R S I

ORACLE

INTERACTIVE APPLICATION FACILITY

USER'S GUIDE

Oracle Users Guide - Version 2.3

Copyright (c) April 1981
By Relational Software Incorporated
All rights reserved. Printed in U.S.A.

I N T E R A C T I V E A P P L I C A T I O N
F A C I L I T Y

TERMINAL OPERATOR USER'S GUIDE

TABLE OF CONTENTS

**tbs - means "to be supplied"

I N T R O D U C T I O N

1.0 **Manual Objectives and Use**

This manual describes the operation of an IAF application. It is intended mainly for persons who will be writing IAF applications. Persons reading this should be familiar with SQL. It instructs the terminal operator on the procedures for executing and controlling the application processing.

After reading this manual the user should be able to execute the desired application, cause data to be added, retrieved, updated, and deleted from the database, and control the order or flow of the processing.

E X E C U T I N G   A N   I A F   A P P L I C A T I O N

2.1 **IAP Execution**

An IAF application is executed interactively from the operator's terminal. A CRT or video type terminal is required. IAF is distributed with support for the DEC VT52, VT100, and Perkin Elmer OWL. Terminals compatible with this list may also be used. Other terminal types can be used, but their characteristics must be defined to IAF using the procedure outlined in Appendix A.

IAF applications are specified to, and generated by, the Interactive Application Generator (IAG) utility described in the IAF Application Design Guide. Following the generation process, an application is immediately available for execution.

To execute an application the user must invoke the Interactive Application Processor (IAP) by entering the following command from the terminal:

IAP <applname> [<terminal type>]

<applname>	is the name of the application specified to the IAG at generation time.
<terminal type>	Specify the terminal type identifier. For the IAF supplied types the identifiers are: "VT100", "VT52", "OWL". This parameter is optional, and if omitted the installation defined default type will be used. See Appendix A for more information on terminal types.

If the application database is secured the following message will prompt the operator to supply a valid userid and password:

```
        SECURE DATABASE:  ENTER NAME AND PASSWORD

             NAME: scott
        PASSWORD : xxxxx
```

The entered password will be displayed as x's on the screen. If valid, application processing will continue.

In addition to database access, the operator must have been granted privileges to read and modify the database tables referenced within the application. Access privileges to specific tables and columns are verified during application execution, and violations reported at that time.

2.2 **Definition of Terms**

This section will define some terms which will be used in describing the operations of an IAF application. It is assumed that the reader understands basic computer concepts and the operation of the terminal device. An understanding of the ORACLE Data Base Management System, and the SQL data language is essential.

APPLICATION IAF applications allow an operator to maintain his database interactively from a video terminal. In addition to storing and retrieving data, the application helps the operator by checking the input, and indicating when an error has been made. Requiring the operator to enter correct input prevents invalid data from being stored.

BLOCK A block in IAF corresponds closely to a view or a table in SQL. One block corresponds to one SQL table. Blocks are made up of one to twenty records and can hold a corresponding number of rows from its associated table. Operators enter data into records of a block and may request IAF to insert, update, delete or query records associated with that block.

```
+------------------------------------------------------------------------------+
|                                                                              |
|          E M P L O Y E E          I N F O R M A T I O N                     |
|                                                                              |
|                     A P P L I C A T I O N                                   |
|                                                                              |
+------------------------------------------------------------------------------+
|                                                                              |
|      EMPNO: 5798          DEPTNO: 40          ENAME: JONES__________        |
|                                                                              |
|      JOB: SALESMAN_______       SAL: 2750____         COMM: 355.25_         |
|                                                                              |
|                                                                              |
|                                                                              |
+------------------------------------------------------------------------------+
```

Block 'EMP1' - Single Record Area

```
+------------------------------------------------------------------------------+
|                                                                              |
|          E M P L O Y E E          I N F O R M A T I O N                     |
|                                                                              |
|                     A P P L I C A T I O N                                   |
|                                                                              |
+------------------------------------------------------------------------------+
|                                                                              |
| EMPNO    ENAME              JOB                SAL        COMM       DEPTNO  |
| -----    -------------      -------------      -------    -------    ------  |
|                                                                              |
| 5798     JONES________      SALESMAN_____      2750___    355.25_    40      |
| 5840     SMITH________      ANALYST______      1250___    _______    10      |
| 5932     JACKSON______      CLERK________      950____    _______    20      |
| ____     _____________      _____________      _______    _______    __      |
| ____     _____________      _____________      _______    _______    __      |
| ____     _____________      _____________      _______    _______    __      |
| ____     _____________      _____________      _______    _______    __      |
| ____     _____________      _____________      _______    _______    __      |
| ____     _____________      _____________      _______    _______    __      |
|                                                                              |
+------------------------------------------------------------------------------+
```

Block 'EMP2' - Multiple Record Areas

Figure 2.1

The top diagram in Figure 2.1 shows a block for entering employee information. The information displayed is from Jones' row in the employee table. A block provides the operator with a window to view one or more rows in a single database table.

RECORD - A record corresponds closely to a row of a table or view. A record is composed of fields. Rows are identified on the status line of the display as being stored records or not. A stored record is one that exists in the database. Records that are stored may be updated or deleted. Records that have data entered into them but not yet stored may be subsequently inserted or have queries executed against them. A query executed against a record will attempt to retrieve records with columns that match the data entered in that record.

CURRENT RECORD - Although the operator may view multiple records, only one record may be processed at a time. When adding new records, the data items for one record are entered stored prior to beginning the next one. The current record area is indicated by the location of the cursor on the screen.

FIELD - A field is a data item on the screen. They are identified by labels which are displayed either above or to the left of the data entry area. Reverse video or underlining characters identify a field's data entry and display area. A field may correspond to a column in the table or contain data to aid the operator. In Figure 2.1 the field "EMPNO" corresponds with the column EMPNO in the EMP table.

Fields which are not columns of the block's table are generally used to provide entry aids for the operator. For instance, when entering a part number, it is helpful to display the associated part name for verification. SQL SELECT statements may be associated with any field in a block that are executed when a value is entered into it. These SELECTs may then load values into other fields of the block which are then displayed.

There are other attributes associated with each field that control how a field may be used, what kinds of values it may accept, etc. More information on SELECTs and attributes may be found in the Application Design Guide.

2.2.1 IAF Screen Format

Figure 2.2 is a layout of an IAF screen. All screen pages have the same format. Lines 1 through 22 contain application fields and descriptive text. Lines 23 and 24 are reserved for use by the IAP. Line 23 is the "System Message Area" where status, help, and error messages are displayed. Line 24 contains status indicators and is also used as an operator entry area for the "QUERY" and "PRINT" functions.

The following is a description of each of the indicators appearing on the status line:

"Page x" is the number of the currently displayed application page. "Char Mode:" indicates whether an entered character will either 'Replace' the character in the cursor location, or be inserted('Insert') to its left. "Mode :" indicates that the operator may either query and update existing records (QUERY/UPDATE), or insert new records (INSERT). "Stored :" tells the operator whether the data displayed within the 'current' record area is stored in the database. Following a "QUERY" request, the number of records which the operator has already viewed is accumulated in the "Count:" indicator. The status areas are discussed in more detail in the section on "Keyboard Functions".

2.3 Keyboard Functions

As data is entered and retrieved, the operator controls the application processing through a set of predefined keyboard functions. A summary of these functions is provided in Figure 2.3. Each function is invoked by depressing one or more keys from the keyboard of the CRT. Since terminals vary greatly in number of function keys and layout, each terminal will have a unique sequence for invoking a particular function. However, each terminal can display what function keys do what by depressing the "escape" key, followed by the 'k' (lowercase) key. This key layout can subsequently be printed out by the operator with the PRINT function described later. Additionally, the function keys may be customized by the user by modifying the crt definition IAF uses. This is described in Appendix A.

```
   +--------------------------------------------------------------------------+
 1 |                                  /\                                      |
 2 |                                 /  \                                     |
 3 |                                  ||                                      |
 4 |                                  ||                                      |
 5 |                                  ||                                      |
 6 |                                  ||                                      |
 7 |       A P P L I C A T I O N      ||                                      |
 8 |                                  ||                                      |
 9 |                                  ||                                      |
10 |                                  ||                                      |
11 |                                  ||                                      |
12 |<---------------------------------++------------------------------------->|
13 |<---------------------------------++------------------------------------->|
14 |                                  ||                                      |
15 |                                  ||                                      |
16 |                                  ||                                      |
17 |                                  ||                                      |
18 |                                  ||   D I S P L A Y   A R E A            |
19 |                                  ||                                      |
20 |                                  ||                                      |
2' |                                 \  /                                     |
   |                                  \/                                      |
23 |   ___________________" System Message Area"_____________________________  |
24 |  Char Mode: Replace   Page: 1   Mode : Insert   Stored : NO  Count :*0   |
   +===============/\===========/\===========/\==============/\=========/\=+
                   ||    "Page" ||           ||              ||         ||
 "Character Mode"  ||   "Number"             ||     "Stored" ||         ||
     Indicator                    "Processing"||    "Record" ||         ||
                                      "Mode"  ||    "Indicator"         ||
                                   "Indicator"                          ||
                                                           "Retrieved"||
                                                        "Record   Count"
```

IAF Screen Layout

Figure 2.2.

Next Field	Perform edit checks on field before advancing to next field in record
Previous Field	Advance cursor to previous field in block.
Clear Field	Clear the contents of the current field from cursor to end of field
Next Record	After Query: Move cursor to next record. If at last record, retrieve next record and display. After Insert: Move cursor to next area and initialize for data entry.
Previous Record	Move cursor up one record area on page
Next Set	After Query Only: Display next set of selected records, one in each record area.
Next Block	Terminate processing of current block and begin processing of next block
Previous Block	Terminate processing of current block and begin processing of previous block
Clear Block	Clear all the data in all record areas of block. Insert Mode: Initialize default values and move cursor to first record area. Query Mode: Release selected records and move cursor to first record area.
Clear Form	Clear data in all application blocks. Processing resumes in first block
Insert	Store current record into database. Must be in "Insert" processing mode
Query	Request selection of specified records. Must be in Query/Update processing mode

" IAP Keyboard Functions "

Figure 2.3 - Part 1 of 2

Update	Request that the changed data fields in the current record be permanently recorded in the database. Must be in Query/Update processing mode
Delete	Request that the current record be deleted from the database. Must be in Query/Update processing mode
Change Processing Mode	Change from one processing mode (Insert or Query/Update) to the other.
Change Character Mode	Change from one character mode (Replace or Insert) to the other.
Move Cursor Left	Move cursor left one position within current field.
Move Cursor Right	Move cursor right one position within current field.
Help	Display help message for current field
Display Attributes	Display the attributes for the current field.
Redisplay	Redisplay current screen page after communications or terminal failure.
Exit	Terminate IAP application processing
Print Form	Request printing of 1) Current page 2) Entire Form 3) Function key layout

" IAP Keyboard Functions "

Figure 2.3 - Part 2 of 2

```
    +---------+---------+    +----------+----------+----------+----------+
    |  MOVE   |  MOVE   |    |          |          |          |          |
    | CURSOR  | CURSOR  |    | QUERY    | INSERT   | UPDATE   |  EXIT    |
    |  LEFT   |  RIGHT  |    |  (PF1)   |  (PF2)   |  (PF3)   |  (PF4)   |
    |  (<-)   |  (->)   |    |          |          |          |          |
    +---------+---------+    +----------+----------+----------+----------+
                             |          |          |          |          |
                             |  NEXT    | PREVIOUS| CLEAR    | REDSPLY  |
                             | FIELD    |  FIELD   | FIELD    |          |
            +----------+     |   (7)    |   (8)    |  (9)     |   (-)    |
            |PREVIOUS  |     +----------+----------+----------+----------+
            | FIELD    |     |          |          |          |          |
            | (BACK    |     |  NEXT    |PREVIOUS  | CLEAR    | DISPLAY  |
            |   SPACE) |     | RECORD   | RECORD   | RECORD   |  ATTR    |
            +----------+     |   (4)    |   (5)    |   (6)    |   (,)    |
                +----------+ +----------+----------+----------+----------+
                |CHARACTER | |          |          |          |     P    |
                | DELETE   | |  NEXT    |PREVIOUS  | CLEAR    |  C  R    |
                |          | | BLOCK    |  BLOCK   | BLOCK    |  H  O  M |
                | (DELETE) | |   (1)    |   (2)    |   (3)    |  A  C  O |
                +----------+ +----------+----------+----------+  N  E  D |
                                                   | CHANGE   |  G  S  E |
+----------+    +----------+ |                     |CHAR MODE |  E  S    |
|  NEXT    |    |          | |        HELP         |          |          |
| FIELD    |    |          | |                     |          |          |
| (TAB)    |    |          | |        (0)          |   (.)    | (Enter)  |
+----------+  +-+          | +---------------------+----------+----------+
              | NEXT FIELD |
              |            |
              |  (RETURN)  |
              +------------+
```

FUNCTIONS	KEYSTROKES
EXIT	CTRL - Z
DELETE	CTRL - D
CLEAR FORM	ESC - C
DISPLAY FUNCTION KEYS	ESC - K
NEXT SET	ESC - S
PRINT FORM	ESC - P

IAF Function Key Layout - VT-100

Figure 2.4

2.3.1 Processing Modes and Actions

An IAF application may be processed in either of two modes: "INSERT" or "QUERY/UPDATE". By selecting a processing mode the operator states their intention to either insert new records or retrieve existing records with the possibility of making changes. Therefore, each mode places restrictions on the functions which the operator may perform. When initially executed, the application is in "INSERT" mode.

CHANGE MODE Function:
An application may be switched from one mode to the other by depressing the 'CHANGE MODE' key. The status indicator "Mode:" on the bottom line will alternate between "INSERT" and "QUERY/UPDATE" as the CHANGE MODE key is depressed.

In an empty block, changing from QUERY mode to INSERT/UPDATE mode will cause the first record to be initialized with default INSERT values. Changing back to QUERY mode will clear the default values.

The following functions are available only in the mode indicated.

Insert Mode

INSERT Function:
This action will cause the contents of the current record area to be inserted into the database table. The application must be in 'INSERT' mode. If the current record is already stored, the request will be rejected. When a record has been sucessfully inserted the "Stored: " indicator will change form 'NO' to 'YES'.

Query/Update Mode

QUERY Function:
This action will retrieve those records in the database table who satisfy the defined criteria. There are two ways to state these conditions. Prior to selecting the QUERY function the operator may enter data into any field. When the query is executed only those records that have a corresponding column value equal to the entered field values will be returned.

For instance, after entering data into this sample screen

```
          DEPTNO: 30      JOB: SALESMAN
```

the following 'where' clause would be generated in the query:

```
        WHERE  deptno=30 AND job='SALESMAN'
```

and only those rows where the 'deptno' column is equal to 30 and 'job' column is equal 'SALESMAN' would be returned.

The second way a user may conditionally retrieve rows is by entering a specific consdition to be met. After the QUERY key has been entered the operator will be prompted on line 24 of the screen for the additional text. Expanding on the above example, the operator may only want to see those salesman in department 30 with a salary greater than $2000. The following response:

```
     QUERY WHERE?: sal > 2000
```

will cause the 'Where' clause:

```
     WHERE deptno=30 AND job='SALESMAN' AND sal>2000
```

to be generated. Only those records which satisfies all these conditions will be returned. Essentially, any condition that could be specified in SQL directly could be specified in IAF providing it may be stated on one line. The user however must be aware of what table is being queried and know the names of the columns in that table. Refer to the SQL language User's Guide for additional details on constructing a 'Where' clause.

Following the entry of all conditions, IAF will display rows returned by the query in each of the blocks records. The 'Count :' indicator will reflect the number of records displayed, and a YES in the stored record indicator (Stored :) indicates the current record is in the database. If the query results in more rows than can be initially displayed, the 'NEXT RECORD' and 'NEXT SET' functions are used to view the remaining records.

Each time a new query is executed the results of the previous query are lost.

UPDATE Function:
Following a QUERY request, the retrieved records may be updated. One record is updated at a time. The operator moves the cursor within the 'current' record to the fields to be changed. Only fields which were designated as 'updatable' may be modified. When all the desired fields have been entered, the UPDATE function is invoked to have the modified data written to the database.

DELETE Function:
The DELETE function deletes the 'current' record from the database. The deleted record is scrolled of the screen, and each remaining record is scrolled up on position. The last record will contain the next record returned by the query or will be blank.

2.3.2 Processing in Either Mode: Field Control

NEXT FIELD:
This function will advance the cursor to the next field in the current record area. If data has been entered into the present field it will trigger the field's edit checking. If an error is detected, an error message is displayed, and the cursor position remains unchanged. The error must be corrected before the cursor can be advanced.

If the cursor is positioned to the last field of the record, NEXT FIELD will advance the cursor back to the first field of the current record area. If the next field of a record is defined on a different screen page, the new page is displayed and the cursor positioned to the appropriate field.

A field may have been defined to use the auto skip feature. In this case if data is entered into every character position, the cursor will automatically skip to the next field. This implied NEXT FIELD will also trigger field editing.

Fields which can not be entered by the user will be skipped over.

PREVIOUS FIELD:
This function is similar to NEXT FIELD except the cursor advances in a backward direction to the previous field. If the cursor is positioned to the first field in the record area, PREVIOUS FIELD will move the cursor to the last field in the current record area. No edit chacking takes place when this function is executed.

If the previous field is on another screen page, the new page is displayed, and the cursor positioned to the appropriate field.

CLEAR FIELD: This function will erase the contents of the current field from the current cursor position to the end of the field.

2.3.2.1 Character Control

CHANGE CHAR MODE:
Two modes of character entry are supported. In 'Replace' mode, each entered character replaces the character previously displayed at the cursor location, and the cursor is moved one space to the right. In 'Insert' mode, entered characters are inserted at the cursor location, with all the characters from the cursor position to the end of the field shifted to the right one position.

The CHANGE CHAR MODE function will change from one character mode to the other. The current character mode is displayed in the 'Char Mode:' indicator on line 24.

MOVE CURSOR LEFT:
This function moves the cursor to the left one position.

MOVE CURSOR RIGHT:
This function moves the cursor to the right one position.

DELETE CHARACTER:
This function will delete the character pointed to by the cursor. Remaining characters in the field are shifted one space to the left, overlaying the deleted character.

2.3.3 Record Control

NEXT RECORD:

Insert Mode:
For a single record block, NEXT RECORD will erase the data area and reinitialize the area with the default values.

A multi-record block will advance the cursor into the next record area, and initialize the fields with their default values. If positioned to the last record area, all the areas will scroll upward one record area, with the top area disappearing from view. The cursor will remain in the bottom area which will be cleared and initialized with the default field values.

Query/Update Mode:
For a single record area block, NEXT RECORD will display the next record retrieved.

For multi-record blocks, the cursor will advance to the next record area. If positioned to the last record area, all areas will scroll upward, and the next retrieved record is displayed at the bottom. The top record will disappear from view.

As each new record is displayed the 'Count:' indicator will be incremented by 1.

NEXT SET OF RECORDS
This function allows the operator to display the next 'n' records, where 'n' is the number of records within the block. The "Count:' indicator is incremented by the number of new records displayed. This function is used in QUERY/UPDATE mode only.

Referring to the sample blocks in Figure 2.2, this function will display the next record in block 'EMP1', whereas the next nine are displayed in 'EMP2'.

For multi-record blocks, PREVIOUS RECORD will move the cursor up one record area. If positioned to the top area, 'At Top of Block' message is returned and the cursor is unchanged.

Once an inserted or retrieved record has scrolled off the top of the viewing area it can not be viewed again without issuing a new query.

Clear Record:
Clear the content of all fields in the current record area. For INSERT mode, the fields are initialized with their default values. If the current record area contains a database record (indicated by Stored : YES) it can not be cleared. CLEAR BLOCK or CLEAR FORM must be used to clear the record area.

2.3.4 Block Control

IAF applications can contain one or more **blocks.** When the application is executed, processing begins with the first defined block.

NEXT BLOCK:
This function will terminate the processing of the current block, causing the first screen page of the next block is displayed. The data within the current block is retained, and available if the block is reprocessed. The cursor is positioned to the first field within the first record area. Blocks are accessed in the order in which they were defined to the IAG. The NEXT BLOCK function in the last block will take the user back to the first block.

PREVIOUS BLOCK:
This function will terminate the processing of the current block and cause the first screen page of the previous block to be displayed The cursor is positioned to the first field within the first record area. Blocks are accessed in the reverse order of definition. If positioned to the first block, PREVIOUS BLOCK will take the user to the last block.

CLEAR BLOCK:
This function will clear the field contents in the current block. All record areas will be cleared. If processing a QUERY, the records which have not been viewed are lost.

CLEAR FORM:
This function clears an entire form or application. All the data within each block is cleared, and processing continues in the first block. The application is in the same state as when initially executed.

2.3.5 Help Functions

HELP Message:
This function will display a help message for the current field. The message is displayed on line 23 in the System Message Area. The content of the message is entered at application generation time.

DISPLAY ATTRIBUTES:
This function will display the attributes of the current field in the message area at the bottom of the screen.

SHOW FUNCTION KEYS
This function will display the function key assignment for the current terminal type.

2.3.5 Control Functions:

REDISPLAY:
In the case of a communication line error, or terminal failure, this function will cause the last screen to be redisplayed.

QUIT:
This function causes normal termination of the IAP.

PRINT:
This function allows the operator to print the current screen page, all the pages of the application, or the function key layout. When PRINT is requested the operator must specify the name of the file where the printed output will be stored. Any valid operating system file name is permitted. Following the file specification the following message will appear on line 24:

```
Select: 0)Abort 1)Current Page 2)Form 3)Funct Key ? _
```

After choosing 1, 2, or 3, the following message will appear:

```
Send File to System Printer ? _
```

A 'Y' response will cause the IAP to issue a system command to have the contents of the created print file printed on the system defined printer.

This page intentionally left blank.

COMMENTS PLEASE

Please assist us in improving this manual and in correcting any documentation errors. Forward your comments to:

Documentation Coordinator
Relational Software, Inc.
3000 Sand Hill Road, Bldg. 3-180
Menlo Park, CA 94025

MANUAL: INTERACTIVE APPLICATION FACILITY - Terminal Operator's Guide

SUBJECT ______________________________

PAGE _____ **INCORRECT** _____ **UNCLEAR** _____ **INCOMPLETE** _____

COMMENTS ______________________________

____

____

____

____

____

____

____

____

Name:

Organization:

Address:

Telephone:

I N T E R A C T I V E - A P P L I C A T I O N F A C I L I T Y

APPLICATION DESIGN GUIDE

TABLE OF CONTENTS

I N T R O D U C T I O N

1.0 Manual Objectives and Use

This manual presents the Interactive Application Facility. It explains the purpose and features of IAF, and describes the procedures for generating an IAF application. The intended audience includes those persons responsible for application design and generation. The level of presentation assumes a working knowledge of the ORACLE Data Base Management System and the SQL Language.

2.0 Structure of the Document

This manual is divided into four sections:

Section 2 - "IAF Overview" - Describes the purpose of IAF and explains the component utilities; Interactive Application Generator (IAG) and Interactive Application Processor(IAP).

Section 3 - "Application Structure and Design" - Presents the features and structure of an IAF application.

Section 4 - "Application Definition - Interactive Application Generator" - Describes the process of defining and generating an application. The generation of the sample "employee" application is discussed.

Section 5 - "Advanced Application Techniques" - Presents some additional techniques for using the features of IAF.

Appendix A - "CRT Interface Utility" - Describes the IAF utility allowing a user to specify a non-standard CRT for use with IAF.

I A F - O V E R V I E W

2.1 Introduction

IAF is an application facility which provides full screen communication with a CRT terminal device, interpretation of operator requests, data validation, and the necessary database operations to store, retrieve and update the requested data. Basic editing functions are supported which verify data types, check ranges, and test for existence within a predefined table of values.

Most CRT terminal types may be used providing they have basic cursor control features and have been previously defined to IAF. As distributed, IAF supports the VT100, VT52, OWL, or a compatible device. Appendix A describes generating new crt definitions.

An application consists of one or more **screen pages.** Figure 2.1 is an example of a single screen page. Each page may contain one or more data input and display areas. A display field may correspond to a column in a database table. In figure 2.1 each display field corresponds to a column in the "EMP" table in the ORACLE sample "PERSONNEL" database. The page may be enhanced with prompts, help text, and format characters to improve readability.

Operators interact with IAF applications via a predefined set of keyboard functions. These functions allow the operator to move from field to field, screen to screen, initiate insert, retrieval and update operations, control the application processing modes, and request help information.

Data is entered a field at a time. After a field is entered, the cursor is automatically moved to the next field area. Edit criteria may be associated with any input field. A field may be tested for data type (character or number), format (ie. date), value range, or existence in a table of acceptable values. If the input field is incorrect, the user is immediately notified, and required to correct the error before the process can continue. In this manner the user is guided through a display until the input process is completed.

```
+------------------------------------------------------------------------------+
|                                                                              |
|                 EMPLOYEE   PERSONNEL   RECORD   FORM                         |
|                                                                              |
|                                                                              |
+------------------------------------------------------------------------------+
|                                                                              |
|        EMPLOYEE  NUMBER : 7956_                                              |
|                                                                              |
|                    NAME : MARTIN__________                                   |
|                                                                              |
|                     JOB : ANALYST_____                                       |
|                                                                              |
|                  SALARY : 4380____                                           |
|                                                                              |
|              COMMISSION : ________                                           |
|                                                                              |
|      DEPARTMENT  NUMBER : 30   NAME : SALES____________                      |
|                                                                              |
+------------------------------------------------------------------------------+
```

Figure 2.1 - 'Sample Application Screen Layout'

When requisite displays fields have been filled, the data may be inserted into the database via keyboard function. Data is stored a row at a time. Once stored, that row is immediately available to other users.

An IAF application may be used to retrieve previously stored data. Upon retrieval, the data is formatted and displayed on the user's terminal. Data is retrieved one row at a time. Once displayed, the user is free to modify any field which has been designated for update. When all modifications are completed, a keyboard function will initiate the updating of the database. Data which is no longer needed may be deleted.

2.2 Component Description

The Interactive Application Facility consists of two utility programs. The **Interactive Application Generator (IAG)** interactively communicates with the designer to define the user's application. The **Interactive Application Processor (IAP)** executes the defined application under the operator's control. Figure 2.2 illustrates the relationship between these utilities.

2.2.1 Interactive Application Generator

The IAG utility is executed from the application designer's terminal. The designer must specify the name of the application being defined. For new applications IAG will begin the question and answer session. For existing applications a previously created reponse file may be used. The questions address the following areas:

- Database to be referenced within the application.
- Specification of associated database tables and columns.
- For each field (column) within a table:
    - Edit criteria to be applied to input data.
    - Initial value assigned to this field.
    - SQL statement to be executed when the field is entered or retrieved.
    - Placement of field on a screen page
- Placement of prompts, explanatory text, and line drawing characters on each screen page.

```
       --------          +-------------------+
     /          \        |    INTERACTIVE    |
    |  Questions |<------->|                  |          +-----------------+
    |     &      |<------->|    APPLICATION    |<-------->|  <applname>.inp |
     \ Answers  /        |                   |<-------->|                 |
       --------          |   GENERATOR (IAG) |          +-----------------+
     ----------          +-------------------+
    /          \                  ||                     "Response File"
    ------------                  ||
                                  ||
"Application Designer"            ||
                                \    /
                                 \/
                         +-------------------+
                         |  <applname>.frm   |   "Application Image File"
                         |                   |
                         +-------------------+
                                  ||
                                  ||
                                  ||
                                  ||
                                \    /
                                 \/
       --------          +-------------------+
     / NAME:___ \        |    INTERACTIVE    |          +-----------------+
    |  ADDR:___  |<------->|   APPLICATION     |<-------->|     USER'S      |
    |     .      |<------->|   PROCESSOR       |<-------->|    DATABASE     |
     \    .     /        |      (IAP)        |          +-----------------+
       --------          +-------------------+
     -----------
    /           \
    -------------

 "Terminal Operator"
```

Application Development Process

Figure 2.2

These questions will be asked repetitively until the application is defined. When the definition is complete, IAG compiles the responses into an internal format and stores the result on disk. The generation process is now complete and the application can be executed.

As the designer responds to each question, the question text and associated response is saved in an input work file. This file may be used as an alternate input source in subsequent sessions with IAG. Hence, by altering a saved answer file and re-executing IAG, existing applications can be easily modified. A standard text editor may be used to modify the reponse file.

2.2.2 **Interactive Application Processor**

An IAF application is executed by invoking the IAP utility. The operator specifies the name of the 'image file' previously created by IAG. If the referenced database is secure, the user will be prompted for a valid user id and password. In addition, a user must be granted privileges to the desired data.

Figure 2.3 illustrates the relationship of IAP to the terminal operator and ORACLE database. On the front end IAP provides the screen handling functions which communicate with a CRT terminal, interpret the operator's request, control the processing flow, and validate, convert, and format the data. The back-end uses the standard facilities of the Host Language Interface to execute SQL statements which communicate with the user's database.

The 'image' file contains a set of tables which describes the application. The information provides the screen formats, individual field descriptions, and overall structure. IAP uses these tables to guide the application processing.

The IAP utility is viewed by ORACLE as any other application program using the SQL interface. As the operator enters data, the appropriate edits are performed and SQL (SELECT, INSERT, UPDATE, or DELETE) statements are executed against the database.

```
                          --------
                        / NAME:___  \
                       |  ADDR:___   |
                       |       .     |
                        \      .    /
 " TERMINAL "             --------
   OPERATOR              ------ ---
                        /          \
                        -------------
                             /\
                            /  \
                             ||
                             ||
                            \  /
                             \/
++==========================================++
||                                          ||
||  SCREEN  HANDLER  -- TERMINAL INTERFACE  ||
||                                          ||
|+-------------------+--------------------+|
||   D A T A         |     APPLICATION     ||
|| EDIT / VALIDATION | FLOW  OF  CONTROL   ||
|+- ----------------+--------------------+|
||                 " S Q L "                ||
||                                          ||
||                 H O S T                  ||
||   L A N G U A G E   I N T E R F A C E    ||
||                                          ||
++==========================================++
||                                          ||
||          "  O   R   A   C   L   E  "     ||
||                                          ||
++==========================================++
                             /\
                            /  \
                             ||
                             ||
                            \  /
                             \/
                  +------------------+
                  |  " U S E R ' S " |
                  |                  |
                  | D A T A B A S E  |
                  |                  |
                  +------------------+
```

Interactive Application Processor - IAP

Figure 2.3

The operator uses the IAP function keypad to control the application flow. The operator is free to choose which screens to process and the order of processing. Repetitive processing of a screen allows the operator to enter or retrieve multiple rows of data. When the operator's tasks are completed, the 'Quit' function will terminate IAP. For additional information on executing IAP refer to the "IAF Terminal Operator's Guide".

A P P L I C A T I O N - D E S I G N

3.1 Introduction

This section will present the basic structure of an IAF application. The objective is to provide the reader with a general understanding of an application's components, providing sufficient information to understand the design and definition of a simple application. The details on defining an application will by deferred to Section 4 : The Interactive Application Generator (IAG)

3.2 Sample Application

Throughout this chapter references will be made to two sample applications. These applications were designed to utilize most of the features of IAF, and will illustrate the concepts being presented.

The first application uses the "EMP", "DEPT", and "PROJ" tables within the ORACLE "PERSONNEL" database. They were chosen because the reader is already familiar with the data in these tables from the SQL Language Examples. A few simple screens will be defined which allow entry and retrieval of data from these tables. Figure 3.1 provides the application's screen layouts which will be referenced in this manual and the "IAF Terminal Operator's Guide". Figure 3.2 shows the columns defined for these database tables.

The second application is an on-line order entry system. It is designed to allow operators to enter orders by storing the information directly into an ORACLE database. Figure 3.3 shows the three screen forms available to the order operator.

```
+----------------------------------------------------------------------------+
|                                                                            |
|     E M P L O Y E E     P E R S O N N E L     R E C O R D                  |
|                                                                            |
+----------------------------------------------------------------------------+
|                                                                            |
|     NUMBER : ____                    SALARY     : _______                  |
|       NAME : _________           COMMISSION     : _______                  |
|        JOB : _________                                                     |
|                   DEPTNO: __   DEPT NAME : __________                      |
|                                                                            |
+============================================================================+
|                                                                            |
|     E M P L O Y E E     P R O J E C T     A S S I G N M E N T S            |
|                                                                            |
+----------------------------------------------------------------------------+
|                   PROJNO         PROJECT   NAME                            |
|                   ___            ______________                           |
|                   ___            ______________                           |
|                   ___            ______________                           |
|                                                                            |
+----------------------------------------------------------------------------+
```

"Employe" Application Screen Layout

Figure 3.1

Database: 'personnel'

Table Name	Column Name	Data Type	Length	Image
emp	empno	number		unique
	ename	char	10	non-un
	job	char	9	non-un
	sal	number		
	comm	number		
	deptno	number		non-un
dept	deptno	number		unique
	dname	char	14	non-un
	loc	char	13	non-un
	empcnt	number		
pe	empno	number		non-un
	projno	number		non-un
proj	projno	number		unique
	pname	char	10	non-un
	budget	number		
	empcnt	number		

"Employee" Application - Database Tables

Figure 3.2

```
+------------------------------------------------------------------------------+
|                                                                              |
|                         ORDER ENTRY APPLICATION                              |
|                                                                              |
|                        " O R D E R   F O R M "                               |
|                                                                              |
+------------------------------------------------------------------------------+
|                                                                              |
|  ORDER NUMBER: _____                                    DATE: ________       |
|                                                                              |
+--------------------------------------+---------------------------------------+
|      CUSTOMER INFORMATION            |        SHIP TO INFORMATION            |
+--------------------------------------+---------------------------------------+
|          CUSTNO:______               |                                       |
|  NAME:_______________                |    NAME:_______________               |
|  ADDR:____________________           |    ADDR:____________________          |
|  CITY:_______________                |    CITY:_______________               |
| STATE:__          ZIP: _____         |   STATE:__       ZIP: _____           |
+--------------------------------------+---------------------------------------+
|        PURCHASER:_______________                                             |
|------------------------------------------------------------------------------+
|Next Form is : Order Item Form        Previous Form is : Order Browse Form    |
+------------------------------------------------------------------------------+


+------------------------------------------------------------------------------+
|                         ORDER ENTRY APPLICATION                              |
|                                                                              |
|                   "  O R D E R   I T E M   F O R M "                         |
|                                                                              |
|------------------------------------------------------------------------------+
|                                                                              |
|   LINENO     PARTNO     DESCRIPTION        PRICE      QTY/UNIT  UNITS        |
|               _____     ______________     ______      ____     ____         |
|   ___     SPECIAL INSTRUCTIONS:  ____________________                        |
|               _____     ______________     ______      ____     ____         |
|   ___     SPECIAL INSTRUCTIONS:  ____________________                        |
|               _____     ______________     ______      ____     ____         |
|   ___     SPECIAL INSTRUCTIONS:  ____________________                        |
|               _____     ______________     ______      ____     ____         |
|   ___     SPECIAL INSTRUCTIONS:  ____________________                        |
|               _____     ______________     ______      ____     ____         |
|   ___     SPECIAL INSTRUCTIONS:  ____________________                        |
|                                                                              |
+------------------------------------------------------------------------------+
|Next Form is : Order Browse Form      Previous Form is : Order Form           |
+------------------------------------------------------------------------------+
```

Figure 3.3 - 'Sample Application - Form Layouts'

```
  ---------------------------------------------------------------------+
                         ORDER ENTRY APPLICATION                        |
 |                                                                      |
 |               " O R D E R   B R O W S E   F O R M "                  |
+---------------------------------------------------------------------+
|     ORDERNO    CUSTNO  CUSTNAME              ORDER DATE   PURCHASER   |
|                                                                      |
|     _____      _____   _______________       ________     ___________ |
|     _____      _____   _______________       ________     ___________ |
|     _____      _____   _______________       ________     ___________ |
|     _____      _____   _______________       ________     ___________ |
|     _____      _____   _______________       ________     ___________ |
|     _____      _____   _______________       ________     ___________ |
|     _____      _____   _______________       ________     ___________ |
|     _____      _____   _______________       ________     ___________ |
+---------------------------------------------------------------------+
| Retrieve Order Summary : Enter field values for desired ORDERS.      |
|                          Select Inquiry function key.                |
+---------------------------------------------------------------------+
| Review entire Order and/or Update Order Information:                 |
|          Move cursor to desired ORDER; Select Previous Block Function |
+---------------------------------------------------------------------+
| Next Form is : Order Form         Previous Form is : Order Item Form |
+---------------------------------------------------------------------+
```

Figure 3.3 - 'Sample Application Form Layouts' (Continued)

To enter an order the operator executes IAP to bring up the "ORDER FORM". This form allows the operator to enter general order information. After this data has been entered and stored, the "Order Item Form" is processed to enter each ordered item. Another form is provided to aid the operator in reponding to customer inquiries. The "Order Browse Form" allows the operator to search the database using one or more search fields to retrieve a list of customer orders. Once located, the operator may use the previous two forms to review and modify the order information.

Figure 3.4 lists the database tables and columns used in this application. The 'order' and 'orderitem' tables are of primary interest, and are used to store the information associated with each order. The other tables are used for reference purposes only. The 'part' table provides a list of valid parts. Later in this section a technique will be presented for restricting part numbers to those in the list. The table also supplies information about the ordered part. In the same manner, the 'state' and 'customer' tables provide a list of valid entries and related information. The state code for the entered city is supplied by the 'citystate' table.

3.3 Application Structure

The next section describes the components of display generation. Section 3.3.1 describes a display block, which is a functional unit of a display. Section 3.3.2 describes field specification; a field is a component of a block. Section 3.3.3 describes screen format capabilities; and Section 3.3.4 discusses terminal support details.

3.3.1 Block Specification

An IAF **application** consists of one or more application **blocks.** Each block consists of a collection of fields which map to a single database table. All tables referenced in an application must reside in the same ORACLE database. An operator may process in only one block at a time.

Within each block, the collection of fields may be displayed multiple times. This allows multiple records (or rows) from the defined table to be displayed simultaneously. However, only one record may be processed at a time.

SAMPLE 'ORDER ENTRY' APPLICATION

D A T A B A S E - T A B L E - D E F I N I T I O N S

Database: 'order'

Table Name	Column Name	Data Type	Length	Image	Nulls
order	orderno	number		unique	no
	date	number			no
	custno	number		non-uniq	no
	purchaser	char	15		no
	shipname	char	15		no
	shipaddr	char	20		no
	shipcity	char	15		no
	shipstate	char	2		no
	shipzip	number			no
orderitem	orderno	number		non-uniq	no
	lineno	number			no
	partno	number			no
	partprice	number			no
	orderqty	number			no
	instruct	char	20		yes
part	partno	number		unique	no
	desc	char	15		no
	price	number			no
	unitqty	number			no
citystate	city	char	15	non-uniq	no
	state	char	2		no
state	state	char	2	unique	no
customer	custno	number		unique	no
	custname	char	15		no
	custaddr	char	20		no
	custcity	char	15		no
	custstate	char	2		no
	custzip	number			no

Figure 3.4

When thinking about an interactive CRT application, it is common to associate a screen page with the logical unit of work. An operator thinks in terms of processing a display a page at a time. Although IAF allows an application to occupy multiple pages, the unit of work is a block, or more precisely a current record within the current block. The distinction is important in understanding the flow of IAF applications. A loose association exists between a block and the application's screen pages. A block may span multiple pages, or multiple blocks could occupy the same page. The application designer has complete freedom to position a field within a block. The display location of a field will be discussed later in this section.

In the sample order entry application three blocks have been defined. Figure 3.5A defines the relationship between each screen page and the associated block and database tables. The 'Order' block was defined with one record display area, therefore only one record occurrence of the order information may be displayed. In contrast, the 'Orderitem' block contains five record display areas permitting up to five orderitem records to be simultaneously displayed.

Figure 3.5B describes the blocks associated with the "employee" application.

3.3.1.1 **Block Control**

IAF provides the capability for an operator, within a multi-block application, to terminate the processing of one block and begin another. When an application is executed the operator is positioned to the first block which was defined in IAG. The terminal operator can move from block to block using the 'Next Block' and 'Previous Block' function keys. The order of block processing is determined by the order of definition. Requesting the 'Next Block' function while processing the last defined block will jump forward to the first block. Conversely, a 'Previous Block' request in the first block will cause the last block to be processed.

Block Name	Form Name	Table Name	Record Areas
order	Order Form	order	1
orderitem	Order Item Form	orderitem	5
browse	Order Browse Form	order	8

Order Entry Application - Block Specifications

Figure 3.5A

Block Name	Form Name	Table Name	Record Areas
emp	Personnel Record	emp	1
projects	Projects Assignments	pe	3

Employe Application - Block Specifications

Figure 3.5B

The blocks of an application are logically chained in the order in which they were defined. The chain for the order entry application is illustrated in figure 3.6. The 'Order' block is the first to be processed. Normally, the operator would depress the 'Next Block' function key to execute the 'Orderitems' block. When the order is completed, 'Previous Block' brings the operator back to the 'Order' block in preparation to enter another order.

Processing a customer inquiry requires the use of the 'Browse' block. The block is obtained by advancing forward using the 'Next Block' or backwards using the 'Previous Block' keys. From this block the operator can query existing orders. When the desired order has been located, the operator can view the expanded order information by advancing to the 'Order' block.

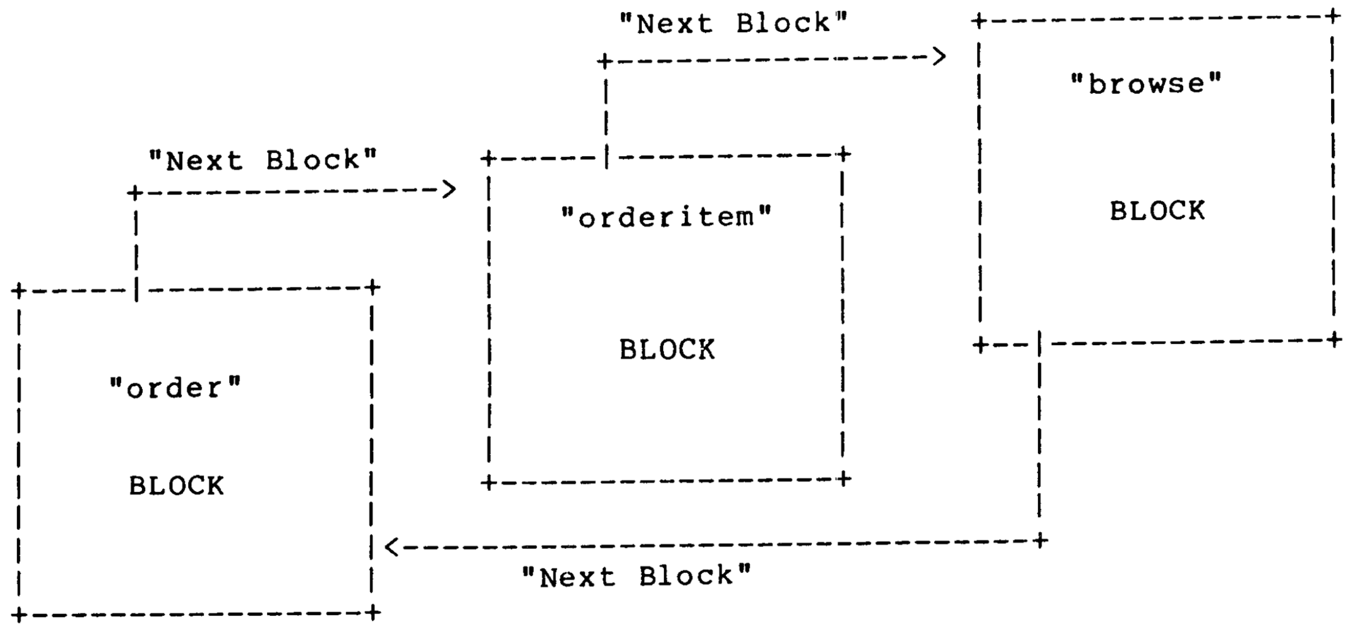
When an operator leaves a block the current data is preserved and will be redisplayed when that block is re-entered.

Advancing blocks in this manner may become tedious, especially if they are processed in a random order. Sequential block processing poses less of a problem. Trial and error will determine an optimal size for an application. Breaking a large application into multiple applications, with fewer blocks, may simplify the operator interface.

3.3.1.2 **Block Processing Modes**

An application block can be processed in either 'Insert' or 'Update' mode. The processing mode states the operator's intentions. In 'Insert' mode, records may only be inserted. In update mode, records may be retrieved; retrieved records can be updated or deleted. Although IAF will permit any user an attempt to retrieve or modify data in a table, ORACLE will reject the request if the appropriate database access has not been granted.

When an application is initially executed it is automatically placed in 'Update' mode. The 'Change Mode' function switches the mode of operation. A mode is retained until explicitly changed by the operator. Advancing from one block to another does not effect the processing mode.



"Order Entry" Application - Block Processing Order

Figure 3.6

3.3.1.2.1 **Insert Mode**

To insert a record the application must be in 'Insert Mode'. 'Clear Block' will clear the block and initialize any default values. The operator can then enter data into any field which was defined as enterable. Fields are entered one at a time until all the fields within a record area are complete. Movement from one field to another is accomplished using the 'Next Field' and 'Previous Field' functions. The 'Insert' function signals the completion of data entry and triggers the insertion of a new row into the table.

3.3.1.2.2 **Update Mode**

Within 'Update' mode, an operator may construct an inquiry to retrieve one or more stored records. To retrieve a record, the operator may supply field values for the associated columns in the target record. The 'Inquiry' function initiates the retrieval of data. IAP will dynamically construct a SQL query with a SELECT clause listing all the database fields in the block, and a WHERE clause specifying an '=' condition for each entered value. The operator may explicitly supply additional retrieval conditions which are added to the generated WHERE clause.

If no field values were entered and no additional conditions are supplied, a WHERE clause will not be generated, and every row in the table will be returned.

For example, using the 'Order' block the operator could request all the orders for customer '18945' . The entered data would be :

```
CUSTNO: 18945
```

Since no additional WHERE clause conditions were specified, the following SQL statement will be created and executed by IAP:

```
SELECT *
FROM   ORDER
WHERE  CUSTNO=18945
```

After depressing the 'Inquiry' key, the operator will be prompted to supply additional WHERE clause text. Up to one line of input is permitted. The entered text is appended to the IAP generated WHERE clause. The 'AND' logical operator is used to connect the generated predicates with those supplied by the operator.

This supplemental text offers the operator greater flexibility in selecting the records to be retrieved. If no fields are entered within the block, only the supplemental text will comprise the WHERE clause. An ORDER BY clause could be specified as part of this text, allowing the resultant rows to be returned in a sorted order.

Following the retrieval request the records satisfying the query will be displayed. The maximum number of records displayed is equal to the number of record areas defined for the block. If more records are retrieved than can be displayed, the 'NEXT SET' function will cause the next page of records to be displayed. The record areas may be scrolled by entering the 'Next Record' function when the cursor is positioned on the bottom line. In this case records are scrolled up , with the next record displayed in the bottom area. The top record will disappear from view. There is no facility to page backward; the retrieval operation must be re-executed to review a passed record.

3.3.1.2.3 **Updating a Retrieved Record**

Retrieved records may be updated one at a time. Only the record in the 'current' record display area can be updated. The 'current' record area is the area in which the cursor is positioned.

Only fields for which update has been allowed may be modified. When all the desired fields have been modified, the 'Update' function is used to write the updated record to the database.

The operator may move the cursor to another record display area using the 'Next Record' and 'Previous Record' function keys.

3.3.1.2.4 **Deleting a Retrieved Record**

The 'Delete' function deletes the 'current' record from the database.

3.3.2 **Field Specification**

A block can contain one or more fields. Each field within a block is identified by a unique name. Field names do not have to be unique across blocks.

'Database' fields map directly to a column in the block defined table. Fields which are not mapped are called 'Control' fields. Control fields can be entered by the operator, or initialized with database information. They are useful for carrying data forward from block to block, or as reference aids for the operator.

The application designer can specify whether a control or database field can be entered by the operator. Updating a database field may be permitted only if the field is not part of the primary key (see Section 3.3.2.3 for a discussion of primary keys).

In the 'Order' block, order number is an example of a database field. Its value is mapped to the 'orderno' column in the 'order' table. An operator can input a value into this field, but since the field is part of the primary key updating is not permitted.

The customer name is a control field and is used to verify that the correct customer number has been entered. Entering this field is not permitted.

3.3.2.1 **Field Data Types**

Each field has an assigned data type. The type will establish a set of validation rules which will be applied when the field is either entered or modified.

3.3.2.1.1 **Character Data Types**

Two character data types are supported. The column associated with this field type must be specified as 'CHAR' on CREATE TABLE statement.

Alpha - Only alphabetic characters A thru Z and space are permitted

Char - Any printable character is acceptable

3.3.2.1.2 **Numeric Data Types**

There are three explicit forms of the numeric data type.

Number - May contain the digits '0' thru '9', 'E', '+', '-', and '.'. Numbers may be entered or displayed in scientific notation (ie. 3,287 is 3.287E3).

Int - Only integer numbers are accepted. Only digits '0' thru '9', '+' and '-' are accepted.

Money - A special number format for field which contain money values. Excludes scientific notation, and is limited to two digits to the right of the decimal point.

3.3.2.1.3 **Date Data Types**

There are three 'date' data types. Each has a specific entry/display and storage format.

Date - The entry/display format is mm/dd/yy; a length of 8 is required. The values for mm, dd, and yy are validated. mm must be in the range 1 to 12, dd must be in range implied by the month including leap years. The value is converted and stored in a Julian Day number format(Discussed below).

Edate - Similar to 'date' type except entry/display format is in European date format (dd/mm/yy). The same validation is used and the value is stored in Julian Day number format.

YYMMDD - The entry/display format is mm/dd/yy and has the same validation as 'date' type fields. The value is stored as a 6 digit integer in the format YYMMDD.

IAF converts and stores 'date' and 'edate' type fields from a calendar date into its corresponding Julian Day Number. On output, fields of this type are assumed to be in Julian Day format, and are converted back to calendar date. Simple mathematical manipulation (such as addition or subtraction of days) may be performed on date data in this format. There are limitations with 'date' and 'edate' type fields. No format conversion is performed on 'date' and 'edate' type fields when referenced in Host Language program calls. Therefore, when these fields are referenced in a host program, they will appear as a julian date in numeric form. If calendar date format is required in such a program, logic would have to be included to perform the conversion from julian number format to calendar date format. The algorithym for converting to and from julian day is included with the documentation.

3.3.2.1.4 **Time Data Type**

Time - Defines a field type of 'time' with an entry/display format of HH:MM:SS. The field must have a length of 8. Entered values will be verified for a valid time. The stored format is the number of seconds from midnight.

The 'time' format is only recognized by IAF; an operator using UFI or a Host Language program would have to perform their own conversion.

3.3.2.3 **Default Values**

A default value may be assigned for any data or control field. The default values are used only when a block is processed in 'Insert' mode. When the block is 'Cleared' the default values will be displayed. The default value may be a literal, system default, or the value of another field. Literal values must conform to the valid formats for the field's data type. The system defaults are the current date for date type fields, and the current time for 'time' fields.

If a field's default value is to be copied from another field, both fields must have the same data type. The copied field should be from a different block. .np Assigning a default value and designating a field as "non-enterable" restricts the value to the default. For example, in the "employee" application, a version of the application could be generated which would restrict user's to entering employees only into a specific department.

If a default is not specified and no data is entered, the field is assigned the "NULL" value.

3.3.2.3 **Field Initialization**

3.3.2.3.1 **Copying the Primary Key**

The term primary key is used in ORACLE to refer to the first column defined in a table. However, in IAF, the term is used in its strictest sense to mean that collection of columns which uniquely identifies a row in a table. Some tables may not have a primary key in this sense if there are rows in the table which are duplicated. Currently in IAF, only tables which have a primary key can be updated. This is because IAF qualifies its updates and deletes by the primary key. Updating a row then, will have the effect of updating or deleting all rows where the set of columns comprising the primary key has the same value as the row being updated.

IAF allows the user to specify those fields in a block which comprise the primary key. Any field which is part of the primary key may have its value copied from a field in a different block. The value will be copied in both 'Insert' and 'Update' modes. In a relational system, the relationship between two tables is established by a common domain. For example, employees and departments by department number, and orders and line items by a common order number. This facility allows these records, which are entered in separate blocks, to be related by automatically copying the common data from one block to another.

In the 'Orderitem' block the order number is not displayed. However, order number is necessary to associate a set of line items with a particular order. In the 'Order' block definition, 'orderno' is defined as a non-display field whose value is copied from the 'orderno' field in the 'Order' block. Thereby the common order number is automatically carried forward into each 'Orderitem' record.

3.3.2.3.2 **Selecting Into a Field**

A SQL SELECT statement may be defined with any field. This query is executed each time a field is entered or modified, and also after a query. This SELECT should not be confused with the SELECTs which are implicitly created by IAP to retrieve records into a block. The SELECT may serve three purposes:

- Test for the existence of a field's value in a table of acceptable values.

- Select information to be displayed which will aid the operator in verifying the correctness of the input.

- Select information to initialize the value of one or more fields.

The last two purposes will be discussed in this section. Refer to section 3.3.2.5 for a discussion on "Existence Checking".

Any field can be assigned a value which is returned by a SQL query. The 'INTO' clause identifies the block fields where the data is to be returned. Each field named is postionally associated with a corresponding column name in the 'SELECT' clause. The 'FROM' clause identifies the queried tables.

Only the first row returned from the query is processed. All subsequent rows are ignored. The fields identified in the 'INTO' statement are assigned the corresponding values returned in the first row.

The 'WHERE' clause should normally contain a single predicate of the form:

```
WHERE <column name> = &<field name>
```

where <column name> is the database column associated with the specified <field name>. The '&' designates the <field name> as a substitution variable whose value will be substituted into the 'WHERE' prior to execution. The <field name> is commonly the field in which the SQL statement is being defined.

For example, 'partno' in the 'orderitem' block has the following query defined:

```
        SELECT  desc, price, unitqty
        INTO    orderitem.partdesc, orderitem.partprice,
                orderitem.unitqty
        FROM    part
        WHERE   partno = &orderitem.partno
```

Field names may be qualified by the block name they reside in and this is generally considered a good idea, especially if the field name appears in more than one block. When a part number is entered the query is executed, and the fields 'partdesc', 'partprice', and 'unitqty' will be assigned the values returned. These values will be immediately displayed and overwrite any previous values. For each new value of 'partno', the SQL query is re-executed, and the fields in the 'INTO' clause re-initialized.

In the above example, 'partdesc' is the description of the entered 'partno'. This information aids the operator in determining whether the correct part number was entered. 'partprice' and 'unitqty' initialize the part and quantity fields and which will be stored with the related line item.

NOTE: When SELECTS are executed, not all fields appearing in the INTO list may be set. In INSERT mode, all items will be set. However in QUERY/UPDATE mode, only those fields which are not columns in the table associated with the block will be set. The reason for this being that when records are retrieved, the stored value should not be overwritten by a field SELECT which executes after a row is returned.

3.3.2.4 **Record Uniqueness**

IAF has an opton that allows a user to verify that a record is unique prior to inserting a record. Unique in this case means that no two records will have the same primary key (as defined above). This is because ORACLE can prevent duplicates only in the case where one column defines the primary key. The uniqueness check is performed when all the fields in the primary key have been entered. If the record is found to exist already, the entire record as it is stored in the database will be retrieved and the message "Record already exists" will be displayed, otherwise the message "End of Query" will be displayed. This option only occurs if in INSERT mode.

IAF will not allow fields within the primary key to be updated. If these fields need to be modified, the record must be deleted and re-inserted with the new values.

3.3.2.5 **Field Editing**

The IAP utility will perform edit checks on fields as they are being entered for either update or insertion. If a field fails an edit check the operator is immediately notified and requested to enter correct information.

The following is a list of the supported edit checks:

Data Type Check
The field will be verified based on the defined data type (Alpha, Char, Number, Date, etc.).

Length Check
The defined length of the field determines the maximum number of characters allowed. Less than the maximum is allowed unless the field is defined as fixed length, requiring each position to be entered (ie. zip code, telephone number, etc.).

Range Check
A value range may be specified for a field of any data type. A high value, low value or both may be specified.

Required Field Check
A value can be required for a field. The value can be entered by the operator or assigned as a result of a copy, SELECT or default value.

Note that if a value is not required for a field, but the associated database column is defined as NO-NULL, the update or insert will be rejected by ORACLE.

Existence Check
As discussed in section 3.3.2.3.2 "Selecting Into a Field", a SQL statement may be defined which checks the existence of the entered value in a database table. The single predicate in the 'WHERE' clause compares the column with the entered field value. If no rows are returned , the field value does not exist in the table, and the entry is rejected.

For example, in the 'Order' block, the entered value for state code is compared with a list of valid state codes. The following SQL statement was defined within the 'shipstate' field:

```
SELECT state
FROM state
WHERE state = &shipstate
```

Note that no INTO clause was specified; therefore, no fields were initialized as a result of this query.

3.3.2.6 **Operator Aids**

A help message can be defined to aid the operator in entering correct information. This message is displayed at the bottom of the screen page when the 'Help' function is requested.

A 'Display Attributes' function allows the operator to request the information about the current field. The message is displayed in the system message area at the bottom of the screen. The attributes include the field's data type, whether is is updatable, and whether it is mandatory.

In addition to this requested information, the current screen page, block processing mode, character insert or replace mode, and number of records retrieved are continuously displayed at the bottom of the screen.

Error and status messages are also displayed when necessary in the system message area.

3.3.2.7 **Field Display**

A field can be defined as either display or non-display. For display fields, the page, line, and column location on the screen must be specified. Each displayed field can have a prompt message which is displayed above or immediately to the left of the field's display location.

More than one record area can be defined for a block. The application designer can define the number of lines in each record display area, the number of display areas, and the base or starting line of the first record area.

For example, the 'Browse' block can contain 8 order records, with each record displayed on a single line. If a query results in more than eight rows, the record display area will scroll up, with new records appearing at the bottom, and earlier records disappearing at the top. The scrolling operation is controlled with the 'Next Record' or 'Next Page' functions. Once a record has scrolled off the screen, the query must be reissued to be viewed again.

Field prompts will always be displayed within the first record area. If multiple areas are defined, the prompts can optionally be repeated in every area.

The 'Orderitem' block allows input and display of multiple order items. The 'SPECIAL INSTRUCTIONS:' prompt is repeated in each record as an operator aid. As records are inserted, the 'Next Record' key will advance the cursor to the next record area down the screen. When the screen is full, the 'Next Record' key will scroll the records up, allowing a new record to be entered in the bottom area.

3.3.3 Screen Formatting

Each application block can occupy one or more screen pages; more than one block can reside on the same page. Page numbers are assigned by the designer. The numbers define specific pages but do not imply a processing order.

A screen overlay of descriptive text may be defined for any page. The text is specified separately from the block and field definitions. IAP will merge the prompts and field values with the screen image format, forming the composite page.

Figure 3.7 shows the line drawing and explanatory text used to create the screen display for the 'Order' block.

3.4 Terminal Support

An IAF application can be executed from any CRT terminal device which has basic cursor control, character by character transmission, and has been defined using the procedure outlined in the "IAF Terminal Operator's Guide". The user has complete flexibility in function key and control sequence definition. A terminal identifier is associated with each definition. This identifier is specified when IAP is executed.

```
+-----------------------------------------------------------------------------+
|                                                                             |
|                           ORDER ENTRY APPLICATION                           |
|                                                                             |
|                           " O R D E R   F O R M "                           |
|                                                                             |
+-----------------------------------------------------------------------------+
|                                                                             |
|                                                                             |
|                                                                             |
+--------------------------------------+--------------------------------------+
|       CUSTOMER INFORMATION           |         SHIP TO INFORMATION          |
+--------------------------------------+--------------------------------------+
|                                      |                                      |
|                                      |                                      |
|                                      |                                      |
|                                      |                                      |
|                                      |                                      |
+--------------------------------------+--------------------------------------+
|                                                                             |
+-----------------------------------------------------------------------------+
| Next Form is : Order Item Form        Previous Form is : Order Browse Form |
+-----------------------------------------------------------------------------+
%page
```

'Screen Formatting Text'

Figure 3.7

Pages 4-54 through 4-55 have been omitted.

Pages 4-54 through 4-55 have been omitted.

A P P L I C A T I O N - D E F I N I T I O N

I N T E R A C T I V E - A P P L I C A T I O N G E N E R A T O R

4.1 Introduction

The Interactive Application Generator (IAG) is used to define an IAF application. Interactively executed from the user's terminal, IAG will enter into a dialogue whereby questions will be asked about the application. These questions fall into the following categories:

+ General questions about application execution

+ Block specification questions

+ Field specification questions

+ Screen layout - Descriptive text specification

As each response is entered, IAG will save the question text and associated response in a user file. IAG can later be directed to use the response file as an alternate input source to regenerate an application.

For simple application changes the response file may be edited using a standard text editor. For more extensive changes the response file can be combined with additional terminal entered input. Commands are provided which direct IAG to alternate between these input sources.

When an application has been completely and correctly defined, IAG will compile the responses into an IAP executable module. To execute the application, the user must invoke the IAP Utility as described in the IAF Terminal Operators Guide.

This section will present the application definition process. Figure 4.1 describes the components of the application development process. A conceptual overview of IAF, and a description of its features was provided in sections 2 and 3. That material should be referenced to provide the overall structure and design of an application, and explain the implications of the IAG questions and responses.

```
    --------          +-------------------+
   /        \         |    INTERACTIVE    |
  |  Questions |<------->|                   |            +------------------+
  |     &      |<------->|    APPLICATION    |<--------->|  <applname>.inp  |
   \ Answers  /         |                   |<--------->|                  |
    --------          |   GENERATOR (IAG) |            +------------------+
   ----------         +-------------------+
  /          \                 ||                        "Response File"
  --------------               ||
                               ||
"Application Designer"         ||
                              \  /
                               \/
                      +-------------------+
                      |  <applname>.frm   |   "Application Image File"
                      |                   |
                      +-------------------+
                               ||
                               ||
                               ||
                               ||
                              \  /
                               \/
    --------          +-------------------+
   / NAME:___ \       |    INTERACTIVE    |            +------------------+
  |  ADDR:___  |<------->|    APPLICATION    |<--------->|      USER'S      |
  |     .      |<------->|    PROCESSOR      |<--------->|     DATABASE     |
   \    .     /       |       (IAP)       |            +------------------+
    --------          +-------------------+
   ----------
  /          \
  --------------

 "Terminal Operator"
```

Application Development Process

Figure 4.1

4.2 Executing the IAG Utility

IAG is invoked from the user's terminal by entering the following command:

```
IAG <applname> [-<options>]
```

Where:

<applname>	Name of the application being defined. Any character string which is acceptable as an operating system file name is allowed.

When IAG is invoked, it will automatically search for an existing response file with the name '<applname>.inp'. If present, this file will be used as the input source. Additionally, for each execution a new version of the response file will be created. When the host operating system is RSX11M, VMS or IAS, the new file will be assigned the next highest version number.

The <options> parameter permits the user to control the creation and use of the response file. This parameter is optional.

T -	Direct IAG to use the response file for input; suppress terminal output of the question and answer text. Only error messages will be displayed.
S -	Suppress question text when creating the new response file. Only a list of the responses will be saved.
O -	Suppress creation of a new version of the response file.

4.3 Defining the Application

An application is defined by responding to a series of questions. These questions fall within the general category identified by the question number:

G-x	-General Questions about the application
B-x	-Block Questions
F-x	-Field Questions - field specification information
D-x	-Field Display Questions - field display information
E-x	-Field Edit Questions - field editing information

Figure 4.2 provides a list of these questions with their associated numbers. The question numbers are used to aid in the discussion, but will not be displayed during IAG execution.

In addition free-format descriptive text can be specified for each screen within an application.

Figure 4.3 provides an outline of the order in which the categories of questions are asked. First, the general questions about the application are presented. Then the first block is specified. Within that block, each field is defined. When all the fields in one block have been defined, that block specification is complete. The block questions will be reasked for each successive block in the application. Lastly, the user can define additional descriptive text for any screen page.

Certain questions are conditionally asked depending on previous responses. The column labeled 'A/C' in Figure 4.2 indicates whether the question is 'A'-Always or 'C'-Conditionally asked. For conditional questions, the last two columns identify the responses to other questions which will cause the indicated question to be asked. If more than one condition is listed, each condition must be met, unless an 'or' is specified.

I N T E R A C T I V E - A P P L I C A T I O N - G E N E R A T O R

Question List

?-no.	Question Text	A/C	Response To	?no.
G-1	Database :	A		
G-2	Sequel Workspace size ?	A		
B-1	Block name:	A		
B-2	Table name:	A		
B-3	Check for uniqueness before inserting Y/N	A		
B-4	Buffer how many records ?	A		
B-5	Base CRT line ?	C	> 1	B-4
B-6	How many physical lines per record ?	C	> 1	B-4
F-1	Field name :	A		
F-2	Type of field :	A		
F-3	Length of Field :	A		
F-4	Is this field in the base table Y/N:	A		
F-5	Is this field part of the primary key Y/N :	C	Y	F-4
F-6	Field to copy primary key from :	C	Y	F-5
F-7	Default value :	A		
F-8	Allow field to be entered Y/N :	C	non-blank	D-1
F-9	Allow field to be updated Y/N :	C	N	F-5
			Y	F-8
F-10	SQL >	A		
F-11	Message if value not found :	C	non-blank	F-10
D-1	Page :	A		
D-2	Line :	C	non-blank	D-1
D-3	Column :	C	non-blank	D-1
D-4	Prompt :	C	non-blank	D-1
D-5	Display prompt above field Y/N :	C	non-blank	D-4
D-6	Display prompt once for block Y/N	C	non-blank	D-4
			> 1	B-4

Figure 4.2

?-no.	Question Text	A/C	Response To	?no.
E-1	Is field mandatory Y/N :	C	non-blank Y	D-1 F-8
E-2	Is field fixed length Y/N :	C	non-blank Y	D-1 F-8
E-3	Auto jump to next field Y/N :	C	non-blank Y	D-1 F-8
E-4	Convert Field to upper case Y/N :	C	non-blank Y	D-1 F-8
E-5	Help Message :	C	non-blank Y	D-1 F-8
E-6	Lowest value :	C	non-blank Y	D-1 F-8
E-7	Highest value :	C	non-blank Y	D-1 F-8
E-8	Must value exist Y/N :	C	non-blank	F-10

A: Question is 'ALWAYS' asked

C: Question is 'CONDITIONALLY' asked

Response to: For conditionally asked questions, question is triggered by the indicated response to the question in the '?-no.' column.

Figure 4.2 (Continued)

```
+------------------+
|GENERAL QUESTIONS |
|    G-x ?'s       |
|                  |
+------------------+
                   +---------------+
                   |BLOCK QUESTIONS|
                   |   B-x  ?'s    |
                   +---------------+
                                   +---------------+
                                   |FIELD QUESTIONS|
                                   | F-x,D-x,E-x ?s|
                                   +---------------+
                                           o
                                           o
                                           o
                                   +---------------+
                                   |FIELD QUESTIONS|
                                   | F-x,D-x,E-x ?s|
                                   +---------------+
                   +---------------+
                   |BLOCK QUESTIONS|
                   |   B-x ?'s     |
                   +---------------+
                           o       +---------------+
                                   |FIELD QUESTIONS|
                           o       | F-x,D-x,E-x ?s|
                                   +---------------+
                           o               o
                                           o
                                           o
+------------------+
| DESCRIPTIVE TEXT |
|                  |
|   SPECIFICATION  |
+------------------+
```

Application Definition Outline

Figure 4.3

```
------------------------------------------------------------------------------
|
|      E M P L O Y E E      P E R S O N N E L      R E C O R D
|
+-----------------------------------------------------------------------------
|
|   NUMBER : ____                  SALARY     : _______
|     NAME : _________         COMMISSION     : _______
|      JOB :_________
|                 DEPTNO: __   DEPT NAME : __________
|
+=============================================================================
|
|      E M P L O Y E E      P R O J E C T      A S S I G N M E N T S
|
+-----------------------------------------------------------------------------
|                 PROJNO          PROJECT   NAME
|                  ___            ______________
|                  ___            ______________
|                  ___            ______________
|
+-----------------------------------------------------------------------------
```

Sample "Employee" Screen Layout

Figure 4.4

Each question requires either a 'YES/NO' or literal value response. A "Y/N" in the question text indicates that a 'YES' or 'NO' is required. All other questions require a character or numeric value. Character values can contain any printable character. If an incorrect response is entered, IAG will display an error message and reask the question. A correct response to each question is required before proceeding to the next question. Each correct response will be added to the <applname>.inp response file.

Questions will continue until the entire application is defined. An application is considered complete when IAG encounters a '%end' response to a descriptive text prompt. The user may prematurely terminate the session by entering a ^Z (Control Z). This will cause a normal exit from IAG with the <applname>.inp response file containing all the valid responses entered prior to termination. No IAP module will be created for incomplete applications.

When an application is executed by IAP, blocks are processed in the order in which they were defined. This order is followed regardless of the order of the actual display pages. For example, if the first block defined is on page 1, the second on page 3, and the third on page 2, the order of page display is 1,3,2.

For fields within a block, the cursor will advance from field to field in the order of definition. To simplify the operator interface, fields should be defined in their order of display (left to right, top to bottom, etc.)

The fields within a block may be displayed on one or more pages. Advancing to a field on another page will cause the new page to be displayed. Unnecessary page switches may be distracting to the operator, and will cause additional delays in transmitting the screen images.

Many of the questions relate to ORACLE database, table and column names and definitions. During definition and compilation, IAG does not access the ORACLE dictionary to verify their existence or validate their data characteristics. Any errors of this type will be detected by IAP during application execution.

4.3.1 General Questions

The questions in the General (G-x) Category are asked only once for an application. Only one application can be defined within a single execution of IAG.

G-1: Database :
Specify the ORACLE database name to be accessed by this application. Only one database can be accessed per application.

G-2: Sequel Workspace size :
Specify the size of the SQL workarea in 1K increments, which is required for this application. If no value is specified, the ORACLE default value will be used.

4.3.2 Block Questions

The block related questions are asked once for each block definition. An application can contain one or more blocks. Following each block definition, the fields within that block must be defined.

B-1: Block Name :
Specify the name of the block being defined. A blank or null response indicates that no more blocks are to be defined, and IAG will skip to the 'Screen Layout Questions' described in section 4.3.4.

B-2: Table Name :
The database table name referenced by this block. Only one table may be referenced within a block. If no value is specified, the table name will default to the block name.

B-3: Check for uniqueness before inserting Y/N :
Specify whether you want IAP to verify, prior to insertion, that a row does not exist in the table with the same primary key value. Refer to section 3.3.2.3 for a discussion on 'Primary Key Specification'.

B-4: Buffer how many records ?
Specify the maximum number of database records which can be displayed within this block. An integer value between 1 and 22 is required. All the records must be contained within one screen page. If only one record occurrence is to be displayed, enter a value of 1.

B-5: Base CRT line ?
Specify the screen line number where the first line of data for the first record of a multi-record block is to be displayed. An integer between 1 and 22 is required.

B-6: How many physical lines per record ?
Specify the number of display lines for a single occurrence of a record. An integer value between 1 and 7 is required.

Note: The following rules apply to the layout of a multi-record block:

- Line 1 through 22 is available to the user. Lines 23 and 24 are reserved for system information.

- (Base line + (Number of Records * Lines per Record)) <= 22

- If the prompts are displayed above the field display area base line value of 1 is invalid.

For a detailed discussion of Multi-Record Block Layout refer to sections 3.3.1 and 3.3.2.4.

4.3.3 **Field Questions**

For each application block, one or more fields may be defined. Field questions fall into three categories: specification, display and edit. In the following sections the questions will be grouped by category. The order in which they are discussed may vary from the order in which they are actually asked. Additionally, questions from the different categories may be mixed.

4.3.3.1 **Field Specification Questions**

F-1: Field Name:
Specify the name of the field being defined. Any charcter string will be accepted. Field names must be unique within a block, but may be repeated in different blocks. For database fields, it must be a column name from the block's associated table. A blank or null response indicates the end of field specification for the current block.

F-2: Type of Field:
Specify the data type for this field. Valid data types are:

Alpha	- Only alphabetic characters A-Z are permitted. Upper and lower case is supported.
Char	- Any printable character is permitted
Number	- A number which can contain the digits 0-9, '.','+','-'. Numbers may be specified in scientific notation (2.3E2 = 230).
Int	- Only integer numbers are accepted.
Money	- A special number format which excludes scientific notation (ie. 3.7E2), and is limited to two digits to the right of the decimal point.
Date	- Only a valid date of the format mm/dd/yy is permitted. mm must be within the range 1 through 12. dd must be within the range implied by the specified month (including leap years). Dates entered in this format will be stored within the database in the internal Julian Day Number format.

1 Edate	- Similar to 'Date' type except the date is entered and displayed using the European format of dd/mm/yy. The same validation is used and the value is stored in Julian Day Number format.
YYMMDD	- A date field which is entered and displayed in the format mm/dd/yy, but stored as a numeric value in the format YYMMDD. This date is not converted to Julian Day Number format.
Time	- Allows the entry and display of a time value in the format HH:MM:SS. The value is converted to the number of seconds since midnight.

F-3: Length of Field:
Specify the length of the field. An integer value between 1 and 79 is required. For database fields, this value should be consistent with the table column length. The column value will be truncated if its length exceeds the field length.

'Time', 'dae''edate', and 'YYMMDD' type fields must be defined with a length of 8.

F-4: Is this field in the base table Y/N :
Specify whether this field is to be mapped to a column in the table defined for this block. If Y, the field name must be the same as a table column name.

F-5: Is this field part of the primary key Y/N:
Specify whether this field is part of the primary key. Refer to section 3.3.2.3 for a discussion of primary keys. At least one primary key field must be defined in each block.

F-6: Field to copy from :
Specify the name of a field in this or another block, whose value will be copied into this field when the block is initialized. Only fields which are part of the primary key may have their value copied from another field. The format of this response is [<blockname>.]<fieldname>. <blockname> is required if <fieldname> is from another block and is not unique within the application. See Section 3.3.2.3.1. for a discussion on copying fields.

F-7: Default value :
Specify the value to be assigned to this field when the block is cleared or initially entered when in 'INSERT' mode. Three forms of default values may be specified:

- A default value may be either a character or numeric literal. Character literals must be enclosed in single quotes (ie. 'CA'). The value must conform to the field's data type. The correct data format must be specified for date and time type fields (03/31/81 , 12:13:46)

- A default value may be copied from any other field in this of another block. The field name is specified as [<blockname>.]<fieldname>; <blockname> is required only if <fieldname> is not unique within the application.

- Date and time fields may be assigned the value of the current date or time. "$$date$$" is used for field types of 'date', 'edate', and 'YYMMDD'; "$$time$$ is used for 'time' field types.

Primary key fields whose value is copied from another field may not have a default value.

F-8: Allow field to be entered Y/N :
Specify, for the insertion of a new row in the database, whether the operator may enter a value. At least one field in each block must be enterable.

F-9: Allow field to be updated Y/N :
Specify, for the updating of an existing database record, whether the operator may modify the current value of this field. If the field is a part of the primary key, the reponse defaults to 'N' and the question is not asked.

F-10: SQL >
Specify the text of a SQL query to be executed when this field is entered. The use of this query is explained in section 3.3.2.3.2. The SQL statement is free format and must conform to the same rules required by UFI or a Host Language Program. The statement is not validated upon entry. The INTO statement, if provided, will be validated when the application is compiled. The remainder of the statement will not be verified until application execution. A blank or null response indicates the end of statement input. If no SQL statement is to be provided, enter a null response (<cr>) to the first prompt.

F-11: Message if value not found :
Specify the message to be displayed to the operator if no rows are returned from the above SQL statement. If the SQL query is used to test for existence of the entered value, this message informs the operator that the value was not found in the order entry example, if a part number is not found in the 'parts' table the message "Invalid Part Number" is displayed.

4.3.3.2 Field Display Questions

D-1: Page:
Specify the page number where this field will be displayed. An integer between 0 and 31 is required. A null, blank, or 0 response indicates that the field should not be displayed. All the fields within a block do not have to be displayed on the same page.

D-2: Line:
Specify the line number where this field will be displayed. For blocks with a single record display area an integer between 1 and 22 is required. For multi-record blocks this is the relative line number within the record display area. It must be an integer which is less than or equal to the response to question B-6: "How many physical lines per record?".

D-3: Column:
Specify the column number where the left-most character of this field will be displayed. An integer between 1 and 78 is required. This value must take into consideration the prompt message extending to the left or the field extending to the right. If either the prompt or field extends beyond the 80 character screen size an error will be reported. Overlaid fields will not be detected.

D-4: Prompt:
Specify a label or prompt message to be visually associated with this field. Enter any valid character string.

D-5: Display prompt above field Y/N :
Specify whether the prompt identified in question D-4 should be displayed above the field display location. If Y, the prompt is displayed on the line above the field, starting in the same column position as the field. If N, the prompt is displayed on the same line immediately to the left of the field.

D-6: Display prompt once for block Y/N :
For a multi-record block, specify whether the prompt should be repeated in every record area. If N, the field prompt will only be displayed in the first record area.

4.3.3.3 **Field Edit Questions**

E-1: Is field mandatory Y/N :
For 'Insert' mode, specify whether a value must be provided for this field. If Y, a value can be entered by the operator, assigned as a result of a SQL select, copied from another field, or assigned a default value. For database fields, if the associated column has been defined as 'NONULL' in the 'CREATE TABLE' statement, Y should be specified.

E-2: Is the field fixed length Y/N:
Specify whether the number of characters entered for this field must equal the field length. (ie. Zip Code requires all 5 digits)

E-3: Auto jump to next field Y/N:
Specify whether cursor should automatically skip to the next field after the maximum number of characters have been entered.
If Y, entry of the last character triggers an automatic 'Next Field', which causes the field to be edited and the cursor advanced to the next field.
If N, after the last character has been entered the cursor will remain in the last postion of the field. Once in the last position, an attempt to enter additional characters will be rejected, and the 'alarm' will sound. The 'Next Field' key must be depressed to initiate field editing and advancing of the cursor.

E-4: Convert Field to upper case Y/N :
Specify whether alphabetic characters should be automatically converted to upper case. Characters will be converted as they are keyed and displayed in upper case. This is equivalent to placing the keyboard into shift lock mode.

E-5: Help Message :
Specify a free format help message to aid operator in entering field data. Message will be displayed on the bottom of the screen when the operator requests the 'Help' function. A maximum of 80 characters is permitted.

E-6: Lowest value :
Specify that the field is to be range checked by providing the minimum value in the range. To pass the range check the field value must be greater than or equal to the specified value. Character literals ('Alpha' or 'Char' type) must be enclosed within single quotes (ie. 'CA'). All data types may be ranged checked. The value for 'date', 'edate', 'YYMMDD', and 'time' field types must be specified in the correct format (3/31/81, 12:12:46, etc.).

E-7: Highest value :
Specify the high value in the range check. The field value must be less than or equal to the high value to be accepted. All data types may be ranged checked. The value for 'date', 'edate', 'YYMMDD', and 'time' field types must be specified in the correct format (3/31/81, 12:12:46, etc.)

E-8: Must value exist Y/N :
Following a SQL select, a Y indicates that at least one row must be returned as a result of the query. Using the SQL query facility in this manner permits the verification that the entered value is contained in a table of all allowable values. For example, to verify that an entered part number is valid, a query of the parts table with the WHERE clause:

```
              partno = <entered part number>
```

must return a row for that part. If Y was specified, and a row was not returned, the entered value will be rejected.

4.3.4 Screen Layout Questions

Each page of an application can be enhanced with additional descriptive text. Following the prompt message:

Enter text for form:

A ':' will be displayed in the first column of the next line. Following the ':', up to 79 columns of descriptive text may be entered representing screen positions 1 through 79. The first line of text corresponds with page 1, line 1. As each line is entered the ':' prompt is redisplayed.

A maximum of 22 lines may be entered for each page. After each 22 lines the page counter is automatically incremented and the line counter reset to 1.

The line counter can be advanced to a specific line by entering the command:

%line

On the next line following this command the new line counter value is entered.

The

%page

will advance the page counter by 1 , and reset the line counter to 1.

The

%end

command will terminate the entry of descriptive text and signal IAG to compile the application. If descriptive text is not included this command can be entered immediately following the "Enter text for form:" message.

4.4 Using the Response File

Each time IAG is executed a new version of the <applname>.inp response file will be created. The format is one line listing the text of the question followed by a line listing the entered reponse. All question text lines are preceded by a ';'. Any line beginning with a ';' in column 1 is treated as a comment and will be ignored. Additional comments lines may be inserted to aid in documentation. Lines without a ';' are treated as responses and are processed by IAG as if they were entered from a terminal. For this reason the order and number of responses must be exactly as originally entered. Figure 4.5 is a listing of the response file associated with the 'employee' application.

An application can be changed and regenerated using the response file, eliminating the need to manually re-enter the original responses. Simple changes, which do not alter the order or number of responses, can be made directly within the response file using a standard text editor.

If an error is detected while processing this modified response file, the error message will be displayed on the user's terminal. The terminal then becomes the source of the input. The failing question will be asked again, and the user will be able to enter a new response. When a valid response is entered, questioning will continue from the terminal. A '%sw' reply will instruct IAG to resume the reading of responses from the response file. In this manner the source of responses can be alternated between the response file and user's terminal.

For example, assume that the field type for 'empno' was to be changed from 'int' to 'number'. However, when the response file was edited the word 'number' was misspelled. While processing this field IAG detected this error and displayed on the designer's terminal the "Invalid data type" error message. The question "Type of field : " is displayed on the terminal and IAG pauses pending a response. The designer now correctly enters the word 'number', and the next question ("Length of field :") is issued to the terminal. The designer enters a reply of '%sw' to resume the use of the response file.

```
;Database :
personnel
;Sequel workspace size ?
3
;Block name :
emp
;Table name :
emp
;Check for uniqueness before inserting Y/N :
y
;Buffer how many records ?
1
;Field name :
empno
;Type of field :
int
;Length of field :
4
;Is this field in the base table Y/N :
y
;Is this field part of the primary key Y/N :
y
;Field to copy primary key from :

;Default value :

;Page :
1
;Line :
7
;Column :
17
;Prompt :
NUMBER :
;Display prompt above field Y/N :
n
;Allow field to be entered Y/N :
y
;SQL>

;Is field fixed length Y/N :
y
;Auto jump to next field Y/N :
n
;Convert field to upper case Y/N :
n
;Help message :
Enter 4 digit employee number -
;Lowest value :
1000
;Highest value :
8000
```

"Employee" Application Response File

Figure 4.5 - Part 1 of 9

```
;Field name :
ename
;Type of field :
alpha
;Length of field :
10
;Is this field in the base table Y/N :
y
;Is this field part of the primary key Y/N :
n
;Default value :

;Page :
1
;Line :
8
;Column :
17
;Prompt :
NAME :
;Display prompt above field Y/N :
n
;Allow field to be entered Y/N :
y
;Allow field to be updated Y/N :
y
;SQL>

;Is field mandatory Y/N :
y
;Is field fixed length Y/N :
n
;Auto jump to next field Y/N :
n
;Convert field to upper case Y/N :
y
;Help message :
Enter employee name -
;Lowest value :

;Highest value :

;Field name :
job
;Type of field :
alpha
;Length of field :
9
;Is this field in the base table Y/N :
y
```

"Employee" Application Response File

Figure 4.5 - Part 2 of 9

```
;Is this field part of the primary key Y/N :
n
;Default value :

;Page :
1
;Line :
9
;Column :
17
;Prompt :
JOB :
;Display prompt above field Y/N :
n
;Allow field to be entered Y/N :
y
;Allow field to be updated Y/N :
y
;SQL>

;Is field mandatory Y/N :
n
;Is field fixed length Y/N :
n
;Auto jump to next field Y/N :
n
;Convert field to upper case Y/N :
y
;Help message :
Enter employee's job title -
;Lowest value :

;Highest value :

;Field name :
salary
;Type of field :
money
;Length of field :
7
;Is this field in the base table Y/N :
y
;Is this field part of the primary key Y/N :
n
;Default value :
1000.00
;Page :
1
;Line :
7
```

"Employee" Application Response File

Figure 4.5 - Part 3 of 9

```
;Column :
51
;Prompt :
SALARY :
;Display prompt above field Y/N :
n
;Allow field to be entered Y/N :
y
;Allow field to be updated Y/N :
y
;SQL>

;Is field mandatory Y/N :
y
;Is field fixed length Y/N :
n
;Auto jump to next field Y/N :
n
;Convert field to upper case Y/N :
n
;Help message :
Enter employee salary
;Lowest value :

;Highest value :
6000.00
;Field name :
comm
;Type of field :
money
;Length of field :
7
;Is this field in the base table Y/N :
y
;Is this field part of the primary key Y/N :
n
;Default value :

;Page :
1
;Line :
8
;Column :
51
;Prompt :
COMMISSION :
;Display prompt above field Y/N :
```

"Employee" Application Response File

Figure 4.5 - Part 4 of 9

```
n
;Allow field to be entered Y/N :
y
;Allow field to be updated Y/N :
y
;SQL>

;Is field mandatory Y/N :
n
;Is field fixed length Y/N :
n
;Auto jump to next field Y/N :
n
;Convert field to upper case Y/N :
n
;Help message :
Enter employee's commission -
;Lowest value :

;Highest value :
3000.00
;Field name :
deptno
;Type of field :
int
;Length of field :
2
;Is this field in the base table Y/N :
y
;Is this field part of the primary key Y/N :
n
;Default value :

;Page :
1
;Line :
10
;Column :
30
;Prompt :
DEPTNO :
;Display prompt above field Y/N :
n
;Allow field to be entered Y/N :
```

"Employee" Application Response File

Figure 4.5 - Part 5 of 9

```
y
;Allow field to be updated Y/N :
y
;SQL>
select dname
into dname
from dept
where deptno = &deptno

;Message if value not found :
Invalid department number
;Must value exist Y/N :
y
;Is field mandatory Y/N :
n
;Is field fixed length Y/N :
y
;Auto jump to next field Y/N :
n
;Convert field to upper case Y/N :
n
;Help message :
Enter employee's department num0er
;Lowest value :

;Highest value :

;Field name :
dname
;Type of field :
char
;Length of field :
10
;Is this field in the base table Y/N :
n
;Default value :

;Page :
1
;Line :
10
;Column :
46
;Prompt :
NAME :
;Display prompt above field Y/N :
n
;Allow field to be entered Y/N :
```

"Employee" Application Response File

Figure 4.5 - Part 6 of 9

```
n
;SQL>

;Field name :

;Block name :
projects
;Table name :
pe
;Check for uniqueness before inserting Y/N :
y
;Buffer how many records ?
3
;Base crt line ?
19
;How many physical lines per record ?
1
;Field name :
empno
;Type of field :
int
;Length of field :
4
;Is this field in the base table Y/N :
y
;Is this field part of the primary key Y/N :
y
;Field to copy primary key from :
emp.empno
;Page :

;SQL>

;Field name :
projno
;Type of field :
int
;Length of field :
3
;Is this field in the base table Y/N :
y
;Is this field part of the primary key Y/N :
y
;Field to copy primary key from :

;Default value :
```

"Employee" Application Response File

Figure 4.5 - Part 7 of 9

```
;Page :
1
;Line :
1
;Column :
22
;Prompt :
PROJNO :
;Display prompt above field Y/N :
y
;Display prompt once for block Y/N :
y
;Allow field to be entered Y/N :
y
;SQL>
select pname
into pname
from proj
where projno = &projno

;Message if value not found :
Invalid project number
;Must value exist Y/N :
y
;Is field fixed length Y/N :
y
;Auto jump to next field Y/N :
n
;Convert field to upper case Y/N :
n
;Help message :
Enter employee's assigned project
;Lowest value :

;Highest value :

;Field name :
pname
;Type of field :
char
;Length of field :
10
;Is this field in the base table Y/N :
n
```

"Employee" Application Response File

Figure 4.5 - Part 8 of 9

```
     ;Default value :

     ;Page :
     1
     ;Line :
     1
     ;Column :
     36
     ;Prompt :
     PROJECT    NAME
     ;Display prompt above field Y/N :
     y
     ;Display prompt once for block Y/N :
     y
     ;Allow field to be entered Y/N :
     n
     ;SQL>

     ;Field name :

     ;Block name :
+------------------------------------------------------------------------+
|                                                                        |
|   E M P L O Y E E     P E R S O N N E L       R E C O R D              |
|                                                                        |
+------------------------------------------------------------------------+
|                                                                        |
|                                                                        |
|                                                                        |
|                                                                        |
|                                                                        |
|                                                                        |
+========================================================================+
|                                                                        |
|  E M P L O Y E E     P R O J E C T     A S S I G N M E N T S          |
|                                                                        |
+------------------------------------------------------------------------+
|                                                                        |
|                                                                        |
|                                                                        |
|                                                                        |
|                                                                        |
+------------------------------------------------------------------------+
%end
```

"Employee" Application Response File

Figure 4.5 - Part 9 of 9

Some changes will cause the order or number of questions IAG asks to be altered. This will occur if the new response triggers a different set of conditional questions. The response file will no longer be synchronized with the list of questions asked by IAG. To compensate, the user can anticipate the new question list and insert new responses in their proper place. If previously asked questions will no longer be asked, their associated responses must be deleted.

Adjusting the response file may become extremely complicated. An alternative approach would be to temporarily switch the input source to the user's terminal. This way the designer could answer each question individually, under the control of IAG, without having to correctly anticipate the new set and order of questions. The IAG will switch from the reponse file to the designer's terminal when a '%sw' response is encountered. The designer must change the original response to a '%sw' in the first question to be answered from the terminal. This question will then be reasked from the terminal.

The designer must decide where to resume the use of the response file. A convenient approach is to resume at the beginning of the next field. In this manner all the questions from the '%sw' to the next field will be answered from the terminal. To accomplish this, the designer must delete all the unwanted responses from the response file.

For example, changing the 'Page:' response from blank to a page number will cause the associated field to be displayed. This will trigger additional questions concerning the field's display position, prompt and data entry attributes. In this case the simplest approach would be to replace the 'Page :' response with a '%sw' and delete all the remaining responses for the field. When executing IAG the 'Page :' and subsequent responses for this field would be entered from the terminal. When the 'Field name :' question is asked again a '%sw' will resume the use of the response file.

This approach may be expanded to add new fields or blocks. For example if a new field was to be added after 'ename' the "Field name : " response for 'job' would be changed to '%sw'. When the input source is switched to the terminal this question will be reasked. The new field would then be defined. When the "Field name : " question is asked again, a response of 'job' is entered. A '%sw' response to the next question, "Type of field : ", will cause the resumption of the response file.

When making this type of modification to an application the user should not suppress the creation of a new response file. The new file will contain the combined responses from both the old response file and user's terminal.

Some text editors will not permit blank lines within the edited file. To support these editors IAG will interpret a '//' in columns 1 and 2 as a blank line.

4.5 **Generating the Sample Application**

This section will examine the IAG question and answer dialogue which defined the application shown in figure 4.4. Two processing blocks were defined for this single screen application. The first deals with employee information which is inserted into or retrieved from the "EMP" table in the sample "PERSONNEL" database. The second allows an existing employee to be assigned to one or more projects. This block references the "PE" table, which was created to allow employees to be assigned to multiple projects. Refer to the SQL Language examples in the ORACLE User's Guide for more details.

The name of the application is "employee" and was generated with the command string:

```
IAG  employee
```

Figure 4.5 is a listing of the "employee.inp" response file which was created from this terminal session. This file is included within the ORACLE distribution system, which allows this application to be generated using the command above. The first question identifies the sample 'personnel' database. An initial value of 3K bytes is specified for the SQL workarea. If that proves insufficient the value could be changed within the 'employee.inp' response file and the application regenerated.

The 'emp' block is the first defined and hence will be the first processed when the application is executed. Since the table and block names are the same, the table name could have been omitted and the block name would have been used. Prior to inserting a new row IAP will check that the primary key is unique within the table. The primary key may consist of one or more columns within the block. For a multi-column key, the combined columns will uniquely identify a row. One record will be buffered for this block which means that only one record will be displayed at a time. With the block questions completed the set of field questions will be asked for each field within this block.

'empno' is a four digit integer which maps to 'empno' column in the base table, 'emp'. Fields which are not in the base table, and do not map directly to a column will be discussed later. 'empno' is the only column in the primary key, and will be used to qualify rows for update and delete operations. Although it is the primary key, its value will not be copied from another field in this or another block.

The display area for this field will begin in the 17th position of the seventh line of page 1. Since the prompt is not displayed above the field, it will be placed immediately to the left. This field can be entered, but cannot be updated. Since fields within the primary key cannot be updated, the question "Allow field to be updated" is not asked, and a response of "N" is assumed. Primary key fields are also mandatory, hence a reply of "Y" is assumed for the question "Is field mandatory".

No SQL statement was defined for this field. This feature will be discussed later for the 'deptno' and 'projno' fields. Since the field is defined as 'fixed length' all four digits of an employee number are required for a valid entry. By selecting 'Auto jump', an automatic 'next field' is generated after the last character is entered. If the entry passes validation, the cursor will be moved to the next field position.

Converting to upper case has no meaning for numeric fields, and either reply is ignored. The 'Help message' will be displayed when the operator enters the 'HELP' function key. The valid range for employee numbers is between 1000 and 8000 inclusively. This range will be checked for both data input or update.

The 'ename' field maps to the 'ename' column in the base table; 'EMP'. Since it is defined as 'alpha' only the letters 'A' - 'Z' will be accepted. Responding "Y" to 'upper case' will force lower case letters to be displayed, and stored in upper case.

When an application is executed, the processing order of fields within a block is determined by the order the fields were defined to IAG. In this block 'empno' will be the first field processed, followed by 'ename', 'salary', etc. This order is independent of the page and display position of the field. If the next field is on a different page, the appropriate page will be displayed automatically.

'salary' is a 'money' type field and is restricted to a number with exactly two positions to the right of the decimal point. Since only a 'Highest value' was specified any value less than or equal to 6000 will be accepted. The 'comm' is not mandatory, allowing the operator to omit this field on insertion or update. Fields which are not entered are stored as nulls within the database.

A SQL SELECT statement was defined for the 'deptno' field. This statement will be executed each time the value of 'deptno' is modified. The purpose of this feature is to :

- initialize other fields with a value from the database.
- display information related to the entered field.
- check the existence of the entered field in a table of acceptable values.

In this case the 'deptno' SELECT will serve to verify that the entered department number exists in the department table, and to aid the operator in verifying, by department name, that the desired number has been entered. After executing the SELECT, the INTO clause causes the 'dname' field in the block to be initialized with the value of the 'dname' column. This value will be immediately displayed for the operator. The row returned is determined by the WHERE clause; the 'deptno' column must be equal to the entered value of the field 'deptno'. The '&' signifies a literal substitution of the field value 'deptno'.

A "Y" response to the "Must value exist" question requires that at least one row be returned. This implies that the entered department number must be within the 'dept' table. If not, the value will be rejected. A special message is defined to inform the operator that the value does not exist. "Invalid department number" will be displayed when a non-existent value is entered.

The 'dname' field is not in the base table. Even though it contains database information, that data is not mapped to or stored in the "EMP" table. In this case it is used to display the department name associated with the entered 'deptno'. The field is used strictly for operator reference and may not be entered. Entering data would serve no purpose, since the entered value would only be displayed, and not retained or used elsewhere.

A blank response to the "Field name" question (following definition of 'dname') signifies the end of field specification for the 'emp' block. The next block defined is 'projects', which refers to the 'pe' database table. In this case up to three records may be simultaneously displayed as illustrated in figure 4.4. A 'base' or starting screen line must be defined for the first record display area. Additionally, each record area may occupy one or more physical lines. For this block, three records may be displayed, beginning on line 19, containing one line for each record area.

'empno' is the first field defined. This field comprises half of the primary key for the 'pe' table. Its value will always be equal to the value of 'empno' in the 'emp' block (see "Field to copy primary key from"). Furthermore, the blank response to "Page:" designates the field as non-display. The value can be neither entered nor updated within this block.

Copying the primary key in this manner establishes a logical connection between the two application blocks. Entered or retrieved 'projects' information is implicitly associated with the last referenced employee number within the 'emp' block. The blocks could have been explicitly associated by requiring the operator to repeat the 'empno' within the 'projects' block. However, this requires additional keystrokes and is error prone. Making the field non-display simplifies the display area, especially since the field is already displayed on this screen page.

'projno' is a three digit integer which comprises the second half of the primary key. Unlike 'empno', its value will be entered by the operator. It will be displayed on the first (and only) line of each record display area. The prompt "PROJNO" will be displayed once on the line above the display or 'scrolling' area (line 18). Prompts may be repeated within each record display area of a multi-record block. However, this is most useful for prompts which are displayed to the left rather than above the field display area.

A SELECT statement has been defined for the 'projno' field. Its purpose is the same as described for 'deptno' in the 'emp' block, requiring the operator to enter a project number which already exists in the 'proj' table. If it does exist, the associated project name will be displayed in the 'pname' field. An appropriate message will be displayed if the value is not found. 'pname' is not in the base table and is provided only as an aid to the operator. Data may neither be entered nor updated in this field.

A blank response to "Field name:" terminates the field definitions for the 'projects' block. The blank response to "Block name:" terminates block specification. The input which follows is a line by line specification of the text used to enhance the screen page. This text is combined with the field defined prompts to create the display screen layout. The text must be supplied a page at a time in the order of the physical page number. The '%page' command is used to indicate the end of one page and the beginning of the next. '%line' commands establish positioning to a specific line on the currently defined page. The '%end' terminates the text input process and signals IAG to compile the application.

Compilation errors will be reported at the designer's terminal. Since IAG does not access the database dictionary during this process, errors relating to invalid 'column' or 'table' specifications will not be detected. These errors will be detected and reported during IAP execution. Once compiled, an application may be immediately executed. Section 3 of the "IAF Terminal Operator's Guide" discusses the operation of this 'employee' application.

A D V A N C E D   A P P L I C A T I O N   T E C H N I Q U E S

5.1 Introduction

This section will present some addtional techniques for using IAF. These techniques include consecutive sequence number genertion, setting up field defaults to eliminate setting up field defaults to elminate redundant data entry, using views, and a more sophisticated use of SQL within an application.

5.2 Assigning Sequence Numbers

In many applications it is necessary to uniquely identify new entries by assigning a consecutive sequence number. Upon entry the new record would be assigned a value one greater than the current/maximum value. An example is the 'orderno' in the sample order entry application.

Within an IAF application, sequence numbers can be automatically retrieved. This is accomplished with a SQL SELECT statement which selects the maximum value of the sequence field plus 1. The SELECT statement in the case of the order entry application is:

```
SELECT max(orderno)+1
INTO   orderno
FROM   order
```

This statement should be defined within any mandatory field. When a value is established for that field, either by operator entry, default value, or initialized via an INTO statement, this SQL statement will be executed and the 'orderno' initialized. Using a field which is not mandatory will not insure that a value will be assigned prior to insert.

The sequence field may optionally be displayed if the operator needs the power to override the derived value. In most cases the sequence field is a part of the IAF primary key and thereby not updatable.

An assigned value is not stored within the database until the record is inserted. Thus, it is possible that another operator may be assigned the same number. To minimize the interval between the time the value is computed and stored, the field used to trigger the SELECT should be at the end of the block. This assumes that the operator will perform the insert shortly after entering the triggering field. Duplicate record insertions can be prevented by including the sequence number in the primary key and requesting a check for uniqueness prior to executing the insert.

5.3 **Operator Defined Defaults**

In many applications the operator will repetitively process a block to insert multiple records. For this type of data entry application it is convenient to assign default values. Literal default values can be assigned when the application is defined, but they remain static over the life of the application. It may be more convenient to permit each operator to define his own set of defaults depending on his talk data entry tasks.

Operator defined defaults can be achieved by creating an additional block expressly for default specification. The fields defined in this block would be referenced in the "Default :" responses of fields in other blocks. This form of default is described in Sections 3 and 4.

When the application is initially executed, the operator would first process this form to enter the desired defaults. When other blocks are cleared or initialized in "INSERT" mode, the referenced default values would be copied. The values could be changed by the operator at any time by reprocessing this default block. The defaults then remain in effect until the application is terminated.

This approach can be extended to permit the operator to retain these defaults across executions of the application. This is accomplished by creating a "default" table within the application's database. Each operator would have their own set of default records. Instead of entering the values within the "defaults" block, the appropriate record could be retrieved. The fields within the "defaults" block would map to columns within this table.

The defaults block could also be initialized from some other database table. In this case the "default" fields would not be from the base table. At least one field from the base table must be defined, and at least one of these base table fields must be designated as part of the primary key. These fields are necessary to fulfill an IAF requirement, but will have no effect on the referenced table since this block would not be used for database modification. A dummy table could be defined for this purpose.

5.4 **Using Views**

As discussed in earlier sections an IAF application allows fields on a screen to be mapped directly to the columns of a database table. There is a one to one correspondence between a field and a single column. The association is implicit, requiring that the field and column names be the same. This precludes the possibility that a screen field could map to the average salary, maximum order number, or computation of (price x qty). However, there may be a need to have application blocks which report this summary information.

An effective way to provide this type of report is through SQL defined views. A view may be defined which simply projects a set of columns from a qualified set of rows for a single table. Although IAF could map directly to the table in this case, it cannot restrict the set of rows nor project a derived column (an arithmetic expression consisting of one or more columns - ie. (qty*price)). A view defined column name would be associated with each item in the select list.

An application block could reference this view as if it were a table. The fields would refer to the view defined columns. ORACLE restricts the use of views to retrieval operations. Therefore, if an operator attempted to insert, update, or delete a record within this block an ORACLE error would be returned. Since there is no mechanism to disable these functions by application block, the designer should define these fields so they can be neither entered nor updated. (Note that IAF requires at least one field in a block to be enterable.) Any view may be referenced by an IAF block. This includes views of joined tables or views which use the built-in functions to create virtual columns (ex: avg(sal) or max(qty)).

In some applications it may be desirable to have a summary report which assembles data from multiple tables. Since multiple tables are involved a different block would have to be generated to retrieve data fromeach table. The operator would have to process each block separately to assemble the desired information. A simpler approach is to create a view which joins the referenced tables. A single block is defined which could present all the data in a single operation. Again, this block could be used for retrieval purposes only.

5.5 **Using the SELECT Statement**

Any field may have an associated SQL SELECT statement. The statement is executed each time the field is modified. A field can be modified as the result of an operator entry, assignment of a default value, record retrieval, copied primary key value, or the object of another field's SELECT statement. Recall that, an INTO clause can direct the output of the first row returned to other fields within the block. On each execution, only the first row will be processed, with the remaining rows ignored.

The main purpose of this feature is to test for the existence of a field's value within a table of acceptable values, and to initialize other fields within the block. The second use offers a great deal of application flexibility. The full power of the SQL select can be utilized. No restrictions are placed on the content of this statement which may include arithmetic expressions, multi-table joins, nested selects, and GROUP BY, ORDER BY, and CONNECT BY clauses. A block field value may be substituted within the SELECT anywhere a character or numeric literal can be used.

5.5.1 **Cross Field Totaling**

For certain applications it is desirable to display a field whose value is computed from other fields. In the order entry example, the 'cost' of an item is equal to the 'quantity' times the 'price'. Although 'cost' would not normally be stored in the database (it could be derived from the 'quantity' and 'price' columns) it may be helpful for the operator to see this value as the data is entered.

The present version of IAF does not explicitly support the equating of a field value to the result of an arithmetic expression of other fields within the block. However, the same result can be achieved with an arithmetic expression in the select list of a SQL query. In this case the expression includes any fields within the block; database columns are not involved. The values of the fields are substituted by the IAP prior to statement execution. SQL will evaluate the expression and pass the result back to the IAP. The IAP places this result into the field specified on the INTO statement.

For example, assume the following fields are part of the 'orderitem' block:

```
PARTNO       QTY     X     PRICE    =    COST
------       ---          -------       -------
A2B451        20             2.50        50.00
```

If the operator entered the 'partno', 'qty', and 'price', the 'cost' will be computed and displayed. If the operator changes either the 'qty' or 'price' a new value of 'cost' is automatically computed. This value will be recomputed each time either the 'qty' or 'price' is modified (including record retrieval). Note that if any field in the expression has not been assigned a value, or the value is null, the value of the expression is null. For example, if the annual salary (annsal = sal*12+comm) was computed using the "EMP" table, and either 'sal' or 'comm' were null, a value for 'annsal' would not be displayed.

The technique employed is to specify the following SQL statement within the 'qty' and 'price' field definitions:

```
SQL>select &qty*&price
SQL>into cost
SQL>from orderitem
```

The select list contains the expression to be computed. '&qty' and '&price' are variables whose associated field values will be substituted prior to statement execution. The INTO clause causes the result of the expression as returned by SQL to be stored and displayed in the 'cost' field. Any table name within the database may be specified in the FROM clause. Although a single row is processed by the query, no column data is returned. A WHERE clause is unnecessary.

Since the cost is not stored within the database record, the 'cost' field is defined as "not from the base table". If the computed value was to be retained the field could be mapped to a database column. In either case the field should not be entered into.

5.5.2 **Cross Field Totaling - Manual Request**

In the previous discussion, the computation was automatically triggered by modifying any field within the expression. This was accomplished by defining the SQL statement in every field participating in the computation. In this manner the SQL statement is triggered for execution each time any of the associated fields are modified. Omitting the statement from any field would inhibit the recompilation when that field was changed. This omission may be desirable if another SQL query must be triggered by that field. If omitted from all the associated fields another triggering mechanism is required.

As an alternative, the computational SQL statement may be defined with the field which is receiving the result. In the order example, the same statement as described in 5.5.1 would be defined within the 'cost' field. When the 'cost' field is modified by the operator, the statement will be triggered, and the newly computed value displayed. With this approach, the operator must manually request recompilation. The field may be modified by entering at least one valid character.

5.5.3 **Combining Detail and Summary Information**

A SQL query can be used to add summary information to the detail field entries within a block. Referring back to the order application, the operator may want to review the order following the entry of the individual line items. This 'review' block would contain the summary of the items ordered in addition to the general order information. A sample layout might be:

```
       " O R D E R   S U M M A R Y   S C R E E N "

      ORDERNO: _5674                        DATE: 04/01/81
      CUSTNO : ‾87653    CUST NAME: Leisure Time Products

      NO. ITEMS ORDERED : 6             SUB TOTAL :$ 349.52
                                        SALES TAX :   27.50
                                                   --------
                                            TOTAL :$ 377.02
```

The orderno, date, customer number and customer name fields map to the base table 'order'. 'orderno' is the primary key and is copied from the 'order' block. Number of items ordered and sub total are computed by the following SQL query:

```
SQL>select count(*),sum(qty*price)
SQL>into no_items,subtotal
SQL>from orderitem
SQL>where orderno = &orderno
```

This statement may be included with any of the four base table fields. The result is that the summary of all the items associated with the order is computed from the 'orderitem' table and displayed in the 'review' block. Within this block, only the base table fields could be modified. The other fields should not be entered into.

The 'no_items' and 'subtotal' fields contain the count of items, and the sum of the cost of each item. The 'salestax' and 'total' fields are computed with the following SQL query defined with the 'subtotal' field :

```
SQL>select &taxrate*&subtotal,(1+&taxrate)*&subtotal
SQL>into   salestax,total
SQL>from   order
```

The 'taxrate' field was derived from the 'state' field in the 'order' block and is stored in the 'order' row.

5.5.4 **Summary Reports**

In section 5.3 a technique was discussed for creating a summary block containing fields from multiple tables. That approach was to create a view which joined all the participating tables. Each column in the view is mapped to a field in the block.

An alternative approach is to use SQL queries to assemble the desired data. The block would contain both 'database' fields from the base table, and 'control' fields to receive the data from the other tables. After retrieving the base table record, the retrieved field values would trigger the defined SQL statements. Each field could have an associated query. These SELECTs would retrieve the data for the 'control' fields. Each of the 'control' fields could have its own SQL statement which would also be triggered.

APPENDIX A

CRT INTERFACE UTILITY

Introduction

This section describes the IAF utility allowing a user to specify a non-standard crt for use with IAF. This utility, called CRT, compiles information from a user database, containing crt-descriptive parameters. The compiled information is then available to the IAP processor as needed.

A common limitation of programs which use the advanced features of a CRT is that the program can only run on the CRT for which it was originally intended. The reason for this is that for every maker of CRT devices, there is a different way of invoking these features. However, most manufacturers have chosen a fairly compatible scheme for controlling CRTs. This compatibility amongst CRTs lets IAP view all CRTs as being equivalent functionally and uses variables (which are read by IAP at program startup) to characterize their differences. These variables or parameters are stored in source form in a user-provided database. After a CRT's parameters have been entered into the database, the CRT utility compiles this source description into a file which IAP can directly access.

Required CRT Features

The following characteristics are required for IAF to interface to a particular CRT:

(1) Transmits each character as its entered
(2) Use the ASCII character set

Besides the above characteristics, the CRT must be able to perform certain control functions. Most CRTs support these functions, however not all of them will employ a compatible method of invoking them. A CRT must support the following features to be compatible:

(1) Move cursor left one position
(2) Move cursor right one position
(3) Absolute cursor positioning to column and row
(4) Clear to end of screen

In addition, the following restrictions apply to (3):

(3a) Row and column numbers must be able to be expressed either as ASCII strings (ie. Row 23 would be expressed as "23") or as a single character whose ASCII value is equal to the number plus an offset (ie. If the offset was 31 then row 23 would be the ASCII character "6").
(3b) Row and column numbers must begin with 0 or 1 and increase monotonically.

If a given CRT meets these requirements then it should be able to be supported by IAF. A possible exception to this might be that the CRT has certain timing characteristics that limit how fast data/command strings can be sent. IAF assumes that the CRT can handle continous transmission of data/commands up to the baud rate of the terminal. Since most terminals meet this requirement for the required functions, this should not generally be a problem.

Optional Features Utilized By IAF

Besides the required functions, IAF can take advantage of some specialized functions that are provided by more sophisticated CRTs. These include:

(5) Underlining (or equivalent attribute)
(6) Reverse video (or equivalent attribute)
(7) Split screen scrolling

The features of (5) and (6) are attributes of the characters being displayed and as such make the following assumptions:

a) A given attribute may be turned on and will stay on until it is explicitly turned off. That is, all characters transmitted after that will have the attributes assigned until another attribute sequence is detected.
b) Turning attributes on or off does not move the cursor.
c) Turning an attribute on or off does not "mark" a certain portion of the display area as having that attribute (ie. the PE OWL has this feature).

The feature (7) refers to is the ability to define a subset of the lines on the CRT which can act as an independent scrolling region from the rest of the lines on the CRT. For example if lines 1 through 7 are defined as a scrolling region, then these lines can be scrolled up or down without affecting lines 8 through 24. Top and bottom line numbers must meet the same requirements as (3a) and (3b).

CRT Definition - Preparation

Defining a CRT can be as simple as entering data into a form. A form is provided for this purpose and can be used as soon as at least one CRT definition exists. Before defining a CRT the following steps must have been performed. Normally these steps would have been carrried out as part of the ORACLE installation procedure.

(1) Create a database called CRT (If space is a premium, any existing database can be used, in which case this step may be skipped). This is where the CRT definitions will reside. This database may be created with DBF as follows:

```
DBF C CRT CRT.DBS 1K
```

A 1k database should be able to hold quite a few CRT definitions. The database may of course be a secure database so that only certain users may define or look at CRT definitions, as in the following example:

```
DBF C CRT CRT.DBS 1K JON/DOE
```

(2) Initialize the database. This consists of creating the tables and inserting data into them. This can easily be accomplished by logging on to UFI, as follows:

```
SQL CRT
```

substituting the particular database name for CRT if not using the CRT database. The SQL statements for initializing the database are stored on a command file provided with the ORACLE system. The following command will cause UFI to process that command file:

```
@CRT.SQL
```

The last thing the command file will do is select out the names of CRTs that have been pre-loaded into the database. If the CRT in question is among these, the CRT may be directly compiled as described in the "Compiling CRT Descriptions" section of this document. Exit from UFI by typing:

```
#EXIT
```

Parameters Used to Define a CRT

Once the CRT database has been created and initialized, the process of CRT definition may begin. The parameters for defining a CRT are usually available in the CRT manual.

Values will be required (possibly NULL) for the following parameters. These parameters are stored in a table called crt. To enter these parameters into the database the crt form provided with ORACLE may be used, or SQL INSERT statements may be used to bootstrap IAP for the first CRT. Some CRT definitions will already exist in the CRT database as a result of running the CRT.SQL command file. If the terminal is already defined, the crt form may be used to add additional CRT definitions. To find out what CRT definitions exist and a description of them, type SELECT NAME FROM CRT while in UFI.

PARAMETER	**TYPE**	**DESCRIPTION**
NAME	CHAR(20)	The name by which the CRT will be referenced. It should conform to the naming conventions for files in the operating system.
LINES	NUMBER	The number of lines of text that will fit on one screen. This value is 24 for most CRTs.
COLUMNS	NUMBER	The number of characters that will fit on on line of text. This is generally 80 or 132 for most terminals.
MSGL	NUMBER	The line to display help messages and diagnostics on. This is normally line 23. It can be no greater than LINES.
MODL	NUMBER	The line to display mode and other status information. This is normally line 24. It can be no greater than LINES.

BASE	NUMBER	Should be 1 or 0 depending on how the CRT numbers its lines. For example a 24 line CRT may number its lines from 0 to 23 or alternativly 1 to 24. Generally numbering starts with 1.
OFFSET	NUMBER	If row and column values are represented as single characters (for instance when postioning the cursor) then an offset value must be supplied. This number is usually 31 (decimal) and when added to a row or column value determines the character used to represent that value to the CRT. Leave the value NULL if the terminal represents these values as ASCII strings. See the section on CRT Requirements for further details concerning offsets.
CLEARSCREEN	CHAR(20)	The escape sequence to clear the screen or clear to end of screen. This is usually a 2 or 3 character sequence beginning with the escape character (033 octal). Later it will be shown how these sequences can be easily represented in the database.
BACKSPACE	CHAR(20)	The escape sequence (or control character) that will move the cursor left one position. This is a non-destructive backspace, that is it should not erase any characters. Control H works on most CRTs.

FORESPACE	CHAR(20)	The escape sequence for moving the cursor right one position. This must also be a non-destructive sequence. A space character is not an acceptable FORESPACE sequence
GOTOXY	CHAR(20)	The model escape sequence for postioning to an absolute column and row on the screen. The word model is used because the actual escape sequence has the actual row and column values put into the sequence when its used as opposed to the model esacpe sequence which has placeholders for the row and column values. More on that later.
TSET	CHAR(20)	A set of escape sequences used to configure the terminal into a particular mode. This sequence is executed once at program startup prior to any other sequnces.
TRSET	CHAR(20)	This sequence is executed prior to leaving the program to reconfigure the terminal back to a starting state. The definition of CRTs does not provide for storing the initial state of the terminal and restoring it to that starting state.
BOLDON	CHAR(20)	This sequence turns the inverse video attribute on. Any other attribute could be substitued for reverse video as well (such as highlighting).

UNDERON	CHAR(20)	This sequence turns the underline attribute on. Like the former, any other attribute could be substituted for underlining.
ALLON	CHAR(20)	This sequnce turns both underlining and reverse video attributes on simultaneously.
ATTOFF	CHAR(20)	This sequence turns all attributes off.
WINDOW	CHAR(20)	This model sequence creates a new scrolling region which is a subset of the lines on the display. It is a model sequence for the same reasons as GOTOXY is. The actual top and bottom lines are substituted at the time the window is created. The model sequence supplies placeholders for the actual values.
SCRUP	CHAR(20)	This sequence causes a window to scroll up one line causing the top line to disappear and a new line to appear on the bottom line of the window.

Control Sequences for IAP

Besides defining the parameters of a CRT, control sequences for invoking IAP functions (ie. NEXTFIELD, PREVFIELD, etc.) must also be defined. These control sequences map a key or keys to a particular IAP function. When that key(s) is pressed IAP will recognize it as a control sequence and execute the appropriate function. To avoid ambiguity, control sequences must begin with a non-printing character. More than one sequence can be defined to invoke a particular function and not all functions have to be defined allowing a subset of functions to be available.

Some CRTs will provide a group of keys (called function keys) which when pressed will send out an escape character followed by one or more characters. Defining these keys as control sequences makes invoking functions easier for the IAP operator. If the CRT does not provide these keys, single control characters make good alternatives. Its also helpful if certain common functions such as NEXTFIELD are mapped to appropriate keys like TAB or RETURN. The BACKSPACE function should be defined to be the same as the operating system's backspace key for consistency.

Since control sequences must use non-printing characters, it would be helpful if these sequences could be entered as printing sequences so that later it would be possible to see what was entered. For that reason the following conventions are used when defining CRT control parameters and control sequences:

\ooo	Where ooo is a 1 to 3 digit octal number representing some ASCII character.
\e	Represents the escape character (\033).
\\	Represents the \ character.

These sequences will be translated to their defined representations prior to being sent to the CRT. Additionally there are similar conventions for representing the placeholders for inserting the x and y screen coordinates used in GOTOXY and the top and bottom lines used in the WINDOW sequence. These formal parameters tell IAP where to insert the actual values when it wants to execute that function. The following conventions are used:

\x	The x coordinate for GOTOXY. This corresponds to the column to postion to.
\y	The y coordinate for GOTOXY. This corresponds to the line to position to
\t	The top line in the WINDOW function.
\b	The bottom line in the WINDOW function.

Function to Control Sequence Mapping

Defining the function to control sequence mapping consists of entering rows into the esc table in the CRT database. This may be done by INSERT statements when bootstrapping IAP or as part of the form crt. The esc table has the following structure:

NAME	CHAR(10)	The CRT the mapping is to be associated with.
FUNC	CHAR(2)	A one or two character abbreviation of the IAP function name.
ESCSEQ	CHAR(20)	The control sequence used to invoke the function.
COMMENTS	CHAR(20)	A description of what key(s) to press to cause ESCSEQ to be generated. This description will be used to drive the IAP function HELPKEYS which describes to the user how to invoke IAP functions.

The allowable values FUNC may take on are contained in the table functions. To list the allowable values while in SQL type SELECT * FROM FUNCTIONS. They are listed here for convienence:

NF	Next Field
NR	Next Record
NB	Next Block
PF	Previous Field
PR	Previous Record
PB	Previous Block
CF	Clear to end of Field
CR	Clear Record
CB	Clear Block
I	Insert record
Q	Query record
U	Update record
D	Delete record
X	Exit IAP
DC	Delete Character (backspace)
H	Help
DA	Display field Attributes
R	Redisplay screen
MR	Move cursor Right
ML	Move cursor Left
CM	Change character Mode (Replace or Insert)
FM	change Functional Mode (Insert or Query/Update)
CA	Clear All
DK	Display function Keys
SB	Scroll forward one Block
P	Print form

Compiling CRT Descriptions

At this point it is assumed that a CRT description has been entered into the CRT database as outlined above. Now it is necessary to compile the CRT description into a format that IAP can read. The output of CRT is a system library file which is read by IAP at startup. The CRT description is compiled by executing the CRT utility as follows:

```
  CRT crtname [user/id]
    or
  CRT crtname database [user/id] -u
```

The first case applies when the crt information has been stored on the crt database and the second case applies when a user database has been used. The user/id parameter is optional depending on whether its a secure database. Crtname is the name entered into the NAME column of the crt table. If no diagnostics come out, then the description was compiled successfully. An output file called crtname.crt will be placed in the system library.

After a description has been compiled successfully, IAP may use that CRT as follows:

```
  IAP formname crtname
     or
  IAP formname
```

The first format is used to specify a particular CRT. The second format is used to specify the default CRT (explained below).

It may be desired to have a particular CRT act as the default. This can be accomplished copying the crt file into a new file called default.crt (which should reside in the system library). In this case the crtname parameter may be omitted when invoking IAP and that CRT will be the default.