields. Specifying a select/omit set for an access path is a two-level process.

You can specify none, one, or many select/omit sets for a given access path. Multiple select/omit sets are joined with a logical OR; in other words, once a record satisfies a select/omit set, other sets are not relevant.

Within each select/omit set, you define one or more conditions. Multiple conditions are joined with a logical AND; in other words, all conditions need to be true for the entire set to be true.

In this tutorial, there is only one select/omit set for each access path and only one condition attached to each.

Static and Dynamic Selection

There are two kinds of access path selection available under i OS: static and dynamic.

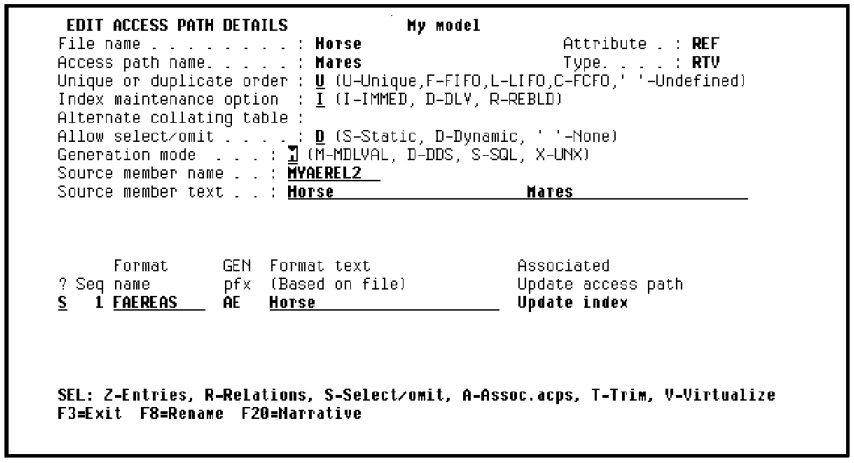
Static selection is permanently built into the access path of the i OS logical file so that the logical file only contains the records that meet the selection criteria.

Dynamic selection is not stored in the access path, but is applied to each record as it is retrieved.

You will use dynamic selection in this tutorial.

Selection Conditions for the Mares Access Path

To specify the condition that mares must be female for the Mares access path, create a select/omit set for the access path, and attach conditions to this set. Type **D** in the Allow select/omit field to select Dynamic i OS selection and type **S** in the Subfile selector to define the select/omit criteria.



Press Enter to display the Edit Access Path Select/Omit panel.

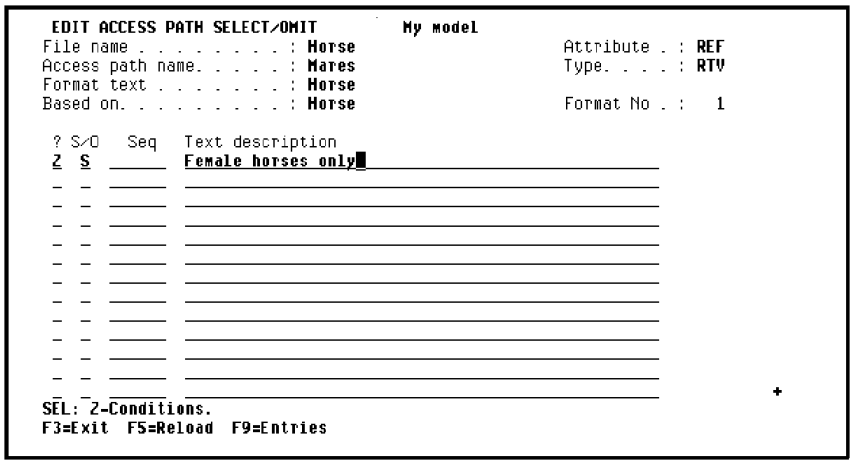
Select/Omit Set for Mares Access Path

To define the select/omit set, you will need to enter the following information on the Edit Access Path Select/Omit panel.

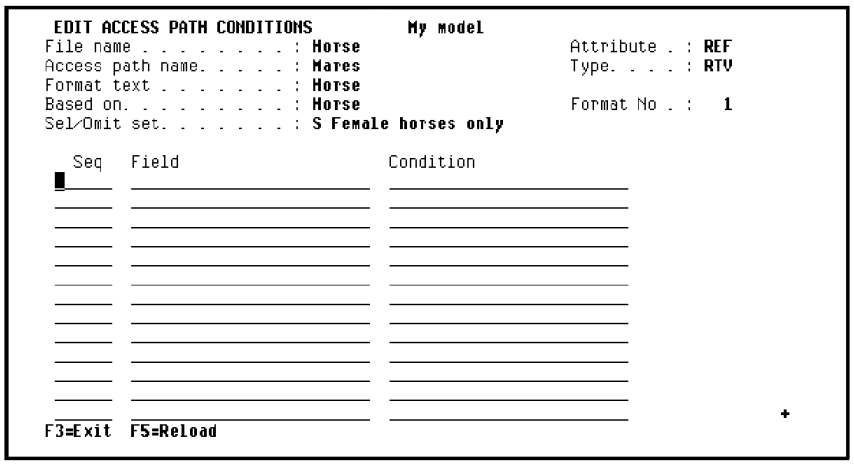
- **Z** against the line defining the select/omit set to zoom into another panel where you enter details (the conditions) for the set.
- **S** for select in the S/O column. Note that you could define a set to omit (**O**) male horses instead. The result would be the same.
- A text description for the set.

  In this case, use **Female horses only** for the description.

Type **Z, S,** and the description of the select/omit set.



Press Enter to display the Edit Access Path Conditions panel. Note the Select/Omit set field.

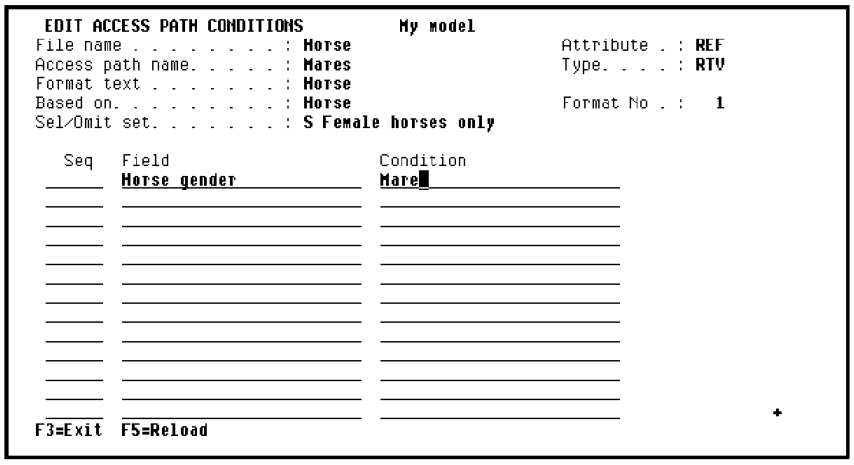


Specifying the Conditions for the Select/Omit Set

Type the conditions to be attached to the select/omit set Female horses only on this panel. In this case, the condition for the select/omit set will be based on the conditions you already specified for the Horse gender field of the HORSE file. In other words, this view of the HORSE file is to contain only records that have a value of Mare (represented by an F) for the Horse gender field. To enter this condition, type the following:

Horse gender in the Field column to specify the field on which the condition is to be based.

Mare in the condition column to indicate the value the Horse gender field must contain for the record to be selected.

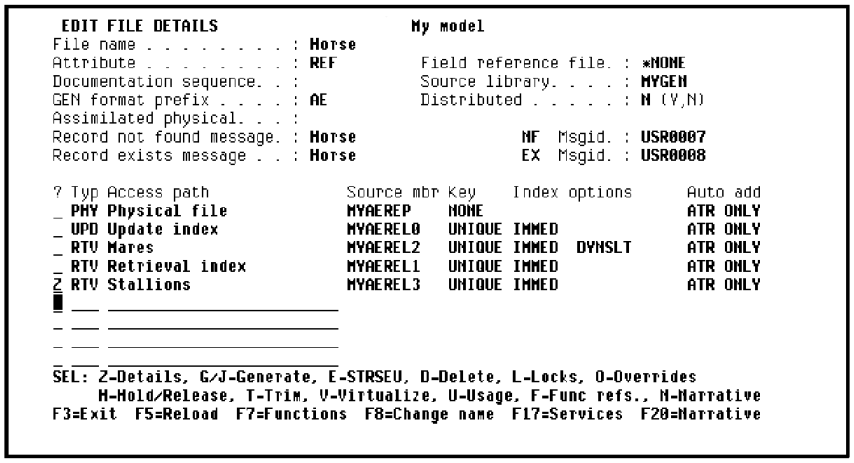


Note: If you type **?** in the Field column,CA 2E displays a list of the fields on the access path on which selection could be based. If you type **?** in the Condition column,CA 2E displays a list of possible condition values for the associated field.

Press Enter to define the condition. The process is now complete for the selection of Mares. Press F13 (Fast exit) to return to the Edit File Details panel.

Selection Access Path for Stallions

You can now specify a selection access path for Stallions by repeating the process you just used to specify a selection access path for Mares. Type **Z** against the Stallions access path.

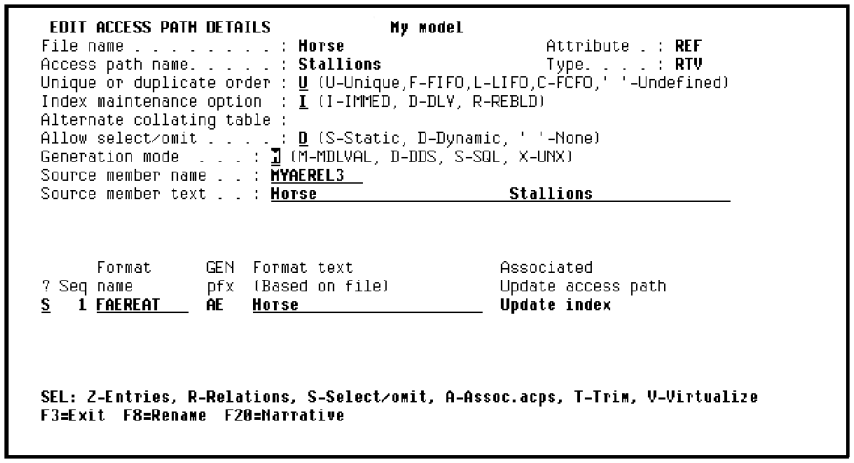


Press Enter.

Access Path Details

To specify the condition that stallions must be male for the Stallions access path, create a select/omit set for the access path, and attach conditions to this set.

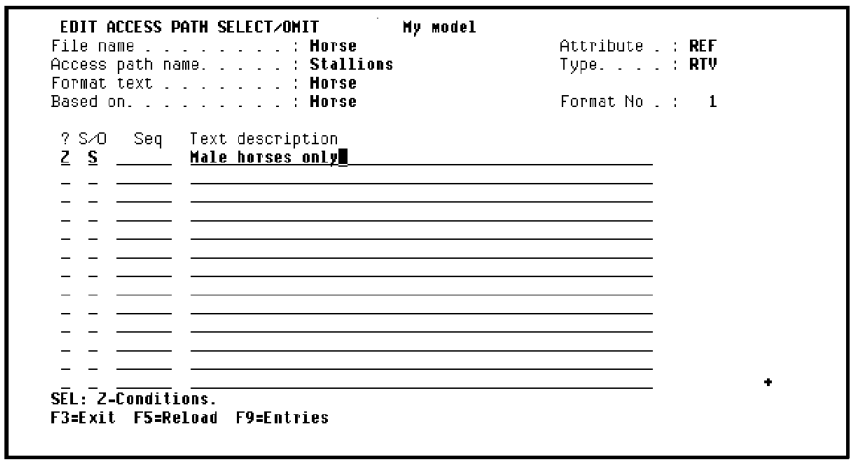
Type **D** in the Allow select/omit field to select Dynamic i OS selection and type **S** in the Subfile selector to define the select/omit criteria.



Press Enter.

Naming the Select/Omit Set

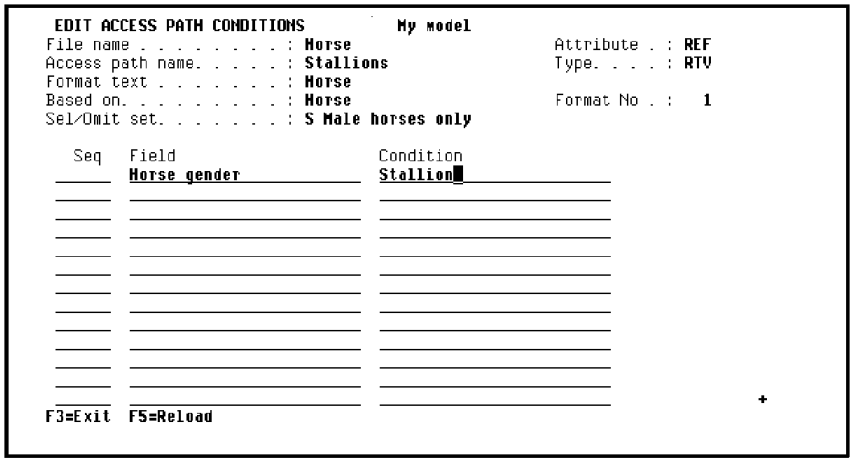
The select/omit set for Stallions should select Male horses only. Type **Z**, **S**, and the description of the select/omit set.



Press Enter.

Specifying Conditions for the Access Path

The selection condition for this access path is also based on the conditions you previously entered for the Horse gender field of the HORSE file. Type the name of the field **Horse gender** and the Selection condition **Stallion**.



Press Enter to finish defining the Male horses only select/omit set. Press F13 (Fast exit) to return to the Edit File Details panel.

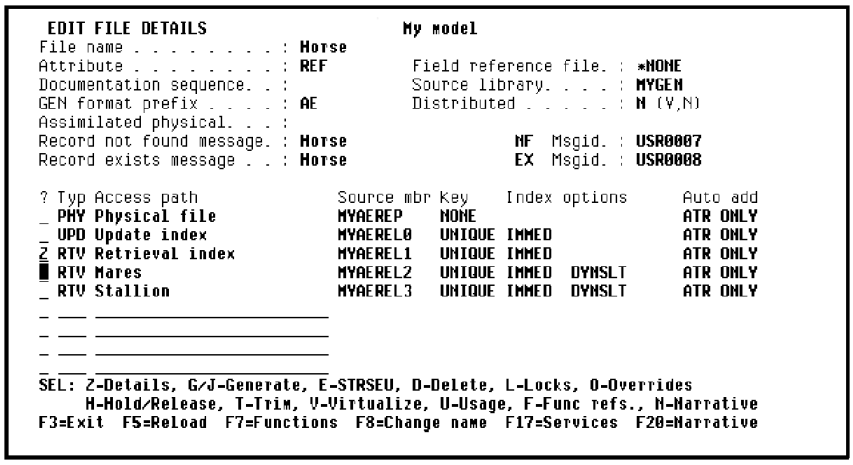
Access Path Relations

The entries to be included on the record format for an access path are defined by the access path relations. Access path relations can include all the file relations, or be a subset of them. The key level relations, however, must be present on every access path.

Modifying the Default RTV Access Path for HORSE

In this step you will modify the default Retrieval access path for the HORSE file to ensure that Dams are female and Sires are male.

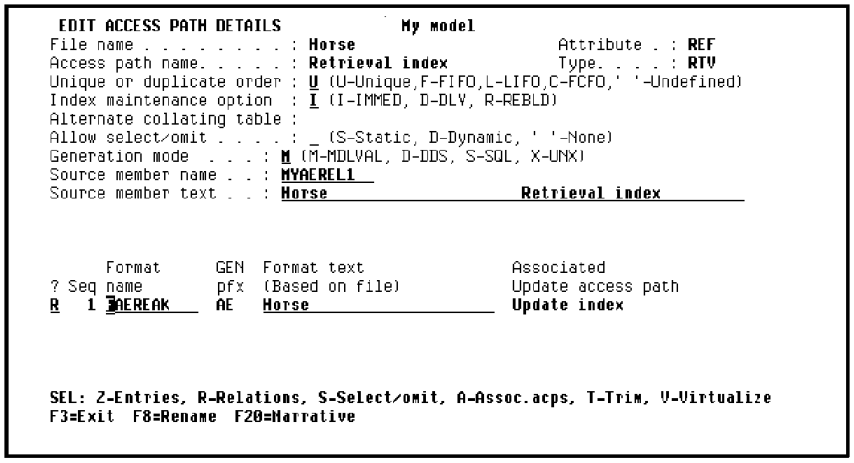
First display the access path details by zooming into the default Retrieval index. Type **Z** against the default RTV access path.



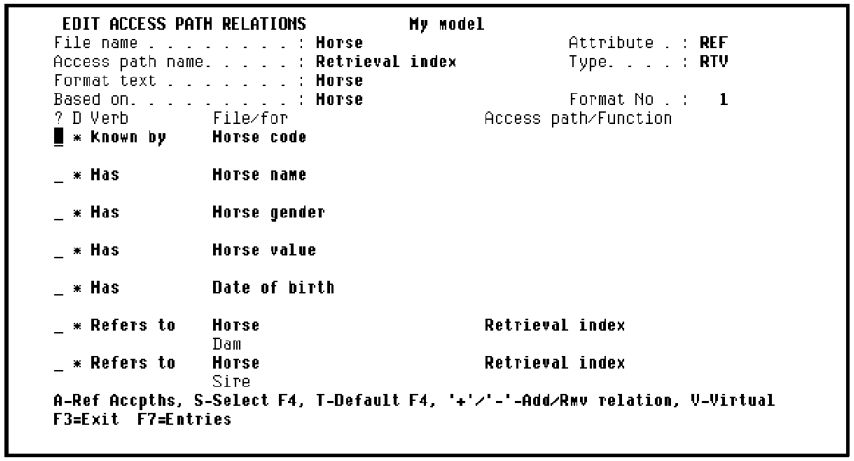
Press Enter to display the Edit Access Path Details panel for the default retrieval access path.

Displaying Access Path Relations

You are now ready to view the list of access path relations for the HORSE Retrieval index access path. Type **R** in the Subfile selector.



Press Enter to display the Edit Access Path Relations panel.



Access Path Relations for the Retrieval Index

The default Retrieval index access path includes all the relations that define a file; in other words, it contains all the fields in the file. The included relations are indicated by * in the D column. You can drop non-key relations in order to include a subset of fields in your logical view of the data. You can later reinstate a dropped relation.

Note that each of the two Refers to access path relations involves a reference to another access path. Any reference to another file by an access path relation is by means of an access path of the referenced file known as the *referenced access path*. By default this is the Retrieval index of the referenced file.

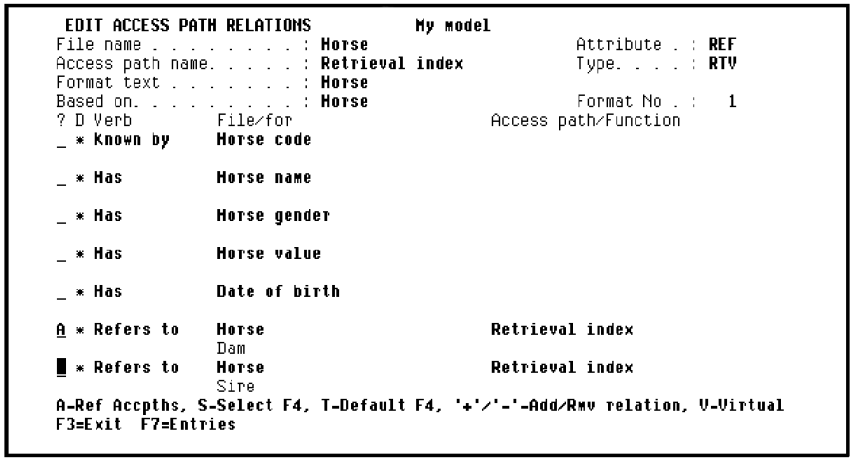
Involution and Access Path Relations

Because the HORSE file Refers to itself in your model (involution), you need to change the default referenced access path for these relations. Otherwise the same access path will be called more than once in functions that use the file. Note that you need to change the default referenced access path only when involution takes place.

The following sections describe how to change the referenced access path for the two Refers to access path relations.

Changing a Referenced Access Path

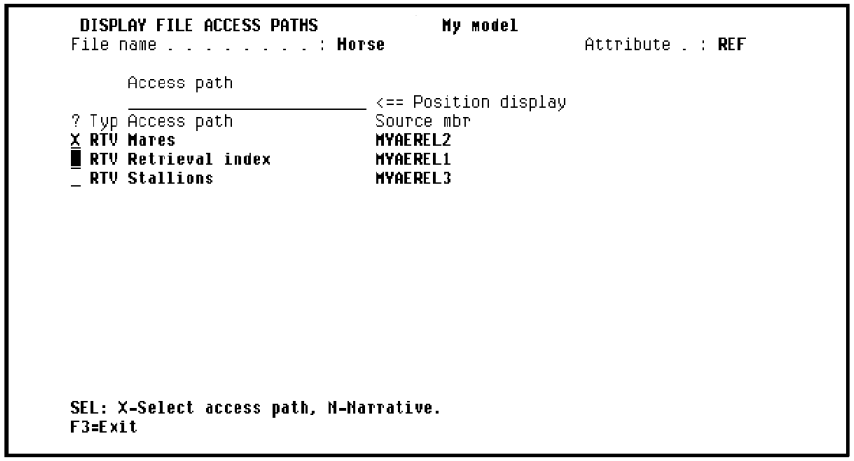
To change the referenced access path used by the HORSE Refers to HORSE For Dam relation, type **A** against the relation.



Press Enter to view access paths for the referenced file.

Selecting an Access Path for Mares

Select the access path that is to be used in place of the default referenced access path. In this case, select the Mares access path by typing **X** against Mares.

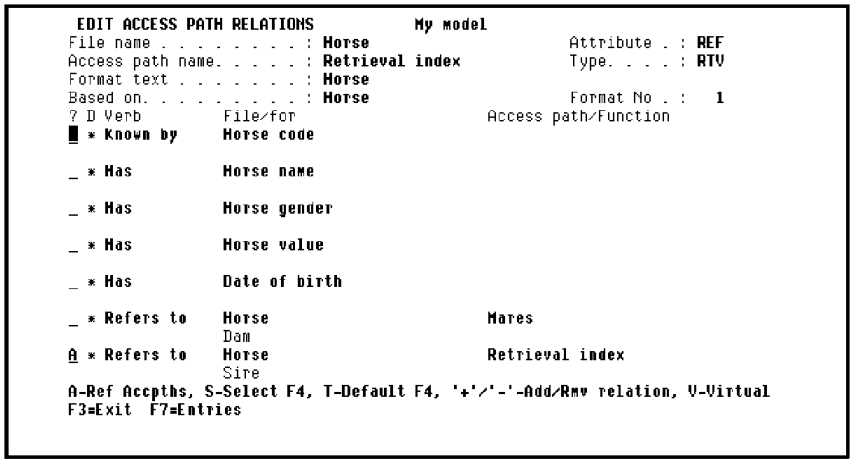


Press Enter to select the access path and return to the Edit Access Path Relations panel.

Selecting an Access Path for Stallions

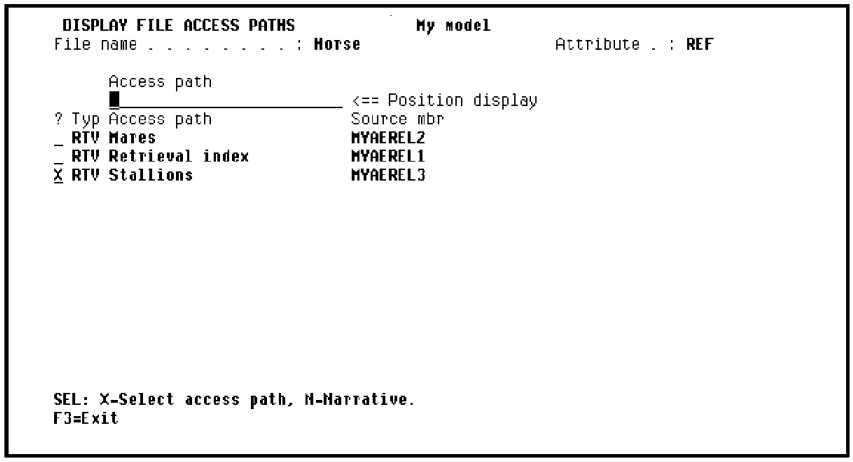
Repeat this process for the HORSE Refers to HORSE For Sire access path relation. In other words, change the referenced access path for HORSE Refers to HORSE For Sire from Retrieval index to Stallions.

Type **A** against the HORSE Refers to HORSE For Sire relation.



Press Enter to view the access paths available for selection from the referenced file.

Type **X** against the Stallions access path.



Press Enter.

Press F13 to return to the Edit File Details panel. Press F3 to return to the Edit Database Relations panel.

Chapter 4: Designing Functions

This chapter introduces the following topics:

- Functions
- Device Designs
- Action Diagrams
- Function Options
- Linking Functions
- Function Parameters

This section contains the following topics:

Introduction to Functions (see page 109)
Device Designs (see page 117)
Action Diagrams (see page 151)
Function Options (see page 189)
Linking Functions (see page 192)
Function Parameters (see page 204)

Introduction to Functions

This topic introduces you to CA 2E functions.

New terms introduced:

- Function
- External function
- Internal function
- Function type
- Edit File (EDTFIL) function type
- Select Record (SELRCD) function type

New panels introduced:

- Edit Functions
- Edit Function Details

Objectives

You will design interactive panels for the four following functions:

- Edit the COURSE file
- Select records from the COURSE file
- Edit the HORSE file
- Select records from the HORSE file

You will also define two additional functions:

- Select Stallions
- Select Mares

Overview of Functions

A *function* defines a list of processes that will operate on the files and fields in your database. CA 2E provides a variety of standard functions to which you can add your own unique processing. CA 2E uses the information contained in your data model to provide all of the default processing for the standard functions.

The set of standard functions consist of database functions, device functions, and user functions. Note that only the first two will be covered in this tutorial. The standard functions consist of a variety of *function types*, each of which performs a specific process. For example, the Edit File (EDTFIL) function type retrieves data from a file and lets the end user update that data using an interactive panel. The Print File (PRTFIL) function type prints a report of the data from a file. The Create Object (CRTOBJ) database function creates a record in your database.

A function can be implemented either as a separate program or as a subroutine within a program as follows:

- An *external function* is implemented as a separate high-level language (HLL) program. Each external function is attached to an access path. In general, the standard device functions are external functions; for example, the Edit File function. The only exception is the Print Object (PRTOBJ) function that is covered in a later chapter of this tutorial.
- An *internal function* is implemented as source code within that of the calling program; in other words, it is implemented as a subroutine. All the standard database functions are internal functions; for example, the Create Object function.

There need not be a one-to-one correspondence between a function and the program that implements it. You can link functions together to create larger processes that become the building blocks of your applications; namely, several internal functions may be implemented as a single program. In other words, a function is a unit of specification, not implementation.

The standard functions may have the following components. Note that not all function types have all four components.

- **Device design**—Specifies the layout of the panel or report used by the function
- **Action diagram**—Specifies the processing steps that make up the function (includes default processing and processing that you define)
- **Function options**—Specifies default features of a function
- **Function parameters**—Specifies fields to be passed into the function and returned from the function at run time

Each of these components is discussed in this chapter.

In addition to the standard functions, CA 2E provides message functions, function fields, and built-in functions, which perform low-level processing such as arithmetic operations. These will all be discussed in this and the following chapters.

Default Functions

CA 2E automatically creates five default functions for each reference file (REF) you declare: two external functions (Edit File and Select Record), and three internal functions (Change Object, Create Object, and Delete Object). Note that for capture files (CPT), CA 2E automatically creates only the three internal functions.

Default External Functions

The two external functions CA 2E automatically creates are:

- An *Edit File* (EDTFIL) function

  This type of function maintains multiple records on the file. An example in this tutorial will be the Edit Course function.

- A *Select Record* (SELRCD) function

  This type of function provides a display of the records on the file, and allows the selection of one record for return to a calling program. An example in this tutorial will be the Select Course function.

Both of these external functions are interactive device functions. Note that CA 2E does not automatically create these device functions for capture files (CPT).

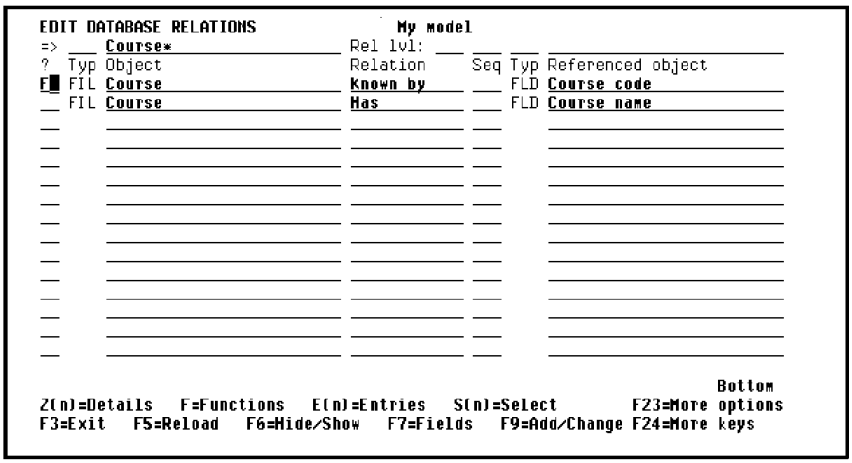
Default Internal Functions

The three internal functions CA 2E automatically creates are Change Object (CHGOBJ), Create Object (CRTOBJ), and Delete Object (DLTOBJ). These functions are used to update the database and are called by external functions as required. They are implemented as subroutines within the generated source code for an external function. Internal functions are specified separately to give you the capability to add additional processing for updating the database. This additional processing is made available to all the functions that call the internal functions.

Defining Edit Course and Select Course

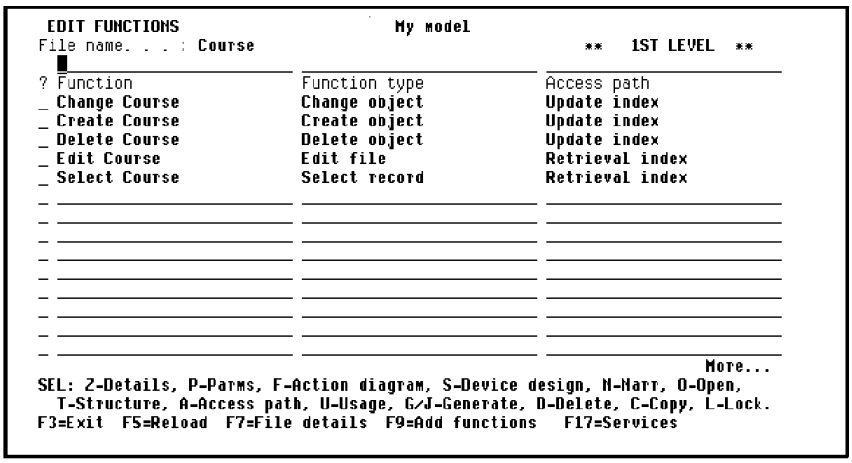
First, display just those relations defined for the COURSE file by entering **Course*** on the selection line of the Edit Database Relations panel.

You can display the functions for a particular file by typing **F** against the file on the Edit Database Relations panel. Type **F** next to any COURSE file relation.



Press Enter to view the five default functions for the COURSE file. The functions for a file are created the first time you enter **F** against the file on the Edit Database Relations panel.

CA 2E displays messages at the bottom of the panel to indicate this.

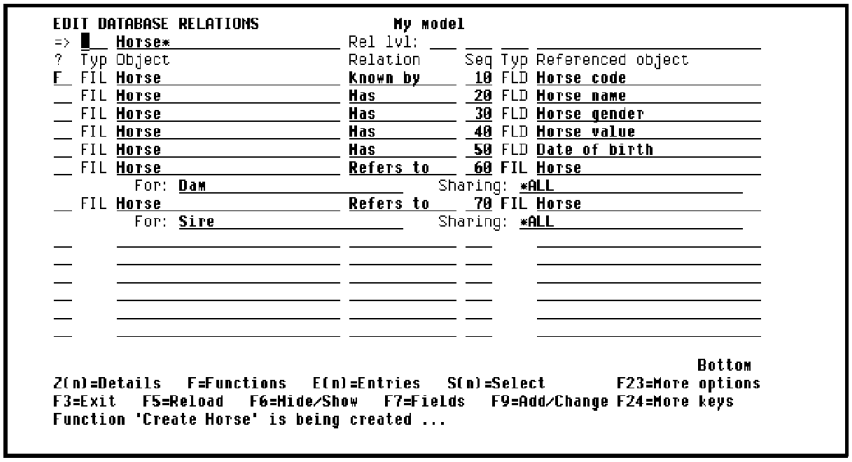


Press **F3** to return to the Edit Database Relations panel.

Displaying Default Functions For HORSE

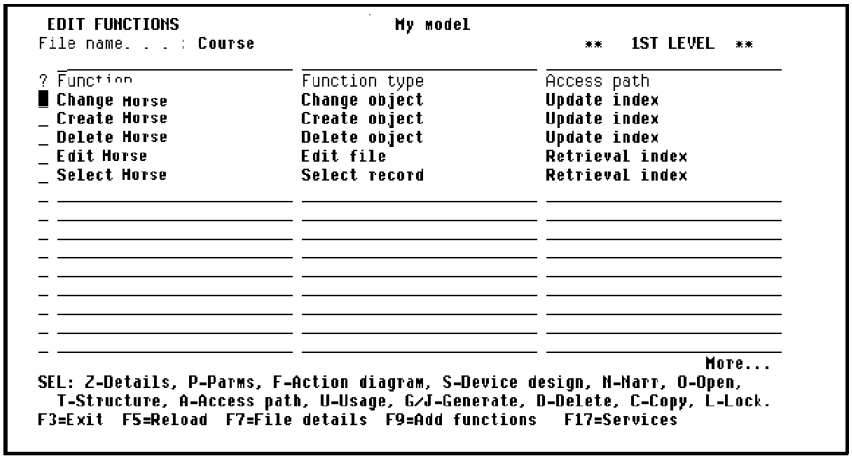
Type **Horse*** on the selection line and press Enter to display only the relations defined for the HORSE file.

Display the functions for the HORSE file by typing **F** against any relation on the HORSE file.



Press Enter to display the default functions created for the HORSE file.

CA 2E creates the default functions the first time you access the functions for a file. Messages to indicate this appear at the bottom of the Edit Database Relations panel. For example, "Function ‘Edit Horse’ is being created."



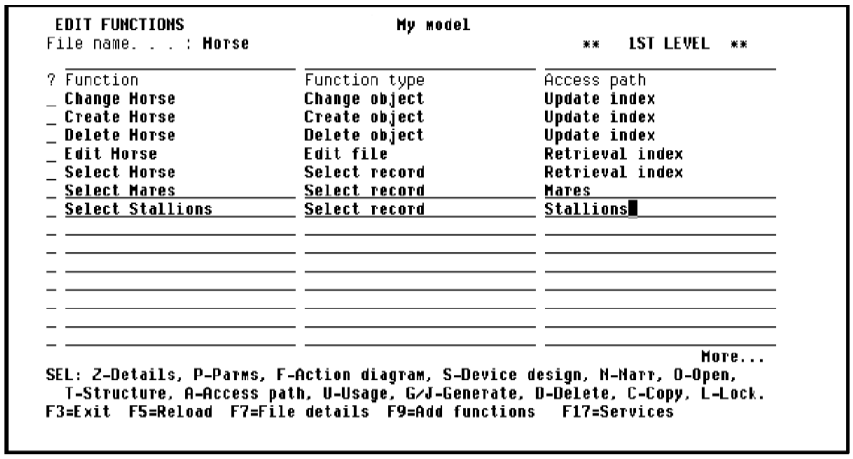
Adding Functions

You can add your own functions to the list supplied by CA 2E. To declare a function, specify a function name, a function type, and the name of the access path that defines how data from the file is presented to the function. You can view a list of possible function types by typing **?** in the Function type column, and a list of the existing access paths based on the HORSE file by typing **?** in the Access path column.

Select Mares and Select Stallions Functions

In this step you will add two new Select record functions, Select Mares and Select Stallions. These functions will enable you to display a list of mares only or stallions only in response to an inquiry. These functions will use the Mares and Stallions access paths you previously defined.

Type the function names, function types, and access paths.



Press Enter.

Creating New Functions

The two messages "Function ‘Select Mares’ is being created" and "Function ‘Select Stallions’ is being created" will appear briefly at the bottom of the panel. The functions will now be declared to your design model and are available for reference by other functions. At this stage no further definition is required.

Note: You will need to define additional aspects of the new functions later in this tutorial before they can be implemented.

Understanding Function Details

Each interactive external function will be implemented by means of:

- A High Level Language (HLL) program
- A DDS display file
- A file containing Help text

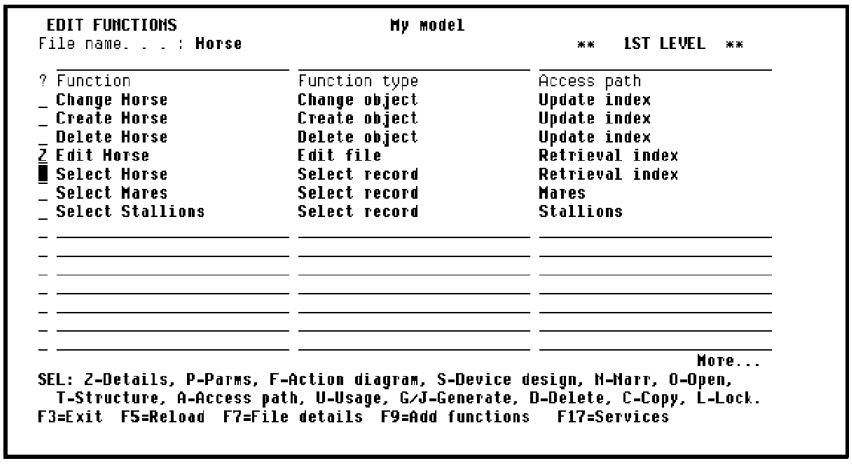
An appropriate source member is created to contain each type of generated source. CA 2E supplies default names for the source members. You can change the default source member names using the Edit Function Details panel.

The implementation language is controlled by the target HLL (higher level language). You may generate in RPG, COBOL, UNIX, or Client Server Applications depending on your version of CA 2E.

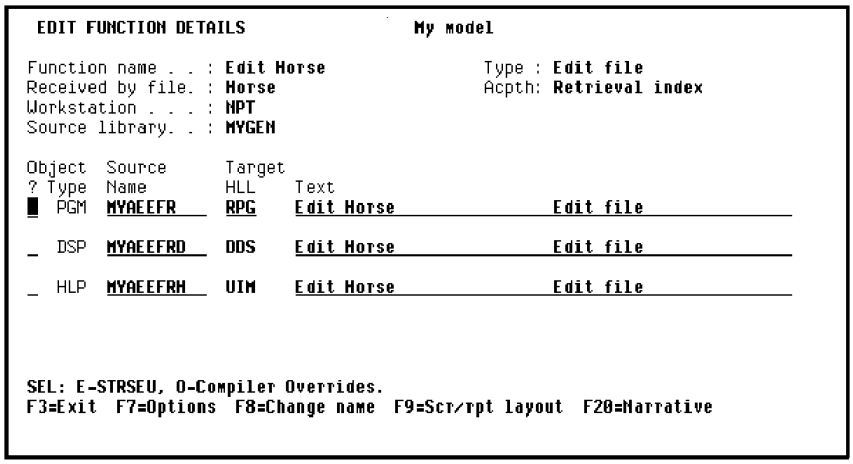
The default implementation language for new functions is controlled by the YHLLGEN model value.

Displaying Function Details

You will now display the details of the Edit Horse function. Type **Z** against the Edit Horse function.



Press Enter to view function details for the Edit Horse function. Note the program (RPG), display file (DDS), and the help text file (UIM).



Device Designs

CA 2E automatically creates a default interactive panel design for each interactive device function. You can modify the default designs to meet the requirements of your application. This topic describes how to use the CA 2E Device Design Editor to modify the device design associated with your Edit Horse and Select Horse functions.

The Edit Horse function will show the use of an action bar (pull-down menu) as part of the panel design. The Select Horse function will show the use of a window.

New terms introduced:

- CA 2E device design
- Usage

New panel introduced:

- Edit Function Devices

Overview of Device Designs

When you declare a function, CA 2E supplies a default panel design. You can modify it to meet your requirements using an interactive panel design tool.

A device design specifies the following:

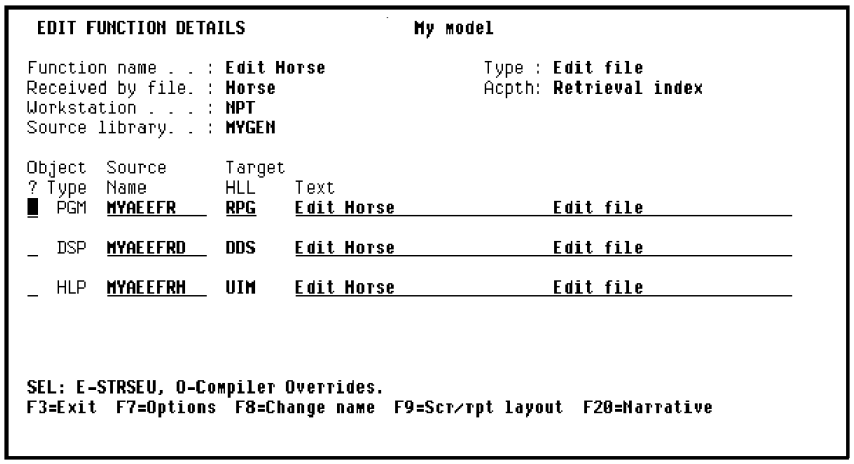
- The selection of fields present on a device panel and the accompanying text for each field
- The display of both fields and text may be conditional.
- The order in which the fields are displayed on the device panel and how they are edited
- The display attributes for the fields
- The use of each field

All of the above are defaulted from the design model. You may override most of these default values.

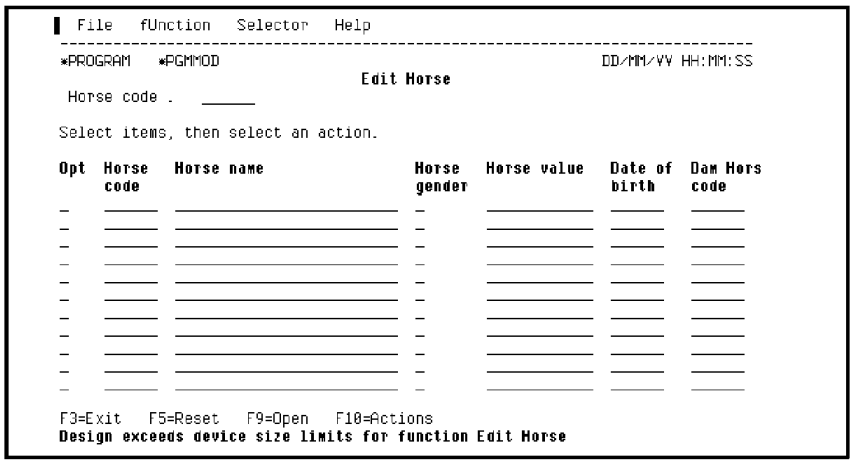
Displaying the Device Design for Edit Horse

The Edit Function Details panel for the Edit Horse function should be displayed on your screen. If it is not, follow the steps at the end of the preceding topic.

There is only one panel design for the Edit Horse function. You can view the default device design from the Edit Function Details panel by pressing F9.



The default panel design for the Edit Horse function displays.



Note: Do not be concerned about the message at the bottom of the panel.

Default Device Design Formats

A panel design consists of header/footer formats and a variable format that depends on the function type. The panel design for the Edit Horse function, which is an Edit File function type, consists of the following formats.

- The uppermost format contains the header lines that give status information about the user, date, time, and mode (New/Open) of the panel.
- At the bottom of the panel, the footer lines give function key explanations. Any execution time messages are displayed on line 24.
- The middle section of the panel contains the formats that vary depending on the function type. In this case we are using an Edit File function type. It contains two formats,
  - A subfile control format.
  - A subfile record format. The subfile record is the format for a single record that is retrieved from the fields of the specified access path according to the function type.

Design Standards and Subfile Selector Options

When the function type includes a subfile, descriptions for the Subfile selector options are included as part of the default device design. These can be

- Part of the footer along with standard function key descriptions
- Placed above the subfile record as part of the subfile control format

  This is the CUA ENTRY design standard.
- Listed as actions in the Selector Choice menu that is part of the action bar at the top of the panel

  This is the CUA TEXT design standard.

The default design standard for CA 2E design models on the IBM i is CUA ENTRY. However, when you created your design model at the beginning of this tutorial using the Create Model Library (YCRTMDLLIB) command, you were told to specify *CUATEXT as the design standard for your model.

When the design standard is set to CUA TEXT, CA 2E automatically creates all Edit File functions with an action bar on the panel design. Note the action bar at the top of the panel design for the Edit Horse function. An action bar provides a set of choices that give the end users access to various actions available on the panel. CA 2E automatically provides a default set of choices depending on the function type. Later in this tutorial you will define a new choice and action on the Edit Horse function’s action bar.

Subfile Control Default

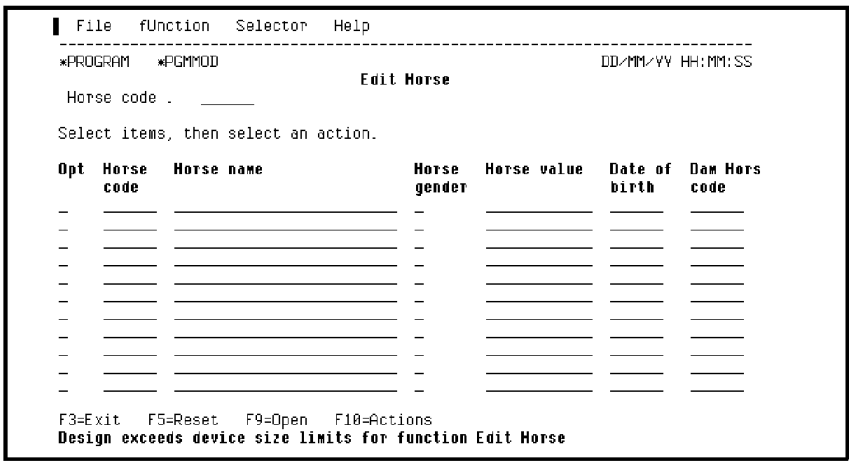
The key field of the HORSE file (Horse code) is placed on the subfile control format as well as on the subfile record. This field is a positioning field that causes the subfile record to begin at a particular record when a value is entered into this field.

Subfile Record Default

To determine the default panel design for an Edit File function, CA 2E places all the fields from the access path side-by-side on a subfile record, with the field names as column headings. If the fields total more than 80 characters, the design continues into a work area on the right. It is up to you, the designer, to adjust the design until the fields are all displayed on the panel. In this tutorial, the fields exceed the panel display. CA 2E alerts you to this fact and displays the message "Design exceeds device size limits for function Edit Horse" at the bottom of the panel.

Displaying the Rest of the Device Design

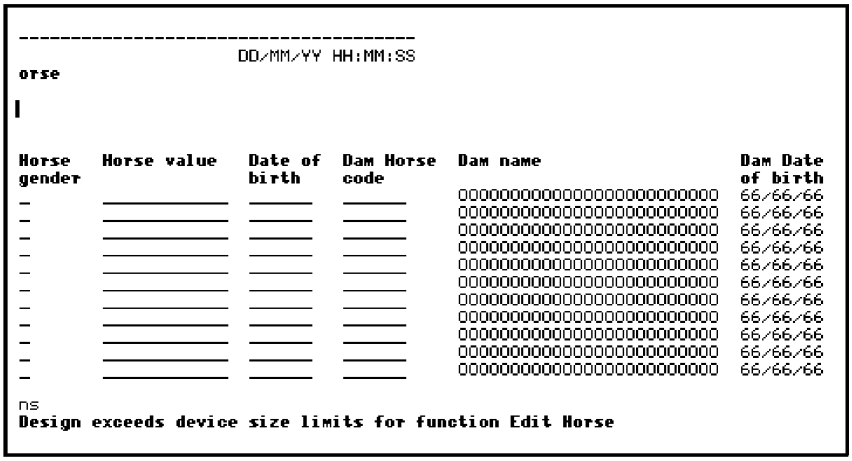
By default, only 80 characters of the panel design may be displayed at one time. You can move the panel window to display more of the panel design by pressing F4, which moves the window 40 columns to the right. Pressing F4 again moves the subfile record another 40 columns to the right. Pressing F1 moves the panel design 40 columns to the left.



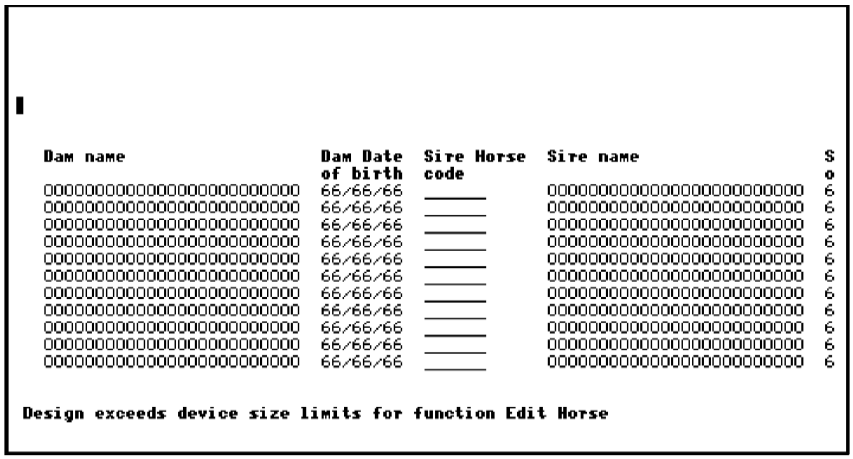
Press F4 to move the device design 40 columns to the right.

Note the virtual fields, Dam name and Dam Date of birth that you added to the default Retrieval access path for the HORSE file. If you had not explicitly added these fields to the access path, they would not appear on this display.

Note also that the virtual fields are for output only and are represented by 0s or 6s.



Press F4 again to move the device design 40 more columns to the right.



Press F15 to return to the left margin of the device design; in other words, back to the first field in the subfile record.

Editing the Default Device Design

The Device Design Editor has a number of facilities to enable you to quickly design a more compact, understandable panel. Many of these facilities involve the use of function keys and are generally dependent on the position of the cursor when the function key is pressed. The device design is updated as you make each modification.

The following table shows some of the function keys available for editing your device design. Most of these will be illustrated in this and other topics in the tutorial. To learn about additional features of the Device Design Editor, refer to the Help text or the CA 2E manual *Building Applications*.

Function Key	Action
F1	Moves the device design 40 columns to the left
F2	Animates the device design using CA 2E Toolkit prototyping facilities
F4	Moves the device design 40 columns to the right
F5	Displays the Edit Screen Format Details for the format on which the cursor is positioned
F7	Displays the Edit Screen Format Relations panel when the cursor is positioned on the first subfile record
F8	Selects a field to be moved (cut) and moves the selected field after the field on which the cursor is positioned (paste)

F9	Moves the field on which the cursor is located and all fields to its right to the next line
F10	Adds a space before the field on which the cursor is positioned; in other words, moves text one column to the right
F15	Moves device design to the left margin
F16	Moves device design to the right margin
F17	Displays a list of device formats
F18	Displays the Edit Screen Field Attributes panel for the field on which the cursor is positioned
F19	Adds a function field to the device design
F20	Edits a function field on the device design
F22	Removes a space before the field on which the cursor is positioned; in other words, moves text one column to the left

Reducing the Width of the Device Design Layout

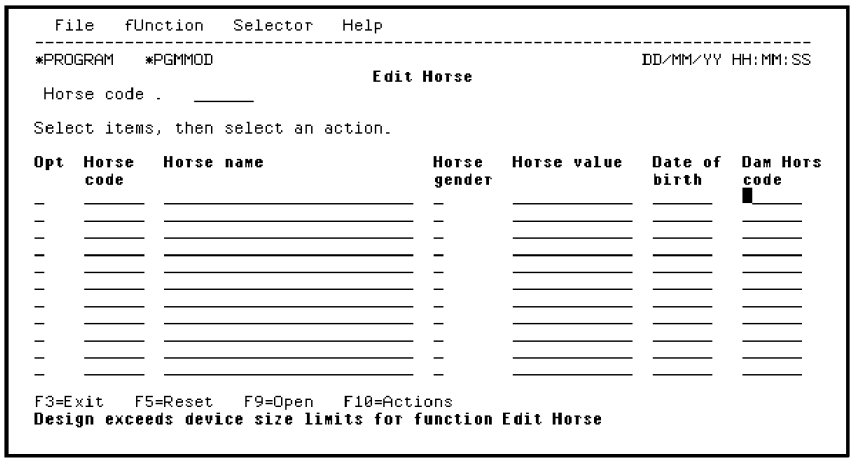
You can reduce the width of the device design by folding the subfile records that make up the device design. This way the fields in each record are arranged over several lines rather than in one long line. You will use the F9 key to do this.

The following steps in this tutorial show how to split the information about each horse into three lines that give:

- Information about the horse itself
- Information about the horse's Dam
- Information about the horse's Sire

The change in position of the fields on the device design is determined by the position of the cursor when you press the function key. To begin the second line with *Dam Horse code,* you will simply place the cursor on this field and press F9. The subfile record format will split into two lines. The device design is updated to reflect the change.

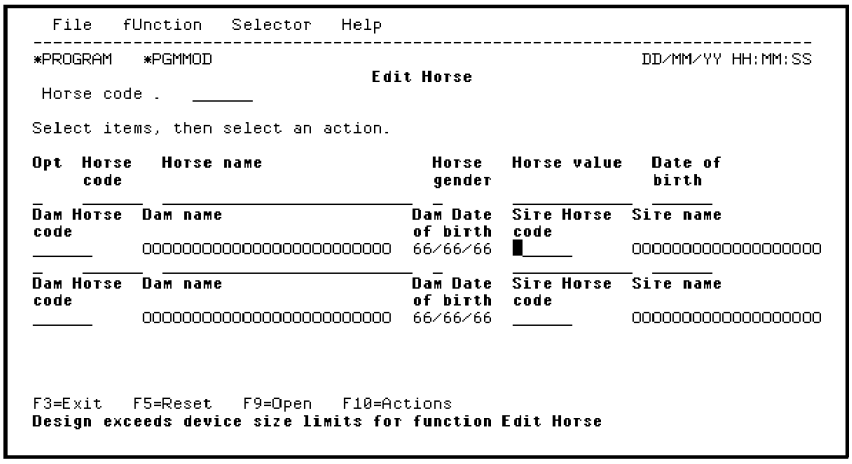
Position cursor.



Press F9. The subfile record format now displays only two HORSE records.

Folding the Device Design Layout Again

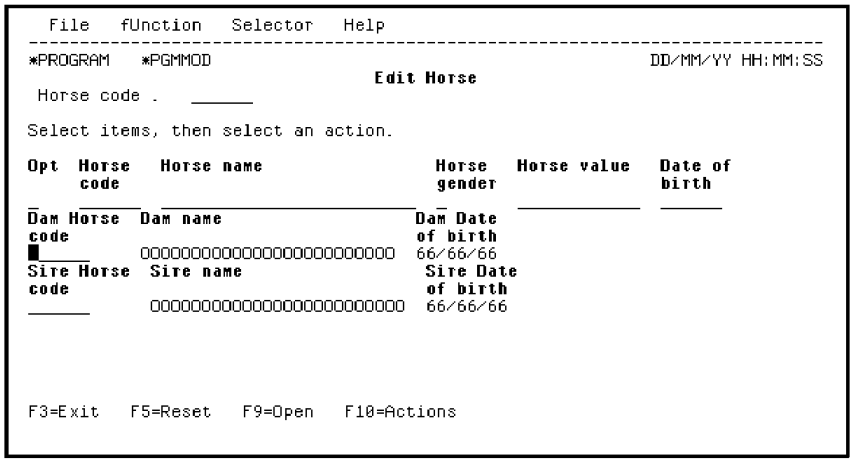
Next, repeat the process to move the fields relevant to the Sire to a separate line from those for the Dam. Place the cursor on the Sire Horse code.



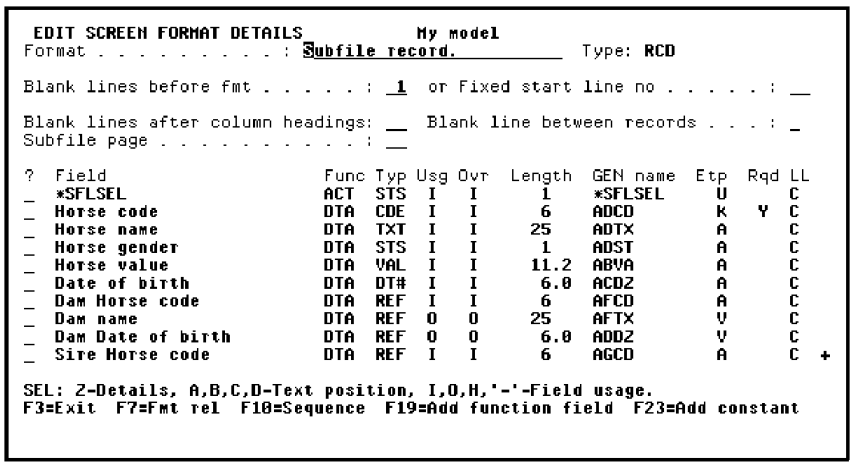
Press F9. The subfile record format now displays only one HORSE record.

Panel Format Details

You can view a list of all the fields available on any particular format using the Edit Screen Format Details panel. To access this panel for the Subfile record format, place the cursor on any field of the first subfile record.



Press F5.



Edit Screen Format Details

The Edit Screen Format Details panel shows all the fields in the Subfile record format. The details shown for each field on the format include the name and type of the field. This panel also allows you to control the positioning of the format with respect to other formats on the device design.

You can also use this panel to specify a field's *usage*; in other words, whether the field is input capable (I), output only (O), or hidden (H). This applies to any field other than the Subfile selector field (*SFLSEL). For example, you could hide a field by typing **H** against the field or you could change an input capable field to output only by typing **O** against the field.

Note: Some restrictions exist when changing an output only field to an input capable field.

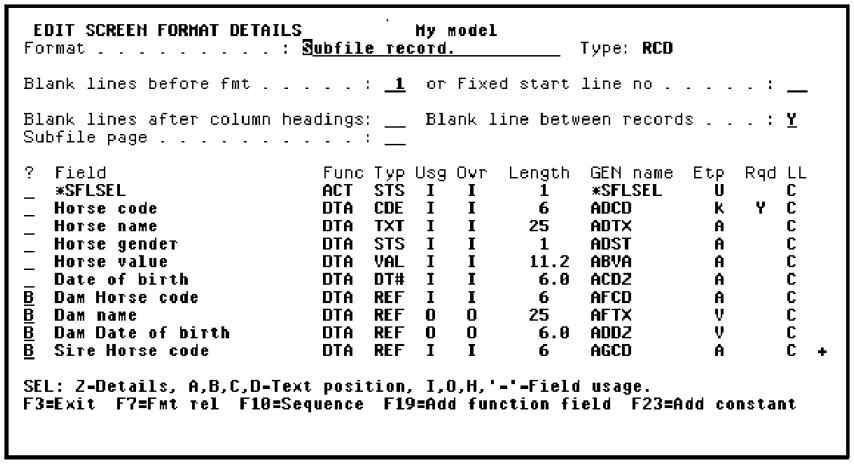
Editing the Panel Format

In this step you will use the Edit Screen Format Details panel to make the following two changes to improve the appearance of the Edit Horse panel.

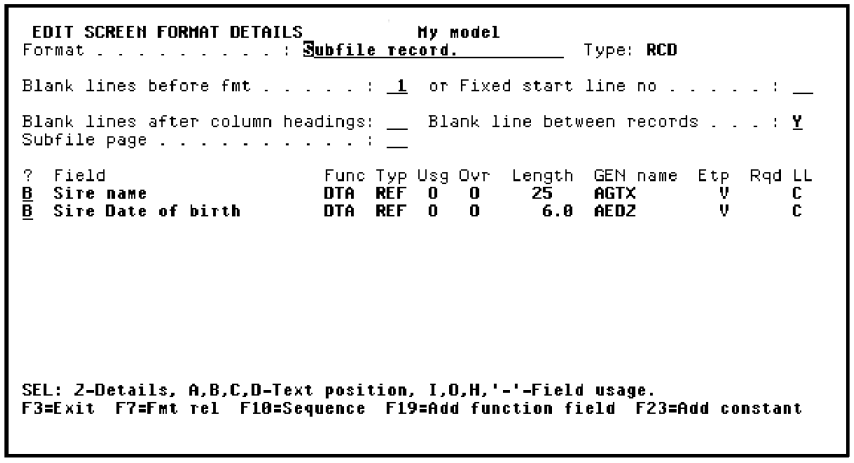
1. Separate the individual subfile records with a blank line by typing **Y** (yes) against the Blank line between records field.
2. Place field labels before the fields on the second and third lines of the record, rather than above them, to clarify the information. CA 2E has four options for positioning the label associated with a field.
   - Above (A) - on the line above a field
   - Before (B) - on the same line as the field
   - Column heading (C) - above a column of instances of the field; this value applies only to a subfile record format
   - Drop (D) - omit the label

To reposition the labels on the second and third lines, you will type **B** against each of the Dam and Sire fields. Note that later in this tutorial you will use the Edit Screen Entry Details panel to overwrite the wording of the field labels.

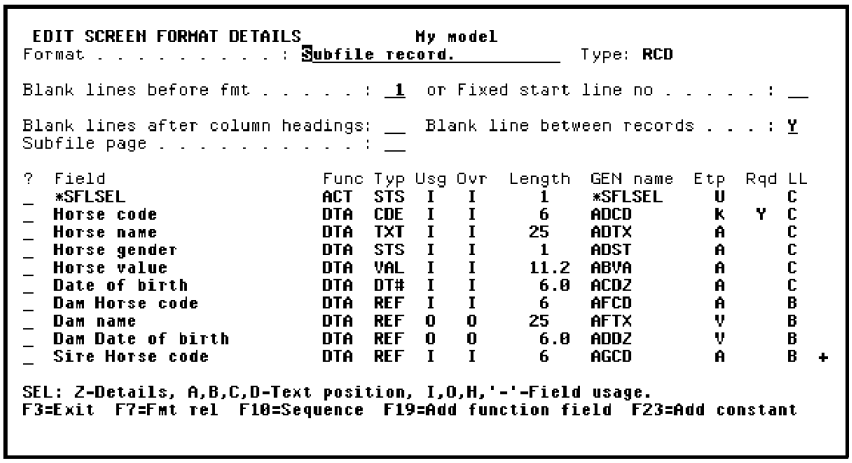
Specify **Y** for the Blank line between records field and type **B** against each of the Dam and Sire fields.



Press Roll Up to display more fields. Type **B** as shown to position the labels for the Sire name and Sire Date of birth before the field on the same line.



Press Enter to confirm the changes. Note that the value in the LL (Label Location) column for each of the Dam and Sire fields has changed from **C** to **B**. Press Roll Down to view the preceding panel.



Press F3 to return to the Device Design Editor.

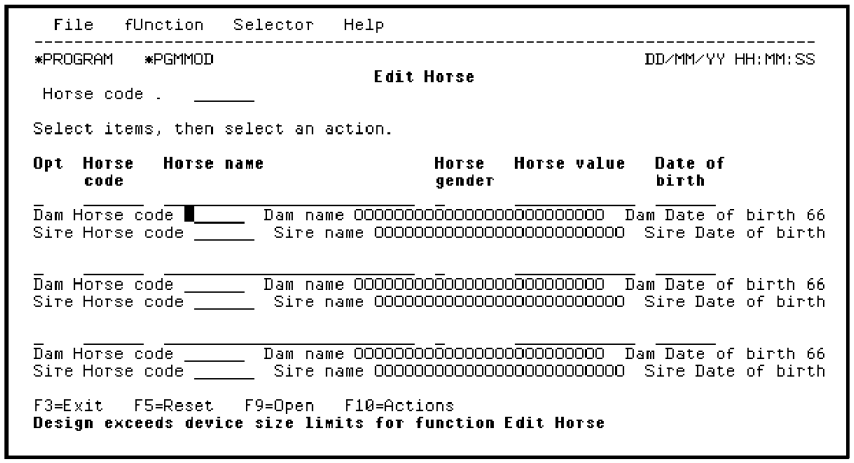
Effect of Screen Format Changes on Device Design

The updates you have made are displayed immediately when you press F3 on the Edit Screen Format Details panel. The new panel layout is clearer, with a blank line between each subfile record and the field labels shown before the fields. However, because the field labels are now on the same line as the fields, the second and third lines of the subfile record are too long to fit on the panel.

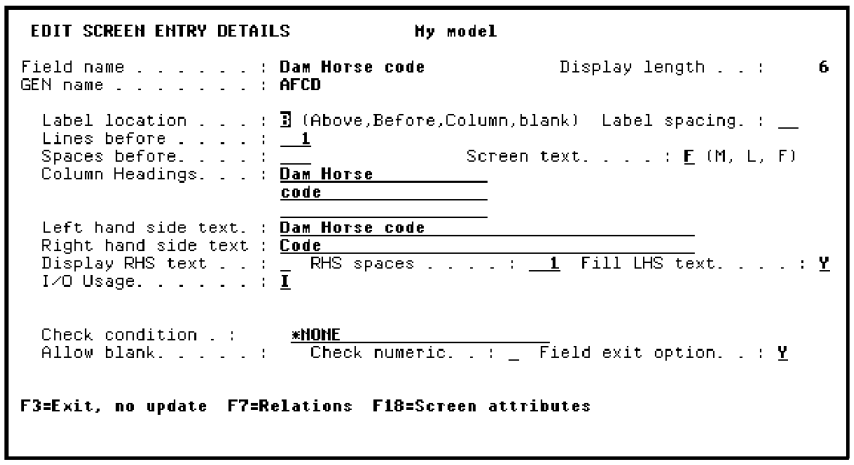
Shortening Field Labels

You can reduce the length of the lines on the panels by shortening the field labels. This is done by editing the individual field entry details using the Edit Screen Entry Details panel.

To display the field entry details for any field, place the cursor on that field or its label in the first subfile record and press the Enter key. To display the field entry details for the Dam Horse code field, place the cursor on the field.



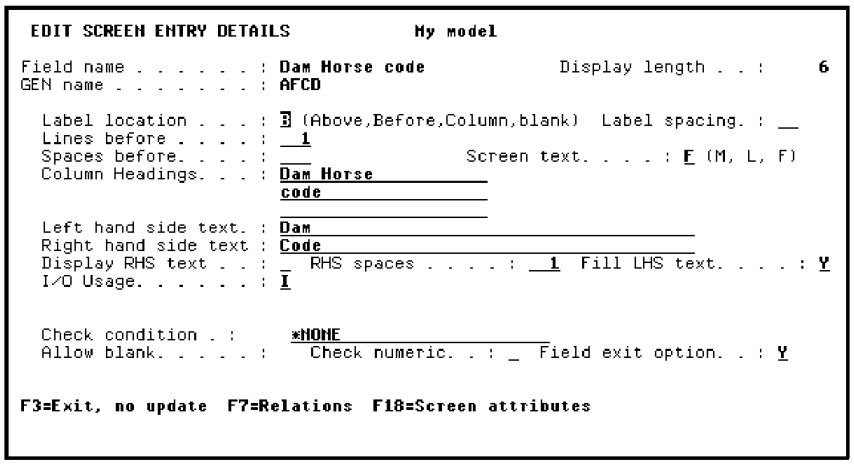
Press Enter to access the Edit Screen Entry Details panel for Dam Horse code.



The Edit Screen Entry Details panel gives details of the label location, spacing, text, and position. Note the different versions of the text to be used for the different label positions (Above, Before).

Shortening Field Label for Dam Horse Code

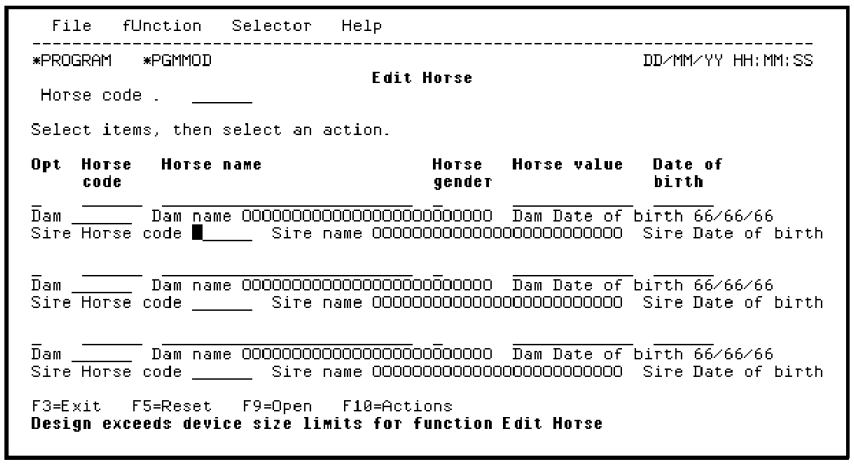
You can shorten the label before the field by changing the Left hand side text (*LHS* text). In this case, change the Left hand side text from Dam Horse code to Dam.



Press Enter.

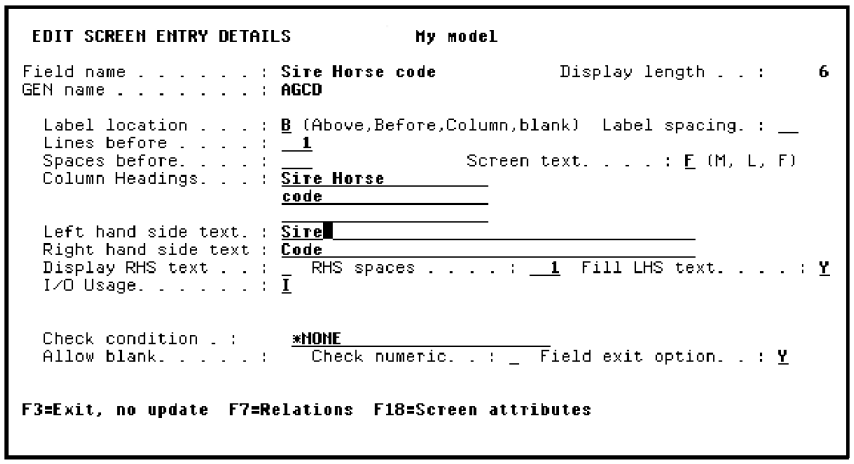
Shortening Field Label for Sire Horse Code

The second line of the subfile record format now fits on the panel. The next step is to shorten the label for the Sire Horse code field using the same process and position the cursor.



Press Enter.

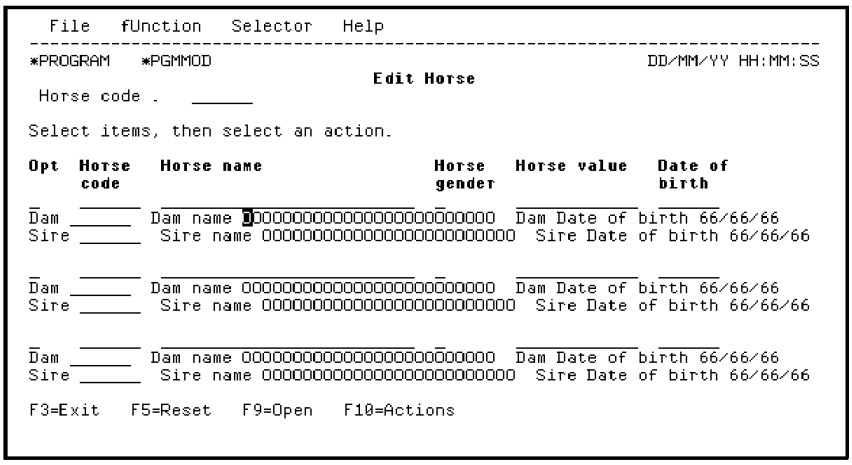
Change the Left hand side text to Sire.



Press Enter to return to the device design.

Removing Field Labels

You can compress the panel layout further by removing the labels Dam name and Sire name. To do so, first place the cursor on the Dam name field.

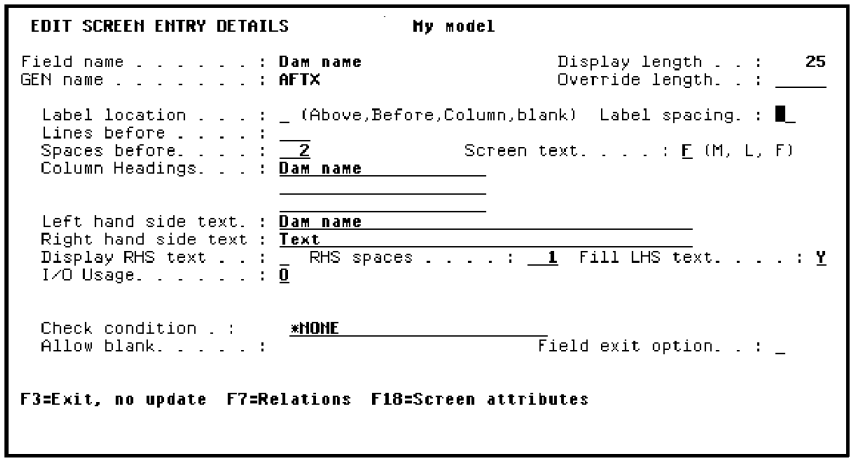


Press Enter.

CA 2E displays the Edit Screen Entry Details panel for the Dam name field.

Removing Field Label for Dam Name

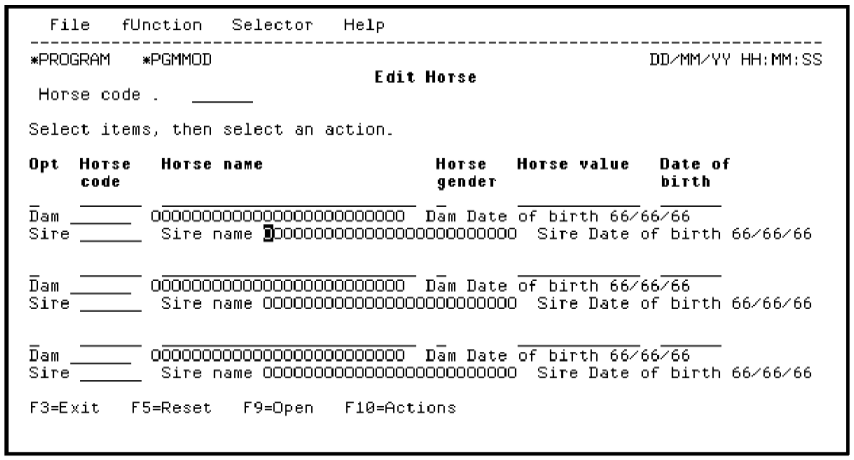
Notice the value of B (Before) in the Label location field; this caused the LHS text to appear as a label before the Dam name field. To remove the label, change the Label location field from B to blank.



Press Enter.

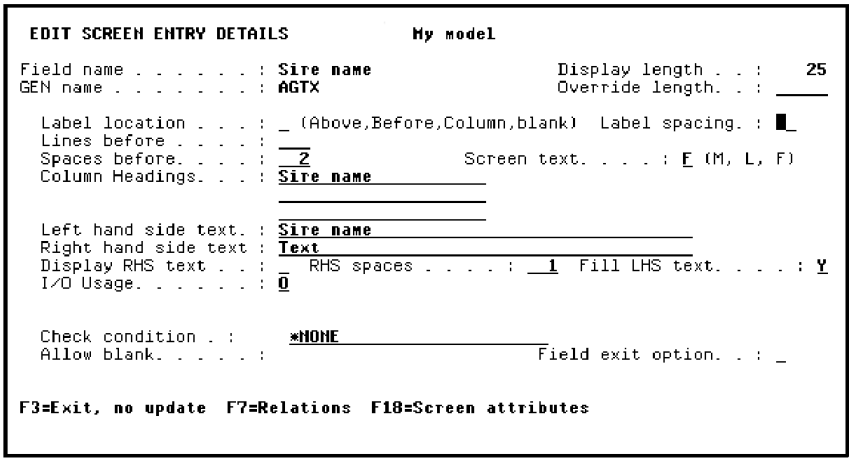
Removing Field Label for Sire Name

Remove the field label for the Sire name field in the same way. Position the cursor on the Sire name field.



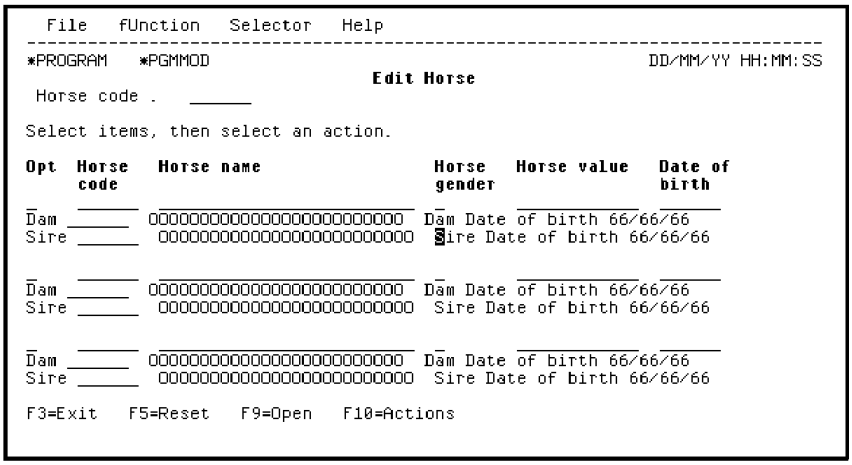
Press Enter.

Notice the value of B (Before) in the Label location field; this caused the LHS text to appear as a label before the Sire name field. To remove the label, change the Label location field from B to blank.



Press Enter.

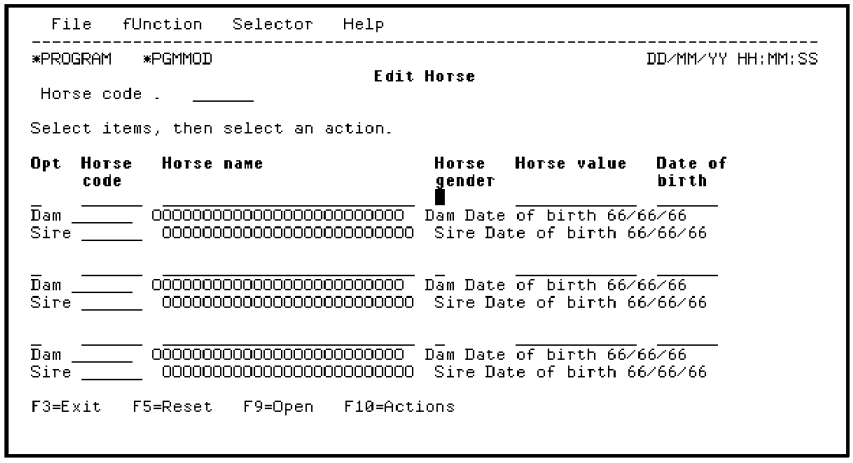
After removing field labels for the Dam name and Sire name fields, the resulting panel will look like the following screen.



Centering a Field with Respect to its Label

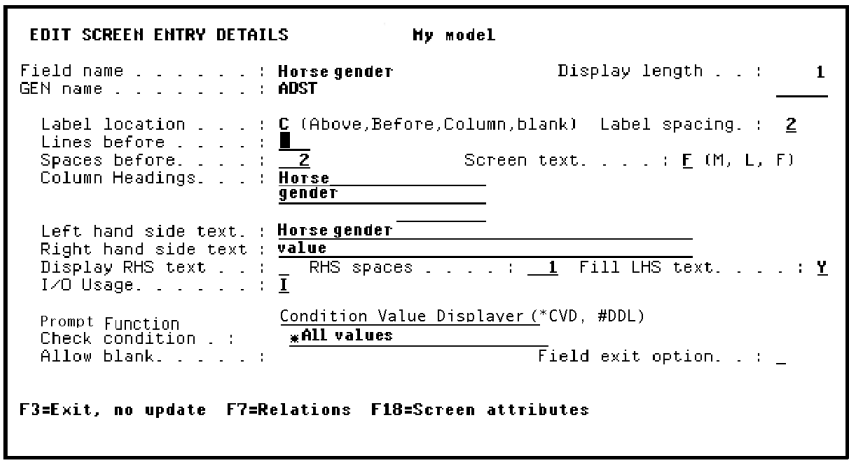
The Label spacing field on the Edit Screen Entry Details panel lets you position a field with respect to its label. This applies to fields on the same line as their label as well as to fields positioned below their label. One use of the Label spacing field is to center a field whose length is shorter than the label length; for example, the Horse gender field.

In this step you will center the Horse gender field below its label. To do this, access the Edit Screen Entry Details panel for the Horse gender field. Position the cursor.



Press Enter.

Type **2** in the Label spacing field to move the field two positions to the right and press Field Exit.



Press Enter to return to the Device Design Editor.

Note that the Horse gender field is now centered below its label.

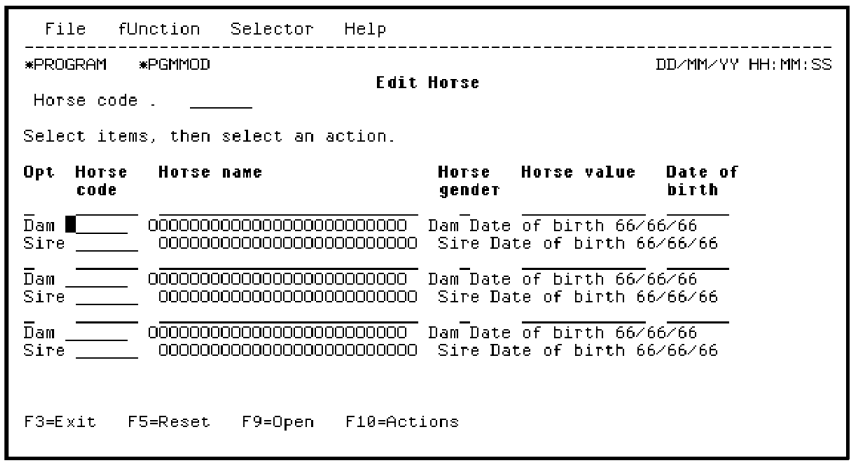
Moving Fields to Right and Left

Field positioning within a device design is relative. For example, each field is positioned with respect to the previous field in the format. You can adjust the position of a field by inserting or removing spaces before it using either function keys or the Edit Screen Entry Details panel.

Using Function Keys

For example, F10 inserts one space before the field label of the field where the cursor is positioned, causing the field and all fields to the right of it to be moved to the right. F22 removes one space before the field and moves the entire line to the left.

Position the cursor on the Dam field.

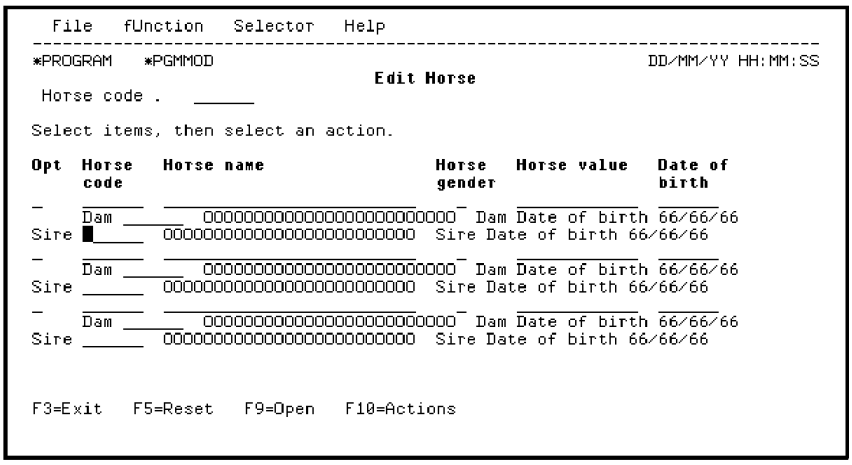


Press F10 five times to indent the line beginning with Dam by five spaces.

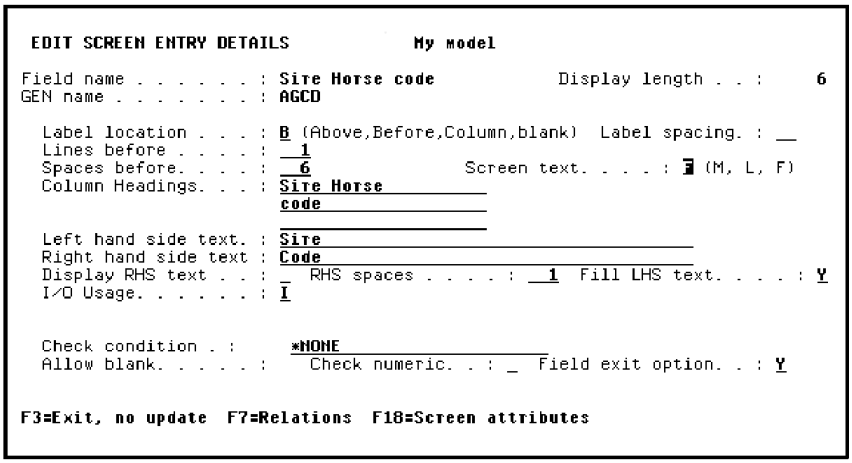
You have just moved the Dam field to the right so that the second row begins below the Horse code field.

Using the Edit Screen Entry Details Panel

You can also move fields by changing the value in the Spaces before field on the Edit Screen Entry Details panel. Position the cursor on the Sire field.



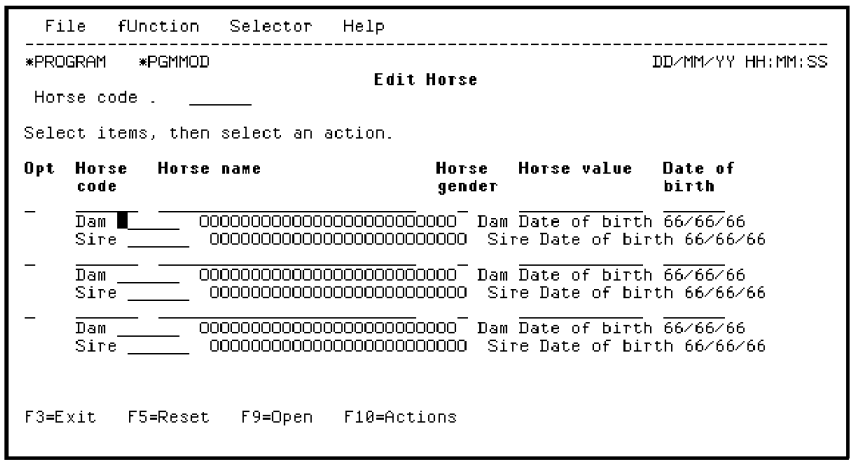
Press Enter to display the Edit Screen Entry Details panel. Change the Spaces before field to **6** as shown and press Field Exit.



Press Enter.

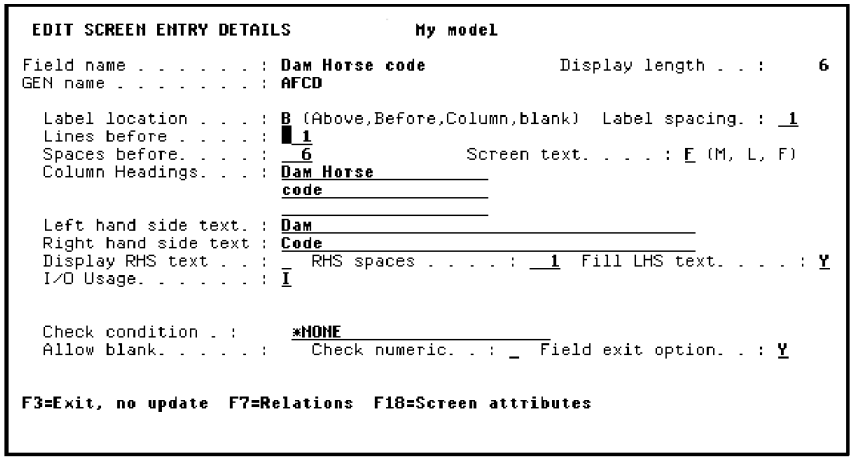
Adjusting the Label Spacing of the Dam Field

The above steps will line up the field labels for Dam and Sire. The entry fields can now be aligned by placing the cursor on the Dam field.



Press Enter.

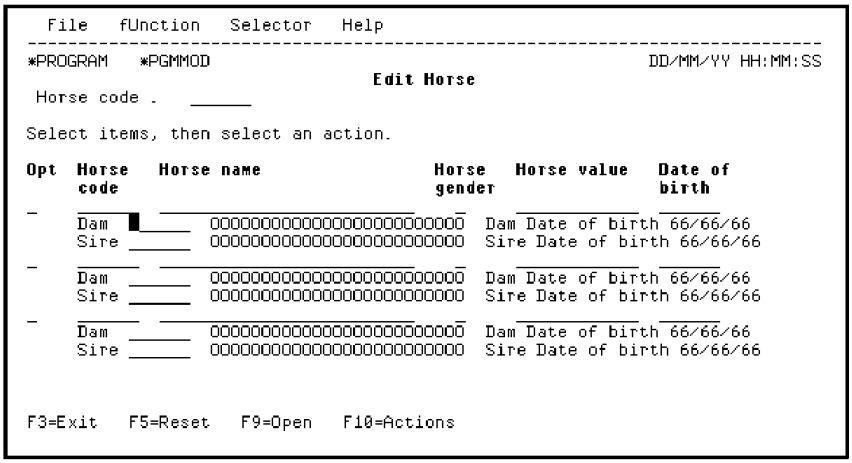
Change the Label spacing to 1 as shown and press Field Exit.



Press Enter.

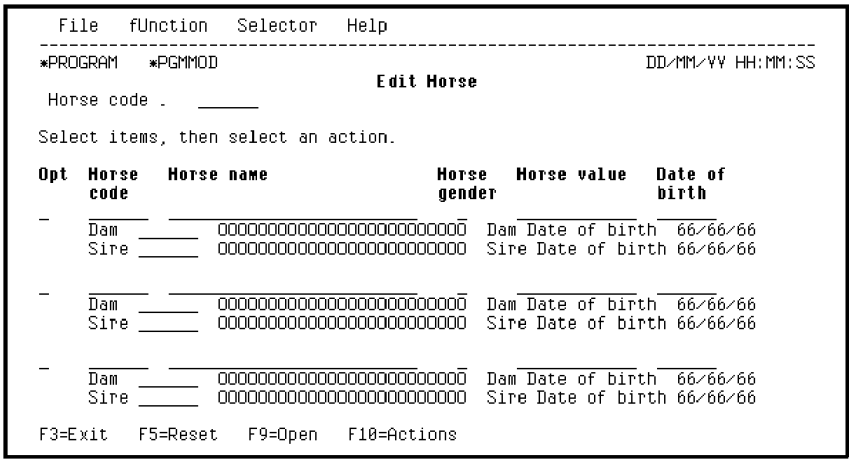
Modified Panel

Now both the labels and the entry fields are aligned.



Optional Exercise

Repeat this process to align the entry fields for Dam Date of birth and Sire Date of birth. To do so, place the cursor on Dam Date of birth, press Enter, and change the Label spacing field to 1. Press Enter to return to the device design. The result should look like this:

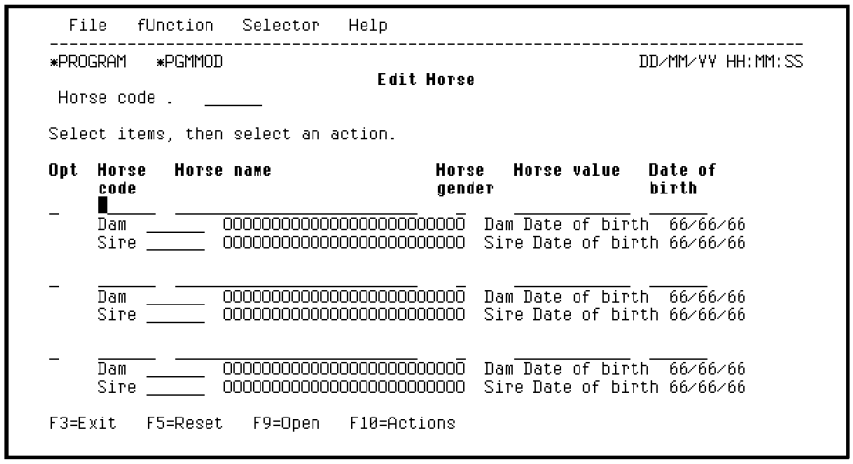


Panel Format Relations

When the Edit Horse function is implemented, the program automatically includes code to check the validity of each input capable field on the subfile record. By default, every relation must be satisfied. This will cause the resulting application to check the HORSE file when the end user specifies a Dam Code. The Dam code must be that of a valid female horse. This presents a potential problem. When the end user adds the first horse to the application, it will be rejected. This is because there are no horses existing on the file at that time that are available as parents (Dam or Sire).

To overcome this problem, you can override the default check values for the panel format relations and make the validation of particular relations optional. When an end user enters a value, it will be validated. However, the end user will not be required to enter a value. This is done using the Edit Screen Format Relations panel.

Place the cursor on any field of the first subfile record.

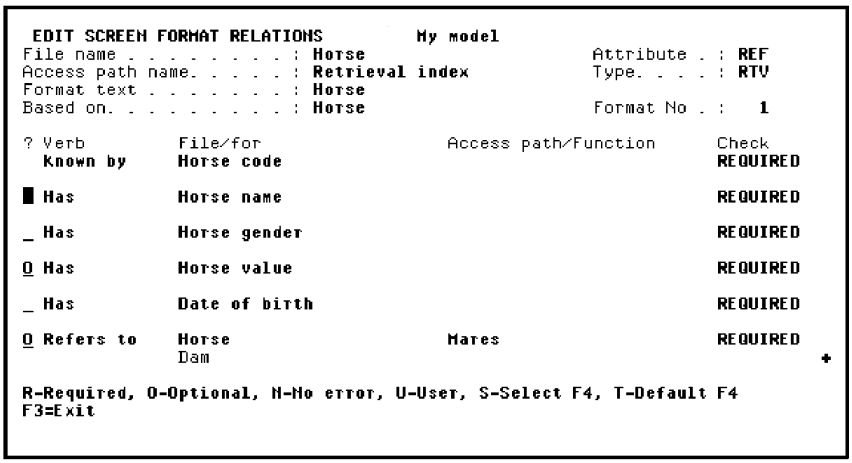


Press F7 to display the Edit Screen Format Relations panel.

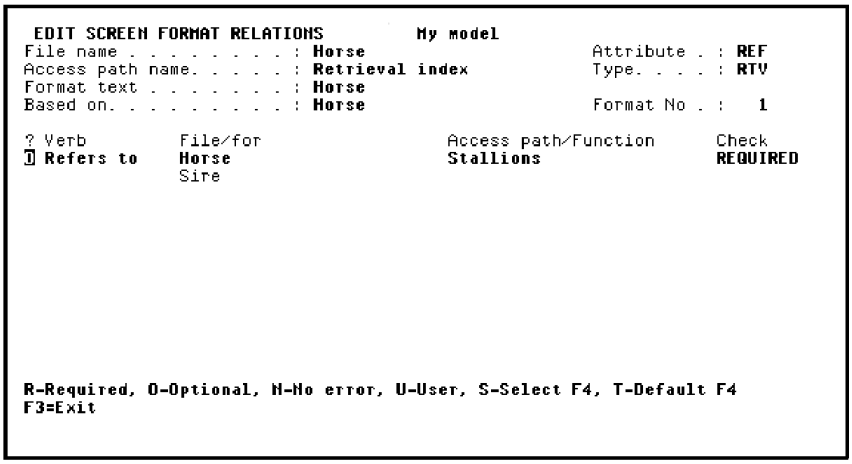
Editing Panel Format Relations

Change the validation of a relation to optional by typing **O** against the relevant relation. In this tutorial, the check condition for the two Refers to relations and the check condition for HORSE Has Horse value should be optional.

Type **O** against the first Refers to relation and the Has Horse value relation.



Press Roll Up to display the second HORSE Refers to HORSE relation and type **O** in the Subfile selector.

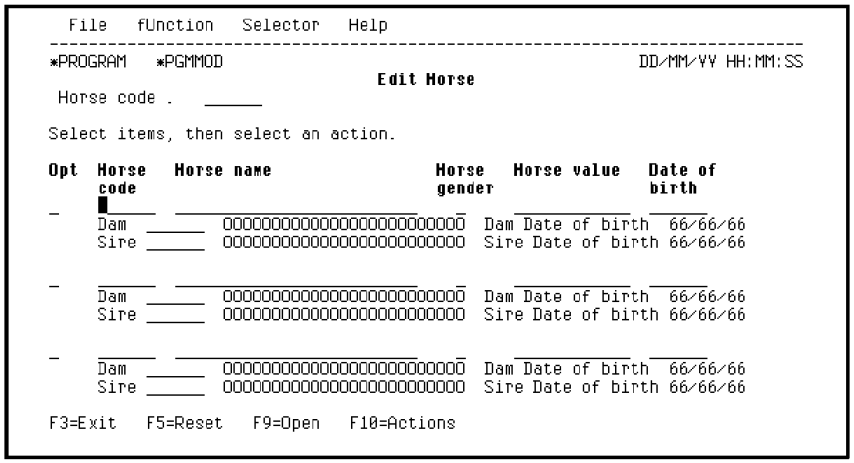


Press Enter to confirm. Note that the value in the Check column for the selected relations changed from REQUIRED to OPTIONAL.

Press F3 to return to the device design.

Completed Device Design

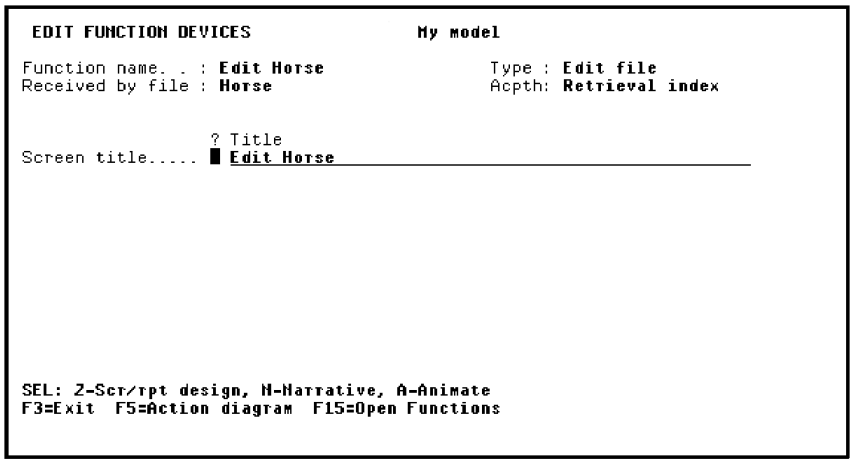
The Edit Horse device design is now complete and should look like this:



Press F3 to save and exit the device design. The Edit Function Devices panel displays.

Exiting the Device Design

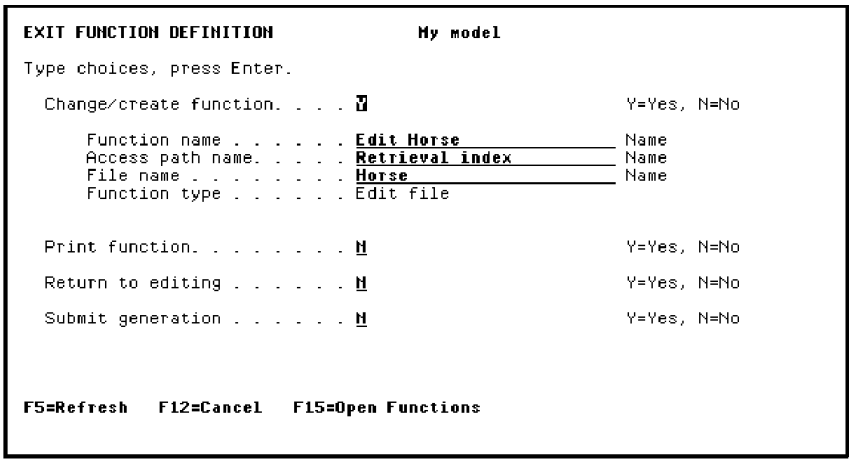
From the Edit Function Devices panel you can change the panel title, display the action diagram for the function, display all open functions (CA 2E allows you to have multiple functions open), return to the device design, or exit and save the device design.



Press F3 to exit and display the Exit Function Definition panel.

Exit Function Definition

The Exit Function Definition panel lets you optionally save any changes to the device design. You can also change the name of the function, access path, or file. From this panel you can also print the function details, return to editing the device design, or submit the function for generation.



Accept the defaults and press Enter to save the device design.

Function Confirmation

When the save process is complete, CA 2E automatically returns to the Edit Function Details panel; the message "Function ‘Edit Horse’ has been saved," appears at the bottom of the panel. Press F3 to return to the Edit Functions panel.

Window Device Design

In this step you will edit a function that has a window for its default display file. When the design standard is set to CUA TEXT, CA 2E automatically creates all Select Record functions with a window defined for the display file. Recall that you specified *CUATEXT on the Create Model Library (YCRTMDLLIB) command when you created your model at the beginning of this tutorial.

The following steps in this tutorial illustrate how to use the Device Design Editor to modify the default window designs associated with the Select Record (SELRCD) function.

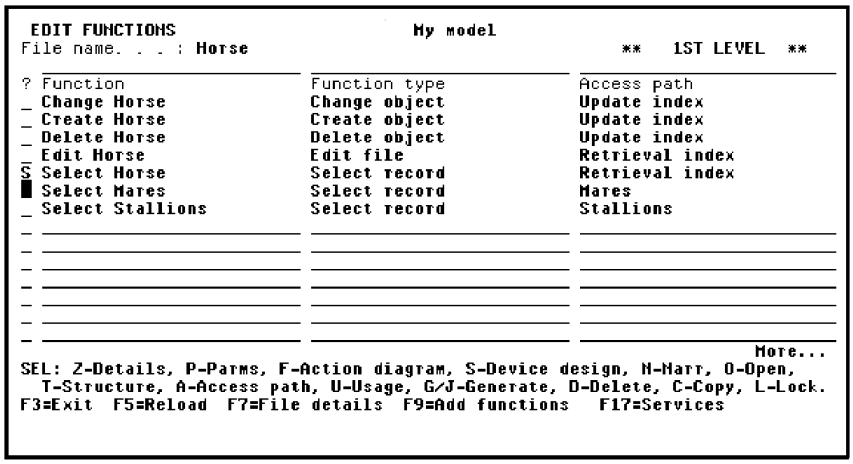
Modifying the default window design involves two basic steps. Both of these steps can be accomplished with the Device Design Editor.

1. Arrange the fields on the panel, hiding any that are unnecessary.
2. Adjust the window dimensions to surround the fields present on the device design.

Invoking the Device Design

You can access the default device design for a function directly from the Edit Functions panel. Compare this to the way you accessed the device design for the Edit Horse function in the tutorial topic you just completed. In that case you typed **Z** to display the Edit Function Details panel and then pressed F9 to access the device design.

Select the Select Horse function by typing an **S** in the ? column (Subfile selector).

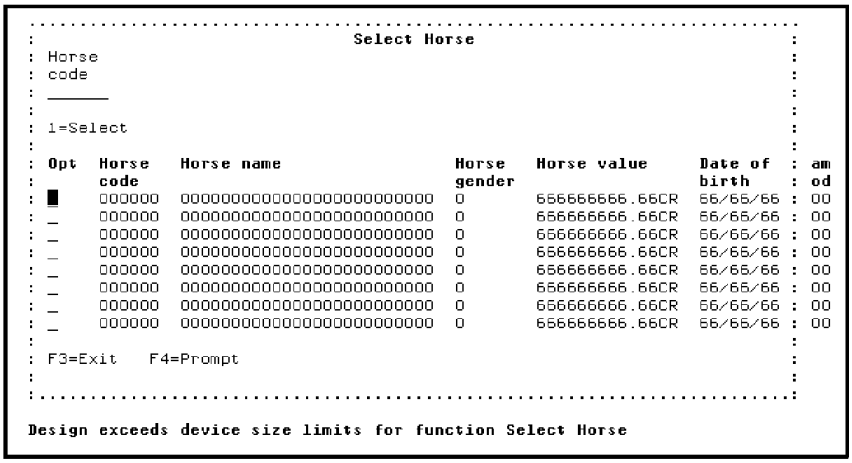


Press Enter to display the default device design.

Default Device Design for Select Horse

A default window device design displays. The subfile control format shows only the key fields and the subfile record format shows all fields on the HORSE file. You can use any of the procedures shown in the previous section to modify the device design. In addition, in this step, you will hide all of the fields in the HORSE file other than Horse code, Horse name, Horse gender, and Date of birth in the subfile record format.

Position cursor on any field in the first subfile record.

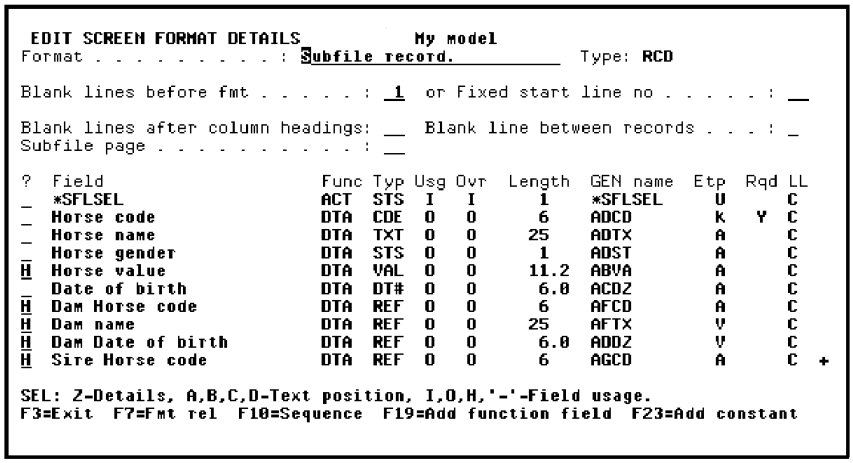


Press F5 to display a list of all available fields for the Select Horse device design window.

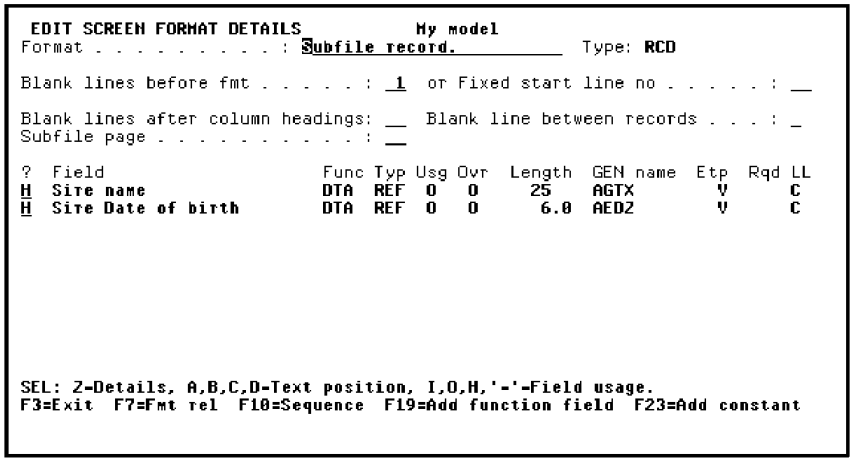
Hiding Fields in the Subfile Record Format

The Edit Screen Format Details panel contains a list of all fields associated with this function. You may not want to show all of the fields on your panel. You can hide fields you do not want displayed by typing **H** in the Subfile selector of the Edit Screen Format Details panel.

In this step you will hide the following non-key fields: Horse value, Dam Horse code, Dam name, Dam Date of birth, Sire Horse code, Sire name, and Sire Date of birth. Type **H** next to these fields.



Press Roll Up to display more fields. Type **H** next to Sire Name and Sire Date of Birth.

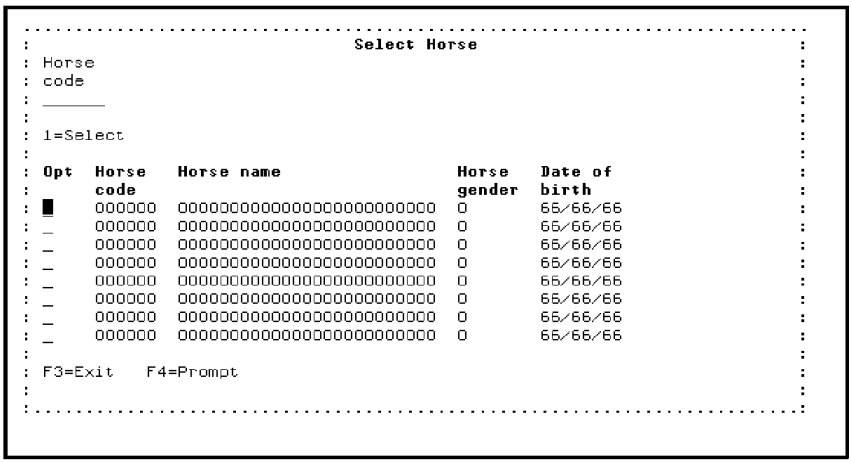


Press Enter. Note that the value of the Ovr (Override) column for the selected fields changed from O to H.

Press F3 to return to the Device Design Editor.

Modified Subfile Control Format

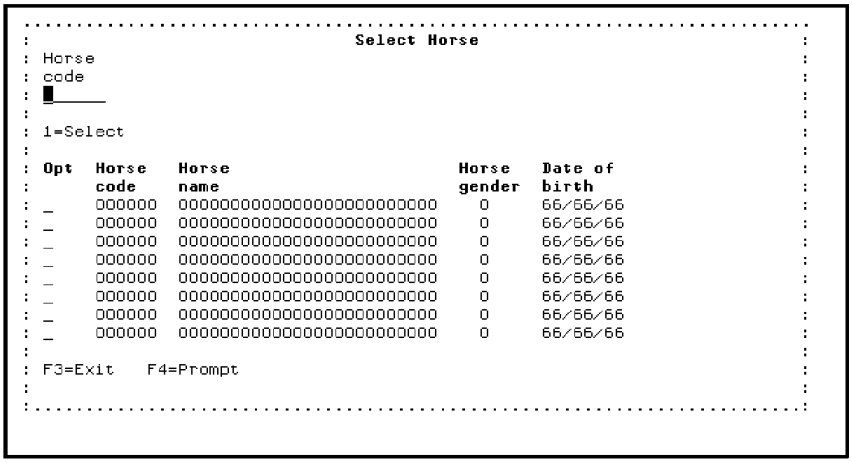
Note that the hidden fields are no longer shown on the subfile record format of the device design for the Select Horse function.



Exercise

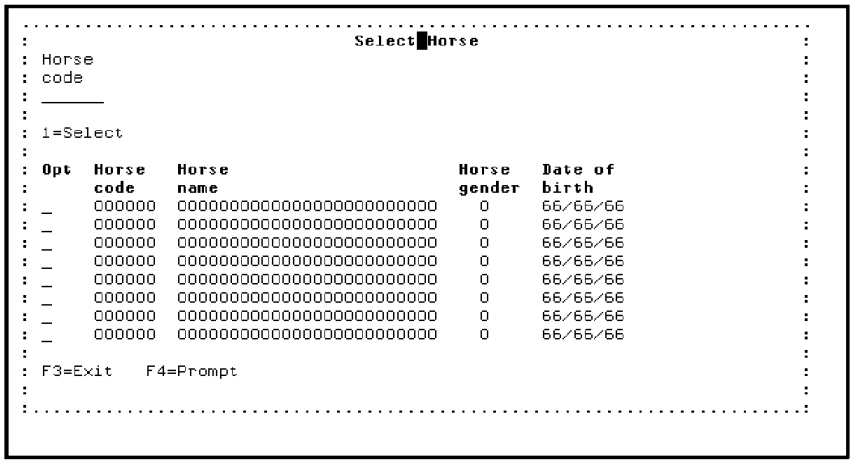
Complete the following exercise to modify the device design for the Select Horse function. Use function keys as you did for the Edit Horse function previously in this tutorial.

Modify the device design for the Select Horse function so that it reflects the design that follows.



Window Options Editor

In this step you will reduce the dimensions of the Select Horse window so that it just surrounds all remaining fields as part of the device design. You can access the Window Options Editor from the function title. Position the cursor on the function title.

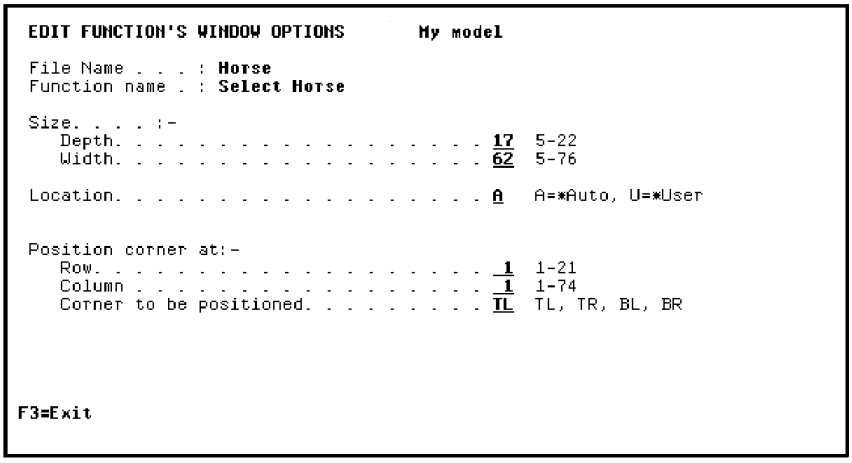


Press Enter.

The Edit Function's Window Options panel contains the parameters that control the size and location of a window on a panel. You can change the default values and value limits for the window depth, width, and position as needed. In this tutorial, you will modify the depth and width of the window.

Changing Window Dimensions

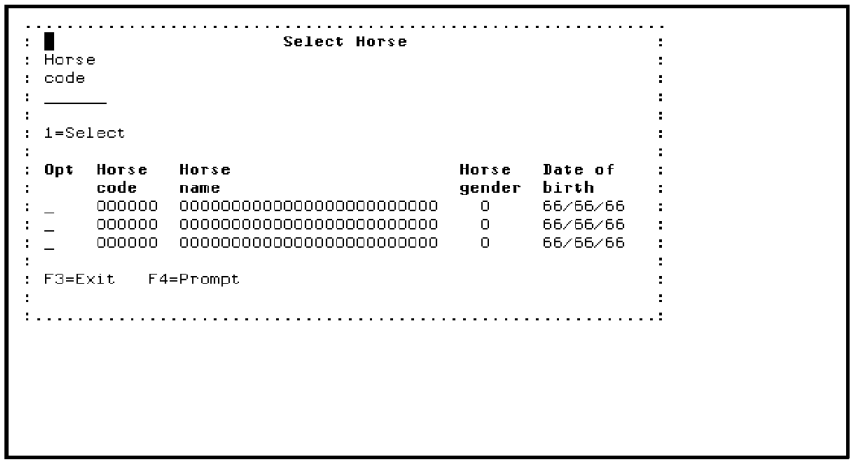
Change the width and depth of the window to include just the fields on the device design. You can feel free to experiment by trying several combinations until you achieve a panel layout that meets your needs. For this tutorial, change the window so that the depth is 17 and the width is 62.



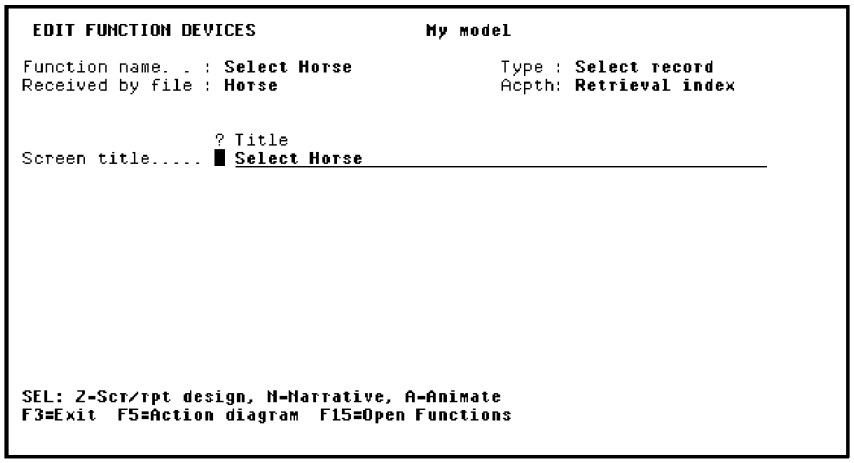
Press Enter and press F3 to return to the Device Design Editor.

Completed Device Design

When you return to the Device Design Editor, the window's borders should surround all of the fields. The device design for the Select Horse window is now complete.

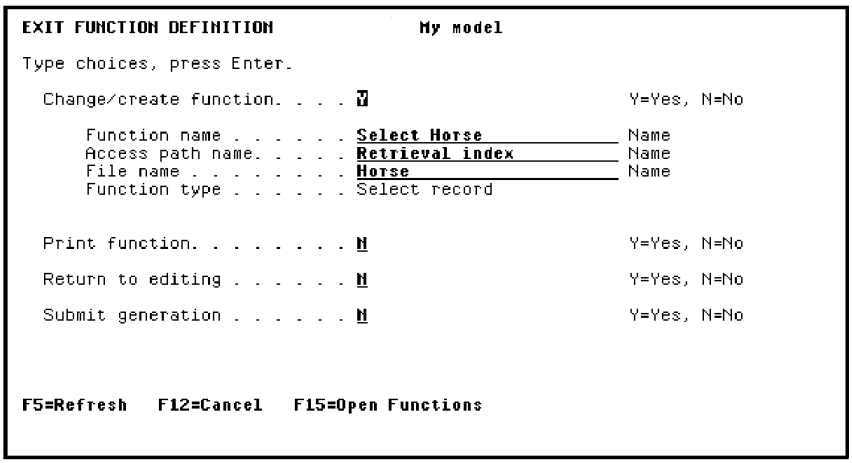


To exit the Device Design Editor, press F3.



Exiting the Device Design

Exit this function the same way as you exited from the Edit Horse function. Press F3 to return to the Exit Function Definition panel.



To save the function, accept the default of Y and press Enter. After your device design is saved, you will be returned to the Edit Function panel.

Exercises

Before continuing with the rest of the tutorial, complete the following exercises.

1. Create the default functions for the RACE and JOCKEY files as you did for the COURSE and HORSE files. You will create the default functions for RACE ENTRY later in this chapter.

   Note: Remember that CA 2E automatically creates the default functions for a file the first time you type **F** against any relation for the file on the Edit Database Relations panel.

2. Check the device design for each of the following external functions to improve its appearance and to ensure that the device design does not exceed the size of the panel: Edit Course, Select Course, Edit Race, Select Race, Edit Jockey, Select Jockey, Select Stallions, and Select Mares. Note that a device design that exceeds the size of the panel will cause errors when you generate the functions in the *Generating, Compiling, and Executing* chapter.

   Note: You access the device design for a function by typing **S** against it on the Edit Functions panel or by pressing F9 from the Edit Function Details panel. To edit the device design, use the function keys you just used to edit the device designs for the Edit Horse and Select Horse functions. Press Help or refer to the table earlier in this topic for a list of function keys.

Action Diagrams

This topic introduces action diagrams and the CA 2E Action Diagram Editor.

New terms introduced:

- CA 2E action diagram
- Action diagram action
- Sequential construct
- Iterative construct
- Conditional construct
- Hidden construct
- User point
- Context
- CA 2E message function
- CA 2E function parameter

New panels introduced:

- Edit Action Diagram
- Edit Action Condition
- Edit Action
- Edit Message Function
- Edit Function Parameters
- Edit Message Function Details

Overview of Action Diagrams

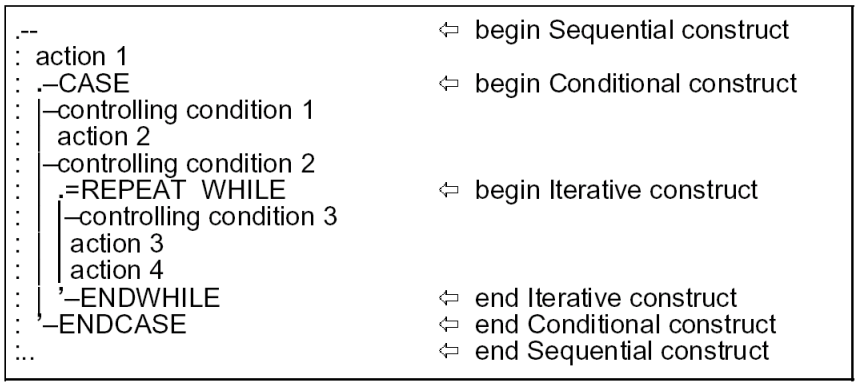
An *action diagram* contains the processing steps that make up a function. Each action diagram consists of a list of *actions*, where each action may be either a call to another function or one of a number of low level built-in functions; for example, *ADD.

The sequence in which actions are executed is controlled by three simple constructs: sequential, conditional, and iterative. The constructs specify a list of actions, and optionally, when and how to execute those actions. Constructs are the basic building blocks of an action diagram. They are always executed from top to bottom. In addition, constructs may be nested; in other words, an action within a construct may be another construct.

Following are brief descriptions of each of the three constructs and the way in which each is shown on the action diagram. Refer to the diagram following these descriptions as you read.

- **Sequential construct—**A sequential construct is the simplest construct. It is a list of actions or other constructs to be executed in the order in which they appear in the action diagram. It is shown on the action diagram enclosed by a bracket of dots (:).
- **Conditional construct—**A conditional construct specifies a condition and a series of actions to be taken if the condition is true. It is equivalent to an IF THEN ELSE logic statement or a SELECT set. This construct appears on the action diagram between CASE and ENDCASE statements and is enclosed by a bracket of broken vertical bars (|). You can specify several mutually exclusive conditions in a single conditional construct.
- **Iterative construct**—An iterative construct includes a list of actions that are to be executed while an initial condition is true. It is equivalent to a DO WHILE logic statement. An iterative construct appears on the action diagram between REPEAT WHILE and ENDWHILE statements. It is indented and enclosed by a bracket of solid vertical bars (|). The controlling condition is specified at the beginning of the bracket.

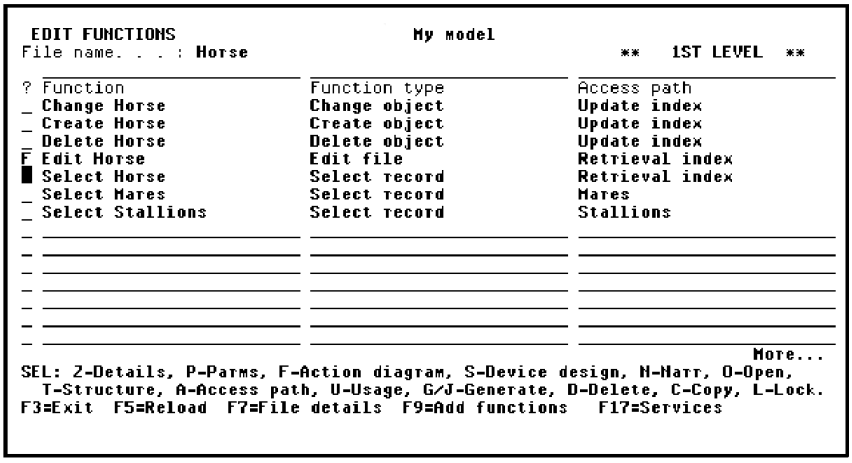
The following diagram shows the general structure of a CA 2E action diagram and the three constructs.



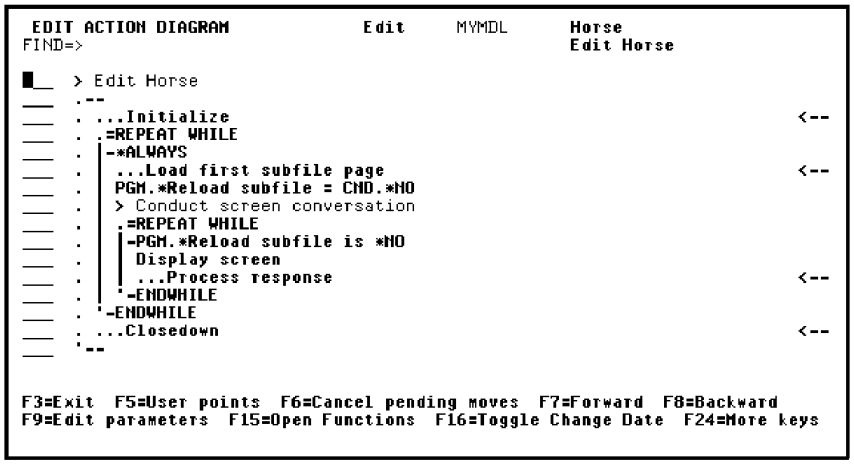
Default Action Diagram

CA 2E supplies a default action diagram for each function. In this step you will view and edit the default action diagram for the Edit Horse function.

From the Edit Function panel type **F** in the Subfile selector for the Edit Horse function.



Press Enter to view the action diagram for the Edit Horse function.



The Action Diagram Editor's Subfile selector provides options for editing an action diagram, including options for inserting and manipulating user-defined actions. This topic demonstrates many of these options. You can type **?** in the Subfile selector to view a list of all allowed values.

Hidden Constructs and User Points

The action diagram is initially shown at a summary level, with many *hidden constructs*. This allows you to see the entire action diagram in outline form on a single panel. The names of hidden constructs are preceded by three dots (for example, ...Initialize). The name of a complete construct is preceded by an angle bracket (for example, >Edit Horse).

You can alter or add to the default processing by editing the action diagram. Only certain parts of the action diagram may be altered. The areas that you can modify are called *user points* and are indicated by an angle bracket (<—) in the right margin of the panel. The rest of the default action diagram is protected to ensure that you do not accidentally change essential function processing.

You can display a selection list of all the user points for a function by pressing F5. You will use this feature later in this tutorial.

Within a user point, each user-defined action is indicated by three chevrons (<<<) in the right margin of the panel. In addition, if you press F5 to view the list of all user points, user points containing user-defined actions are indicated by three chevrons in the right margin.

Edit Horse Action Diagram

Apart from initialization, the action diagram for the Edit Horse function consists essentially of an iteration of two steps. The first step displays a panel to the user. The second processes the response that is entered. We will examine the second step in more detail. It includes validation and update processing.

The basic validation the default Edit Horse function performs is derived from the definition of the HORSE file relations, fields, and conditions you specified in the data model. In this tutorial, you will add extra validation routines to check that a horse is younger than its parents. To do this, you will need to find the relevant part of the action diagram to edit.

Displaying Hidden Constructs in a Action Diagram

You can view the contents of a hidden construct by typing **S** (show) against any line indicated by three dots (...) and pressing Enter. This expands the selected hidden construct. Note that when you expand a hidden construct the three dots change to **>** to indicate that the entire construct is displayed. You can hide the construct again with a Subfile selector value of **H** (hide).

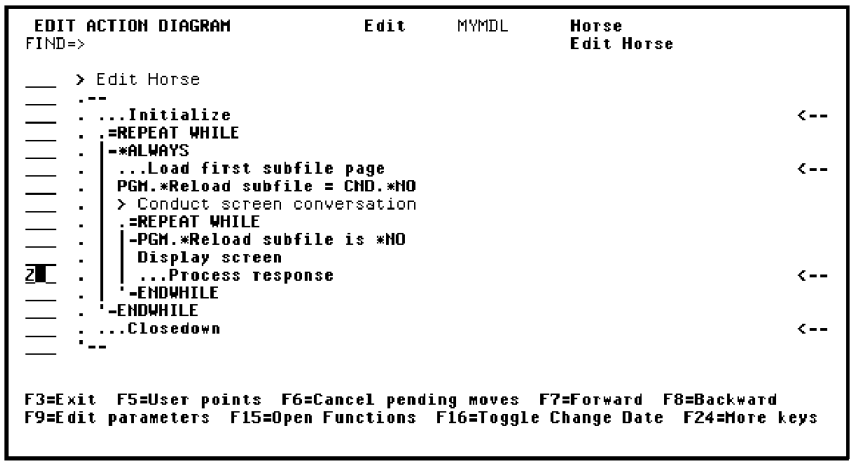
When hidden constructs are displayed, the whole action diagram occupies more than one panel. To simplify editing an action diagram, you can display hidden constructs individually by typing **Z** (zoom) in the Subfile selector. You will use this in the next step.

As you will see in the following steps you can zoom into hidden constructs at continually deeper levels of the action diagram. At any point you can reverse the effect of the last **Z** you typed by typing **U** (unzoom) in the Subfile selector. To return to the topmost level, type **T** (top) in the Subfile selector.

Adding Extra Validation to Edit Horse

To add extra validation conditions to the Edit Horse function, you must access the part of the action diagram that processes the data keyed into the displayed subfile page. This is the hidden ...Process response construct. Note the arrow in the right margin (<—). This indicates that this construct contains one or more user points where you can add user-defined validation.

To display this construct, type **Z** against ...Process response.

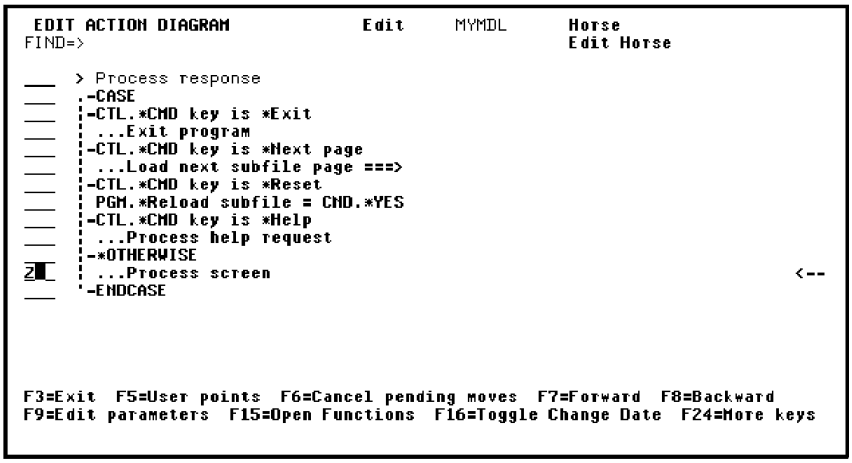


Press Enter.

The Process Response Construct

The Process response construct is now displayed. It is made up of four mutually exclusive actions based on which function key was pressed. For example, the Exit key exits the program, the Next page key loads another subfile page, and the Reset key reloads the subfile page to process the input data. In this case, you will zoom into the hidden ...Process screen construct.

Type **Z** against ...Process screen.

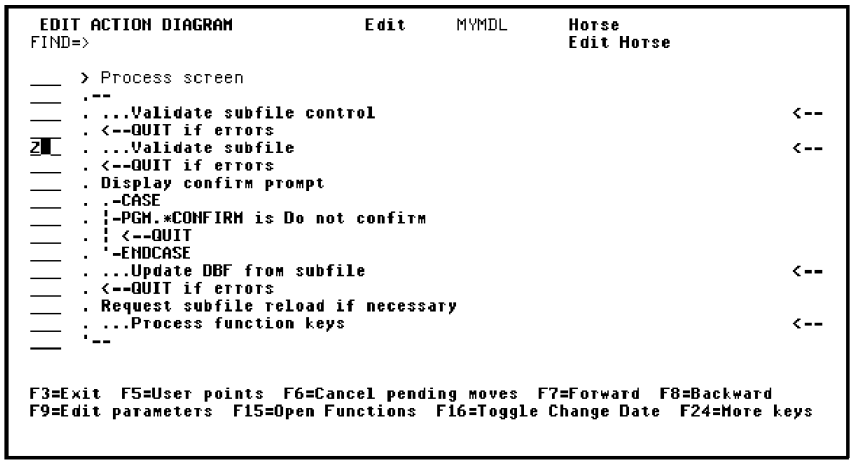


Press Enter.

The Process Screen Construct

The device design for the Edit Horse function is made up of two parts: a subfile control format and a repeating subfile record format. Default processing for this function involves validation of the subfile control and subfile record before updating the database file.

Zoom into the ...Validate subfile construct by typing **Z** against it.



Press Enter.

The Validate Subfile Record Construct

Note the hidden ...Validate subfile record construct that results when you expanded the ...Validate subfile construct. Validation of subfile record input is carried out on any subfile records that have been changed by the end user. Subfile record fields and relations are automatically validated according to the specifications declared in the design process. This is summarized as follows:

- Fields must satisfy their data types; for example,

  Date of birth must be a valid date.

  Horse value must be numeric. Note that zero is valid because it is an optional relation.

  Horse name must be present and lower case is valid.

- Fields must satisfy any check conditions; for example,

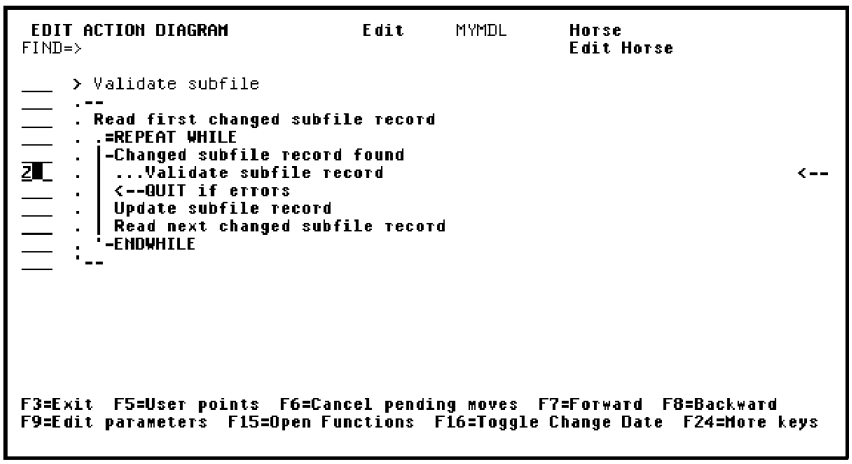
  Horse gender must be male (M) or female (F).

- Any file-to-file relationships must be satisfied; for example,

  Dam code, if specified, must be the code of a valid horse. In addition, the horse must be female because the HORSE Refers to HORSE For Dam relation uses the Mares access path that selects female horses only.

  Sire code, if specified, must be the code of a valid horse. The horse must be male because the HORSE Refers to HORSE For Sire relation uses the Stallions access path that selects male horses only.

Type **Z** against ...Validate subfile record.

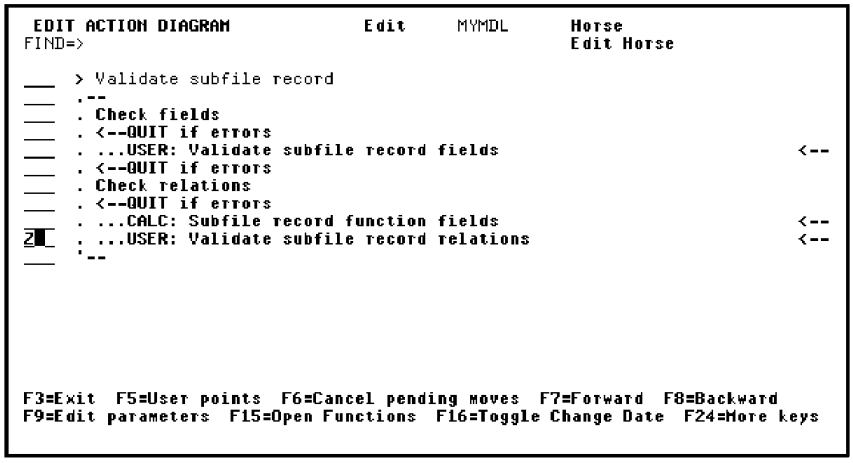


Press Enter.

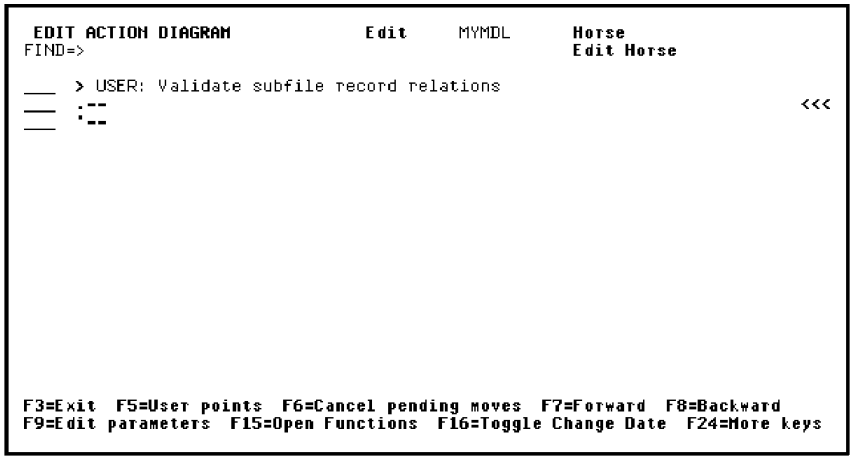
Adding a Validation Procedure

In this step you will use the action diagram editor to add your own validation procedure. Add the procedure to the USER: Validate subfile record relations user point.

Type **Z** against ...USER: Validate subfile record relations.



Press Enter.



Subfile Record Relations

The three chevrons (**<<<**) in the right margin of the panel indicate that you may add your own processing at this point.

Editing the Action Diagram

This topic describes some of the Subfile selector values available for editing an action diagram. You can type **?** in the Subfile selector to view a selection list of all values. Note that these Subfile selector values are sometimes referred to as *commands*.

Most of the Subfile selector values shown here will be demonstrated in this and the following topics. To learn about additional features of the Action Diagram Editor, refer to the online Help or the CA 2E guide, *Building Applications*.

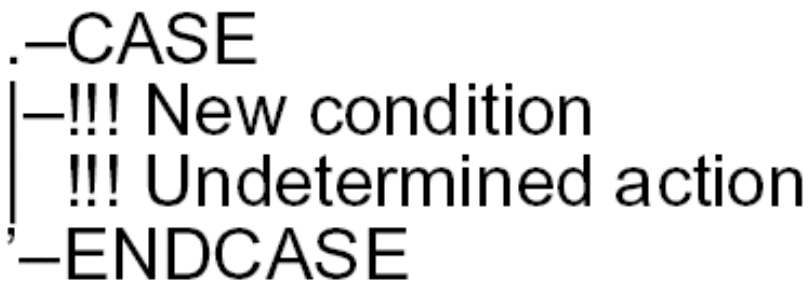
Showing and hiding hidden constructs—These Subfile selector values were covered previously in this topic; this is just a review. To display a hidden construct individually use **Z** (zoom). To reverse the effect of the last **Z** you typed, use **U** (unzoom); to return to the topmost level, use **T** (top).

To hide a low level construct, use **H** (hide). The value **H** is generally used to simplify an action diagram display so it is easier to follow. To view the contents of a hidden construct, use the value **S** (show).

Inserting new constructs—Use the following Subfile selector values to add new constructs to an action diagram:

Subfile Selector Value	Inserts
IA	An action
IS	A sequence of actions (Sequence construct)
II	An iteration loop (Iterative construct)
IC	A case condition (Case construct)
IX	A new condition within a Case construct

The new construct starts on the line following the one on which the insertion was specified. Inserted constructs are initially defined in general terms. For example:



This is a generic definition of a conditional construct; where, !!! New condition indicates a conditional statement and !!! Undertermined action indicates the action to be performed if the condition is true. You can use these general structures to check the program logic. After you are sure the logic is correct, you can specify the details of the conditions and actions.

Fast paths for inserting actions—The following Subfile selector values provide fast paths that you can use to enter various actions.

Subfile Selector Value	Inserts
I*	A comment
I+	An *ADD built-in function
I-	A *SUB built-in function
I=	A *MOVE built-in function
IM	A Message function

Entering construct details—You can take an additional fast path to defining the details for any of the fast path constructs/actions by appending **F** to the Subfile selector value (for example, **IAF**). Including the **F** causes CA 2E to display the appropriate panel where you enter details for the construct or action.

Moving and copying constructs—To move or copy a construct, type **M** or **C** in the Subfile selector of the selected construct. Type **A** (after) on the line above the location in the action diagram to which you want to move or copy the construct. The construct will be placed After this line. Alternatively, you can type **B** (before) on the line below the target location. The construct will be placed Before this line.

You can also move or copy a block of constructs using the **MM** (move block) or **CC** (copy block) commands. These commands are used in pairs; for example, type **MM** on the first line in the block to be moved and type another **MM** on the last line of the block to be moved. The entire block must be at the same construct level as indicated by the action diagram indentations. Use **A** (after) or **B** (before) as for the **M** and **C** commands.

Note: CA 2E provides a Notepad utility that lets you copy constructs from one action diagram to another. The Notepad is a temporary action diagram that serves as a work area where you can copy and append constructs from the current action diagram. Press F18 to toggle between the action diagram and the Notepad.

Deleting constructs—To delete a construct, type **D** against the first line of the construct. You can also use a pair of **DD'**s to delete a block of constructs at the same construct level.

Editing the Edit Horse Action Diagram

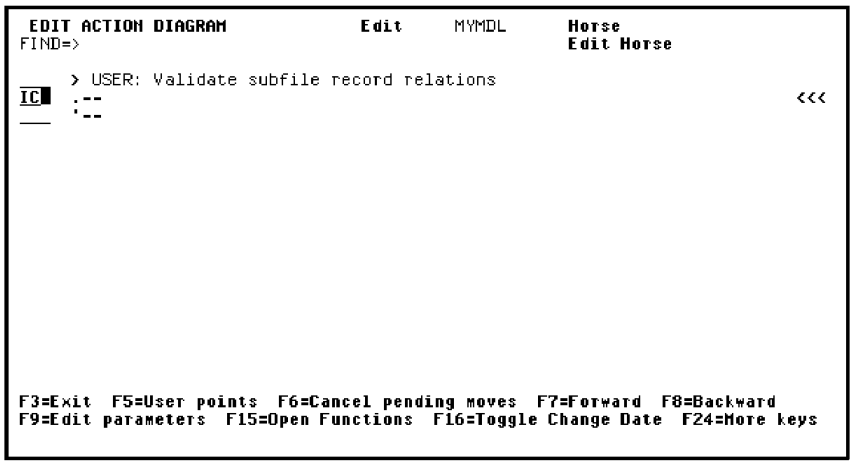
In this tutorial, you will change the action diagram for the Edit Horse function as follows. Your objective is to insert logic to validate that a horse's Date of birth is later than that of both its Dam and Sire; otherwise, display an error message.

To do this you will add:

- Two case conditions
- Two actions

Inserting a Condition

Insert the first condition by typing **IC** on the line showing the three chevrons (<<<) in the right margin.

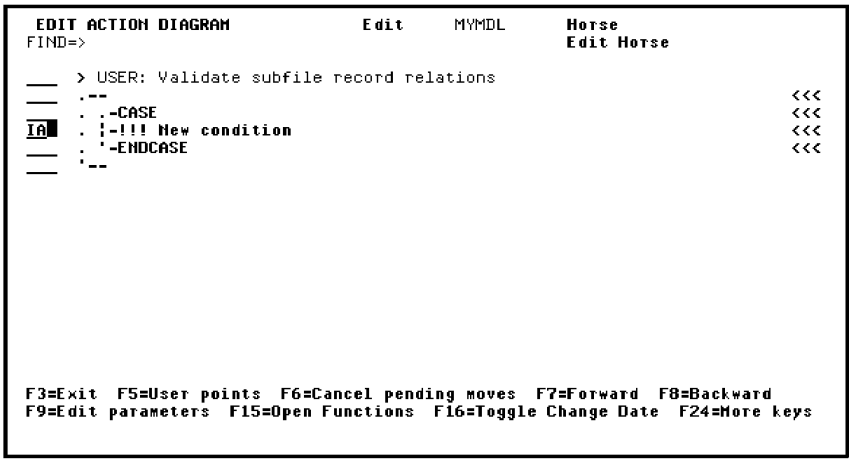


Press Enter.

Inserting a Single Action

An empty CASE construct has been inserted in the action diagram. The CASE construct by itself only specifies the condition and not the action that is to be taken if the condition is satisfied. You will now add this action.

In this case, you only require a single action. As a result, type **IA** (Insert action).



Press Enter to insert the new action.

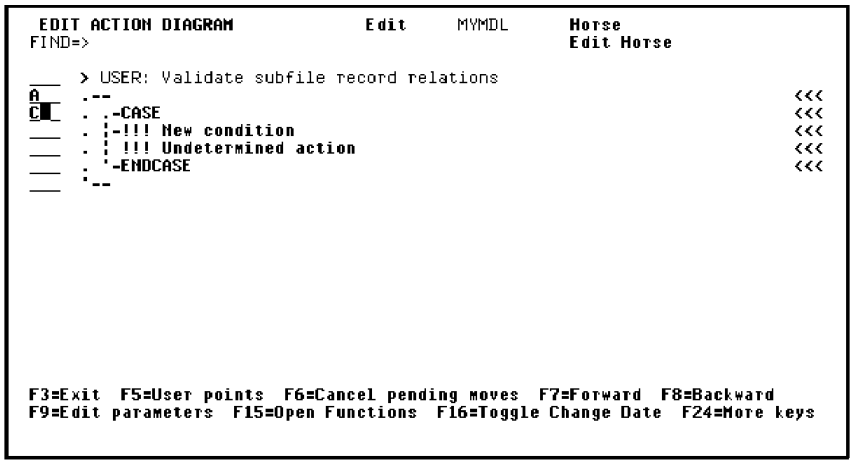
Note: If you wanted more than one action to result from the condition, you could add more actions using the **IA** option repeatedly or by inserting a sequence construct using the **IS** option.

The logic of the action diagram for the first validation is now complete.

Adding Another Condition by Copying

In this step you will add a second condition to validate a horse's date of birth with respect to that of its Sire. You can use the same process you used in the previous step or you can copy the first construct. In this case you will copy the first construct.

Type **A** on the line above the location where you want the construct copied; type **C** next to the first line of the construct you want to copy.

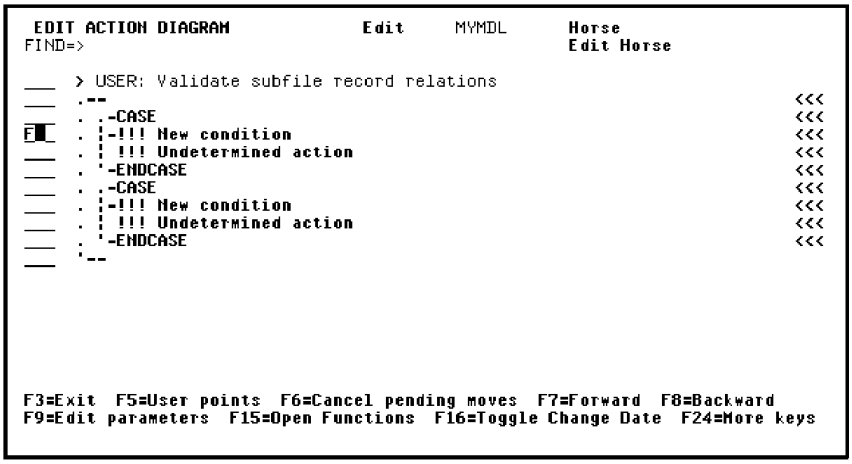


Press Enter.

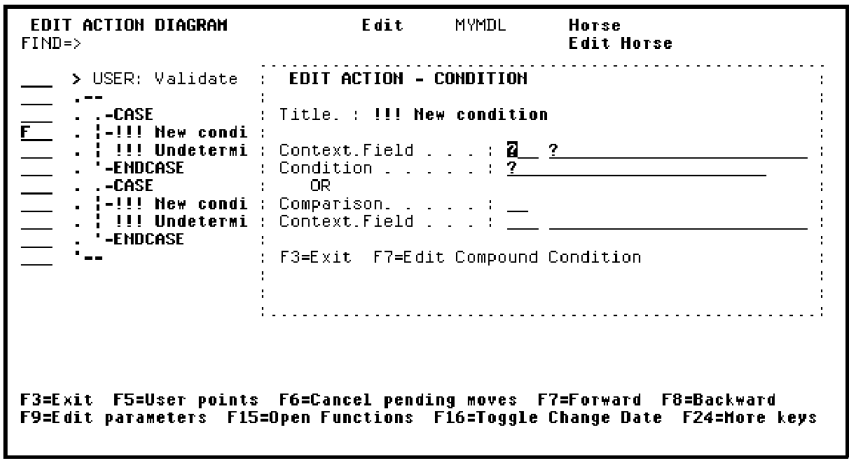
Specifying Details of Conditions

Now that both case conditions have been added, you are ready to specify the details of the conditions and actions. In this step you will specify the details of the condition: Is the dam's date of birth greater than or equal to that of the horse. You will specify the action to send an error message later.

First you will specify the controlling condition for the first !!! Undetermined action. Type **F** against the first condition as shown to edit that line of the action diagram.



Press Enter to display the Edit Action - Condition window.



The Edit Action Condition Window

The Edit Action - Condition window lets you enter the terms of the condition that you require. Type the name of the field to be evaluated and either the condition it must satisfy or the name of another field to compare it to.

If you type **?** in any field on this panel, other than the Comparison field, CA 2E displays a list of possible values.

Contexts

Within a function there are several possible sources for a field's value; for example, a field may be found on the panel display, in a database file, or as an entry parameter. A CA 2E *context* specifies which source of a particular field to use. Different contexts are available at different points in a function's action diagram.

A context is identified by a three-character mnemonic; for example, CTL - subfile control, RCD - subfile record, CON - constant data, and PAR - entry parameter. This mnemonic is prefixed to each field used in an action diagram to show the source of the field. For example, RCD.Date of birth means that the value of the Date of birth field displayed in the subfile record is to be used in the action diagram.

When you enter **?** to view a list of possible contexts, the most sensible context appears at the top of the selection list.

Adding Details of the First Condition

In this step, you will test that a horse is younger than its dam; in other words, you will test whether the Dam Date of birth is greater than or equal to the horse's Date of birth. Since dates of field type DT# are stored as an alphanumeric, YYYY-MM-DD, you can do a direct comparison of these dates.

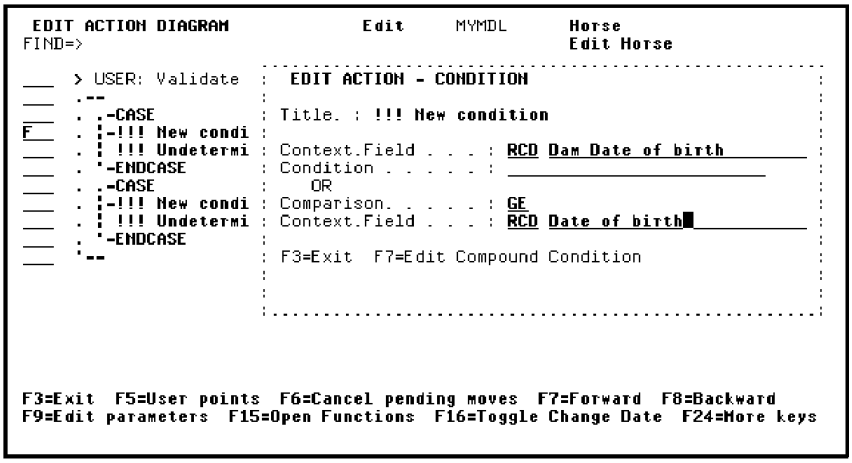
To express the condition, you will use the GE (greater than or equal) comparison operator. The basic question that needs to be asked is:

Is Dam Date of birth GE Date of birth?

If this condition is true, display an error message; if the condition is false, continue to the next action.

Note that when you add this condition to the action diagram for Edit Horse, the context for both fields in the comparison will be RCD since they are both from the subfile record.

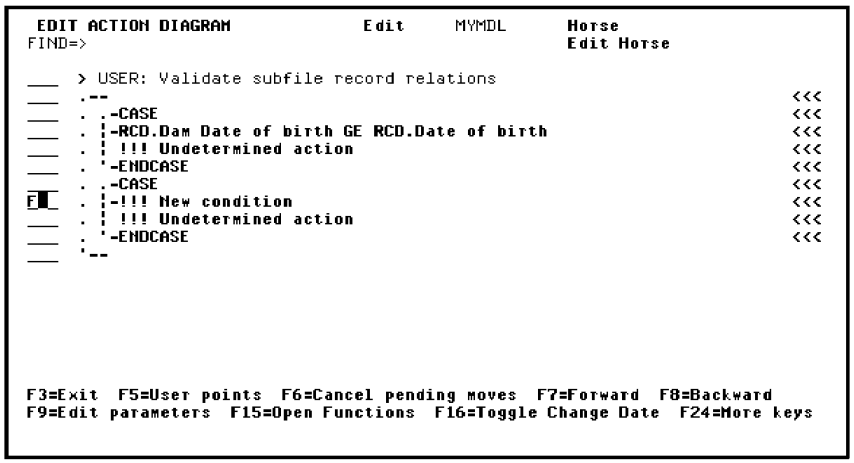
Type the details of the comparison as shown; namely, type **RCD** and **Dam Date of birth** for the first context and field of the condition, blank the Condition field, and type **RCD** and **Date of birth** for the second context and field.



Press Enter to return to the action diagram.

Adding the Second Condition

Notice that the test for the condition has been updated. You are now ready to edit the second condition. To edit the second condition, type **F** against !!! New condition.

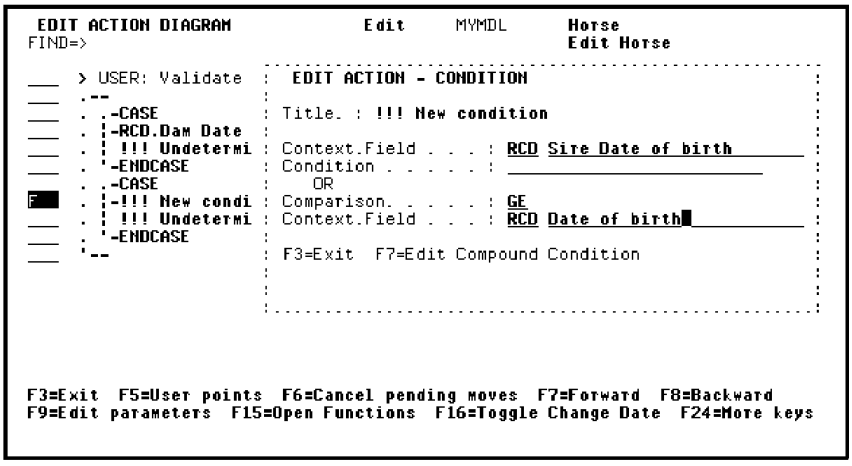


Press Enter.

Adding Details of the Second Condition

The second condition is similar to the first. In this case, you need to compare Sire Date of birth to Date of birth. Again, both fields have the RCD context since they are from the subfile record.

Type the comparison details as shown; namely, type **RCD** and **Sire Date of birth** for the first context and field of the condition, blank the Condition field, and type **RCD** and **Date of birth** for the second context and field.



Press Enter.

Adding Actions

In the following steps you will add two actions to display an error message at the bottom of the panel when either of the two conditions you just defined is true. This requires that you:

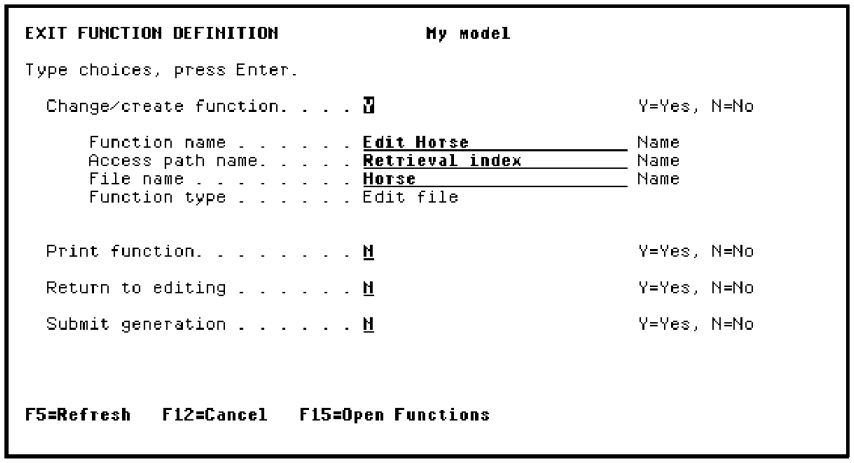
1. Define two message functions
2. Define parameters for the message functions
3. Change the text of the error messages to include the dates that caused the error using substitution variables
4. Select the message functions as the actions within the action diagram

Exiting the Action Diagram

There are two ways to define a message function; you will use both in this tutorial. The first requires that you exit the action diagram and return to the Edit Database Relations panel.

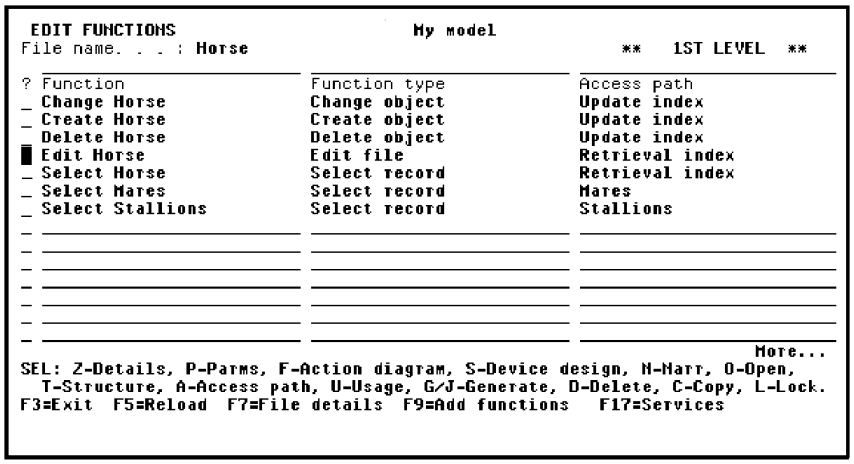
Press F13 to Exit the Action Diagram Editor and display the Exit Function Definition panel. Accept the defaults on this panel to save all changes you made to the action diagram.

Note: The F13 function key (Fast exit) provides a quick way to exit the action diagram. You can use F3, but it will take you back through the action diagram one level at a time.

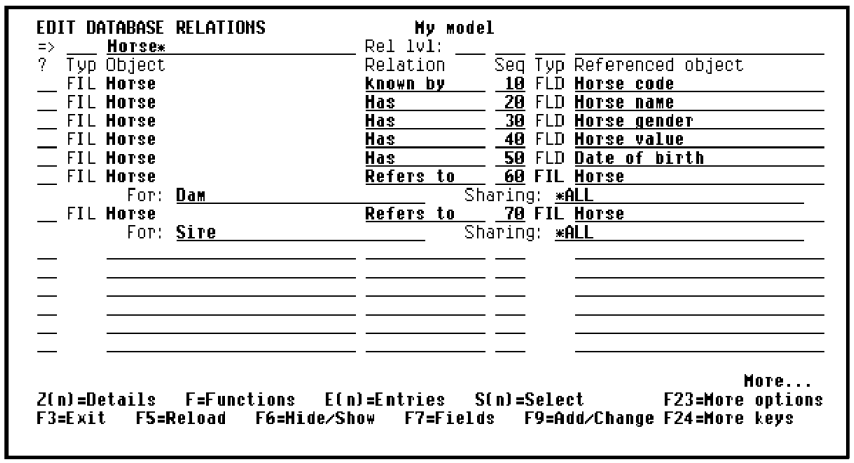


Press Enter to return to the Edit Functions panel.

The message "Function ‘Edit Horse has been saved" displays at the bottom of the panel.

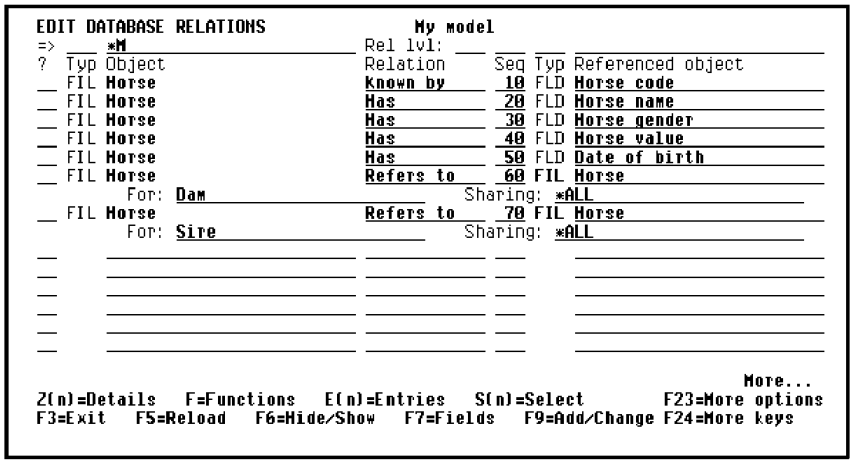


Press F3 to return to the Edit Database Relations panel.



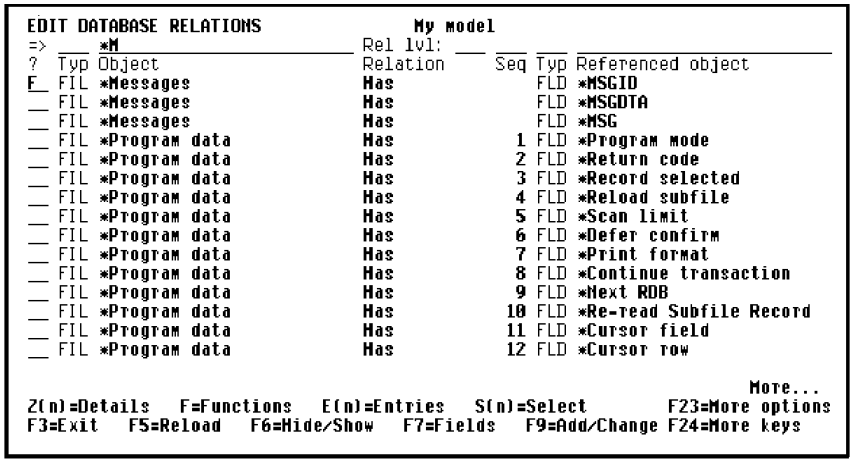
Message Functions

All message functions for CA 2E programs are stored in a system file called *Messages. Begin by positioning the Edit Database Relations panel to this file. To do so, type ***M** on the selection line.



Press Enter.

Type **F** against one of the relations for the *Messages file to display the existing message functions attached to this file.

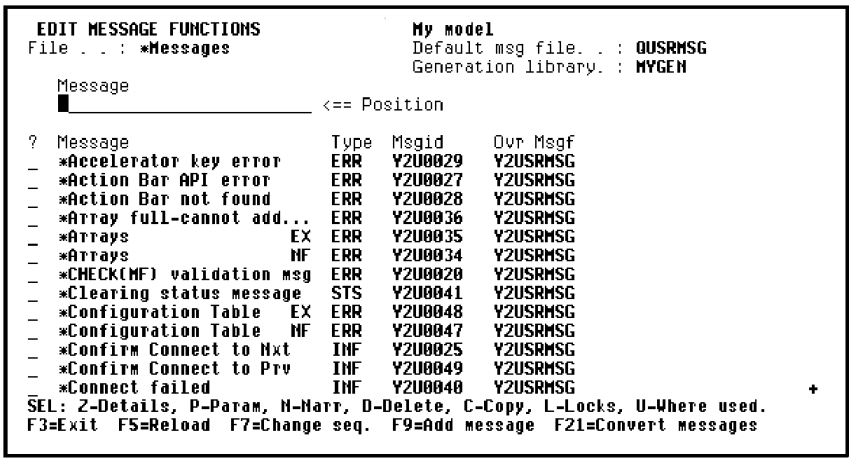


Press Enter.

Displaying the Message Functions

The Edit Message Functions panel displays a list of the message functions available on the file *Messages. A number of general purpose messages are supplied with CA 2E. The types of messages available include, ERR (Error message), INF (Information message), EXC (Execution message), and STS (Status message). You can scroll through the messages by pressing Roll Up and Roll Down.

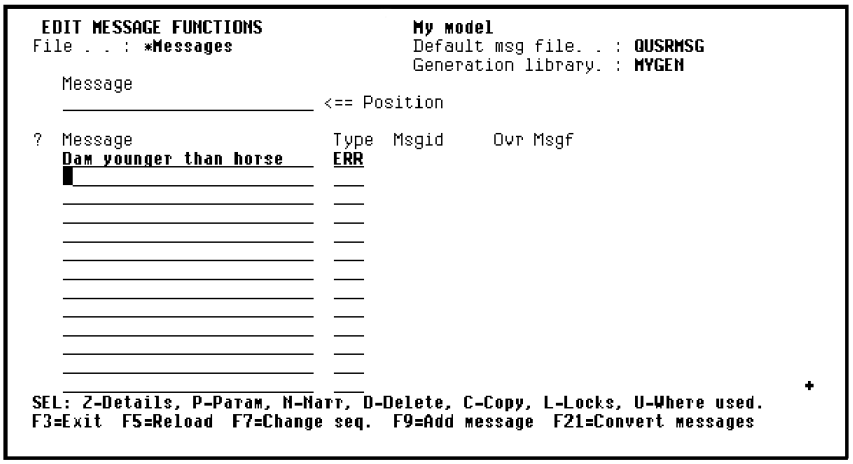
You can select one of the messages displayed or you can create your own. To send an explicit message about a horse, you will need to create a new message.



Adding a New Message Function

Press F9 to add a new message function that will be executed if Dam Date of birth is greater than or equal to Date of birth.

Name the message function Dam younger than horse. This is to be an error message so the type will be ERR. Type the message name.



Press Enter. The new message function is added to the *Messages file.

Note: You could also add the message function to be executed when Sire Date of birth is greater than or equal to Date of birth now. Instead you will use another method for creating message functions to add the second message function later in this tutorial.

Defining Parameters

Parameters define the values to be communicated between functions at execution time. How a parameter is used in the called function will depend on the type of the called function, the processing specified for that function, or both.

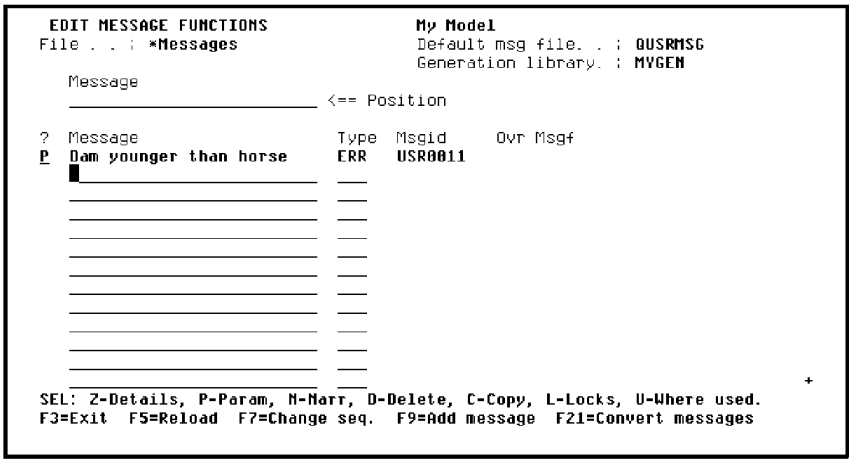
The next step is to define the parameters that will be passed to the message function by the calling program, Edit Horse. The objective is to include the dates that caused the error in the text of the error message.

Specifying a field as a parameter to a message does two things.

First, it makes the field available for use as a substitution variable in the message. In this case, the error message text will include the current values for Dam Date of birth and Date of birth.

Second, for input-capable fields, the field will be highlighted in reverse image on the panel if the error occurs. In this case, the Date of birth field would be highlighted.

Type **P** against the message function as shown to define its parameters.



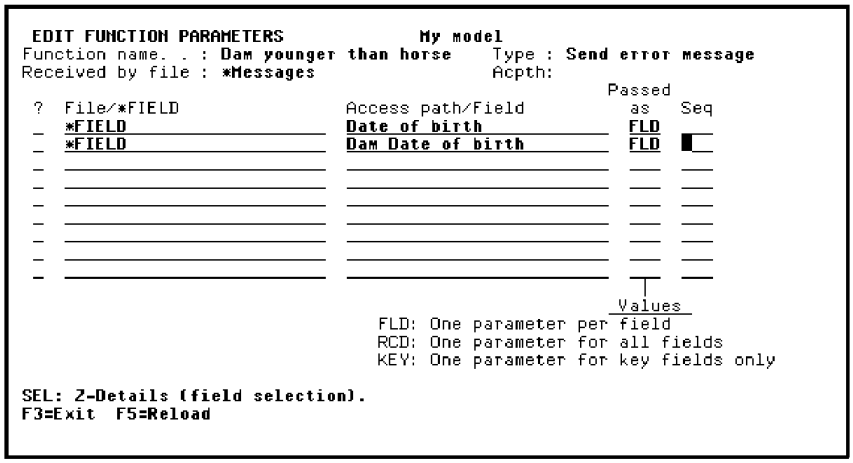
Press Enter to display the Edit Function Parameters panel.

Specifying Parameters for the First Message Function

You are now ready to specify the message parameters. There are two ways of specifying function parameters. The simplest is to name the individual fields that are to be passed as parameters (*FIELD). In this case, you will define two fields as parameters: Date of birth and Dam Date of birth.

Note: The fields on this panel will be discussed in more detail in the Function Parameters topic later in this chapter.

Type the message function parameters.



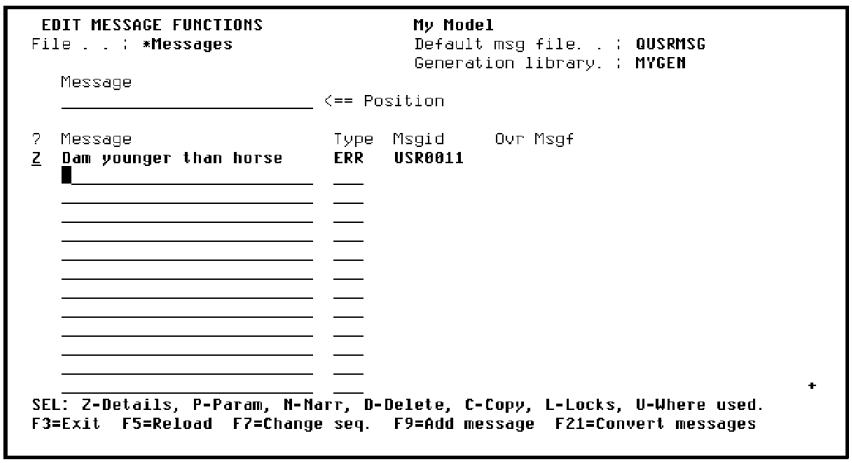
Press Enter. Your selection of input function parameters is now confirmed.

Note: You can view the details of parameter usage by typing **Z** against a parameter and pressing Enter. By default, the parameter usage will be input. If you display the details, be sure to press F3 to return to the Edit Function Parameters panel before you continue with the tutorial.

Press F3 to return to the Edit Message Functions panel.

Message Function Details

Once the parameters have been defined, you can display the message function details. Type **Z** next to the message function.



Press Enter to display the Edit Message Function Details panel. This panel displays details for the function, including the parameters you just defined. At this point you may change the message text, which has been defaulted to the name of the message function.

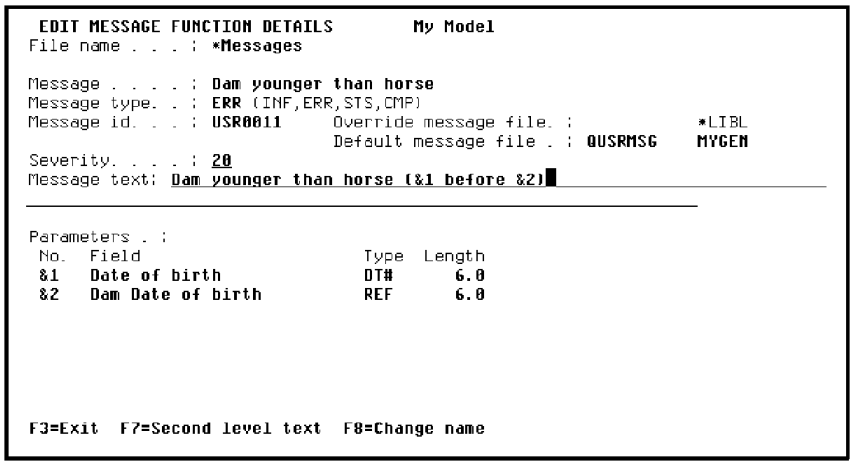
Adding Text to the First Message

You will use the two parameters you just defined as substitution variables within the message text. Substitution variables are indicated by the symbol "&n", where ‘n’ is the number of the parameter. Note the numbers assigned to the two parameters in the No. column.

Add the text "(&1 before &2)" to indicate that the date of birth of the horse is prior or equal to the date of birth of the dam.

When the message appears on the panel, the value contained in that field will be substituted in the message text. For example, the resulting message might read: "Dam younger than horse (10185 before 11294)." Substitution variables help the end user understand which fields are in error.

Type the new text containing the substitution variables, "(&1 before &2)", after the default message text.

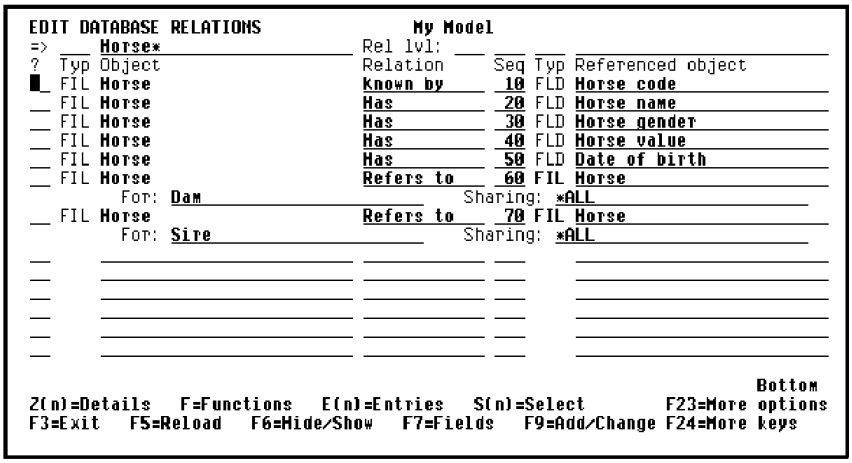


Press Enter to return to the Edit Message Functions panel.

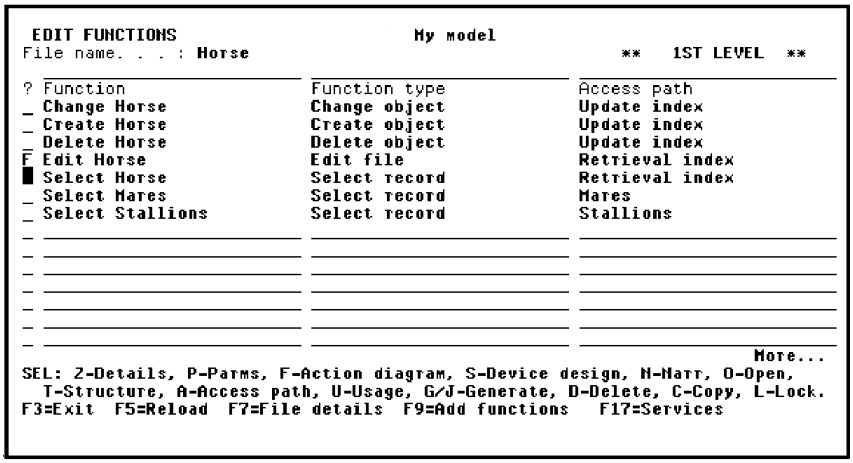
Returning to the Edit Horse Action Diagram

In this step you will insert the action you just defined into the Edit Horse action diagram and define another message function as the action when the horse's date of birth is earlier than its sire's date of birth. This time you will define the message function from within the action diagram.

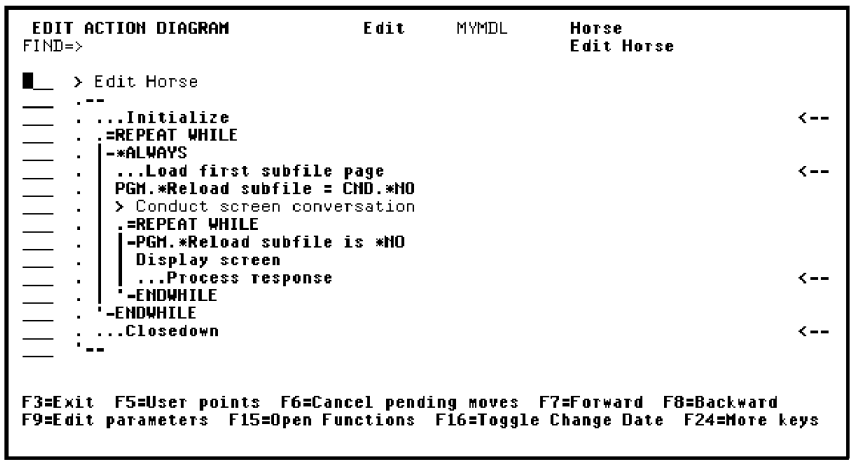
Press F3 to return to the Edit Database Relations panel. When the Edit Database Relations panel displays, type **Horse*** on the selection line and press Enter to display relations for the HORSE file.



Type **F** against one of the relations and press Enter to display functions for the HORSE file. Type **F** against the Edit Horse function.



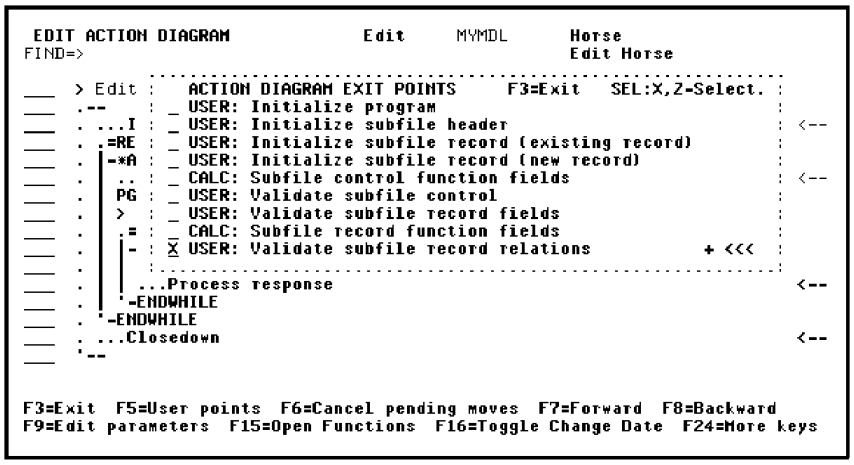
Press Enter to display the Edit Horse action diagram.



Displaying User Points for Edit Horse

Recall that you added the two conditions and actions at the user point USER: Validate subfile record relations. To return to that user point quickly, press F5 to display a list of all user points for the Edit Horse function. If you wish, you can press Roll Up to view the additional user points.

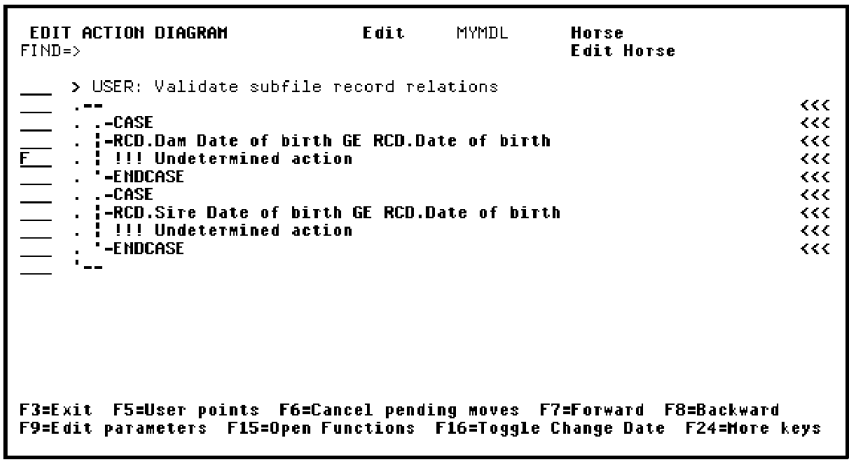
Type **X** (or **Z**) against USER: Validate subfile record relations.



Press Enter.

Add Action for First Condition

Type **F** against the first !!! Undetermined action as shown.



Press Enter to display the Edit Action - Function Name window.

The Edit Action Function Name Panel

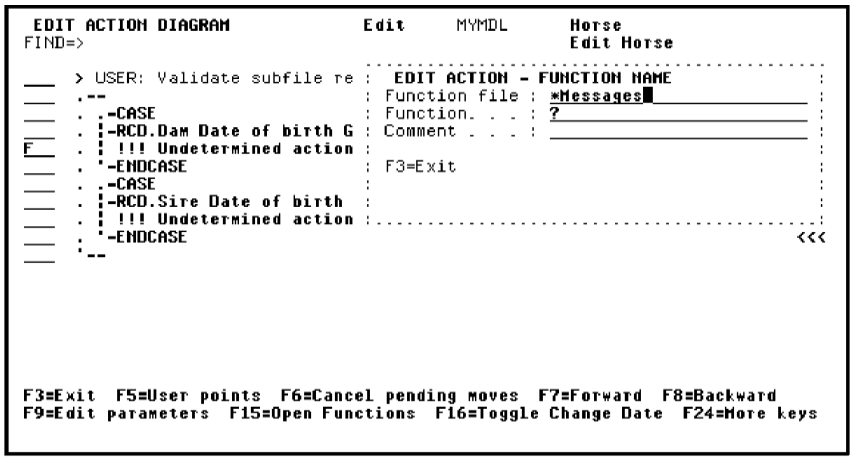
The Edit Action - Function Name window lets you specify the function to be called if the first CASE condition is satisfied.

Because all functions are associated with a file, when you specify a function you must give both a file and a function name. Functions can be divided into three groups:

- Functions attached to user-defined files. For example the Edit Horse function that you are currently editing.
- Functions attached to CA 2E shipped system files. For example, CA 2E user-defined message functions are attached to a system file called *Messages.
- Built-in shipped functions, which are also attached to CA 2E shipped system files. Such functions provide commonly required low level actions; for example, to manipulate fields. Some examples are, *ADD, *SUB, and *MOVE. If you do not specify a file name, CA 2E assumes you are referring to a built-in function.

Message Functions

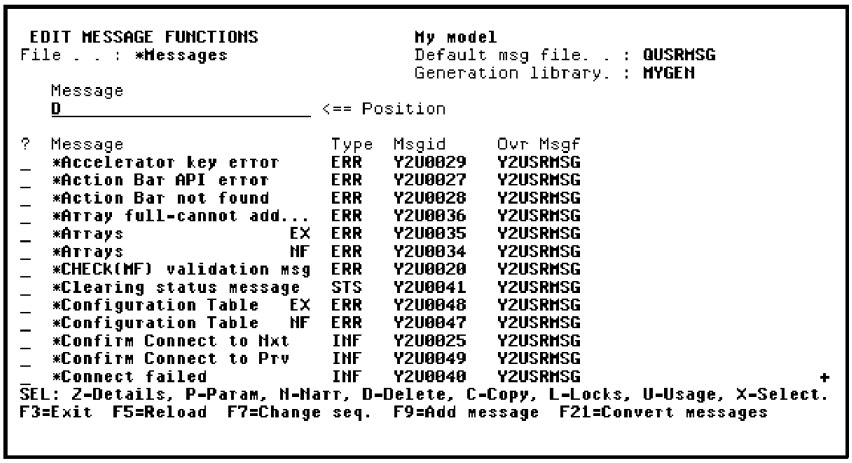
Because you want to send the user-defined message you just defined, type ***Messages** for the Function file as shown and leave the **?** in the Function field.



Press Enter.

Displaying the Message Functions

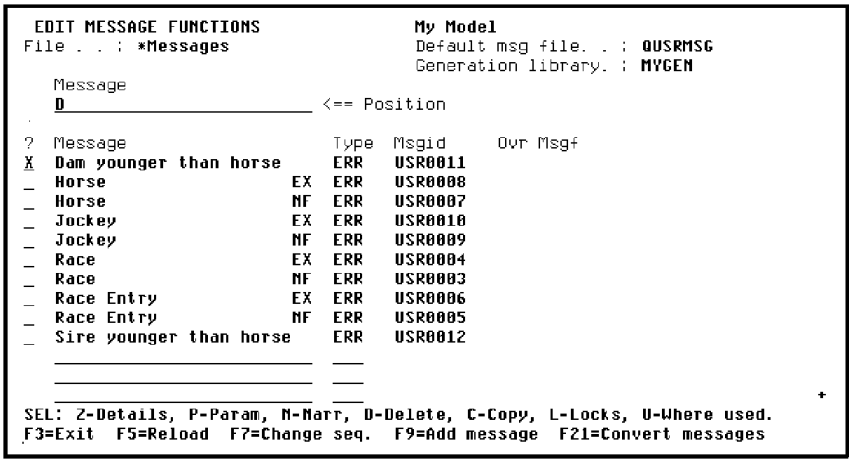
The Edit Message Functions panel displays a list of the message functions available on the file *Messages. Type **D** in the positioner field at the top of the panel as shown to position to the Dam younger than horse message function you just defined.



Press Enter.

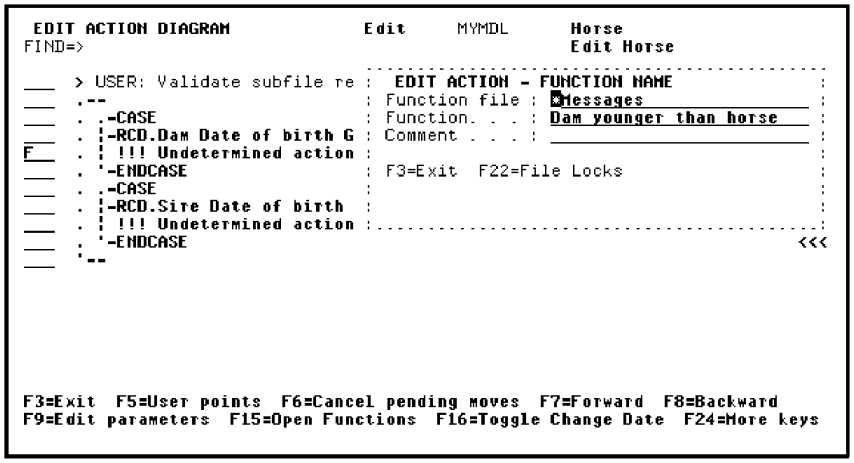
Selecting a Message Function

Type **X** against the message, Dam younger than horse, as shown to select it as the action in the action diagram.



Press Enter.

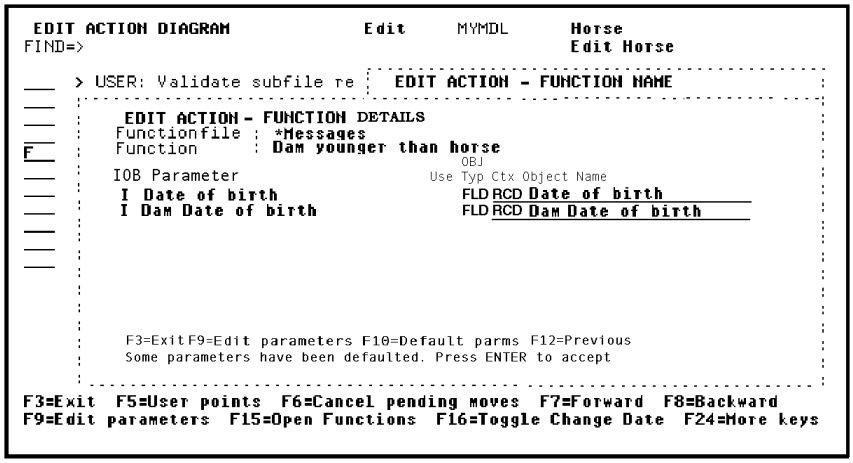
The Edit Action - Function Name window redisplays with the Function field containing the name of the Dam younger than horse message function.



Press Enter.

Function Details for the First Action

The details of the function, including the parameters which are to be passed to the message function, are displayed. Check that the information shown is correct.



Press Enter to accept the information shown and to return to the action diagram.

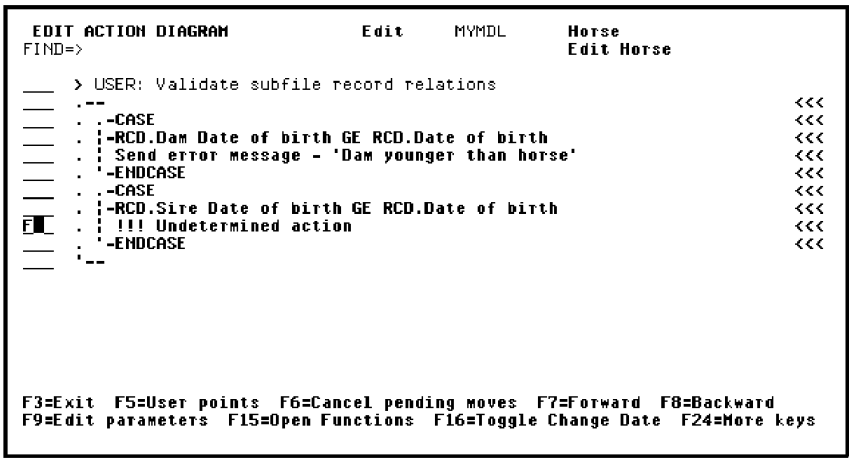
Action Diagram with the First Action Defined

The action diagram now includes a call to the Dam younger than horse message function. In other words, "Send error message - ‘Dam younger than horse’" has replaced the first !!! Undetermined action.

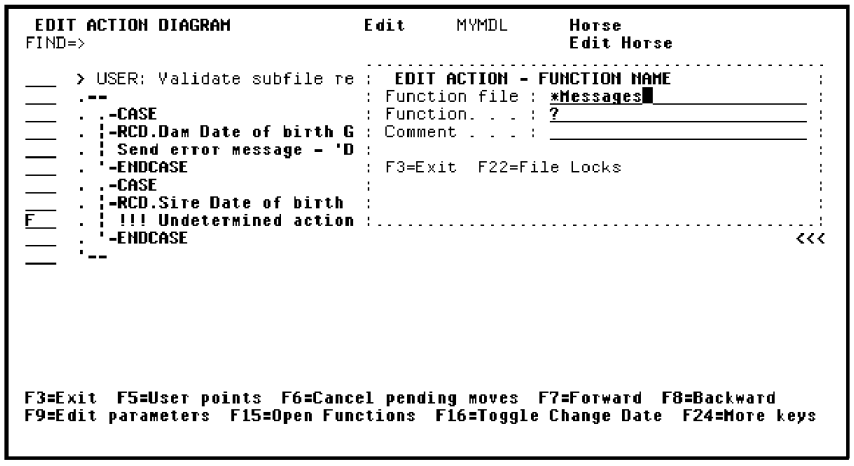
Defining the Second Action

In this step you will define the message function to be called when the sire is younger than the horse. The entire process will take place within the action diagram. Note that you could have defined both message functions in this way.

Type **F** against the second !!! Undetermined action.



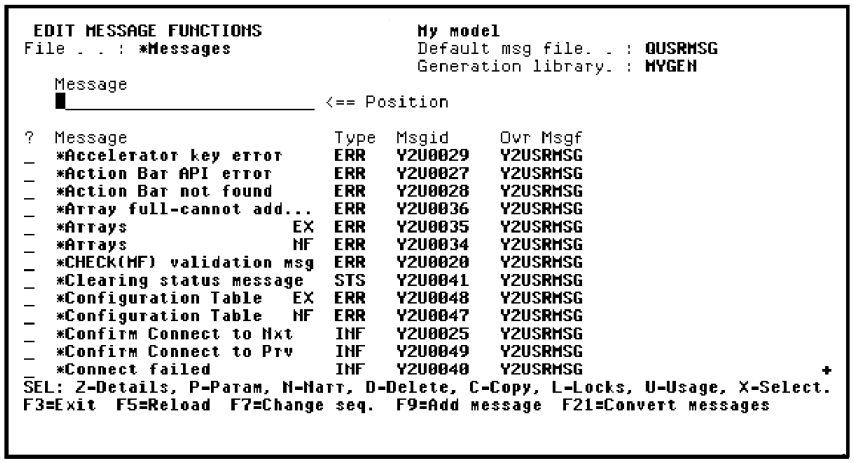
Press Enter. A blank Edit Action - Function Name window displays as before. Type ***Messages** for the Function file field as shown. Leave the **?** in the Function name field.



Press Enter.

Adding the Second Message Function

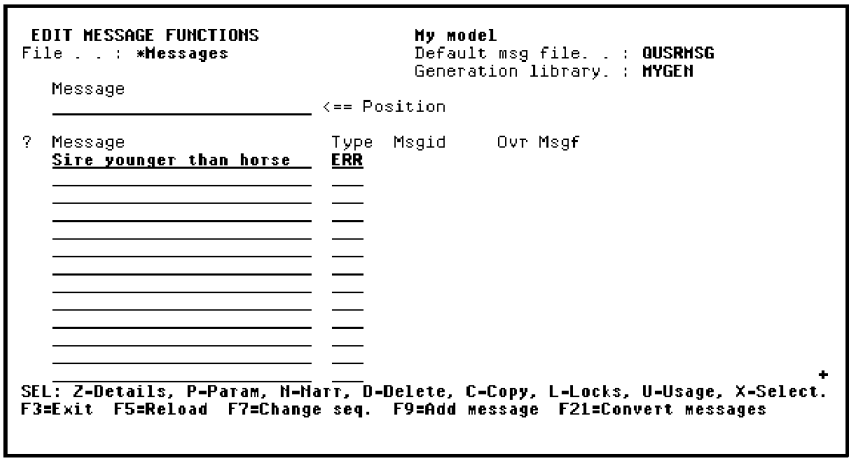
All message functions attached to the *Messages file are displayed. You will now create a new message function to be executed if Sire Date of birth is greater than or equal to Date of birth as you did for the first message function.



Press F9 to add the new message.

Details of the Second Message Function

To define the second message function you need to specify a function name and a function type as you did for the first message function. Name the message function **Sire younger than horse**. The type will be **ERR**. Type the message name and type.



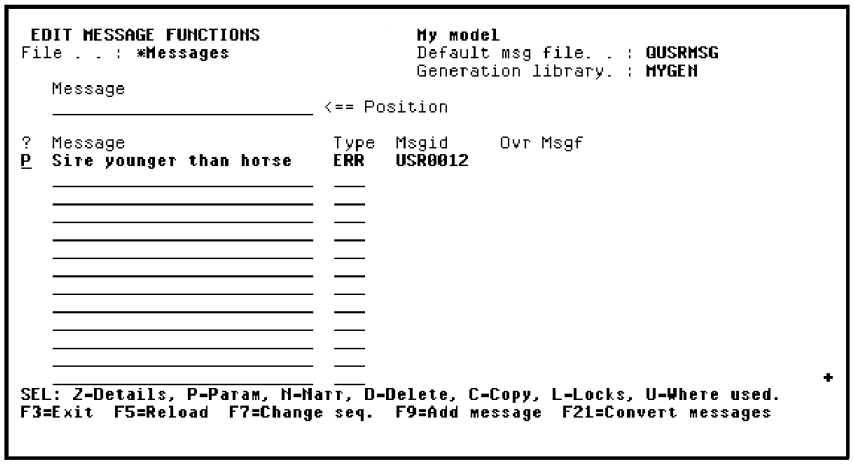
Press Enter.

The second message function has now been added to the *Messages file.

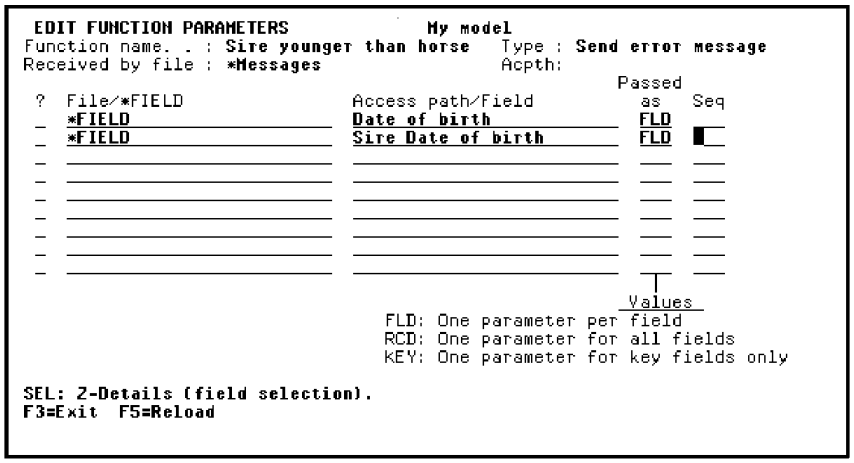
Defining Parameters for the Second Message Function

Parameters define the values to be communicated between functions at execution time. As before the objective is to include the date that caused the error in the text of the error message.

Define the parameters for the second message function in the same way as you did for the first message function. Type **P** next to the message function.



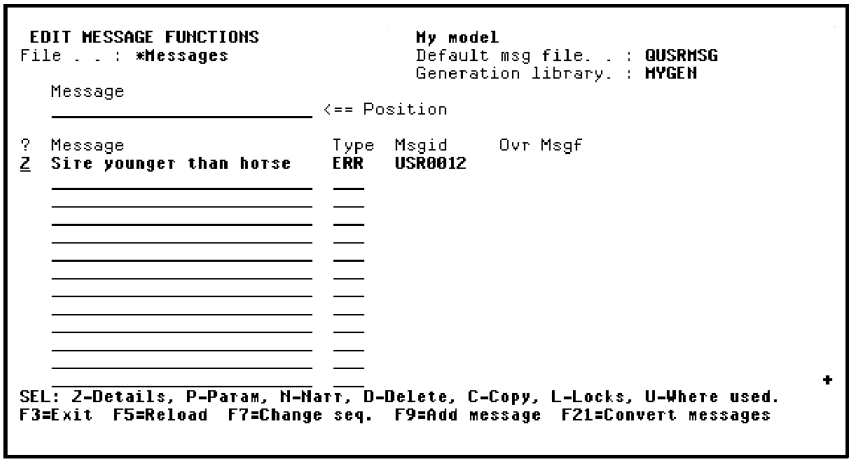
Press Enter to display the Edit Function Parameters panel. Type the message function parameters as shown.



Press Enter to confirm. Press F3 to return to the Edit Message Functions panel.

Message Function Details

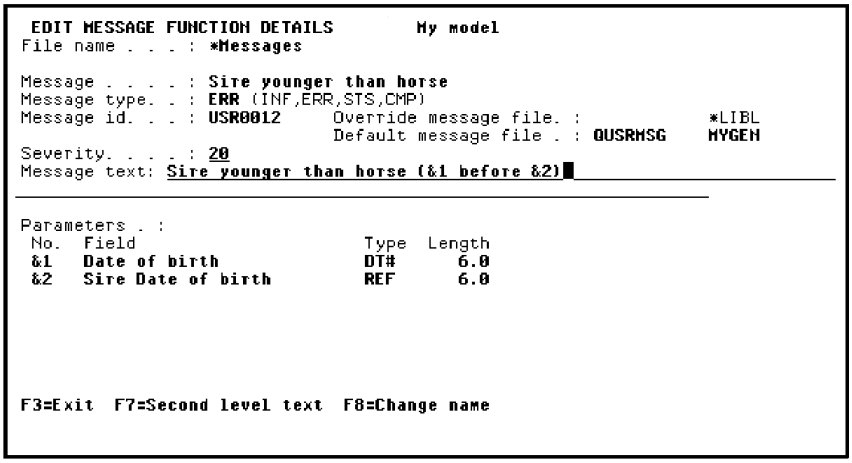
Type **Z** against the second message function as shown.



Press Enter.

Adding Message Function Text

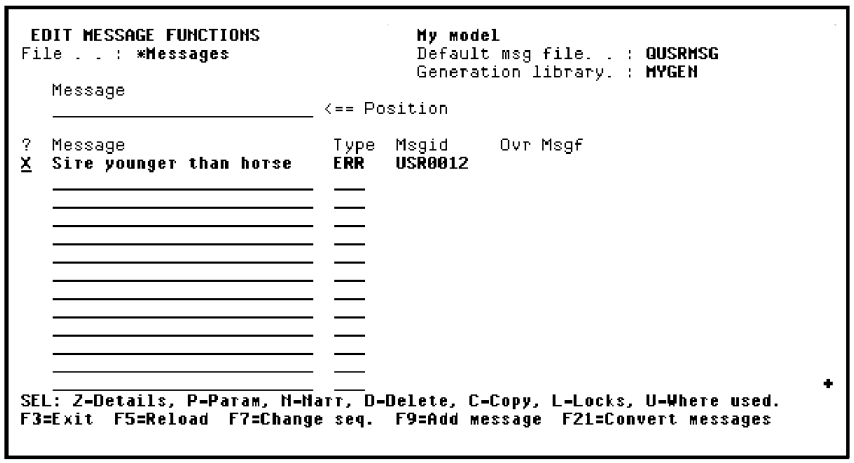
Type the new text containing the substitution variables, "(&1 before &2)", after the default message text as shown.



Press Enter to return to the Edit Message Functions panel. Both required message functions are now completely defined.

Selecting the Second Message Function

You can now select the message function to be executed when the second condition is satisfied. Type **X** to select the message function for the action diagram.



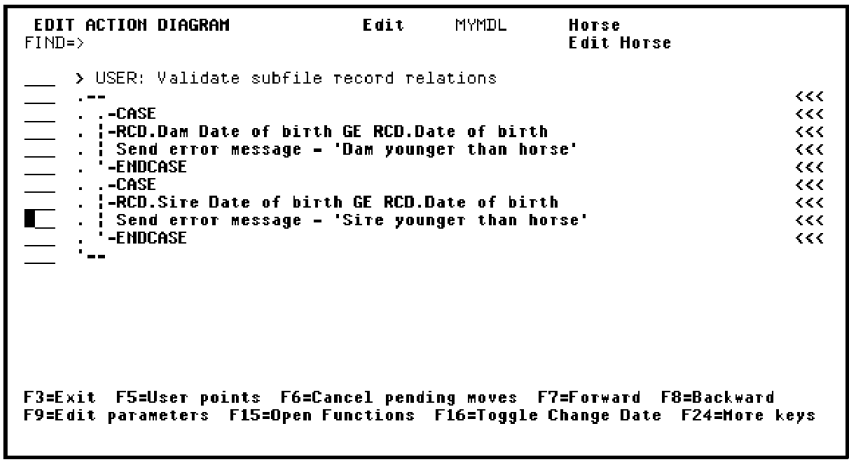
Returning to the Action Diagram

You will press Enter three times to return to the Action Diagram Editor as follows:

1. Press Enter the first time to return to the Edit Action - Function Name window showing the name of the new message function.
2. Press Enter a second time to return to the Edit Action - Function Details window showing the function parameters. A message at the bottom of the window indicates that some parameters have been defaulted. Note that the default context for the parameters is RCD (subfile record).
3. Press Enter the third time to accept the defaulted parameters and return to the Action Diagram Editor.

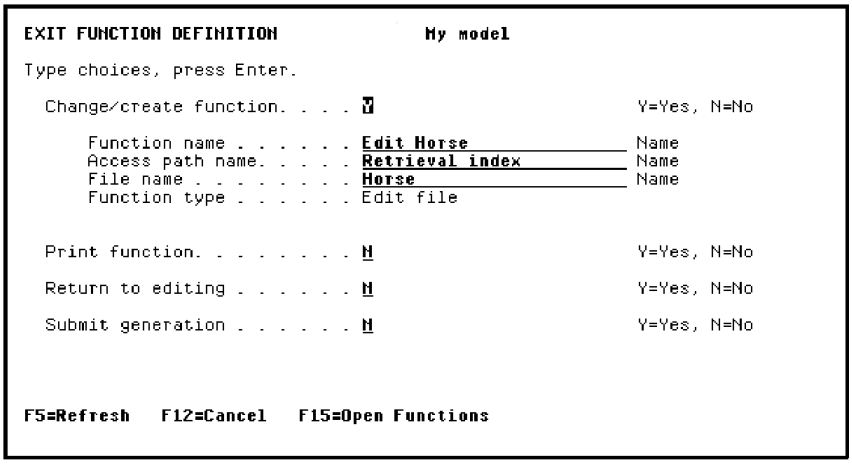
Completed Action Diagram

The USER: Validate subfile record relations construct is now complete.



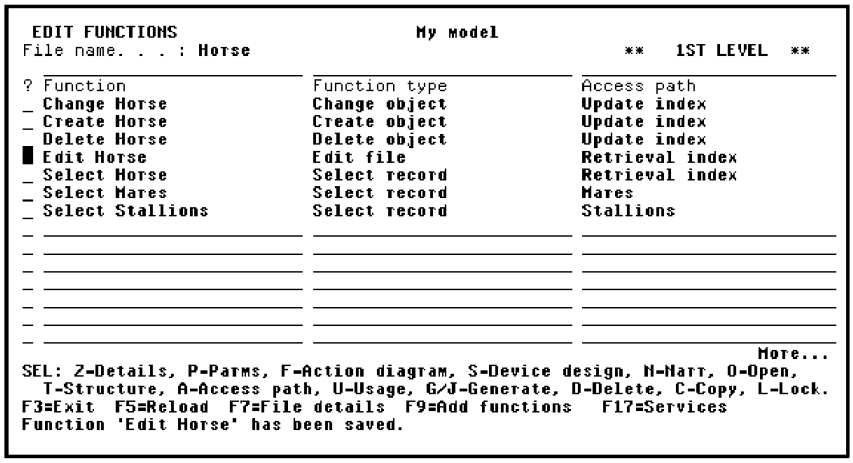
Exiting the Action Diagram

Press F13 (Fast exit) to Exit the Action Diagram Editor and display the Exit Function Definition panel. Accept the defaults on this panel to save all changes you made to the action diagram.



Press Enter to return to the Edit Functions panel.

The message "Function ‘Edit Horse has been saved" displays at the bottom of the panel.



Function Options

This topic discusses how to change the function options for a function.

New term introduced:

- Function Option

New panel introduced:

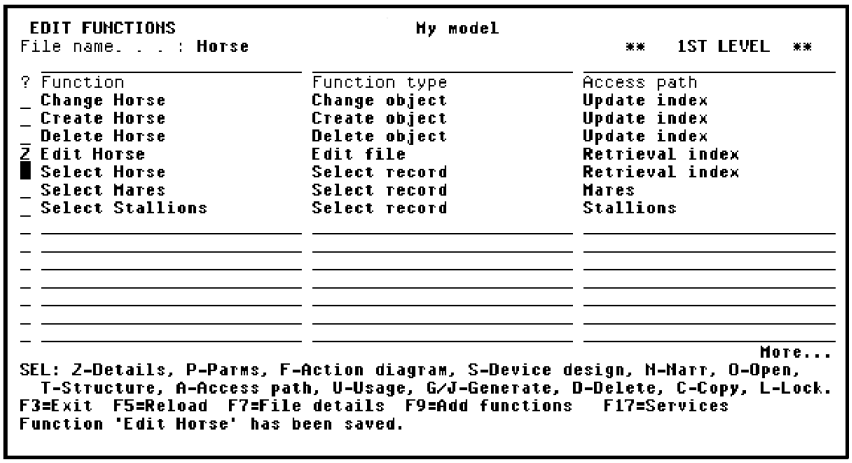
- Edit Function Options

Objectives

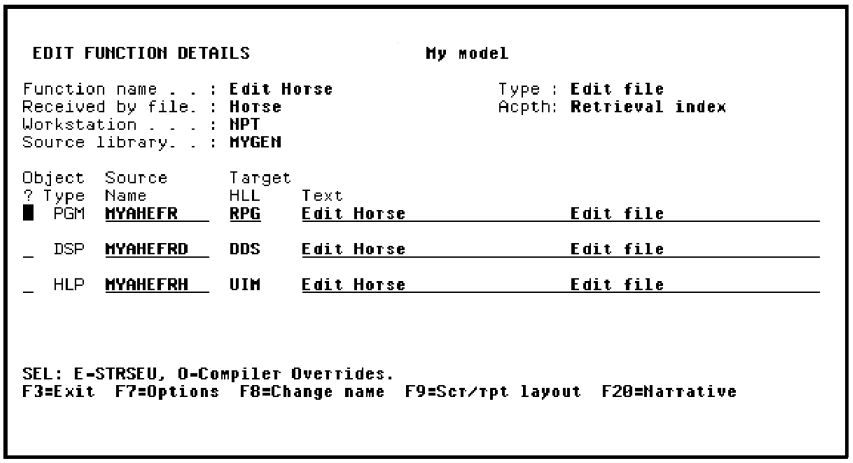
In this topic you will remove the capability to delete records from the Edit Horse function and change the default confirm prompt to Y.

Accessing the Function Options Panel

From the Edit Functions panel, type **Z** against the Edit Horse function.



Press Enter to display the Edit Function Details panel.



Default Function Options

To view the function options for Edit Horse, press F7. The Edit Function Options panel gives you a simple means of modifying certain aspects of the behavior of a function. The actual function options available and their defaults depend on the function type. In this step you will change the two following function options for the Edit Horse function.

Delete record
This option specifies whether you can delete database records. The default is Y, which means you can delete database records. If you change this value to N, the function code will not include logic to delete database records.

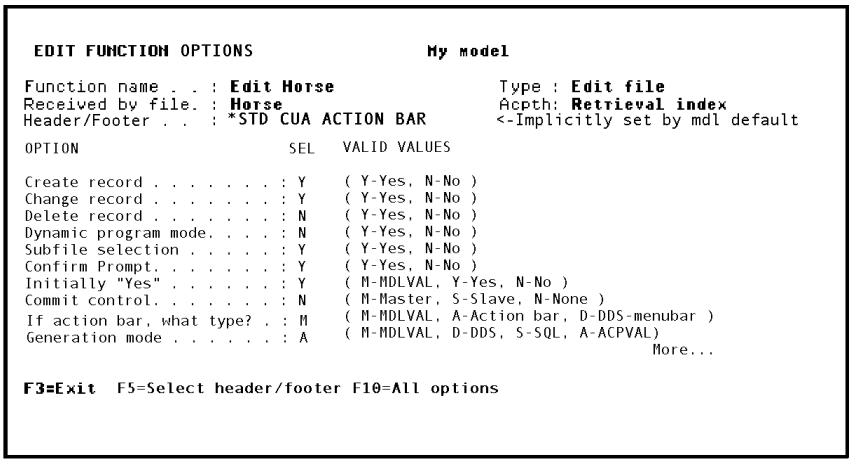
Confirm prompt
The Confirm prompt option specifies whether the Edit Horse function prompts for confirmation before updating the database files. The default is Y.

If the Confirm prompt option is Y, a related option, Initially "Yes", specifies whether the default for the confirm prompt will be Y or N.

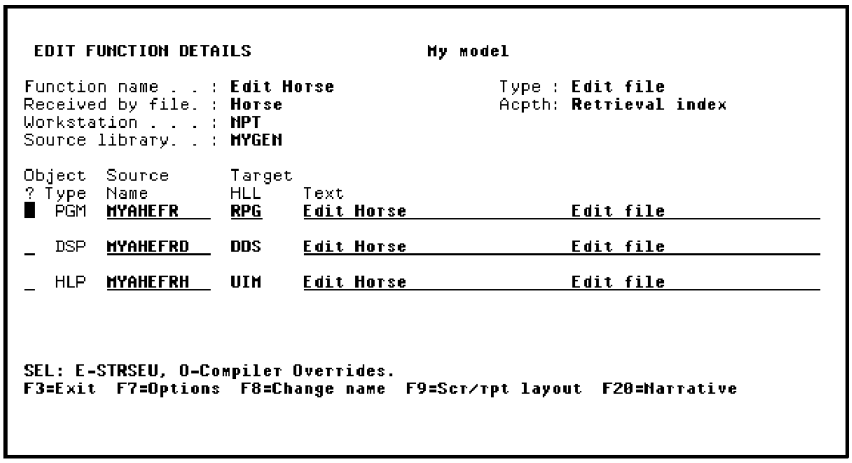
Note: When the default value for a function option is M, it means that the function option's default value depends on the setting of a model value. The current setting of the model value is shown highlighted or underlined.

Changing Function Options

From the Edit Function Options panel for the Edit Horse function, change the value for the Delete record option to N. Change the initial value of the confirm prompt by typing **Y** for Initially 'Yes'.



Press Enter to return to the Edit Function Details panel.



Press F3 to return to the Edit Functions panel.

Linking Functions

This topic shows how to use CA 2E to link two functions together to build a larger function. In the course of doing this, you will define the second function and an access path upon which it is based.

New terms introduced:

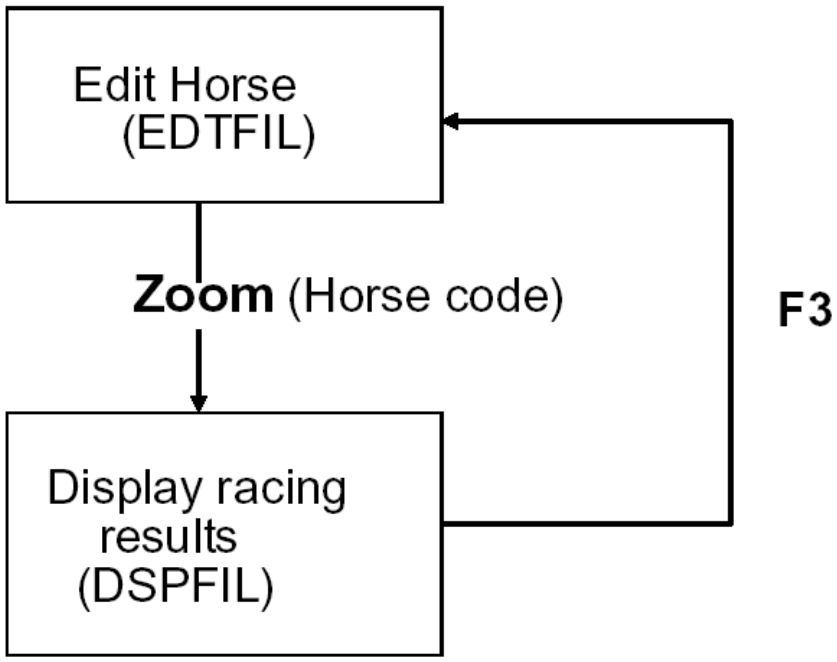
- Built-in function
- *MOVE function

New panels introduced:

- Action Diagram User Points
- Edit Access Path Format Entries

Objectives

Modify the Edit Horse function to give it a Subfile selector option. The new facility should let the user zoom against the detail line for a particular horse on the Edit Horse panel in order to view a subsidiary display of the races in which the horse has raced.



To achieve this, you will define a new function, Display Racing results. The new function will display information from the RACE ENTRY file for the horse selected. You also need to modify the action diagram of the existing Edit Horse function to call the new function. Note that CA 2E lets you define the new function while you are in the process of modifying the action diagram of the existing function.

The two functions will be implemented as separate programs. The Horse code of the selected line will be passed as a parameter between the two programs in order for the Display Racing results program to display race entries for a particular horse.

Steps Required to Link Functions

In this topic, you will define a Display Racing results function and link it to the Edit Horse function. You will also update the Edit Horse function. The process is similar to that of adding a message function. To achieve the objectives for this topic you will need to:

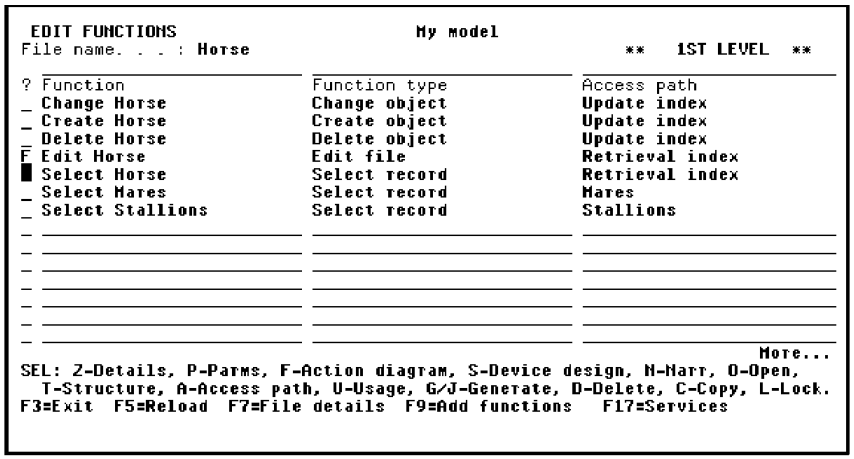
1. Modify the action diagram of the Edit Horse function to specify the condition under which the Display Racing results function is to be called.
2. Define the Display Racing results function by giving it a name and type and selecting an appropriate access path based on the RACE ENTRY file.
3. Specify the parameters of the Display Racing results function; namely, Horse code.
4. Select the Display Racing results function for inclusion in the action diagram of the Edit Horse function.
5. Modify the action diagram of the Edit Horse function to defer updating the HORSE file if a line selection is made.
6. Modify the Selector Choice menu of the action bar of the Edit Horse function's device design to show the availability of the Display Racing results function.

Modifying the Edit Horse Action Diagram

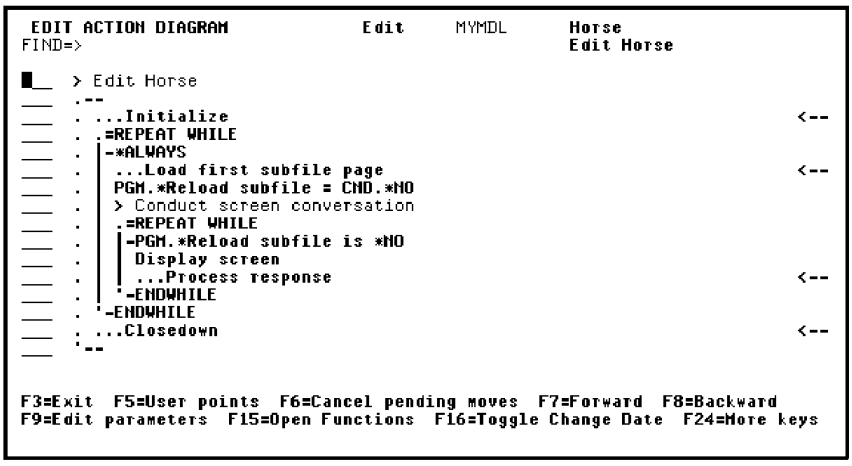
The first step is to modify the action diagram of the Edit Horse function so that the Display Racing results function is called when the end user types **/** in the Subfile selector field and selects the Display Racing results action from the Selector Choice menu of the Edit Horse function's action bar.

You will access the Action Diagram Editor for the Edit Horse function and locate the USER: Validate subfile record relations user point. Recall that user points are the areas in an action diagram that you can modify.

From the Edit Functions panel, type **F** next to the Edit Horse function to enter the action diagram.



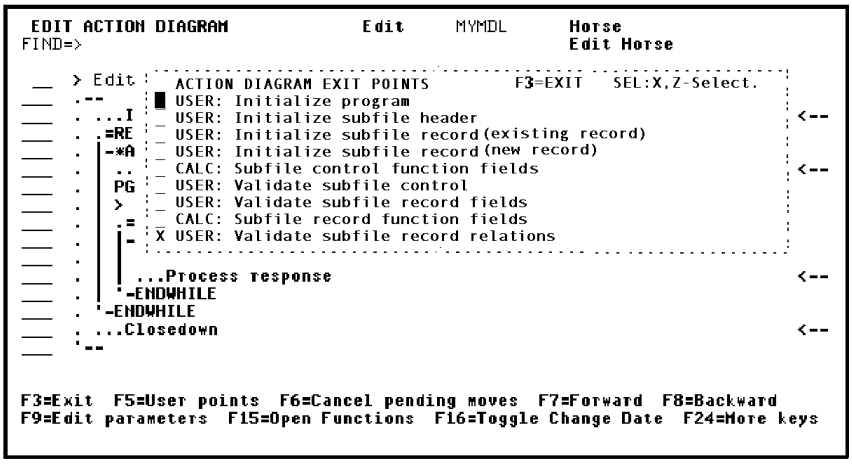
Press Enter.



Obtaining the Action Diagram User Points

Press F5 to view the list of user points for the Edit Horse function.

Type **X** against USER: Validate subfile record relations.



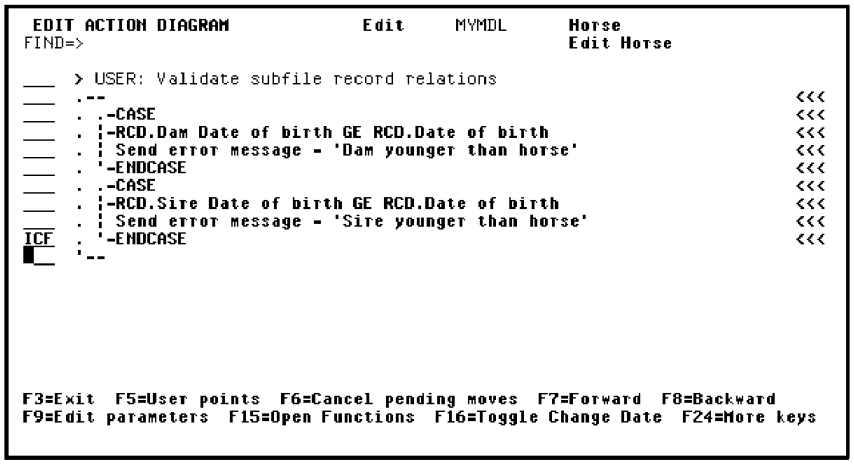
Press Enter.

Adding a New CASE Construct

Insert a condition in the action diagram that defines when the Display Racing results function is to be called. You can insert the new condition after the validation conditions you added in the previous topic.

Inserting a Condition

Type **ICF** to insert a new condition.



Press Enter to display the Edit Action - Condition window.

Entering the Condition

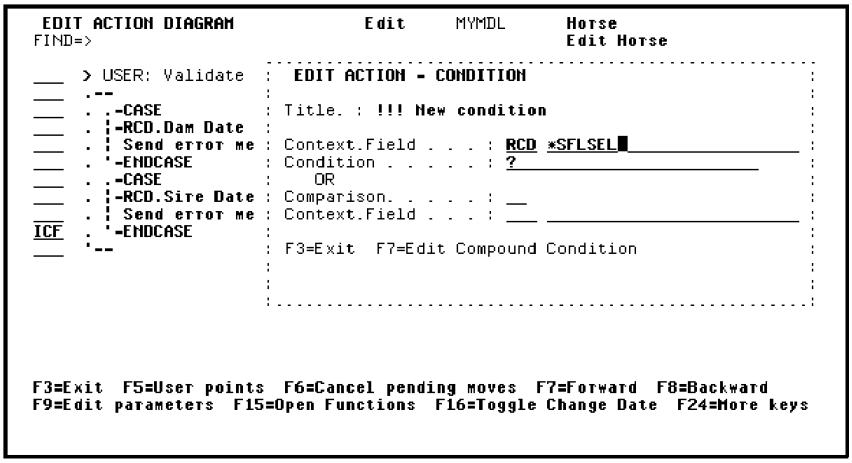
In this step you will add the *Zoom#1 condition for the Subfile selector field, *SFLSEL, to indicate when the racing results for a given horse should be displayed. The *SFLSEL field is automatically provided on function types that have subfile record formats unless you explicitly suppress it using the function options.

The *Zoom#1 condition will automatically be added to the Selector Choice menu on the action bar of the Edit Horse device design as an action because it is used here in the Edit Horse action diagram.

Entering the Condition Details

The *SFLSEL field is in the subfile record; as a result, its context is RCD. You will specify the context and the field name and leave the **?** in the Condition field to display a list of conditions available for the field.

Type the context **RCD**, field name ***SFLSEL**, and **?** for the condition.

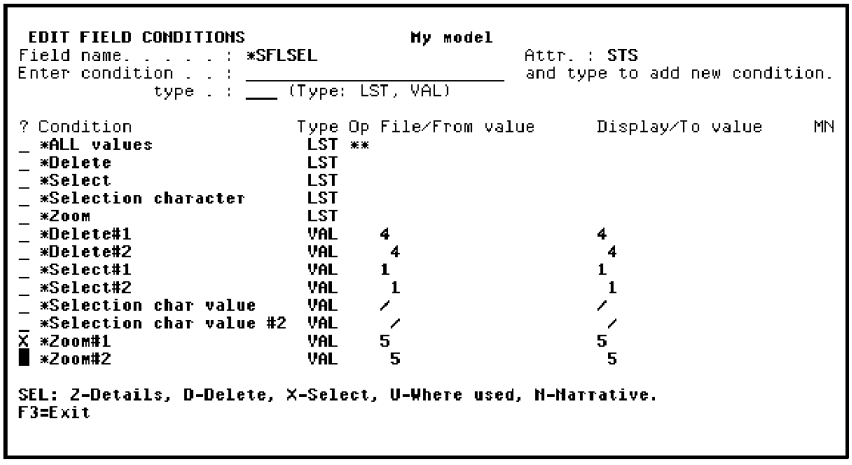


Press Enter.

Selecting the Zoom#1 Condition

The Edit Field Conditions panel shows the existing conditions for the *SFLSEL field. You may select an existing condition or create a new one.

Type **X** against the *Zoom#1 condition.

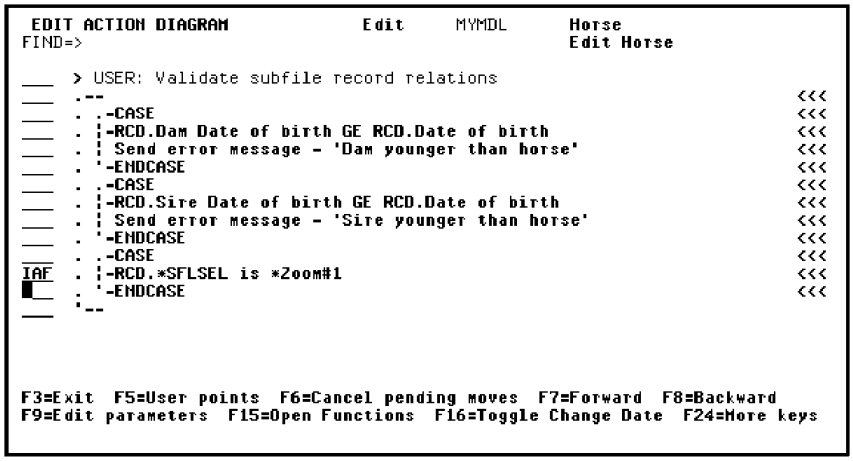


Press Enter to return to the Edit Action - Condition and note that Zoom#1 has been inserted in the Condition field. Press Enter again to return to the Action Diagram Editor.

Specifying a Function as the Action

You have now finished defining the condition. Next you will specify a function to be called as the action when the condition is true.

Type **IAF**.

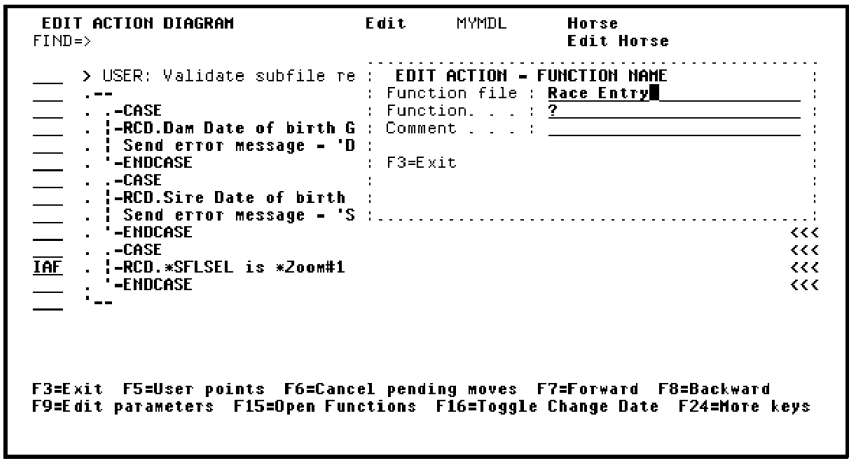


Press Enter to display the Edit Action - Function Name window.

Naming the Function

To specify a reference to a function, type the name of the file to which the function is attached and the function's name. If you enter a file name and leave the **?** for the Function name field, CA 2E will display a list of functions that already exist for the file. You can select one or define a new one. In this case, the function to display race results does not yet exist. Note that the new function will be attached to the RACE ENTRY file, not the HORSE file.

Type **Race Entry** and leave the **?**.



Press Enter to go to the Edit Functions panel for the RACE ENTRY file.

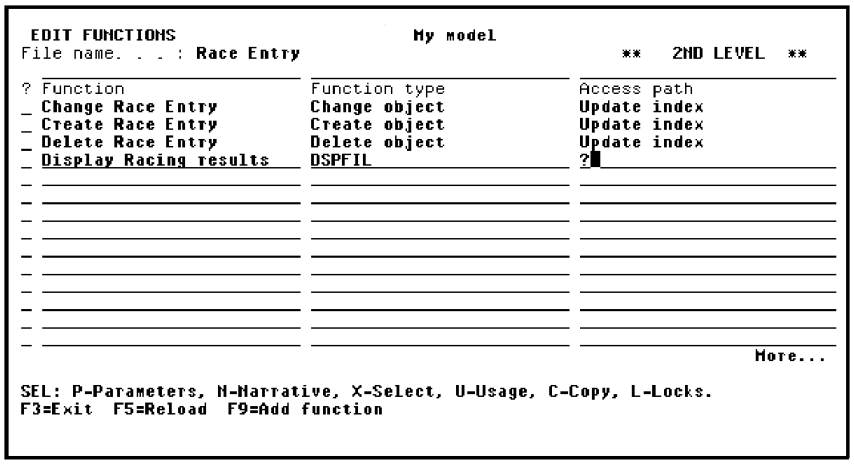
Selecting a Function

The Edit Functions panel shows all the existing functions for the RACE ENTRY file. Note that if this is the first time you are viewing functions for RACE ENTRY, the default functions (Change Race Entry, Create Race Entry, and Delete Race Entry) are created before the Edit Functions panel displays.

Creating the New Function and Access Path

You are going to define a new function by typing the name of the new function, Display Racing results, and the function type, DSPFIL (Display File). You also need to specify an access path for the function that will display racing results in horse order.

Type **Display Racing results** for the Function name, **DSPFIL** for the Function type, and **?** to display the existing access paths.



Press Enter.

Selecting an Access Path

Because you typed **?** in the access path field, CA 2E displays the Edit File Details panel. You can either select one of the existing access paths displayed or create a new one. In this case, you will create a new access path.

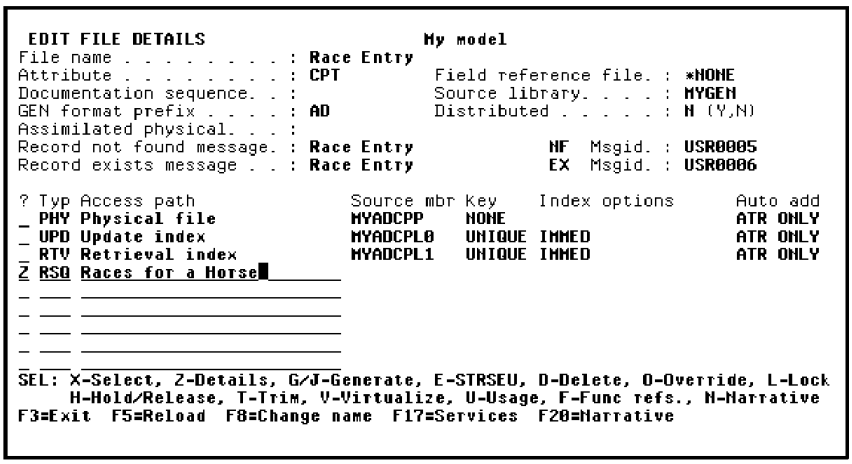
Creating a New Access Path

The existing RTV and UPD access paths for RACE ENTRY will be keyed according to the key relations for the RACE ENTRY file as follows.

1. Course code
2. Race date
3. Race time
4. Entry number

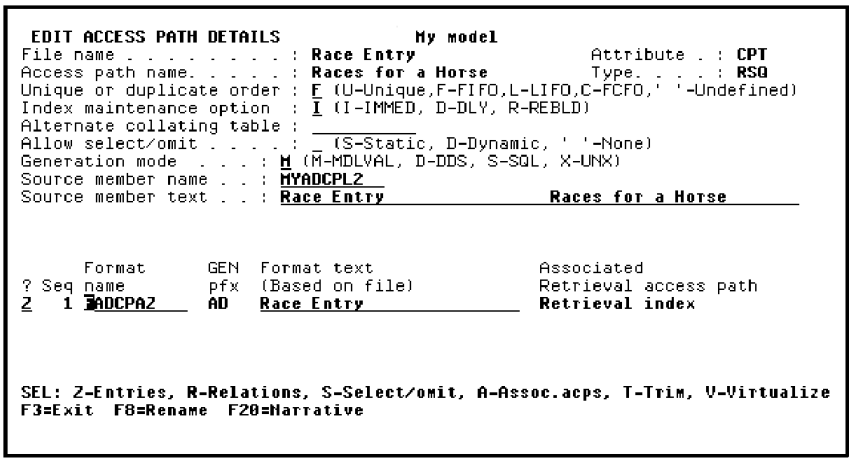
Because you want to see horse racing results in horse order, you need to specify a new access path. Call it Races for a Horse. Use a Resequence access path (RSQ) to specify an alternative key order.

Type **Z** in the Subfile selector field, **RSQ** for the access path type, and **Races for a Horse** for the access path name.



Press Enter.

The Edit Access Path Details panel is displayed. Type **Z** against the format.



Press Enter.

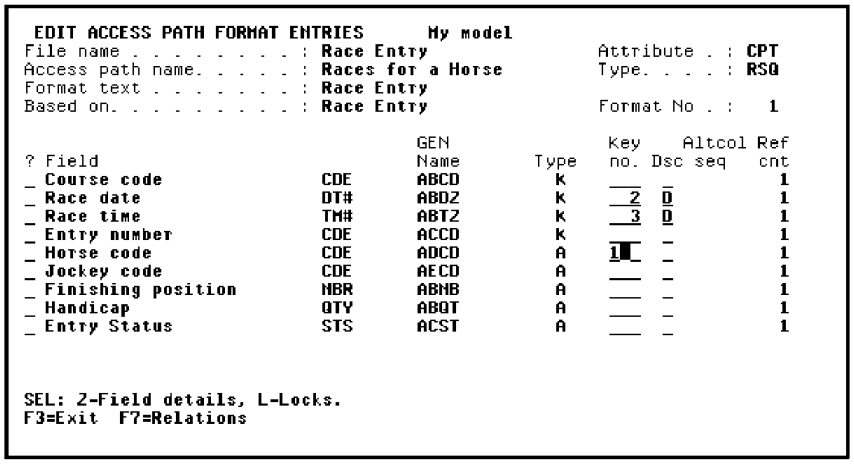
Specifying the Access Path Details

On the Edit Access Path Format Entries panel you will specify the new key order for the RSQ access path. Initially, the keys default to be the same as for the RTV type access path. Remember, you cannot change the keys for a RTV access path.

Defining the Access Path Key

The objective for the resulting application is to give the end user the capability to view race results for a selected horse. The recommended order by which to retrieve the RACE ENTRY file is by Horse code, Race date, and Race time, with the most recent races retrieved first. Specify a key of Horse code followed by Race date (descending order) and Race time (descending order). Before you change the order, clear the numbers from the original key order.

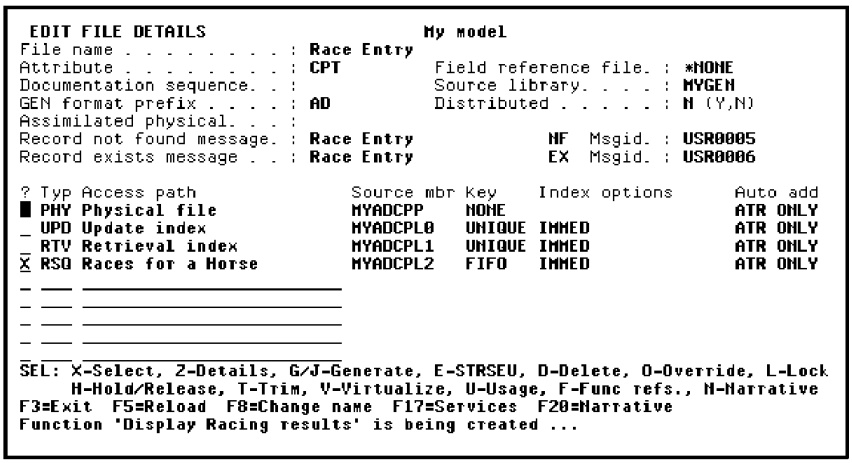
Type the details as shown. In other words, blank the Key no. field for Course code and Entry number; type **2**, **3**, **1** in the Key no. field for Race date, Race time, and Horse code, respectively; and type **D** (descending order) in the Dsc column for Race date and Race time.



Press Enter to confirm the keys. Press F13 to return to the Edit File Details panel.

Selecting the Access Path

Now select the new access path for use in the Display Racing results function. Type **X** to select the Races for a Horse RSQ access path.



Press Enter. The message "Function ‘Display Racing results’ is being created" appears at the bottom of the panel. When the function is created, CA 2E returns to the Edit Functions panel, which now includes the Display Racing results function.

Function Parameters

Function parameters specify which fields can be passed between the calling and the called functions. Each call can pass different values in these fields.

New terms introduced:

- Function parameter
- Parameter usage
- Parameter role
- Restrictor parameter

New panels introduced:

- Edit Function Parameter Details
- Work with Choices
- Work with Actions of a Choice
- Edit Action
- Work with Actions of a Choice

Objectives

In this topic you will specify the parameters that will be passed from the Edit Horse function to the Display Racing results function in order to view races for a particular horse.

Understanding Parameter Usage and Role

You need to assign a usage and role for each parameter to direct the calling and called functions to use the parameter in a particular way.

- **Parameter Usage**—A parameter's usage determines how the parameter is to be received from or returned to the calling function. The four possible usages are Input only, Output only, Both, and Neither. In this tutorial you will use only the first two usages:
  - **Input only—**A value for the parameter is passed to the called function. The called function returns the value to the calling function without changing it.
  - **Output only—**The called function returns a value for the parameter to the calling function when the called function ends.
- **Parameter Role**—The role of a function parameter specifies how the parameter will be used in the called function. The four possible roles are Map, Restrictor, Positioner, and Vary. In this tutorial you will use only the Restrictor role.

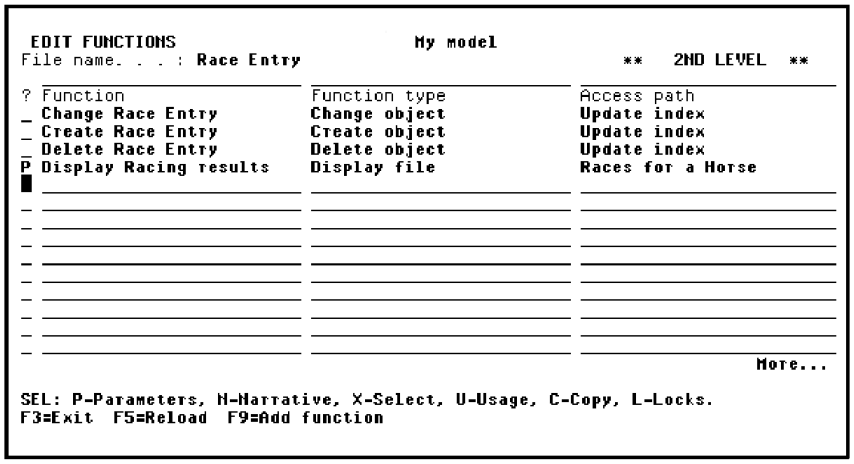
The Restrictor role is used to restrict which records from a database file will be displayed, changed, or printed by the called function. A Restrictor parameter must be a key field on the access path to which the called function attaches. If there are multiple keys, the parameter sequence must match the key sequence.

You will specify the Restrictor role for the Horse code parameter to restrict the Display Racing results function to show only race entries for a particular horse. In other words, the function can only process database records whose keys match the Horse code parameter.

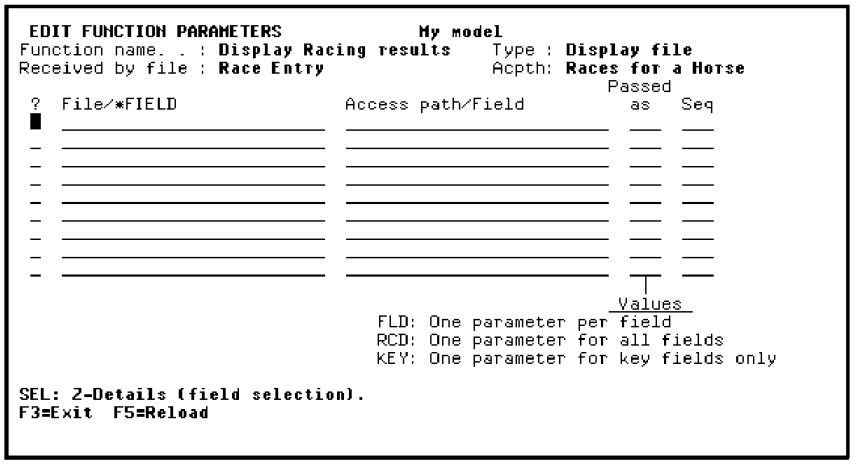
Specifying Function Parameters

In this step you will specify the parameters that will be passed to the Display Racing results function.

Type **P** to edit the parameters.



Press Enter to display the Edit Function Parameters panel.



Defining Function Parameters

The Edit Function Parameters panel lets you specify parameters for a function in one of two ways:

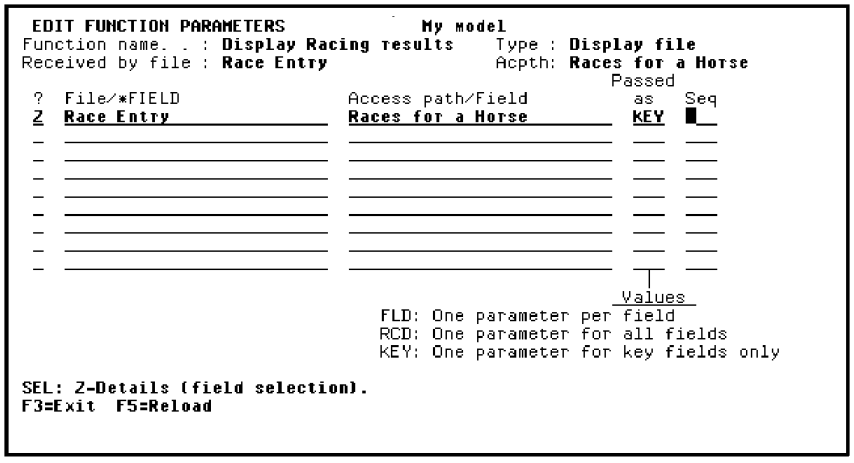
- You can specify individual fields as parameters by typing the keyword ***FIELD** under File/*FIELD and typing the field name under Access path/Field. In the Passed as column, type **FLD** to indicate that each field is passed as an individual parameter.
- You can specify a set of fields from a file by typing a file name under File/*FIELD and the name of an access path based on the file. In the Passed as column, type one of the following.
  - **RCD** to indicate that all the fields from the access path are to be passed as a single parameter. **KEY** to indicate that only the key fields from the access path are to be passed as a single parameter.

  In this case, you can zoom to display the Edit Function Parameter Details panel to select fields from the specified access path.

You used the *FIELD method to specify parameters earlier in the tutorial when you defined error message functions for the Edit Horse function. In this topic you will use the access path method.

Specifying Parameters Using an Access Path

Type the details as shown: type **Race Entry** as the file name, type **Races for a Horse** as the access path, type **KEY** as the Passed as value, and type **Z** in the Subfile selector to select the key fields on the access path to be used as parameters.



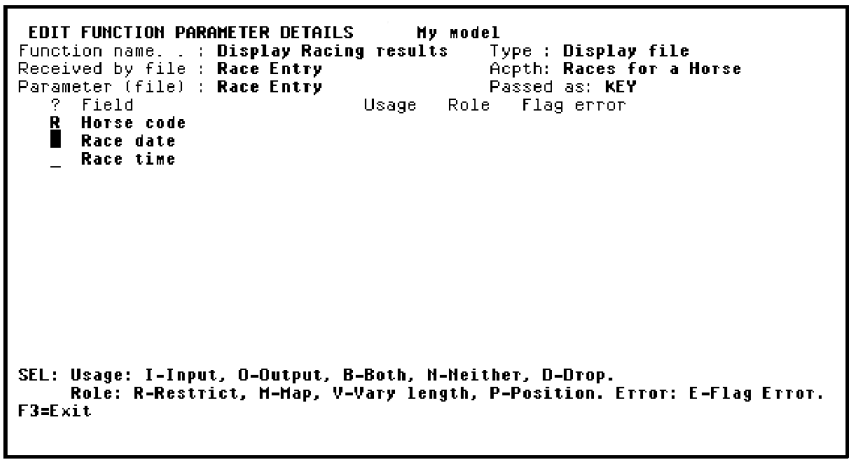
Press Enter to display the Edit Function Parameter Details panel.

Defining Parameter Details

From the Edit Function Parameter Details panel, you can define parameter details.

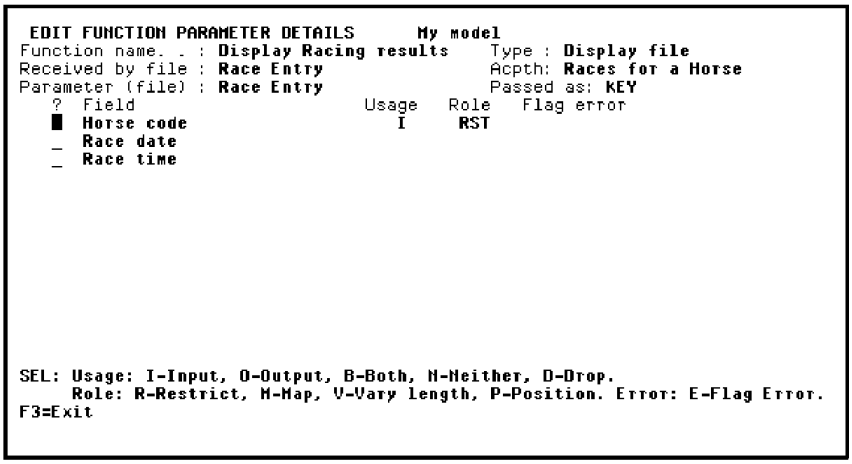
The Horse code from the selected subfile record is to be passed as an input parameter to the Display Racing results function and will be restricted to show only race entries for this horse. This is where you specify that Horse code is an input only, restrictor parameter.

Type **R** against Horse code.



Press Enter.

Note that the Usage for Horse code defaulted to I (Input only) and the parameter role changed to RST (restrictor).



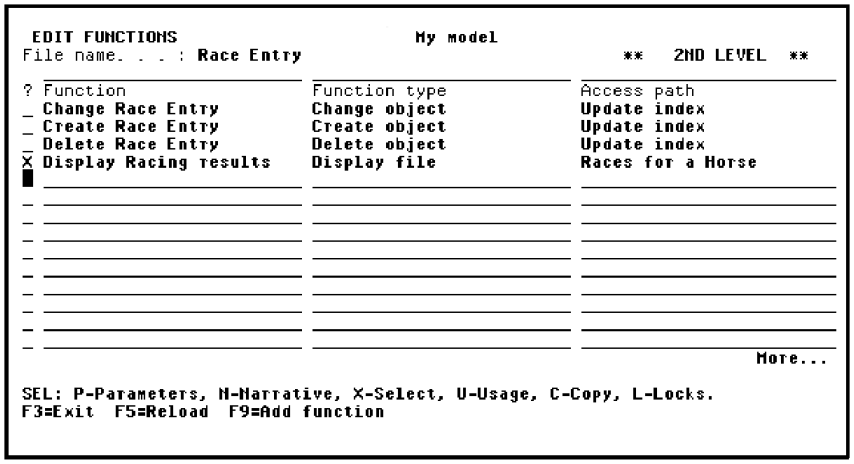
Completing the Parameter Details

You have now finished specifying the parameter details. Fast exit to the Edit Functions panel by pressing F13.

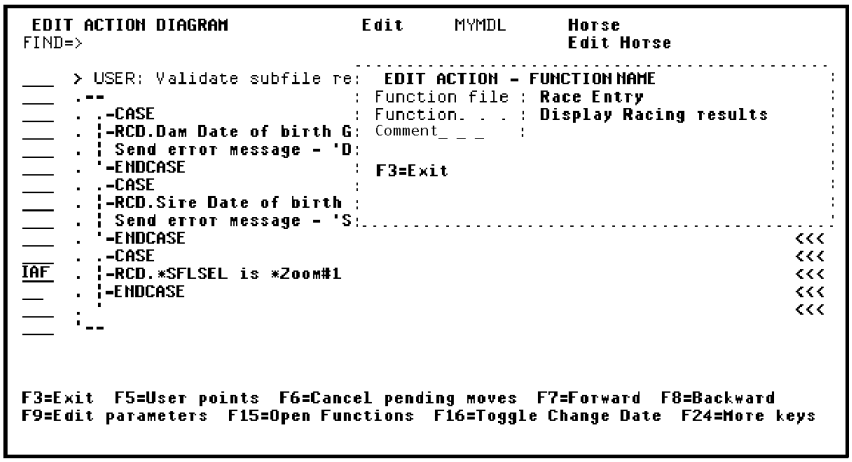
Selecting the New Function

The Display Racing results function is now defined. Remember that you were defining an action in the Edit Horse action diagram. Select the function you have just created as the action you are defining.

Type **X** to select the Display Racing results function.



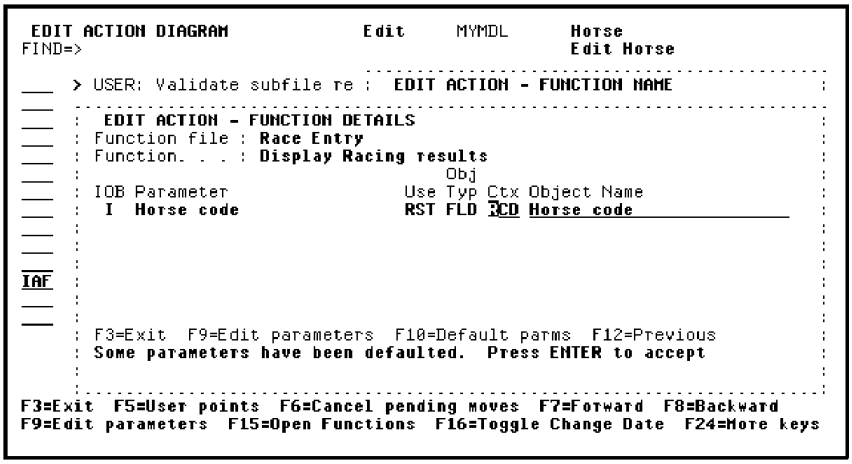
Press Enter to return to the Action Diagram Editor. Note that the name of the Display Racing results function has been inserted in the Edit Action - Function Name window.



Press Enter to display the Edit Action - Function Details window.

Returning to the Edit Action - Function Details Window

This window confirms that you have selected Display Racing results as the function for your action. CA 2E has defaulted the value of Horse code from the subfile record to pass as the parameter to the Display Racing results function. Note the RCD context.



You can override these defaults; however, in this case, you will accept the defaults. Press Enter to accept the defaults.

The action diagram shows that the Display Racing results function is called when the end user selects the Zoom#1 option.

Suppressing the Confirm Prompt and File Update

In this step you will insert an action in the Edit Horse function to suppress both the confirm prompt and the update of the HORSE file when the user zooms to display racing results. As a result, the Edit Horse panel will remain open for input on return from the Display Racing results function. This step is not strictly necessary, but will make the function easier to use.

CA 2E supplies a set of system fields that control the execution of functions. *Defer confirm is the system field that controls whether the confirm prompt is displayed or suppressed. *Defer confirm is a status field (STS) that has two condition values: Proceed to confirm (the default) and Defer confirm. Note that the name of the field and the name of one of its values are the same, namely, Defer confirm.

The action you will insert will override the default for the *Defer confirm field using the *MOVE built-in function and the CND context.

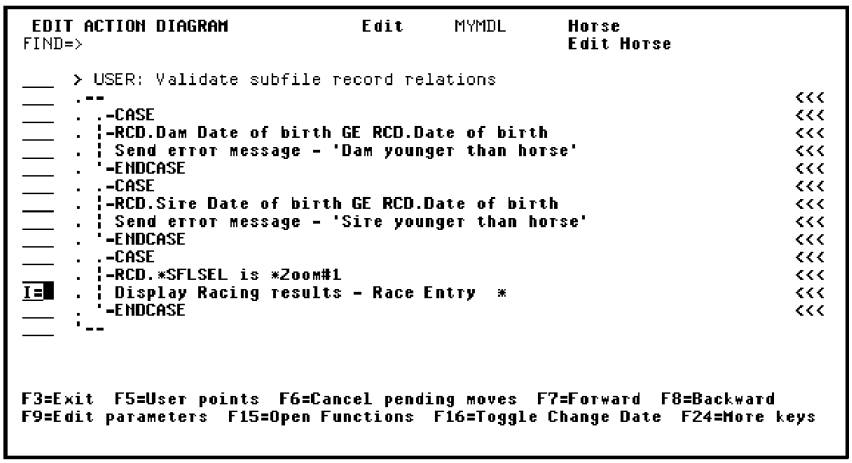
- The *MOVE built-in function lets you move the value of one field to another field. It has two parameters: an input parameter, which is the field to be moved, and an output parameter, which is the field into which the input field is moved.
- The CND context lets you specify a condition value and gives you the capability of specifying a condition value as a field value or a function parameter.

You will use the *MOVE function to move the condition value Defer confirm (CND context) to the system field *Defer confirm, thus overriding the default value of the *Defer confirm system field.

Inserting *MOVE as an Action

CA 2E provides the I= command as a shortcut for inserting a *MOVE built-in function as a new action. The I= command performs three steps in one: IA (inserts an action), F (edits action details), and specifies the *MOVE built-in function as the action.

Type **I=** next to the function to insert a *MOVE built-in function.



Press Enter.

Specifying Parameters for the *MOVE Function

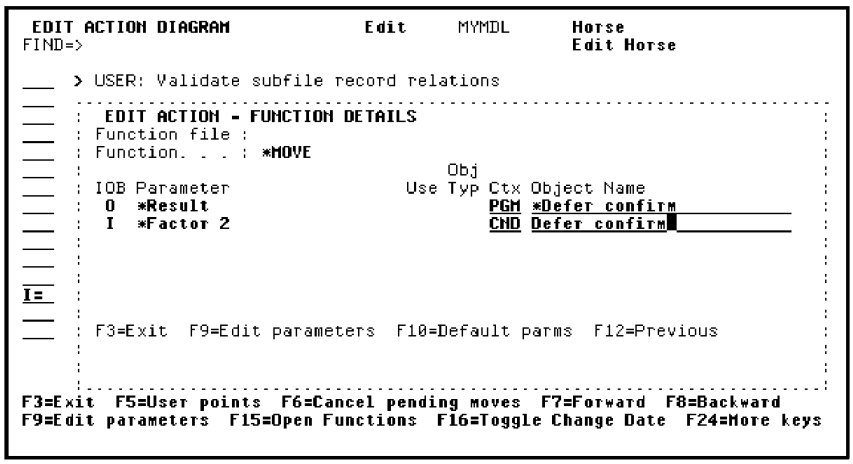
From the Edit Action - Function Details window, you can specify the parameters for the *MOVE function.

The *MOVE built-in function moves the value of one field to another field and thus has two parameters. On the Edit Action - Function Details window the output parameter is indicated by *Result and the input parameter is indicated by *Factor 2.

Recall that you can type **?** in any of the fields on this panel to display a selection list of possible values for the field.

Specifying Defer Confirm

Type parameters for the *MOVE built-in function. In this case, you will type the system field ***Defer confirm** as the output field. Because it is a system field, the appropriate context is **PGM**. The input field is the Defer confirm condition. Type **CND** for the context. This lets you use a condition value as if it were a field value and gives you the capability of specifying a condition value as a function parameter.



Press Enter.

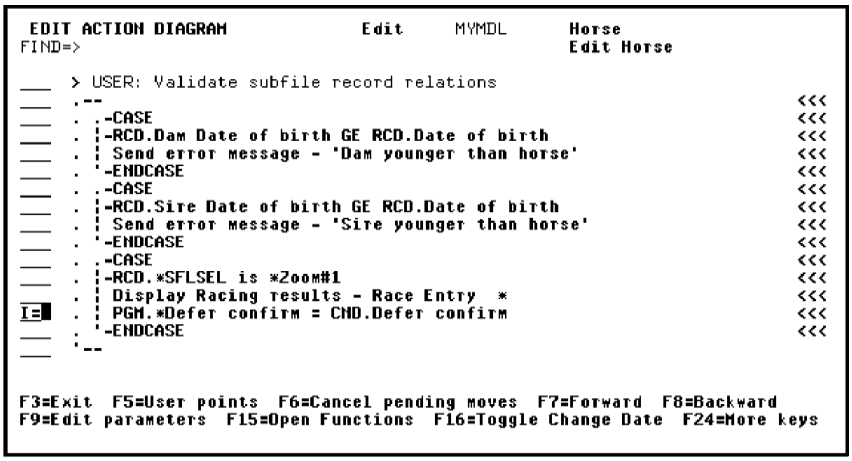
Your action diagram now shows the action that will suppress the confirm prompt.

Reload Subfile

To activate the action bar and call the Display Racing results program, the end user types **/** in the Subfile selector of the Edit Horse program. As the Edit Horse program is now designed, the **/** remains in the Subfile selector when control is returned to Edit Horse. To prevent this, you will add processing in the action diagram to refresh the Edit Horse panel on return from the Display Racing results function. To reload and redisplay this panel, you need to override the default value of the *Reload subfile system field. *Reload subfile is a status field with two conditions: *NO and *YES and has the PGM context.

To accomplish this task, use the method you just used to override the default value of the *Defer confirm system field. In other words, you will insert another *MOVE function to move a value of *YES to the *Reload subfile system field.

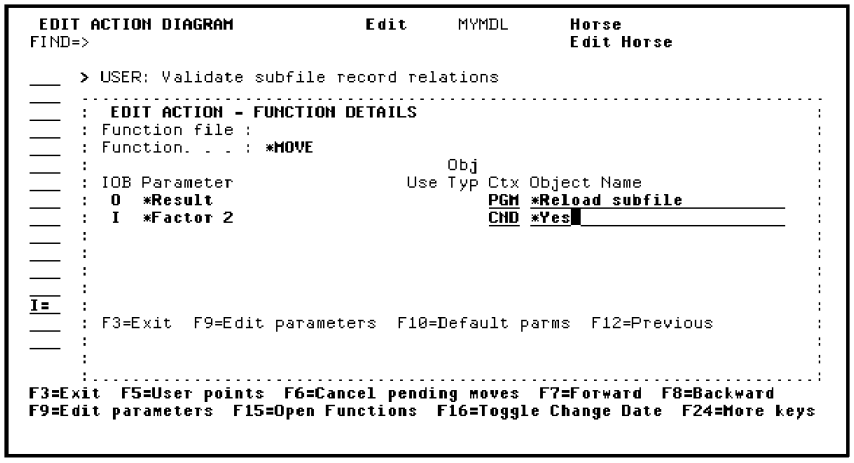
Type **I=** as shown to insert a *MOVE built-in function as an action in the action diagram.



Press Enter.

Specifying Reload Subfile

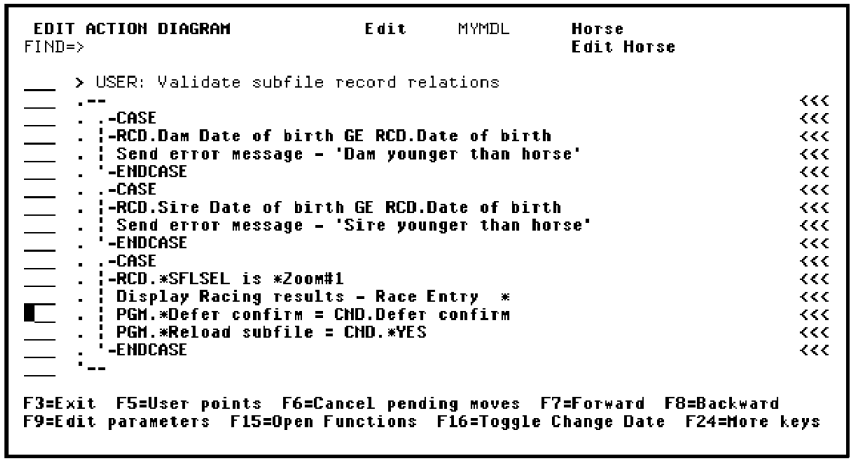
Type the details as shown. In other words, type **PGM** for the output parameter context, type ***Reload subfile** for the parameter name, type **CND** for the input parameter context, and type ***Yes** for the condition to be moved.



Press Enter to return to the action diagram.

The Completed Action Diagram

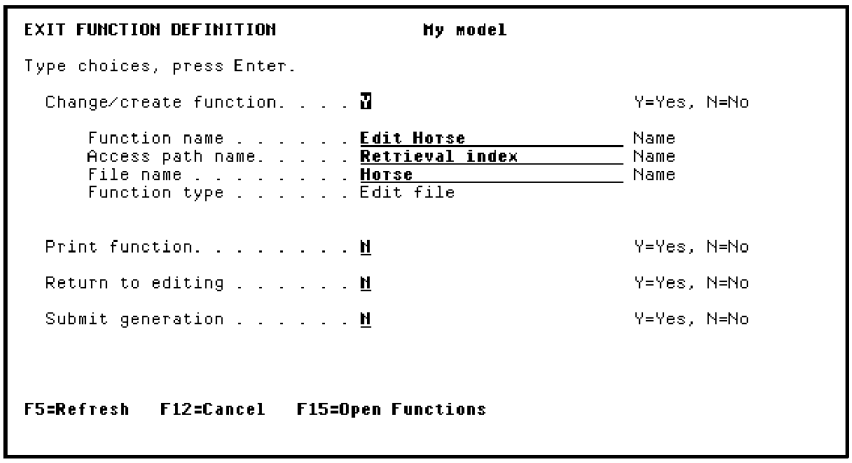
The action diagram now shows the action to reload the subfile.



Press F13 to fast exit to the Exit Function Definition panel.

Saving the Action Diagram

You have now finished specifying the interface to call the Display Racing results function from the Edit Horse function.

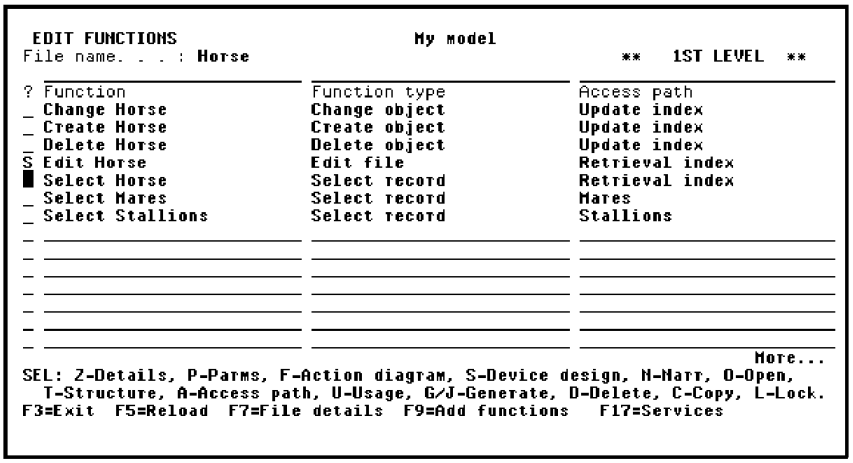


Accept the default of Y (Change/create function) by pressing Enter. You have now completed the update to the action diagram for Edit Horse.

Updating the Edit Horse Function's Action Bar

In this step you will change the text in the Selector Choice menu on the action bar at the top of the Edit Horse function device design to show that the Display Racing results option is available. You will use the Device Design Editor to do this.

Type **S** against the Edit Horse function as shown to access the Device Design Editor.

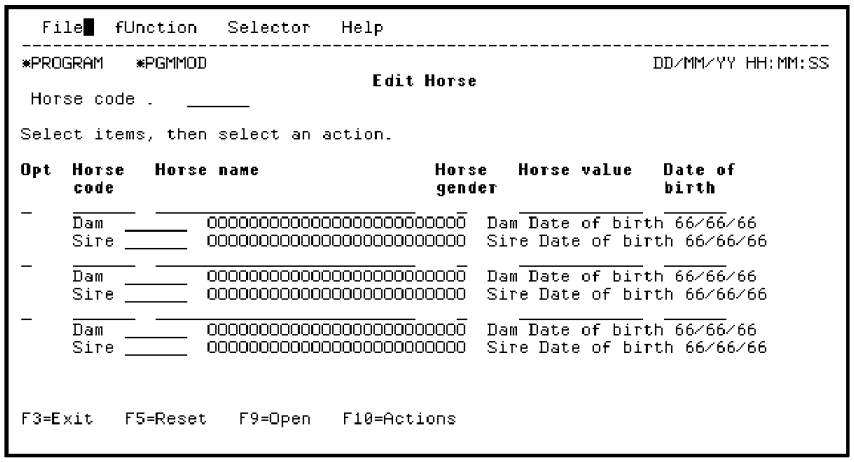


Press Enter.

Updating the Edit Horse Panel's Zoom Action Text

CA 2E has already updated the Edit Horse panel and added the Zoom#1 action to the action bar on the Edit Horse panel. Change the text from Zoom#1 to Display Racing results to provide a better explanation to the end user.

Move the cursor onto the action bar.

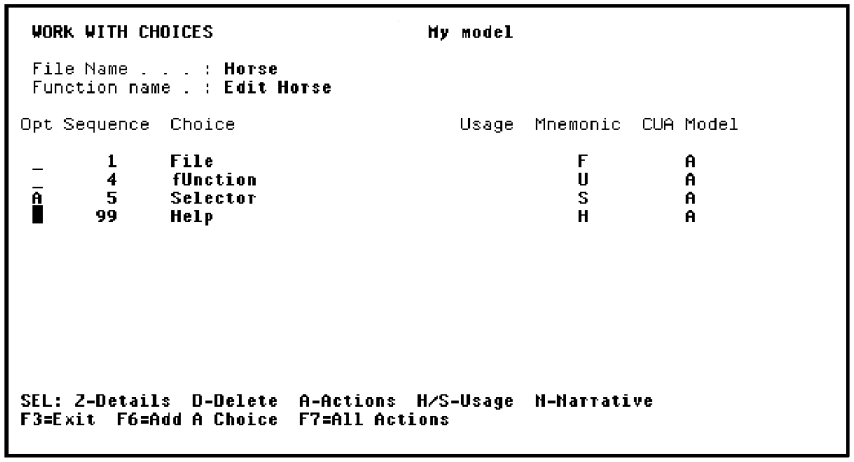


Press Enter to access the Work with Choices panel.

Working with the Selector Choice

From the Work with Choices panel, you can access the Work with Actions panel for a particular choice. In this case, you will work with the actions listed in the Selector Choice menu.

Type **A** next to Selector.

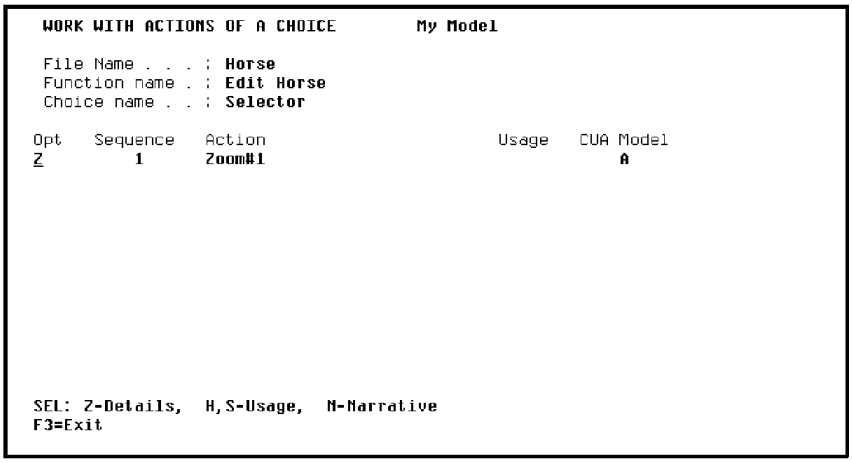


Press Enter.

Modifying an Action

The Work with Actions of a Choice panel lists all the actions available for a particular choice. You will zoom into the Zoom#1 action to view the details for that action.

Type **Z** in the Opt field against the Zoom#1 action.

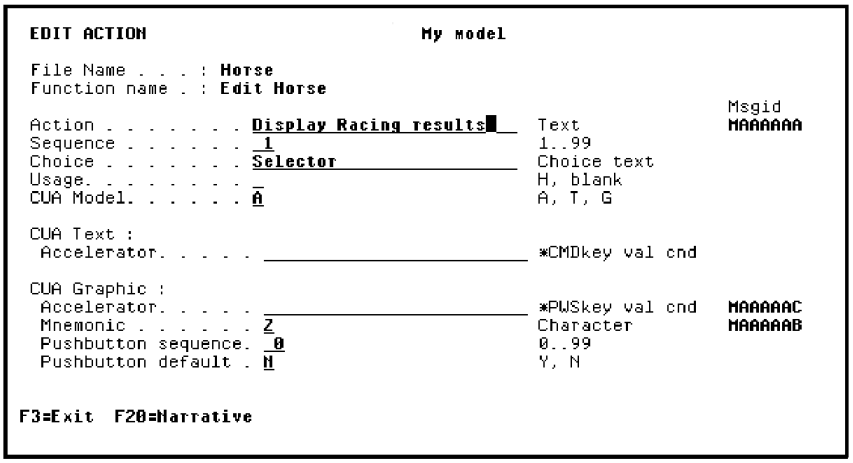


Press Enter.

Changing the Action Text

Update the text for the Zoom#1 action to read "Display Racing results."

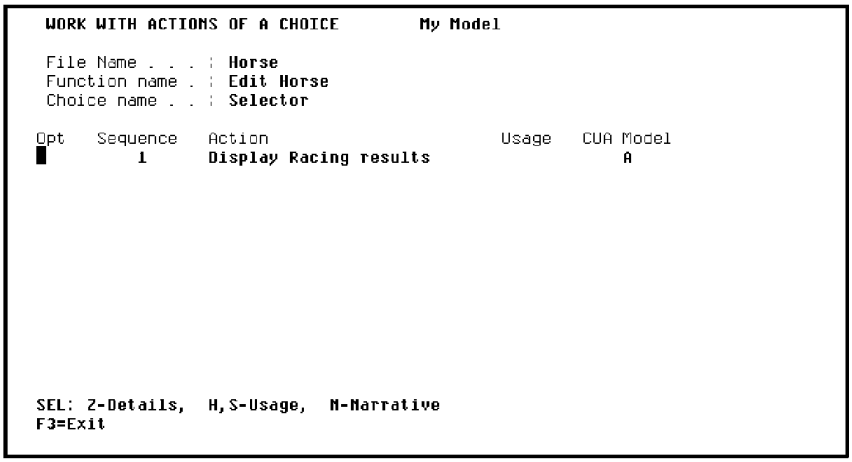
Type **Display Racing results** in the Action text field.



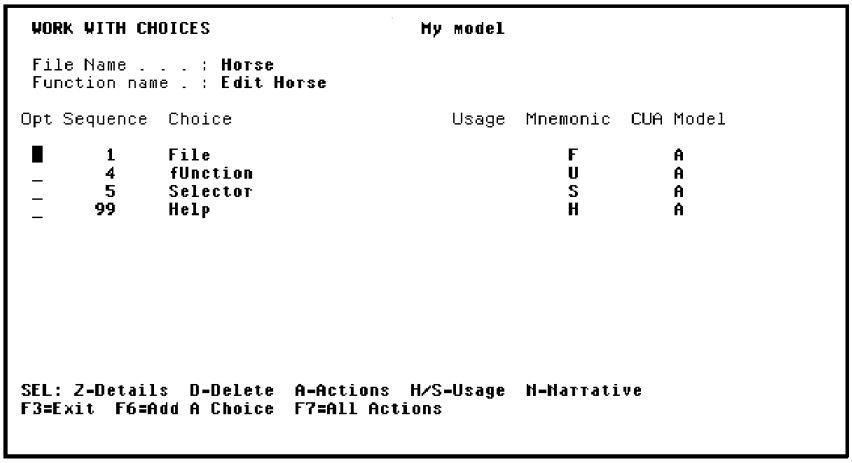
Press Enter to confirm. Press F3 to exit.

Showing the New Action Text

The Work With Actions of a Choice panel now shows the new action text for the Selector Choice menu.

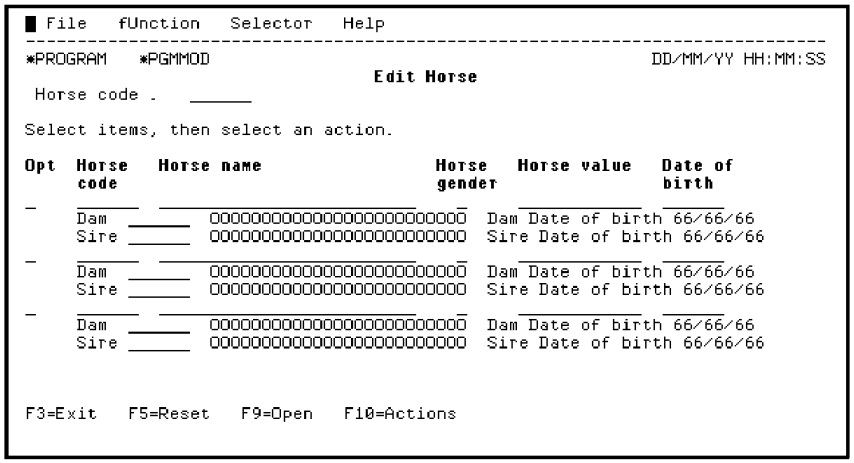


Press F3 to return to the Work With Choices panel.



Exiting the Action Bar Editor

Press F3 to exit the Work with Choices panel and return to the device design.



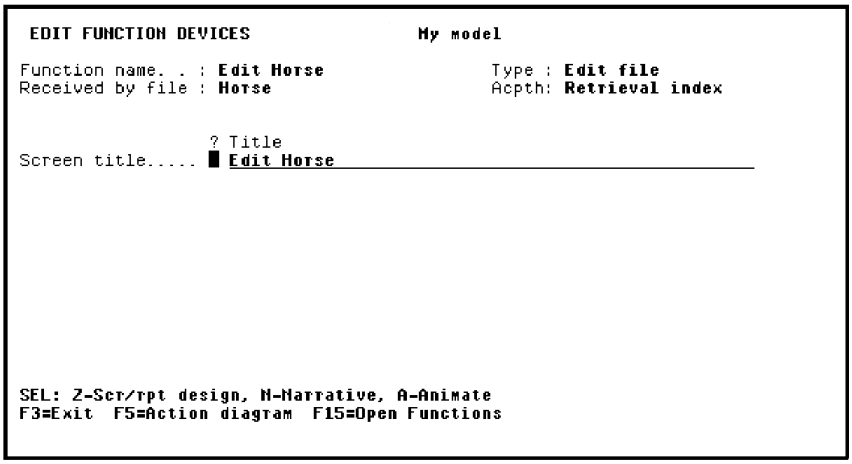
Exiting the Modified Panel

Press F3 to exit the modified device design.

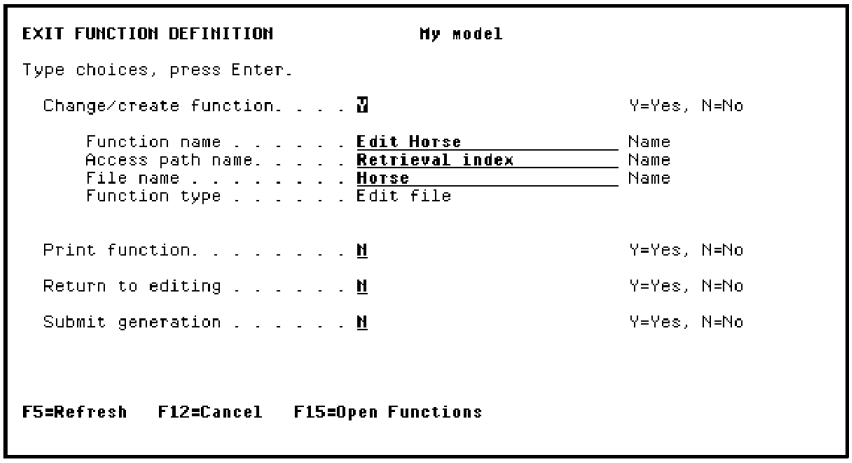
Saving the Modified Panel

After exiting the device design, CA 2E displays the Edit Function Devices panel.

Note: From this panel, you can use Subfile selector option **Z** to return to the device design or you can press F5 to edit the action diagram for the Edit Horse function.



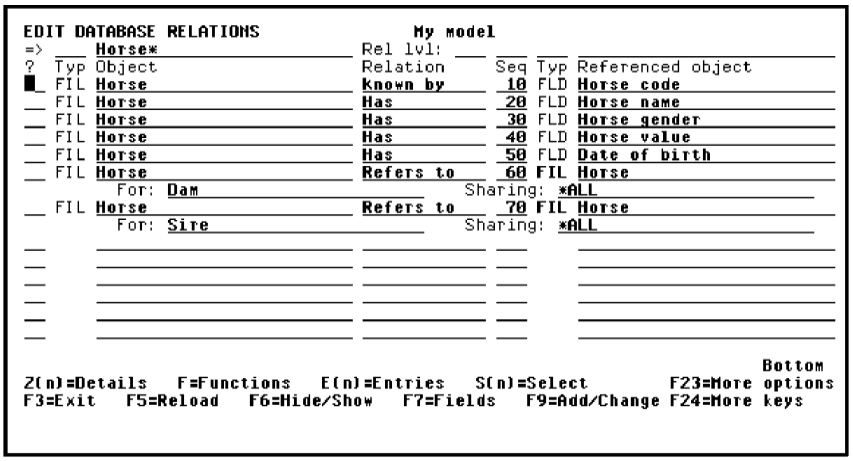
Press F3 to return to the Exit Function Definition panel.



Press Enter to save the device design and return to the Edit Functions panel.

Edit Database Relations Panel

Press F3 to return to the Edit Database Relations panel.



Exercise

Check the device design for the Display Racing results function to improve its appearance and to ensure that the device design does not exceed the size of the panel. The latter prevents errors when you generate the functions in the *Generating, Compiling, and Executing* chapter.

Note: You can access the device design for a function by typing **S** against it on the Edit Functions panel or by pressing F9 from the Edit Function Details panel.

To edit the device design, use the function keys you used to edit the device designs for the Edit Horse and Select Horse functions in the Device Designs topic earlier in this chapter. Press Help or refer to the table in the Device Designs topic for a list of function keys.

Chapter 5: Generating, Compiling, and Executing

This chapter introduces the following topics:

- Implementing Access Paths and Functions
- Executing and Testing Compiled Programs

This section contains the following topics:

Implementing Access Paths and Functions (see page 223)
Executing and Testing Compiled Programs (see page 238)

Implementing Access Paths and Functions

This topic discusses how to generate source to implement the access paths and functions you have created. You will then compile the source to produce executable i OS objects: files and programs.

New terms introduced:

- Job list
- Creation

New displays introduced:

- Display Services Menu
- Submit Model Creates
- Submit Model Generations & Creates
- Convert Condition Values

Objectives

Submit the batch generation of all the access paths and functions you have designed. After generating the source code, each access path and function will be compiled. You will then execute the Edit Horse program.

Overview of Implementation

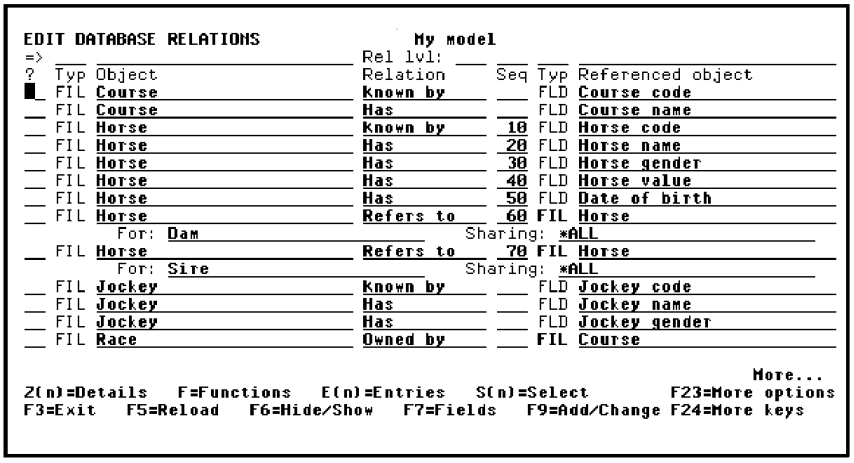
Once your application design is complete, you are ready to generate the source members for the database files, programs, display files, and help text that will implement the access paths and functions. Once generated, the source will need compiling.

You can generate source either interactively or in batch.CA 2E automatically keeps a list of the members to be generated and compiled in batch or to be generated interactively and not yet compiled. In this tutorial, you will generate source code and compile executable objects for all of your access paths and functions as one batch job. To do this, you will use options available on the Display Services Menu.

Source Generation and Compilation of Access Paths and Functions

You must generate and compile the source for your access paths and functions before you can test your application programs. The first step is to indicate which access paths and functions you want generated and compiled. The Display Services Menu includes options to facilitate this process.

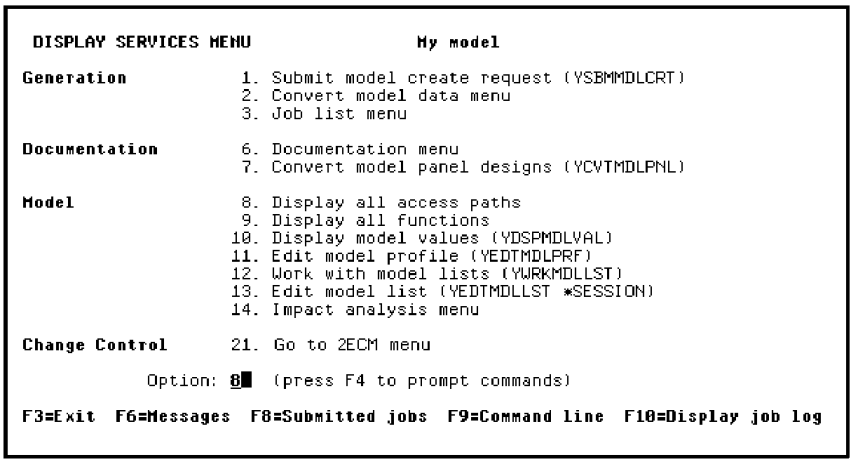
From the Edit Database Relations panel, press F17 to go to the Display Services Menu.



Display all Access Paths for Selection

The Display Services Menu contains an option to display a list of all access paths designed in the design model.

Select the Display all access paths option.

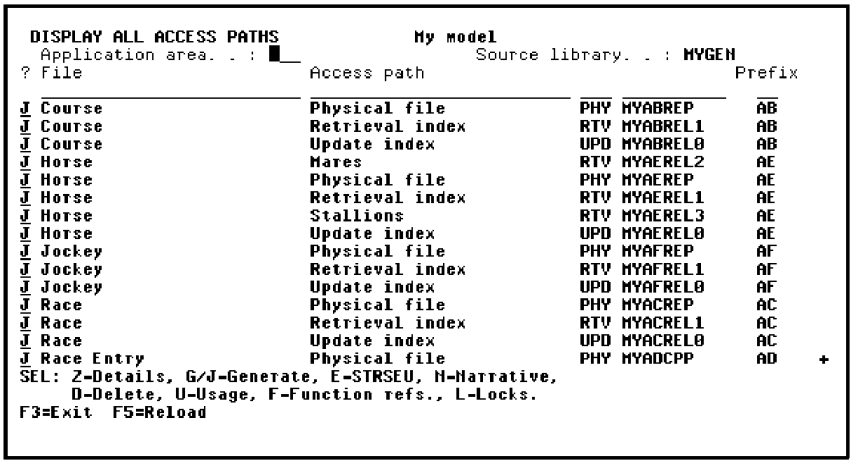


Press Enter.

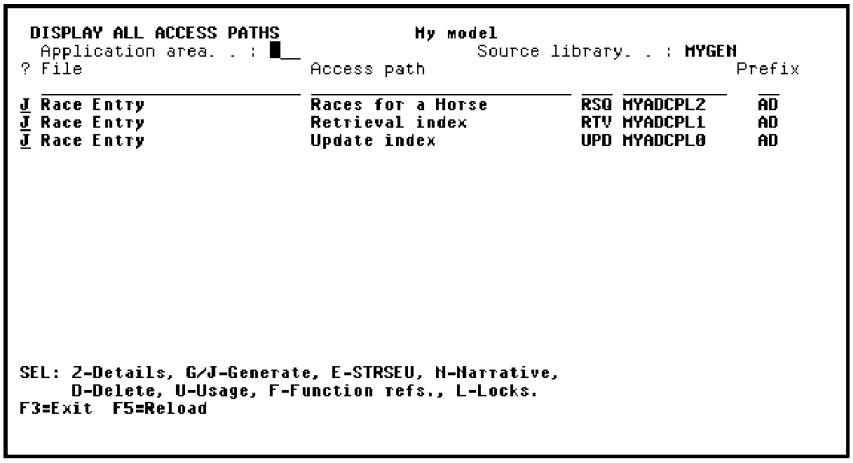
Selecting All Access Paths for Batch Generation and Compilation

The Display All Access Paths panel presents a list of all access paths in the design model. It lets you perform many design, control, and generation tasks for one or more access paths from one panel.

Select all of the existing access path designs for batch generation and compilation by typing a **J** in the Subfile selector next to each access path in the list.



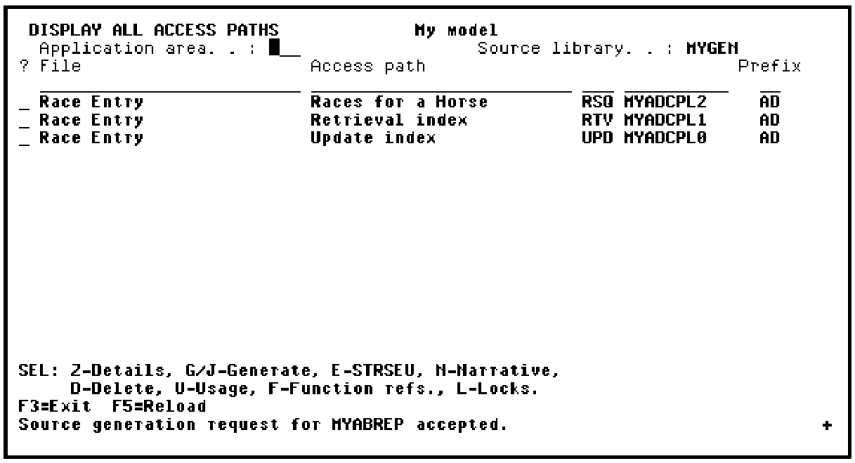
Press Roll Up to display the next screen of access paths. Select all remaining access paths with a **J**.



Press Enter.

Completing the Request

When the process is complete, the Display All Access Paths panel will be redisplayed with messages displayed at the bottom of the panel. The messages will state that the source generation requests have been accepted.

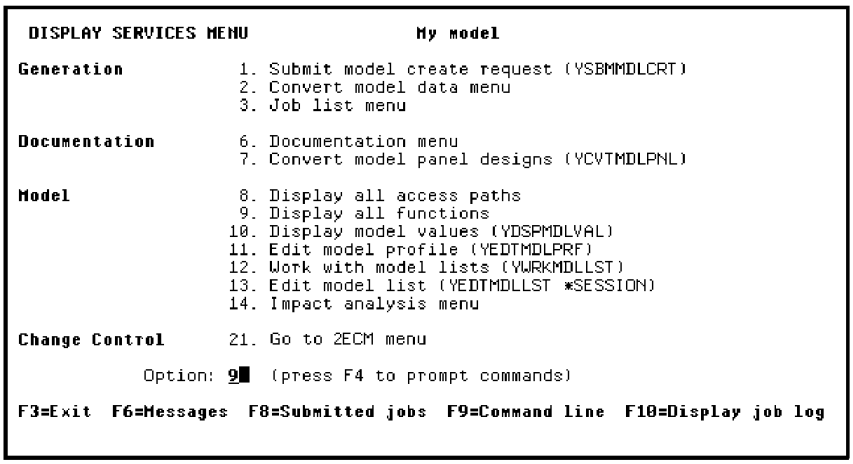


Press F3 to return to the Display Services Menu.

Display all Functions for Selection

In this step you will repeat this process for the functions in your design model. The Display Services Menu includes an option to display a list of the functions in the design model.

Select the Display all functions option.



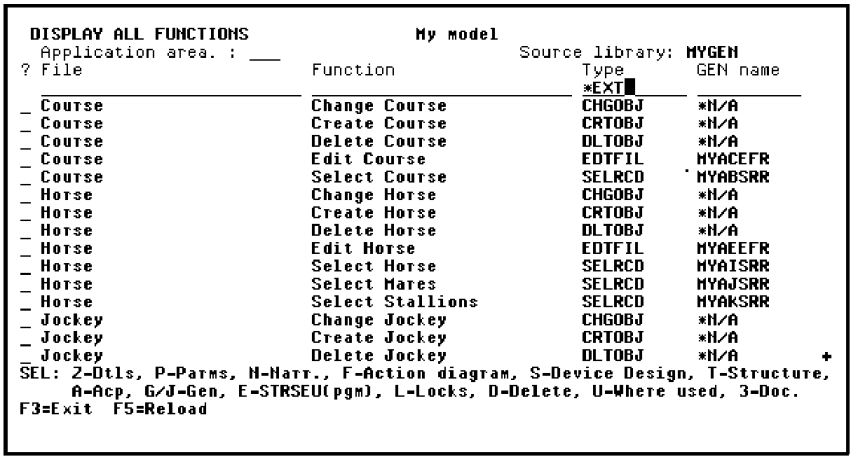
Press Enter.

Selecting All External Functions for Batch Generation and Compilation

The Display All Functions panel displays a list of all functions in the design model. It lets you perform many design, control, and generation tasks for one or more functions from one display.

Only external functions need to be generated and compiled. External functions can be identified by the source member name found in the GEN name column; internal functions show *N/A in this column. Start by displaying just the external functions in your design model.

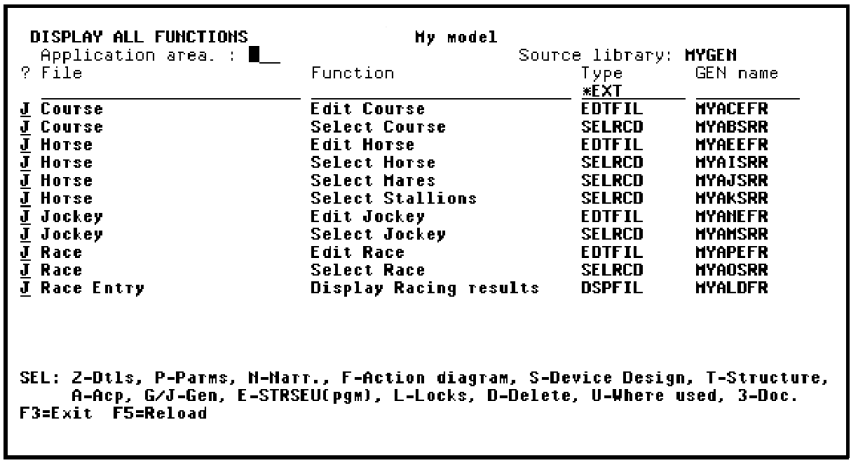
To do so, type ***EXT** in the Type column on the selection line.



Press Enter to redisplay the Display All Functions panel showing only external functions.

Note: Write down the implementation names shown in the GEN name column for the following functions: Edit Course, Edit Horse, and Edit Jockey. You will need these later when you test the compiled programs.

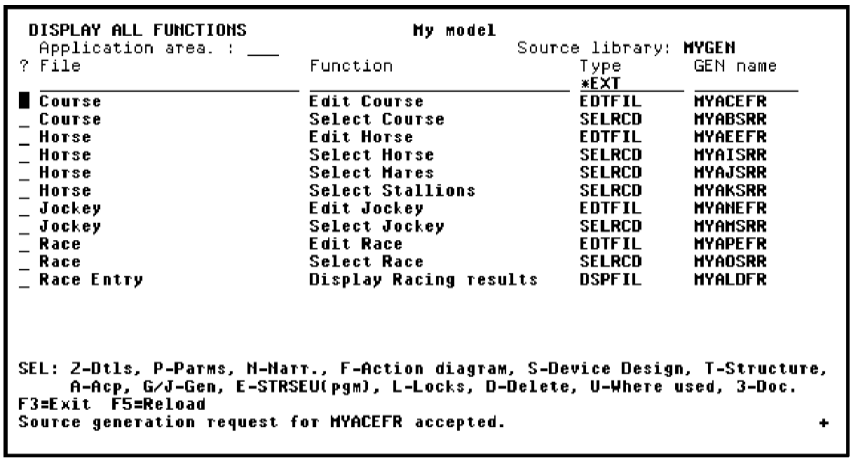
Select all of the external functions for batch generation and compilation by typing a **J** in the Subfile selector next to each function on the list.



Press Enter.

Completing the Request

When the process is complete, the Display All Functions panel will be redisplayed with messages displayed at the bottom of the panel. These messages state that the source generation requests have been accepted.



Press F3 to return to the Display Services Menu.

Submit Batch Generation and Creation

In this step you will use the Display Services Menu to Submit generations and compilations of all the access paths and external functions you have selected.

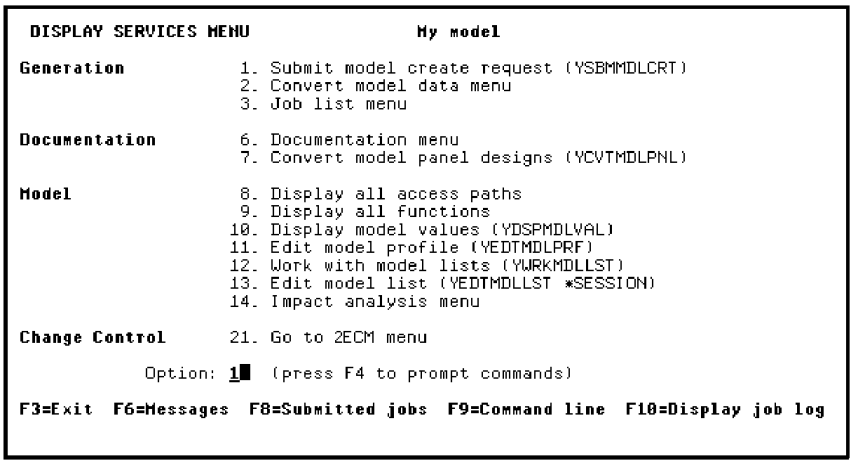
Generating and Creating Objects

When you submit a request for generation/compilation, the CA 2E generator automatically does the following.

- The member names to be generated are placed in a job list. The same job list controls generation and compilation. You can review this job list during generation to monitor the process and edit the job list.
- Source is produced and placed in the appropriate source file in the generation library associated with your CA 2E design model.
- The generated source is compiled from the source file.
- CA 2E assigns a status to each member on the list. Once source is successfully generated for the members,CA 2E automatically submits a request to compile the generated source. If errors occur,CA 2E flags the specific members in error.

You can submit generations and compilations of all the items in the list from the Display Services Menu.

To submit your generation/compilation request, select the Submit model create request (YSBMMDLCRT) option.

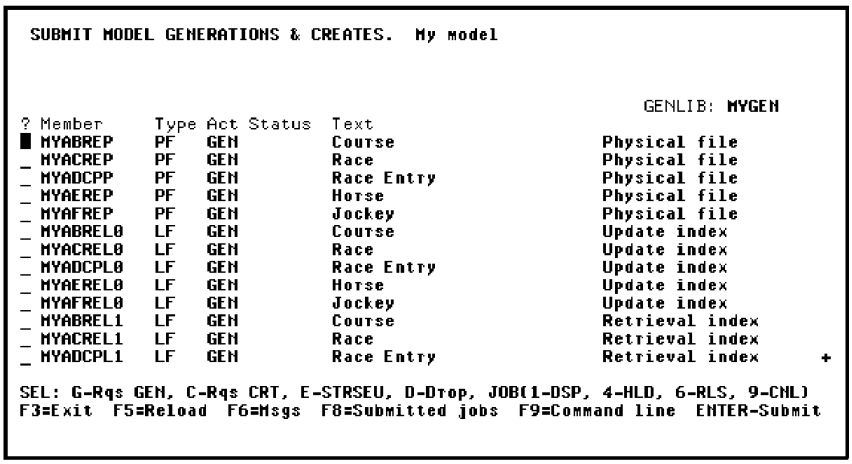


Press Enter to execute the Submit Model Create Requests (YSBMMDLCRT) command.

Note: CA 2E supplies default parameter values for the YSBMMDLCRT command based on your model profile and options you specified when you created your model. You can override these defaults by pressing F4 instead of Enter to prompt the command.

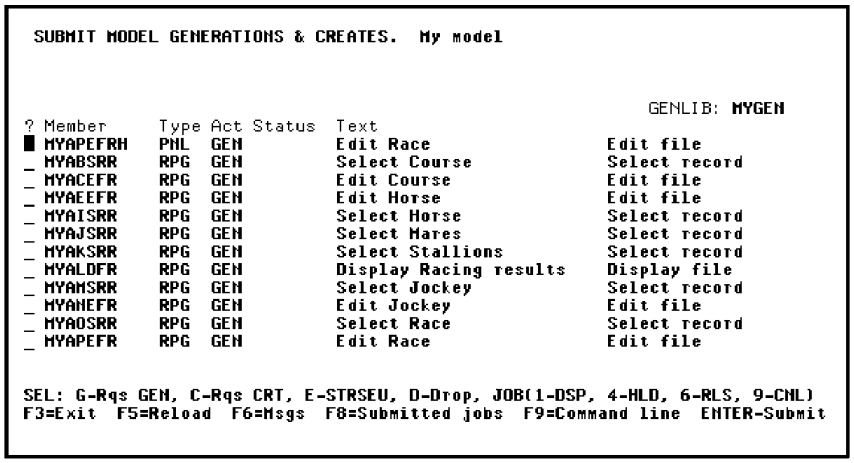
List of Objects to be Generated and Created

CA 2E displays a list of the source members to be generated and compiled. Each member has either GEN or CRT next to it to indicate whether the member has been submitted for generation or compilation. In this case, all the functions and access paths should show GEN. You can review the list by pressing Roll Up and Roll Down.



```
SUBMIT MODEL GENERATIONS & CREATES.  My model

                                                          GENLIB: MYGEN
? Member     Type Act Status  Text
_ MYABREP    PF   GEN         Course                    Physical file
_ MYACREP    PF   GEN         Race                      Physical file
_ MYADCPP    PF   GEN         Race Entry                Physical file
_ MYAEREP    PF   GEN         Horse                     Physical file
_ MYAFREP    PF   GEN         Jockey                    Physical file
_ MYABREL0   LF   GEN         Course                    Update index
_ MYACREL0   LF   GEN         Race                      Update index
_ MYADCPL0   LF   GEN         Race Entry                Update index
_ MYAEREL0   LF   GEN         Horse                     Update index
_ MYAFREL0   LF   GEN         Jockey                    Update index
_ MYABREL1   LF   GEN         Course                    Retrieval index
_ MYACREL1   LF   GEN         Race                      Retrieval index
_ MYADCPL1   LF   GEN         Race Entry                Retrieval index      +

SEL: G-Rqs GEN, C-Rqs CRT, E-STRSEU, D-Drop, JOB(1-DSP, 4-HLD, 6-RLS, 9-CNL)
F3=Exit  F5=Reload  F6=Msgs  F8=Submitted jobs  F9=Command line  ENTER-Submit
```

Press Roll Up until you see the bottom of the list.

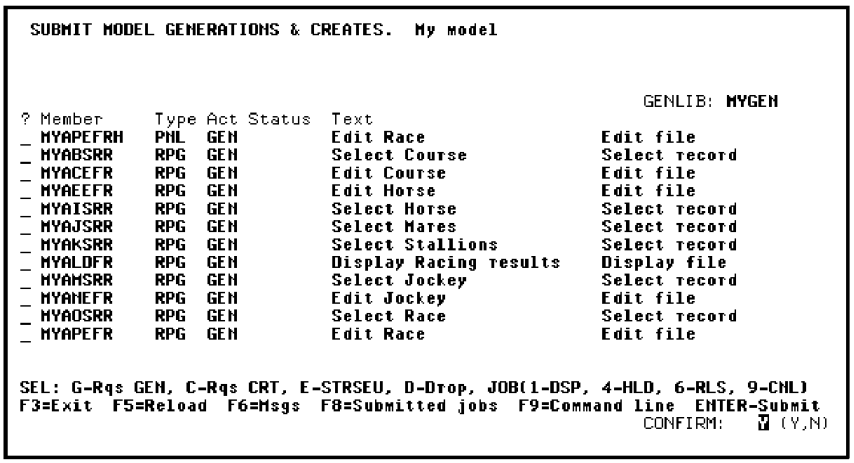


```
SUBMIT MODEL GENERATIONS & CREATES.  My model

                                                          GENLIB: MYGEN
? Member     Type Act Status  Text
_ MYAPEFRH   PNL  GEN         Edit Race                 Edit file
_ MYABSRR    RPG  GEN         Select Course             Select record
_ MYACEFR    RPG  GEN         Edit Course               Edit file
_ MYAEEFR    RPG  GEN         Edit Horse                Edit file
_ MYAISRR    RPG  GEN         Select Horse              Select record
_ MYAJSRR    RPG  GEN         Select Mares              Select record
_ MYAKSRR    RPG  GEN         Select Stallions          Select record
_ MYALDFR    RPG  GEN         Display Racing results    Display file
_ MYAMSRR    RPG  GEN         Select Jockey             Select record
_ MYANEFR    RPG  GEN         Edit Jockey               Edit file
_ MYAOSRR    RPG  GEN         Select Race               Select record
_ MYAPEFR    RPG  GEN         Edit Race                 Edit file

SEL: G-Rqs GEN, C-Rqs CRT, E-STRSEU, D-Drop, JOB(1-DSP, 4-HLD, 6-RLS, 9-CNL)
F3=Exit  F5=Reload  F6=Msgs  F8=Submitted jobs  F9=Command line  ENTER-Submit
```

Press Enter to submit the job list.

Confirming the Job List

The panel will be redisplayed with the option to confirm at the bottom right-hand corner. Accept the default of Y.

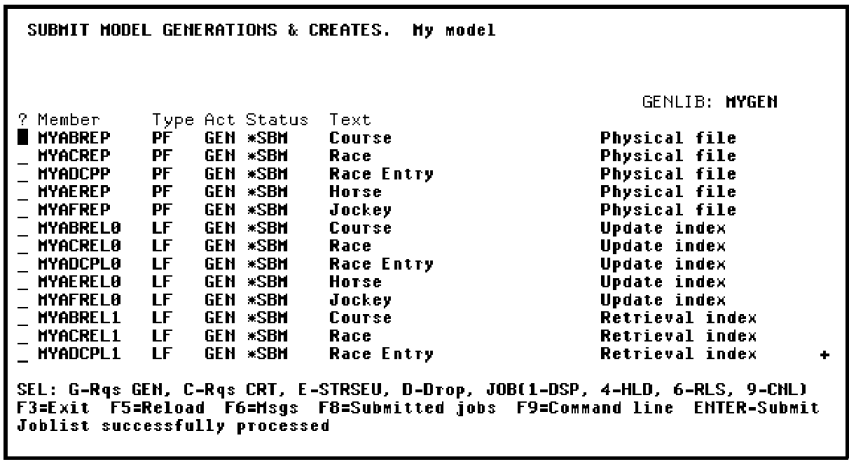


Press Enter.

Successful Submit for Generation and Compilation

CA 2E submits the jobs to generate and/or compile members in this list. If you request batch generation, compilation requests are also submitted by default. Compilation creates the i OS objects once the source has been generated. As a result, compilation is also referred to as *creation*. All generations are carried out in a single job called YGENSRC; a separate compilation job is submitted for each source member.

After confirming the list of objects, you will see a series of messages at the bottom of the panel. These are "Job YGENSRC is being prepared," "Existing objects are being deleted," and finally "Joblist successfully processed." In batch processing, the generations/ compilations take place in the background. You may continue to specify new objects (functions, access paths) while this is happening.



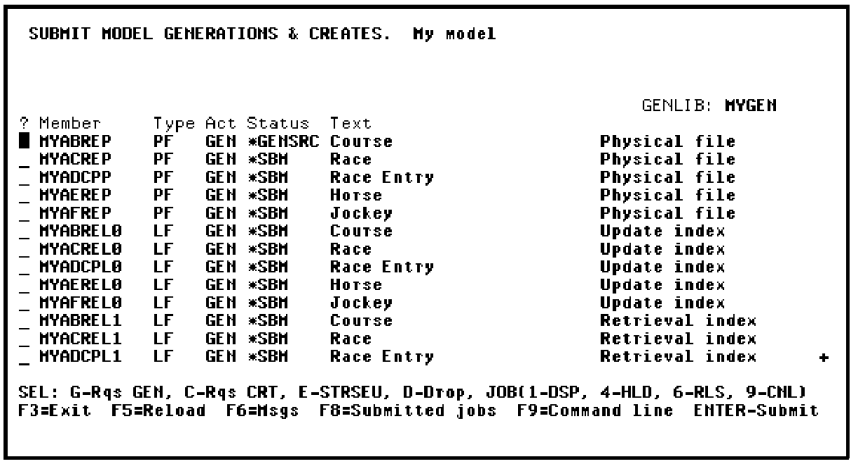
Examining the Job List

Once submitted, each item in the list is initially shown with a status of *SBM. As each item in the list is processed, the status in the list is updated as follows.

Status	Description
*GENSRC	The source member is being generated
*JOBQ	The source is on the job queue to be compiled
*ACTIVE	The source is being compiled

Press F5 to refresh the list and see the latest status.

Once the source members have been successfully compiled, they are removed from the list.



When the compilations have been submitted, press F3 to return to the Display Services Menu. You can check the status of your submitted jobs from the Display Services Menu by pressing F8.

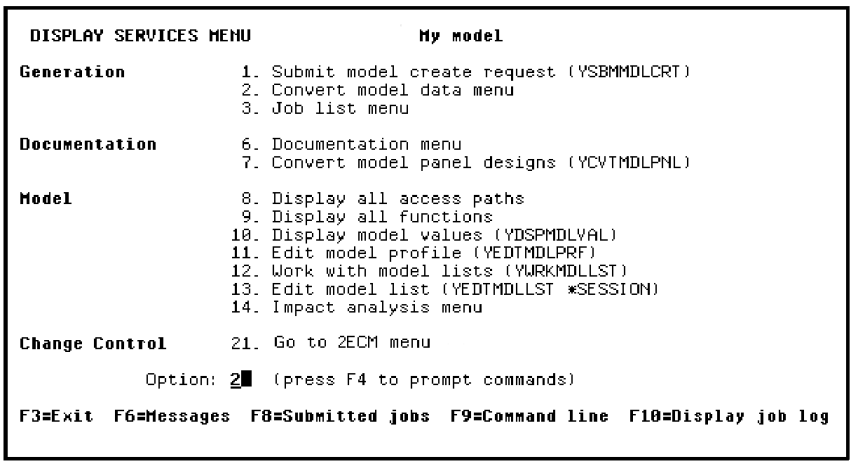
Converting Condition Values to a Database File

Before calling your program, you must convert the values that are entered into status fields using the Convert Condition Values (YCVTCNDVAL) command. This command moves the values you defined for status fields from the model library to the condition values list database file in the generation library.

When you convert condition values to a database file, you give end users the capability to prompt for valid condition values when they use the application. Horse gender is an example of a status field. In this case, the end user will be able to prompt the application and display a list of valid values for Horse gender, namely, M and F.

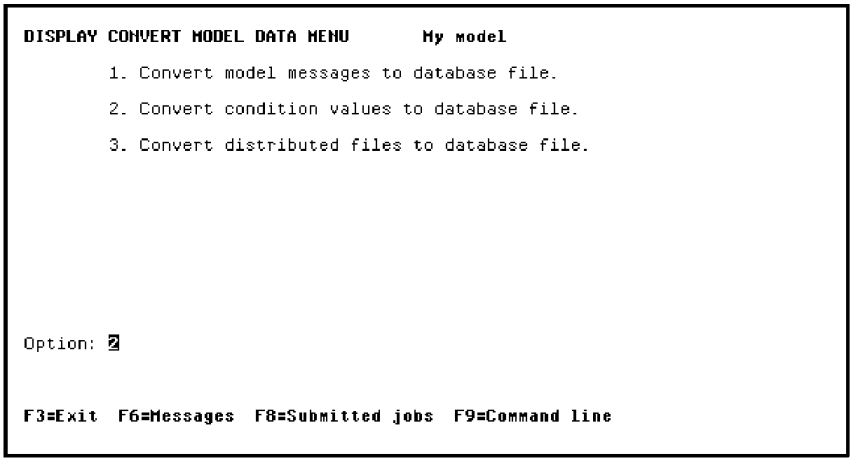
You can run the YCVTCNDVAL command to convert the condition values to a database file using the Convert model data menu, which is an option on the Display Services Menu.

Select the Convert model data menu option.



Press Enter.

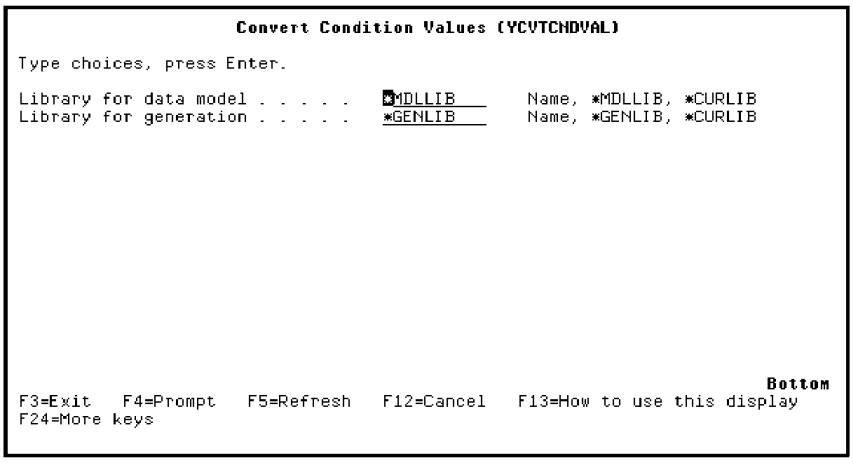
Select the Convert condition values to database file option.



Press Enter to prompt the YCVTCNDVAL command.

YCVTCNDVAL Command Prompt

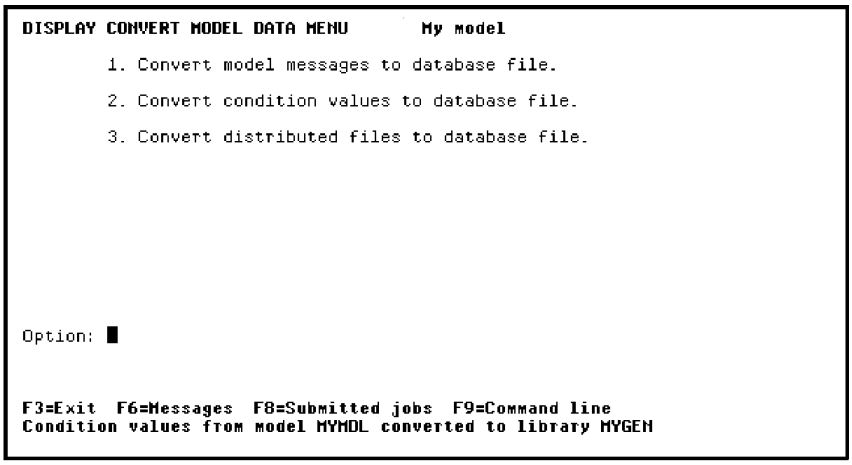
CA 2E supplies default values for the CA 2E design model library and the generated library. Accept the defaults.



Press Enter to execute the YCVTCNDVAL command.

Confirming Conversion of Condition Values

Once the values allowed for the status fields in your model have been converted to the database file, you will be returned to the Display Convert Model Data Menu. A message appears at the bottom of the panel to indicate that the conversion is complete.

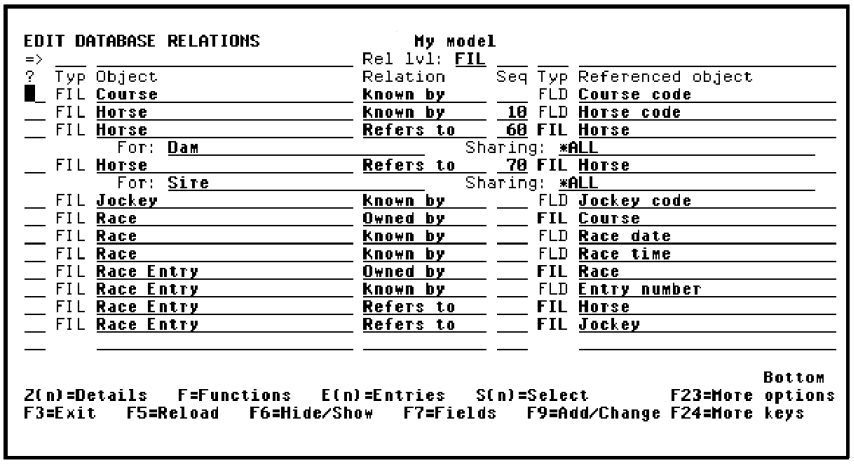


Press F3 twice to return to the Edit Database Relations panel.

Exiting the CA 2E Design Model

When all the objects have been created, your program for editing the HORSE file is complete.

Press F3 to exit your design model.



Re-synchronizing an CA 2E Design Model

The Exit Edit Relations window displays. This window allows three options:

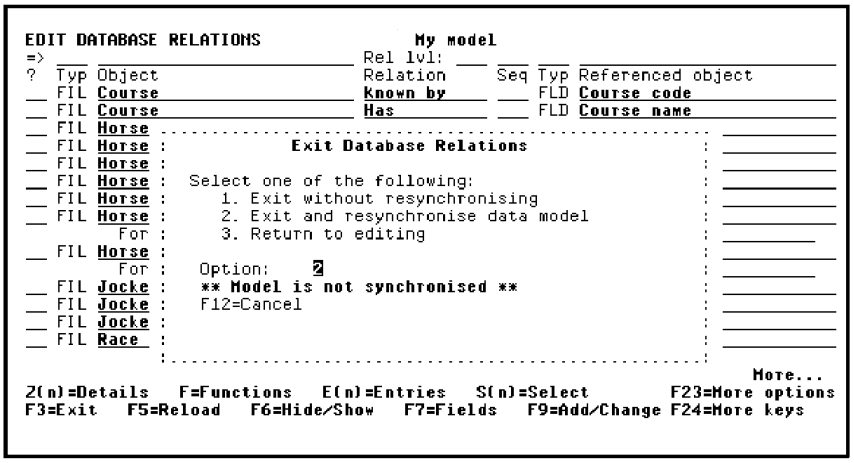
- Exit without re-synchronizing
- Exit and resynchronize data model
- Return to editing

If you have added or changed any files, fields, or relations in the current session, a message will be displayed on this panel, "Model is not resynchronized." In this case, the Exit and resynchronize data model option will be the default. If you select this option, the message "Data model is being resynchronized" will appear at the bottom of the panel.

The re-synchronizing process causes all the relations to be resolved into entries. Only users of type *DSNR can resynchronize a model.

Note: If you have previously exited your design model, your model may already be resynchronized. In this case, the Exit Edit Relations panel defaults to Exit without re-synchronizing.

Accept the default (option 1 or 2).



Press Enter to exit your design model.

Executing and Testing Compiled Programs

This topic explains how to call your compiled applications (programs) and test them by entering data.

Calling the Program

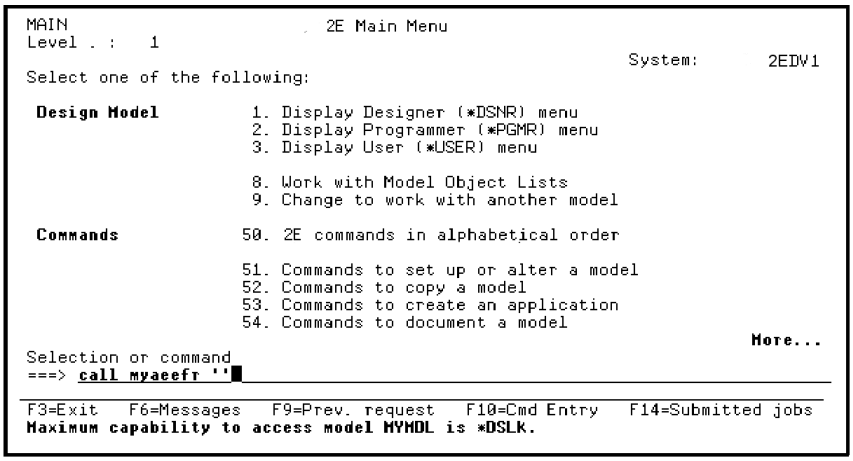
Once the compilations you have submitted have completed successfully you can test the Edit Horse and Edit Course programs. A message is sent to your message queue when each compilation finishes.

Call your Edit Horse program from any command entry line by using the i OS CALL command to invoke your program, for example, MYAEEFR, as shown below. From within your model, you can display a command entry line by pressing F9 from the Display Services menu.

Note: If you do not know the name of your Edit Horse program, you can obtain it from the Edit Function Details panel or the Display all functions option on the Display Services Menu.

When you call your program you need to specify a dummy *Return code parameter. All CA 2E programs require a dummy parameter to be passed to the program (‘ ‘ or ‘’). This return code can be used to communicate between programs in more complex applications.

Type the program name that corresponds to your Edit Horse function and return code.



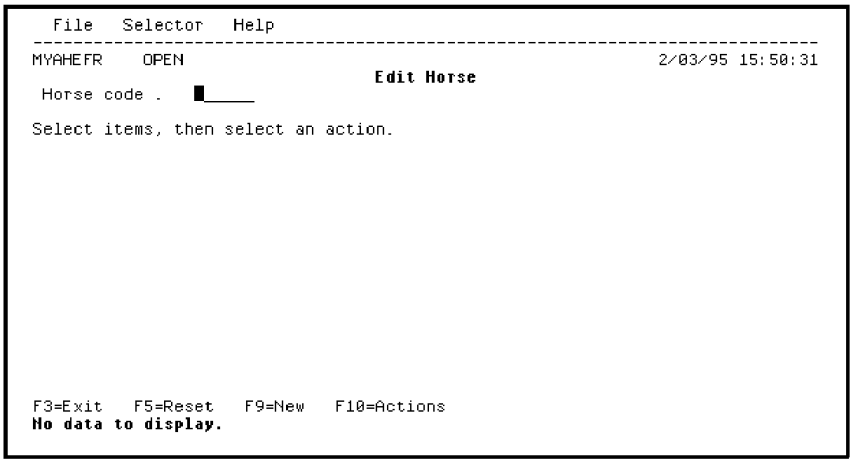
Press Enter.

Note: To test complex programs, you can use the CA 2E Call a Program (Y2CALL) command. This command loads your model and determines the parameters required by an external function directly from details contained in the model. You can provide values for all input-capable fields and you can re-use these values for subsequent calls. You can also retrieve and display output parameters when the called program ends.

The Edit Horse Function Panel

An empty panel will be displayed. The Edit File function operates in two alternate modes, one to add new records (New mode) and one to change existing records (Open mode). Use the F9 function key to toggle between the two modes.

Note that the program mode is Open. Since there are no existing records to update, you first need to change to New mode to add records to the file.



Press F9 to switch to New mode to add data to the HORSE file.

Adding Data to the HORSE File

An empty input panel is displayed. Note that the program mode is New. This is where you may add new data. The panel should have the design you created and data should be validated according to the rules you specified in the design. Test this out by entering invalid values. Also try pressing F4 for the Horse gender field to display a selection list of available values.

To add data to the database file, type horse details.

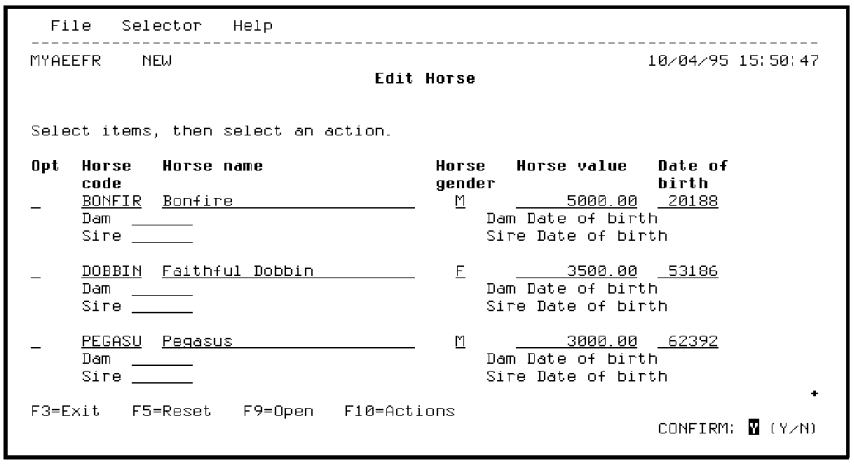
```
  File   Selector   Help
--------------------------------------------------------------------------------
MYAEEFR    NEW                                          10/04/95 15:47:14
                                  Edit Horse

Select items, then select an action.

Opt  Horse code  Horse name                 Horse gender  Horse value  Date of birth
_    BONFIR      Bonfire                   M             5000         020188
     Dam ______                               Dam Date of birth
     Sire ______                              Sire Date of birth

_    DOBBIN      Faithful Dobbin           F             3500         053186
     Dam ______                               Dam Date of birth
     Sire ______                              Sire Date of birth

_    PEGASU      Pegasus                   M             3000         062392
     Dam ______                               Dam Date of birth
     Sire ______                              Sire Date of birth
                                                                            +
F3=Exit   F5=Reset   F9=Open   F10=Actions
```

Press Enter.

Confirming Data Entries for the Horse File

You will be prompted with a Confirm prompt before the update is completed. As you saw in the function option topic of this tutorial, the confirm prompt is an optional feature of a function. You could omit it from the design if desired.

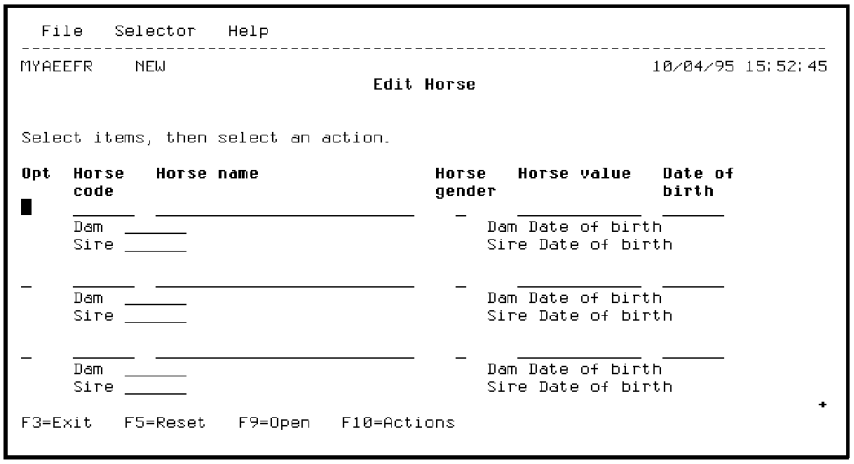
Earlier in the tutorial, in the Function Options topic, you set the initial value of the Confirm prompt to Y. Verify that the initial value of the Confirm prompt is Y.



Press Enter.

Note that once added, the key fields cannot be changed except by deletion.

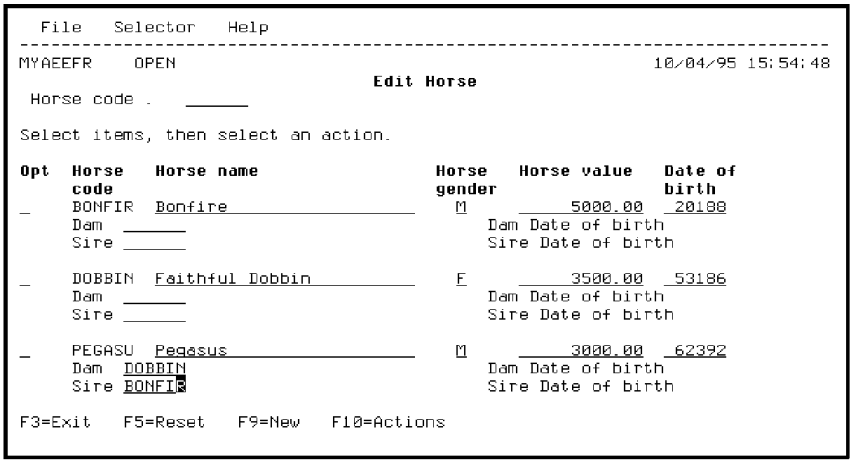
After the new data has been successfully added, another empty subfile page is displayed, ready for more entries.



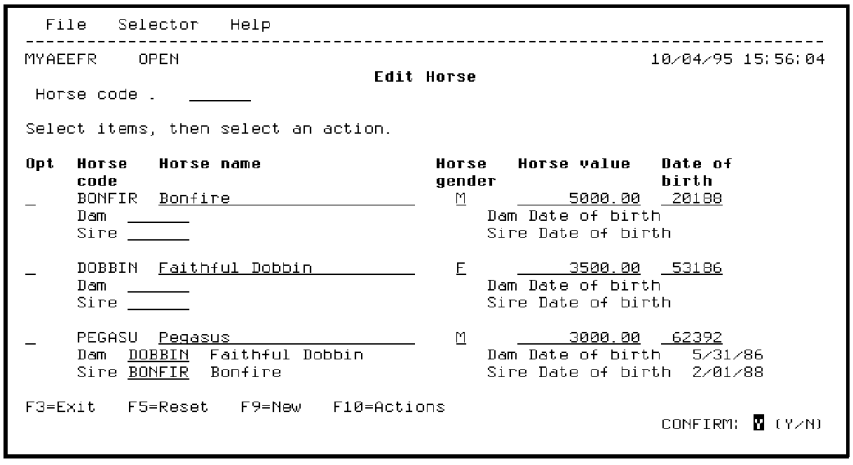
Switching from New to Open Mode

Press F9 to switch to Open mode to view the records you just added to the database.

Now that you have several horses in your database, you can make them parents. Type the details of the Dam and Sire of Pegasus.



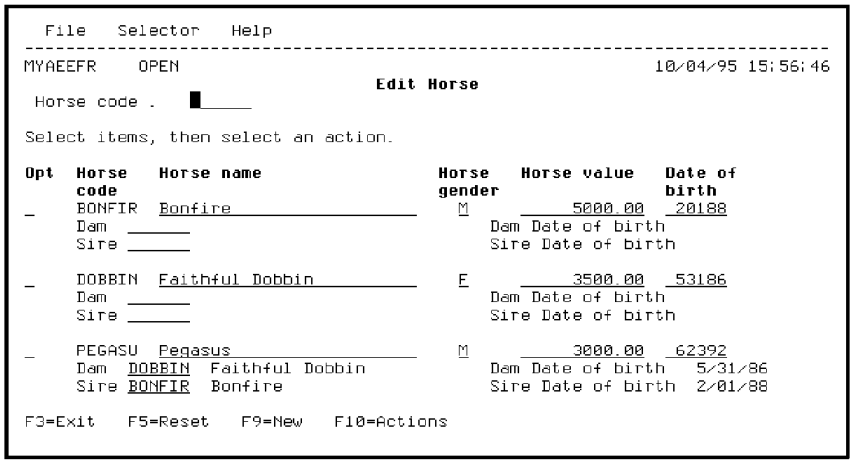
Press Enter to retrieve the values for the four virtual fields you added, Dam name, Sire name, Dam Date of birth, and Sire Date of birth.



Press Enter to accept the default value of Y for the Confirm prompt.

Exiting the Edit Horse Program

Exit your program by pressing F3.



Exercises

Do the following exercises in the order shown. If you did not write down the program names you can obtain them from the Edit Function Details panel or the Display all functions option on the Display Services Menu.

1. Call the Edit Course program and add some race courses.
2. Call the Edit Jockey program to add some jockeys.

Test the Select Stallions and Select Mares functions, by calling the Edit Horse program and typing **?** or pressing F4 in the Sire and Dam fields.

Chapter 6: Maintaining Your Application

This chapter introduces the following topics:

- Application Maintenance
- Prototyping (Animation)
- Model Object List Processing
- Function Versioning
- Model Object Cross References
- Impact Analysis

Note: These topics are not needed to complete the tutorial. In addition, some of the concepts covered are advanced and may be more suitable after you complete the *Advanced Functions* and *Report Functions* chapters. As a result you can safely skip them now and return to them after completing the *Report Functions* chapter.

This section contains the following topics:

Application Maintenance (see page 245)
Animating an Interactive Device Design (see page 246)
Working with Model Object Lists (see page 266)
Function Versioning (see page 288)
Model Object Cross References (see page 313)
Impact Analysis (see page 324)

Application Maintenance

The life of an application does not end once development is complete. Over time, the business requirements may change or users may request additional functionality. These application updates can be accomplished easily by applying the same principles and procedures that have been discussed so far.

For example, if you change the length of a field,CA 2E automatically modifies all files and functions that use the changed field, including associated panels and reports. You will need to examine the panel and report designs to ensure that they have not exceeded their limits, and then regenerate the files and functions. The Submit model create process provides options to save data prior to generation and compile.

To aid you in managing changes and identifying the impact of a change on your model,CA 2E provides a set of powerful tools that are the subject of the rest of this chapter.

Animating an Interactive Device Design

Before making a change, you can test and demonstrate the proposed device design using a process called *prototyping* or *animation*. In this topic you will use CA 2E animation to simulate the link between the Edit Horse function and the Display Racing results function.

New terms introduced

- Animation
- Prototyping
- CA 2E Toolkit

New panels introduced

- Animate Function Panels
- Open Functions
- Toolkit Work with Panel Title Details
- Toolkit Work with Panel Command Key Usage
- Toolkit Edit Choices
- Toolkit Edit Actions

Overview of CA 2E Animation

CA 2E animation provides a direct link between CA 2E and the Toolkit prototyping facilities. *Toolkit* is a set of implementation, support, and system utilities that include menu design, panel design, prototyping, documentation aids, and object list processing.CA 2E animation lets you transfer control to Toolkit, where you can interactively simulate your CA 2E panel designs, and then easily return to CA 2E.

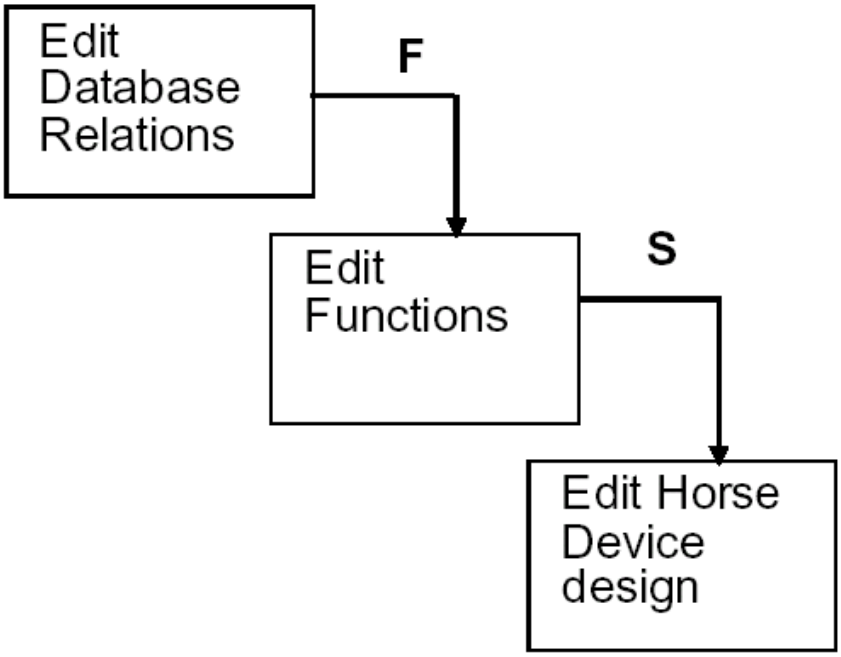
You can animate a function before you make any changes to its default action diagram. This lets you thoroughly test your device design before you commit to specific programming logic, which is more difficult and time-consuming to change. In addition, you do not need to generate and compile a function before you animate it.

For example, you can use animation to demonstrate a proposed design to end users for their review and approval. You can immediately implement any suggestions for improvement of the device design using the CA 2E Device Design Editor and then return to Toolkit to demonstrate the updated design. You can repeat this process until you are satisfied with the design. At that point, you can begin editing the action diagram to implement your design.

Animate the Edit Horse Device Design

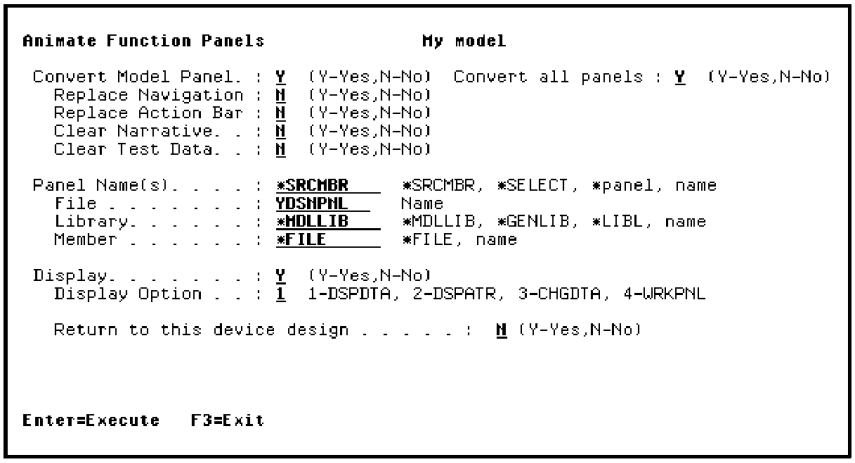
This step demonstrates the process required to animate the link between the Edit Horse and Display Racing results functions.

Begin by displaying the device design for the Edit Horse function. Here's a reminder.




```
  File   fUnction   Selector   Help
-------------------------------------------------------------------------------
*PROGRAM   *PGMMOD                                          DD/MM/YY HH:MM:SS
                                  Edit Horse
 Horse code .   ______

Select items, then select an action.

Opt  Horse    Horse name                  Horse    Horse value     Date of
     code                                 gender                   birth
_    ______   ____________________________ _       ______________  ______
     Dam  ______  00000000000000000000000000  Dam Date of birth  66/66/66
     Sire ______  00000000000000000000000000  Sire Date of birth 66/66/66
_    ______   ____________________________ _       ______________  ______
     Dam  ______  00000000000000000000000000  Dam Date of birth  66/66/66
     Sire ______  00000000000000000000000000  Sire Date of birth 66/66/66
_    ______   ____________________________ _       ______________  ______
     Dam  ______  00000000000000000000000000  Dam Date of birth  66/66/66
     Sire ______  00000000000000000000000000  Sire Date of birth 66/66/66


F3=Exit   F5=Reset   F9=Open   F10=Actions
```

Press F2 to display the Animate Function Panels panel.



Animate Functions Panel

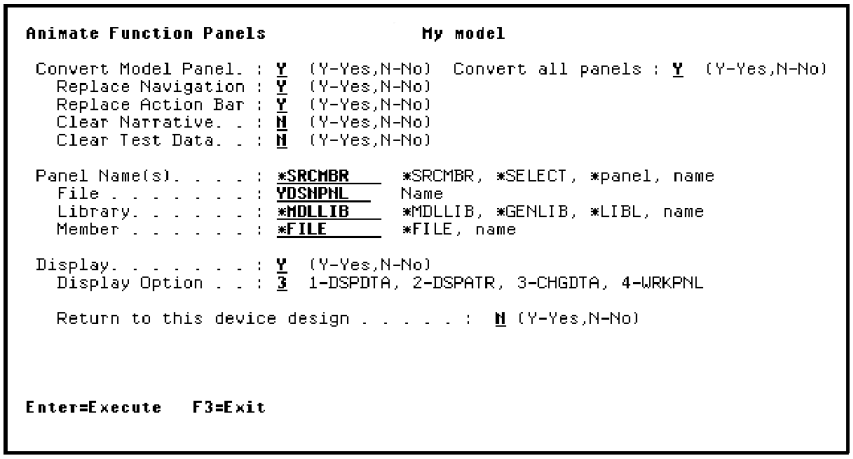
This panel is the link between CA 2E and Toolkit. It lets you convert a CA 2E device design to a Toolkit panel design and optionally transfer control to Toolkit. Use options on this panel to specify the tasks you want to accomplish in Toolkit; for example, you can

- Display and work with Toolkit panel designs
- View field and display attributes
- Enter sample data
- Animate panels, windows, and action bars from the current CA 2E device design
- Demonstrate and test navigation using function keys, subfile selection, and action bars

Animating Edit Horse

In this step you will convert the Edit Horse device design to a Toolkit panel design by accepting the default of Y for the Convert Model Panel field. You also need to convert the CA 2E command key and action bar navigation so you can simulate the link between the two functions. To enter sample data, you need to change the Display Option field to 3 (CHGDTA).

Type **Y** for Replace Navigation, **Y** for Replace Action Bar, and **3** for the Display Option field.



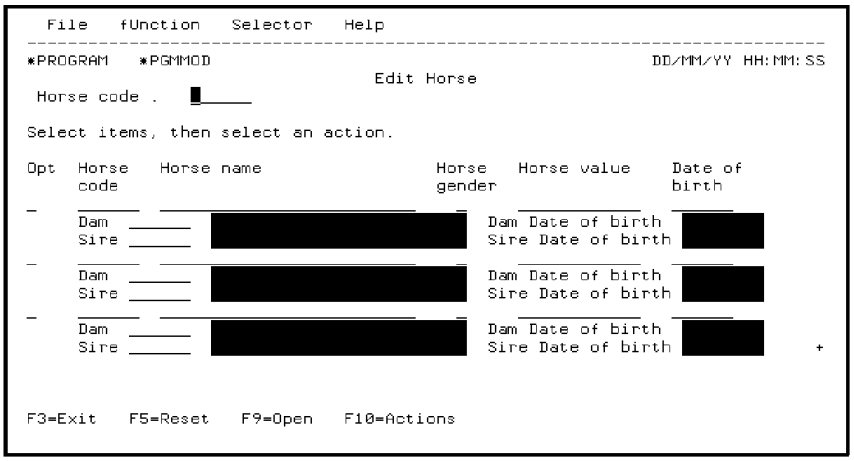
Press Enter to display the Toolkit panel design corresponding to your CA 2E device design. Notice the messages at the bottom of the panel as CA 2E converts the device design and the function key and action bar navigation.

Entering Sample Data for Edit Horse

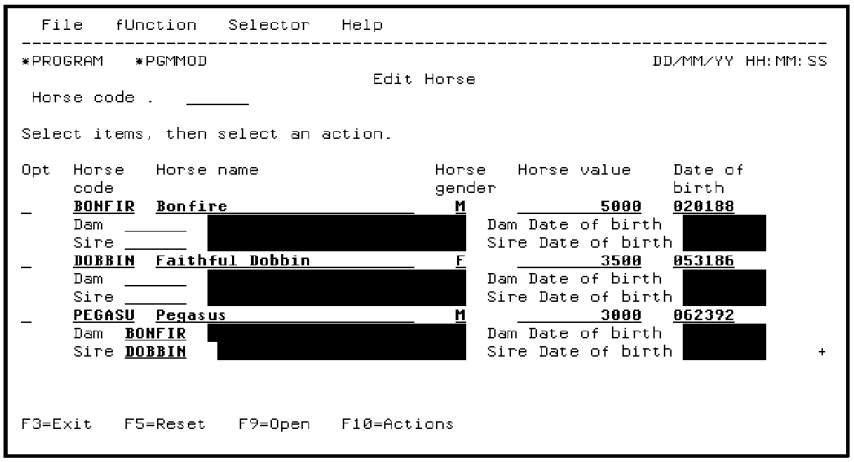
You can now enter sample data in the input-capable fields. These are indicated below by underscores. Output-only fields are shown in black.

Note: The Toolkit panel design reflects the colors or other display attributes you included in your CA 2E

device design.



Type the following sample data. Use the Tab and Field Exit keys as you would on a real panel.

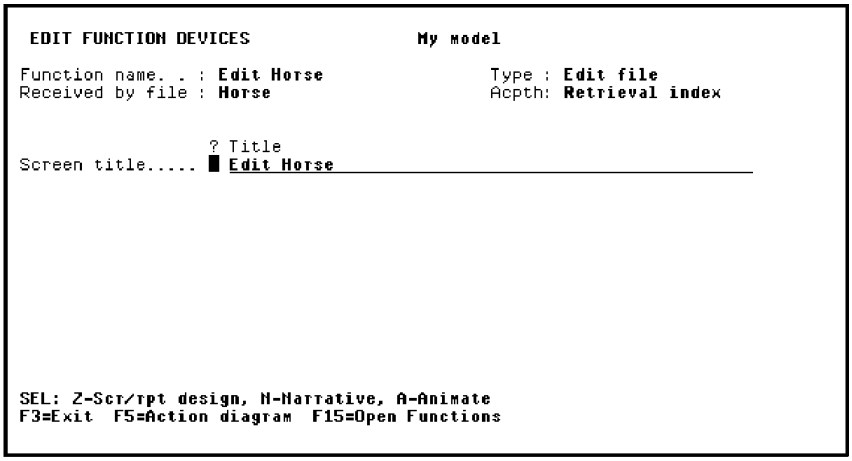


Press Enter to confirm your entries. This data will be retained until you choose to Clear Test Data on the Animate Function Panels panel. Press F3 to return to the CA 2E device design for Edit Horse.

Editing and Maintaining Multiple Functions

In the next step you will animate the Display Racing results function using the same process you just used for the Edit Horse function. Since you will need to access the Edit Horse function again later in this process, it will save time not to exit and close it while you animate the Display Racing results function. The Open Functions panel lets you maintain several functions simultaneously.

From the Edit Horse device design, press F3 to exit to the Edit Function Devices panel.

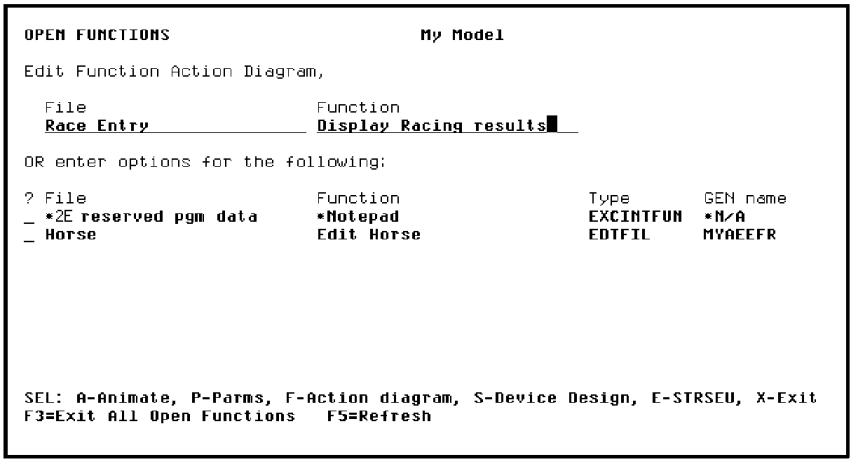


Open Functions Panel

Press F15 to access the Open Functions panel. You can access this panel from many CA 2E panels; for example, option O on the Edit Functions panel lets you open multiple functions. This panel lets you open, edit, and maintain several functions simultaneously; in other words, you do not need to exit and close one function before you open another function for editing. This capability can save a significant amount of time. You can switch quickly and easily between the action diagrams, device designs, generated source, and parameter definitions for all open functions.

Note that the Edit Horse function is listed on the Open Functions panel since you have not yet exited and closed it. To open another function, type the name of the file and the name of the function in the File and Function fields. If you are uncertain of the names, type **?** in these fields instead to display selection lists. You can also type * for the File field if the function you wish to open is attached to the same file as the first open function listed, in this example, the HORSE file.

In this case, you want to open the Display Racing results function on the RACE ENTRY file, type the following names:

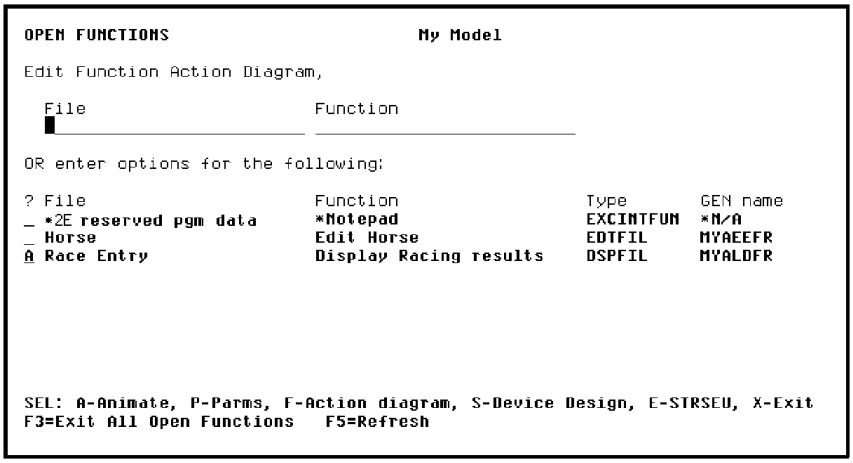


Press Enter to open the Display Racing results function and display its action diagram. Press F15 to return to the Open Functions panel.

Note that Display Racing results now appears on the list. From this panel you can edit a function's device design, action diagram, or parameters; you can also animate the function.

Animating the Display Racing Results Function

Type **A** in the Subfile selector for the Display Racing results function as shown to animate this function.



Press Enter to display the Animate Function Panels panel.

Converting Command Key Navigation

In this step you will convert both the device design and the command keys defined for the CA 2E device design, and edit the command key assignments in Toolkit.

To accomplish this, type **Y** for Replace Navigation and type **4** (WRKPNL) for the Display Option.

Press Enter. The CA 2E device design for the Display Racing results function is converted to a Toolkit panel design and control is transferred to Toolkit.

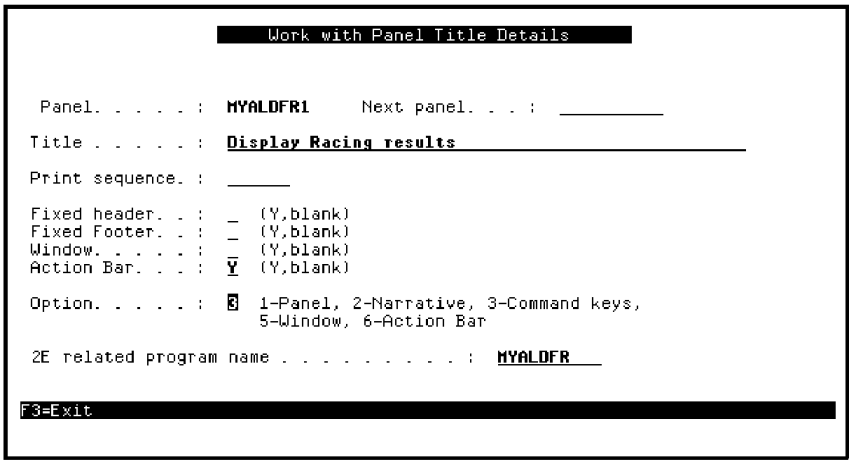
Working with Toolkit Panel Designs

The Toolkit Work with Panel Titles panel displays. You can use this panel to edit the Toolkit panel design, including the panel layout, command keys, and action bar. Note the following on this panel.

The CA 2E related program name at the bottom of the panel; in this case, MYALDFR. This is the program implementation name of the CA 2E function corresponding to this Toolkit panel design. It serves as the link back to the CA 2E function. If the function is not already open on return to CA 2E, it is automatically added to the Open Functions panel before its device design is displayed.

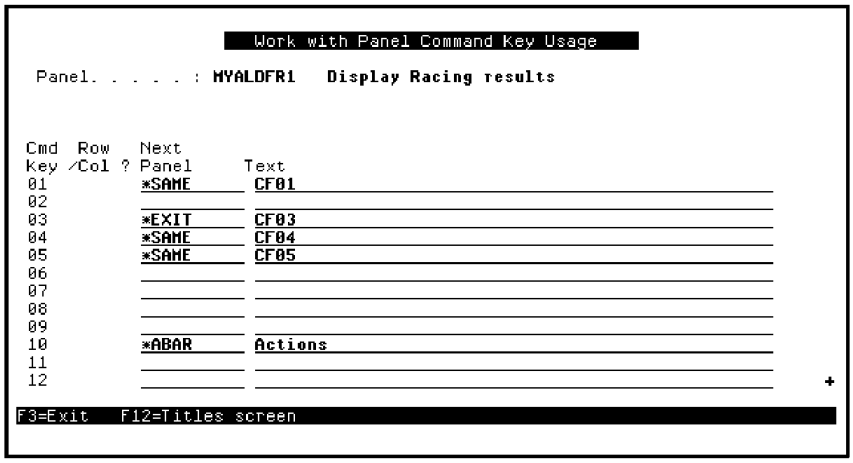
The name of the panel design; in this case, MYALDFR1. By default this is the CA 2E program implementation name followed by 1. You will need this name when you set up the link in the Edit Horse Toolkit panel design.

Because you requested that command key navigation be replaced during the conversion, type **3** for the Option field as shown to view the results of the command key conversion.



Toolkit Command Key Navigation

This panel shows which panel is to be displayed when a command key is pressed during program simulation in Toolkit. Entries are shown only for command keys defined for the function in CA 2E.



The possible values for the Next Panel column are:

Value	Description
*PRV	Display the previous panel
*SAME	Redisplay the current panel
*EXEC	Execute the command string shown in the Text column
*EXIT	Exit the program
*ABAR	Activate the action bar
name	Display the named Toolkit panel design

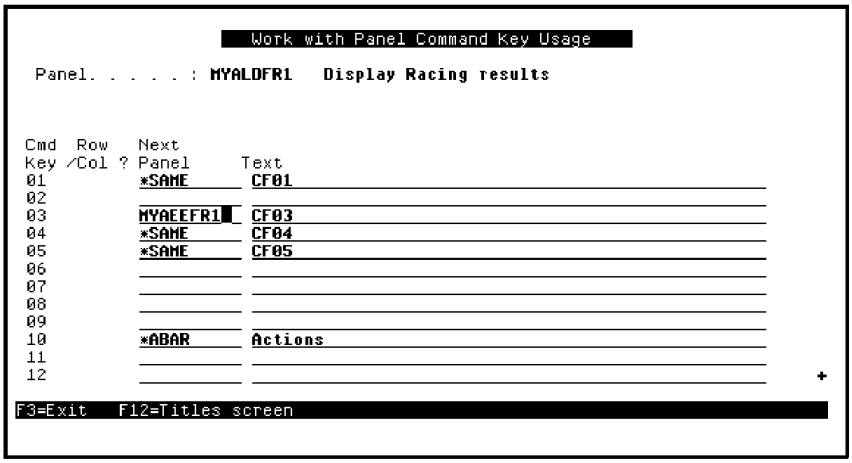
If you had not converted the command key navigation when you converted the Display Racing results function, only the F3 command key would be defined. This is done by default to provide a way to return to CA 2E that you can easily remember. You can also return to CA 2E by pressing the Home key.

Note: Refer to the documentation for your terminal or computer to learn which key is the Home key on your system.

Reassigning a Command Key in Toolkit

Since the Display Racing results function is called from the Edit Horse function, the F3 command key should return to Edit Horse. Type the name of the Toolkit panel design that corresponds to the Edit Horse function in the Next Panel column for the F3 command key as shown. By default this is the function's implementation name followed by a 1; for example, MYAEEFR1.

Type the Toolkit panel design name that corresponds to your Edit Horse function. You can also type a **?** to select from a list of Toolkit panel designs.



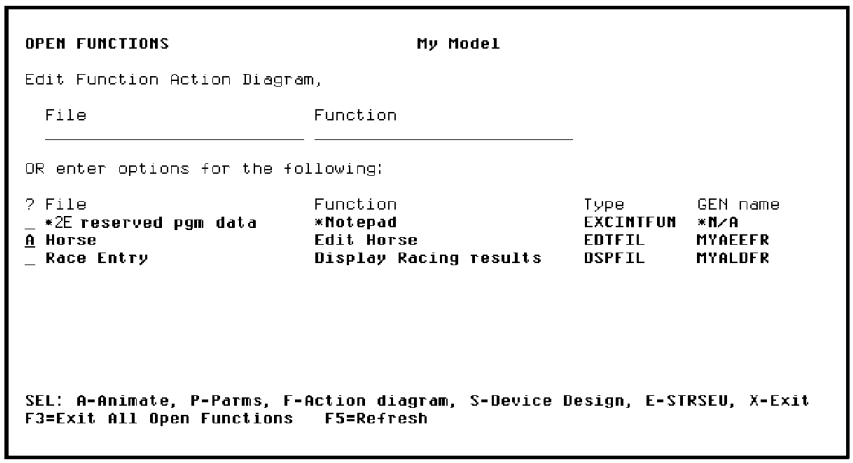
Press Enter to confirm your changes. Press F3 to exit.

Press Enter to update the panel design and return to the CA 2E Open Functions panel.

Setting Up Action Bar Navigation for Edit Horse

In this step you will set up the command key and action bar navigation for the Edit Horse function so you can simulate the link between the Edit Horse and the Display Racing results functions using Toolkit.

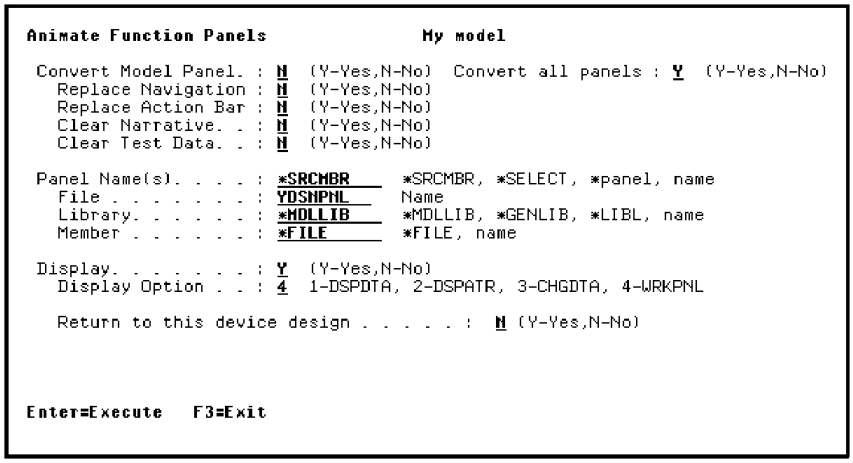
Type **A** in the Subfile selector to animate the Edit Horse function.



Press Enter.

Because you have already converted this panel design, including the command key and action bar navigation, you can change the value of the Convert Model Panel to **N**.

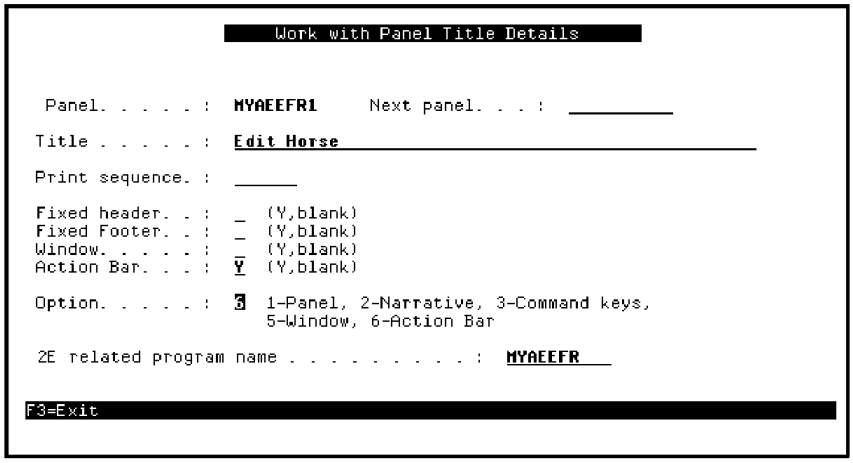
Type **N** for Convert Model Panel and change the Display Option to **4** (WRKPNL) to work with the Toolkit panel design.



Press Enter.

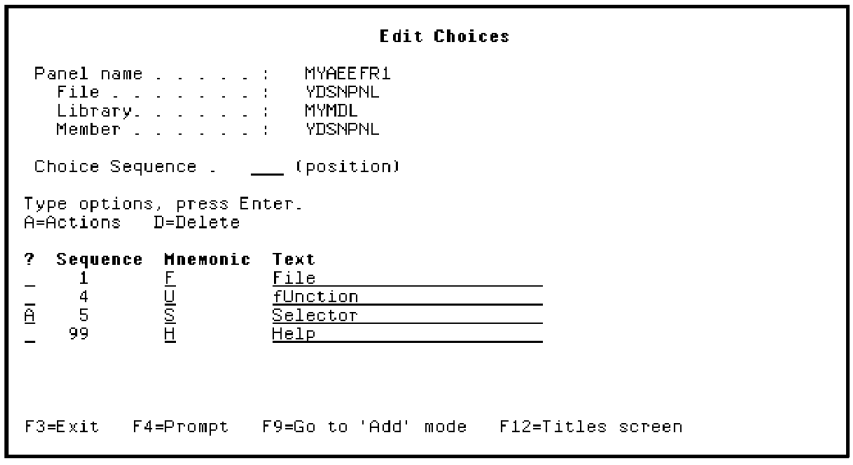
Assigning Action Bar Navigation

You need to change the action bar navigation so the Display Racing results Toolkit panel design displays when you select it from the Selector Choice menu on the Edit Horse panel design. Type **6** in the Option field to set up the action bar navigation.



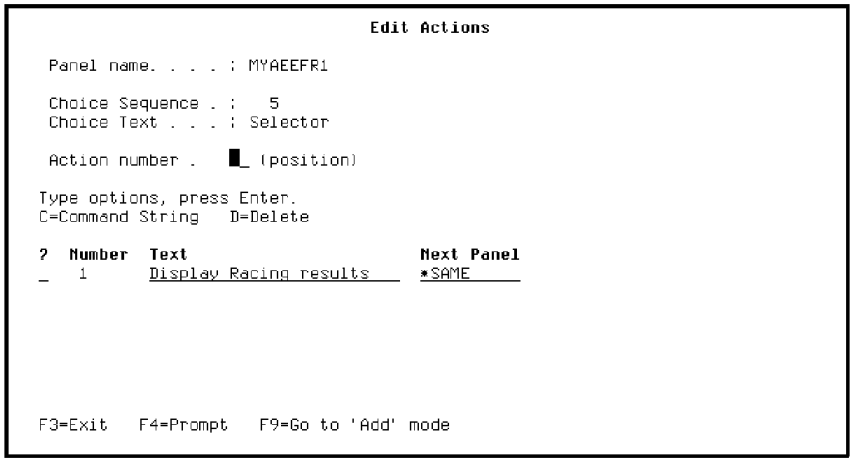
The Toolkit Edit Choices panel shows the action bar choices that were defined on the CA 2E device design. You need to edit actions for the Selector Choice menu.

Type **A** to edit the actions for the Selector Choice menu.

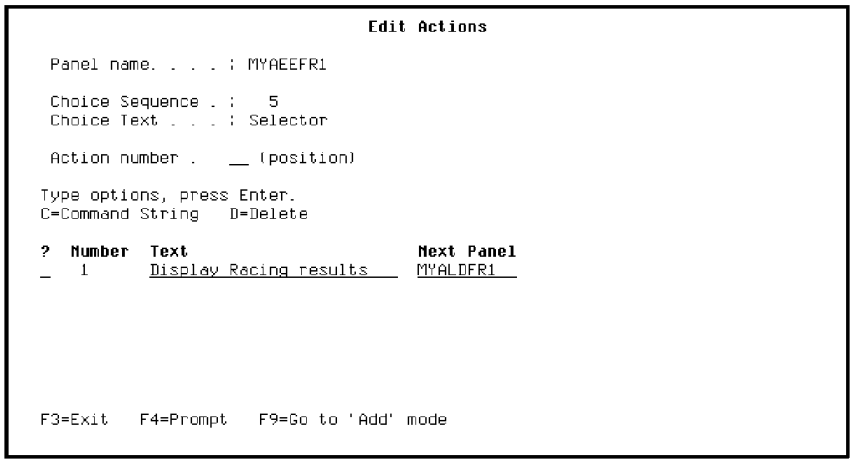


Press Enter to display the Toolkit Edit Actions panel.

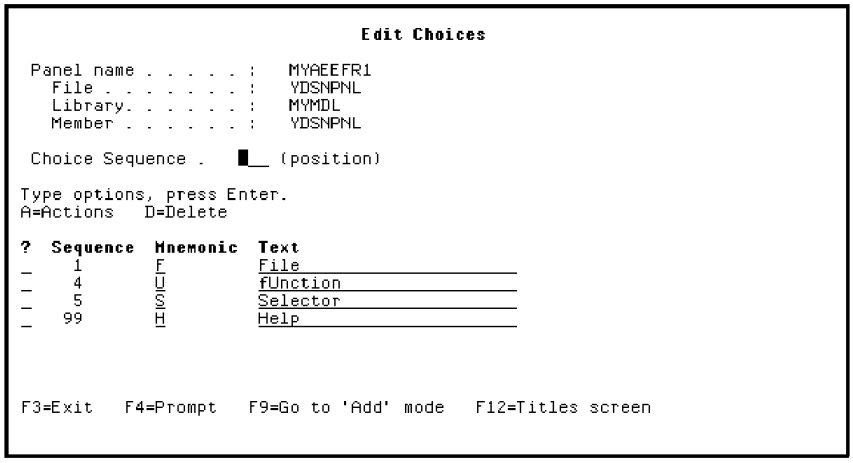
Note that the Next Panel column contains *SAME. This means that, by default, if you select Display Racing results from the action bar while simulating your program, the Edit Horse panel would be displayed again. Instead, you want to display the Toolkit panel design for the Display Racing results function.



Type the name of the Toolkit panel design assigned to the Display Racing results function in the Next Panel column as shown. By default, this is the CA 2E program implementation name followed by a 1; in this example, MYALDFR1 or type **?** for a selection list of Toolkit panel designs.



Press Enter to confirm the change. Press F3 to return to the Edit Choices menu.

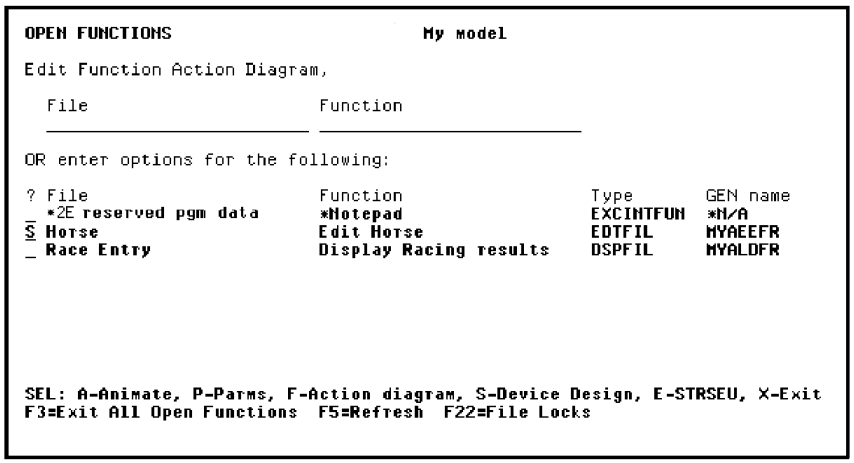


Press F3 to exit.

Press Enter to exit and update the Toolkit panel design.

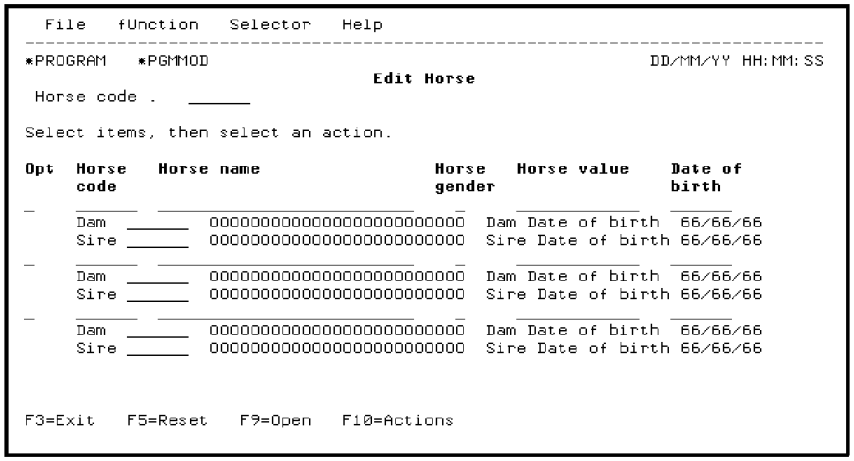
Testing the Function Link

You are now ready to demonstrate the link between the two functions. Type **S** in the Subfile selector for Edit Horse to display the CA 2E Edit Horse device design.



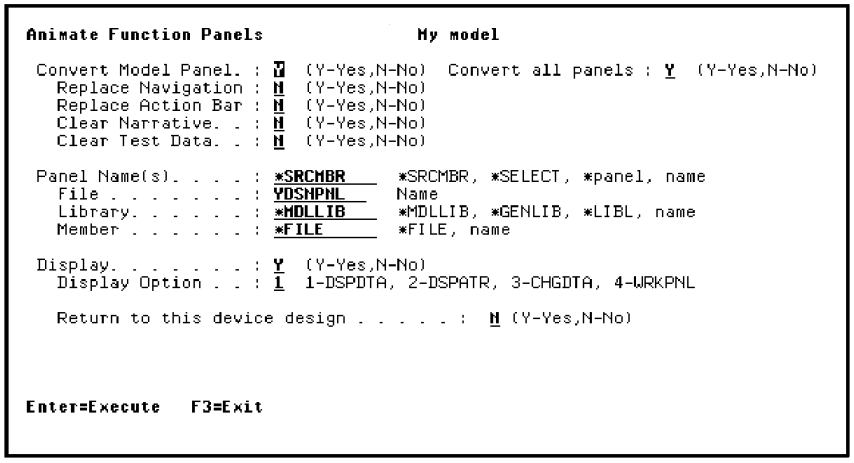
Press Enter.

Note: You could have typed **A** in the Subfile selector instead to animate the function. However, if you start the animation from the device design, it is easier to switch between the CA 2E device design and the Toolkit panel design, namely, you can press F2 from the device design to animate and press F3 from the panel design to return to CA 2E.



Animating Edit Horse

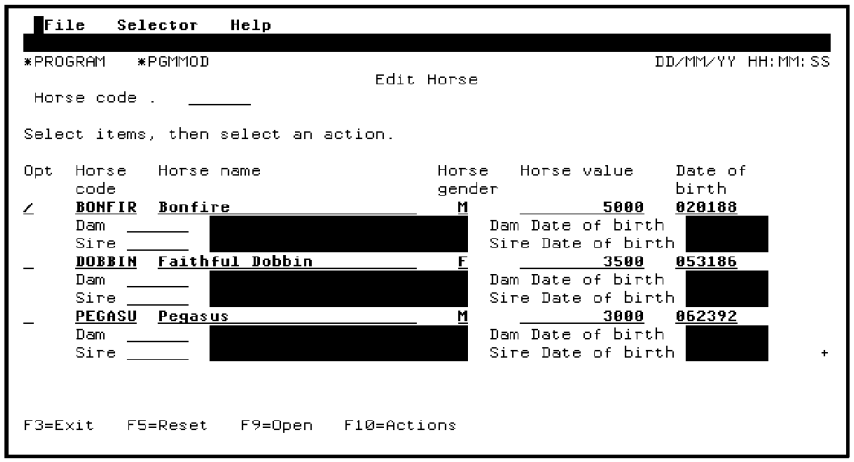
Press F2 to display the Animate Function Panels panel. Since you have already converted the CA 2E device design to a Toolkit panel design you can either accept the defaults or type **N** for the Convert Model panel field. Typing **N** can save time. In this case accept the defaults.



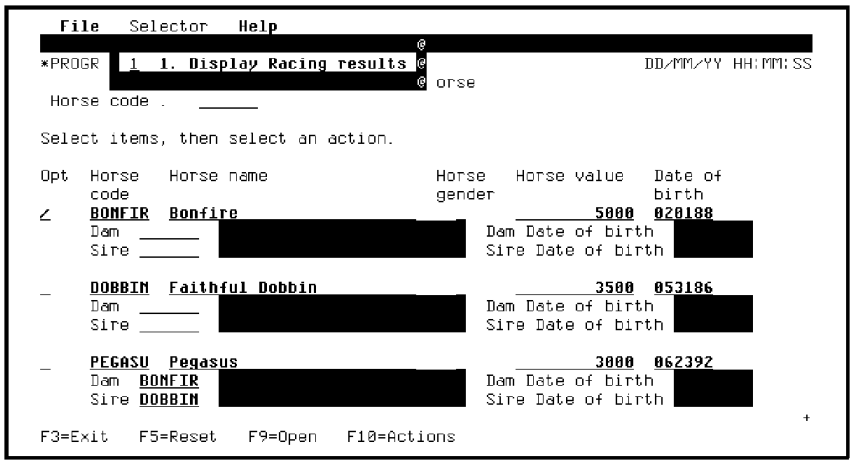
Press Enter. The Toolkit panel design for the Edit Horse function displays showing the sample data you entered earlier. Input-capable fields are indicated by underscores; output-only fields are shown in black.

Activate the Action Bar

To demonstrate the link between the two functions type **/** against one of the subfile records and press F10 to activate the action bar.



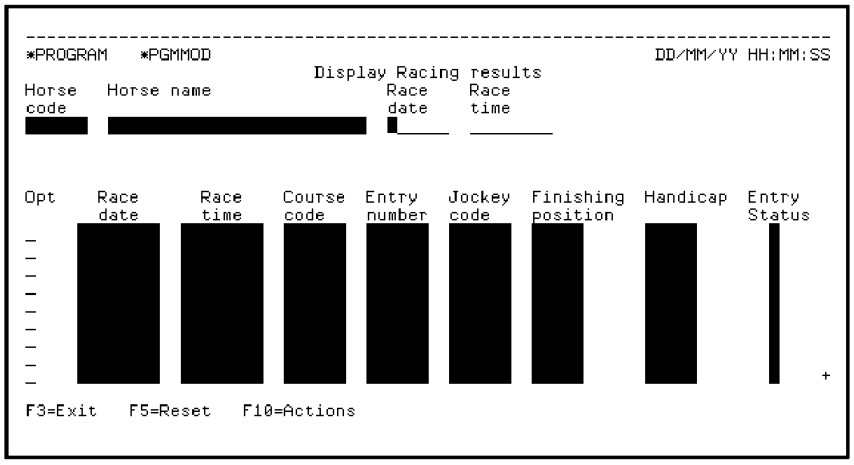
Type **S** to select the Selector Choice menu from the action bar; note that it includes Display Racing results as an action. Type **1** to select Display Racing results.



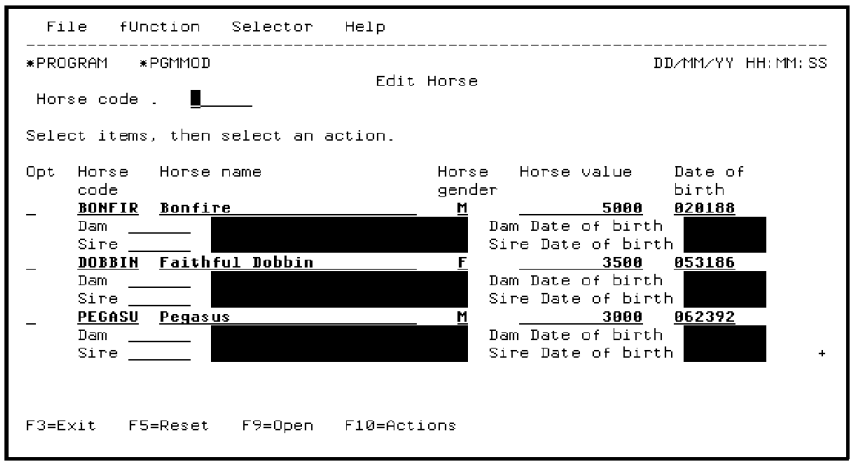
Press Enter.

Display Racing Results Panel Design

The Toolkit panel design for the Display Racing results function displays. Input-capable fields are indicated by underscores; output-only fields are shown in black.



Next test the command key navigation you set up for the Display Racing results panel design by pressing F3. This should cause a return to the Toolkit panel design for Edit Horse.



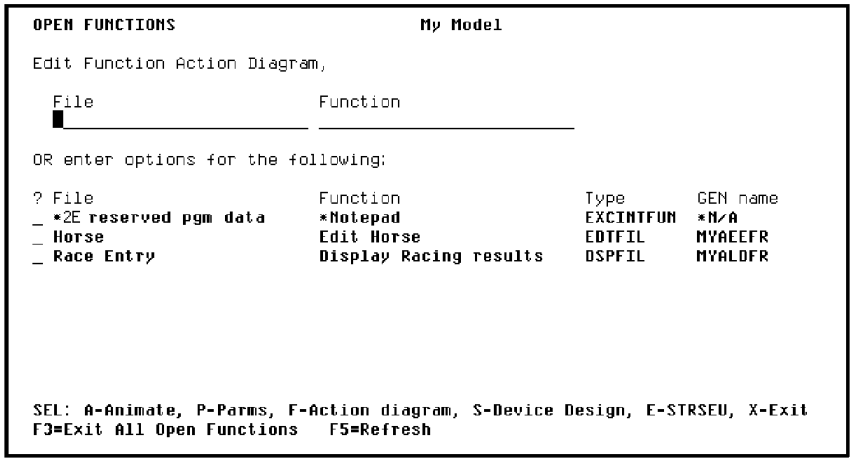
Press F3 to return to the CA 2E device design for the Edit Horse function.

You could now make design changes to the device design and repeat the animation process until you are satisfied with the results.

Note: If you have defined command key or action bar navigation in Toolkit, do **not** choose the options to replace action bar or command key navigation when you reanimate. In other words, accept the defaults for the Replace options on the Animate Function Panels panel.

Exiting Both Functions

Press F3 to exit the Device Design Editor. At the Edit Function Devices panel press F15 to return to the Open Functions panel. From this panel you can either exit all open functions, or you can exit functions individually using the **X** Subfile selector option. In this case, you will exit all open functions.



Press F3 to exit all open functions. You will proceed through the following steps.

The Exit Function Definition panel displays a second time for the Edit Horse function. Press Enter to accept the defaults.

The Exit Function Definition panel displays first for the Display Racing results function. Press Enter to accept the defaults.

The Edit Functions panel for the HORSE file displays. Press F3 to exit and return to the Edit Database Relations panel.

Press F3 to exit the model.

Working with Model Object Lists

This topic introduces the CA 2E model object list utility.

New terms introduced

- Model object
- Model object type
- Model object list
- Model object list entry
- Session list
- All Objects list (*ALLOBJ)

New panels introduced

- Edit Model Object List
- Display Model Object
- Work with Model Lists
- Subset Model Objects
- Position the List window

Objectives

To introduce the Edit Model Object List panel, to use some of its options and capabilities, and to encourage you to explore the model object list facilities.

Note: This topic presents only examples of ways to use model object lists. After you complete these examples be sure to experiment on your own until you feel comfortable with this tool. Refer to the CA 2E guide, *Generating and Implementing Applications* for more information.

Overview of Model Objects and Model Object Lists

By definition a *model object* is anything in the model that you can refer to by name; for example, a file (HORSE) or a function (Edit Horse).

Model Object Types

Within CA 2E there are seven types of model objects, many of which you have defined and used in this tutorial. Each *model object type* has a 3-letter identifier as shown in the following table.

Model Object	Model Object Type
Files	FIL
Fields	FLD
Conditions	CND
Access Paths	ACP
Functions	FUN
Messages	MSG
Arrays	ARR
Application Areas	APP

Model Object Lists

A *model object list* is a logical grouping of model objects. The way in which you use model object lists is limited only by your imagination. For example, a model object list might consist of all

- Model objects changed since a specified date
- Model objects related to a particular development project
- Access paths and functions that need to be generated and compiled as a result of a change to the model
- Model objects needed to implement the accounts payable feature of your application
- Functions called by an external function in which an execution error occurred
- Model objects that use a specified model object

Since a model object list can be used in many ways, it can be referred to by other names based on the way in which it is used. For example, a specific model object list might be called a session list, copy list, change list, or model list, for short. In reality, these are all just variations on the same concept.

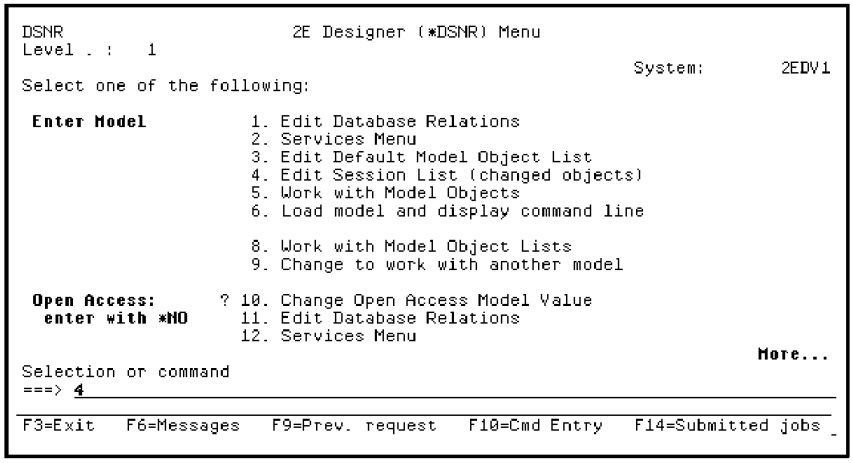
Model Object List Entry

A model object list is comprised of a set of references to model objects within the model. Each reference is known as a *model object list entry* or *list entry* for short. Each list entry contains information about a model object at the time the list entry was created. In other words, it provides a persistent historic record of the object at the time the list entry was created.

Editing Your Session List

At the beginning of this tutorial, you created a session list with the same name as your model profile. A *session list* is a model object list to which all model objects you change, add, or delete during a session are logged. Your session list should now contain all objects you added or changed while working on the horse racing model.

From the CA 2E Designer (*DSNR) Menu, select the Edit Session List (changed objects) option.



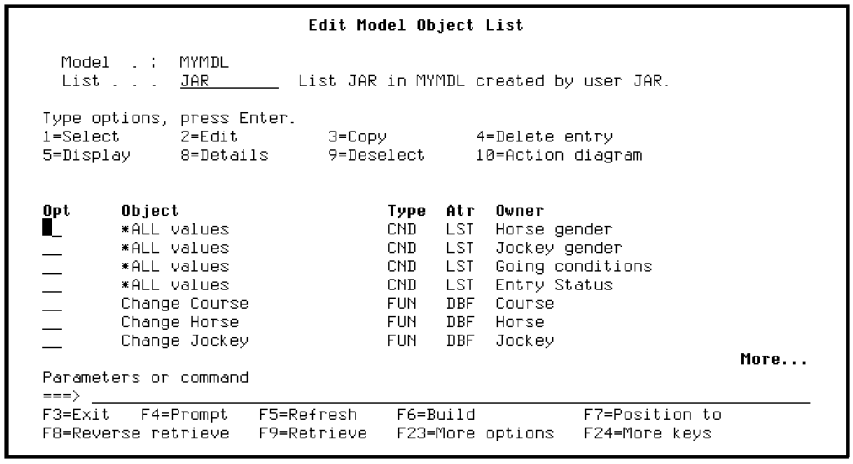
Press Enter.

The Edit Model Object List panel displays showing all model objects you changed during the tutorial.

Edit Model Object List Panel

The Edit Model Object List panel serves as an alternate entry point into your model. It has a PDM-like interface that you can use to perform most functions available from the Edit Database Relations panel. The exceptions are that you cannot edit relations or create model objects.

In the following steps you will use the Edit Model Object List panel to edit the session list you created at the beginning of this tutorial.

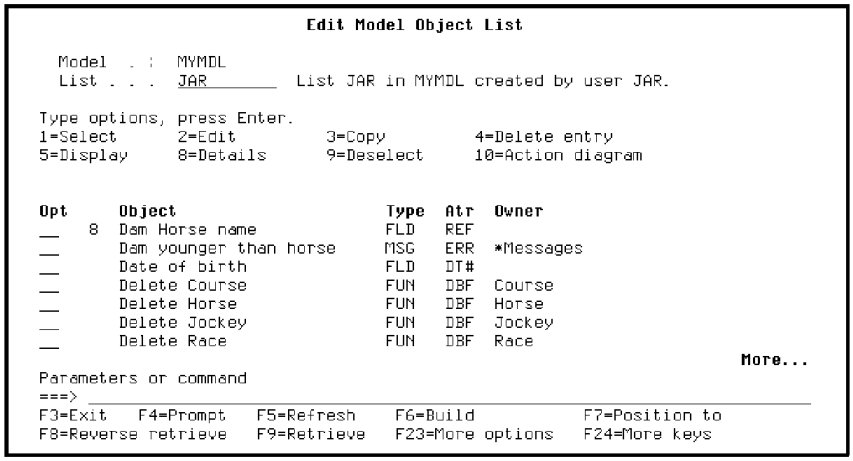


Viewing Model Object Types

By default model objects are listed alphabetically by object name and type. Press Roll Up several times to view some of the list. Note the values in the Type column.

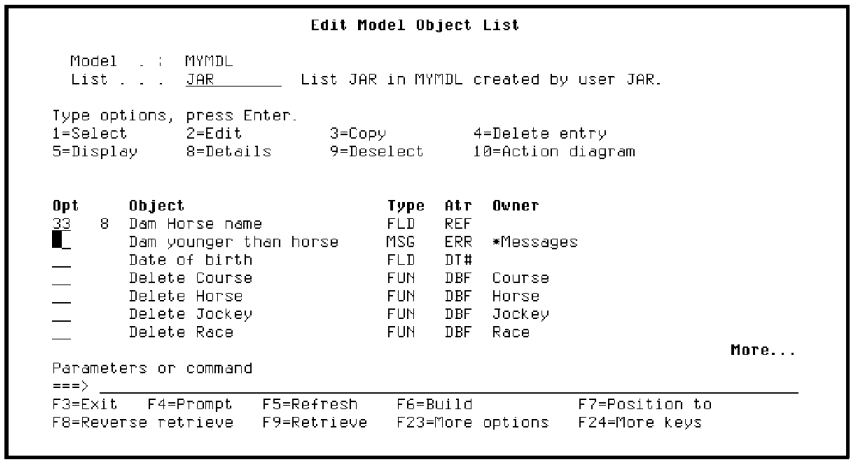
List Entry Differs from Model Object

Press Roll Up until you see the list entry for Dam Horse name. Note the number 8 to the right of the Subfile selector. This indicates that information for the list entry on your session list differs from that of the actual model object that the list entry represents. Recall that earlier in the tutorial, you changed the name of this field from Dam Horse name to Dam name.

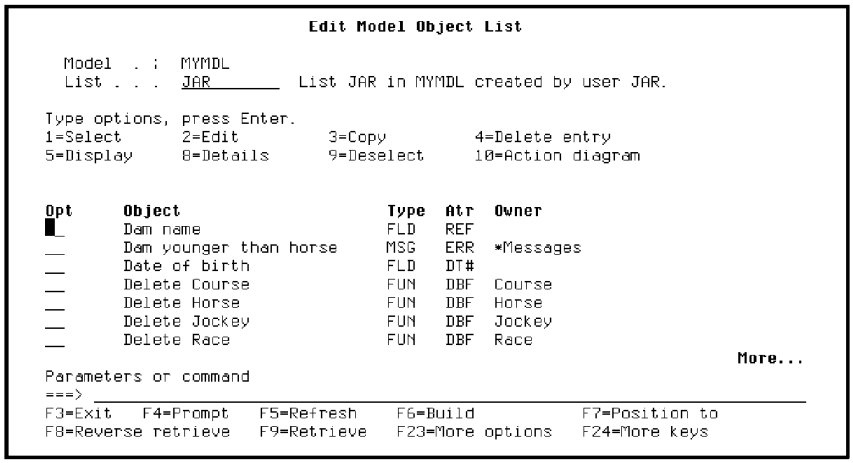


Note: CA 2E allows this difference to occur so you can use your model object list as a historical record.

You can update the model object list entry to reflect the current information for the actual model object. To do so, type **33** for the Subfile selector for Dam Horse name.



Press Enter. The list entry is updated from the detail information for the actual model object. In this case, the name of the list entry changes to match the name of the model object it represents.

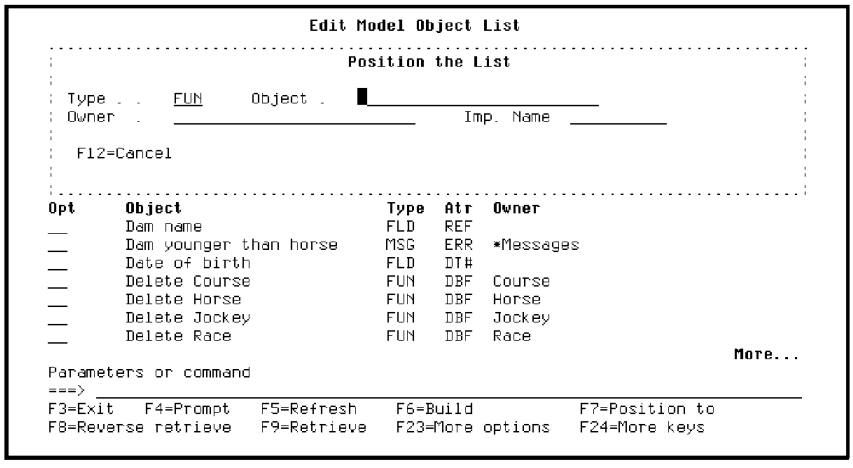


Positioning a Model Object List

You can position the model list displayed to a specific object type, object name, owner, or implementation name. This topic presents a few examples of ways to use positioning. Feel free to experiment on your own.

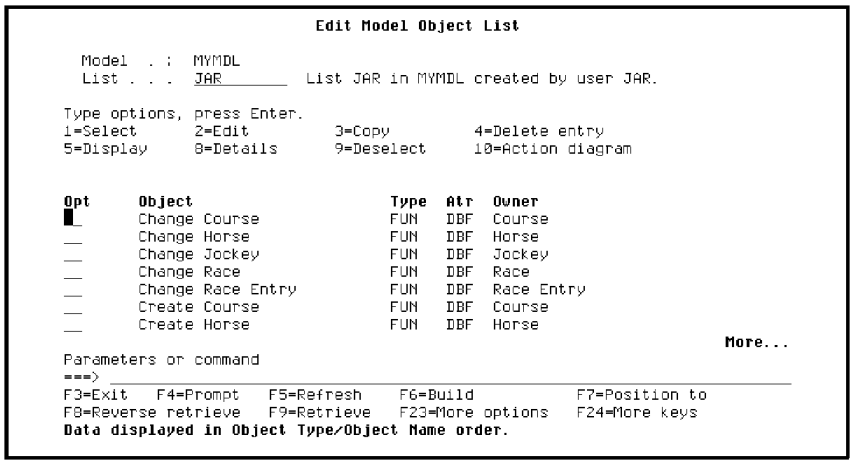
Press F7 to display the Position the List window. The values you enter in this positioner window also determine the order in which the model objects display.

Type **FUN** (function).



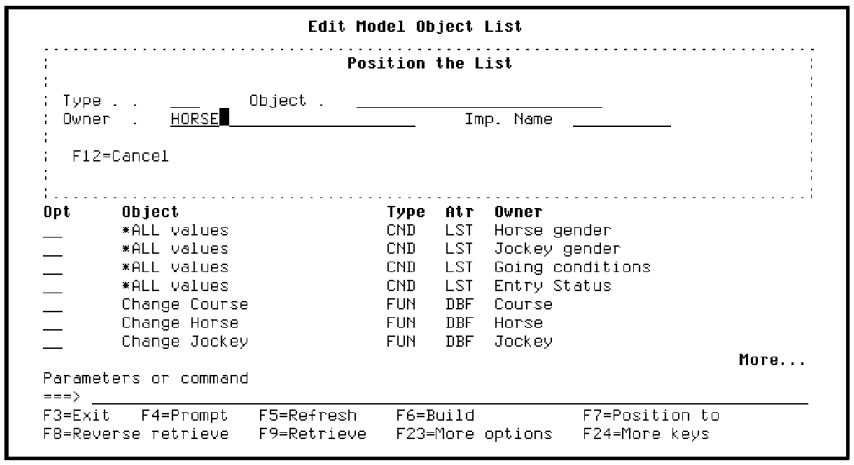
Press Enter.

Note that the list is now positioned at the first model object of type FUN. The following model objects are displayed alphabetically by type, and for each type, alphabetically by object name. Note the message displayed at the bottom of the panel indicating this.



Scroll through the list to verify this.

Press F7 again. Blank out the Type field and type **HORSE** for Owner.



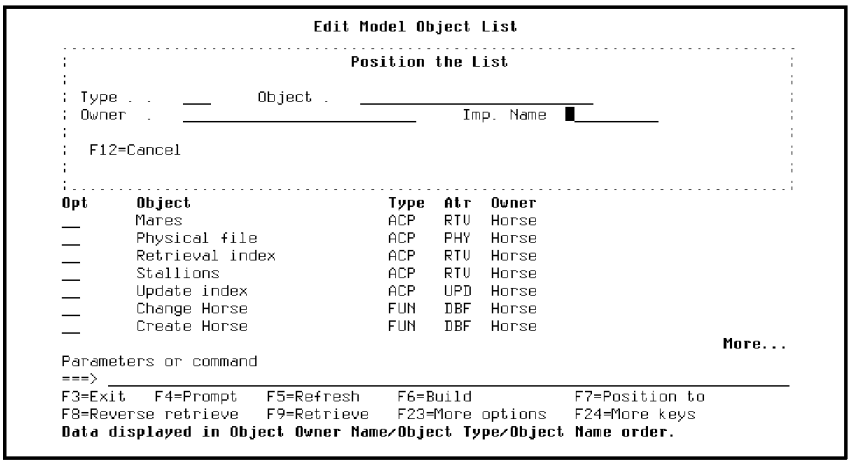
Press Enter.

Note that the list is now positioned at the first model object that has HORSE as its owner. Note the message at the bottom of the panel. The following model objects are displayed alphabetically by Owner, and for each Owner, alphabetically by Object Type, and then by Object Name. Scroll through the list to verify this.

Returning to the Top of the List

You can also use the positioner window to return quickly to the top of the model list displayed.

Scroll down so the list is not positioned at the top. Press F7 again. Blank out all fields.



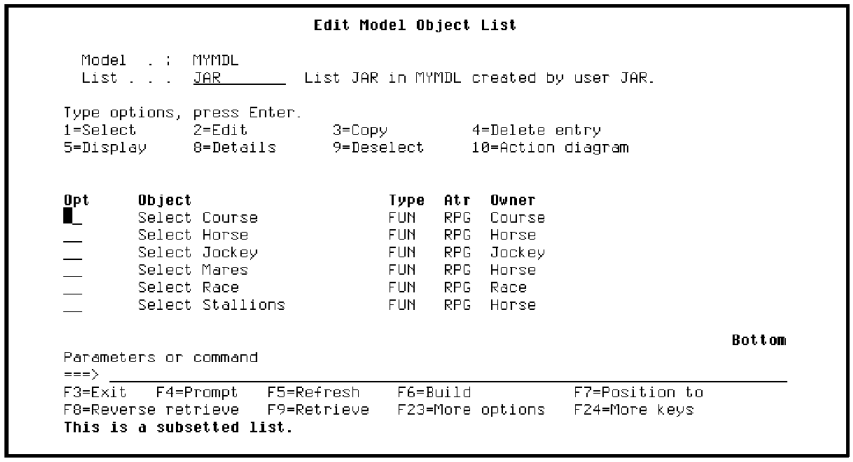
Press Enter. The list will be repositioned to the top of the list in order by object name and type.

Viewing a Subset of a Model Object List

You can also display a subset of a model object list, based on such criteria as Object name, Type, Owner, Implementation name, and the date the model objects were created or changed.

Suppose you want to view all Select Record (SELRCD) functions on your session list. To do so, press F17 to display the Subset Model Objects panel. (Press F24 to see more command keys.) Type ***FUN** for the Type field and type **SELRCD** for the Function type field.

Press Enter twice.



The list now displays only the model objects that meet the criteria you entered on the Subset Model Objects panel. Note the message at the bottom of the panel indicating that the list is subsetted; in other words, not all model objects in the list are displayed.

Subsetting is useful when you work with long lists or when you want to work only with specific types of model objects; for example, functions or access paths. Spend time experimenting with this feature.

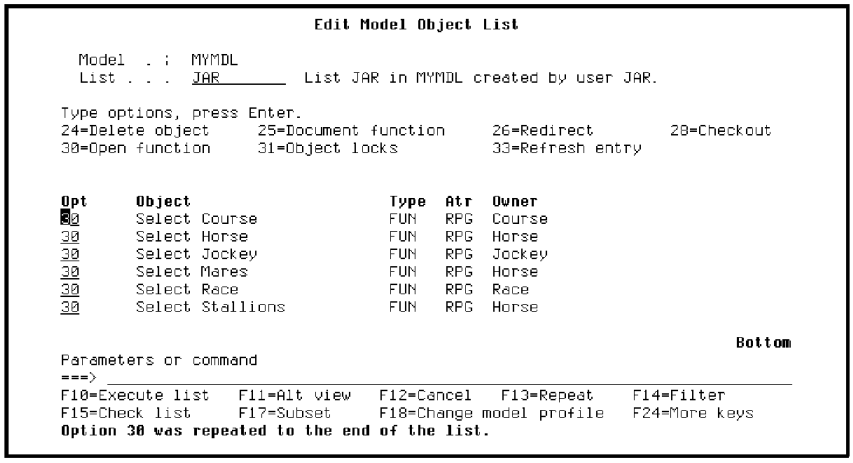
Opening Multiple Functions at One Time

Suppose you want to work with each of the Select Record functions on your model list; for example, to edit their device designs or action diagrams. Press F23 three times to view additional Subfile selector options. Option 30 lets you open several functions at once.

Type **30** in the Subfile selector of the first function displayed, Select Course.

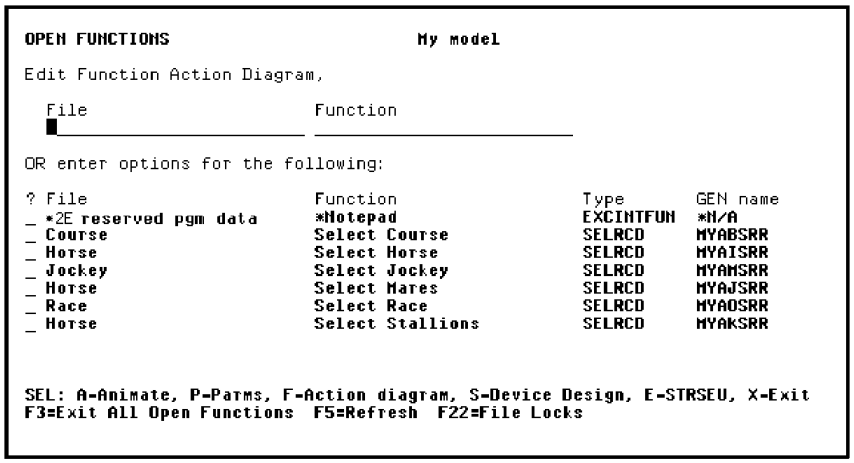
Next, press F24 to view additional command keys. The F13 command key automatically repeats the Subfile selector option you entered for all following subfile records to the end of the list.

Press F13 to repeat option 30 for each function displayed.



A message at the bottom of the screen tells you that the option was repeated to the end of the model list.

Press Enter to open each of the selected functions. The functions are added to the Open Functions panel.



Working with Open Functions

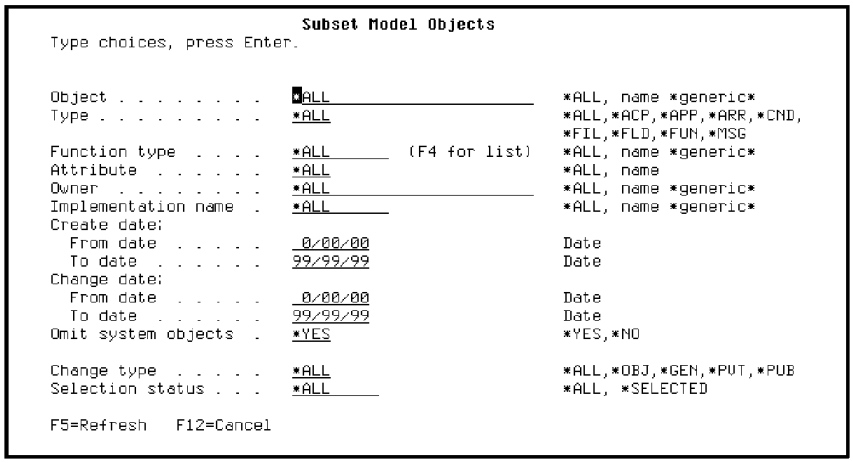
From the Open Functions panel you can work with each of the functions listed without incurring the overhead of exiting and reopening each time you need to work with a different function.

Press F3 to exit all the open functions.CA 2E displays the Exit Function Definition panel for each open function in turn where you can choose whether to save the function and submit it for batch generation.

When you exit the last open function,CA 2E returns to the Edit Model Object List panel.

Displaying the Unsubsetted Model List

Because only SELRCD functions are displayed, press F17 to display the Subset Model Objects panel. Press F5 to restore the default values for all fields.



Press Enter to display the original unsubsetted list.

Editing Model Objects

Subfile selector option **2** on the Edit Model Object List panel accesses the appropriate CA 2E editing panel according to the type of the selected model object. The following table shows examples for object types and panels you have used earlier in this tutorial.

Model Object Type	Tutorial Panel
Type	CA 2E Editing Panel Displayed
ACP	Edit Access Path Details
CND	Edit List Condition for LST Edit Field Condition Details for VAL
FIL	Edit File Details
FLD	Edit Field Details
FUN	Edit Function Details
MSG	Edit Message Function Details

If you type **2** for the Subfile selector of several model objects of differing types,CA 2E displays the appropriate editing panel for each model object, one at time.

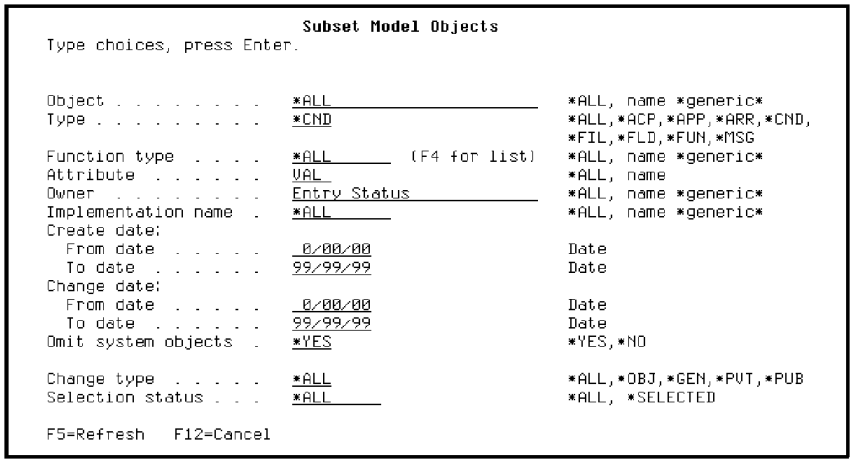
Exercise

Type **2** in the Subfile selector for model objects of several different types of model objects and press Enter. Notice which CA 2E editing panel displays in each case. Press F3 to return to the Edit Model Object List panel.

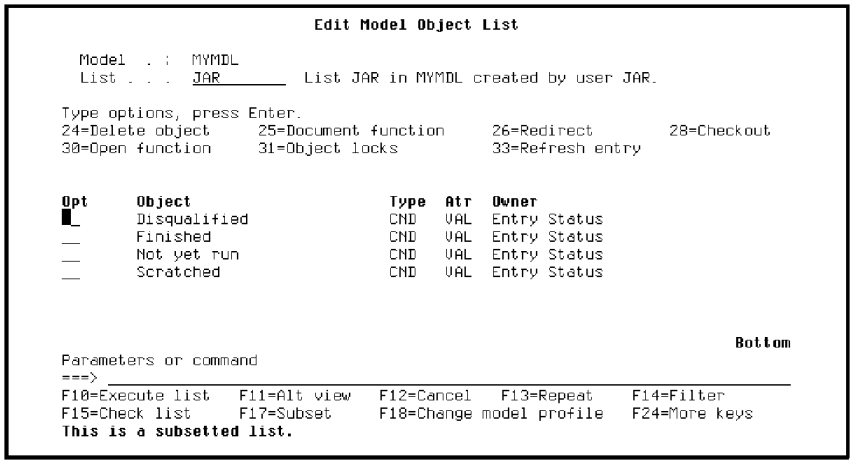
Editing Conditions for Entry Status

You can also edit only specific objects by subsetting the model list first. Suppose you want to view and possibly change the condition values you defined for the Entry Status field on the RACE ENTRY file. For example, you can change the condition names and status values. Note that you cannot add a new condition in this way since you need to use the Edit Database Relations panel to create a model object.

Press F17 to display the Subset Model Objects panel. Type ***CND** for the Type field, **VAL** for the Attribute field, and **Entry Status** for the Owner field.



Press Enter twice to display the subsetted list.



Exercise

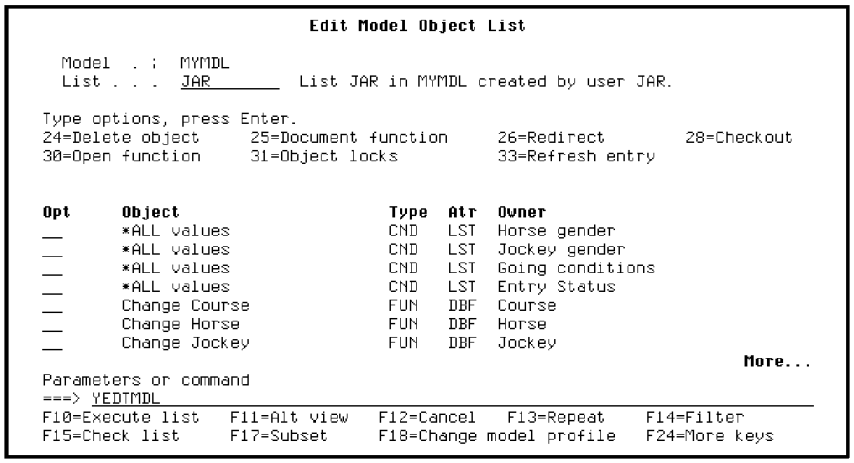
View and optionally change each condition value. Namely, type **2** for the first condition value displayed, and then press F13 to repeat this option for the rest of the condition values. Press Enter to display the Edit Field Condition Details panel for each condition value. Press F3 to exit each editing panel and finally to return to the Edit Model Object List panel.

Displaying the Unsubsetted Model List

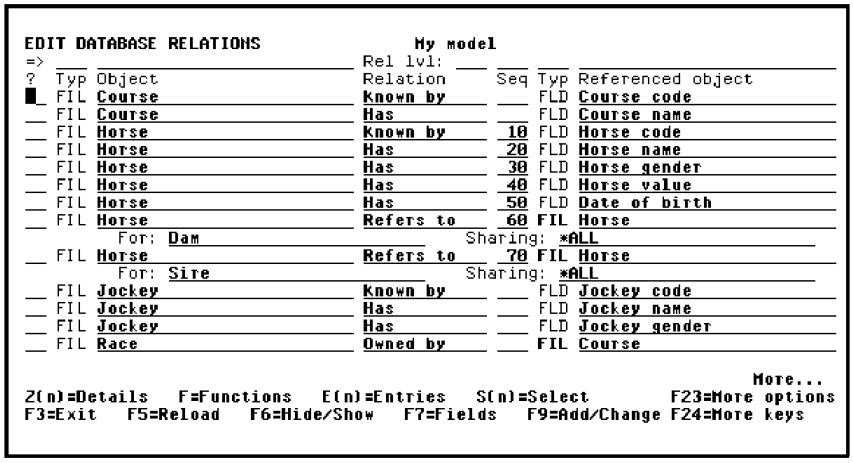
From the Edit Model Object List panel, press F17 to display the Subset Model Objects panel. Press F5 to restore the default values for all fields and press Enter to display the original unsubsetted list.

Editing Relations and Creating Objects

As already mentioned you cannot edit relations or create model objects using the Edit Model Object List panel. If you need to edit a relation or create a model object, you can transfer temporarily to the Edit Database Relations panel by entering the Edit Model (YEDTMDL) command on the command line.



Press Enter.



When you finish editing relations, press F3 to return to the Edit Model Object List panel.

Deleting a Model List Entry

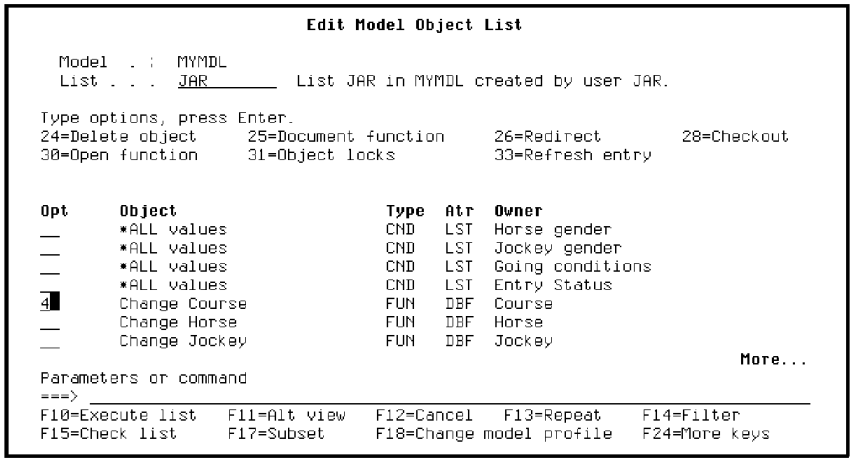
The Edit Model Object List panel provides two deletion options. Option 24 deletes the actual model object from the model; option 4 deletes a list entry from the displayed model object list. When you delete a model object list entry (option 4),CA 2E displays a confirm panel for each selected list entry.

Note that you cannot delete a model object if it is used by other objects in the model.CA 2E provides impact analysis tools to help you determine whether and how a model object is used by other model objects. Impact analysis is discussed later in this chapter.

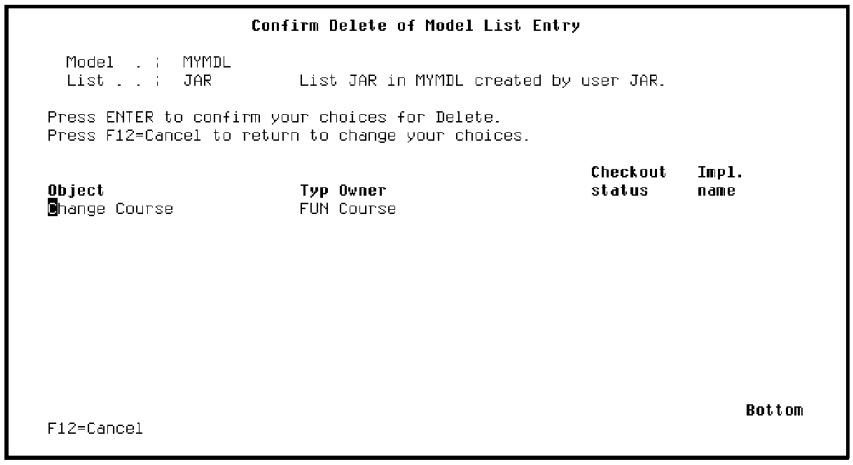
Note: Because you can use a model object list as a historic record for your model, it can contain entries for model objects that you have deleted from the model. These list entries are indicated by an X to the right of the Subfile selector.

For practice you will delete a list entry from your session list in this step. You will replace it later so your session list continues to represent all changes to your model. Remember, you are deleting a list entry, not the model object.

Enter **4** in the Subfile selector for the Change Course internal function.



Press Enter.CA 2E displays a confirm panel.



Press Enter to confirm the deletion.

All Objects List

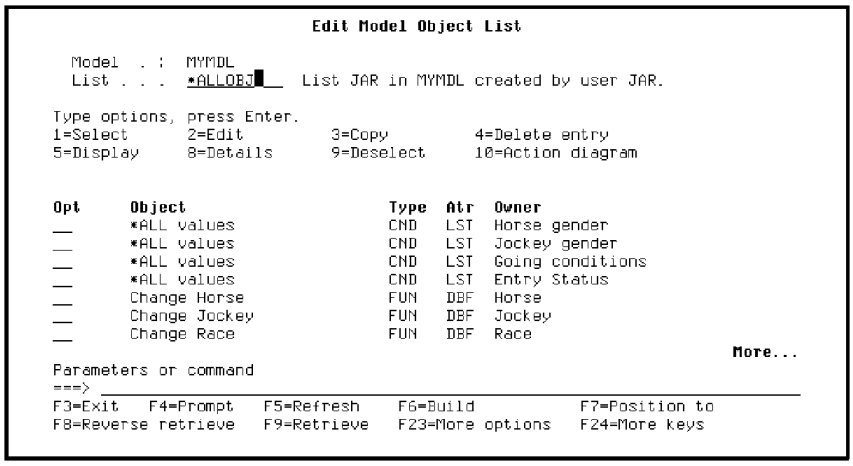
Each model contains a special list containing current information about each object in the model. This list is referred to as the All Objects list and is accessed within the model as *ALLOBJ.

The major purpose of the All Objects List is to provide a central location for model object information and to record information related to changes. The information for each model object is contained in a model object description, or detail. As you create, delete, update, and generate model objects,CA 2E automatically updates the corresponding model object description in the All Objects list to reflect the changes.

Accessing the All Objects List

One way to access the All Objects List is to type *ALLOBJ in the List field at the top of the Edit Model Object List panel.

Type ***ALLOBJ** for the List field.



Press Enter.

The List field now contains *ALLOBJ. This means there is a one-to-one correspondence between the model objects displayed and the actual model objects that comprise your model. In particular, note that the model object you deleted from your session list, namely, Change Course, appears on the All Objects List.

How *ALLOBJ and Model Lists Differ

Technically, the All Objects list is not really a model object list; however, it is often useful to think of it as a list. In general, you can use the Edit Model Object List panel to work with the All Objects List as you would a named model object list. However, since the All Objects list represents actual model objects, note the following exceptions.

You can delete a model object, but you cannot delete a list entry. In other words, you can use option 24, but not option 4.

You cannot add a model object to the All Objects list since that requires creating a model object.

While the All Objects list contains all active model objects, named model object lists typically represent a subset of all model objects in the model.

Displaying Alternate Views of Detail Information

The Edit Model Object List panel provides five alternate views of detail information for each model object. The information shown in the five views is dynamic when you are working with the All Objects list; in other words, if you update a model object,CA 2E automatically updates the information displayed to reflect the change. For named model object lists, the information displayed is static and reflects the state of the model object at the time it was added to the model object list.

The following views are available:

- Basic information
- Implementation information
- Component Change information
- Change information
- Check out information

Details about each of these views are beyond the scope of this tutorial. To learn more, see the CA 2E module, *Generating and Implementing Applications.*

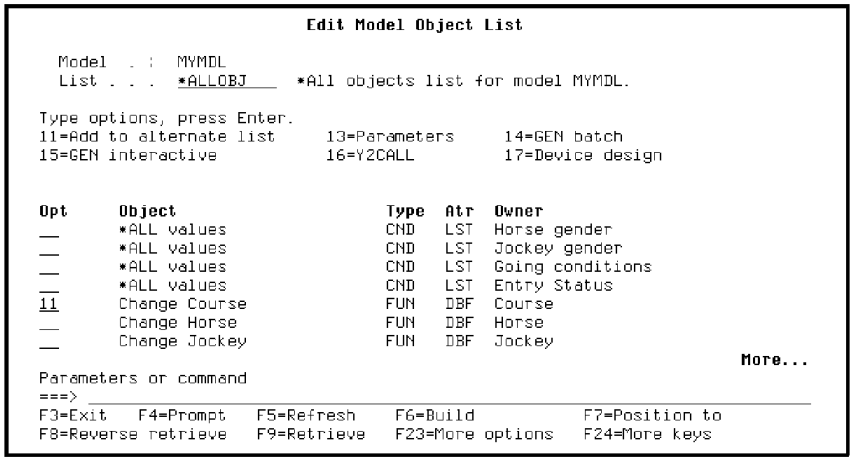
Exercise

Press F11 to view the five alternate views of information from the detail for the model objects displayed. Some of this information depends on the type of the model object so you should scroll through the list while viewing each alternate view.

Restoring the Deleted Session List Entry

In this step you will restore the Change Course function to your session list. Subfile selector option 11 lets you select model objects to be added to another model object list. The default for this alternate list is the Model list for commands specified in your model profile. In your case it is the same as your session list.

Type **11** in the Subfile selector for the Change Course model object.



Press Enter. A message at the bottom of the panel indicates that the model object has been added to your session list.

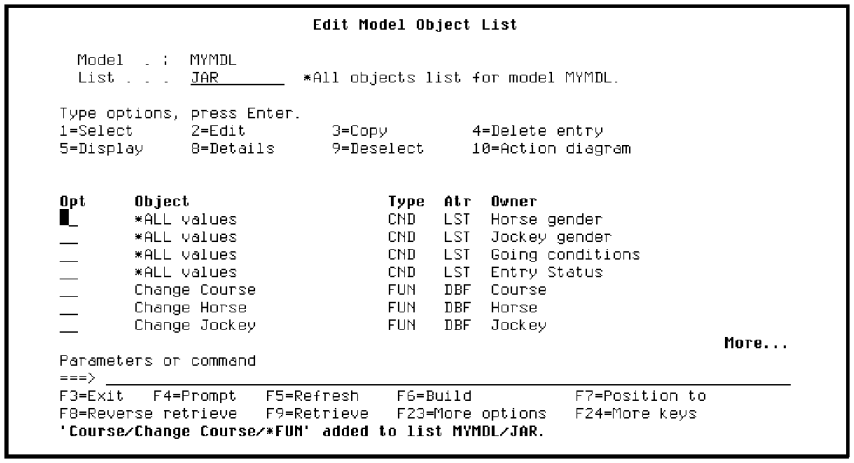
Note: Alternatively you can press F4 instead of Enter to prompt option 11. This lets you enter the name of another alternate list or *SELECT to select from a list of all model object lists.

Accessing Other Model Lists

You can type the name of a model list in the List field on the Edit Model Object List panel to transfer quickly among model object lists defined for your model. You can also press F4 to prompt a selection list of all model object lists you have defined.

Returning to Your Session List

Type the name of your session list in the List field as shown. Be sure to substitute the name of your session list.

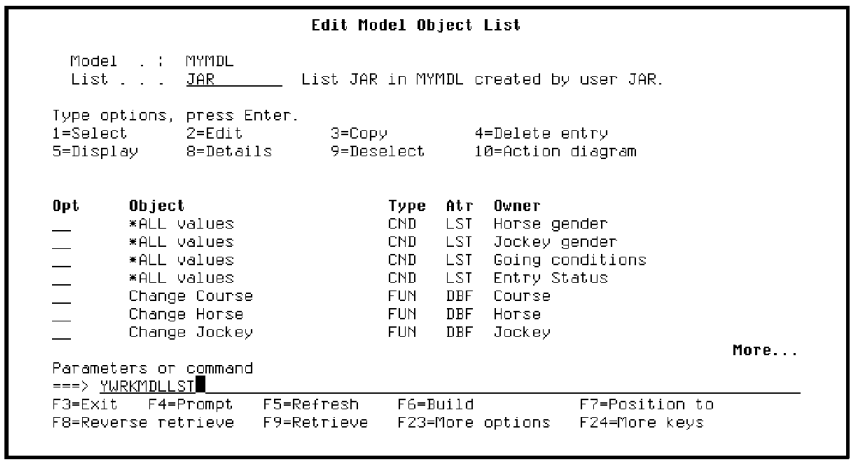


Press Enter. Note that Change Course has been restored to your session list.

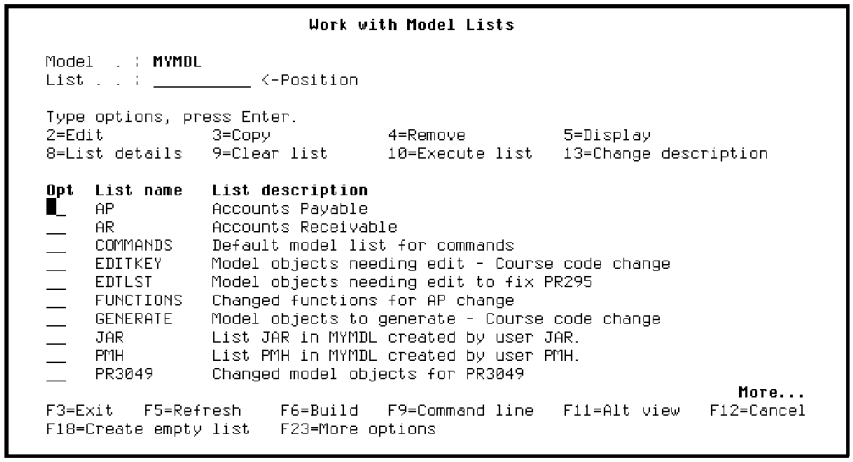
Working with Model Object Lists

The Work with Model Lists panel lets you manage the model object lists in your model. You can access this panel from the CA 2E Designer (*DSNR) Menu, the Display Services Menu, or by typing the Work with Model Lists (**YWRKMDLLST**) command on a command line

Type **YWRKMDLLST** on the command line.



Press Enter. The Work with Model Lists panel displays showing all the model lists defined for your model. In your case, only your session list will be shown. The following is an example of how this panel might look in a working model.



For any list displayed you can do the following:

- View the list entries
- Edit the list
- Clear all entries from the list
- Copy the list to another list
- Change the list's descriptive text
- Remove the list

If you type option **2** in the Subfile selector of any list, the Edit Model Object List panel displays for the selected list. As a result, this panel provides another method of entering your model.

Press F3 to return to the Edit Model Object List panel for your session list.

Function Versioning

This topic describes how to create and test a version of a function.

Note: You can create versions for both functions (FUN) and messages (MSG); however, this topic discusses only function versions.

New terms introduced

- Version
- Group
- Current
- Redirection

New panel introduced

- Work with Versions

New commands introduced

- Create Model Version
- Compare Model Objects
- Redirect Model Object

Objectives

To create a version of the Edit Horse function, edit its action diagram, test the change, and replace the original function with the new version.

Overview of Versions

A *version* of a function is similar to a copy of the function. The major difference is that a copy is completely independent of the original function. Although a version is also a separate model object,CA 2E maintains an internal link between a function and its versions. A function can have an unlimited number of versions.

A function and its versions are known as a *group*. The interactive Work with Versions panel lets you view and work with a group of versions.

In any group of versions, one of the versions in the group may be *current*. The current version is the version that is active in the model; in other words, it is the function that is referenced by other objects in the model and that appears on CA 2E editing panels.

Benefits of Versions

Three benefits of using versions are:

- You can test changes on a version of a function without interfering with the functionality of the existing model.
- When you finish testing a new version of a function and make it active (current) in the model, the original function remains unchanged and can easily be made active again if needed.
- Only the currently active version of a function is displayed on CA 2E editing panels. As a result, the panels are not cluttered with inactive versions.

A Reason NOT to Use Versions

When you make a version of a function current in the model,CA 2E globally changes all the model objects that referenced the original function to reference the version instead. If not all of the referencing model objects need the changed functionality, you should create a new function rather than a version. After updating and testing the new function, you would then need to update references to the new function manually.

Using Versions to Update an Existing Function

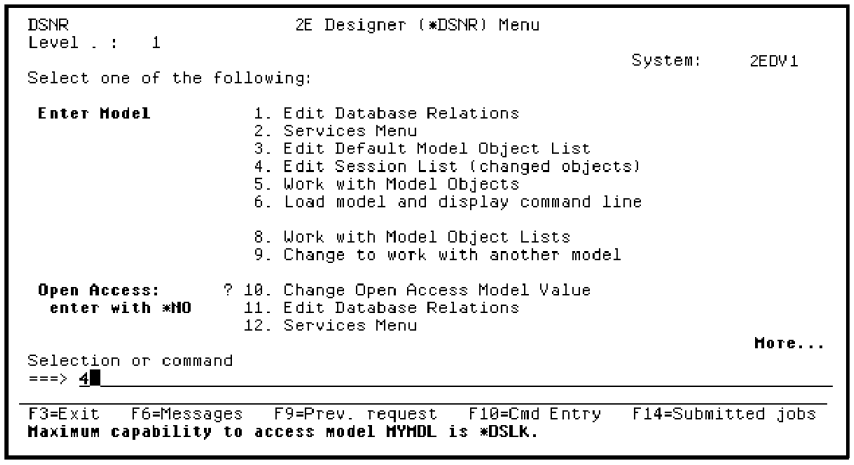
Following are the steps needed to update and test changes for a working external function using CA 2E's versioning feature. You will use this process to add an action to the action diagram for the Edit Horse function.

1. Create a version of the function you want to change.
2. Edit the version.
3. For an external function, generate the source and create the program object for the version.

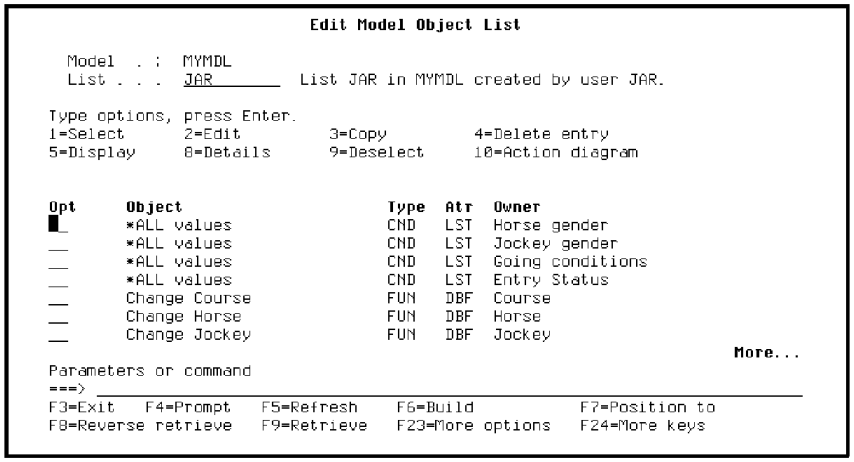
   Note: Internal functions and messages do not result in separate program objects. As a result, the testing process for them is more complex and beyond the scope of this tutorial. Refer to the CA 2E module, *Generating and Implementing Applications*, for more information.
4. Test the version by calling the program object.
5. When you are satisfied that the version works properly, make the version current.
6. If errors occur, make the original function current again.

Accessing the Session List

The Edit Model Object List panel should be displayed on your screen for your session list. If it is not, from the Designer Menu, type **4** on the Selection line as shown to select the Edit Session List (changed objects) option.



Press Enter to display the Edit Model Object List panel for your session list.



Positioning to the Edit Horse

Since you are going to create a version of the Edit Horse function, you first need to locate it. One way to do so is by using the F7 command key, which lets you position and sort entries for the displayed model object list.

Press F7 to display the Position the List window. There are several ways in which you can specify information on this window in order to locate the function you need. For example,

- If you know the name of the function, simply type the name in the Object field
- If you know the implementation name of the function (the program name), type the name in the Imp. Name field
- If you know the name of the file that owns the function, type the name of the file in the Owner field and scroll down to locate the function

Note that you can type partial names in the Object, Owner, and Imp. Name fields on the Position the List window; for example, if you type **H** or **H*** for the Object field, the list will be positioned at the first object name that begins with the letter H.

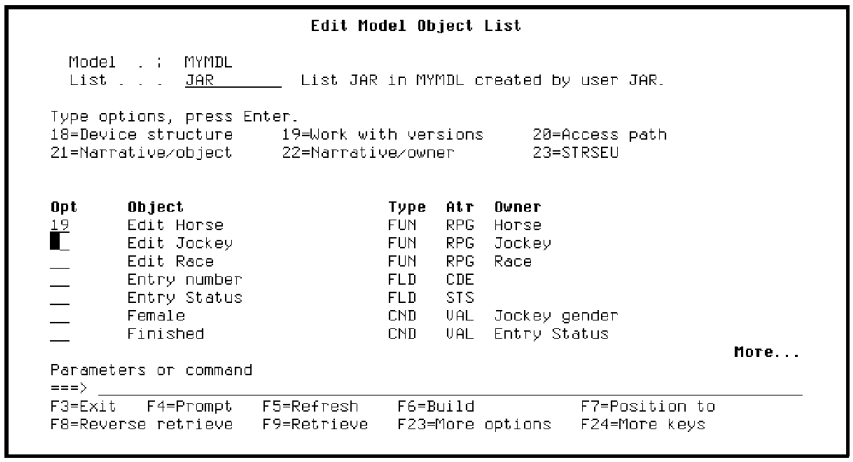
Because you know the function name, type **Edit Horse** in the Object field as shown.

Press Enter.

Creating a Version for Edit Horse

To create a version you first need to access the Work with Versions panel for the selected function. Press F23 twice to view additional Subfile selector options. The appropriate option is 19.

Type **19** in the Subfile selector for the Edit Horse function.



Press Enter.

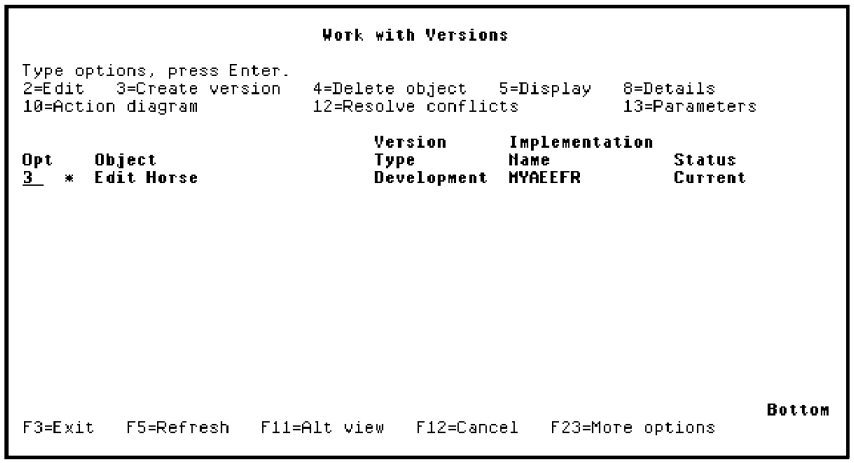
Working with Versions

The Work with Versions panel provides a set of options for working with a group of versions. Use it to perform tasks on versions, such as,

- Create a new version
- Edit
- Delete
- Display
- View detail
- Generate
- Impact analysis

The versions are displayed on this panel in reverse chronological order; in other words, the most recently created version appears at the top of the list of entries. The current version is shown highlighted with an * to the right of the Subfile selector, and the Status column contains the word Current. Since you have not yet created versions for Edit Horse, only one entry is listed.

To create a new version for the Edit Horse function, type **3** in the Subfile selector.

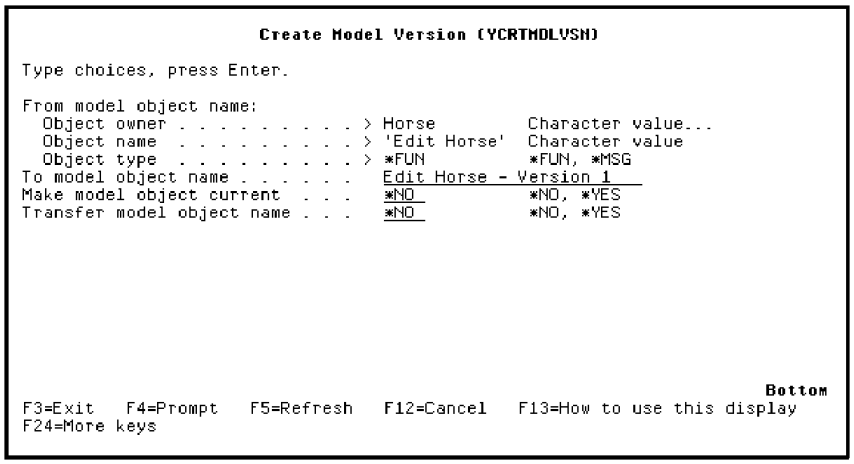


Press Enter.

Naming the Version

The prompt panel for the Create Model Version (YCRTMDLVSN) command displays. By default,CA 2E automatically generates a name for the version based on the name of the original function. The automatically generated name consists of the original function name followed by seven digits to ensure uniqueness; for example, Edit Horse1101481. In addition, by default, the original function retains its name and remains current (active).

In this example, you will specify a version name and accept the other defaults. Type **Edit Horse - Version 1** in the To model object name field.



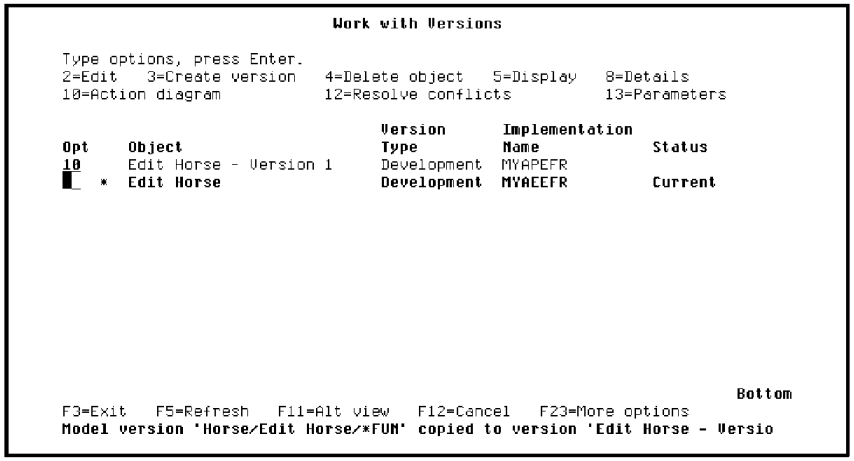
Press Enter. Note the messages at the bottom of the panel as CA 2E creates the new version.

Implementation Name for the New Version

The new version you just created, Edit Horse - Version 1, now appears at the top of the list on the Work with Versions panel. Note that a new implementation name has been assigned to the version; in other words, the version is a separate object in the model. However, since the version is not current, you will not see it on CA 2E editing panels.

Editing the New Function

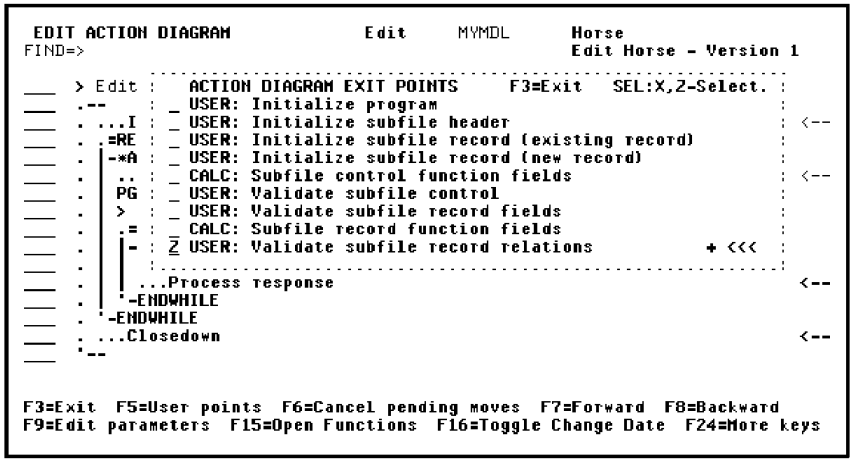
You are now ready to edit the action diagram for Edit Horse - Version 1 in order to insert a call to an information message function. Type **10** in the Subfile selector.



Press Enter.

Press F5 to display the user exit points and select the user point to which you added logic to call the Display Racing results function for a selected horse.

Type **Z** against the Validate subfile record relations user point.

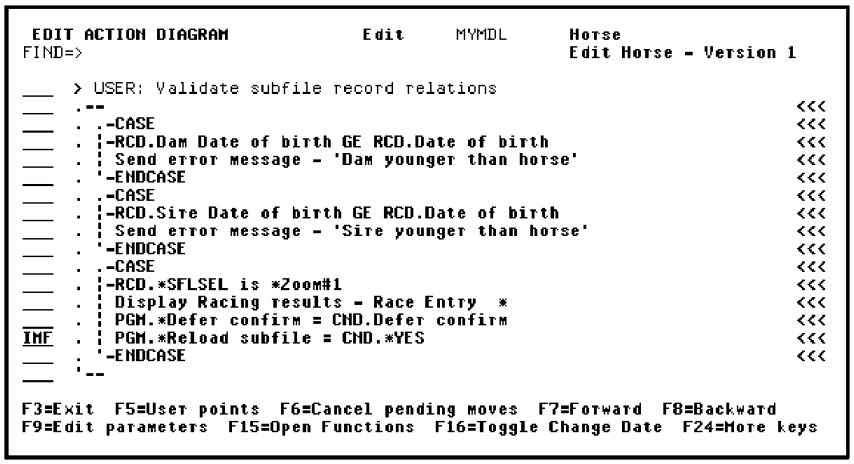


Press Enter.

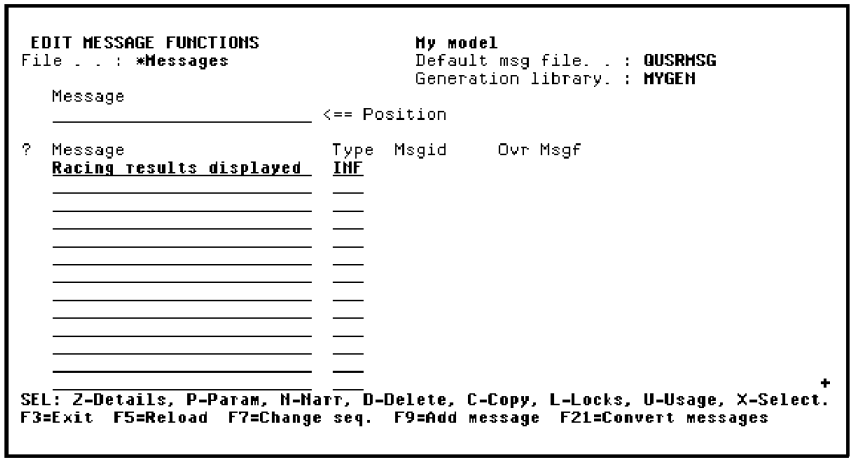
Inserting a Message Function

The objective is to add an information message to the Edit Horse function on return from the Display Racing results function. You will use the IMF command to insert the message. This command provides a shortcut for inserting and prompting a message function in one step. Remember, to view a list of available commands, type **?** in the Subfile selector.

Type **IMF** in the Subfile selector as shown to insert an information message function after calling the Display Racing results function.



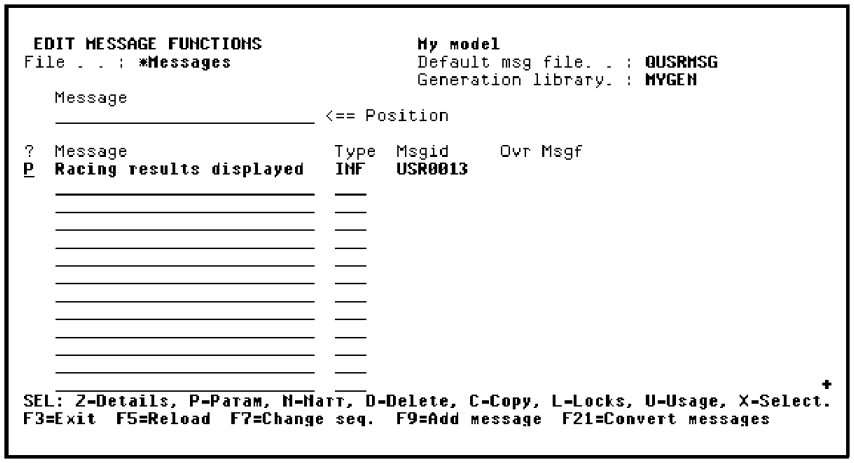
Press Enter. When the Edit Message Functions panel displays, press F9 to add the new message.



Press Enter.

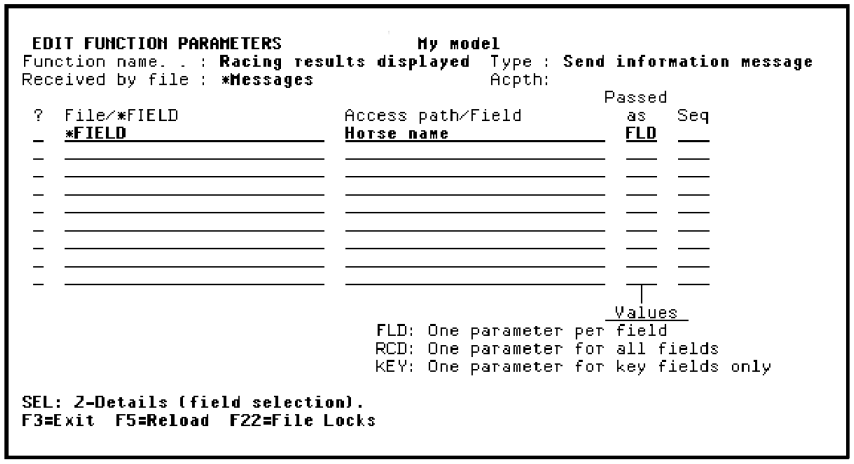
Specifying a Parameter for the Message Function

Next specify a parameter for the message function in order to include the name of the horse in the message. Type **P** as shown in the Subfile selector.



Press Enter.

You need to specify that the *Horse name* field is to be a parameter for the message function. As a result, type ***FIELD** to indicate the parameter is a field, type **Horse name** as the field name, and type **FLD** in the Passed as column.



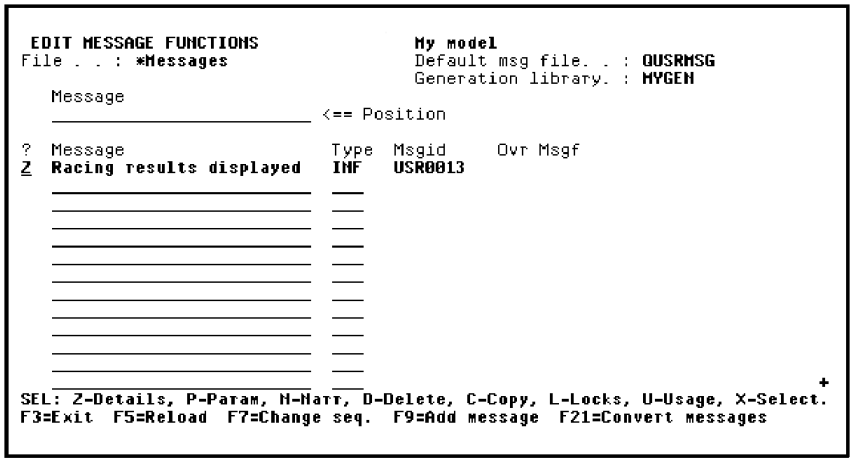
Press Enter.

Press F3 to return to the Edit Message Functions panel.

Editing the Message Text

Next you will edit the message text to include the value of the Horse name parameter. Remember that, by default, the name of a message function and its associated text are the same.

Type **Z** in the Subfile selector for the Racing results displayed message function.

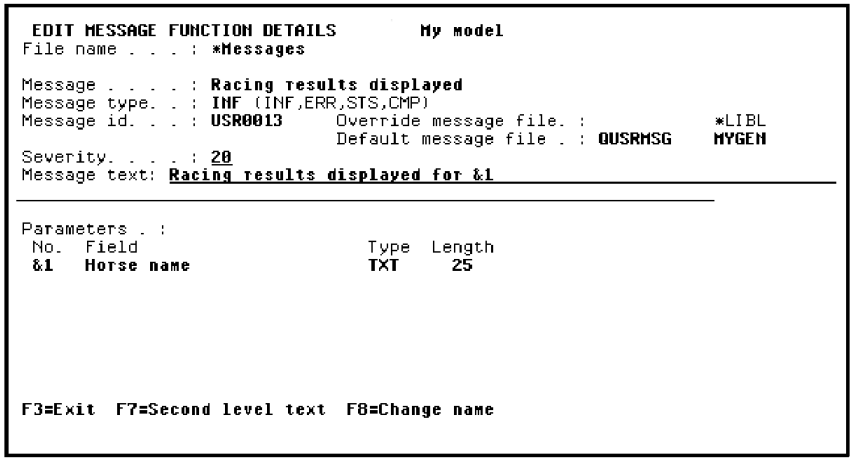


Press Enter to display the Edit Message Function Details panel.

Using a Substitution Variable

Note the &1 in the No. column; this is the substitution variable that represents the Horse name parameter. You will use this substitution variable to include the selected horse's name in the text of the message. Recall that you used this process earlier in the tutorial when you defined error messages for the condition that compared a horse's date of birth with those of its parents.

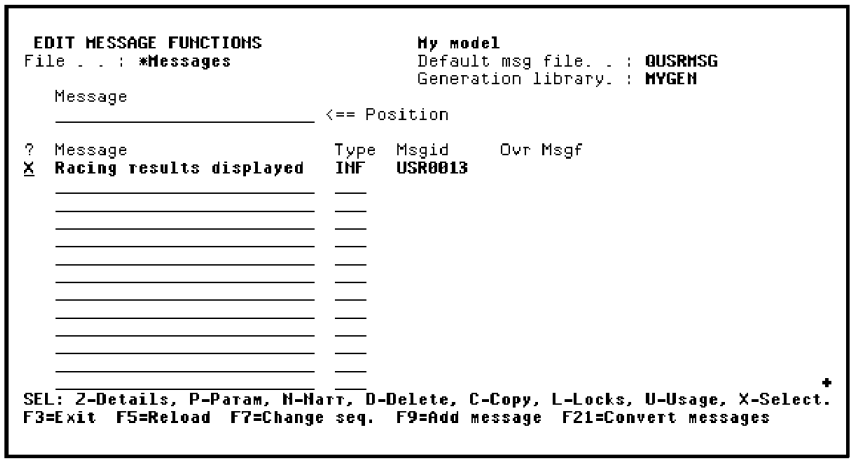
The original message text is "Racing results displayed." Add "for &1" to the end of this text.



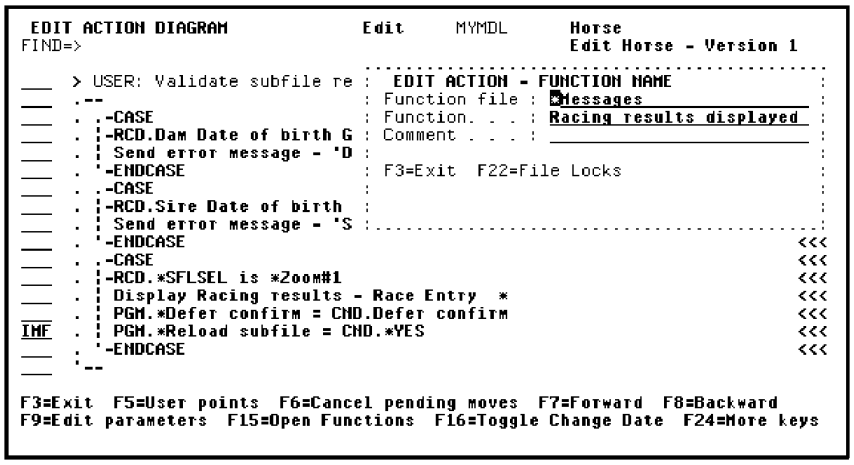
Press Enter to return to the Edit Message Functions panel.

Selecting the New Message

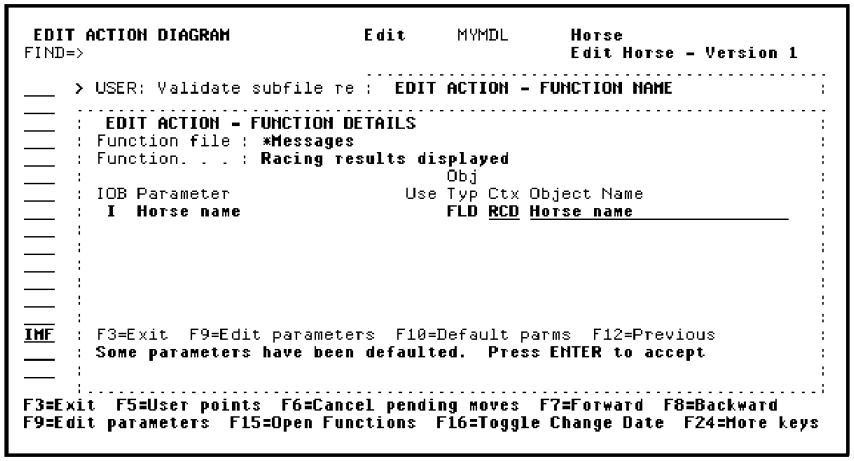
You still need to select this message function for inclusion in the Edit Horse function. Type **X** in the Subfile selector.



Press Enter to display the Edit Action - Function Name window. Note that the fields in this window have been filled in based on the information you just entered.



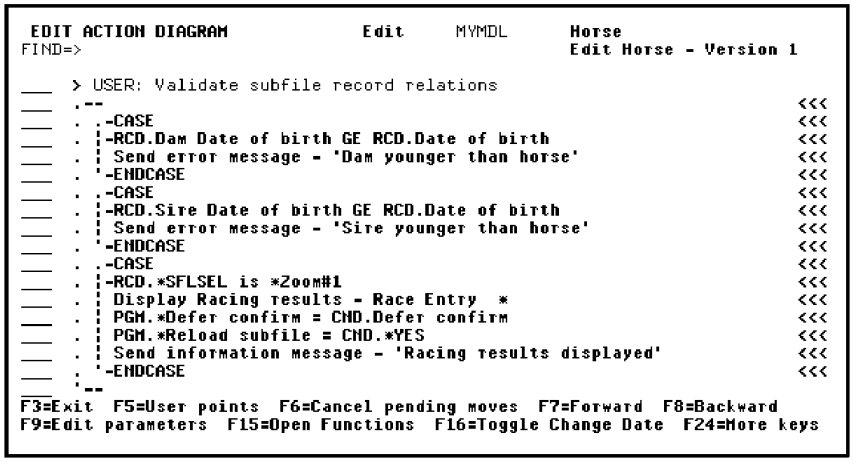
Press Enter to view the parameters for the message function.



Checking the Default Parameters

Note the message at the bottom of the window indicating that parameters have been defaulted. In most cases the default parameters shown are correct. However, you should review the information displayed before you accept the defaults.

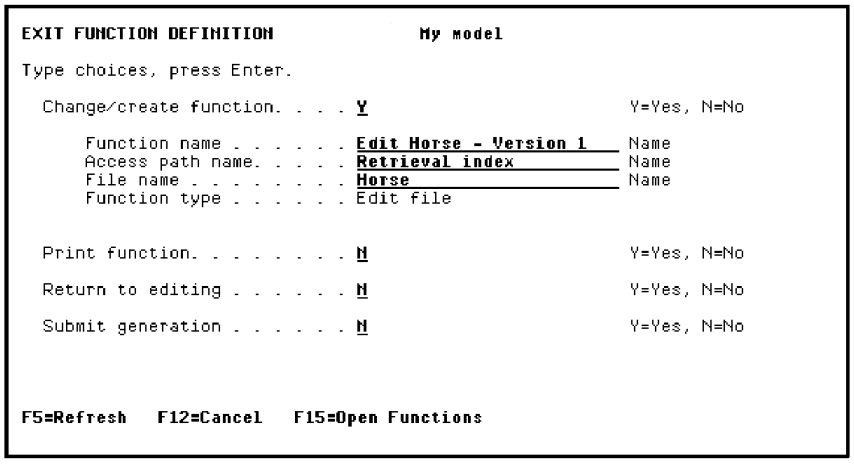
In this case, the defaults are correct; namely, Horse name is an input (I) parameter and the source of its value is the RCD (Subfile record) context. Press Enter to accept the defaults and return to the Action Diagram Editor.



Note that the call to the information message function, Racing results displayed, has been inserted in the action diagram.

Returning to the Action Diagram

Press F13 to exit and display the Exit Function Definition panel.



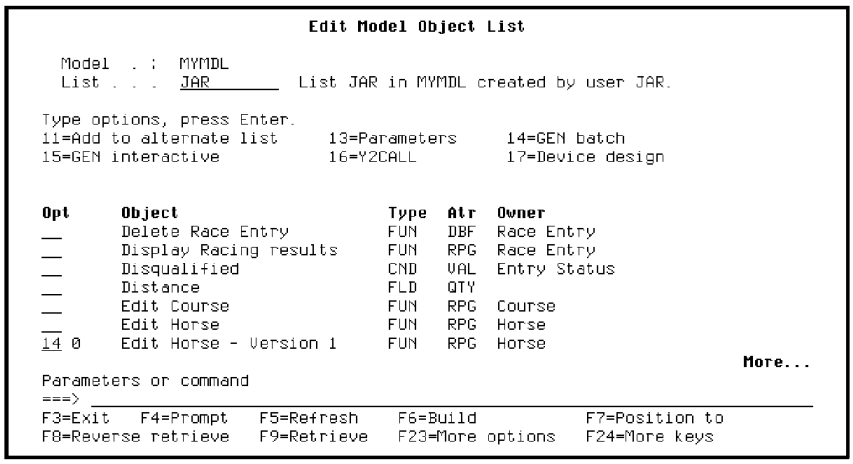
Press Enter to accept the defaults and return to the Work with Versions panel.

Press F3 to return to the Edit Model Object List panel. Note that a **0** displays to the right of the Subfile selector of the Edit Horse - Version 1 list entry to indicate that it is a non-current version.

Submitting the Function Version for Generation

You can submit generation requests and process job lists directly from the Edit Model Object List panel. Press F23 to see additional Subfile selector options. The appropriate option for submitting access paths and functions for batch generation is 14.

Type **14** in the Subfile selector for the Edit Horse - Version 1 function.

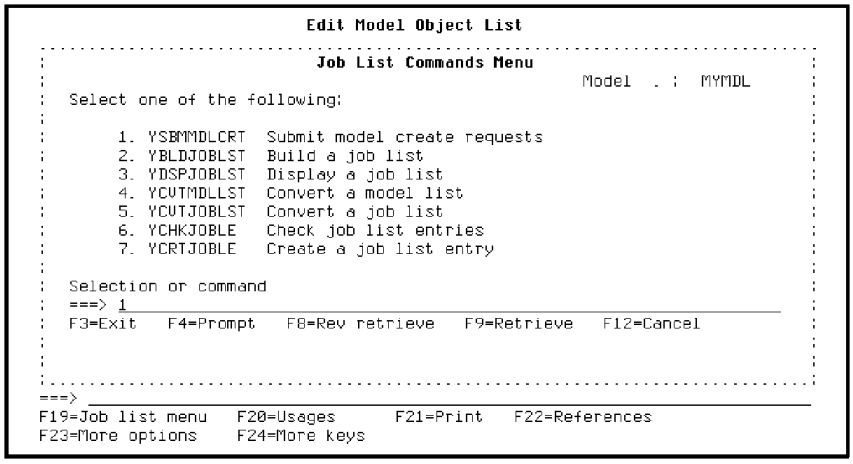


Press Enter. A message at the bottom of the panel indicates that the request for batch generation has been submitted.

Viewing Job List Commands

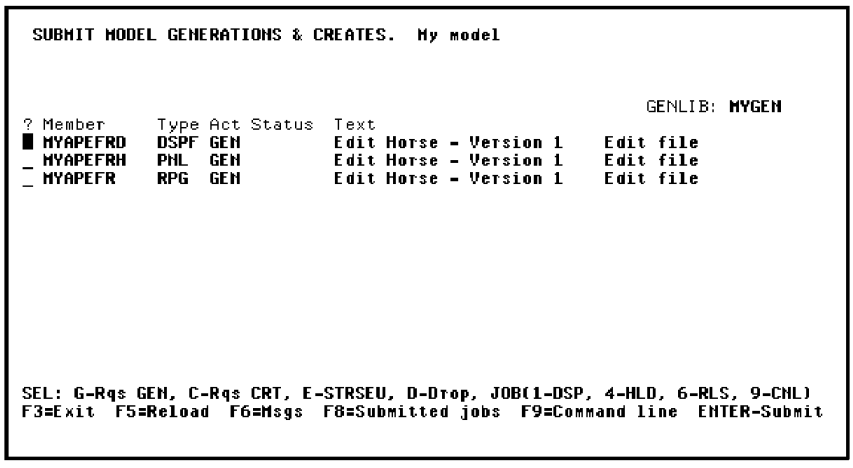
You also have access to a set of job list commands from the Edit Model Object List panel. This includes the Submit Model Create Requests (YSBMMDLCRT) command, which you previously accessed from the Display Services Menu. Press F24 twice to view additional command keys. The appropriate command key is F19.

Press F19 to display the Job List Commands Menu. Type **1** on the Selection line to select the YSBMMDLCRT command.



Press Enter.

When the prompt screen for the YSBMMDLCRT command displays, press Enter again to accept the defaults and display the job list.



Press Enter twice to submit the request to the job queue. Check the status of the job as described in the *Generating, Compiling, and Executing* chapter. Once the generation and compilation completes successfully, you can test Edit Horse - Version 1 to see if the new message displays.

Press F3 twice to return to the Edit Model Object List panel.

Displaying Alternate Views

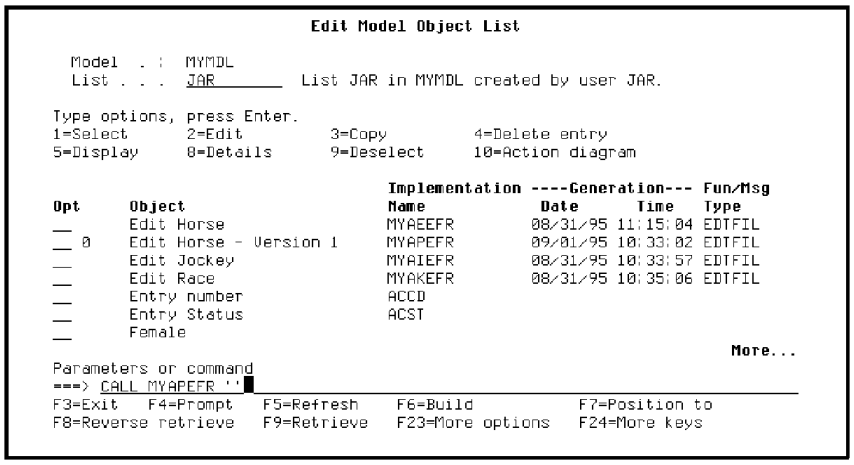
If you do not know the implementation name for Edit Horse - Version 1, you can use the F11 command key to view additional information for the model objects displayed. Five alternate views are available; however, this tutorial discusses only the first two. You can press F11 five times to see all five views.

Press F11 to display the alternate view showing the implementation name. Note the name that corresponds to Edit Horse - Version 1, MYAPEFR on the panel below.

Testing Edit Horse - Version 1

On the Command line, type the following to test Edit Horse - Version 1. Be sure to substitute the correct implementation name.

CALL MYAPEFR ' '



Press Enter.

The interactive panel for Edit Horse - Version 1 should display showing the data you entered earlier for the Edit Horse function.

Following are the steps needed to test your change.

1. Type **/** in the Subfile selector for any horse.
2. Press F10 to access the action bar.
3. Type **S** to access the Selector Choice menu.
4. Type **1** next to the Display Racing Results action.
5. Press Enter.

   Note: Because you will not enter data for the RACE and RACE ENTRY files until the *Advanced Functions* chapter, Display Racing Results will have no data to display; however, you should see the new information message on return to Edit Horse - Version 1.
6. Press F3 to return to Edit Horse - Version 1. The new message should display at the bottom of the panel.

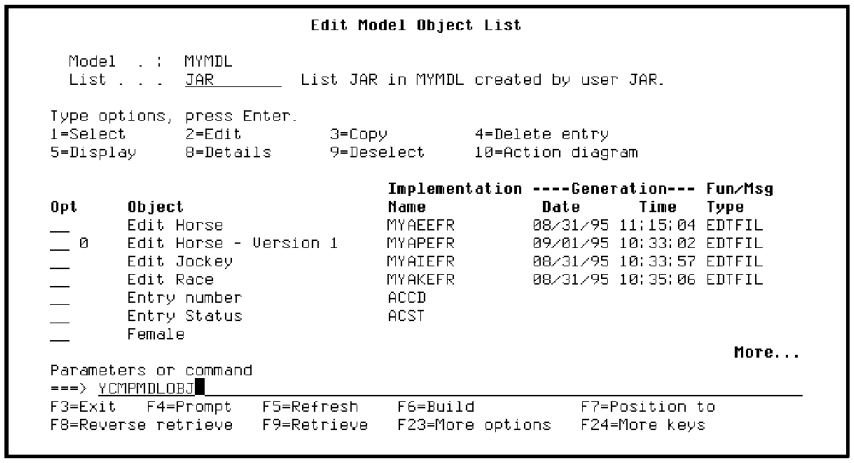
Press F3 to return to the Edit Model Object List panel.

Comparing Two Versions of a Function

You can compare two versions of a function using the Compare Model Objects (YCMPMDLOBJ) command. This command is useful for solving problems that arise while testing a new version and for identifying changes made to one version of a function for retrofitting to another version.

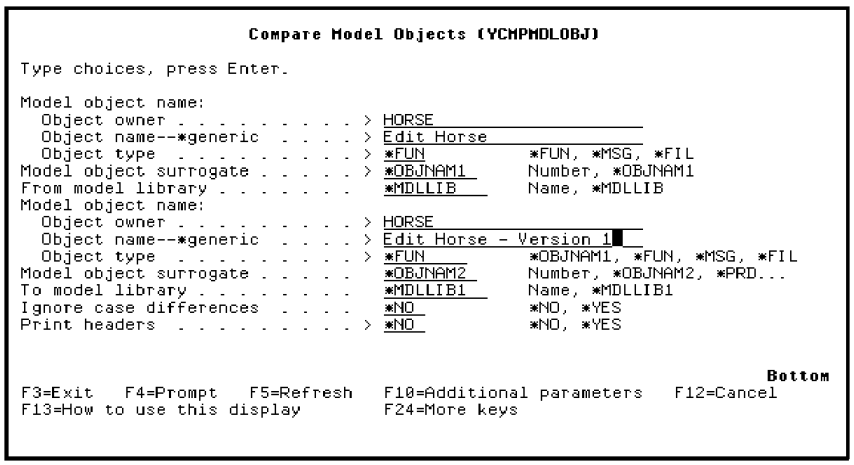
Note: You can also use this command to compare two functions or two messages. In other words, the model objects being compared need not be versions.

To compare two versions of a function, type YCMPMDLOBJ on the Command line.



Entering Parameters for YCMPMDLOBJ

Press F4 to prompt the command. Enter parameters to identify the functions to be compared; namely, enter the Object owner, name, and type of both functions. In this case, type **HORSE**, **Edit Horse**, ***FUN**, **HORSE**, **Edit Horse - Version 1**, and ***FUN** as shown. Accept the defaults for the other parameters.

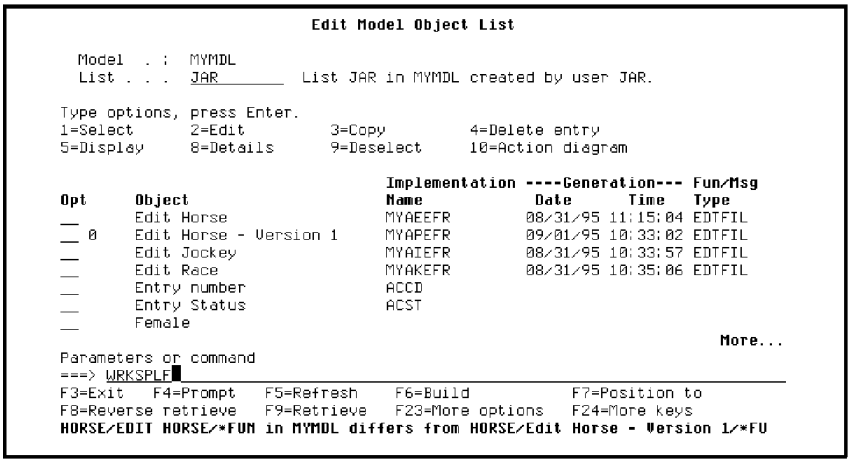


Press Enter.

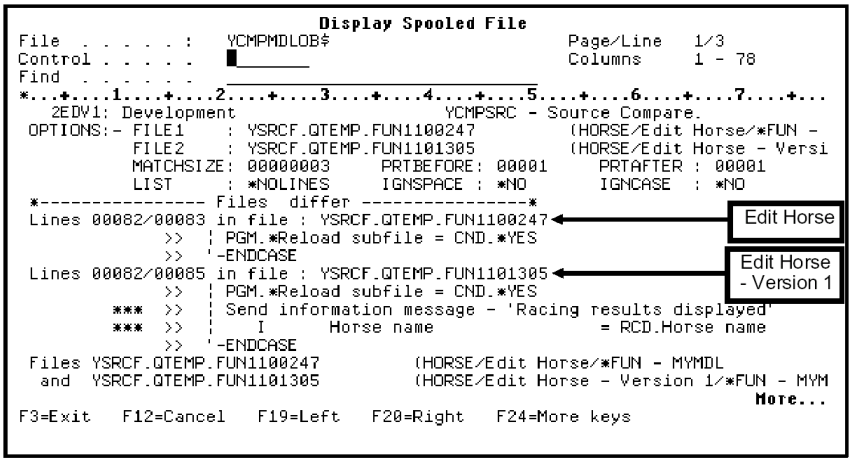
When the command completes, it returns automatically to the Edit Model Object List panel. If differences were found a message displays at the bottom of the panel.

Viewing Differences between Versions

Use the i OS Work with Spool Files (WRKSPLF) command to view the differences. Type WRKSPLF on the Command line.



Press Enter. Use Subfile selector option 5 to display the job named YCMPMDLOB$. The differences are shown as follows.

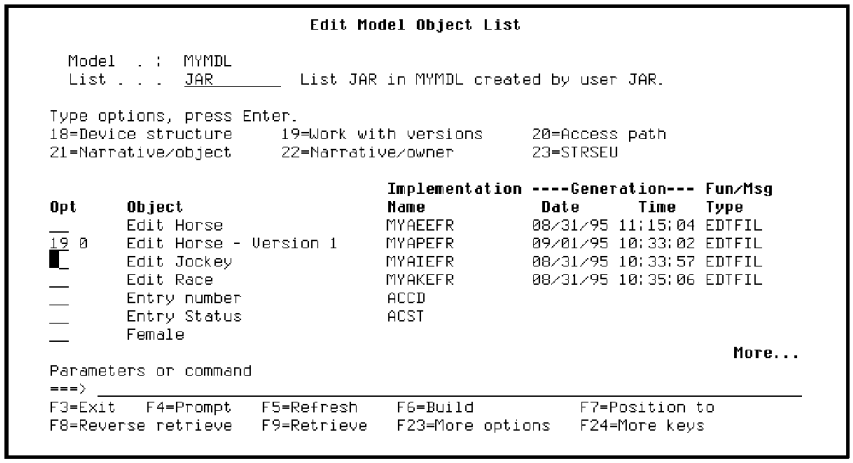


Press F3 twice to return to the Edit Model Object List panel.

Making the Edit Horse - Version 1 Current

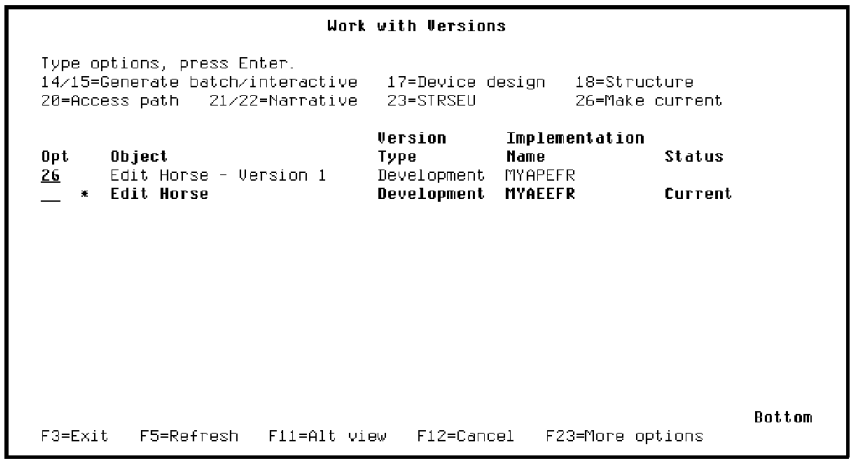
Once you have tested Edit Horse - Version 1 and are satisfied that both the new and old functionality work correctly, you can replace the original Edit Horse function in the model with the new version by making the version current.

First display the Work with Version panel by typing **19** in the Subfile selector for either version of the Edit Horse function.



Press Enter.

On the Work with Version panel, press F23 to view additional options. The appropriate option is 26, Make current. Type **26** in the Subfile selector for Edit Horse - Version 1.

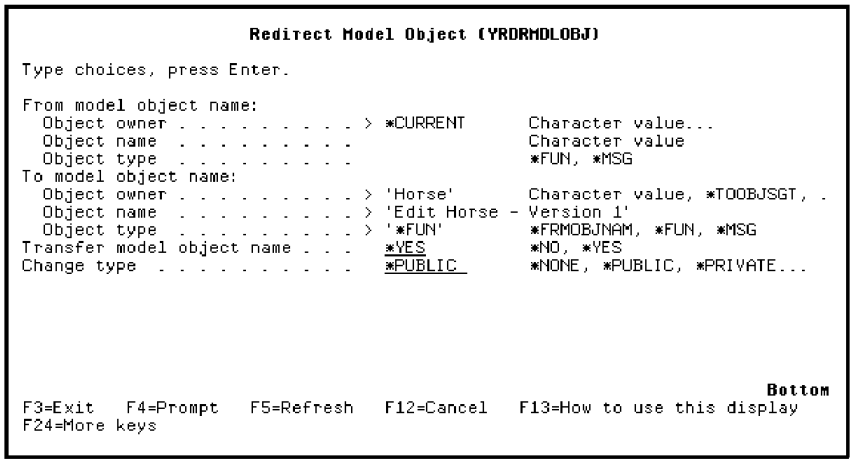


Press Enter.

Naming the New Current Version

The prompt screen for the Redirect Model Object (YRDRMDLOBJ) command displays.

When you make a version current, by default, the object names of the original and new current versions are exchanged; this is indicated by *YES in the Transfer model object name parameter. The implementation names of the two versions are always exchanged. To avoid misunderstanding, be sure to inform other developers when you make a new version current.



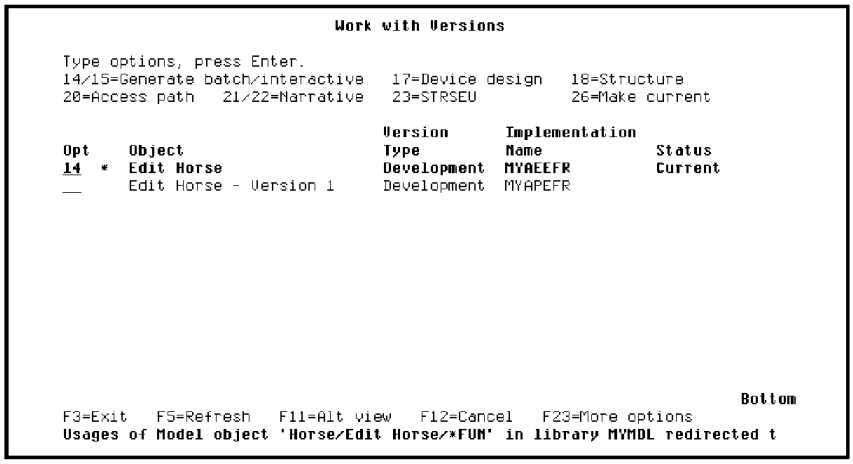
Press Enter to accept the defaults.

Note: If other objects in the model reference Edit Horse, they are automatically changed to reference the new current version instead. For example, if another function called Edit Horse,CA 2E would change the function so that it calls the new current version of Edit Horse instead. This process is known as *redirection*. Messages display at the bottom of the panel during redirection.

The new current version now contains the call to the information message. Its object name is now Edit Horse and its implementation name is MYAEEFR.

Regenerating the New Current Version

You need to generate and compile the new current version so the source and program object contain the new functionality. Type **14** in the Subfile selector as shown to submit a request for batch generation.



Press Enter to submit the request. Press F3 to return to the Edit Model Object List panel. The 0 to the right of the Subfile selector of the Edit Horse list entry indicates that it is a non-current version.

Exercise

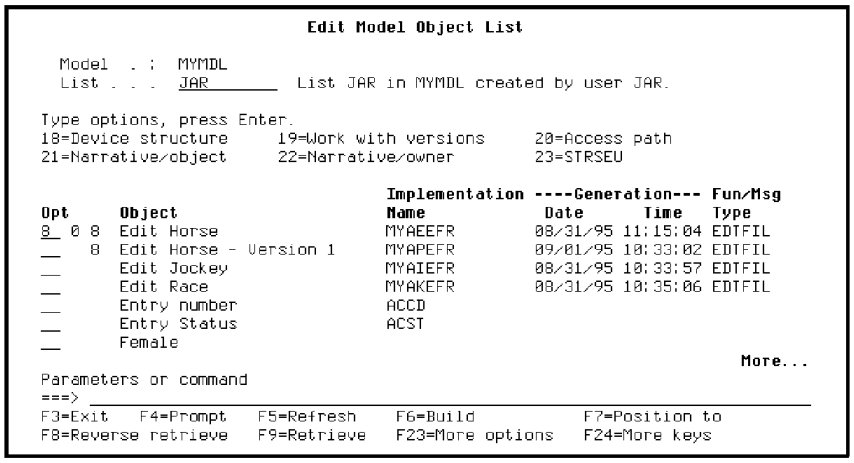
Submit the batch job using the procedure you used previously in this topic. Press F19 to display the Job List Commands menu and select the YSBMMDLCRT option. When you finish, return to the Edit Model Object List panel.

Viewing Model Object Information for the Versions

Note that an 8 appears to the left of both the Edit Horse and Edit Horse - Version 1 list entries. This indicates that the list entry information does not match that of the corresponding model object. In this case, the object names and implementation names are mismatched since they were exchanged during the process of making Edit Horse - Version 1 current.

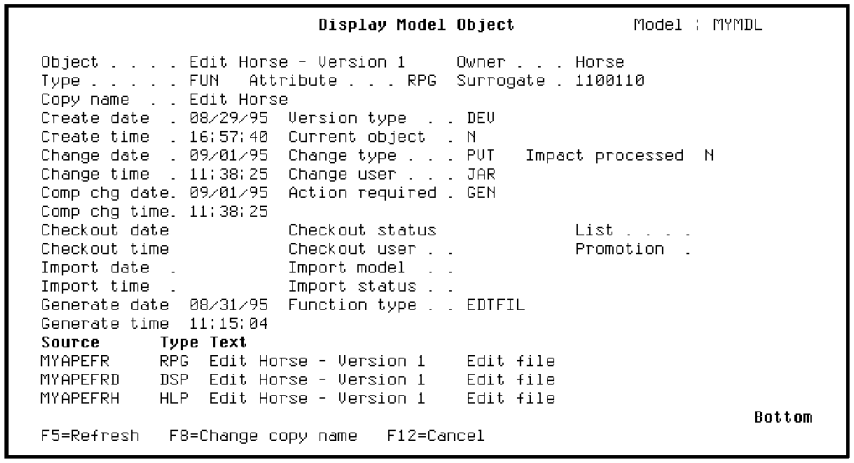
Note: This difference is allowed to occur so you can use your model object list as a historical record.

To view current details for a model object, use Subfile selector option 8. (Note that you can press F23 to see other options.) For example, type **8** against the Edit Horse list entry.



Press Enter.

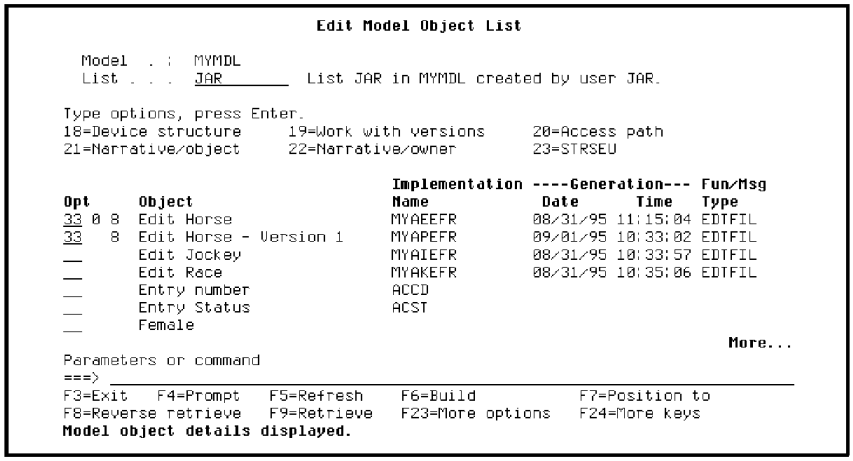
Note that the Object Name, Edit Horse - Version 1, and the implementation name shown in the Source column, MYAPEFR, do not match the corresponding information shown on the Edit Model Object List panel for the Edit Horse list entry.



Press F12 to return to the Edit Model Object List panel.

Refreshing List Entries for the Versions

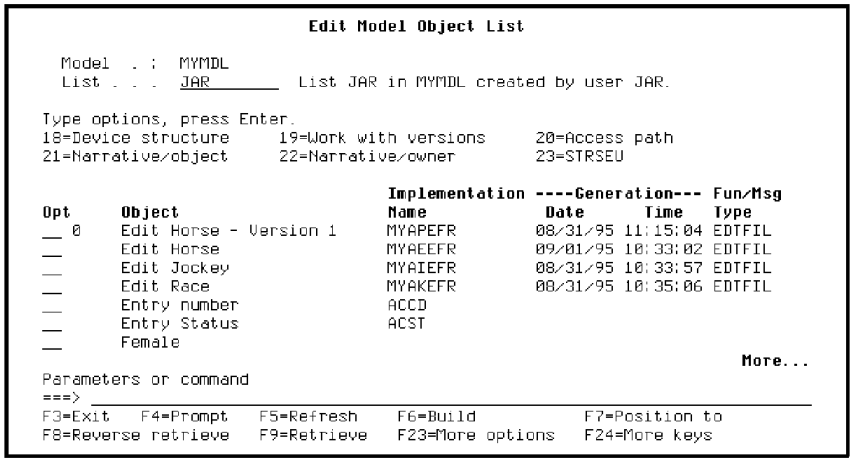
To update the two list entries to match the actual model objects, type **33** in the Subfile selector for both entries.



Press Enter.

Note: If many discrepancies exist between entries on a model object list and the actual model objects, you can refresh the entire list by pressing the F15 command key. This executes the Check a Model List (YCHKMDLLST) command.

The information for the list entries for the two versions now match that of the corresponding model objects.



Exercise

Use the Position the List panel to display all entries for your session list.

Note: Press F7, blank out all the fields, and press Enter.CA 2E automatically repositions your session list at the top in order by Object Name and Object Type.

Model Object Cross References

This topic introduces the CA 2E model object cross reference panels.

New terms introduced

- Model object cross reference
- Expansion
- Usages
- Using objects
- References

New panels introduced

- Display Model Usages
- Display Model References

Objectives

To use CA 2E's model object cross reference facilities, including the Display Model Usages panel and the Display Model References panel.

Overview of Model Object Cross References

CA 2E's *model object cross reference* facilities consist of a set of commands and interactive panels that you can use to determine, for any model object, which other objects it references and which other objects it is used by.

The process of determining either usages or references for a model object is know as *expansion*. Using model object cross reference facilities, you can expand usages or references for a model object to any level.

You can display or print model object usages and references; you can also convert them to a model object list.

Model Object Usages

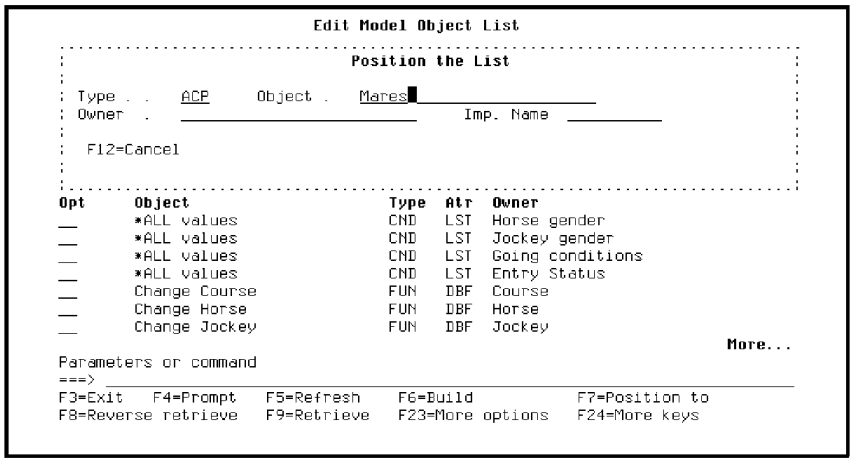
Usages for a model object are all the model objects that use it. Usages are external to the model object and require the model object in order to be complete. A model object's usages are sometimes referred to as *using objects*.

Suppose you want to change the Mares access path. Before you do so, you need to determine which model objects use it. In this step you will use the Display Model Usages panel to display usages for the Mares access path.

Accessing Your Session List

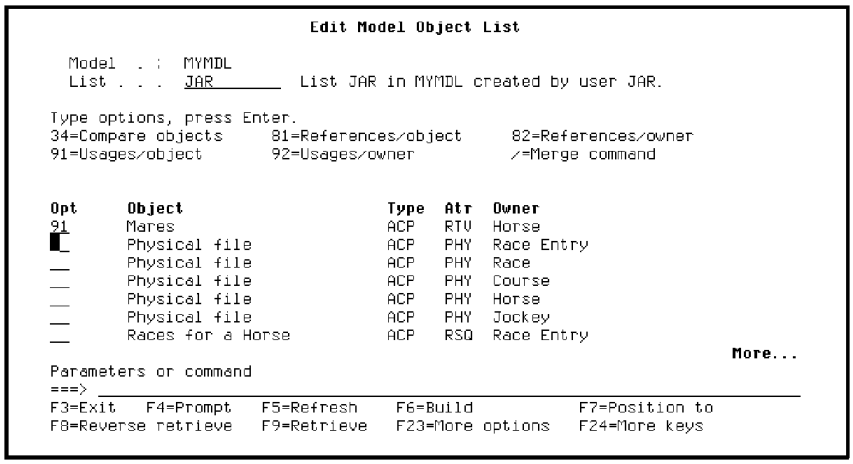
Start by accessing the Edit Model Object List panel for your session list. Use one of the methods you used earlier in this chapter.

First position the session list to the Mares access path. Press F7 to display the Position the List window. Type **ACP** for the Type field and **Mares** for the Object field.



Press Enter. Your session list is now displayed in order by Object Type and Object Name beginning with the Mares access path.

Press F23 until option 91 is displayed. Option 91 displays usages for the selected object. Type **91** in the Subfile selector for the Mares access path.



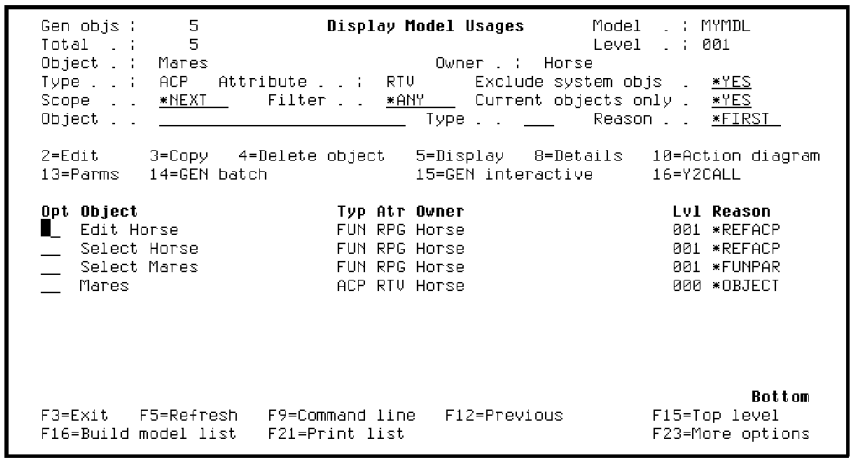
Press Enter. Notice the messages at the bottom of the panel as CA 2E expands model usages.

Display Model Usages Panel

When all usages have been expanded, the Display Model Usages panel displays showing the first level of objects in the model that use the Mares access path. Notice the 001 in the Lvl column for each using object; this indicates that the object uses the Mares access path directly.

The Display Model Usages panel provides a variety of controls and filters including, recursion, scope, and positioning to help you analyze your model to determine the impact of proposed changes.

Note: Most of the options provided on the Display Model Usages panel are beyond the scope of this tutorial. However, you can use the online help if you would like to experiment further. Or, you can refer to the CA 2E guide, *Generating and Implementing Applications*.



Usage Reason

The Reason column shows how each of the listed model objects use the Mares access path. For example,

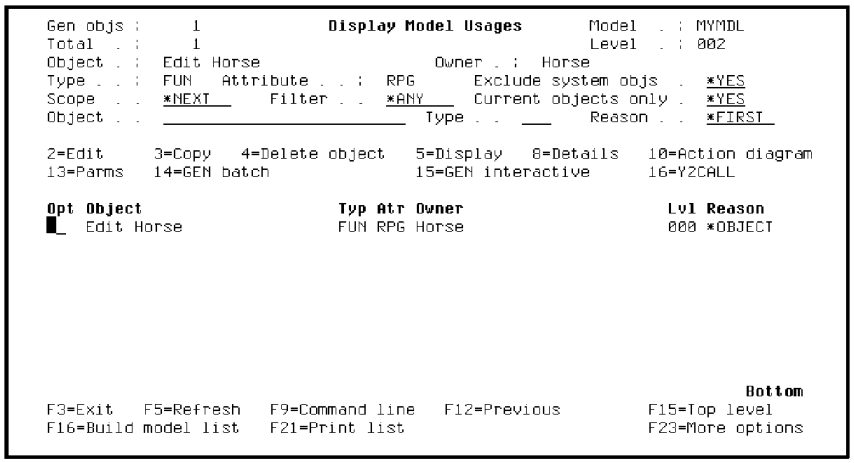
- *REFACP - The Edit Horse and Select Horse functions use Mares as a referenced access path.
- *FUNPAR - The Select Mares function uses Mares as the definition of a function parameter.
- *OBJECT - This identifies the model object for which you requested usages. This object is included on the list of usages so you can use the Subfile selection options on the original model object.

The model objects that use the Mares access path are all external functions that will at least need to be regenerated and compiled if you change the Mares access path. However, if your proposed change requires relations to be added to or removed from the Mares access path, you might also need to edit these functions.

In this model no other model objects use the using objects for Mares. You can test whether other objects use the objects displayed by typing **91** in the Subfile selector for any of the using objects.

Exercise

Type **91** against Edit Horse and press Enter to display its usages. The Display Model Usages panel redisplays showing only the model object itself. This indicates that no other model objects use Edit Horse; in other words, it has no usages.



Using Usage Levels

Notice that the Level number in the upper right corner has changed to 002. This indicates the number of times you have expanded usages beginning with the original model object.

For a more complex example, a model object may have many levels of using objects. The value of *NEXT in the Scope field lets you step through the expansion of usages one level and one model object at a time. In other words, you can type **91** for any using objects to determine its usage. The Level shown at the upper right of the screen shows the current level. The F3 key lets you back up one level at a time; the F15 key returns to the top level.

Note that you can change the Scope field to *NOMAX to expand all usages for a selected model object. However, for working models this can require significant processing time.

Exiting Model Usages

Press F3 twice to return to the Edit Model Object List panel.

Press F7 to display the Position the List window. Blank out all fields and press Enter to reposition the session list and display all entries.

Model Object References

References for a model object are the model objects it refers to internally. In other words, references are the model objects the referring model object requires in order to be complete or to exist. For example, Display Racing results and Change Horse are references of the Edit Horse function.

This step shows how to access the Display Model References panel and how you can use it to solve problems in program applications.

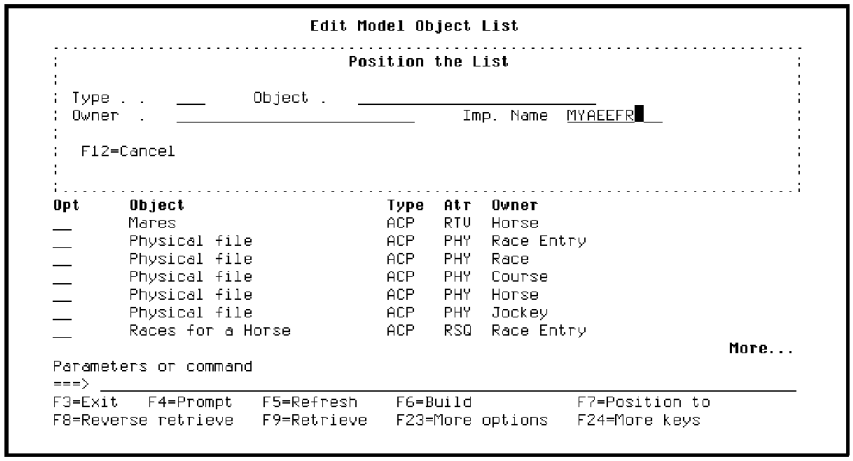
Accessing the *ALLOBJ List

Start by accessing the Edit Model Object List panel for the All Objects list (*ALLOBJ). Since your session list is currently displayed, type ***ALLOBJ** for the List field at the top of the panel and press Enter.

Positioning *ALLOBJ to an Implementation Name

Suppose you only know the implementation name of the program in which an error occurred; for example, MYAEEFR (Edit Horse).

From the Edit Model Object List press F7. Type the implementation name of the program in the Imp. Name field as shown. Be sure to substitute the name corresponding to your Edit Horse function.

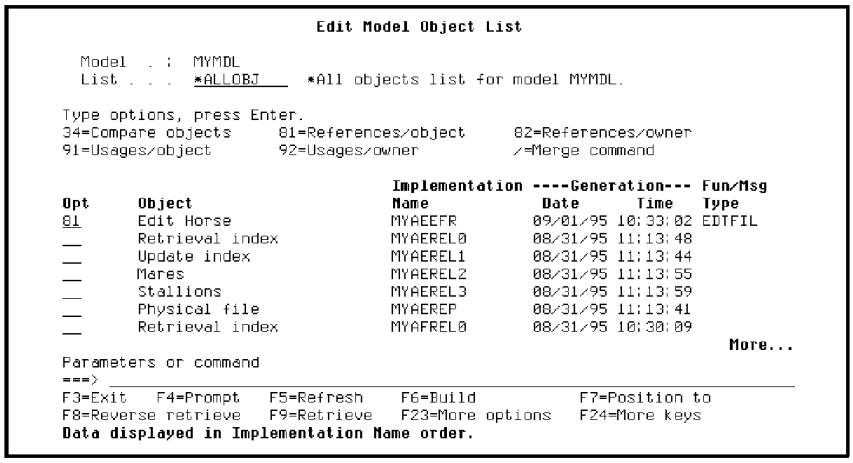


Press Enter.

The *ALLOBJ list is now positioned at the Edit Horse function. Note the message at the bottom of the panel and that the alternate view showing the Implementation name displays automatically. You can change views by pressing F11.

Displaying References for Edit Horse

Option 81 lets you display references for a selected model object. Type **81** in the Subfile selector for Edit Horse.

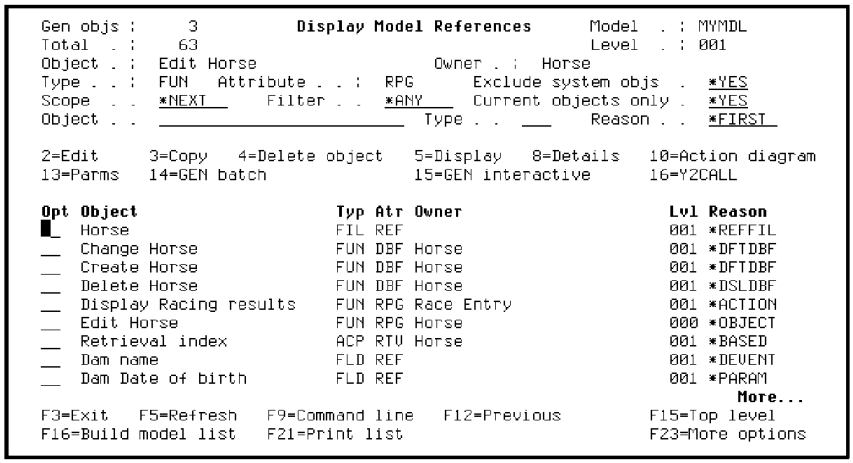


Press Enter. Notice the messages at the bottom of the panel as CA 2E expands the model references.

Display Model References Panel

When all references have been expanded, the Display Model References panel displays showing all objects in the model referenced by Edit Horse. This panel provides a variety of controls and filters including, recursion, scope, and positioning to help you analyze your model.

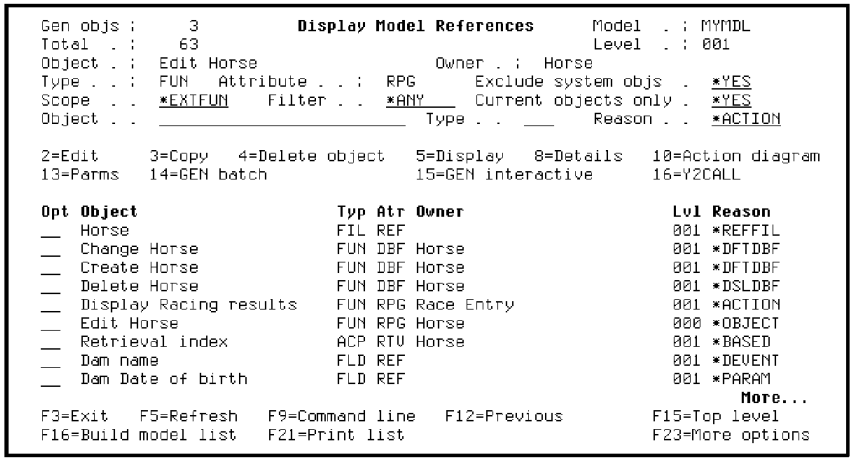
Note: Most of the options provided on the Display Model References panel are beyond the scope of this tutorial. However, you can use the online help if you would like to experiment further. Or, refer to the CA 2E module, Generating *and Implementing Applications*.



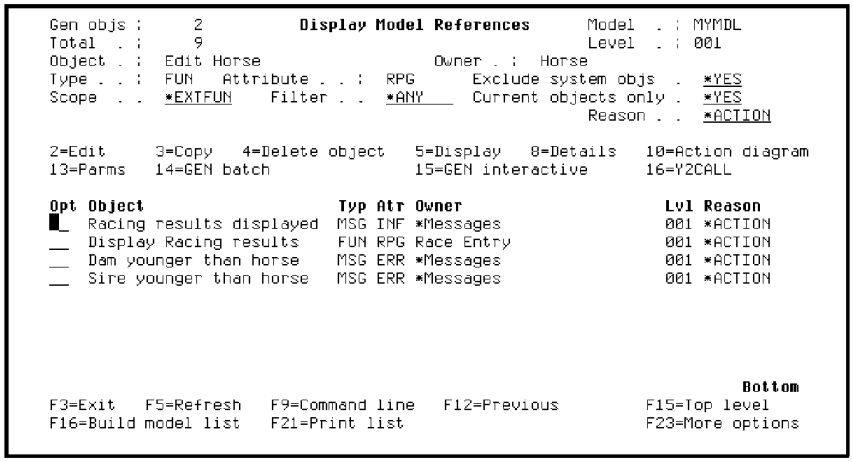
Displaying Only External Functions

Since you are using references to solve a problem in an application program, you only need to see external functions that are referenced in the Edit Horse action diagram.

Type ***EXTFUN** for the Scope field and type ***ACTION** for the Reason field.



Press Enter. Only the functions that make up the program in which the error occurred are now displayed. In a working model this list would generally be much longer. This is a useful starting place for the developers whose task it is to fix the problem.



Creating a Model List of the References

In this step you will use the F16 command key to create a model object list and convert the references displayed to model list entries. You can specify the name of an existing list or you can type a new name and CA 2E will automatically create the model list.

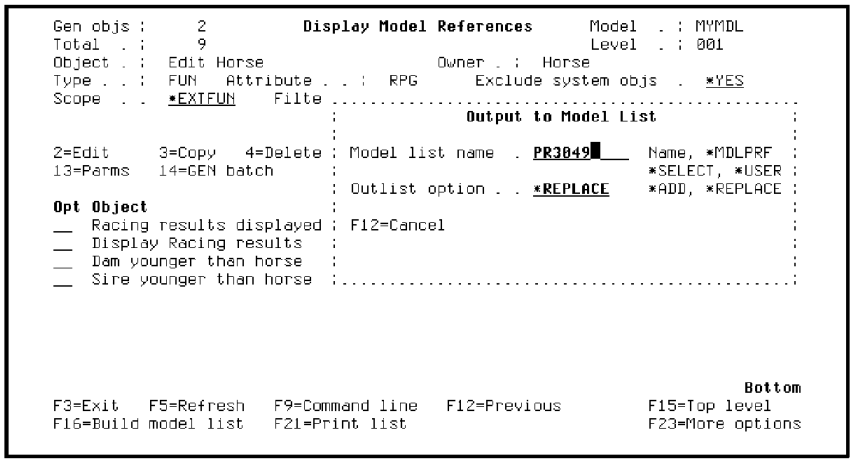
If you specify the name of an existing list be sure the Outlist option is correct. A value of *REPLACE clears the list before converting the references; a value of *ADD appends the new entries to any existing entries.

Naming the List of References

You should assign a name to the list that will be meaningful to the development staff. By default the model list name is the Model list for commands specified in your model profile.

Press F16 to display the window where you will enter the name of the model list that is to contain the references. Suppose you are responding to Problem Report 3049; as a result, you might name the model list PR3049.

Type **PR3049** in the Output to Model List window.



Press Enter. Notice the messages at the bottom of the window as the references are converted to entries on the model object list. On return to the Display Model References panel a message at the bottom of the panel verifies that the conversion is complete.

Using the List of References

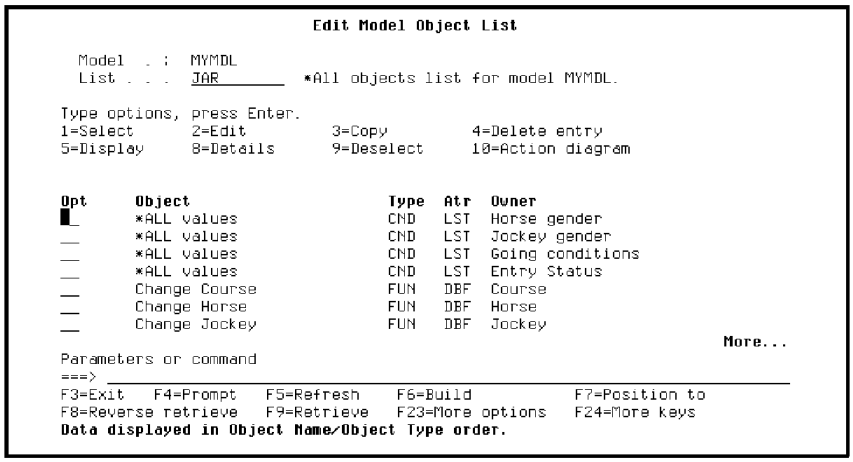
You can now print the PR3049 model list using the F21 command key. You can either give the printed copy of the list to your development staff or they can use the model list online as an aid to solving the problem.

Exiting Display Model References

Press F3 to return to the Edit Model Object List panel. Reposition the list to the top in Object Name/Object Type order as follows: press F7 to display the Position the List window, blank out all fields, and press Enter.

Redisplay Your Session List

Type the name of your session list in the List field at the top of the panel. Recall that by default your session list has the same name as your user profile.



Press Enter.

Impact Analysis

This topic introduces the CA 2E impact analysis tools, including simulating a proposed change and component change processing.

New terms introduced

- Change type
- Private change
- Public change
- Simulating a change
- Component change processing

Objectives

To show how to determine the impact of a proposed change to a model object on other objects in the model.

Overview of Impact Analysis

Impact analysis lets you determine the impact of a proposed change or an actual change to any object in your model. For example, you can determine which objects are affected if you change the length of a key field or which functions need to be regenerated if you change logic in the action diagram of an internal function.

Change Type

Whenever you change a model object,CA 2E assigns a *change type*. The change type describes the way in which a change impacts other objects that use the changed object. The four possible change types are Object only (*OBJONLY), Generation required (*GEN), Private (*PRIVATE), and Public (*PUBLIC). The first two indicate changes that affect only the changed object itself. As a result, this tutorial covers just the last two change types.

- *PRIVATE - this change type indicates a change to an object that requires that you regenerate and compile all access paths and external functions that use it.

  For example, if you change the action diagram of an internal function, you need to regenerate the external functions that call it.

- *PUBLIC - this change type indicates that the interface of the model object with other objects has changed. As a result, some model objects that use the changed object may require editing. When the editing is complete, you need to regenerate the access paths and external functions that use the changed object.

  For example, if you change the parameters of an internal function you need to edit all functions that call it and then regenerate all external functions that use it.

The change type depends on which attributes of a model object are changed and is derived internally by CA 2E.

Simulating a Change to a Model Object

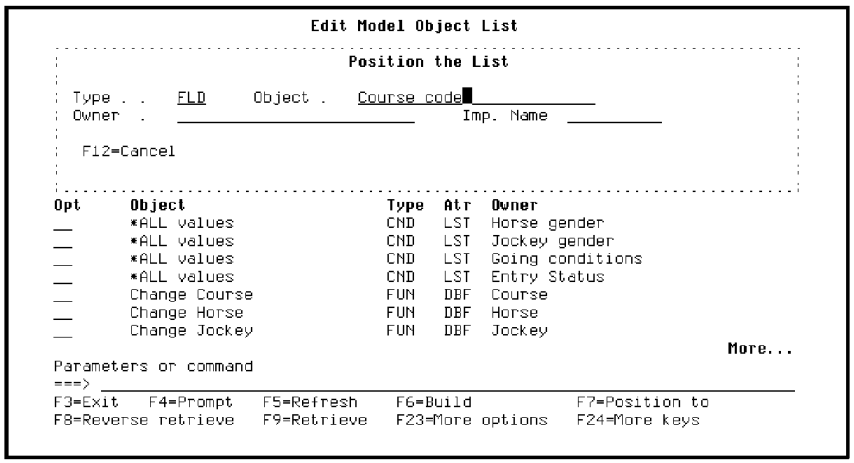
Simulating a change to a model object lets you see the impact of a proposed change on other objects in the model before you actually make the change. Simulation identifies which other model objects need to be edited or generated as a result of the proposed change.

When you change a model object, the only objects that can be affected by the change are those that *use* the changed object. As a result a major part of simulating a change consists of expanding usages for the object to be changed.

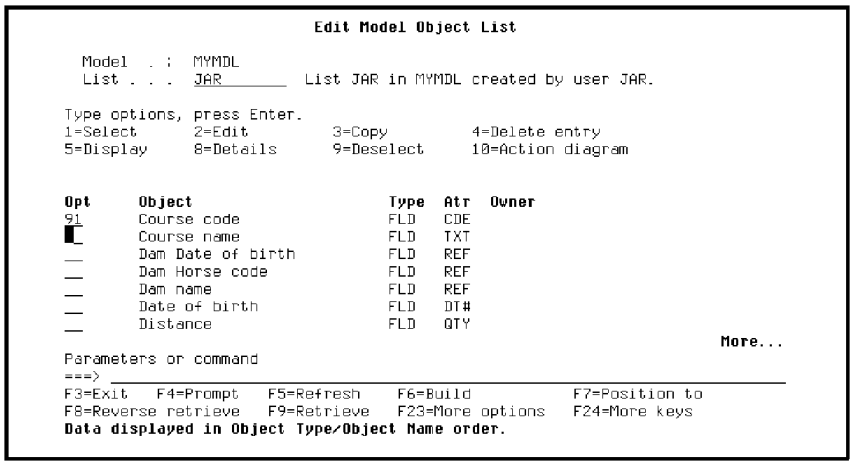
Suppose you want to change the length of the *Course code* field. This is a *PUBLIC change; in other words, some using objects will need to be edited to incorporate the change.

Positioning the List to Course Code

From the Edit Model Object List panel, press F7 to display the Position the List window. Type **FLD** for the Type field and type **Course code** for the Object field.



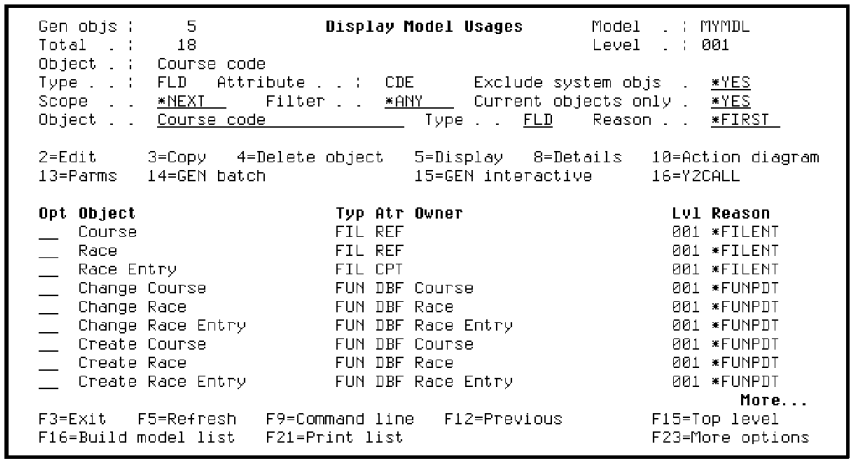
Press Enter. Type **91** in the Subfile selector for *Course code* as shown to display the Display Model Usages panel.



Press Enter.

Positioning the Usages to Course Code

To simulate a change to Course code, you first need to locate it. Since Course code has many usages, an easy way to locate it is to use the positioning options. Type **Course code** in the Object field and type **FLD** in the Type field.

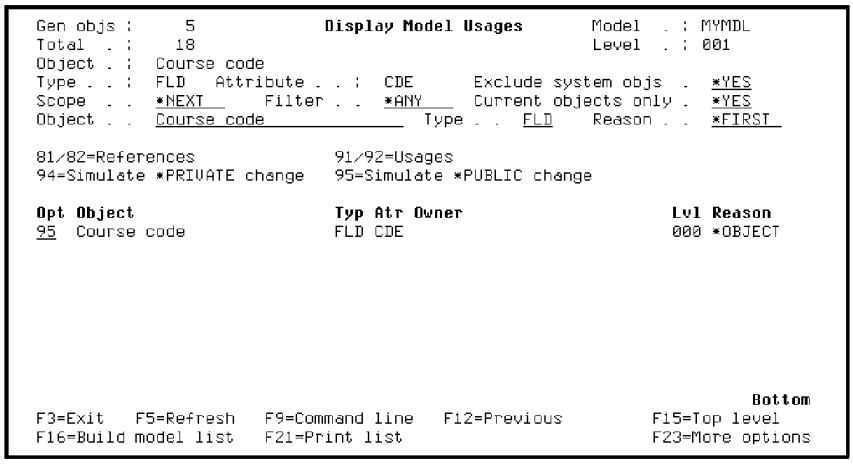


Press Enter.

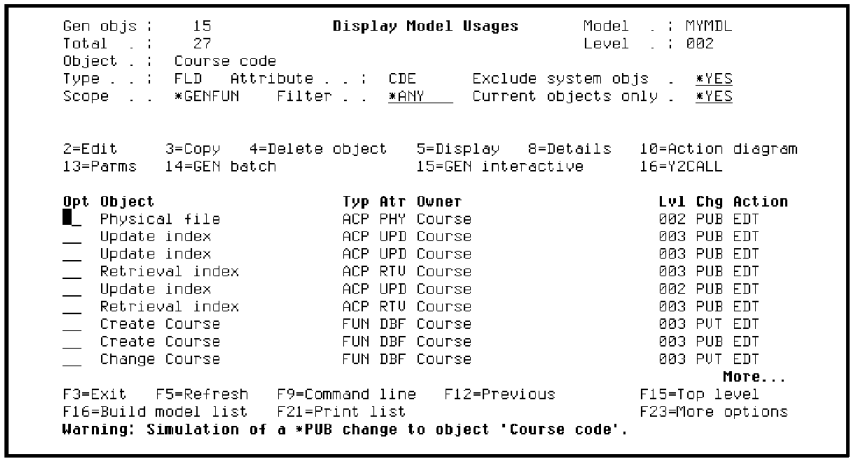
Simulating a *PUBLIC Change

Press F23 to display additional Subfile selector options. Option 95 lets you simulate a *PUBLIC change for the selected model object.

Type **95** in the Subfile selector for Course code.



Press Enter to display all objects that will be affected by a *PUBLIC change to Course code.



Interpreting the Results

When you simulate a change,CA 2E expands usages for the object to be changed up to the first external function for each sequence of model objects that use it. This is indicated by *GENFUN in the Scope field. In addition, only those objects that need to be edited or generated to implement the proposed change are displayed, not all usages.

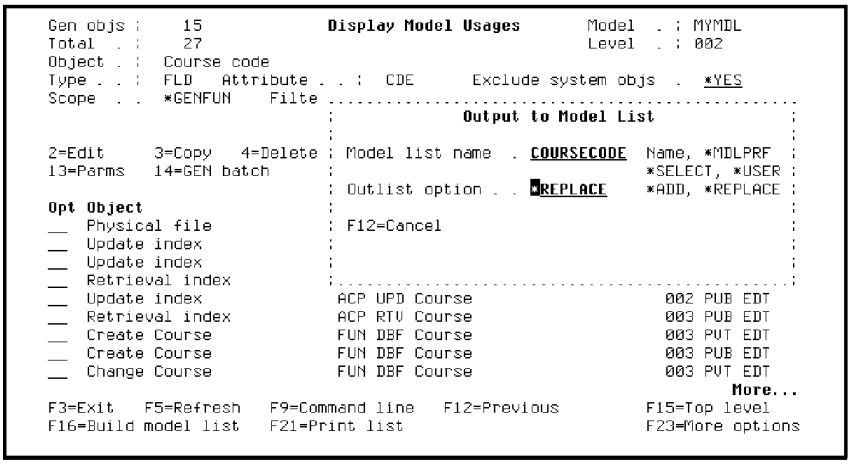
Note the numbers in the Lvl column; they indicate the usage level for each using object. The Action column indicates whether the using object needs to be edited (EDT) or regenerated (GEN).

As you scroll through the list, do not be concerned about what appear to be duplicate entries; they do not indicate errors.

Converting the Simulation Usages to a List

In order to further study the impact of your proposed change, convert the usages displayed to a model object list. Be sure to give it a descriptive name; for example, COURSECODE.

Press F16 and type **COURSECODE** in the Model list name field in the window that displays as shown. Since this is a new list you can accept the default for the Outlist option.



Press Enter. Notice the messages at the bottom of the window as CA 2E creates the list and converts the displayed usages to model list entries.

Press F3 twice to return to the Edit Model Object List panel. Use the F7 key to reposition the session list to the top and display all entries.

Optional Exercise

Type **COURSECODE** in the List field and press Enter to display the model object list you just created. Use the F11 key to view information for each list entry.

When you finish viewing this list, type the name of your session list in the List field and press Enter.

Component Change Processing

Component change processing refers to an automated impact analysis tool. When you simulate a change you are in effect simulating component change processing. The difference is that component change processing updates the All Objects list for the changed object and all objects affected by the change, including an indication of whether the object needs to be edited or regenerated.

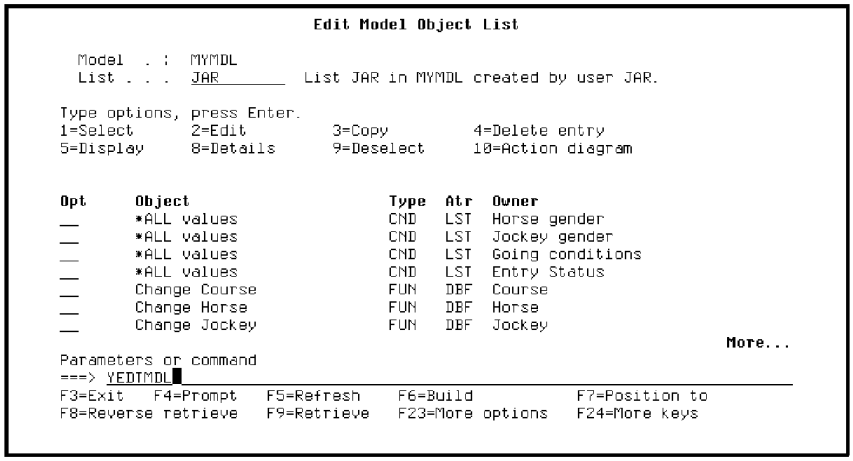
Component change processing is optional; you control it using the model profile and the Component Change Processing (YCMPCHG) model value. You can choose to run it interactively, by request, or in batch.

Further discussion of this tool is beyond the scope of this tutorial; however, the simulation topic should have given you some familiarity with the process. Component change processing is discussed in detail in the CA 2E module, *Generating and Implementing Applications*.

Accessing Edit Database Relations

At this point you can exit the model by pressing F3. If you want to continue with the tutorial, you can use either the Edit Model (YEDTMDL) command or the Start CA 2E (YSTRY2) command to reenter the model at the Edit Database Relations panel.

For example, type YEDTMDL on the command line.



Press Enter.

Chapter 7: Advanced Functions

This chapter demonstrates the use of the Edit Transaction (EDTTRN) function with a Span (SPN) access path and the use of function fields. This chapter introduces the following topics.

- The Edit Transaction Function Type
- Span Access Path
- Defining an Edit Transaction Function
- Using Function Fields
- Defining Function Field Parameters

This section contains the following topics:

Span Access Path and Edit Transaction Function (see page 331)
Introduction to Function Fields (see page 349)

Span Access Path and Edit Transaction Function

This topic demonstrates the use of the Edit Transaction (EDTTRN) function with a Span (SPN) access path.

New terms introduced

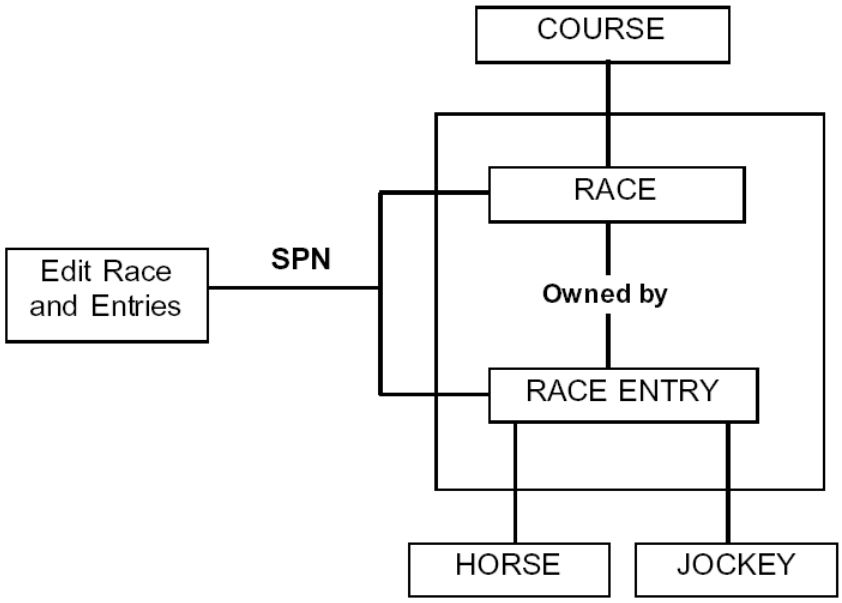
- Access path format
- New panels introduced
- Display Access Path Formats

Objectives

You will specify a SPN access path based on the RACE and RACE ENTRY files.

Overview

The Edit Transaction function allows two database files to be updated simultaneously. The Edit Transaction function type requires a SPN access path, which is a logical file with two formats.



In order for a SPN access path to be built over two files, the two files must be connected by a file-to-file relation; for example, an Owned by relation. The Edit Transaction function provides a display consisting of a subfile control format for the owning file and a subfile record format for the owned file.

You will set up an Edit Transaction function for the RACE and RACE ENTRY files. This provides the facility to enter both the race details and the race entry details for a given race on the same interactive panel.

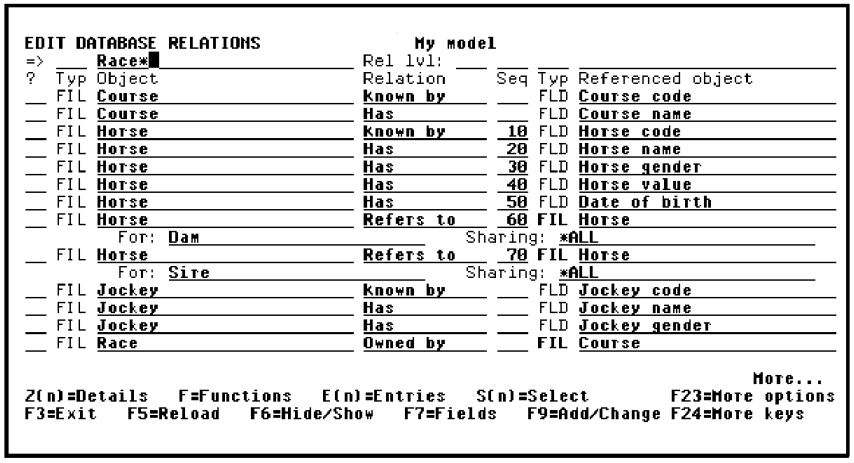
Overview of Processing Steps

To implement the Edit Transaction function you will:

1. Define and generate a SPN access path for the RACE and RACE ENTRY files.
2. Create the Edit Transaction function and modify the device design for the Edit Race and Entries function.
3. Generate source for the function.
4. Compile the access path, display file, and program source.

Entering Your Design Model

Return to the Edit Database Relations panel for your design model. Select the RACE and RACE ENTRY files by typing **Race*** on the selection line.

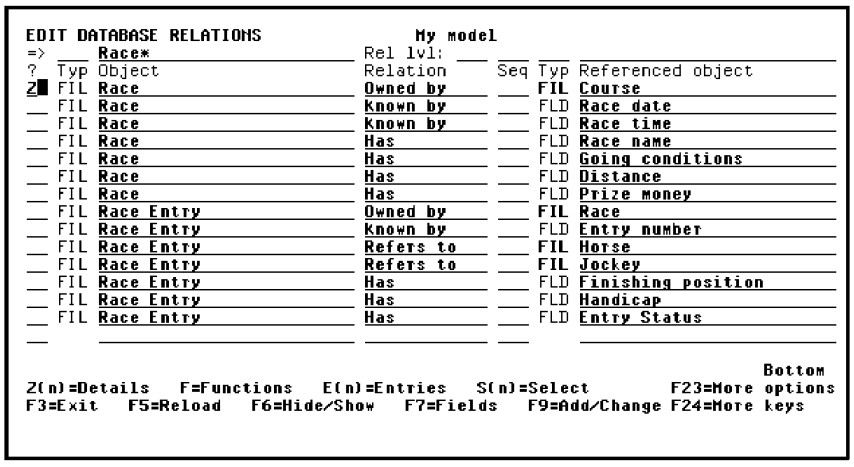


Press Enter.

Creating a Span Access Path

Create a SPN access path on the RACE file. A SPN access path must be attached to the file that is to form the header format of the access path. In this case, you will create the SPN access path over the RACE file.

Type **Z** against any relation on the RACE file to display all existing access paths.



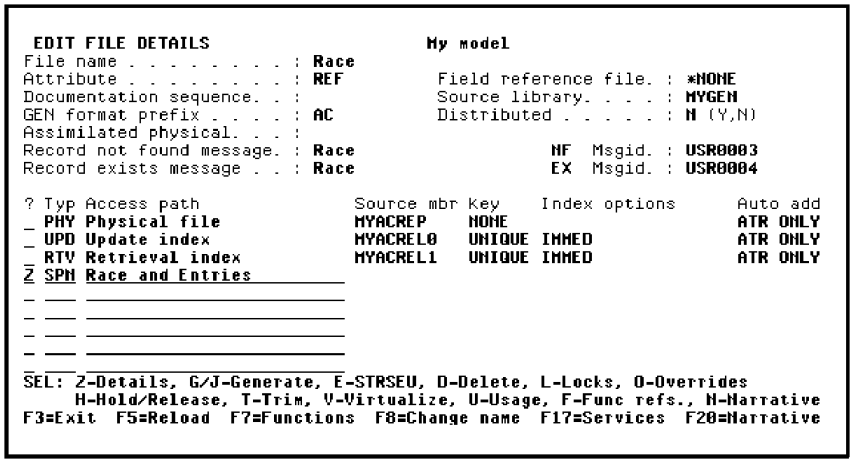
Press Enter. The Edit File Details panel displays.

Defining the Span Access Path

To specify the SPN access path you need to:

1. Define two access path file formats, one for the subfile control (header) and one for the subfile record (detail)
2. Specify the keys for those formats

Define the new access path by calling it Race and Entries of type SPN. Type the access path type, name, and **Z**.



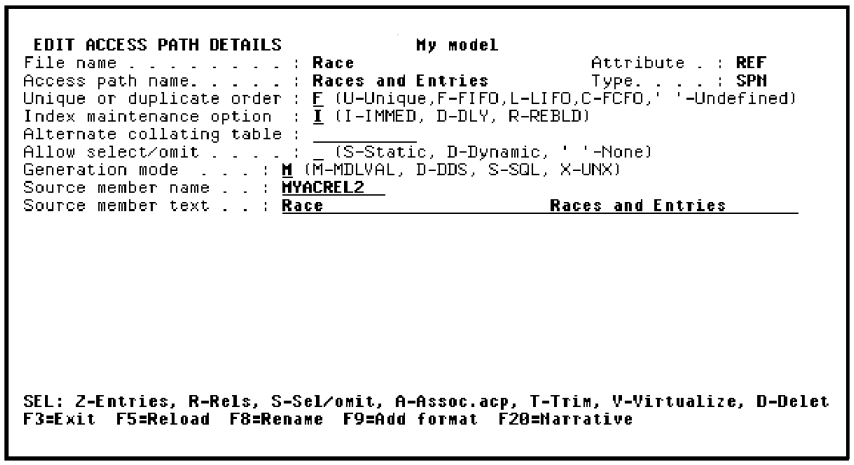
Press Enter. The Edit Access Path Details panel displays.

Entering the Access Path Formats

An access path format defines a view upon a physical file. By default, it makes all of the non-virtual entries in that file available. An access path format can be defined as a list of access path relations that will be resolved into a list of field entries. Relations can be removed from a view on the Edit Access Path Relations panel.

When you previously used the Edit Access Path Details panel, the access path format was automatically provided. For the SPN access path you must explicitly add the formats. Note that it is important that you add the formats in the correct sequence: header format first, detail format second.

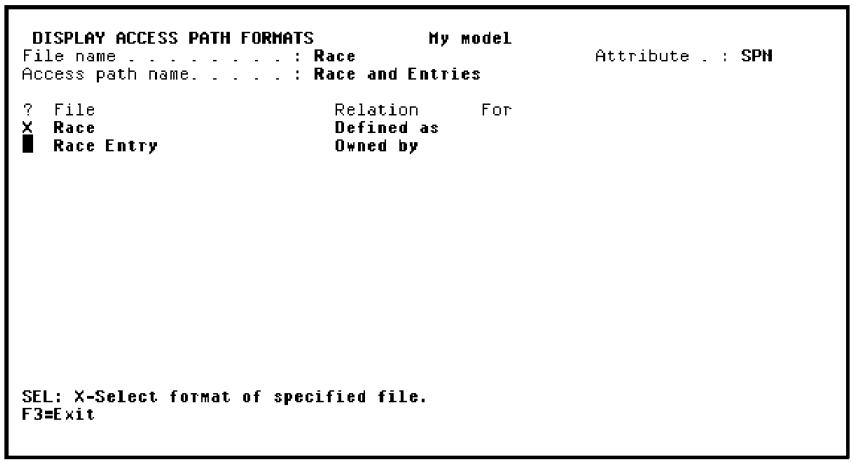
Press F9 to display the Display Access Path Formats panel where you will add the format for the subfile control (header).



Selecting the First Access Path Format

The Display Access Path Formats panel shows you the files that you can select for the SPN access path. In this case, the two files shown are connected to the RACE file by an Owned by and a Defined as relation.

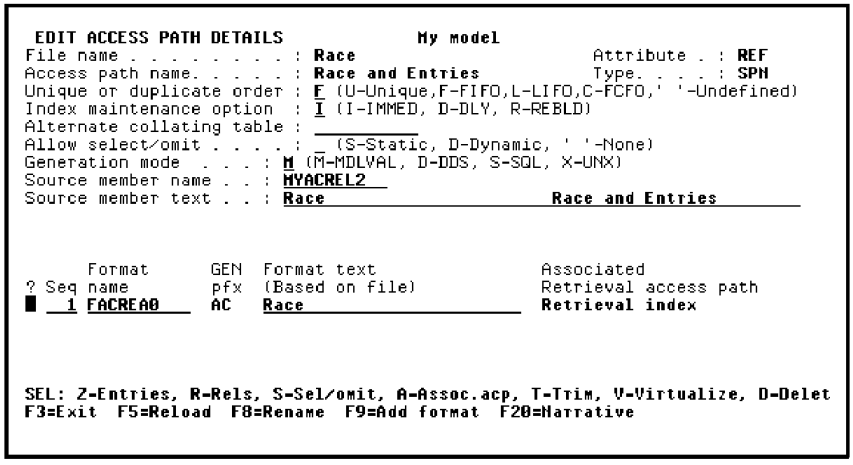
Select the RACE format with **X**.



Press Enter.

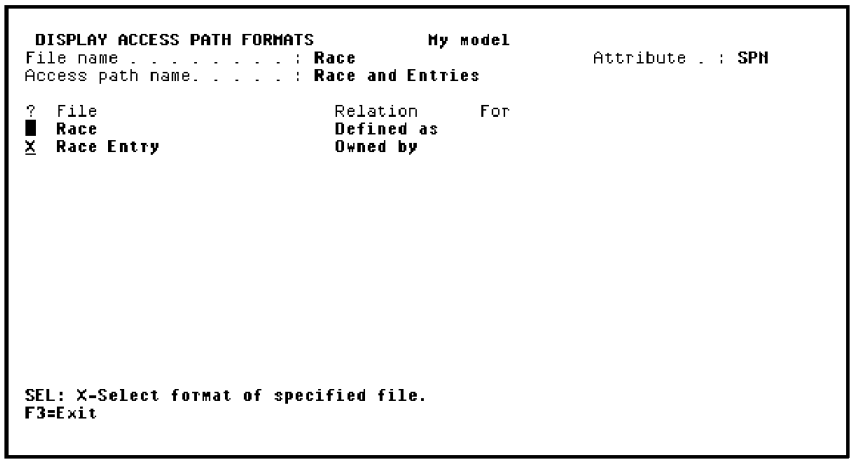
Confirming Selection of the First Format

The selection of the first format for the SPN access path, Race and Entries, has now been confirmed. Press F9 to add a second format for the subfile record (detail).



Specifying the Second Format

The Display Access Path Formats panel displays showing the file-to-file relations for the RACE file. Type **X** to select the Race Entry format for the subfile record.

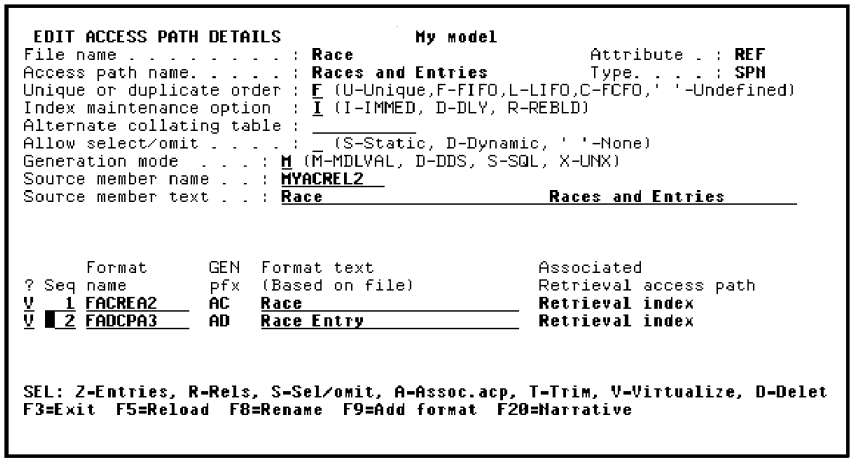


Press Enter to return to the Edit Access Path Details panel.

Adding Virtual Fields to the Access Path Formats

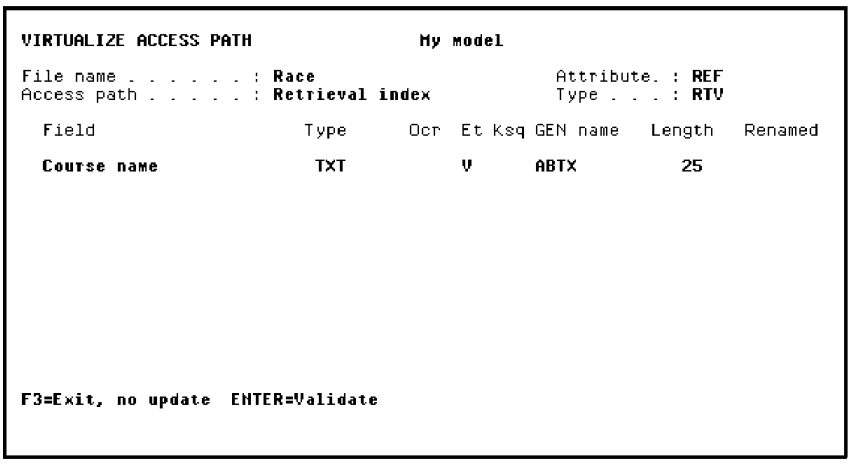
When the end user enters races and race entries, it would be helpful to have the following virtual fields display on the interactive panel: Course name, Horse name, and Jockey name. As mentioned previously, each access path initially contains all of the relations for the file on which it is based, but none of the virtual entries. As a result, if you want virtual fields included on the two formats for the Race and Entries access path, you need to add them explicitly.

In this step, you will add these virtual fields to the Race and Race Entry access path formats. Type **V** in the Subfile selector for both the Race format and the Race Entry format.



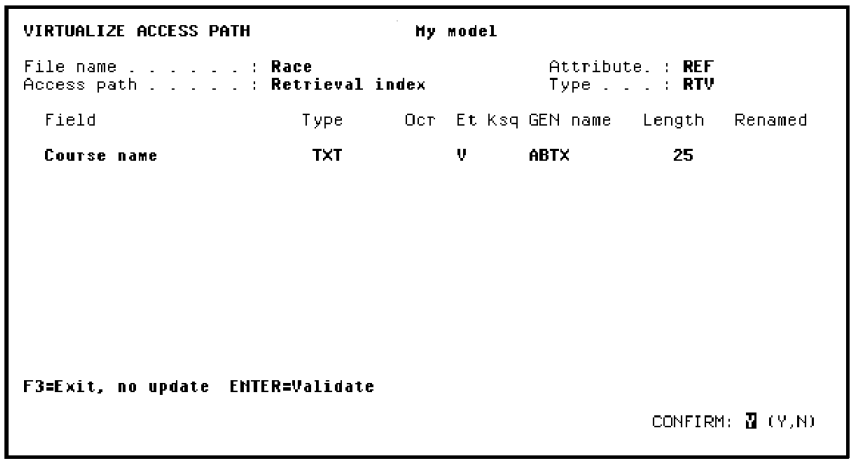
Press Enter.

The Virtualize Access Path panel displays showing the virtual field, Course name that can be added to the Race access path format.



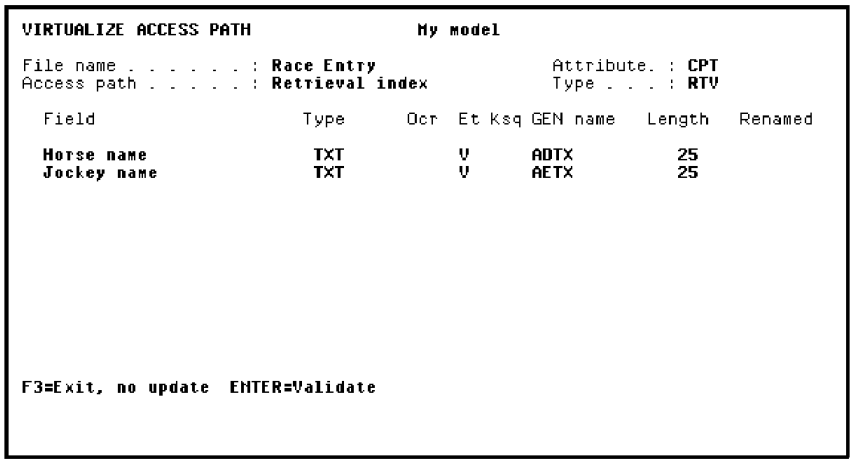
Press Enter to add the Course name virtual field to the Race access path format.

A Confirm prompt displays in the lower right-hand corner of the panel.

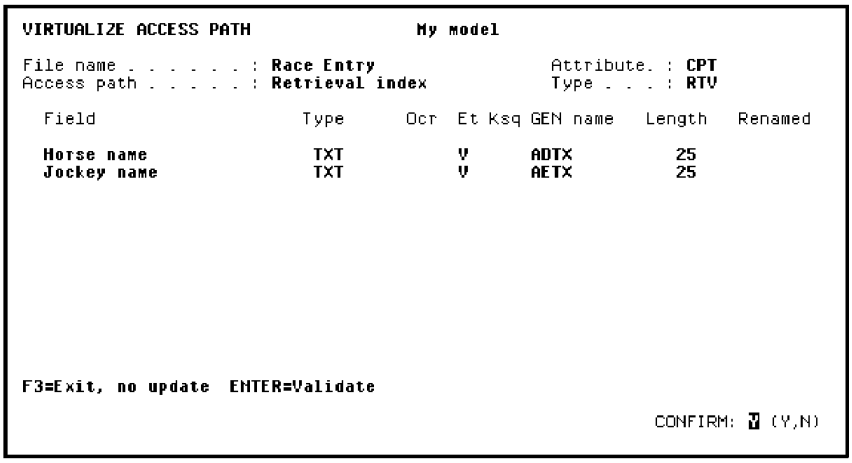


Press Enter to confirm the addition of the Course name virtual field.

CA 2E again displays the Virtualize Access Path panel showing the virtual fields that can be added to the Race Entry access path format. In this case, the virtual fields are Horse name and Jockey name.



Press Enter to add these virtual fields to the Race Entry access path format. A Confirm prompt displays in the lower right-hand corner of the panel.

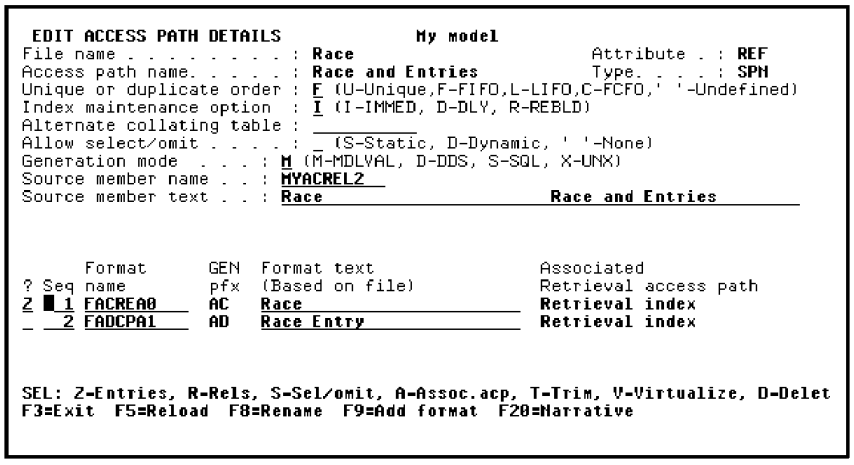


Press Enter to confirm the addition of the Horse name and Jockey name virtual fields and return to the Edit Access Path Details panel.

Details for the First Access Path Format

The Edit Access Path Details panel displays both formats. Next, display the details for the access path format by zooming against the Race access path format. This displays the Edit Access Path Format Entries panel.

Type **Z** against the Race access path format.

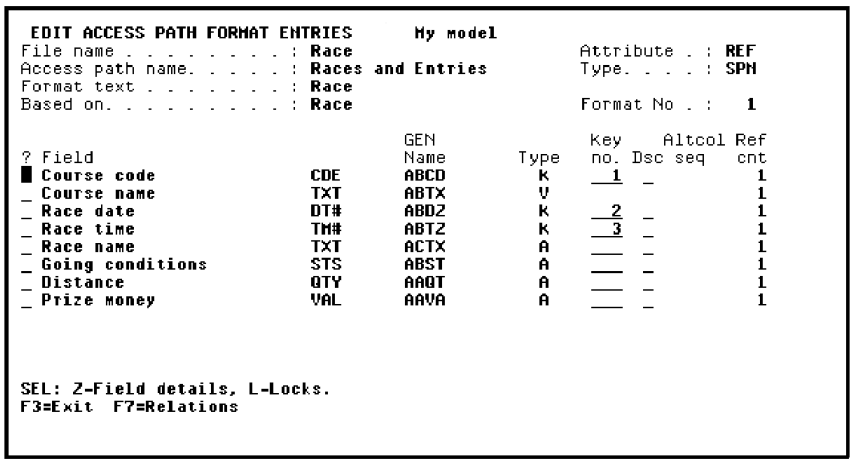


Press Enter.

Specifying the Key for the First Access Path Format

The next step is to check the keys of the Race access path format. You may specify any of the fields from the RACE file as keys, in either ascending or descending order. However, there must be a common key between the two formats. For example, the header record key fields must be part of the detail record's key.

The keys will be defaulted to the keys as specified by the file relations. In the case of two files linked by an Owned by relation, the default keys are recommended. If the two file formats are linked by a Refers to relation, you need to change the key order.



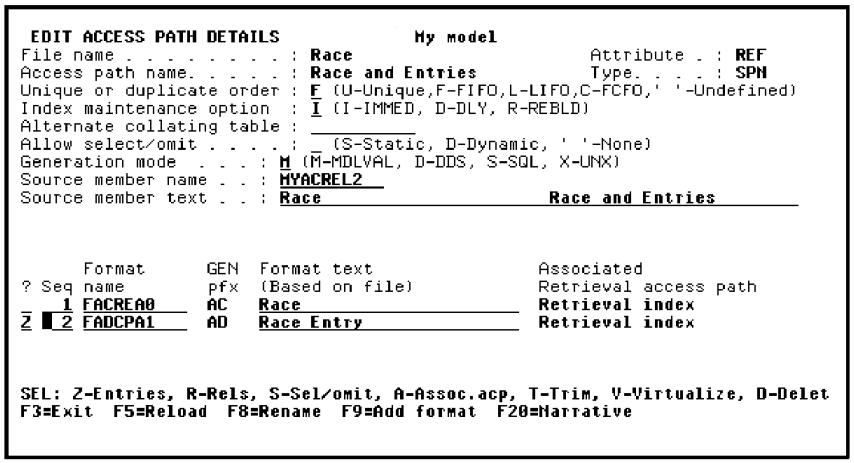
Because in this example, RACE ENTRY is Owned by RACE, press Enter to accept the defaults.

Press F3 to exit and return to the Edit Access Path Details panel.

Details for the Second Access Path Format

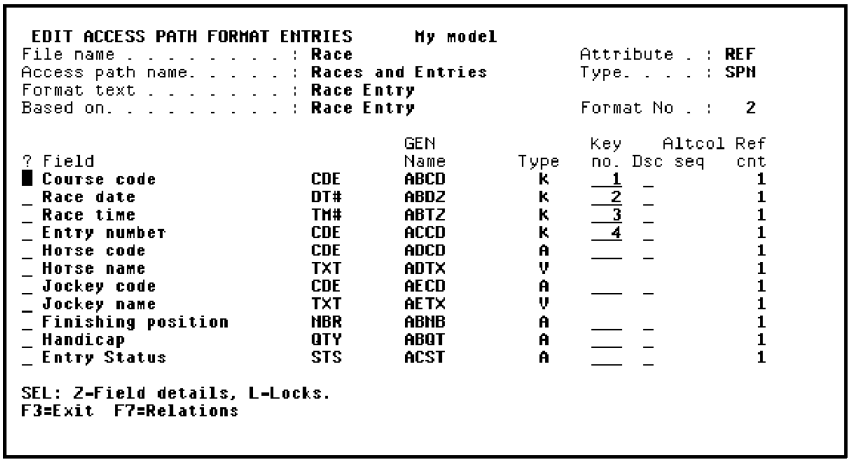
In this step you will zoom into the Race Entry access path format to check its key.

Type **Z** against the Race Entry access path format.



Press Enter.

The keys Course code, Race date, Race time, and Entry number have been selected by default to be the keys of the format since these are the keys as defined by the file relations for the RACE ENTRY file.

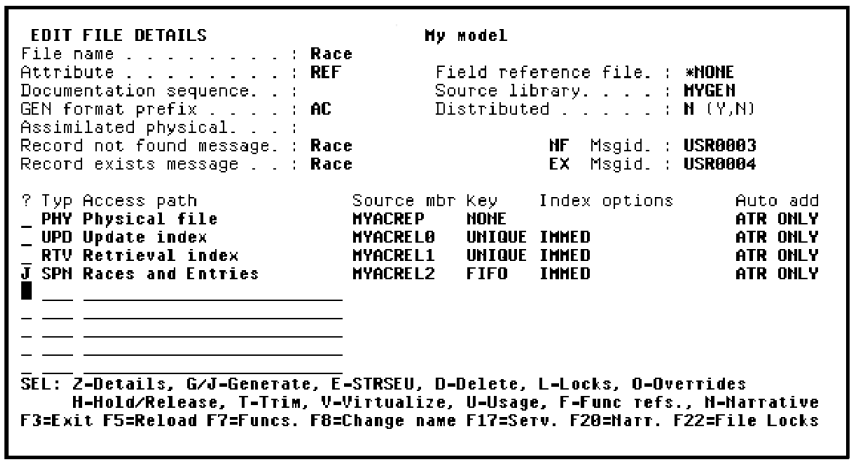


Press Enter to confirm the keys.

Press F13 to exit and return to the Edit File Details panel.

Requesting Batch Generation of the SPN Access Path

You have now finished the definition of the Races and Entries SPN access path. Request batch generation of the Races and Entries access path by typing **J** against it



Press Enter.

The Edit Transaction Function

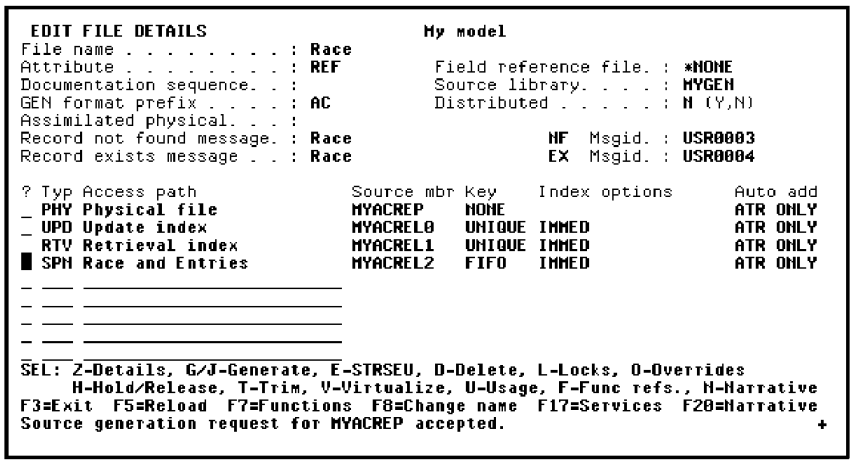
In this step you will define a function of type Edit Transaction (EDTTRN).

Objectives

You will create an Edit Transaction function for the RACE and RACE ENTRY files. The name of the function will be Edit Race and Entries and it will use the SPN access path Race and Entries you just created. This will provide the facility to enter both the race details and the race entry details for a given race on a single panel.

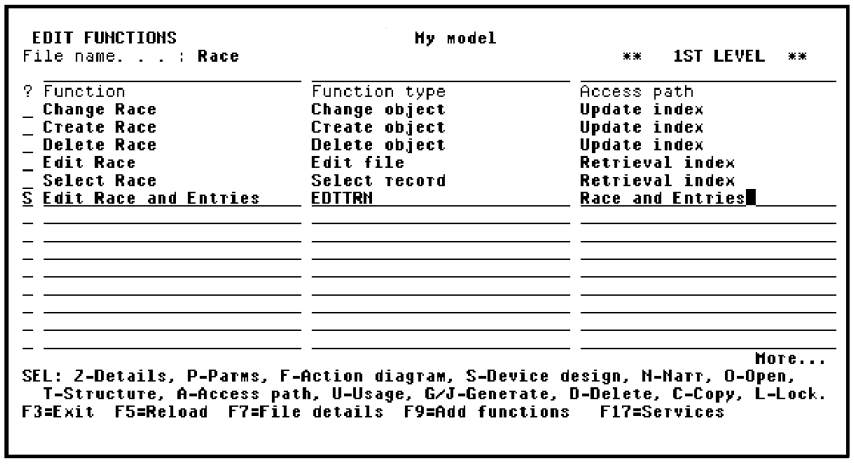
Defining the Edit Transaction Function

From the Edit File Details panel, you can transfer directly to the Edit Functions panel for the RACE file by pressing F7.



Press F7 to access the Edit Functions panel.

Type the function details; namely, type **Edit Race and Entries** for the Function name, **EDTTRN** for the Function type, **Race and Entries** for the Access path, and **S** to edit the device design.

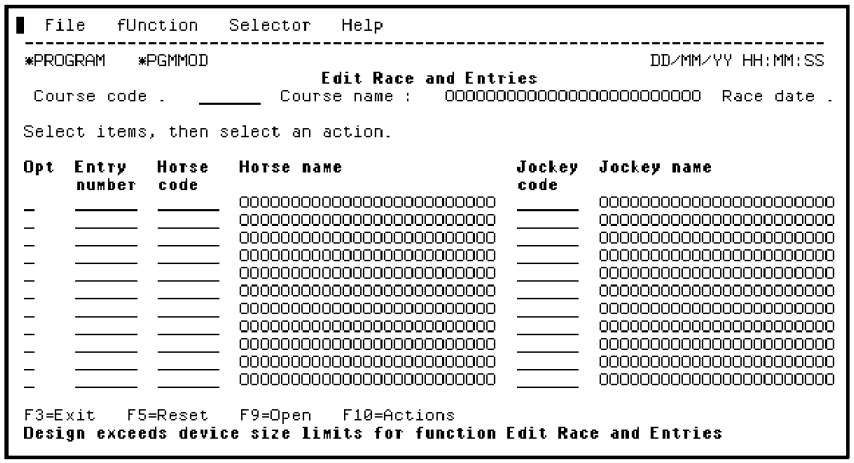


Press Enter to create the Edit Race and Entries function and display the device design.

The Default Device Design

The default device design for the Edit Race and Entries function should look like the panel below. Note that the subfile control (header) is positioned at the top with the subfile record (detail) below. The common key of the two file formats displays only in the subfile control and acts as an implicit restrictor parameter to the subfile detail records.

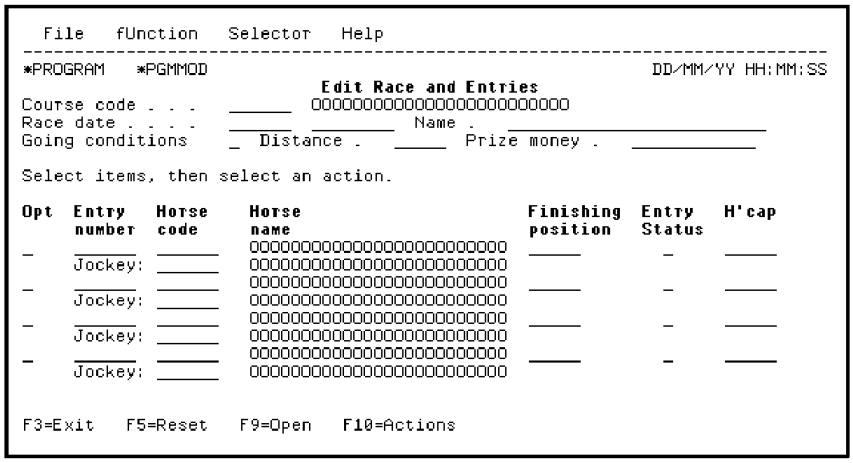
Note also the virtual fields Course name, Horse name, and Jockey name. If you had



not explicitly added these fields to the formats of the SPN access path, they would not appear on this display.

The Modified Device Design

Update this default device design to look like the one shown on the next panel. Use the function keys you used to edit the device design for the Edit Horse and Select Horse functions earlier in this tutorial.

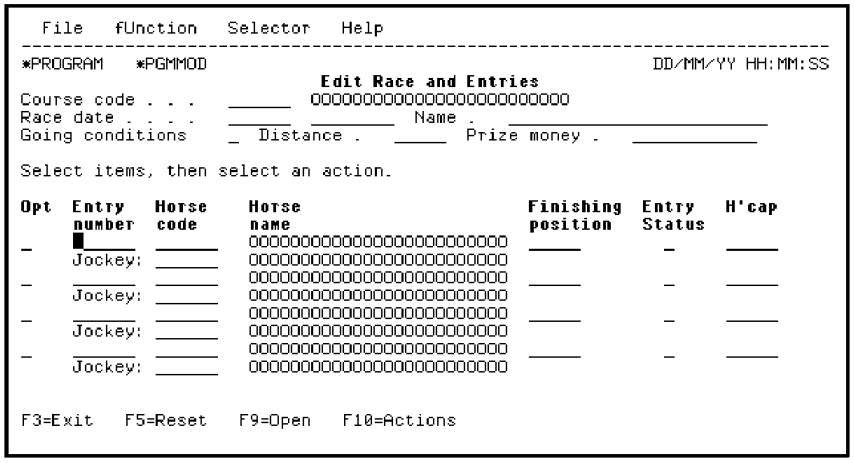


Note: To move a field to another position on the panel, place the cursor on the field to be moved and press F8. Position the cursor on the field that appears before the desired position for the field you are moving. Press F8. For example, place the cursor on the Finishing position field, and press F8. Then position the cursor on the Horse name field, and press F8 again. This moves the Finishing position field after Horse name.

Defining Optional Entry Fields

You have now updated the device design to meet the initial requirements. However, by default, CA 2E defines all input-capable fields as required entry fields. To allow the end user to enter race entries without specifying all the details about those entries, you can make some of the fields optional. Note that only non-key fields can be optional.

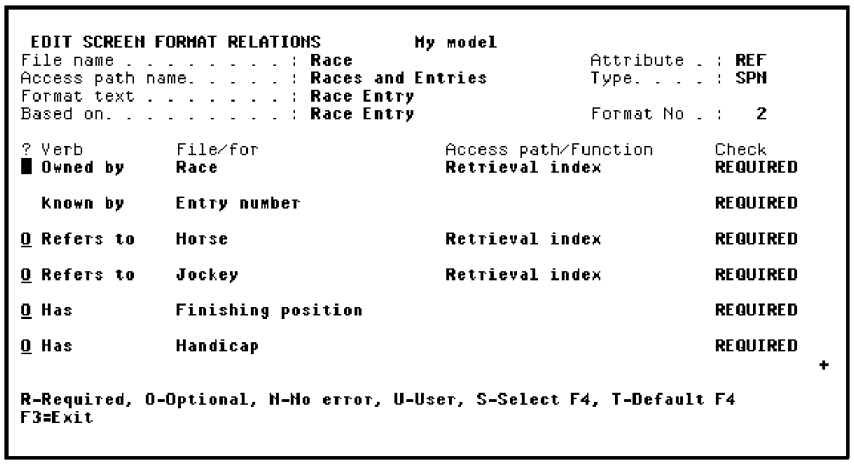
To make some of the fields in the RACE ENTRY file optional, position the cursor on a field in the first subfile record.



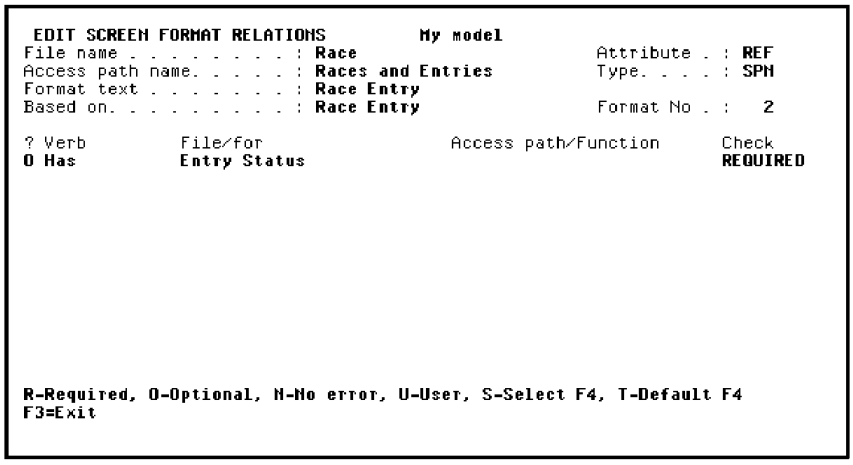
Press F7.

This displays the relations for the subfile record format. You specify optional relations by typing **O** against the relations.

Type **O** against the relations.

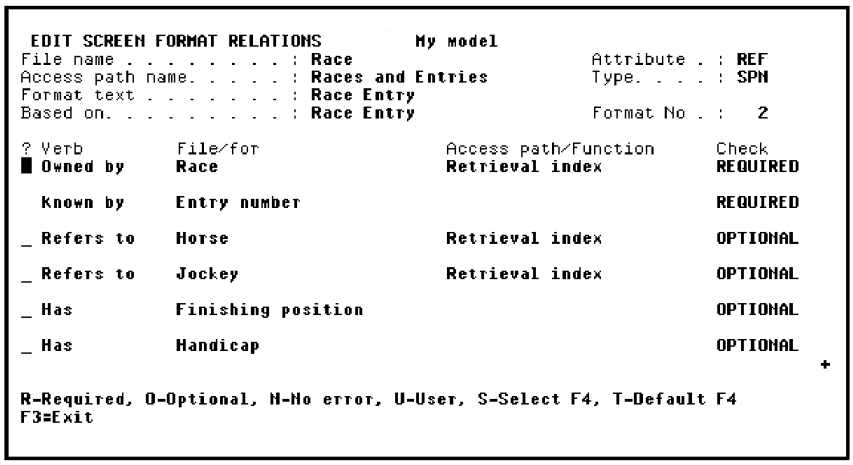


Press Roll Up to view the remaining relations listed. Type **O** against the relation.



Press Enter to confirm the changes to the format relations. Note that the values in the Check column changed from REQUIRED to OPTIONAL.

Press Roll Down to view the beginning of the list of relations.



Press F3 return to the device design.

Introduction to Function Fields

You will now add function fields to your device design.

New terms introduced

- Function field
- New panels introduced
- Edit Device Function Field

Objectives

Add a new field to the subfile control format of the Edit Transaction panel that shows the number of horses that finished the race. To do so you will define a function field containing the total number of race entries having an Entry Status of Finished.

Overview of Function Fields

Function fields are non-database fields that may be used in device designs and action diagrams. The same function field can be used in many different functions.

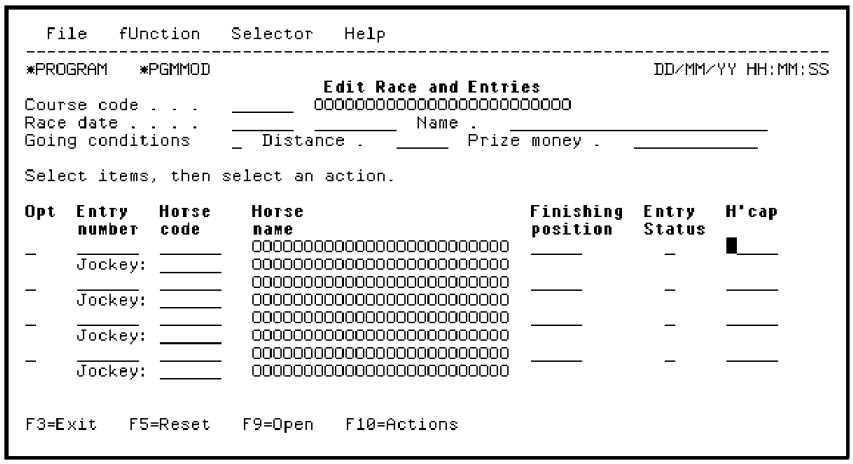
There are six types, or usages, of function fields. Four provide standard field-level functions: SUM, MIN, MAX, and CNT; two are user-defined and let you define your own function fields: USR and DRV.

Function field parameters specify which field values are to be passed into the function field at execution time and which field is to be returned from the function field as the result. All function fields with the exception of USR fields have one output parameter, the function field itself. The system-defined function fields have a single input parameter. Derived function fields, of usage DRV, are specified in the action diagram and can have up to nine input parameters.

Adding Function Fields

In this step, you will define a function field of usage DRV to evaluate whether or not a given race entry finished a race. The DRV field will contain a value of 1 if the horse finished and 0 if it did not. You will also define a function field with a usage of SUM to calculate the number of finishers. The sum of the values in the DRV function field gives the count of the number of finishers.

Position the cursor on the field that is to be displayed on the panel before the function field, in this case Handicap.

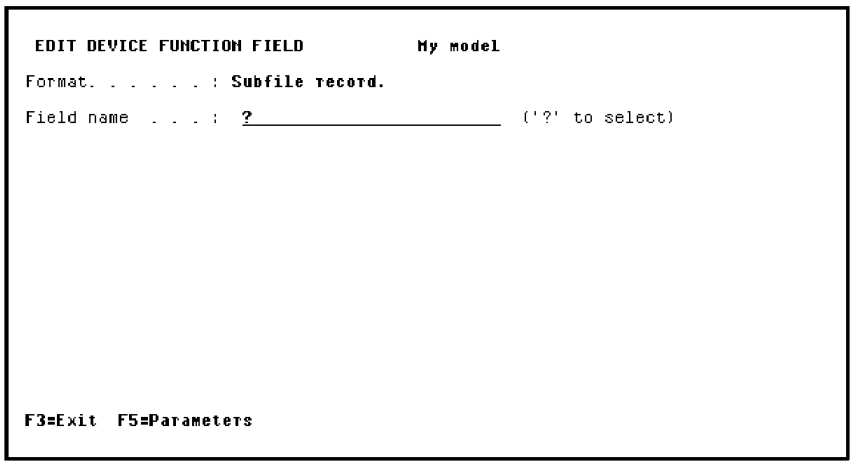


Press F19 to display the Edit Device Function Field panel.

Naming the Function Field

If the function field already exists and you know its name, you can fill in the name on the Edit Device Function Field panel. Otherwise, type a **?** to display existing fields.

Type **?** for the Field name.



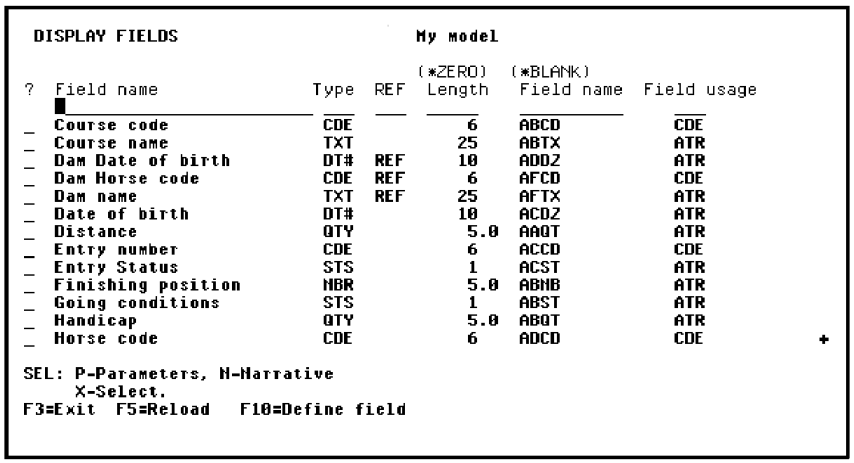
Press Enter.

Displaying Existing Fields

The **?** displays a list of all existing fields. You can select an existing function field or you can create a new one. In this tutorial, you will create two new function fields.

Defining Two New Function Fields

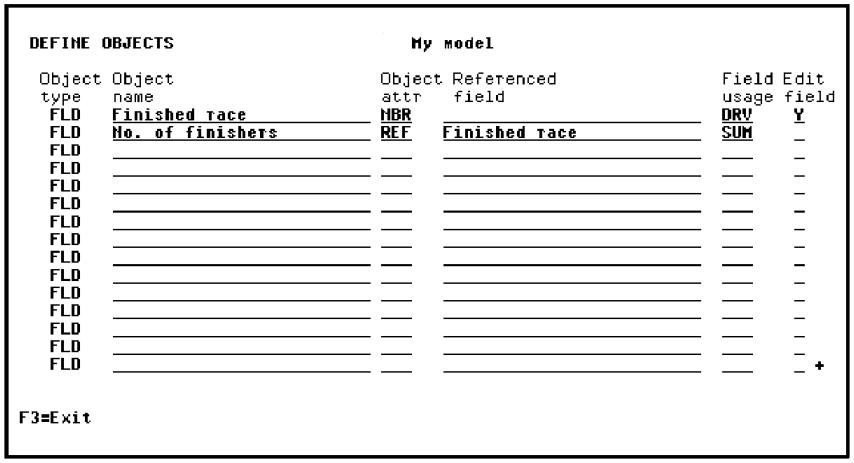
To add the function fields you require, press F10 to display the Define Objects panel. You can define both function fields at this point.



Here are the details for the two function fields:

- **Finished race**—a function field that appears on each subfile record and evaluates whether or not the horse finished a race. Its value is derived from the value of the Entry Status field. This requires a usage of **DRV**. You can edit the default field details directly by typing **Y** in Edit field.
- **No. of finishers**—a function field in the subfile control and is a sum of the values in the Finished race field of each subfile record. It is defined as a REF field based on the field Finished race.

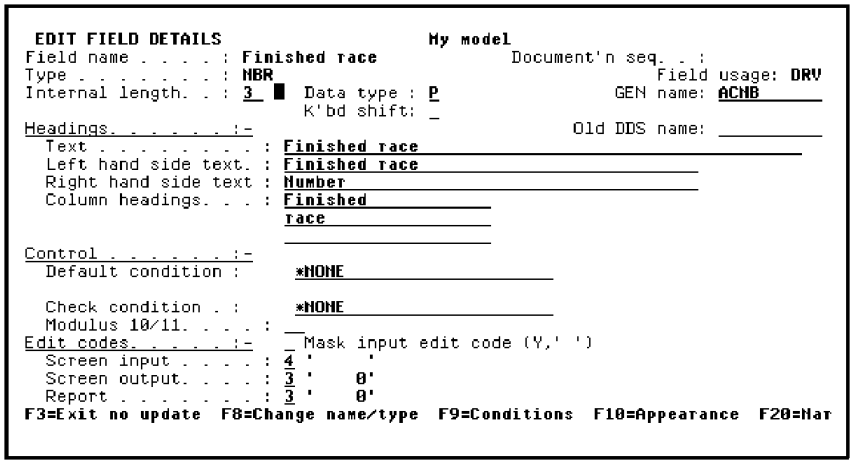
Type the new function fields.



Press Enter.

Overriding the Field Defaults

Since you specified **Y** for Edit field, you will be shown the field details for the Finished race field. You can now change the internal field length to 3 from the Edit Field Details panel. Type **3** for internal length and press Field Exit.

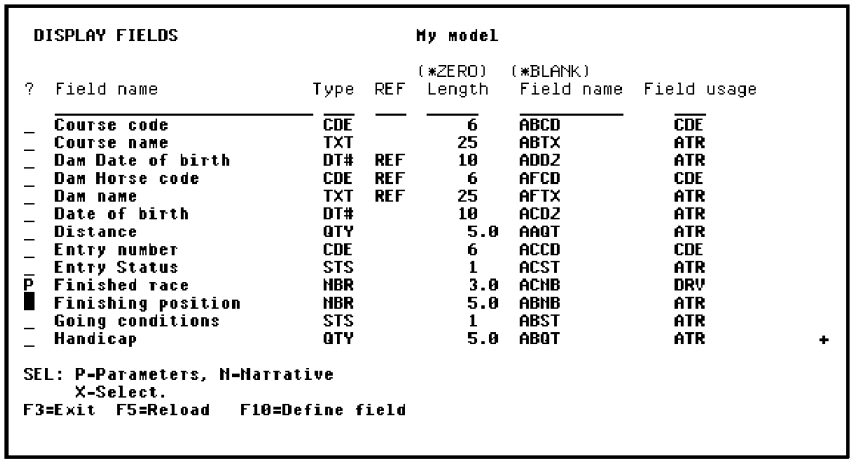


Press Enter. Press F3 to return to the Display Fields panel.

Displaying the Function Fields

The Display Fields panel now has your new function fields displayed. Note that you need to press Roll Up to view the No. of finishers function field.

Type **P** against Finished race as shown to edit its parameters.

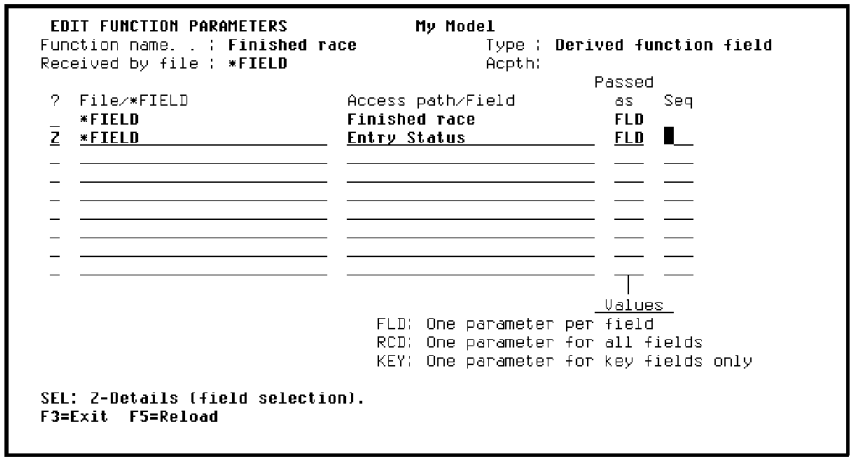


Press Enter to display the Edit Function Parameters panel.

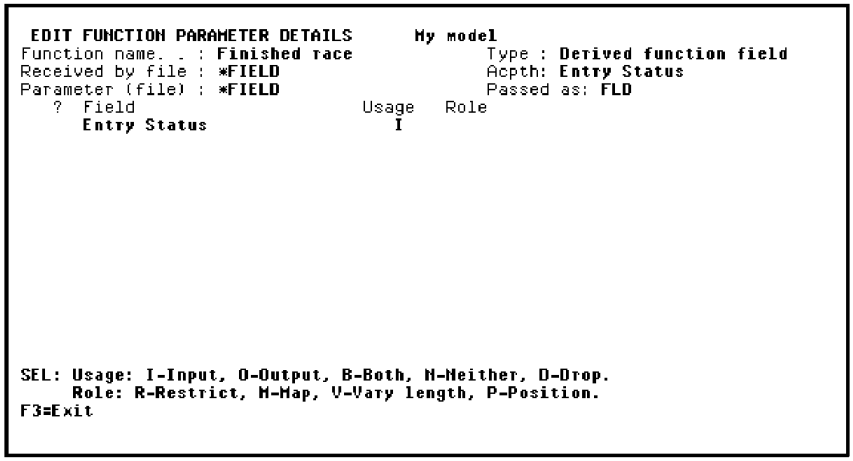
Defining Function Field Parameters

Next you are going to specify that Entry Status is an input parameter to the Finished race function field. You will use the same Edit Function Parameters panel you used to specify function parameters earlier in this tutorial.

Type the details for the Entry Status parameter as shown. The usage of this parameter defaults to input. Check the parameter usage by also typing **Z** against the field.



Press Enter. The Edit Function Parameter Details panel displays showing the Usage for Entry Status as I (input).



Press F3 to return to the Edit Function Parameters panel.

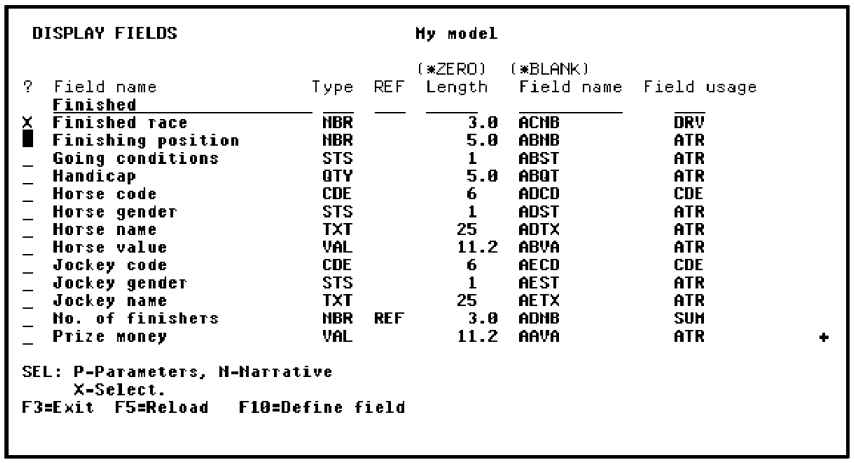
The derived (DRV) function field itself, Finished race, is automatically defined as an output parameter. Check this by displaying the parameter details. Type **Z** against the Finished race field to display the Edit Function Parameter Details panel. Note that the Usage for this parameter is O (output).

Press F3 twice to exit and return to the Display Fields panel.

Selecting the Function Field

You have just finished declaring the function fields in CA 2E. After returning to the Display Fields panel, you need to select the Finished race function field for inclusion on the device design. First, type **Finished** in the positioner line and press Enter to position to the new function fields.

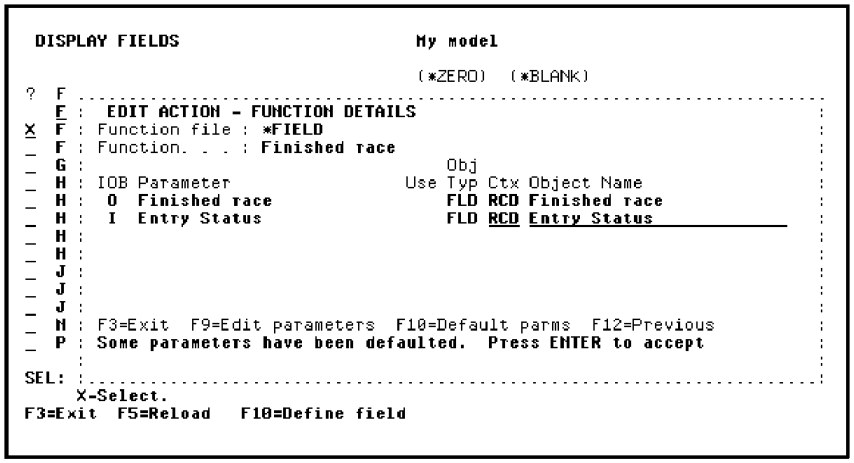
Type **X** next to Finished race.



Press Enter.

Accepting the Parameters

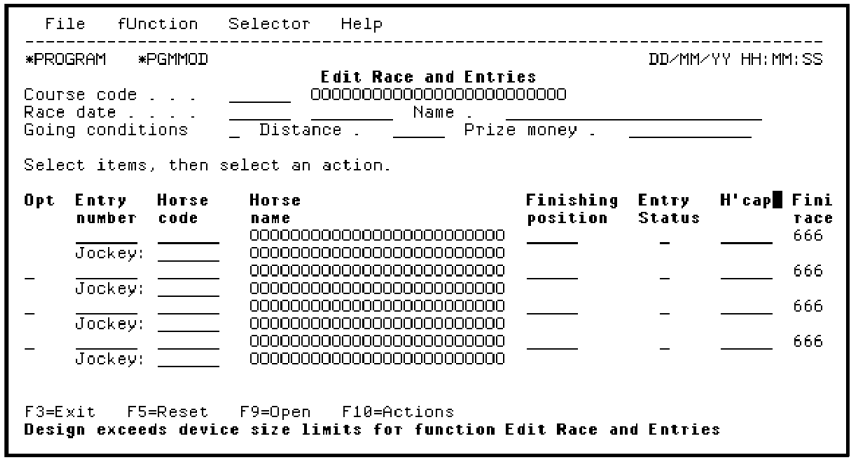
The Edit Action - Function Details window is displayed showing the parameters for the Finished race function field. Note that the input parameter is Entry Status, the output parameter is Finished race, and both parameters have the RCD (subfile record) context.



Press Enter to accept the default parameters and return to the device design.

The Modified Device Design

After returning to the device design, note that the Finished race function field has been added to the subfile record.



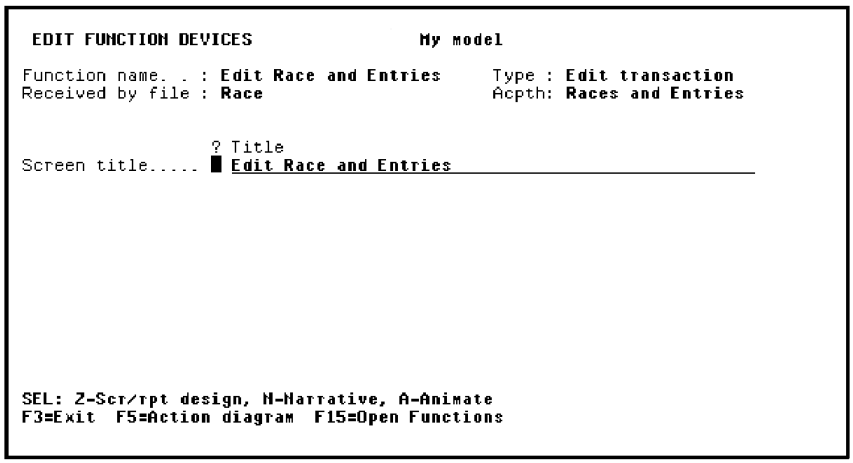
Note that since the device design exceeds the limits of the panel, you will need to adjust this device design later in the tutorial.

Press F3 to exit and display the Edit Function Devices panel.

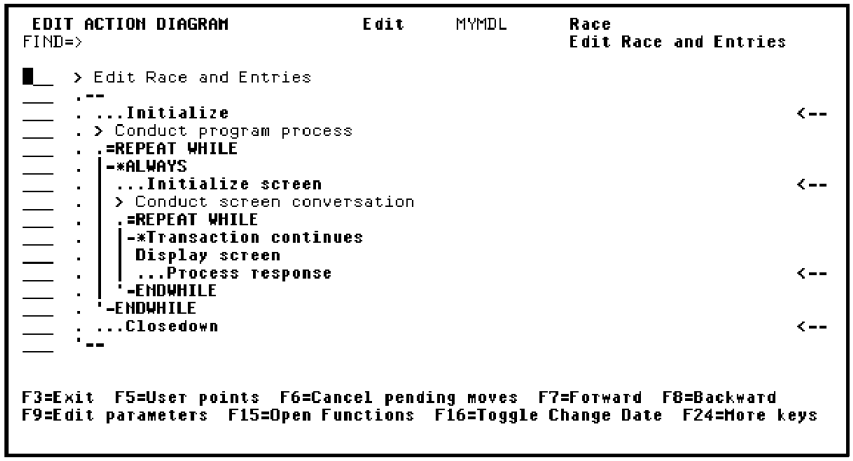
Defining a Function Field Action Diagram

You will now define the action for the user-defined (DRV) function field, Finished race. This will require entering logic into the action diagram for the Edit Race and Entries function.

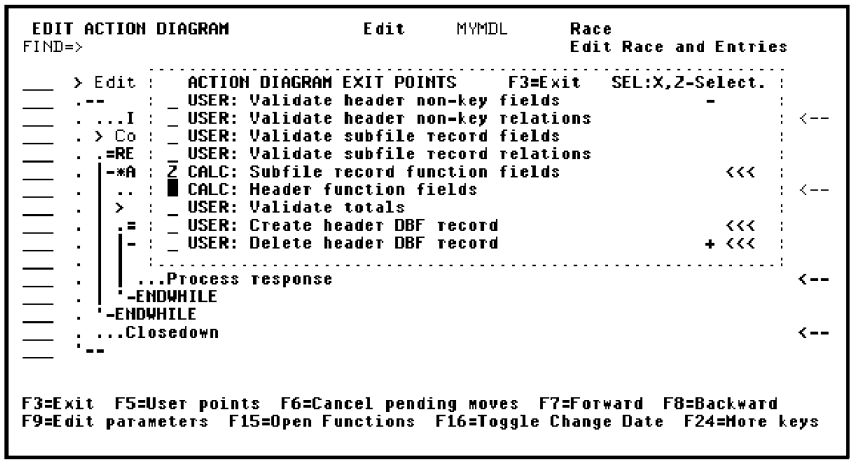
Press F5 to view the action diagram for the Edit Race and Entries function.



Press F5 to view the Action Diagram User Points.



Recall that the user points are the places in the action diagram that you can change.

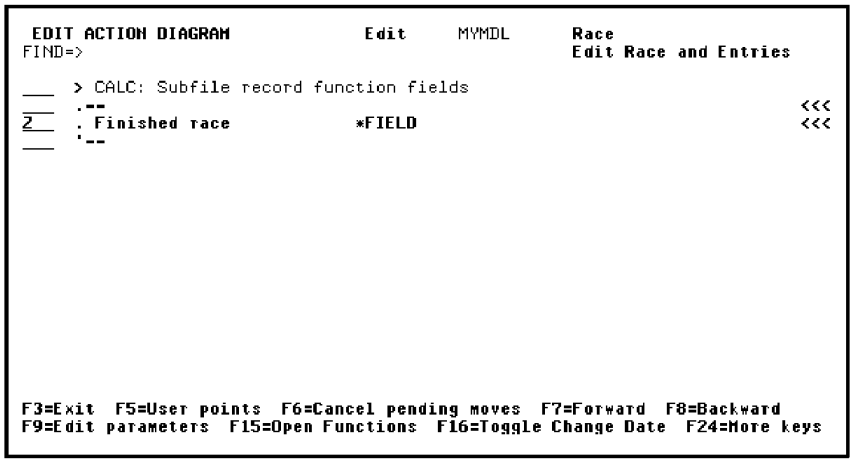


Press Roll Up to see the next set of user points.

Type **Z** against the CALC: Subfile record function fields user point.

Press Enter. Note that CA 2E automatically inserted the Finished Race function field in the action diagram for the Edit Race and Entries function.

Type **Z** against Finished Race as shown to insert logic for the function field.



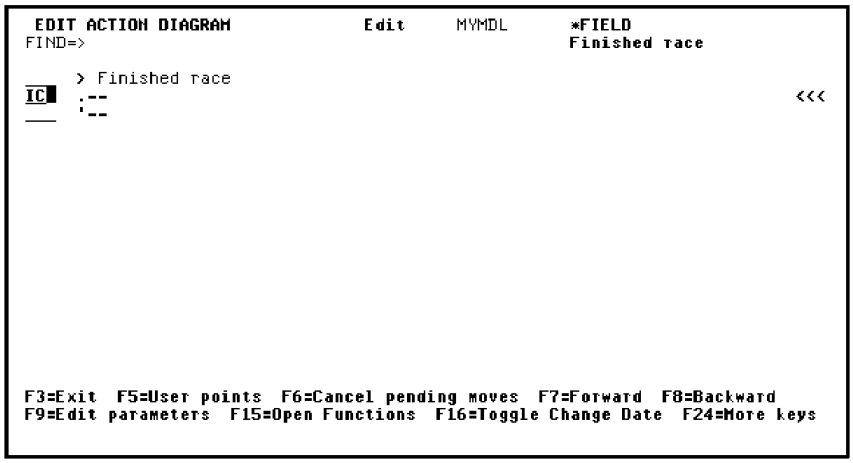
Press Enter.

Adding a CASE Construct

The action diagram for the function field Finished race will be displayed. Note that you are now editing the Finished race function as shown in the upper right corner of the panel. You will use the Action Diagram Editor to insert the following conditions and actions for the Finished race function field.

```
CASE
   Condition 1      Does Entry Status = Finished?
   Action 1         Yes, set Finished race to 1
   Condition 2      *OTHERWISE
   Action 2         Set Finished race to 0
ENDCASE
```

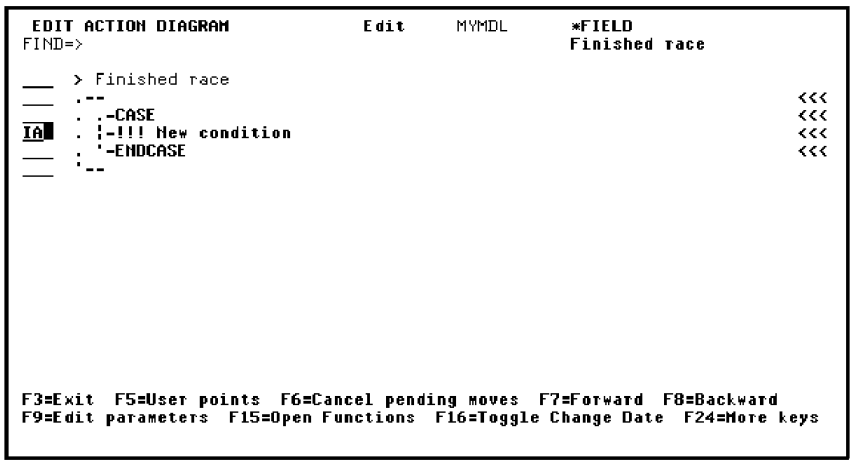
Type **IC**.



Press Enter.

Adding an Action

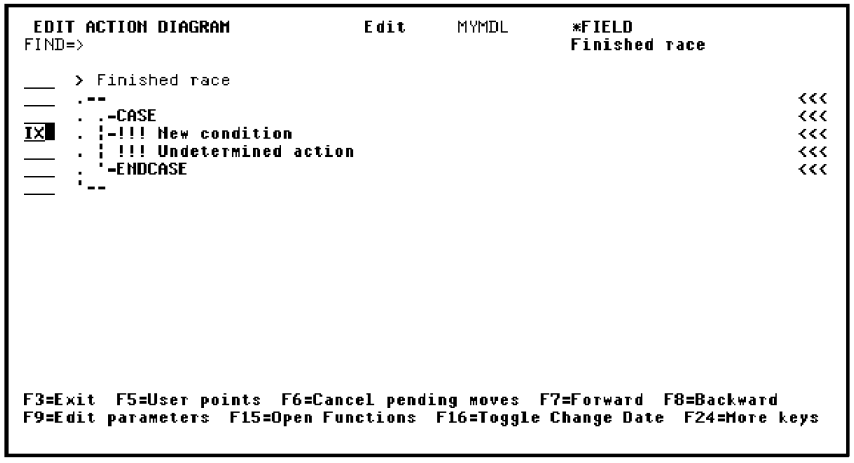
Insert an action into the action diagram. Type **IA**.



Press Enter.

Adding a Second Condition

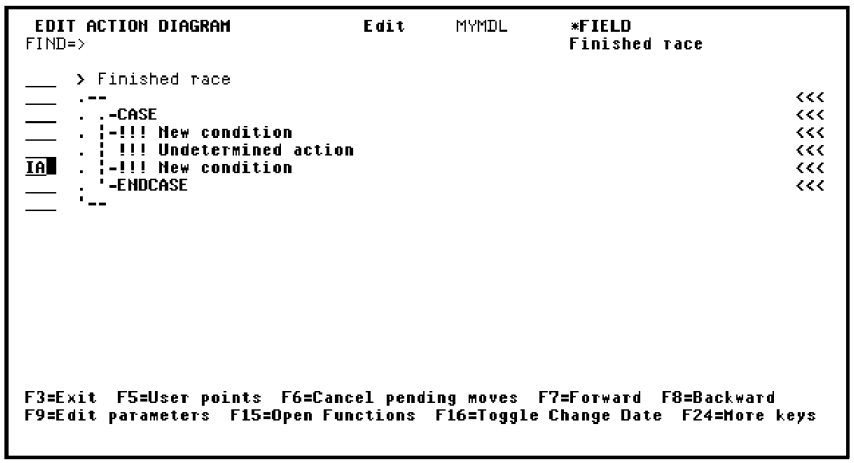
Add another condition to the CASE construct by typing **IX** against the first !!! New condition as shown. Recall that IX inserts a new condition within a CASE construct.



Press Enter.

Adding a Second Action

Add another action to the CASE construct by typing **IA** against the second !!! New condition.

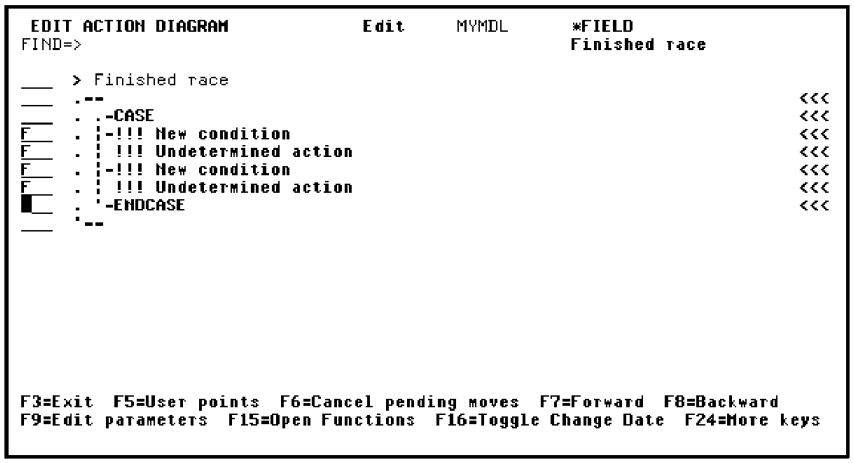


Press Enter.

Defining the Conditions and Actions

To define the conditions and actions, type **F** against them all, and press Enter. Recall that F lets you edit the action or condition details for a line in the action diagram.

Type **F** against the conditions and actions.

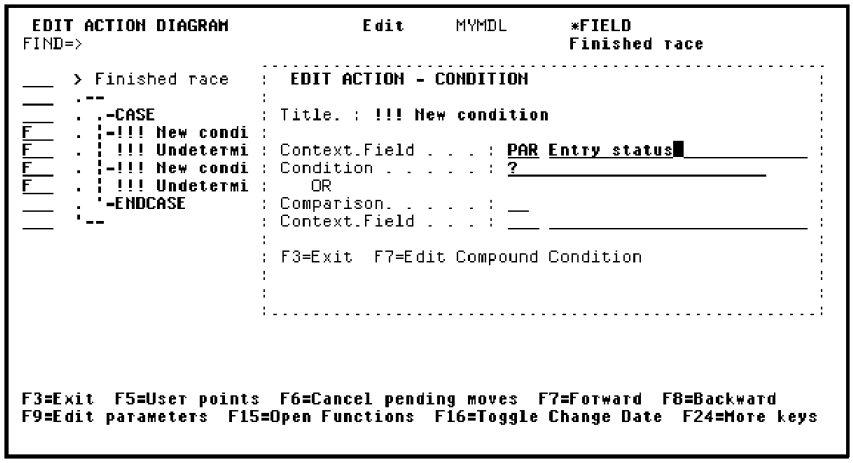


Press Enter.

Specifying the First Condition

The first condition specifies that the action is only to be executed if the Entry Status is Finished.

Type the details as shown. Type **PAR** (parameter) for the context, **Entry Status** for the Field, and **?** in the condition field to view a list of the available conditions.

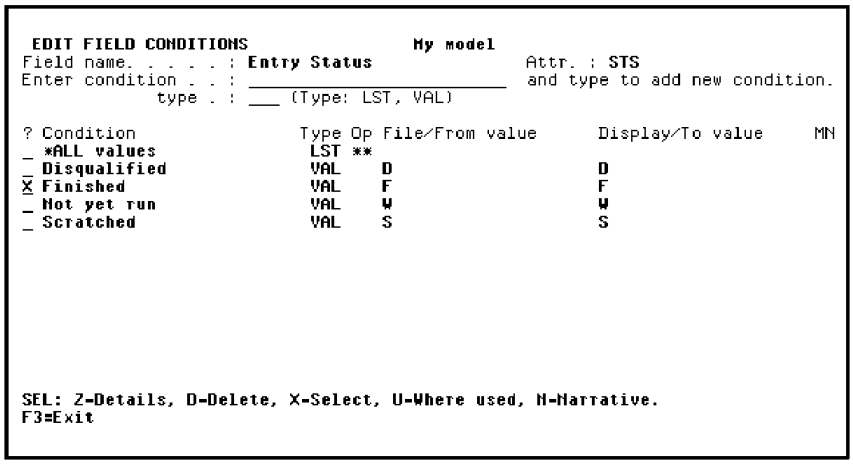


Press Enter.

Defining Field Conditions for the Status Field

If you completed the earlier exercise of defining conditions for the Entry Status field, the Edit Field - Condition window panel will include the list of conditions shown here. If you have not already done so, you can add them at this point.

Type **X** against the condition Finished.



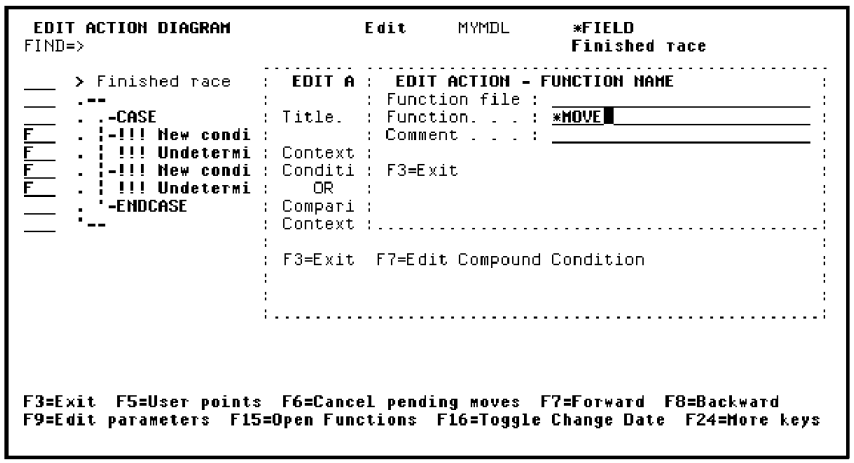
Press Enter twice.

Specifying the First Action

You have just finished defining the first condition in the CASE construct you are defining. Because you also typed **F** against the first action in the CASE construct, CA 2E displays the Edit Action - Function Name window to specify the action. In this case, the action will be the built-in function *MOVE.

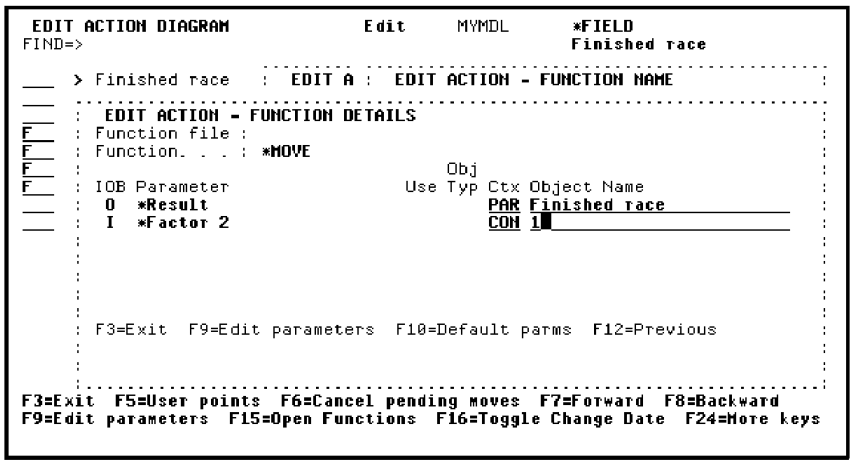
The action you are specifying is executed when the first condition is true; namely, when Entry Status is Finished. The action is to return a constant value of one in the Finished race function field.

Leave the Function file field blank to indicate a built-in function and type ***MOVE**.



Press Enter to enter parameter details for the *MOVE function.

Type the details for the *MOVE parameters.

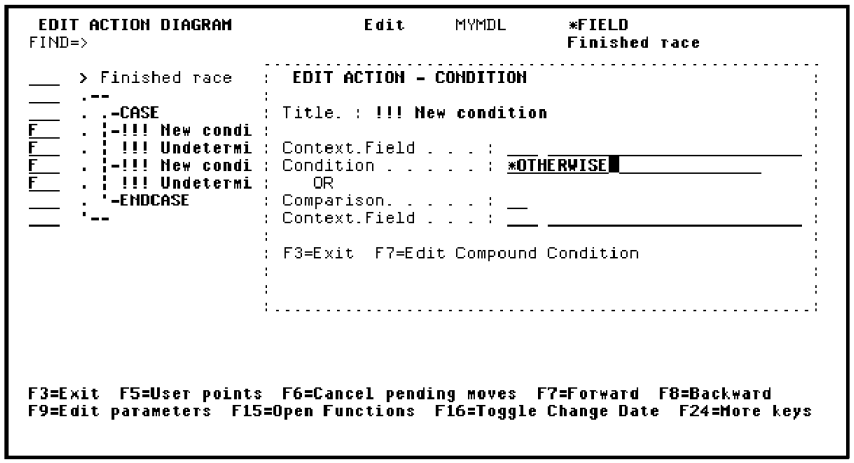


Press Enter.

Specifying the Second Condition

Because you typed **F** against the second new condition in the action diagram CASE construct, CA 2E displays the Edit Action - Condition window. The second condition specifies the action to be executed in all cases other than that specified in the first condition. A special value of *OTHERWISE is used for this case.

Remove all the ? and specify ***OTHERWISE** for the condition.

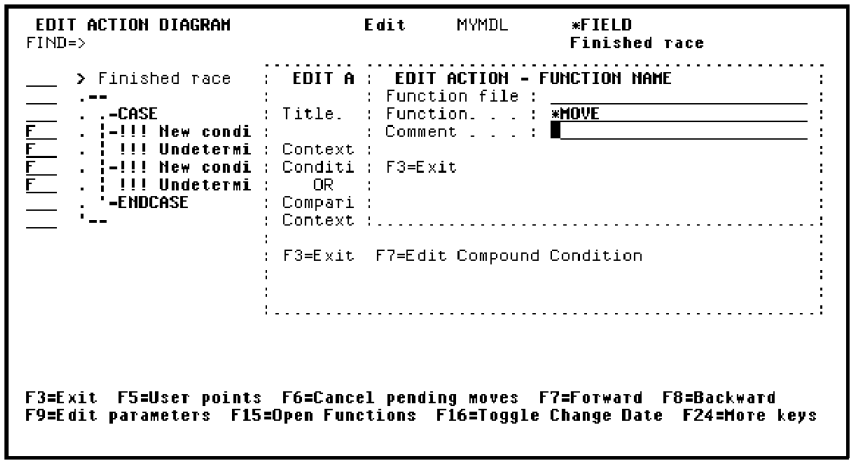


Press Enter.

Defining the Second Action

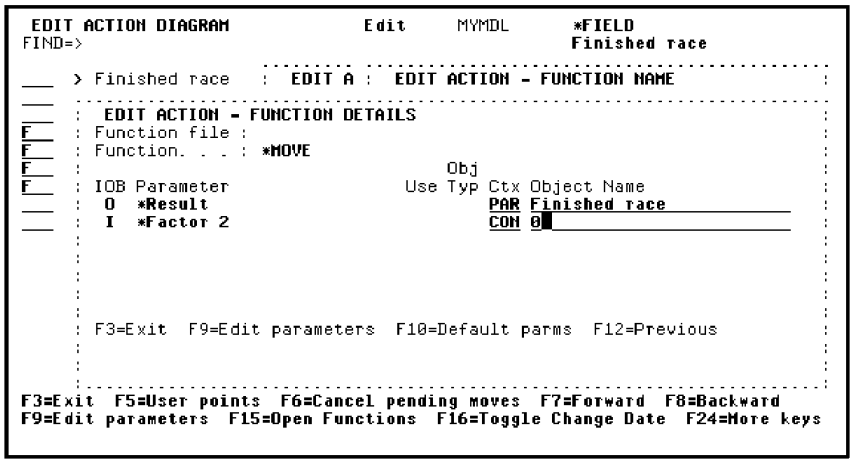
Because you typed **F** against the second action, CA 2E displays the Edit Action - Function Name window. For the second action, you will specify the action to take when the first condition is not true. The action is to return a value of zero in the Finished race function field. You will again use the *MOVE built-in function.

Leave the Function file field blank and specify the ***MOVE** function.



Press Enter to enter parameter details for the *MOVE function.

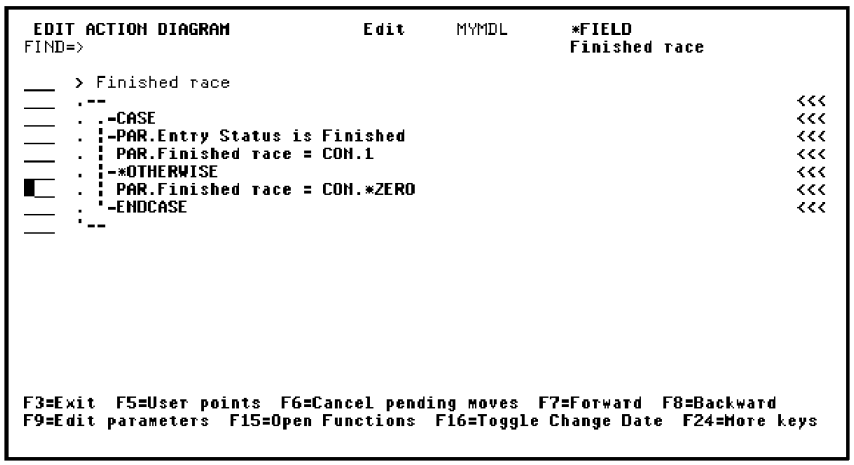
Type the details for the *MOVE parameters.



Press Enter to accept the parameters and return to the Action Diagram Editor.

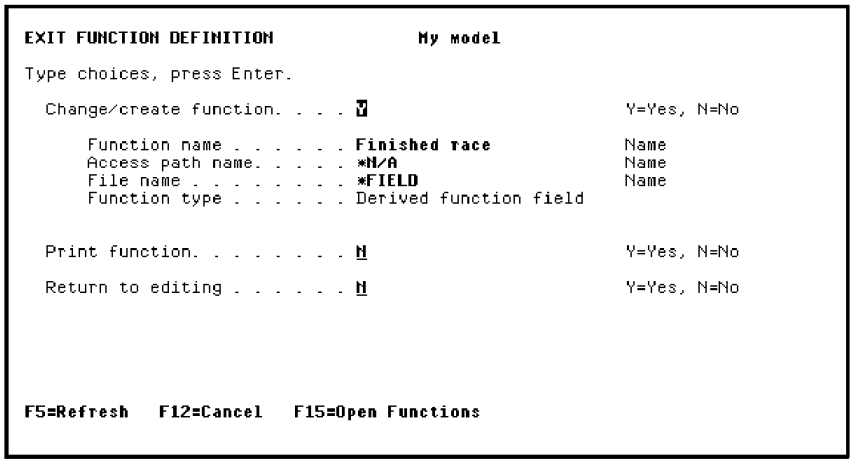
Complete Action Diagram

The completed action diagram should look like this.

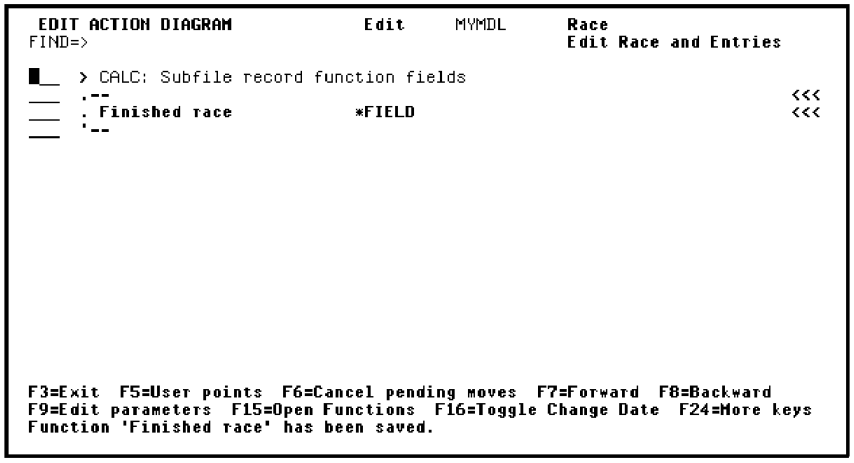


Press F13 to exit the action diagram for the Finished race function field.

Accept the default of **Y** to save the Finished race function field.

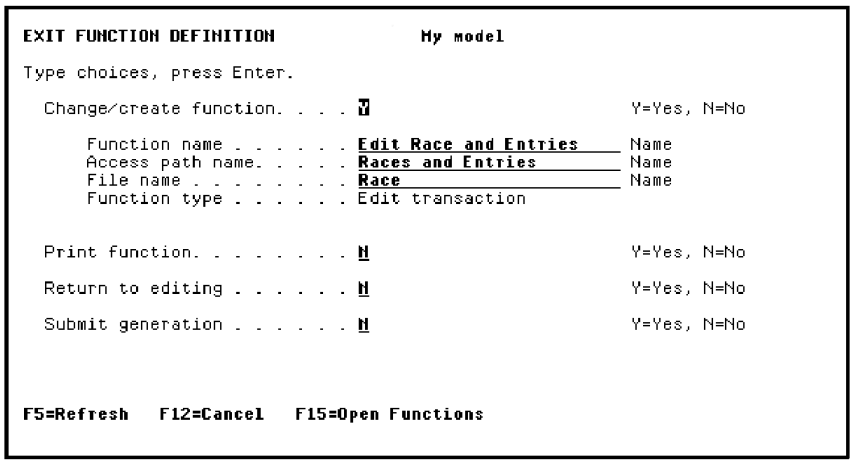


Press Enter to return to the action diagram for the Edit Race and Entries function.



Press F13 to exit the action diagram for the Edit Race and Entries function.

Accept the default of **Y** to save the function definition.

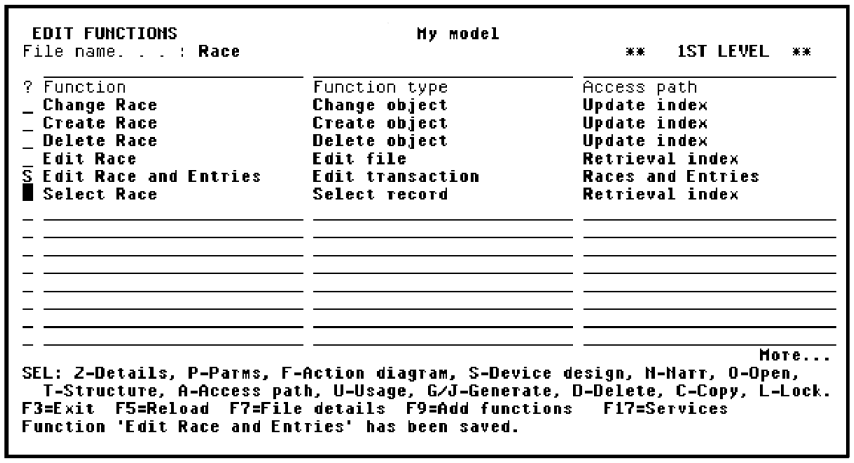


Press Enter.

Returning to the Device Design

You have now finished defining the Finished race function field. Return to the device design to define the No. of finishers function field.

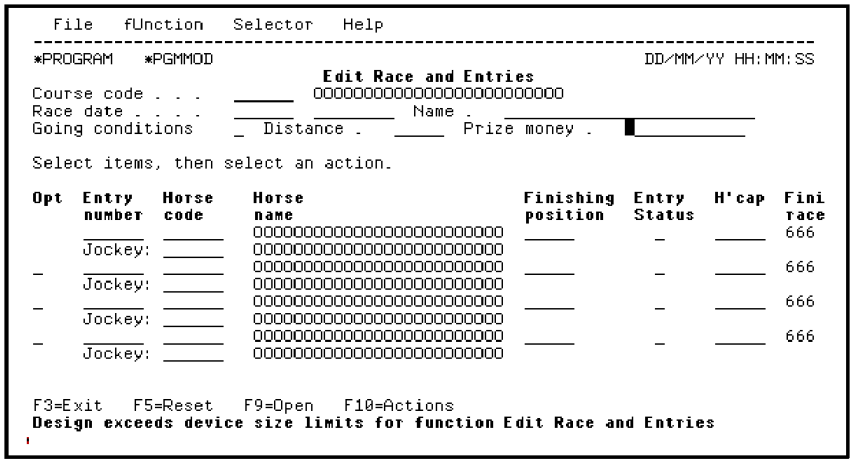
Type **S** against the Edit Race and Entries function.



Press Enter.

Adding the Second Function Field

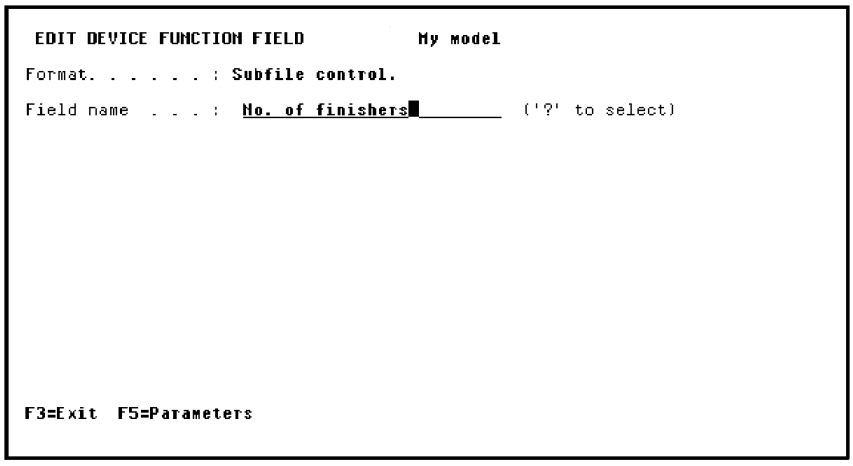
Add the second function field to the header format (subfile control). Position the cursor on the field before the desired location for the function field, in this case Prize money, and press F19.



Specifying the Second Function Field

The required field already exists; as a result, you can type the name No. of finishers directly. Alternatively you could type **?** to obtain a selection list.

Type the function field name No. of Finishers.

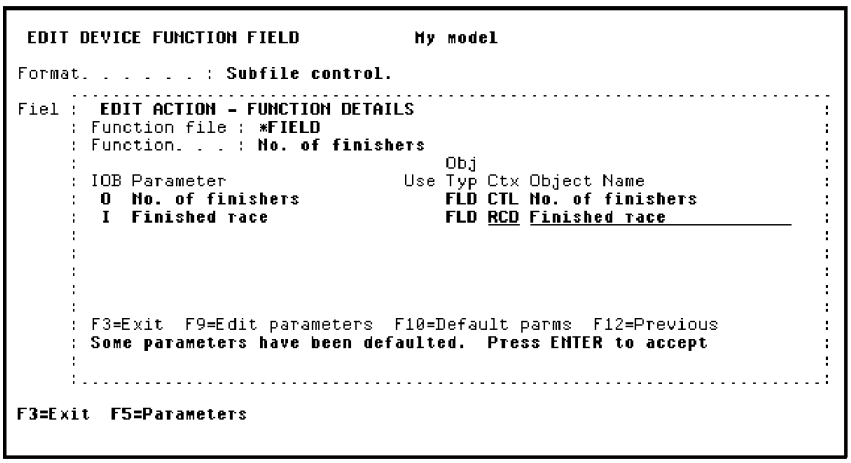


Press Enter.

Defining Parameters for the No. of finishers Function Field

The parameters for the No. of finishers function field are defaulted. The input parameter is Finished race from the RCD (subfile record) context; the output parameter is the No. of finishers function field itself in the CTL (subfile control) context. Recall that you defined the No. of finishers function field with the SUM field usage.

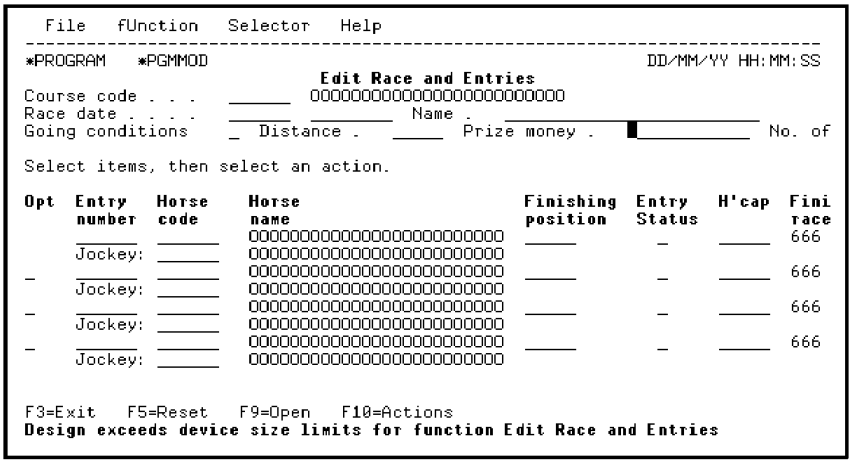
This causes the values in the input parameter field (Finished race) for each detail record to automatically be added together and the sum to be placed in the output parameter (No. of finishers).



Press Enter to accept the default parameters as shown.

The Added Function Field

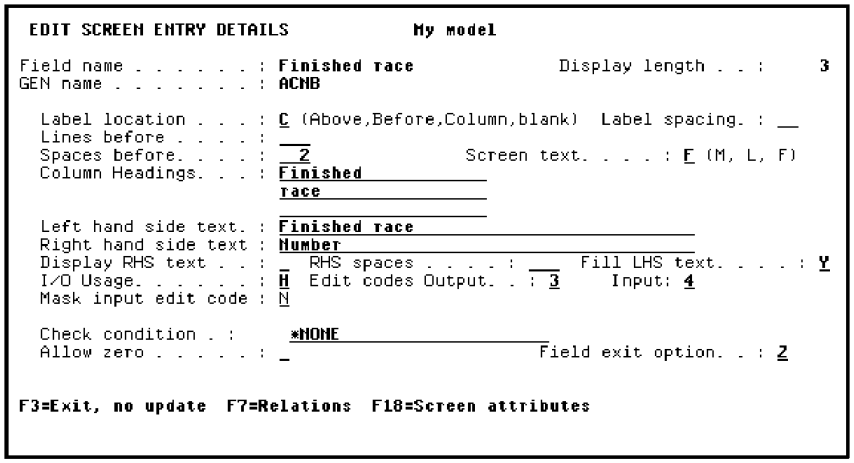
The No. of Finishers field will be added to the subfile control format.



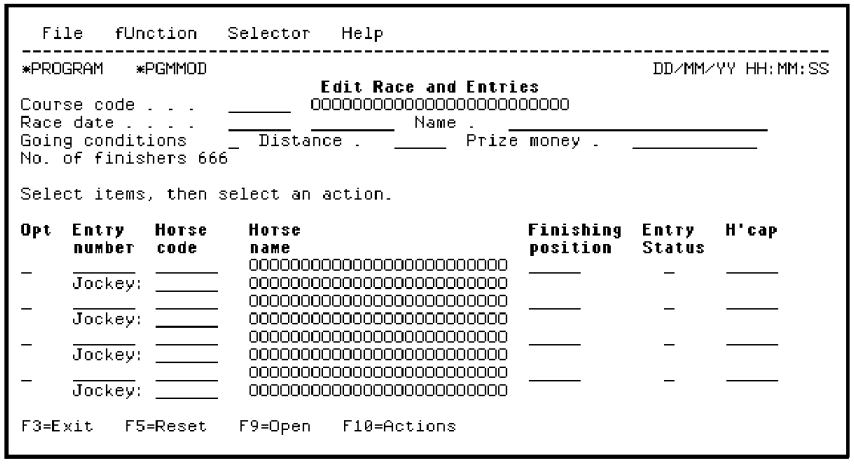
Readjusting the Device Design

Adjust the device design so that all fields fit on the panel. Move the No. of finishers function field to a new line and hide the Finished race function field so that it is not displayed.

To do the latter, position the cursor on the field and press Enter. The Edit Screen Entry Details panel displays. Change the I/O usage for the field to hidden (H).



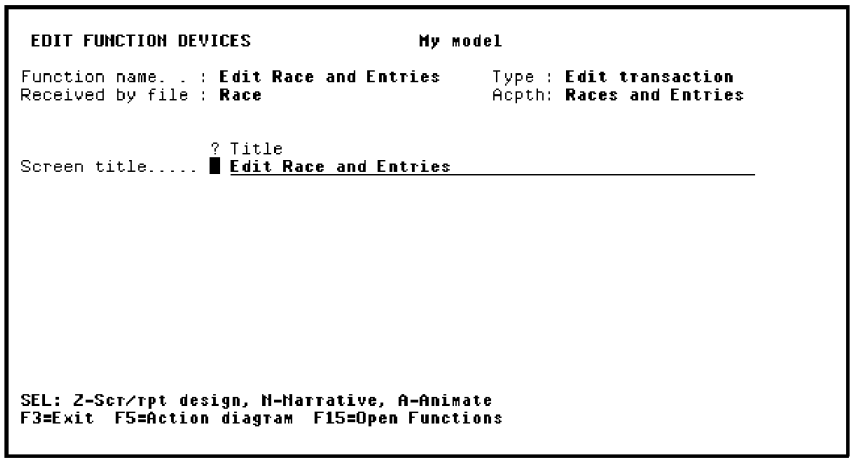
Press Enter. The final device design should look like this:



Press F3 to go to the Edit Function Devices panel.

Exiting the Device Design

You have finished creating the function Edit Race and Entries and designed an appropriate panel layout.

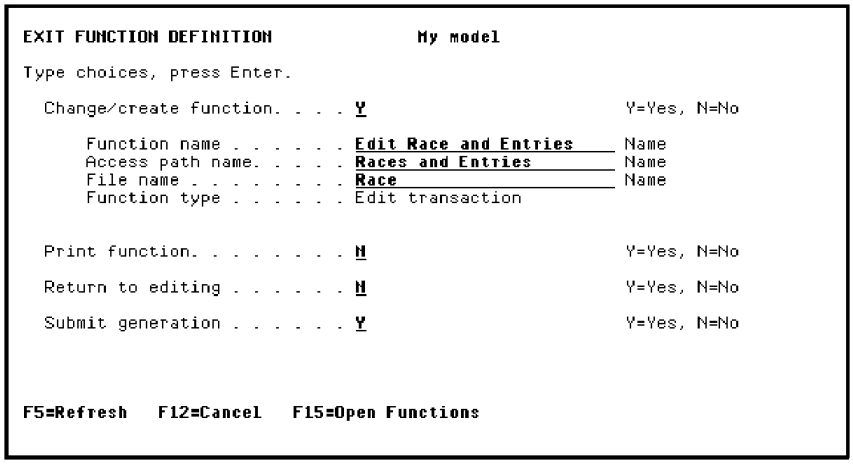


Exit from the Edit Function Devices panel by pressing F3. The Exit Function Definition panel displays.

Submitting the Function for Generation

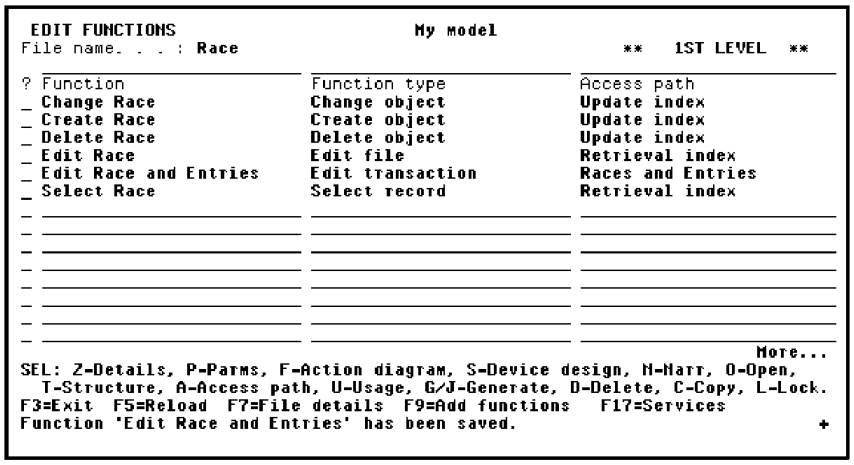
In addition to saving changes to the function, you can use the Exit Function Definition panel to submit the function and display file for generation by setting the Submit generation field to Y.

Accept the default of Y for Change/create function to save your changes and change the Submit generation field to **Y**.



Press Enter.

You will be returned to the Edit Functions panel. Messages indicating that the function has been saved and submitted for generation will appear at the bottom of the panel.



You have now created a program that allows the end user to update the two files, RACE and RACE ENTRY, simultaneously.

Exercises

Do the following exercises:

1. Submit the request for generation and compilation for the Races and Entries SPN access path and the Edit Races and Entries function. You do this by selecting the Submit model create request (YSBMMDLCRT) option from the Display Services Menu (F17). For a program to compile successfully, all files and access paths used by the program must have been generated and compiled first.

   Note: Be sure to write down the implementation name for the Edit Races and Entries function. You will need this name for testing the program using the CALL command.

2. Call the Edit Races and Entries function and add a few races to the database and some entries for each race. When you call the program, remember to specify the null or blank parameter for the return code. For example,

   call myaretr ‘’

   Try pressing F4 on fields like *Horse code* and *Jockey code* to view a selection list of available entries.

   Note: CA 2E automatically provides this prompting capability for file-to-file relations, such as, RACE ENTRY Refers to HORSE. When the end user types a **?** or user prompt (F4) for a key or foreign key field, by default CA 2E calls the related Select record function, such as, Select Horse. A function used in this way is known as a *prompt function*.

3. Call the Edit Horse program and test the link you defined between the Edit Horse program and the Display Racing Results program. You should be able to select a horse and display its racing history.

   For example, the steps required to call Display Racing results for Faithful Dobbin are:

   a. Type **/** in Subfile selector for Faithful Dobbin.

   b. Press F10 to access the action bar.

   c. Type **S** to access the Selector Choice menu.

   d. Type **1** beside the Display Racing Results action.

   e. Press Enter.

Press F3 to return to the Edit Horse function.

Chapter 8: Report Functions

This chapter introduces the two functions used to produce reports. These are:

- The Print File (PRTFIL) Function
- The Print Object (PRTOBJ) Function

You will use these functions to define the two following reports:

- A simple report of all the horses recorded in the database.
- A more complex report showing all the horses in the database and for each horse, a list of the races in which the horse competed.

This section contains the following topics:

Introduction to the Print File Function (see page 375)
Introduction to the Print Object Function (see page 397)

Introduction to the Print File Function

In this topic you will learn how to define a Print File function to produce a simple report of all the horses recorded in the database.

New terms introduced

- PRTFIL (Print File) function
- CNT (count) function field
- CUR (current report format) report context
- NXT (next report format) report context

New displays introduced

- Edit Report Design
- Display Report Formats
- Edit Report Format Details

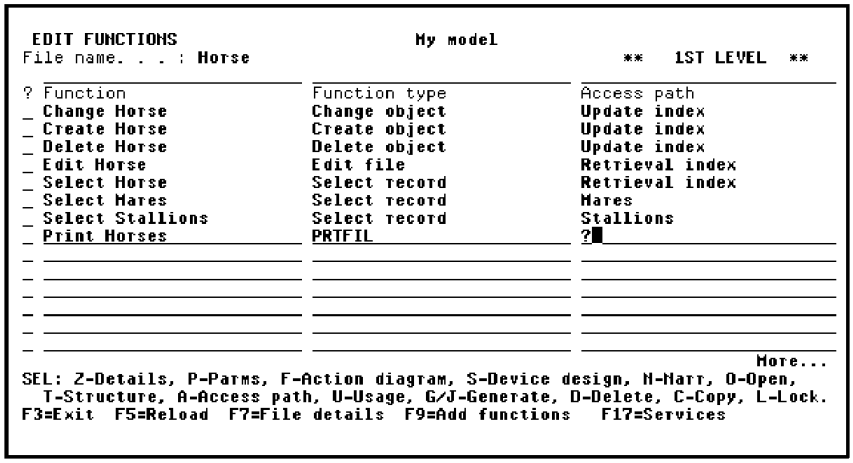
Objectives

You will be producing a report that lists all the horses recorded in your database. In addition, the report will give the total number of horses recorded and the total value of all the horses. The following steps are involved:

1. Create a Print Horses function based on the HORSE file. The function will require the creation of a RSQ (resequence) access path to retrieve records in *Horse name* order.
2. Specify the *Total horse value* and *Total number of horses* function fields on the report design.
3. Modify the layout of the report, as required.
4. Generate and compile the source for the access path and the report.
5. Test the implemented function.

Defining the Print File Function

Starting from the Edit Functions panel for the HORSE file, specify the Print Horses function. A Print File (PRTFIL) function is defined the same way as other functions. Type **Print Horses** for the Function name, **PRTFIL** for the Function type, and **?** in the access path field to view a list of existing access paths.

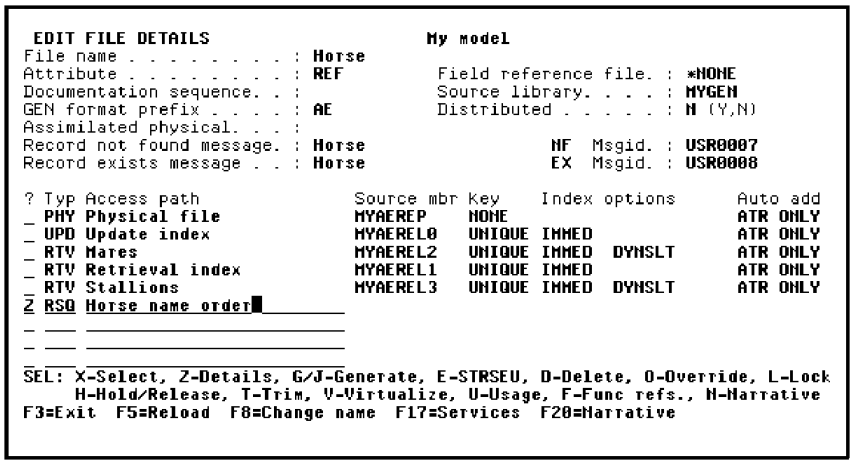


Press Enter. The Edit File Details panel displays as a result of entering **?** for the Access path.

Specifying a New Access Path

Use the Edit File Details panel for the HORSE file to create a new access path that will cause the report to print horse details in alphabetical order by Horse name, rather than in Horse code order. To achieve this, specify a new access path of type RSQ called Horse name order.

Type **RSQ** for the access path type, **Horse name order** for the Access path name, and **Z** to zoom into the access path.

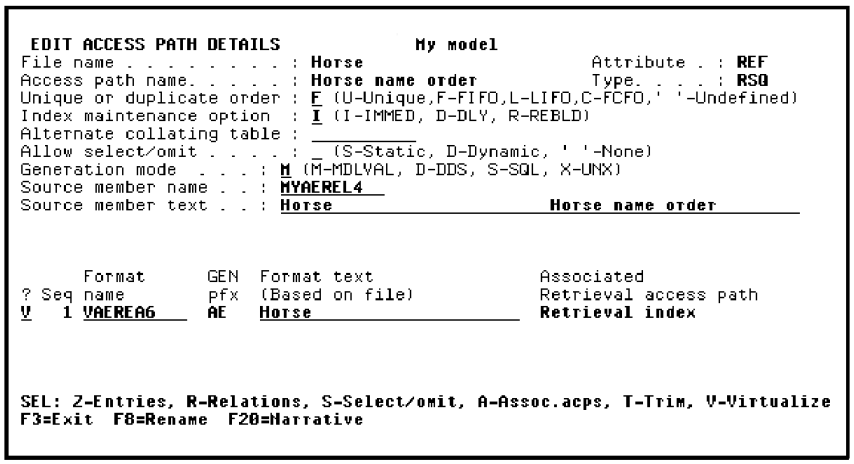


Press Enter.

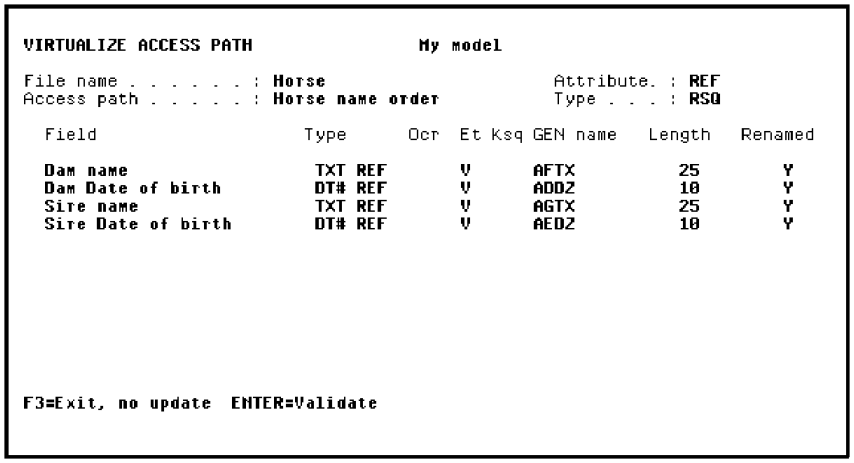
Adding Virtual Entries to the Access Path

Remember that virtual entries are not automatically added to an access path. As a result, you need to add the Dam name, Dam Date of birth, Sire name, and Sire Date of birth virtual fields explicitly so they will appear on the report you are designing.

Type **V** against the access path format.



Press Enter to display the Virtualize Access Path panel.

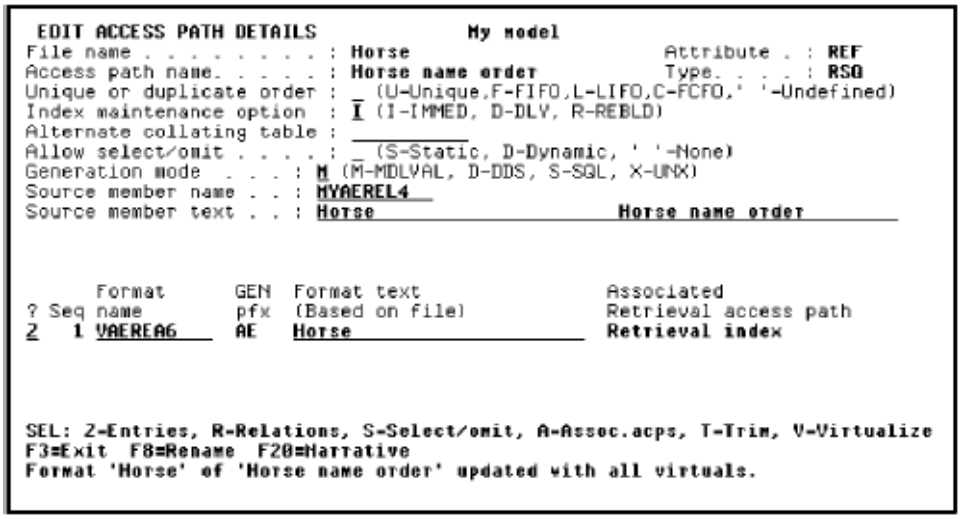


Press Enter twice to validate and confirm the virtual entries displayed. Notice the message at the bottom of the Edit Access Path Details panel.

Edit Access Path Details

You used the Edit Access Path Details panel earlier in this tutorial to specify a RSQ access path. You will use the same process here to specify a new key order on the access path format.

Type Z against the access path format.



Press Enter

Specifying Access Path Details

On the Edit Access Path Format Entries panel, specify the new key order for the RSQ access path. The initial key order default is the same as for the default Retrieval index access path.

The New Key Order

Specify an alternative key order. For this type of access path, you can choose any field except a virtual field as a key field. Use Horse name as the key.

In the Key no. column, clear the 1 next to Horse code and type **1** against Horse name.

```
 EDIT ACCESS PATH FORMAT ENTRIES      My model
File name . . . . . . . . : Horse                    Attribute . : REF
Access path name. . . . . : Horse name order         Type. . . . : RSQ
Format text . . . . . . . : Horse
Based on. . . . . . . . . : Horse                    Format No . :   1

                                    GEN                Key    Altcol Ref
? Field                             Name       Type    no. Dsc seq   cnt
_ Horse code                CDE     ADCD         K     ___              1
_ Horse name                TXT     ADTX         A     1__  █           1
_ Horse gender              STS     ADST         A     ___  _           1
_ Horse value               VAL     ABVA         A     ___  _           1
_ Date of birth             DT#     ACDZ         A     ___  _           1
_ Dam Horse code            CDE REF AFCD         A     ___  _           1
_ Dam name                  TXT REF AFTX         V                      1
_ Dam Date of birth         DT# REF ADDZ         V                      1
_ Sire Horse code           CDE REF AGCD         A     ___  _           1
_ Sire name                 TXT REF AGTX         V                      1
_ Sire Date of birth        DT# REF AEDZ         V                      1

SEL: Z-Field details, L-Locks.
F3=Exit  F7=Relations
```

Press Enter. Press F3 twice to exit.

Generating and Compiling the New Access Path

At the Edit File Details panel, request batch generation and compilation for the newly defined access path using the J option.

Type **J** next to the Horse name order RSQ access path.

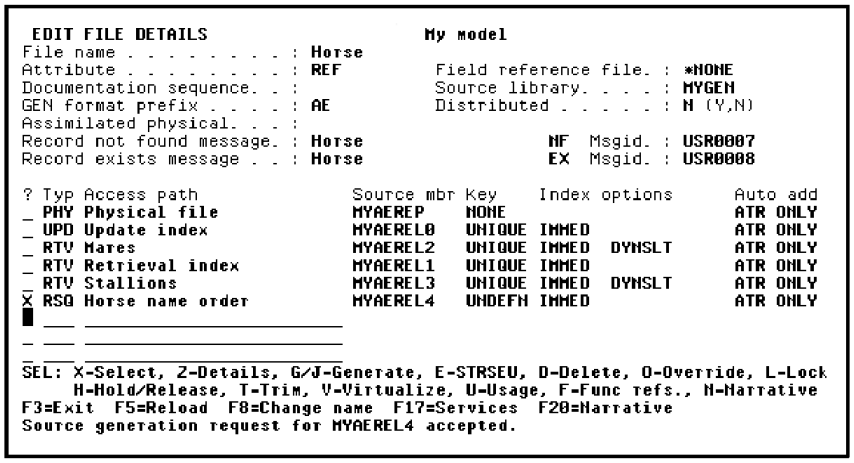
```
 EDIT FILE DETAILS                      My model
File name . . . . . . . . : Horse
Attribute . . . . . . . . : REF          Field reference file. : *NONE
Documentation sequence. . :              Source library. . . . : MYGEN
GEN format prefix . . . . : AE           Distributed . . . . . : N (Y,N)
Assimilated physical. . . :
Record not found message. : Horse                  NF  Msgid. : USR0007
Record exists message . . : Horse                  EX  Msgid. : USR0008

? Typ Access path                Source mbr Key    Index options      Auto add
_ PHY Physical file              MYAEREP    NONE                      ATR ONLY
_ UPD Update index               MYAEREL0   UNIQUE IMMED              ATR ONLY
_ RTV Mares                      MYAEREL2   UNIQUE IMMED  DYNSLT      ATR ONLY
_ RTV Retrieval index            MYAEREL1   UNIQUE IMMED              ATR ONLY
_ RTV Stallions                  MYAEREL3   UNIQUE IMMED  DYNSLT      ATR ONLY
J RSQ Horse name order           MYAEREL4   UNDEFN IMMED              ATR ONLY
█ ___ ______________________
_ ___ ______________________
_ ___ ______________________
SEL: X-Select, Z-Details, G/J-Generate, E-STRSEU, D-Delete, O-Override, L-Lock
     H-Hold/Release, T-Trim, V-Virtualize, U-Usage, F-Func refs., N-Narrative
F3=Exit  F5=Reload  F8=Change name  F17=Services  F20=Narrative
```

Press Enter to request generation.

Selecting the Access Path

Now that you have created the required access path, select it for use with the Print Horses function.

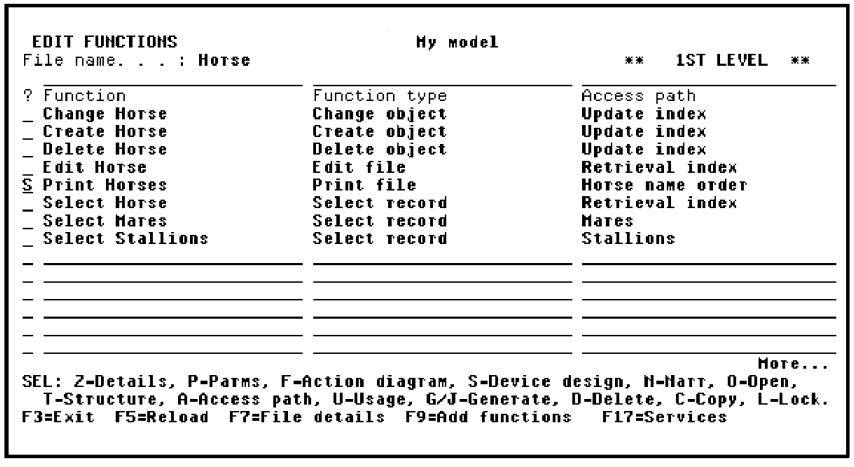
Type **X** to select the Horse name order access path.



Press Enter.

The Report Design

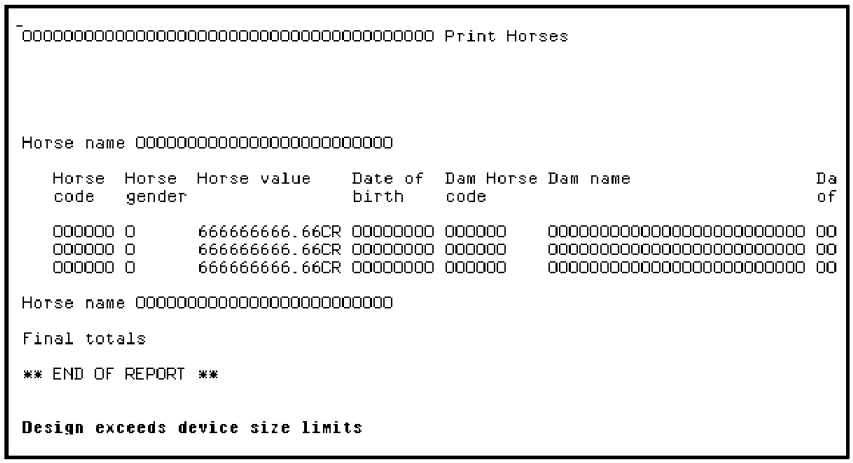
After you select the access path, CA 2E creates the Print Horses function and returns to the Edit Functions panel. From here you can display the report layout. To do so, type **S** against the function.



Press Enter.

The Default Report Layout

Initially, the report layout will look like the one shown here.



The default layout is made up of the following formats:

The page header format at the top contains a title (Print Horses) plus information such as the date and page number. The header format is repeated on each page of the printed report. An End of report format is placed at the end of the report design.

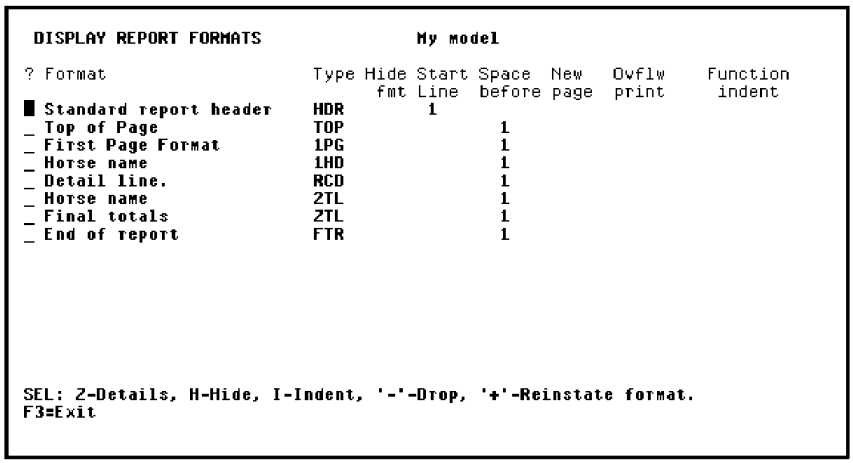
A heading format and corresponding total format is provided for each Level break. For example, for each key of the based-on access path there is a heading and total format. If the access path has a unique key, the number of level breaks is one less than the number of keys. In this case, the access path does not have a unique key. Since there is one key level, one heading format is provided that, by default, has Horse name as an output field.

The detail format contains the subfile records from the based-on access path. It is shown on the Print Horses report layout between the Horse name heading and Horse name total formats. All the fields from the access path are available in this format. The key field (Horse name) is hidden by default because it already appears on the heading format.

You can modify the report layout the same way you modified the device designs of the earlier functions. The editors for the report layout and the panel design are very similar. However, a report is by default 132 characters across and extends vertically past the bottom of the panel. You can press the Roll Up key to roll the panel by half a panel at a time.

Displaying the Report Formats

The report formats are shown on the Display Report Formats panel. To access this panel, press F17 from the device design (report layout).

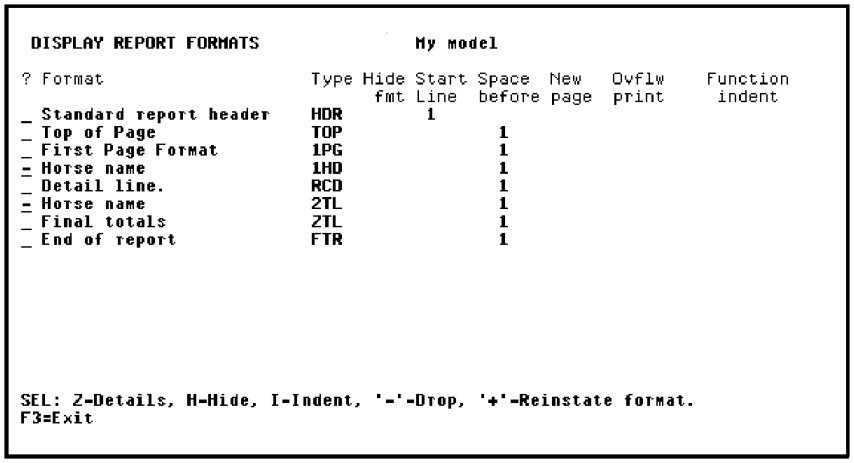


- The Top of page format (TOP) enables you to define fields to be printed at the top of each page in addition to the Standard report header (HDR). This format is empty by default.
- The First page format (1PG) can be used to add title information to the beginning of the report. This format is empty by default.
- The Level-heading format (1HD) contains fields appropriate to the key level; in this case, Horse name. By default, this format prints before the first detail within a level break.
- The Detail line format (RCD) contains the records from the access path on which the Print File function is based. By default, this format is printed for every record read. This format cannot be dropped, only hidden.
- The Level total format (2TL) contains fields appropriate to the key level. By default, this format is printed after the last detail record or subtotal within a level break.
- The Final total format (ZTL) contains totals of all previous total levels. By default, this format contains only the constant Final totals; you can add your own function fields to provide totals.

Dropping Formats from a Report Design

In this step you will remove the Horse name heading and total formats. To remove a format, you can either hide it or drop it completely. If you hide a format it will still be logically present. In this example, you will drop both formats completely.

Type **'-'** against the two Horse name formats.

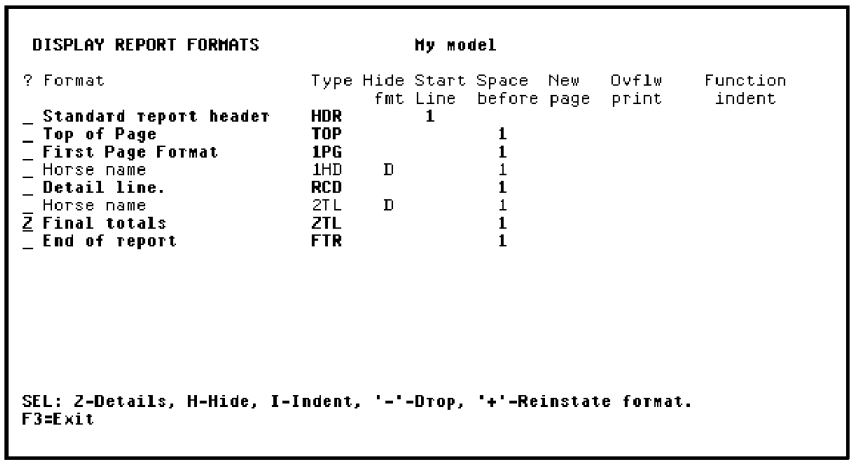


Press Enter. Notice the D in the Hide fmt column for the two formats you dropped.

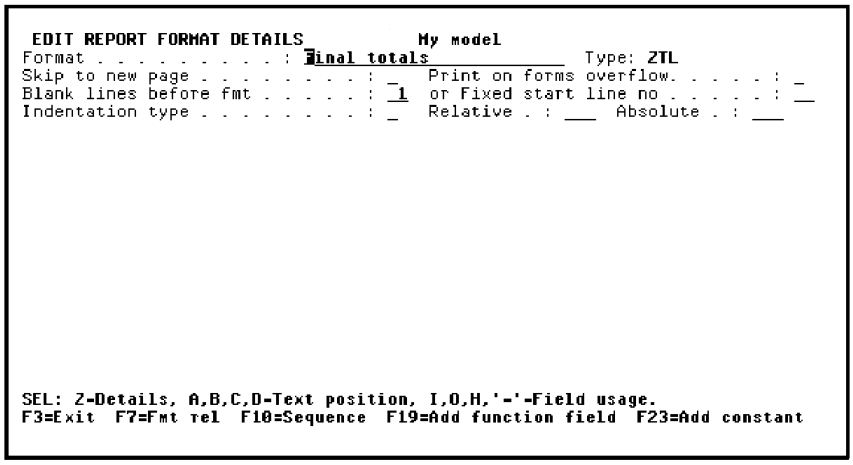
Adding Function Fields

In this step, you will add two function fields to the Final totals format (ZTL).

Type **Z** against Final totals format.



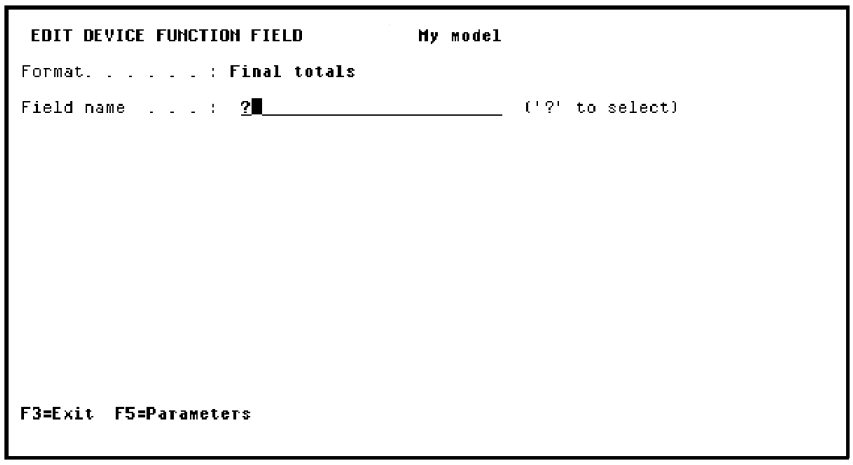
Press Enter to display the Edit Report Format Details screen for the Final totals format.



Press F19 to add a function field to this format using the Edit Device Function Fields panel.

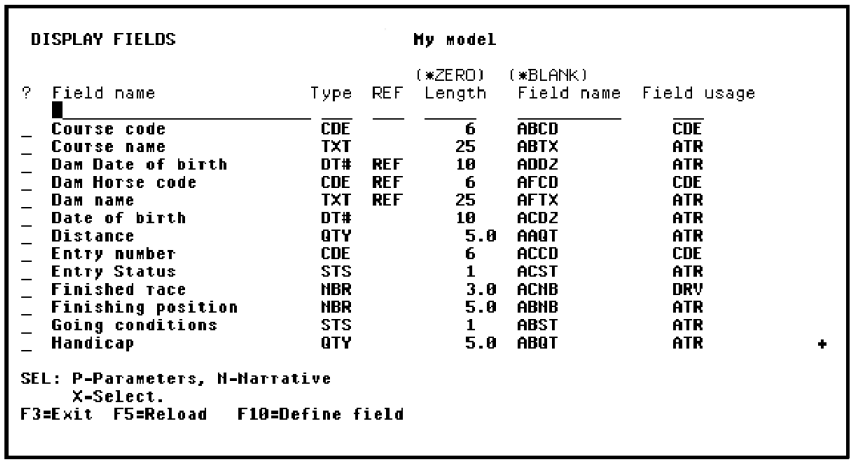
Displaying Existing Function Fields

To display existing function fields, type **?** for Function field.



Press Enter.

All fields defined in your model are displayed, including the two function fields you defined for the Edit Race and Entries function (EDTTRN).



```
 DISPLAY FIELDS                          My model

                                        (*ZERO)  (*BLANK)
?  Field name                 Type  REF  Length   Field name  Field usage
   ____________________________ ___  ___  _____    __________    ___
_  Course code                 CDE           6     ABCD          CDE
_  Course name                 TXT          25     ABTX          ATR
_  Dam Date of birth           DT#  REF     10     ADDZ          ATR
_  Dam Horse code              CDE  REF      6     AFCD          CDE
_  Dam name                    TXT  REF     25     AFTX          ATR
_  Date of birth               DT#          10     ACDZ          ATR
_  Distance                    QTY           5.0   AAQT          ATR
_  Entry number                CDE           6     ACCD          CDE
_  Entry Status                STS           1     ACST          ATR
_  Finished race               NBR           3.0   ACNB          DRV
_  Finishing position          NBR           5.0   ABNB          ATR
_  Going conditions            STS           1     ABST          ATR
_  Handicap                    QTY           5.0   ABQT          ATR          +

SEL: P-Parameters, N-Narrative
     X-Select.
F3=Exit  F5=Reload   F10=Define field
```

To define two new function fields, press F10.

Defining New Function Fields

The first new function field, Total horse value, will provide a total of all the Horse value fields on the detail format. A function field of type SUM is used for this purpose. Recall that you used a function field of this type previously to calculate the Total Number of finishers in a race.

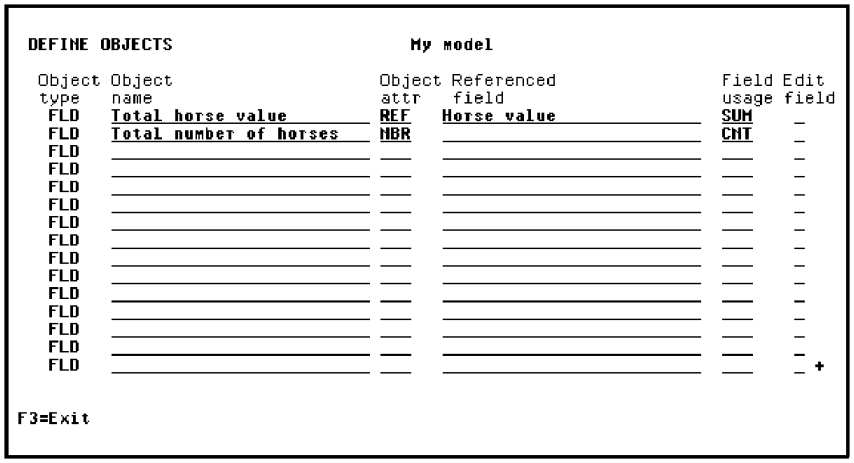
The second new function field, Total number of horses, will calculate the total number of horses recorded. It will count the number of occurrences of Horse code field on the detail format. A function field of type CNT can be used for this purpose.

There are two ways to define a function field:

- As a REF field based on the field you wish to sum or count
- As a new field with input parameter(s) specified separately.

You will use the REF field method to define the Total horse value function field. However, you cannot use this method for the Total number of horses function field since the CNT field must be numeric and the field you are counting is alphanumeric (CDE).

Define the Total horse value and Total number of horses function fields.



Press Enter.

Specifying Function Parameters

When you pressed Enter, the two new function fields were created. Press Roll Up twice to view the two new function fields.

The Total horse value function field is now fully defined. However, you must still specify the field used to arrive at the Total number of horses function. Type **P** next to this function field as shown to specify its parameters.

```
 DISPLAY FIELDS                          My model

                                          (*ZERO)   (*BLANK)
?  Field name                  Type  REF   Length    Field name  Field usage
_  Sire Horse code             CDE   REF      6      AGCD        CDE
_  Sire name                   TXT   REF     25      AGTX        ATR
_  Total horse value           VAL   REF     11.2    ACVA        SUM
P  Total number of horses      NBR           5.0    AENB        CNT

SEL: P-Parameters, N-Narrative
     X-Select.
F3=Exit  F5=Reload   F10=Define field
```

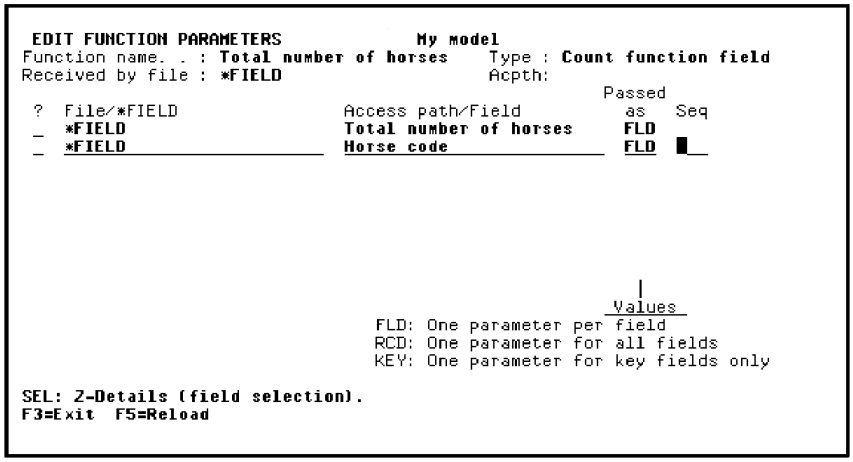
Press Enter.

The Default Parameters

As mentioned previously, the function field itself is the output parameter. In this step, you need to specify an input parameter, which is the field to be counted. In this case, the Horse code field is the input parameter.

You can specify function parameters in two different ways. These can either be individual fields or an access path from which you select individual fields. In this example, you will specify the parameter as an individual field.

Type the parameter details.



Press Enter.

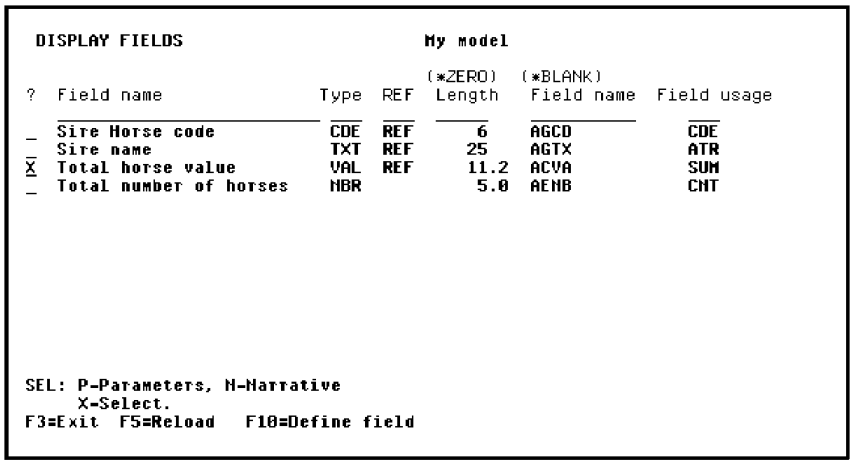
The parameter usage defaults to input. You can check this for yourself by typing **Z** against Horse code on the Edit Function Parameters panel.

Press F3 to exit.

Selecting the Function Fields

You have finished defining the two new function fields. You must now select them to appear on the Final totals format.

Type **X** against Total Horse value.

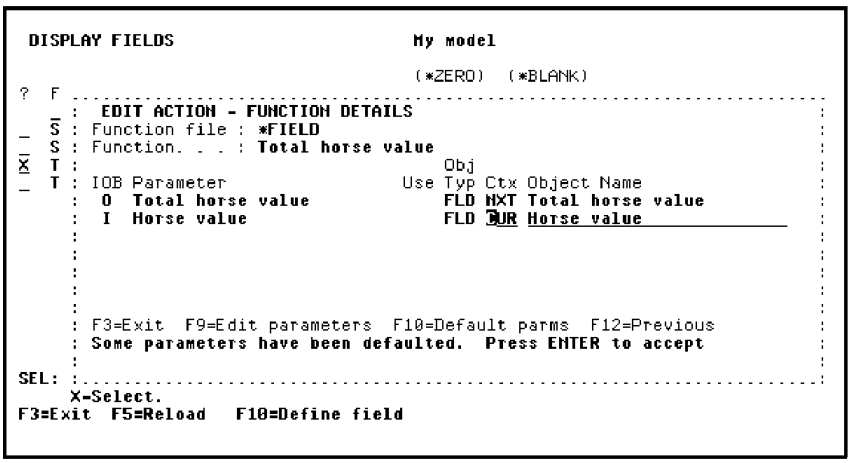


Press Enter.

Confirming the Function Details

The Edit Action - Function Details window will prompt you to confirm details of the function parameters. This panel shows the contexts from which the values of the parameters will be taken: NXT (next report format) and CUR (current report format).

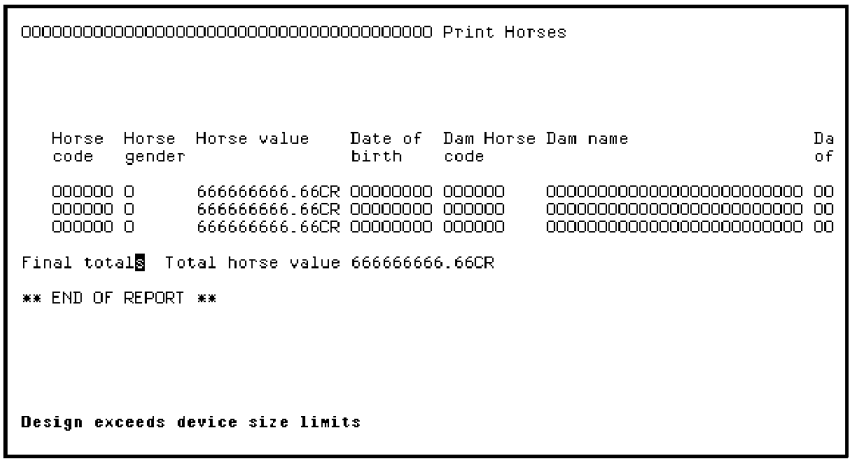
The values for the Horse value parameter are taken from the CUR context. This context corresponds to the format upon which the processing of the function field takes place (the Detail line format). The NXT context corresponds to the format that follows the Detail line. In this case, the NXT context is the Final totals format.



Press Enter to confirm the parameters. Press F13 to return to the report device design.

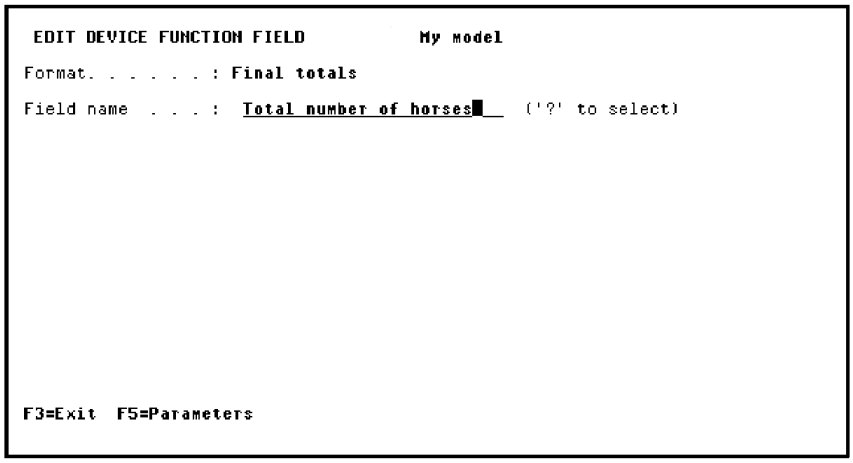
Adding the Total Number of Horses Function Field

The Total horse value function field is now present on the Final totals format. To add the other function field, position the cursor on the Final totals format.



Press F19 to add the Total number of horses function field.

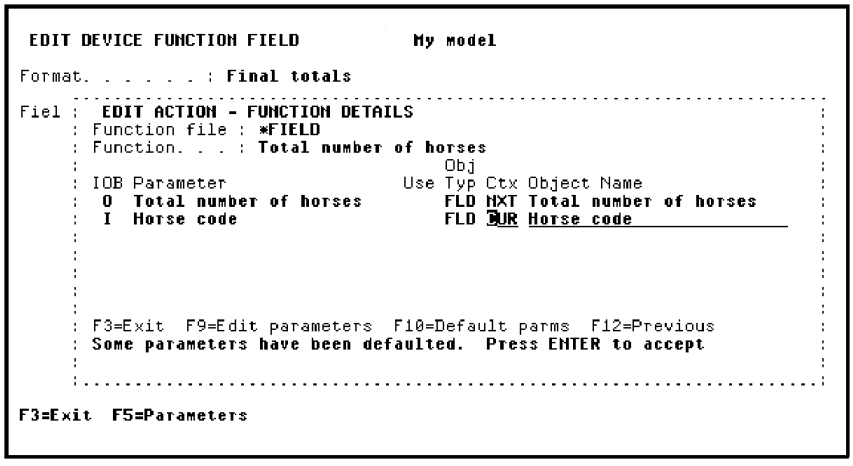
Type **Total Number of horses** in the Field name field.



Press Enter.

Confirming the Parameters

CA 2E automatically provides defaults for the parameters and contexts. In this case you need only accept them.



Press Enter to accept the parameters and return to the report device design.

Exercise

Both function fields are now present on the Edit Report Format Details panel of the Final totals format. Check this yourself by pressing F17 from the report device design and typing **Z** against the Final totals format on the Display Report Formats panel. Press F13 to return to the report device design.

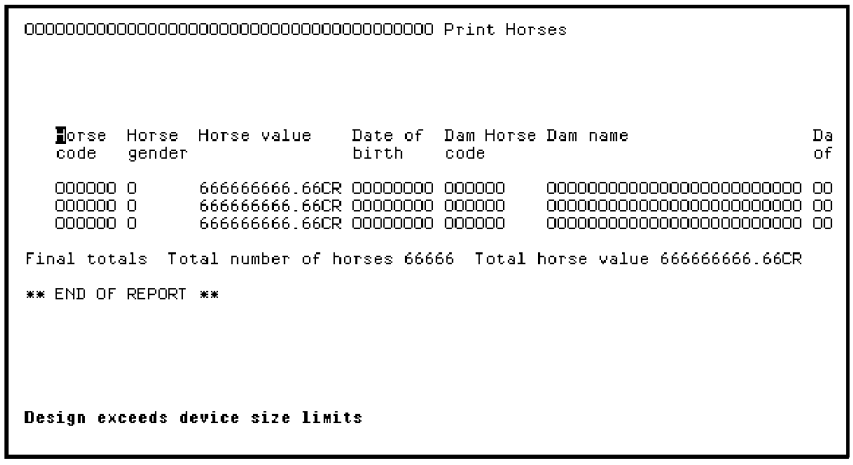
The Report Design with Function Fields

The report layout should now look like the one below, with the function fields displayed on the Final totals format. The words *Final totals* are a constant and can be removed.

Completing the Report

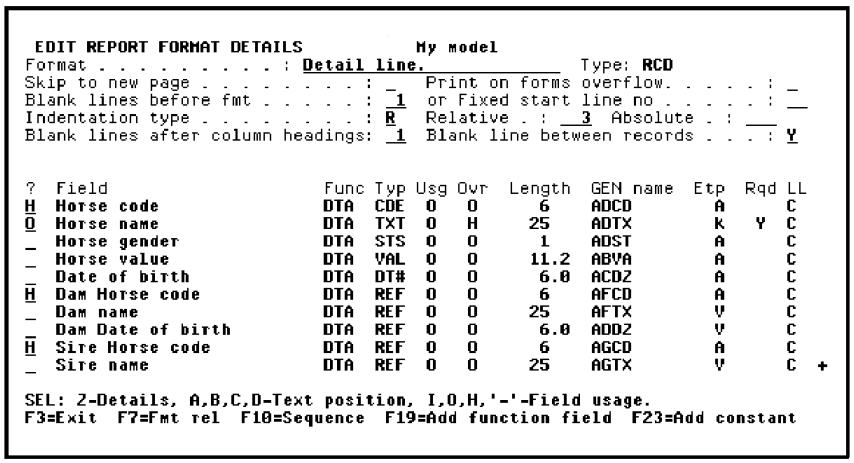
In this step, to make the report easier to read, you will specify a blank line between records and hide the Horse code, Dam code, and Sire code fields. In addition, you will change the Horse name field from hidden to output. By default, the Horse name field is hidden because it was originally present on the Horse name header format. Since the Horse name header format has been dropped, it now makes sense to have the Horse name appear on the Detail line format.

Place the cursor on the Horse code field.



Press F5.

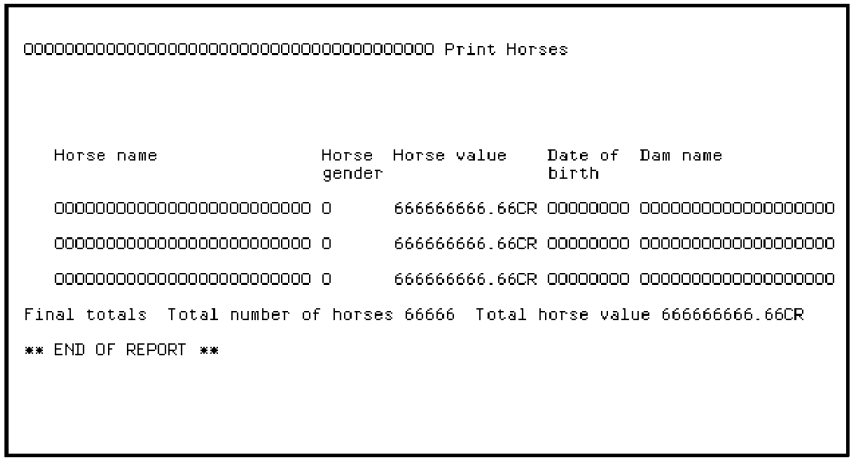
Type the options as shown; namely, type **Y** for Blank line between records, type **O** against Horse name, and type **H** against Horse code, Dam Horse code, and Sire Horse code to hide these fields on the report.



Press Enter. Note the changes in the Ovr column as a result of your entries. Press F3 to return to the device design.

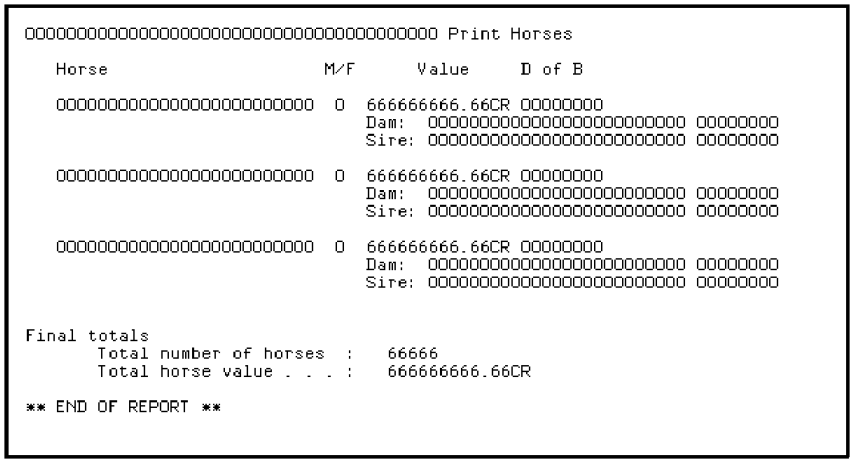
The Modified Report Device Design

The report device design should now look like the one shown below. Since you have hidden some fields on the Detail format, the design no longer exceeds the 132 character size limit.



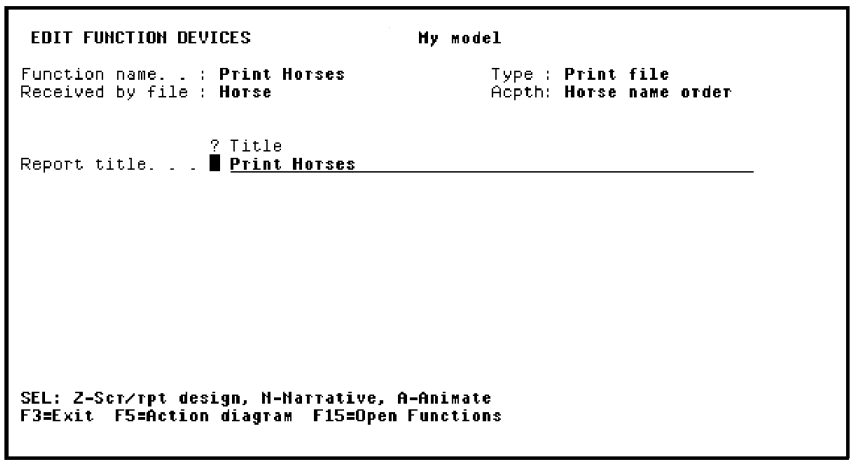
The Completed Report Layout

Update the report device design by moving fields and changing labels and label spacing until the design meets your needs. Here's an example.

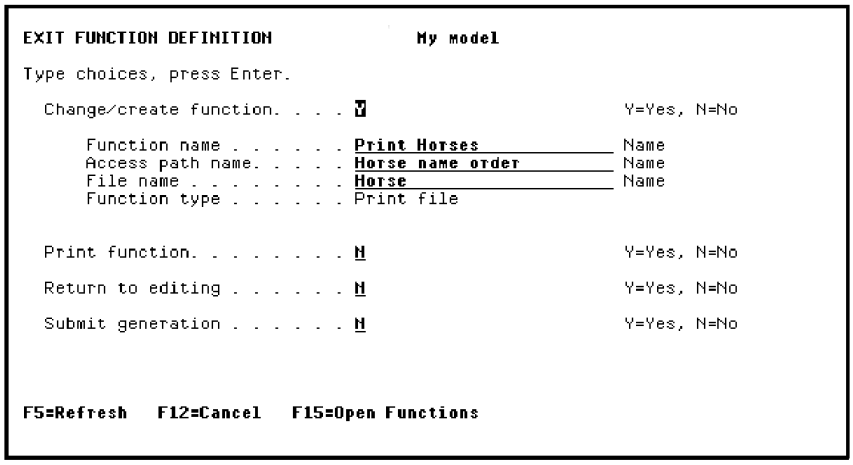


Note that to produce this design you will need to drop the Top of Page format (TOP) and the First Page Format (1PG) using the method you used to drop the 1HD and 2TL formats previously; hint, press F17.

When you have finished the report layout, press F3 to exit the device design.



Press F3.



Saving the Report Device Design

Save the changes by accepting the defaults and pressing Enter.

Generating and Compiling the Function

You have now finished specifying the Print Horses function. Type **J** next to this function on the Edit Functions panel and press Enter to request batch generation of the Print File and program that will implement the function.

Finally, compile the generated source for the function and the access path. Press F17 to access the Display Services Menu and select the Submit model create request (YSBMMDLCRT) option. Be sure to note the implementation name (member) of the Print Horses function so you can call it in the next step.

Running Your Program

When the Print Horses function has successfully compiled, try it out by calling the program from a command line. For example,

```
call myaspfr ''
```

Following is an example of the resulting report showing data you entered earlier using the Edit Horse program.

Your company name here Print Horses

Horse	M/F	Value	D of B
Bonfire	M	5000.00	2/01/83
		Dam:	
		Sire:	
Faithful Dobbin	F	3500.00	5/31/86
		Dam:	
		Sire:	
Pegasus	M	3000.00	6/23/92
		Dam: Faithful Dobbin	5/31/86
		Sire: Bonfire	2/01/88

Final totals

Total number of horses	:	3
Total horse value . . .	:	11500.00

** END OF REPORT **

Use the i OS Work with Spooled Files (WRKSPLF) command to view or print your report.

Introduction to the Print Object Function

You will learn how to embed a Print Object function into the existing Print File function to produce a more detailed report.

New terms introduced

- PRTOBJ (Print Object) function
- QRY (Query) access path

New displays introduced

- Edit Device Structure

Objectives

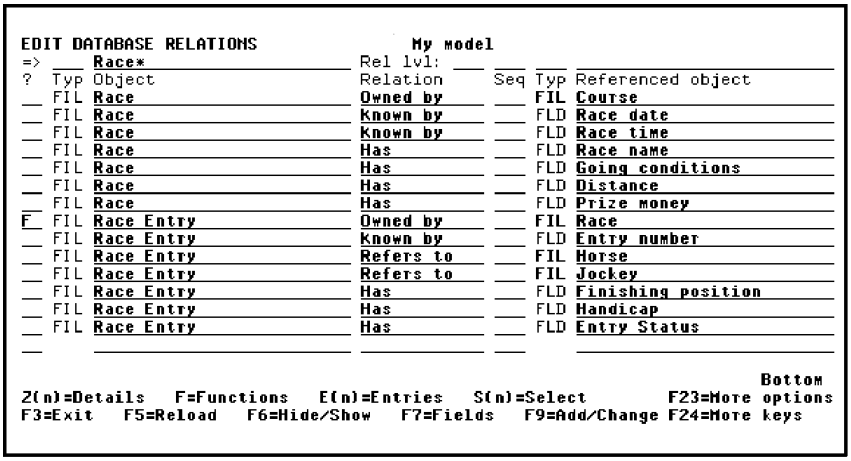
At this point, you have created a function to print a list of all the horses stored in the database. A more useful report might also list the races for which each horse has been entered. To produce such a report, you need to combine records from the RACE ENTRY file with the existing Print Horses function. You can achieve this using a Print Object function. Use the following steps to do this.

1. Specify a Print Object function based on the RACE ENTRY file.
2. Define a Query access path to be used with the RACE ENTRY file.
3. Combine the Print Object function with the Print File function and specify the parameters to be passed between them.
4. Generate and compile the new Print Horses function.
5. Test the program.

Defining the Print Object Function

To define a Print Object function, start from the Edit Database Relations panel and go to the Edit Functions panel on the RACE ENTRY file.

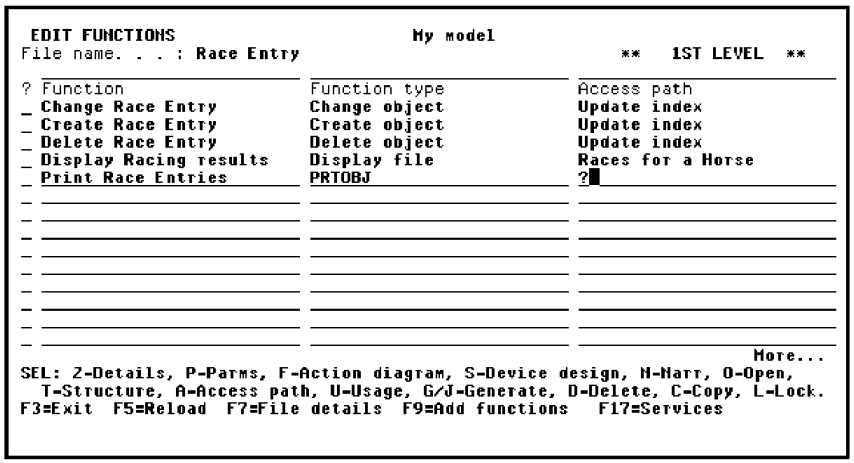
Type **F** against RACE ENTRY.



Press Enter.

Adding a Print Race Entries Function

Specify the Print Object function by typing **Print Race Entries** for the function name, **PRTOBJ** for the function type, and **?** to display the available access paths.



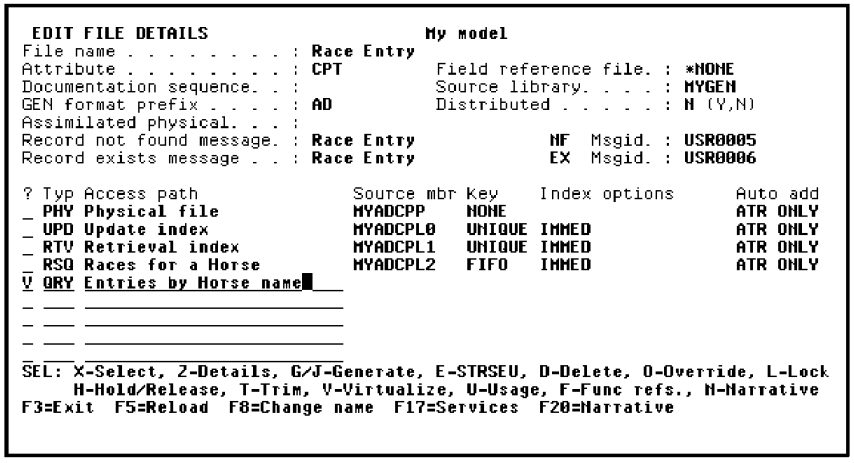
Press Enter to display the Edit File Details panel.

Specifying a Query Access Path

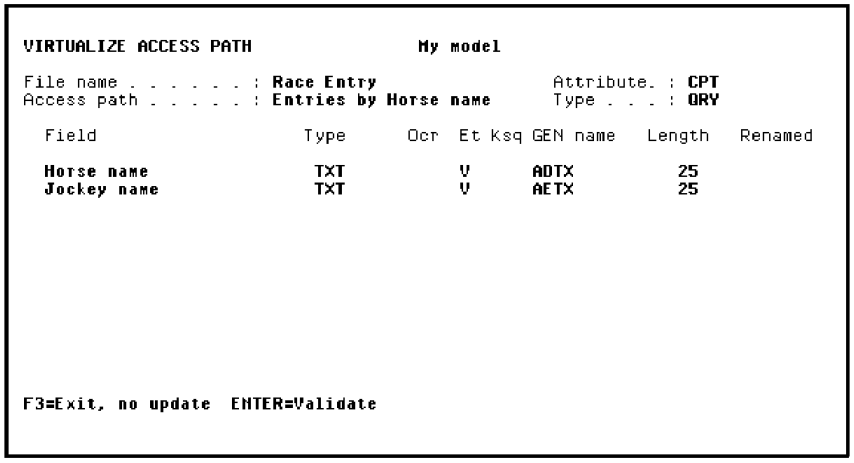
The Edit File Details panel shows the existing access paths for the RACE ENTRY file. You will create a new access path for the Print Race Entries function.

To produce the desired report, the records in the RACE ENTRY file need to be processed in order by Horse name. To do this, you need to create an access path keyed on the Horse name field; however, Horse name is a virtual field on the RACE ENTRY file. In order to use a virtual field as a key, you need to create a query (QRY) access path. In addition, since virtual entries are not automatically included on an access path, you need to add them explicitly.

Specify a QRY access path named **Entries by Horse name** and type **V** against it.

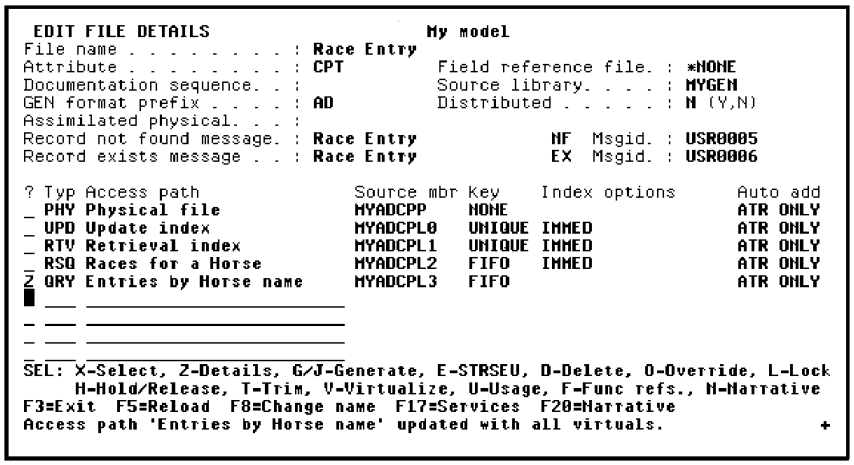


Press Enter to view the virtual entries that are available for inclusion on the access path; namely, Horse name and Jockey name.



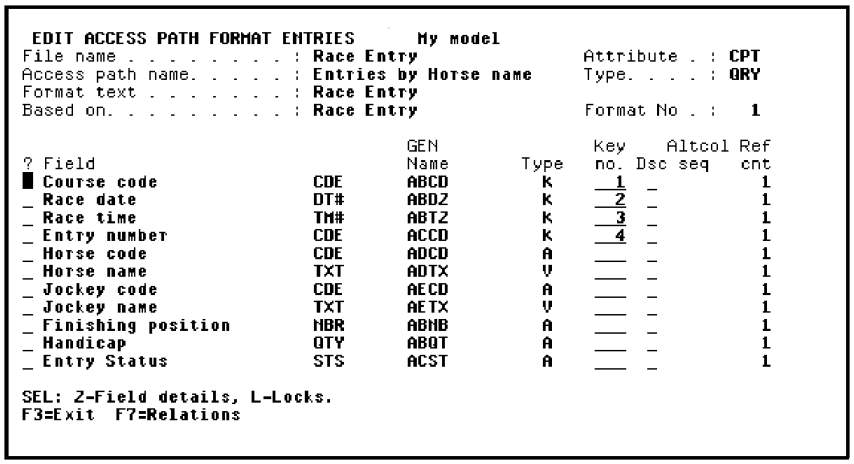
Press Enter twice to validate and confirm the inclusion of the two virtual entries and return to the Edit File Details panel. Note the message at the bottom of the panel indicating that the access path has been updated with all virtuals.

Type **Z** against the Entries by Horse name access path to begin the process of specifying Horse name as the key.



Displaying the Access Path Format Entries

Press Enter to display the Edit Access Path Details panel. Next, type **Z** against the displayed format and press Enter to access the Edit Access Path Format Entries panel.

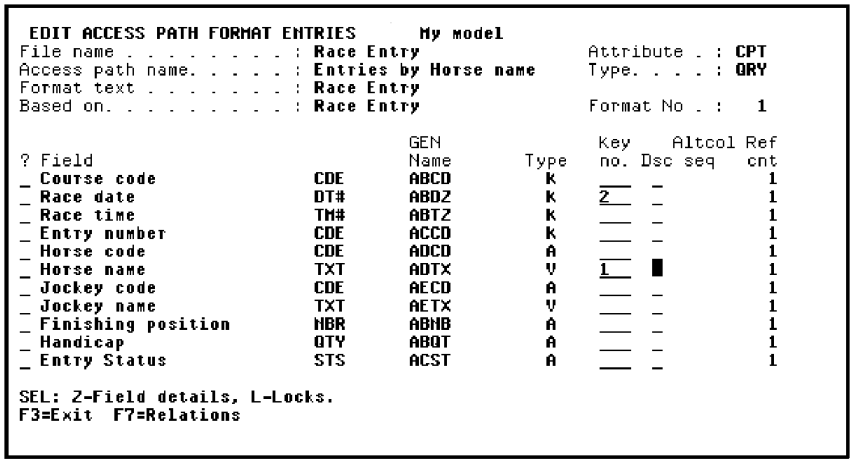


On the Edit Access Path Format Entries panel, the key order defaults to that of the RTV access path. Note that all fields, including virtual fields, are available as key fields.

Specifying an Alternative Key

Specify Horse name and Race date as the alternate keys of the access path. You can assume that each horse runs in no more than one race per day; as a result, Race time is not required as a key.

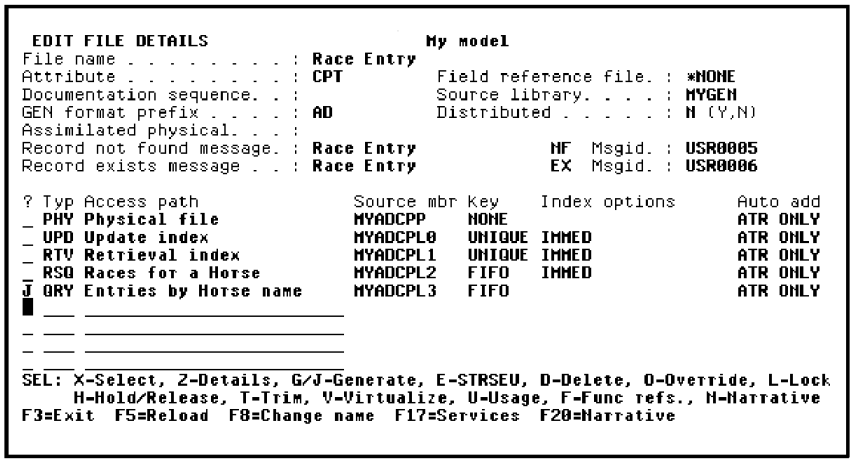
Define the keys by changing entries in the Key no. column as shown; namely, blank the Key no. field for Course code, Race time, and Entry number, and type **1** for Horse name.



Press Enter. Press F3 twice to exit to the Edit File Details panel.

Compiling the QRY Access Path

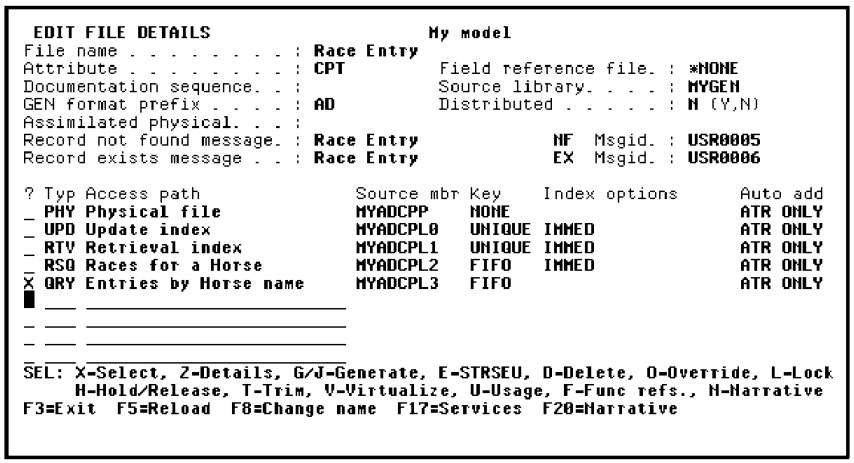
The access path you have just defined must be generated and compiled before running the application. You have the option to add it to the job list now. To do so, type **J** next to the QRY access path.



Press Enter.

Selecting the Access Path for the Function

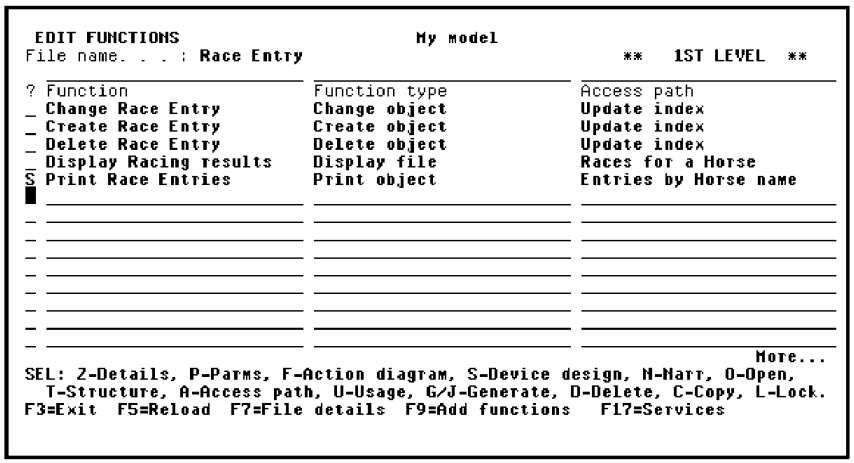
Now that you have defined the QRY access path, you need to select it for the Print Race Entries function. To do so, type **X** next to the QRY access path.



Press Enter.

Displaying the Report Device Design

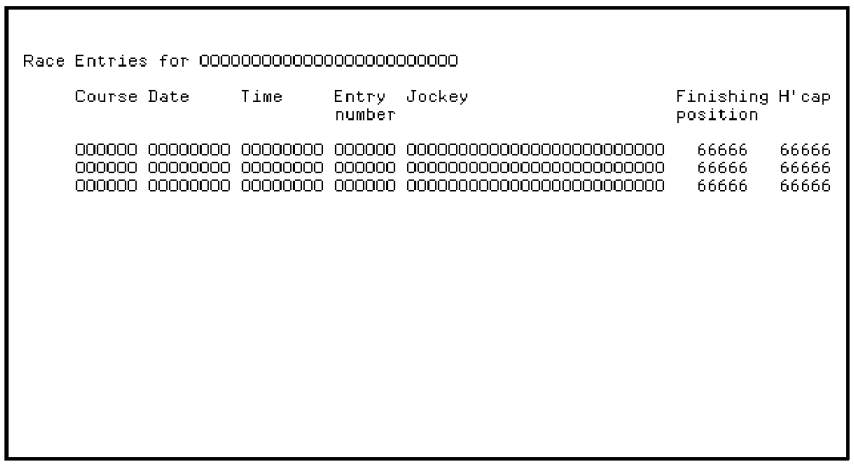
Now that you have created the PRTOBJ function, check the device design to see whether the default options meet requirements. To access the device design, type **S** next to the Print Race Entries PRTOBJ function from the Edit Functions panel.



Press Enter.

The Default Report Device Design

Initially, the report device design should look like the one shown.



The default layout is obtained in a similar way to the previous Print File function. The access path has a non-unique key. A heading and total format are provided for each key of the access path. All the fields from the access path format are output on the detail line records, except for the key fields, which are hidden.

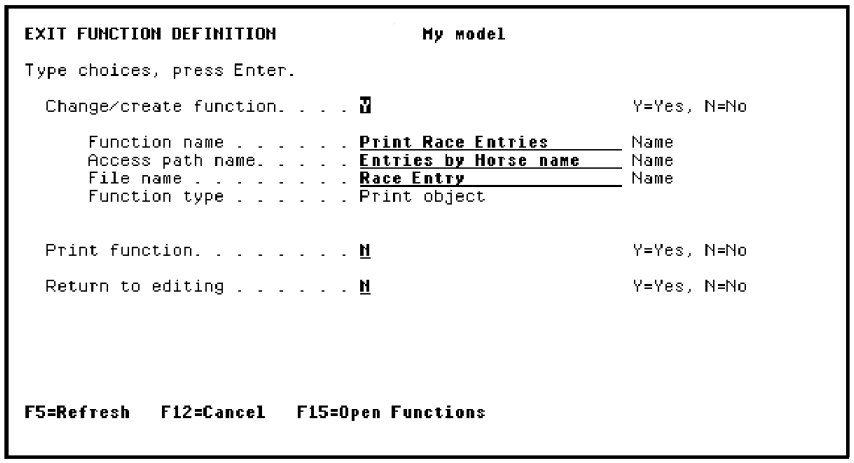
A PRTOBJ function is an internal function. It does not represent a complete report and must be embedded within an existing Print File function. The embedding Print File defines the standard header format, page size, and additional details for the combined report.

Exercise - Updating the Report Layout

Update the report layout using the same techniques as those in the device design updates you completed earlier in this tutorial. Perform the following steps.

1. Display the report formats (F17) and drop the following formats: Race date header, Final totals, Horse name total, and Race date total.
2. On the Detail line format, change the usage of the Race date field from hidden to output.

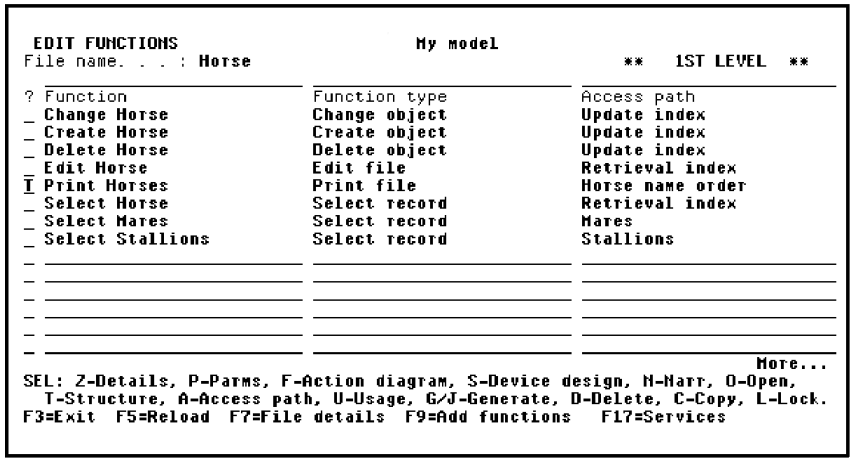
   Hide all fields that are not required and adjust labels and label spacing until the design resembles the one shown.



3. Press F3 twice to exit.

Exiting the Device Design

Save your design by accepting the defaults on the Exit Function Definition panel.



Press Enter to return to the Edit Functions panel.

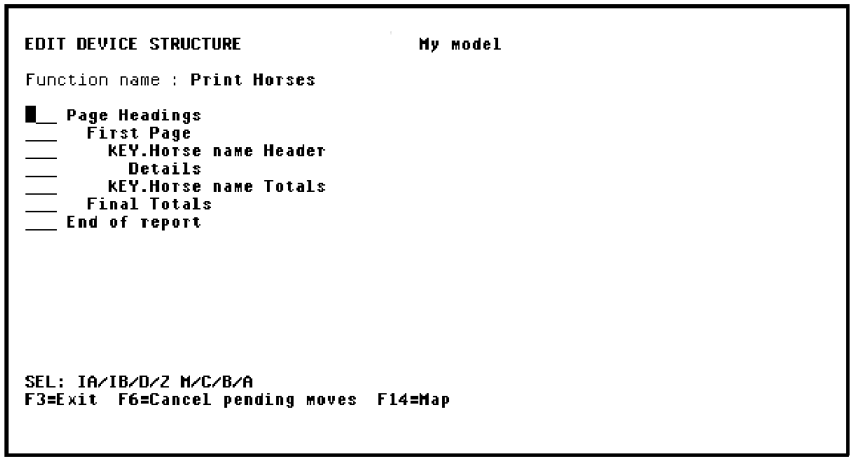
Combining the Report Functions

You have just finished defining the two report functions. Press F3 to exit the Edit Functions panel for the RACE ENTRY file and return to the Edit Database Relations panel.

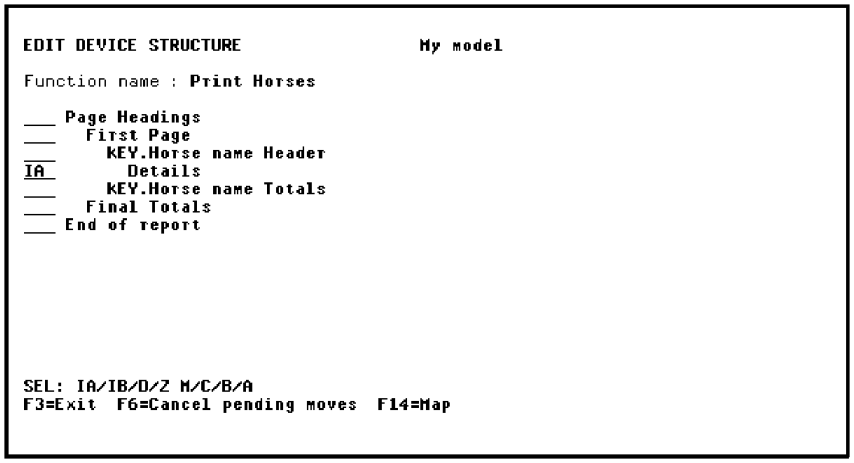
Go to the Edit Functions panel for the HORSE file. To do this, type **Horse*** on the selection line and press Enter. Then type **F** next to any HORSE relation and press Enter.

The next step is to combine the two functions so that the records from the Print Object function (each race entry) will be printed after each detail record in the Print File function (after the horse to which it applies). To do this, you will use the Edit Device Structure panel.

Type **T** against the Print Horses function.



Press Enter.

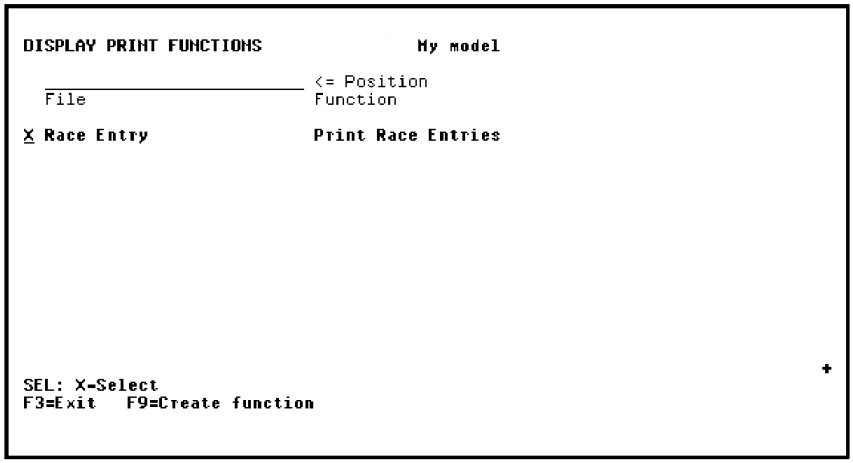


Embedding the Print Race Entries Function

The Edit Device Structure panel represents the layout of the report in terms of the formats from which it is constructed. This display is only available for functions of type PRTFIL or PRTOBJ. It has facilities for embedding PRTOBJ functions within the report and moving and copying them.

Many Print Object functions can be embedded within the same Print File function. A Print Object function can contain other embedded Print Object functions. This makes it possible to construct complex reports. In this tutorial, you will be embedding one Print Object function in the previously defined Print File function.

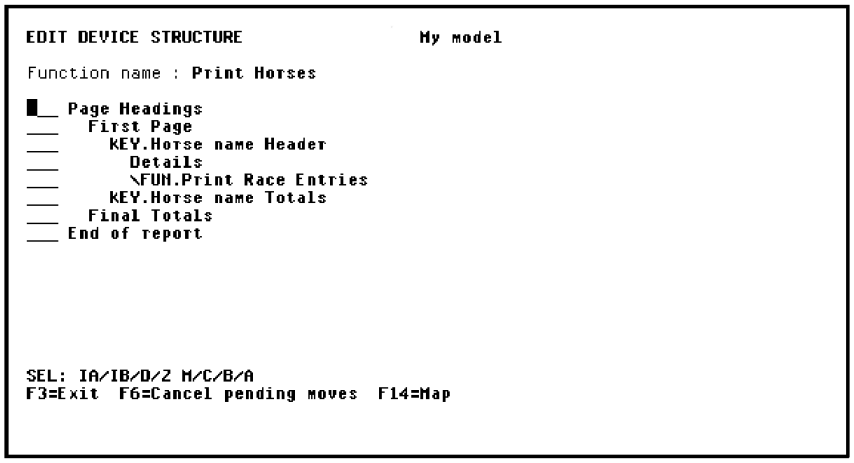
Insert a Print Object function after the Details format by typing **IA** (Insert After) in the Subfile selector.



Press Enter.

Selecting the PRTOBJ Function

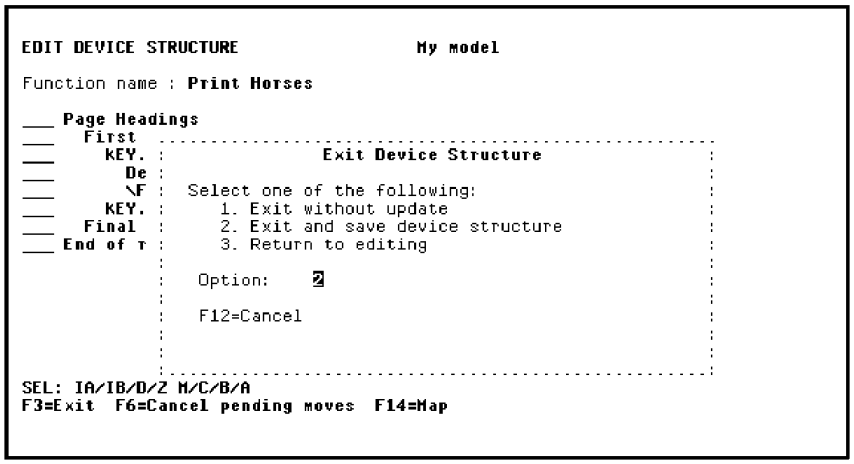
The Display Print Functions panel is now shown. This panel displays a list of all the Print Object functions that are available for inclusion in the Print File function. In this case there is just one, Print Race Entries. Type **X** to select the Print Race Entries function.



Press Enter.

The Modified Device Structure

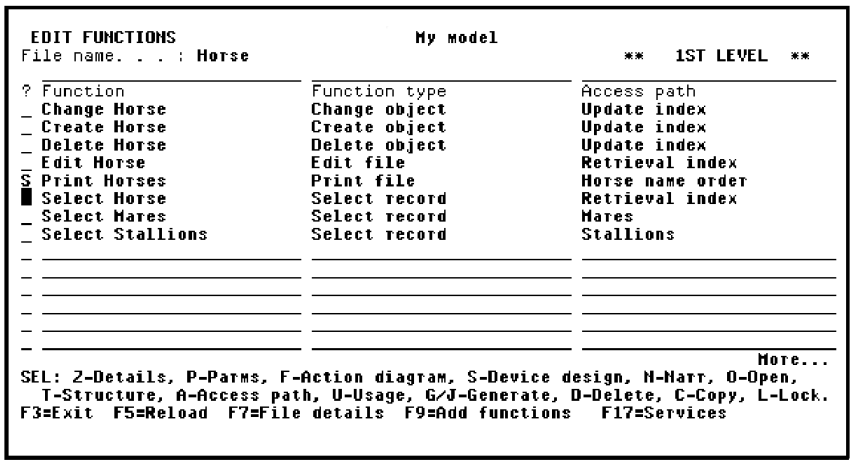
The modified device structure is shown below. The backslash character indicates that the Print Race Entries function is attached to the preceding Details format and will be printed after each detail record.



Press F3 to exit.

Saving the Device Structure

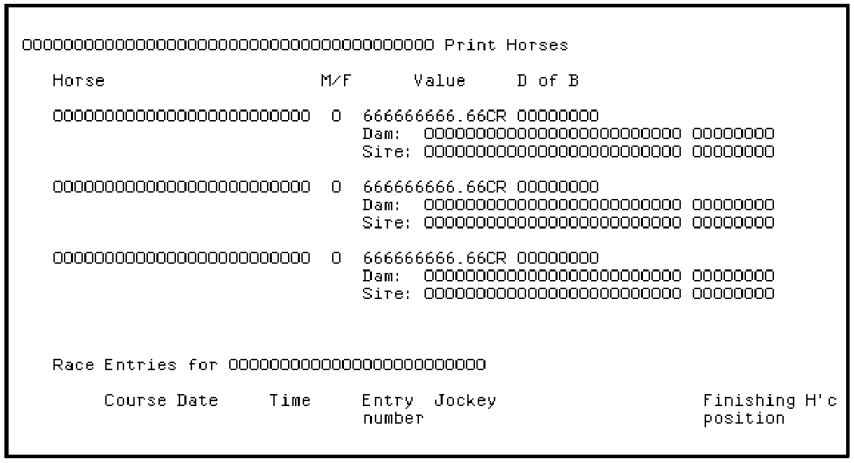
Save the modified device structure by selecting option **2** from the Exit Device Structure panel.



Press Enter to return to the Edit Functions panel.

Displaying the Report Device Design

Examine the effect of the new device structure on the layout of the Print Horses report device design. Type **S** next to the Print Horses function on the Edit Functions panel.



Press Enter.

The New Report Device Design

The modified report device design is shown with the fields resulting from the embedded Print Object function.

View the Rest of the Report Device Design

You can display the rest of the report device design by pressing Page Down (or Roll Up depending upon your keyboard).

```
EXIT FUNCTION DEFINITION                My model

Type choices, press Enter.

  Change/create function. . . . Y                        Y=Yes, N=No

      Function name . . . . . . Print Horses             Name
      Access path name. . . . . Horse name order         Name
      File name . . . . . . . . Horse                    Name
      Function type . . . . . . Print file


  Print function. . . . . . . . N                        Y=Yes, N=No

  Return to editing . . . . . . N                        Y=Yes, N=No

  Submit generation . . . . . . N                        Y=Yes, N=No



F5=Refresh   F12=Cancel   F15=Open Functions
```

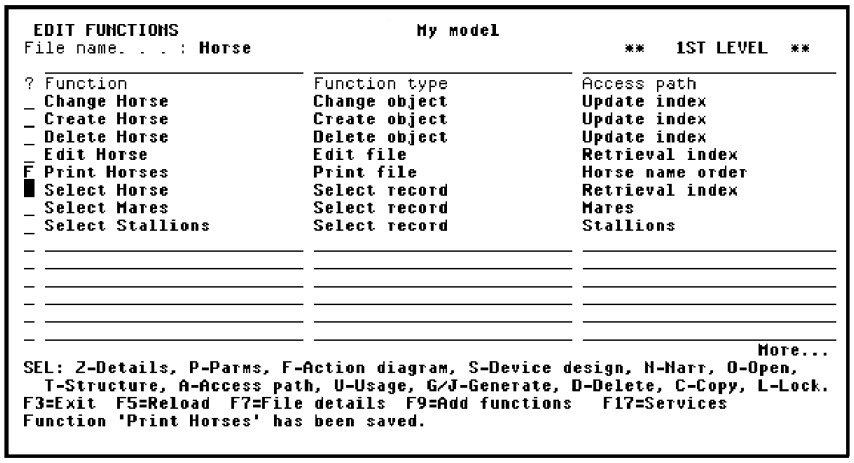
Note: If your report device design does not look like that shown above, adjust it until it looks similar. Do not be concerned with exceeding the panel limits. For reports, CA 2E allows 132 characters per line.

Modifying the Combined Report Device Design

You are restricted in the placement of the fields from the Print Object function. Place the cursor on any PRTOBJ field and press F10. All the PRTOBJ fields are moved in a single block. In other words, you cannot manipulate the PRTOBJ fields individually from this panel. You are only allowed to alter the indentation of the whole PRTOBJ function relative to the left-hand margin of the PRTFIL function in which it is embedded. It is possible to specify individual formats of the PRTOBJ. You can specify an absolute indentation that is fixed relative to the left-hand margin of the report, regardless of any adjustments made to the indentation of the PRTOBJ as a whole.

In the current report design, the PRTOBJ fields are displayed after all the detail line records. This implies that all the horses will be printed followed by all the race entries. The advantage of the report as shown in the above panel is that it enables you to easily distinguish between the different formats. However, to meet the original requirements, you need to change the report functions to print one horse at a time followed by the corresponding race entries. You will do this in the next step.

Press F3 twice to exit the Device Design Editor. Save your design by accepting the default options.



Press Enter.

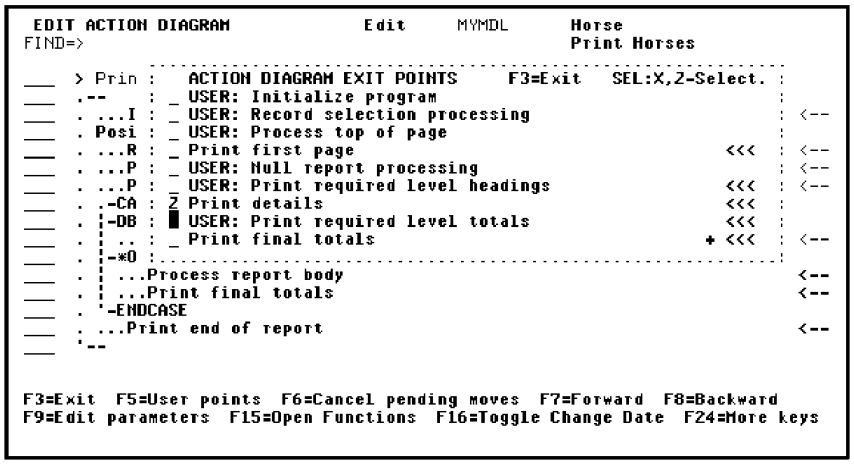
Specifying Parameters

You have now combined the two report functions and have a suitable report design. However, the function is not yet fully defined. If you generate and compile the program now, it would print all the race entries in the RACE ENTRY file after every horse. You need to pass a restrictor parameter from the Print Horses function to the Print Race Entries function so that only the race entries that apply to each horse are printed after each horse's details.

There are two places where you can specify the parameters of the Print Race Entries function.

- From the Edit Functions panel of the RACE ENTRY file by typing **P** against Print Race Entries.
- From the action diagram of the Print Horses function.

In this example, you will use the second method. Go to the Action Diagram Editor for the Print Horses function. To do so, type **F** next to the Print Horses function on the Edit Functions panel.

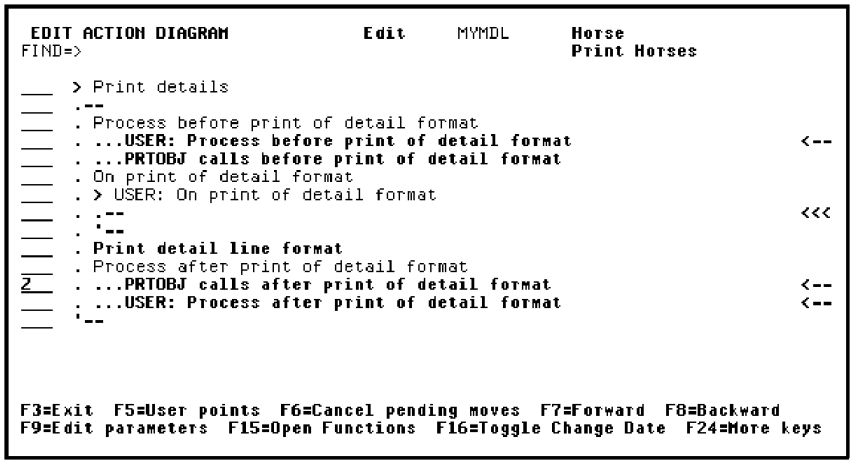


Press Enter.

Editing the Action Diagram

Display the user points of the action diagram by pressing F5. To specify the parameters of the Print Race Entries function, locate the point in the action diagram at which the function is called. Since the function was attached to the Detail format, you will zoom into the Print details user point.

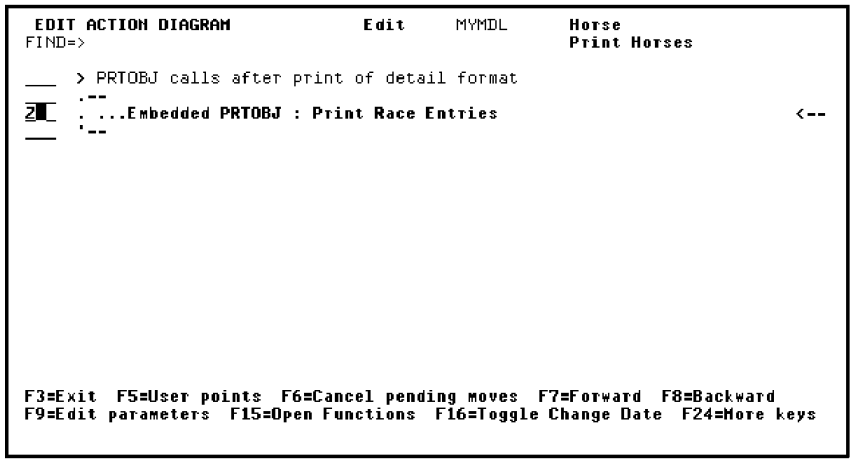
Type **Z** against the Print details user point.



Press Enter.

The Print Details Construct

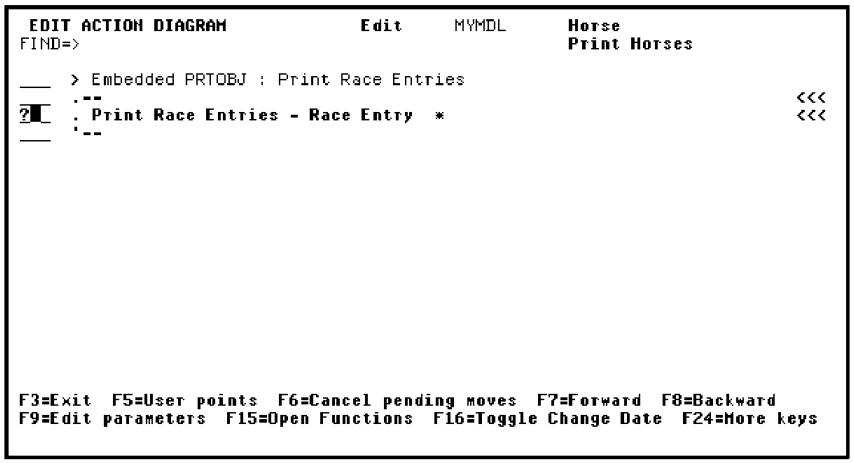
You embedded the PRTOBJ function after the detail line format. To reach the equivalent point in the action diagram, type **Z** against the hidden construct, ...PRTOBJ calls after print of detail format.



Press Enter.

Editing the Print Race Entries Function

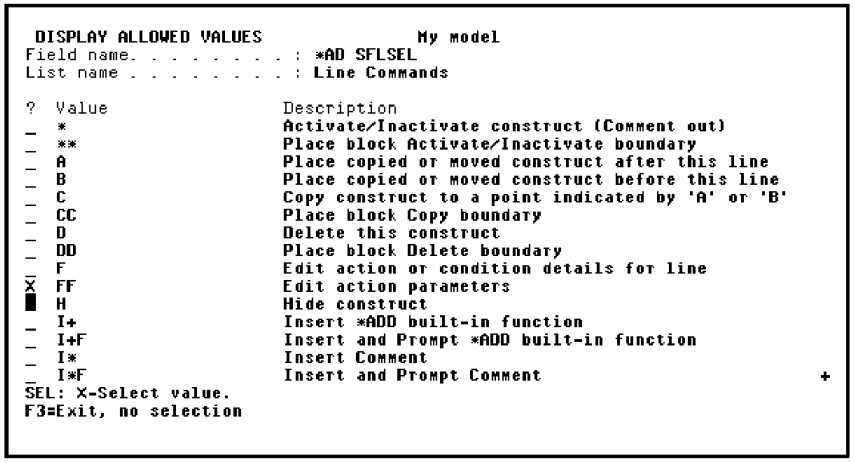
You have now reached the point in the action diagram where the PRTOBJ is embedded. Type **Z** against the hidden construct, ...Embedded PRTOBJ : Print Race Entries.



Press Enter.

The Action Diagram Editor Subfile Selector Values

You can type **?** in the Subfile selector of the Action Diagram Editor to display a list of allowed values. Type **?**.

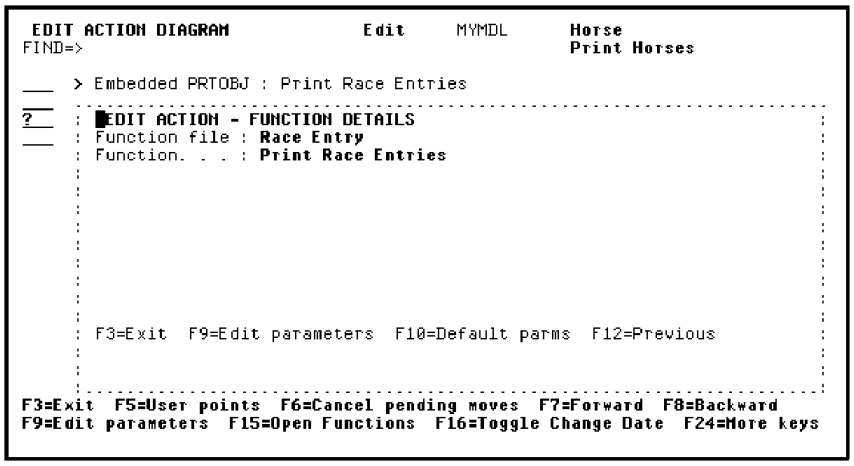


Press Enter.

Editing the Function

Since you are in the process of specifying a restrictor parameter, you should select FF, Edit action parameters.

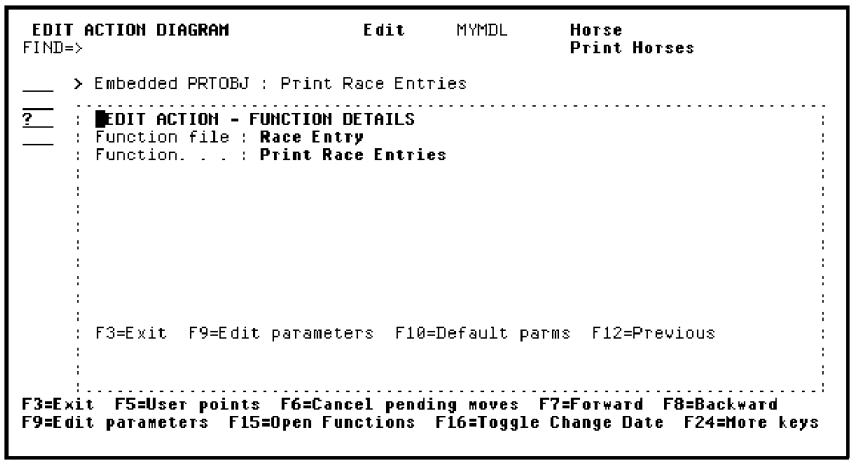
Type **X** as shown to select the FF value.



Press Enter.

Displaying the Function Parameters

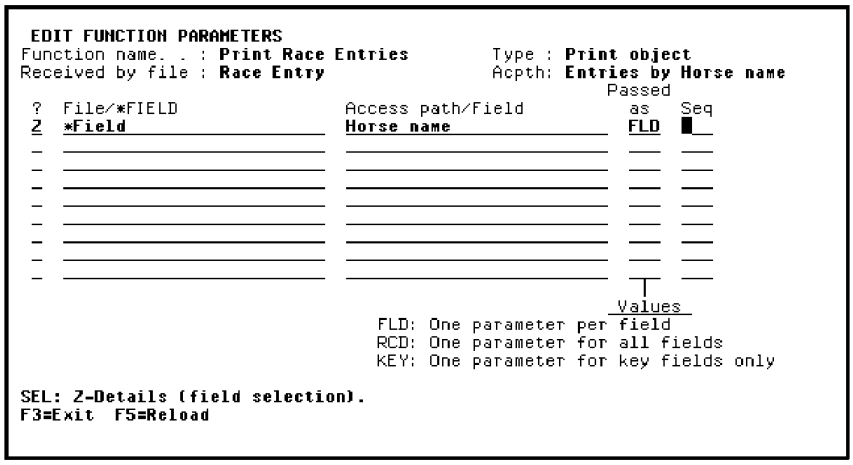
The Edit Action - Function Details window is now displayed.



To edit the function parameters, press F9. The Edit Function Parameters panel lets you edit parameters for the Print Race Entries function.

Editing the Function Parameters

To define the Horse name field as a restrictor parameter, type **Z** in the Subfile selector to specify additional details, and type ***Field**, **Horse name**, and **FLD**.

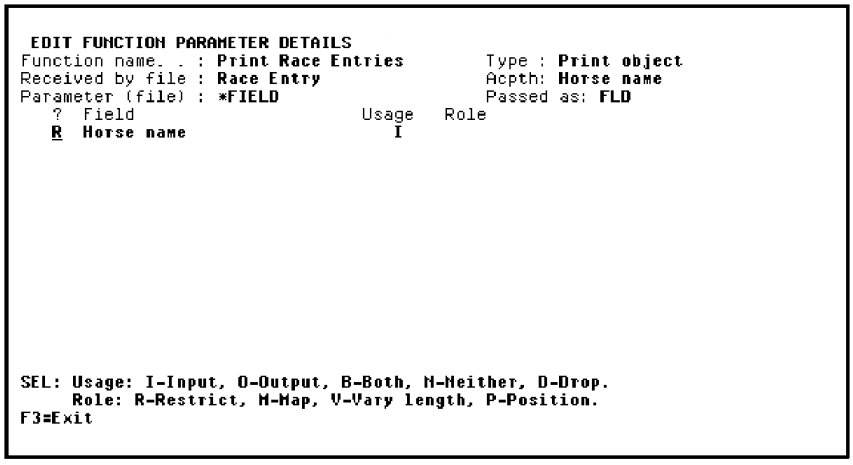


Press Enter.

Specifying a Restrictor Parameter

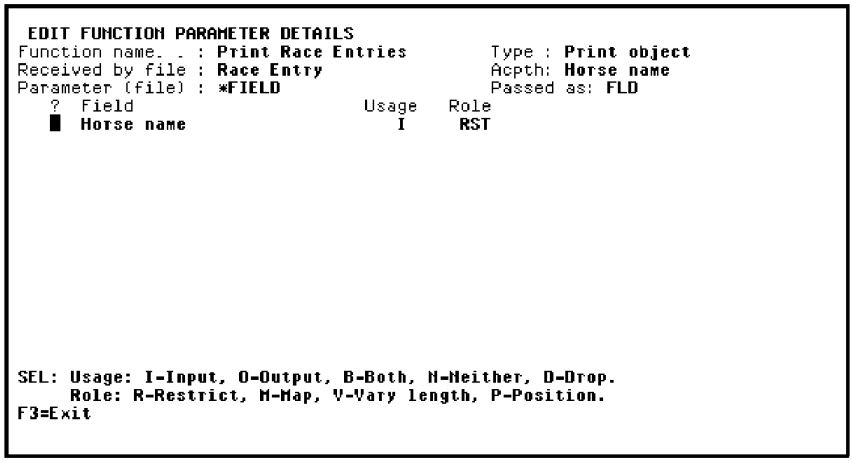
On the Edit Functions Parameter Details panel you will specify the usage and role of the Horse name parameter. In this case, the parameter defaults to Input (I). You will specify that the parameter has a restrictor role by typing **R** in the Subfile selector. This will cause the Print Race Entries program to process only those records on the RACE ENTRY file for the horse identified by the Horse name parameter.

Type **R** against the Horse name parameter.



Press Enter.

CA 2E confirms that the parameter is a restrictor parameter by showing RST in the Role column.

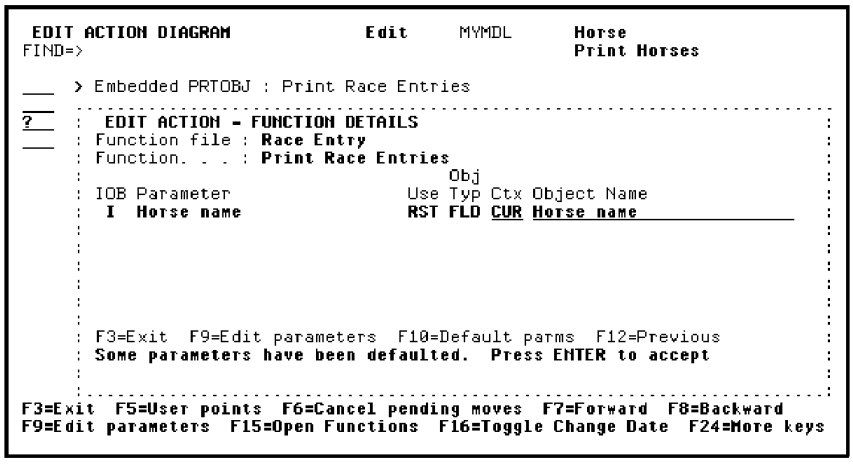


Press F3 twice to exit and return to the Edit Action - Function Details window.

Confirming Parameter Details

The final step is to confirm the values for the parameters you specified. Note that the default context for the Horse name parameter is CUR (current). Since the Detail line is being processed at this point in the action diagram, the CUR context corresponds to the Detail line format.

Note: If you had used the Edit Functions panel rather than the Action Diagram Editor to define parameters, you would still need to access the Action Diagram Editor to confirm the parameter values.



Press Enter to accept the parameter details.

Exit Action Diagram

Exit the action diagram and save the design you have just completed. To exit the action diagram quickly, press F13.

Generating and Compiling the Functions

You have now finished defining the Print Horses function. Accept the default values on the Exit Function Definition panel and change the N on the Submit generation option to **Y**. This will save the function and add it to the joblist for generation. Then select the Submit model create request (YSBMMDLCRT) option from the Display Services menu (F17) to submit the generation and compilation for the function and the Query access path.

Testing the Program

Once the program has compiled, call it from any command line. Various messages relating to the execution of the Query access path (via the i OS Open Query File (OPNQRYF)) command are displayed while the report is produced. A sample report listing data entered with the Edit Horse and Edit Race and Entries programs follows.

Your company name here Print Horses

Horse	M/F	Value	D of B
Bonfire	M	5000.00	2/01/88
		Dam:	
		Sire:	

Race Entries for Bonfire

Course	Date	Time	Entry number	Jockey	Finishing position	H'cap
CAMDEN	5/01/95	1:11:00	2	M Brown	3	0
GOLDEN	6/12/95	1:30:00	3	M Brown	3	0

Faithful Dobbin	F	3500.00	5/31/86
		Dam:	
		Sire:	

Race Entries for Faithful Dobbin

Course	Date	Time	Entry number	Jockey	Finishing position	H'cap
CAMDEN	5/01/95	1:05:00	1	A Miles	2	0
GOLDEN	6/12/95	1:00:00	2	M Brown	1	0

Pegasus	M	3000.00	6/23/92
		Dam: Faithful Dobbin	5/31/86
		Sire: Bonfire	2/01/88

Race Entries for Pegasus

Course	Date	Time	Entry number	Jockey	Finishing position	H'cap
CAMDEN	5/01/95	1:00:00	3	M Brown	1	0
GOLDEN	6/12/95	1:02:00	1	M Brown	2	0

Final totals

Total number of horses :	3
Total horse value . . . :	11500.00

* * END OF REPORT * *

Glossary

access path

A view of the data in a physical file, in a given key sequence, implemented as an i OS physical or logical file.

access path format entry

A process that establishes which fields are present in the access path, which of those fields are key fields for the access path, and the order of those key fields.

access path relation

The set or subset of a file's relations that apply to a particular access path.

access path types

Types which differentiate the functionality of the various access paths. The types of access paths include Physical, Update, Retrieval, Resequence, Span, and Query.

action bar

A construct that appears at the top of an end user panel. It provides a set of choices which define the actions available to the end user.

action bar choice

An option that allows application users access to the actions available for the panel.

action diagram

A notation for defining and showing procedural processing logic. It is made up of a number of basic logic constructs: Iterative, Sequential, and Conditional. Most functions have an action diagram associated with them.

action diagram editor

A facility that lets you modify the default processing logic automatically supplied for a function by adding action diagram constructs at specified user points.

All Objects Model Object List

A unique list within a model that contains an up-to-date detailed model object description for each model object in the model. A model object's description is updated automatically whenever it changes. It is also referred to as *ALLOBJ.

animation

A process that converts a device design to a Toolkit panel design that you can use to interactively simulate (prototype) a system design. This is useful for demonstrating a design to end users to ensure that the design meets operational requirements before you begin programming.

application

A set of implementation objects, primarily programs and files that together satisfy business needs. For example, a payroll application or an order entry application.

attribute

A property of an entity. It is any detail that qualifies, identifies, classifies, quantifies, or expresses the state of an entity. Attributes are usually implemented as fields.

based-on file

The file to which a given object is attached.

built-in function

Type of function which provides commonly required low level operations such as addition and multiplication.

capture file

One of the two types of database files. Capture files generally contain transaction data.

change type

The change type indicates the impact a change to a model object has on other model objects. The change type indicates whether impacted model objects require editing and/or regeneration to ensure the integrity of the model and the application.

component change processing

An automated impact analysis tool that determines how a change to a model object affects other objects in the model and records whether the affected objects need to be edited or regenerated.

condition

A value that specifies the circumstances under which an action or set of actions are to be executed. It also defines the particular value for a field.

conditional construct

A construct that allows steps to be conditionally executed within the action diagram logic and generally conforms to nested IF THEN ELSE logic statements or SELECT sets.

construct

The system of notation in an action diagram to define a boundary for procedural logic.

context

The source of a field within a function. A particular field name can be present several times in a function. The context of the field differentiates between these instances of the field. For example, a field may appear on a panel and also be part of a database file.

cross reference facility

The operations provides for identifying dependencies between objects. This is also referred to as the model object cross reference facility.

current version

This refers only to versions of functions or messages. A version is current when it is the version that is active in the model. In other words, the current version is the one used by and referred to by other model objects and is the one shown on panels that are not specific to processing versions. Only one version in a group of versions can be current at a time.

data model

A representation of the information used by a business. A data model contains groups of data called entities, attributes of the entities, and the relationships between those entities or between an entity and an attribute.

database function

An internal function which reads or updates the database. The database functions are: Change Object (CHGOBJ), Retrieve Object (RTVOBJ), Create Object (CRTOBJ), and Delete Object (DLTOBJ).

design model

The library that holds the i OS database files that compose the model. The model library has an associated library, the generation library, which contains the source code generates for the model.

design object

See model object.

designer

The user type that can change any aspect of the design model including data relationships and functional specifications. In general when a designer is using a model, no other user can access that model. A designer also has the authority to lock and unlock objects. Also known as *DSNR.

developer

A term used to designate either a designer (*DSNR) or a programmer (*PGMR) when a distinction between the two user types is not necessary. Designers and programmers are often referred to collectively as "developers."

device design

The layout of a panel or report associated with a function.

device function

A function which has a device design associated with it, such as Edit Record (EDTRCD), Display File (DSPFIL), Edit Transaction (EDTTRN), and Print File (PRTFIL).

domain

A set of possible values an attribute can take, along with its definition of length and data type, such as numeric or alpha.

dynamic selection

An option that specifies that all records are included in the access path regardless of any select/omit criteria. The records are then filtered by the system as they are read by a program.

end user

The person who uses the generated application once it is in production.

entity

An object which is significant or relevant to the business it represents; for example, a company, a person, a product. An entity is a potential file to be created in the data model of your application system.

entity-relationship diagram (ERD)

A graphic chart used to record the entities and entity relationships using boxes, lines, and notation.

expansion

The process of resolving all relations for a file or for all files in a model. For example, synchronizing a model causes the expansion of all relations to rebuild the file entries for all files in the model.

external function

A function which is implemented as a separate HLL program rather than as a subroutine. All functions are either external or internal.

field ()

A description of an item of data. It must have a name and a field type. Field names may be up to 25 characters long. The names of fields must be unique within the design model.

field type

An assigned type which defines the type of data that the field represents; for example, a number or a date.

file ()

A list of relations that describe an entity. can automatically resolve the relations into a list of file entries or fields. File names may be up to 25 characters long. The names of files must be unique within the model.

file entry

A field that arises as a result of resolving relations. An entry indicates the presence of a field on a file. The three types, depending on the type of relation from which the field is resolved, are: key field entry, attribute entry, and virtual field entry.

file type

The description of how the file will be used. Each file must be one of the allowed file types: structure (STR), reference (REF), or capture (CPT).

first normal form (1NF)

The normalization rule that eliminates repeating groups of data from an entity.

foreign key

A file entry that is the result of a relationship between two entities. The relationship is defined by a non-key attribute of one entity that exists as the primary key of the other entity.

function

A template process that operates on a file or field. A function serves as a building block with which to design applications.

function field

A field whose value is not physically stored in the database but is derived from other fields or files. Function fields can be placed on device designs and used in action diagrams.

function options

The optional features of the standard functions.

function parameter

A field whose value is passed into or out of a function when it is executed. Each parameter can have a parameter role and a parameter usage.

function parameter role

The parameter role that specifies how a parameter is to be used in a function into which it is passed. Standard parameter roles include Restrictor, Positioner, Map, and Vary.

function parameter usage

The parameter attribute that specifies whether a function parameter is to be passed to and/or returned from a function. Function parameter usage includes Input, Output, Both, and Neither.

function redirection

A process whereby the using objects of a given function (FUN) or message (MSG) are redirected to use an alternative FUN or MSG. This occurs when you make a version of a function or message current.

function type

One of a group of process templates provides such as EDTFIL.

generalization

The process during data modeling of combining two entities of the same class into one entity and renaming it.

generatable objects;defined generation

The creation of application source code from a function definition and device design.

generation library

The library that holds generated source code, compiled objects, and help text for a model.

group

This term relates to versions of functions or messages. Versions of the same model object form a group of versions.

i OS

The IBM i operating system.

impact analysis

The process of determining the impact of a proposed or actual change to model objects in order to ensure the integrity of a set of changes by identifying and including dependent objects. impact analysis tools include, cross reference commands, interactive panels to determine usages and references, simulation, and automated component change processing.

implementation

The process of setting up your application and moving it into production once you have generated and compiled source objects.

implementation objects;defined implementation object

Objects such as generated source and compiled objects as opposed to model objects. They are also known as traditional or 3GL (third generation language) objects.

internal function

A function which is generated as in-line code within the calling function.

involution

A process that occurs when an entity refers to itself. This is known as an involuted relation. Also called a self referencing relation.

iterative construct

A programming construct that represents repetitive statements or loops that will be implemented as REPEAT WHILE logic within the main action diagram logic.

key

The field(s) used to uniquely identify a record in a file.

list entry

This is another term for model object list entry.

logical file (i OS)

A logical view of the data in the physical file based on key sequence. Access paths can be implemented as logical files.

message function

Function types which allow you to send error, status, or information messages to the invoking program or to send information to other destinations such as submitting a job to batch.

model object

design objects such as, access paths, functions, and fields, as opposed to implementation objects such as, generated source or a compiled object.

model object description

A reference to a model object in the All Objects list (*ALLOBJ). It contains detailed information about the actual model object. A model object's description is automatically updated whenever the model object changes, and as a result, always reflects the current state of the object.

model object list

A list of references to model objects.

model object list entry

A reference to a model object in a named model object list. It reflects the state of the model object at the time the list entry was created or refreshed, and as a result, provides an historical record of the model object.

model profile

A user profile associated with a model where you can define defaults for various processes and file specifications for an interactive session.

model value

A specific value for a model that controls particular features of the interactive use of , code generation, or implementation.

normalization

The process of eliminating data redundancy using specific standards and rules in data modeling.

notepad

A temporary action diagram which can be used to copy constructs between action diagrams.

object ()

A design element of a model; for instance, a field or function. See also model object.

object type

Each object must belong to one of the object types: file, field, condition, message, function, access path, or application area.

open functions

All functions a single user currently has open for editing.

owning model object

Certain object types exist associated with an owning type. For example, functions (FUN) and access paths (ACP) are owned by a file (FIL) object type.

panel

A panel is displayed by a device function. Panels are composed of a header, a footer, and a body.

panel entry

The appearance of a field on a panel device design. Panel entries arise either from the resolution of access path relations, from function parameters, or from explicit specification by the application user.

parameter

See function parameter.

primary key

A chosen attribute or group of attributes assigned to a particular entity to uniquely define it within an entity relationship model.

program

The result of generating and compiling a function.

programmer

A user type who can create, change, and delete any help text, access paths, functions, and arrays, including working with action diagrams and field conditions for database and function fields. A programmer cannot alter the database files or fields. In general, programmers cannot use a model while a designer is using it. Also known as *PGMR.

prototyping

The presentation of a realistic mock up of a system used to verify that the design meets the operational requirements.

redirection

See function redirection.

reference file

One of the two types of database files: Reference files contain static data, in contrast to capture files which contain volatile transactional data.

references

The references for a model object are all objects the model object refers to, or contains. For example, an internal function is a reference of the external function that contains it. In other words, references are the model objects the referring model object requires in order to be complete or to exist.

referential integrity checking

The process that ensures that proper relations between files are maintained. This usually means making certain foreign keys contain valid values in the referenced files. functions generate source code by default to provide referential integrity checking.

relation

A connection between one entity and another or between an entity and an attribute. When specified in , a relation defines the connection between files or between a file and a field in the file.

relation type

A relation must be one of a fixed number of types; the type is indicated by the verb used to represent the relation such as Has or Known by.

restrictor parameter

One of the allowed function parameter roles. If a function has a restrictor parameter, it can only process database records whose keys match the restrictor. Any device design is modified automatically to meet restrictions. Restrictor parameters must be key field entries on the access path to which the function is attached.

second normal form (2NF)

The normalization rule that eliminates attributes that are not dependent on the primary key. In 2NF, non-key attributes are fully functionally dependent on the primary key.

select/omit set

A choice that defines criteria for the inclusion or exclusion of records from the physical file.

sequence construct

The simplest action diagram construct which specifies a list of actions or other constructs that are to be executed sequentially.

session list

A model object list to which all objects you change during a session are logged.

simulation of a change

A process that lets you see the impact of a proposed change to a model object on other objects in your model before you actually make the change.

source file (i OS)

A file specially formatted to contain program source code. It is usually multi-member with each member containing a different set of source.

standard function

A function which specifies entire programs (external functions) or subroutines (internal functions). There are three main classes of standard functions: device functions, database functions, and user functions.

static selection

An option that specifies that only those records that satisfy select/omit criteria are included in the access path.

subfile

The part of a panel that contains a repeating detail format such as that needed for a list longer than one page.

subfile selector

A field on the left hand side of a subfile used for specifying an action to be taken against a particular subfile record or records. For device designs there will normally be a subfile selection value explanation text field to explain the allowed values for the subfile selector.

synchronize a model

The expansion of relations to rebuild the file entries for all files in the model.

third normal form (3NF)

The normalization rule that eliminates non-key dependencies between attributes of an entity. In 3NF, non-key attributes are mutually independent.

unique key

A file key that ensures each record in the file is unique.

usages

The usages for a model object are all objects that use the model object. They are also called using objects.

user

A user type that is limited to viewing the model and cannot change it. Also known as *USER.

user function

A function that is entirely user specified.

user point

A point in an action diagram at which you can insert logic. A user point is indicated by an arrow on the far right in the action diagram editor.

version

A model object that originated as a copy of either a function (FUN) or a message (MSG). Versions of the same originating model object are linked together to form a group of objects.

virtual field

A field that is logically, but not physically, present on an access path. Virtual fields are the result of specifying that fields in a related file are to be included on an access path.

virtual field entry

An entry which describes the presence of a field on a file or access path as a result of a join with another file. Virtual fields can only be specified for the Owned by, Refers to, and Extended by relations.

window

A panel on which information appears that is not necessarily a full panel. Windows can be varying sizes and can be positioned in varying locations.

Index

A

Access Path Details • 96, 101
Access Paths • 89
Access Paths • 88
Access Paths and Functions • 223
Accessing Session List • 290
Action • 151, 218
Action Bar Navigation • 258
Action Diagram • 160
Action Function Name Panel • 179
Adding • 61, 62, 70, 73, 78, 115, 159, 172, 176, 338, 384, 399
Animate Functions Panel • 248
Application • 36, 245
Attributes for Each Entity • 38

B

Batch Generation • 343

C

CA 2E • 23
Change Type • 325
Changing • 105, 148
Check Condition • 65
Completing Requests • 229
Condition • 162
Condition Values • 235
Confirm Prompts • 211
Creating • 115, 333
Cross References • 313

D

Data Model • 35
Database Relations • 47
Declaring More Files • 56
Default • 89, 112, 119, 382
Default Device Design • 122
Defining • 31, 50, 183, 208, 376
Deleting a Model List Entry • 281
Details of Extended Relations • 73
Development Life Cycle • 22
Device Design Formats • 119
Device Designs • 117
Diagrams • 151
Display Functions for Selection • 227
Displaying • 67, 116
Documenting Relations • 58

E

Edit • 47, 126, 179, 331
Editing • 122, 127, 160, 268, 277, 295
Editing Message Text • 298
Embedding Print Race Entries • 407
Entering • 48, 83
Entities • 36
Entity Unique Identifier • 39
Exiting the Action Diagram • 169
Extending Relations • 68, 72
External Functions • 112

F

Field Condition Detail • 62
Field Conditions • 61
Field Details and Conditions • 58
Fields • 349
File Attributes • 57
File Function • 375
Function • 199
Function • 189
Function • 204
Function • 288
Function • 349
Function Details • 116, 175
Function Fields • 384
Function Keys • 135
Function Parameters • 206
Function Text • 186
Functions • 112, 115, 171
Functions Introduction • 109, 255, 271

G

Generating and Creating Objects • 230

H

Hidden Fields in Subfile Record Format • 145

I

Impact Analysis • 324

Inserting • 162, 163
Interactive Device Design • 246
Introduction • 19
Invoking the Device Design • 143

J

Job List Commands • 303

L

Linking Functions • 192, 194
List of References • 322
Lists • 266

M

Maintenance • 245
Message • 171, 175, 186
Model • 266, 313, 316, 322
Model Objects • 277
Modifying an Action • 218
More Relations • 70

N

Naming • 102, 199, 294
New Case Construct • 195
New Field Name • 83
New Function • 295
New Function • 209
New Functions • 115
New Message Function • 172
Normalize Entities • 41

O

Object • 266, 313
Object Attributes • 52
Object Function • 397
Objects • 50
Open Functions Panel • 252
Options • 189

P

Panel Format • 127
Parameter Details • 208
Parameters • 204
Path Access Formats • 335
Positioning Model Object List • 271
Print • 375, 397
Print Function • 376
Print Race Entries • 399

R

Reassigning Command Keys • 256
Reduce Width of Device Design Layout • 124
Referenced Path • 105
Relation Statements • 48
Relations Panel • 221
Reload Subfile • 214
Removing Field Labels • 131
Renaming Fields • 82
Report Layout • 382
Requirements • 31
Resynchronizing Design Model • 238
Retrieval Access Path • 92

S

Saving Modified Panels • 220
Screen Format Details • 126
Second Action • 183
Select/Omit Set • 102
Selected Relations • 67
Sequence of Relations • 71
Session List • 268
Shortening Field Labels • 129
Single Action • 163
Span Access Path • 333
Specifying • 52, 65, 77, 206
Static and Dynamic Selections • 97
Submitting Function for Generation • 372
Substitution Variable • 299

T

Testing Compiled Program • 238
Text • 218
Text to the First Message • 176
Transaction • 331

U

Understanding Function Details • 116
Usages Panel • 316
Using • 23, 135

V

Validation of Relation Statements • 49
Validation Procedure • 159
Versioning • 288

Versions • 294
Viewing a Subset • 274
Virtual Field Entries • 79
Virtual Fields • 77, 78, 338
Virtual Fields • 76

W

Window Dimensions • 148
Windows Device Design • 142