ORNL/TM--11639

Energy Division

DE91 000534

SQTTEXT: A TOOL FOR EDITING STRUCTURED QUERY LANGUAGE (SQL) TEXT WITHIN ORACLE SQL*Forms APPLICATIONS

Patricia F. Daugherty
Paul T. Singley

Date Published — August 1990

Prepared for the
Military Traffic Management Command
Falls Church, VA 22041-5050
under Interagency Agreement DOE 1405-1405-A2

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6285
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400

CONTENTS

ABBREVIATIONS, ACRONYMNS, AND INITIALISMS

IAP	Integrated Application Processor
IRDS	Information Resource Dictionary System
MTMC	Military Traffic Management Command
ORACLE	Relational DBMS and family of software products
ORNL	Oak Ridge National Laboratory
RDBMS	Relational Database Management System
SQL	Structured Query Language
SQL*Forms	ORACLE Application Generator
SQL*Plus	ORACLE's Enhanced SQL
WHIST-MOD	Worldwide Household Goods Information System for Transportation-Modernization

ABSTRACT

SQTTEXT is an ORACLE SQL*Forms application that allows a programmer to view and edit all the Structured Query Language (SQL) text for a given application on one screen. This application is an outgrowth of the prototyping of an on-line system dictionary for the Worldwide Household Goods Information System for Transportation-Modernization decision support system being prototyped by the Oak Ridge National Laboratory, but it can be applied to all SQL*Forms software development, debugging, and maintenance. The system dictionary and SQTTEXT were written in version 2.3 of ORACLE's application generator, SQL*Forms.

SQL*Forms greatly simplifies users' access to ORACLE databases, but the design of the tool should be friendlier to those programming, debugging, and maintaining SQL*Forms applications. Because SQL*Forms version 2.3 forces the programmer to view each component of a program through a window specific to that component, it is impossible to get an overview of the whole application at one time. The SQTTEXT application allows experienced ORACLE programmers to increase their productivity by allowing access to all Structured Query Language (SQL) text within a SQL*Forms application via one screen.

The SQTTEXT application is based on the eight ORACLE Integrated Application Processor (IAP) tables that store information about SQL*Forms applications. The SQTTEXT application displays the trigger level, trigger name, and SQL text associated with triggers, and also displays block names, base table names, and the SQL text for WHERE/ORDER BY clauses. This report provides a step-by-step explanation of how to use SQTTEXT, descriptions of the IAP tables, and a listing of the SQTTEXT code.

SQTTEXT is a very powerful tool for experienced ORACLE programmers who have at least a rudimentary understanding of the IAP tables. However, it should not be used by inexperienced users because it is possible to destroy inadvertently the relationships among the eight IAP tables.

1. INTRODUCTION

Software prototypers at Oak Ridge National Laboratory (ORNL) have developed a tool to increase the productivity of programmers who are developing, debugging, and maintaining ORACLE SQL*Forms applications or forms. The tool, called SQTTEXT, is an ORACLE SQL*Forms application developed on a Solbourne workstation running ORACLE version 6.0 and SQL*Forms version 2.3 under SunOS 4.0.3 (a Sun port of BSD Unix). The SQTTEXT application was developed as a part of the prototyping of a system dictionary for the Worldwide Household Goods Information System for Transportation-Modernization (WHIST-MOD) project. The WHIST-MOD system is a decision support system that ORNL is designing and prototyping for the Military Traffic Management Command (MTMC). This system will in aid the movement and storage of the household goods of military service members. SQTTEXT was developed to allow ORNL prototypers to see all the code in a SQL*Forms application in one place. The successful use of SQTTEXT requires a basic understanding of the Integrated Application Processor (IAP) tables underlying SQL*Forms applications.

Although SQL*Forms applications greatly simplify users' access to ORACLE databases, the design of the tool should be friendlier to those programming, debugging, and maintaining SQL*Forms applications. Because SQL*Forms version 2.3 forces the programmer to view each component of a SQL*Forms application, such as a block, field, or trigger, through a window specific to that component, it is impossible to get an overview of the whole application at one time. SQL*Forms windows must also be navigated in an order controlled by the SQL*Forms tool. These windows are arranged in a hierarchical tree structure. This fragmentation of the SQL text forces the form designer/programmer to traverse many windows to compare two triggers and makes it impossible to read all of the SQL text for an application at one time. In addition, there is no quick way to determine whether a trigger is a form-, block-, or field-level trigger. All of these problems caused ORNL developers to seek a tool that would allow them to circumvent some of the awkward features of SQL*Forms. The SQTTEXT application is the result of this search. This application has proved to be very useful in the rapid prototyping environment at ORNL, and it can be used by anyone designing, prototyping, debugging, or maintaining SQL*Forms applications. In addition, it can be useful to anyone who needs an overview of a SQL*Forms application.

The intended audience for this document consists of experienced ORACLE SQL*Forms application designers/programmers. This document contains terms specific to the ORACLE packages and is not intended for persons unfamiliar with ORACLE products. It has four sections and two appendices. The second section contains a discussion of the reasons for developing the tool, a description of the data structures that the tool uses, a brief description of how the tool works, and a brief discussion of when to use and not to use the tool. The third section describes the SQTTEXT screen and explains how to use SQTTEXT. Finally, the fourth section presents a summary. In addition, there is an appendix that describes the tables used in the SQTTEXT application and an appendix that contains a listing of the application code.

2. THE SQTTEXT ENVIRONMENT

This section discusses the SQTTEXT application's role in the WHIST-MOD prototype and the reasons why SQTTEXT was developed. It also describes the tables that SQTTEXT uses and the data SQTTEXT displays. Finally, it discusses situations in which SQTTEXT should and should not be used.

2.1 THE WHIST-MOD SYSTEM DICTIONARY

ORNL is designing and prototyping the WHIST-MOD decision support system for MTMC. This system will aid the movement and storage of the household goods of military service members. By definition, a decision support system should provide a user with access to information they need to help them make intelligent, informed decisions regarding his/her organization's operation and management. To best accomplish this goal, the decision support system must provide relevant, high-quality data in a form that is readily accessible to both inexperienced and seasoned computer users and must be flexible enough to adapt to changing information demands.

Using the ORACLE relational database management system (RDBMS), ORNL has developed an on-line system dictionary that provides WHIST-MOD users with information on how the data are structured in ORACLE database tables and how these data structures are related. It also provides WHIST-MOD programmers with information about the programs that constitute the system. This facility allows users and programmers to access detailed information about the system, including data elements, database tables and views, and applications. This system dictionary fits within the Information Resource Dictionary System (IRDS) standard developed by the National Institute for Standards and Technology (NIST). The IRDS is the basis for a Federal Information Processing Standard (FIPS) that will be used to set the minimum requirements for this type of functionality provided to federally purchased or developed information systems. The SQTTEXT application is an outgrowth of the design and prototyping of the WHIST-MOD system dictionary.

2.2 PROBLEMS WITH SQL*Forms

SQL*Forms is one of several ORACLE products that allow an end user to store, change, retrieve, and work with information in an ORACLE database. The user interface for WHIST-MOD was written in SQL*Forms. To the operator, a SQL*Forms application resembles a paper form, but it provides data validation, navigation, and help. SQL*Forms permits a designer to lay out a form on a screen and modify it interactively.

While SQL*Forms applications greatly simplify users' access to ORACLE databases, the design of the tool should be friendlier to those programming, debugging, and maintaining SQL*Forms applications. Because SQL*Forms version 2.3 forces the programmer to view each component of a SQL*Forms application, such as a block, field, or trigger, through a window specific to that component, it is impossible to get an overview of the whole application at one time. These windows are not the type of windows associated with workstations and graphical user interfaces; they cannot be independently opened, closed, or moved, and they cannot concurrently run programs. Also, SQL*Forms windows must be navigated in an order controlled by the SQL*Forms tool. The windows permit the programmer to only choose an action or display and edit a few lines of code associated with a single trigger. The windows are arranged in a hierarchical tree structure; to move among t the

windows at the same level, the designer must traverse up the tree, across, and then down. For instance, to move between two block-level components, the designer must traverse up to the block level, change blocks, and then traverse down to the destination component. Figure 2.1 illustrates the windows a programmer would see when traversing up or down to the trigger attribute level. This hierarchical structure may require the programmer to go through as many as ten windows to get to and from neighboring components.

In a SQL*Forms application, most of the code that controls the functionality of an application is contained within triggers. Triggers can be associated with a whole form, a block, or a field. Because the SQL text contained within a trigger is displayed in individual windows, there is no way to see all the SQL text for an application at one time. This fragmentation of the SQL text forces the form designer/programmer to traverse many windows to compare two triggers and makes it impossible to read all of the SQL text for an application at one time. In addition, there is no quick way to determine whether a trigger is a form-, block-, or field-level trigger.

Although the problems described above might be rectified in SQL*Forms version 3.0, this version will not be available on the WHIST-MOD target platforms by the time prototyping is completed. The SQTTEXT application has aided ORNL designers during prototyping and the porting of applications among versions of ORACLE and among hardware platforms. This application has rapidly become a very valuable tool for ORNL programmers who use version 2.3 of SQL*Forms. It may also be useful for higher versions.

The SQTTEXT application should be able to run on any version of SQL*Forms that uses the same IAP table structures as those found in version 2.3 of SQL*Forms. Because ORACLE can operate on a wide variety of hardware platforms and operating systems, it is probable that SQTTEXT could be used as a productivity tool for a multiplicity of other projects.

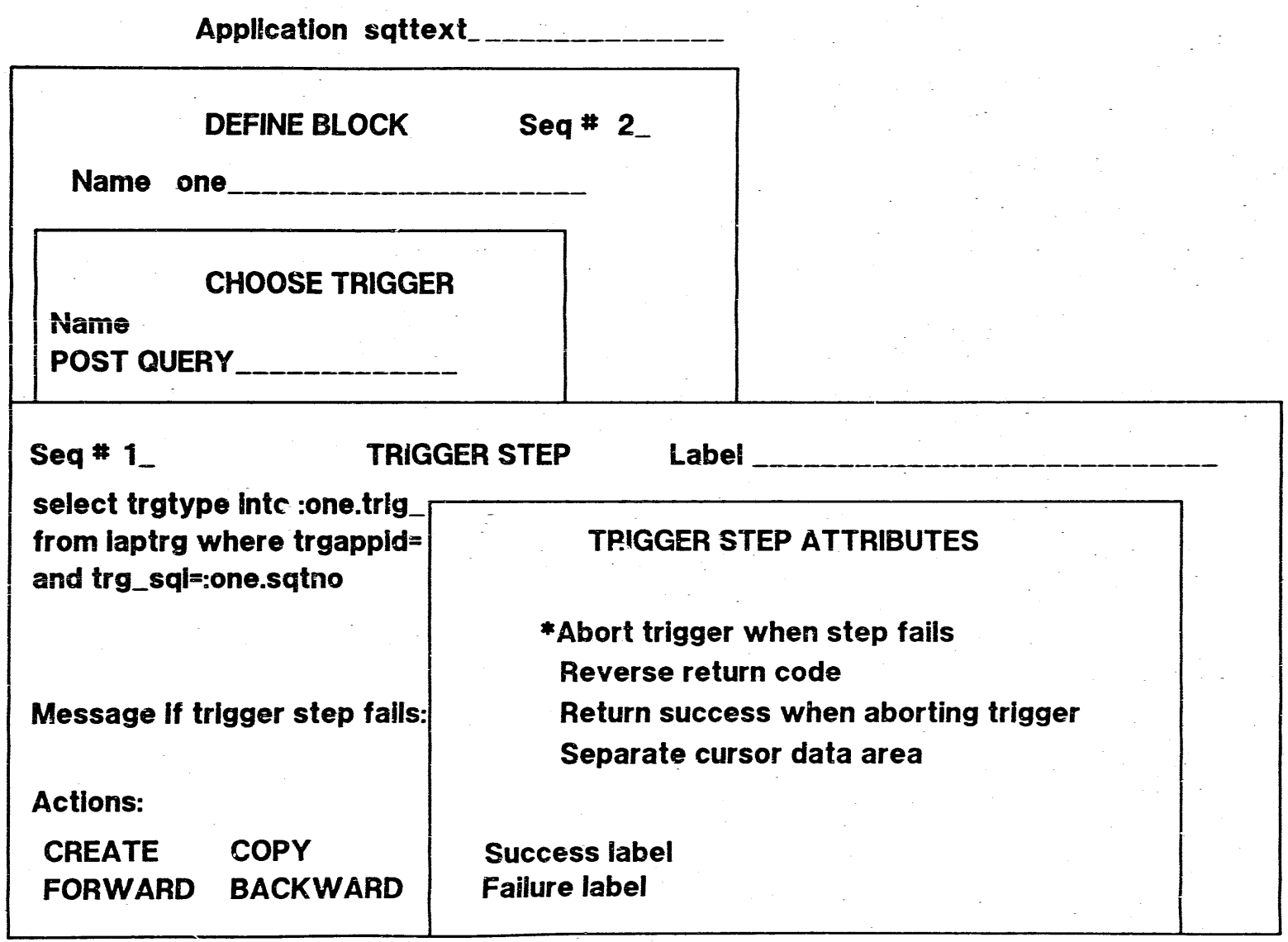

Fig. 2.1. Multiple windows in SQL*Forms.

2.3 THE ORACLE INTERACTIVE APPLICATION PROCESSOR (IAP) TABLES

When the SQL*Forms tool is installed, a set of tables called IAP tables is created. These tables contain information about a form such as the name of the form, and the blocks, comments, fields, screens, SQL text, and triggers within the form. As shown in Table 2.1, there are eight IAP tables in version 2.3 of SQL*Forms:

Table 2.1 IAP tables

Table Name	Type of Data
IAPAPP	General form data
IAPBLK	Block-level data
IAPCOMMENT	Comments
IAPFLD	Field-level data
IAPMAP	Form boilerplate
IAPSQLTXT	Trigger and WHERE/ORDER BY text
IAPTRG	Trigger-step data
IAPTRIGGER	Trigger-level data

Descriptions of these tables can be found in Appendix A. ORACLE documentation states that these tables can be used for report preparation or editing through SQL*Forms. ORACLE reserves the right to change the names or structures of these tables at any time and without notice.

For those willing to take the risk that ORACLE might change these tables, the IAP tables can be a very powerful tool for application development and maintenance. In addition to the uses mentioned by ORACLE, these IAP tables can be used to create, maintain, document (on-line and with hard copy), and delete applications.

When developers create an application in SQL*Forms, they must generate the form. This process produces an executable version of the form, which is stored in a disk file with a name ending in .frm. In addition, the process updates the source code version of the form, which is stored in a disk file with a name ending in .inp. The .inp file for the SQTTEXT application can be found in Appendix B.

In addition to generating the form, the programmer can save the form to the IAP tables. When an application is saved to these tables, SQL*Forms automatically manages the insertion, modification, or deletion of information describing the application in the IAP tables. After the form is saved to the database, the IAP tables can become a very powerful tool for application development and maintenance.

The IAP tables can be viewed and edited through SQL*Plus and can be used by SQL*Forms applications such as SQTTEXT to maintain and modify other SQL*Forms applications. If the IAP tables are used in these ways, the user must keep several points in mind. These tables are linked via a unique identifier that is assigned by the system when a form is first saved. SQL*Forms uses this and other keys to link the IAP tables. If a user-developed application alters these keys, the user must be aware of all relationships that the alteration affects and ensure that those relationships are properly maintained. If the relationships are not maintained, the user risks destroying the form being modified and possibly destroying one or more other forms.

2.4 THE SQTTEXT APPLICATION

SQTTEXT is an ORACLE SQL*Forms application that allows the user to examine and modify via one screen all the SQL text for an application. The primary source of data for SQTTEXT is the IAP table IAPSQLTXT. SQTTEXT also draws information from the IAP tables IAPAPP, IAPBLK, IAPFLD, and IAPTRIGGER. (See Appendix A for the structures and contents of the IAP tables).

The SQL text code that controls the workings of a SQL*Forms application is contained in triggers and WHERE/ORDER BY clauses associated with blocks. Each line of SQL text in an application is stored in the IAP table called IAPSQLTXT in an 80-character column called *SQTTEXT*. In addition to storing the SQL text, the IAPSQLTXT table contains keys that associate each row of SQL text with the form that contains it (the IAPAPP table), the block that contains it (the IAPBLK table), and the trigger that contains it (the IAPTRG table).

The SQTTEXT application takes advantage of the information in IAPSQLTXT not only to display all the SQL text for an application but also to show the user the level (form, block, or trigger) and trigger name for each row of trigger SQL text. SQTTEXT also displays, in a separate portion of the screen, the block name and base table name for the WHERE/ORDER BY SQL text. The user can scroll through both the trigger and the WHERE/ORDER BY text. In addition, the user can query on any field in the form and edit the SQL text in both sections.

2.5 WHEN TO USE SQTTEXT

The SQTTEXT application is most useful when the programmer/designer is debugging or changing an application, but it may be used at any other time when it is desirable to see all of the SQL text in an application on one screen. One of the best ways to use this application is within a window on a workstation. The ORNL developers commonly have the SQTTEXT application running in one window, the application to be modified running in another window, and SQL*Plus running in another window. A fourth window running SQTTEXT against another application is very useful when two applications are being compared.

Because SQTTEXT does not provide any syntactical or content checks on SQL text added via SQTTEXT, editing of SQL text should be performed *ONLY* by programmers who are very comfortable with SQL*Forms and have at least a rudimentary understanding of the IAP tables. Less experienced programmers can safely use SQTTEXT to view an application.

2.6 WHEN *NOT* TO USE SQTTEXT

Because the application does not allow SQL text rows to be added, it cannot be used for building an application from scratch. It should not be used when the programmer cannot recover from changes made through SQTTEXT (e.g., the user is running SQTTEXT against the only copy of an application). It is highly recommended that the user make a copy of the application before attempting to alter it via SQTTEXT.

3. HOW TO USE SQTTEXT

3.1 STARTING AND EXITING SQTTEXT

There are two ways to start SQTTEXT. The first is to run the SQTTEXT application within SQL*Forms. To do this, the user should perform the following steps in the order shown.

• Start SQL*Forms.	Type "sqlforms".
• Choose the application.	Type "sqttext" in the *Name* field in the CHOOSE FORM window.
• Run the application.	Move to the RUN option, then press the ENTER/SELECT key.

An alternative method is to use the RUNFORM application processor. To do this, the user should perform following steps in the order shown.

• Start the application processor.	Type "runform".
• Choose the application.	Type "sqttext".

To exit SQTTEXT from within SQL*Forms, the user should perform the following steps in the order shown.

• Exit the SQTTEXT screen.	Press the EXIT/CANCEL key.
• Exit SQL*Forms.	Press the EXIT/CANCEL key.

To exit SQTTEXT from within RUNFORM, the user should do the following

• Exit the SQTTEXT screen and RUNFORM.	Press the EXIT/CANCEL key.

3.2 CHOOSING AN APPLICATION

The next step after starting the SQTTEXT application is to choose a SQL*Forms application to be viewed and edited. To do this, the user should perform the following steps in the order shown.

• Enter the name of the application to be examined.	Type the application name in the field labelled "Application". If the name cannot be found within the IAPAPP table (e.g., the user has misspelled the application name), the program will wait until either a

correct application name is entered or the EXIT/CANCEL key is pressed.

After a correct application name is entered, SQTTEXT will retrieve all the SQL text associated with the chosen application found in the IAPSQLTXT table. Figure 3.1 shows what would be retrieved if the user were to choose to examine the SQTTEXT application itself. The data retrieved are displayed in two blocks: an upper block containing trigger data and a lower block containing WHERE/ORDER BY–clause data.

3.3 THE UPPER (TRIGGER) BLOCK

The upper block will contain all the SQL text associated with triggers. The *Level* field will contain:

- the word "Form" if the SQL text is associated with a form-level trigger,
- the block name if the SQL text is associated with a block-level trigger, or
- the field name if the SQL text is associated with a field-level trigger.

The *Trigger* field will contain the name of the trigger associated with each row of SQL text. The *SQL Text* field will contain the SQL text for each trigger. If a trigger contains more than one row of SQL text, the *Level* and *Trigger* information will be repeated for each row of associated SQL text.

3.4 THE LOWER (WHERE/ORDER BY) BLOCK

The lower block will contain the SQL text contained in the WHERE/ORDER BY clauses associated with each block. The *Block* field will contain the block name for each line of the WHERE/ORDER BY clause. The *Base Table* field will contain the name of the base table associated with each WHERE/ORDER BY clause. The *SQL Text* field will contain each line of SQL text associated with a WHERE/ORDER BY clause. If a WHERE/ORDER BY clause contains more than one row of SQL text, the *Block* and *Base Table* information will be repeated for each row of associated SQL text.

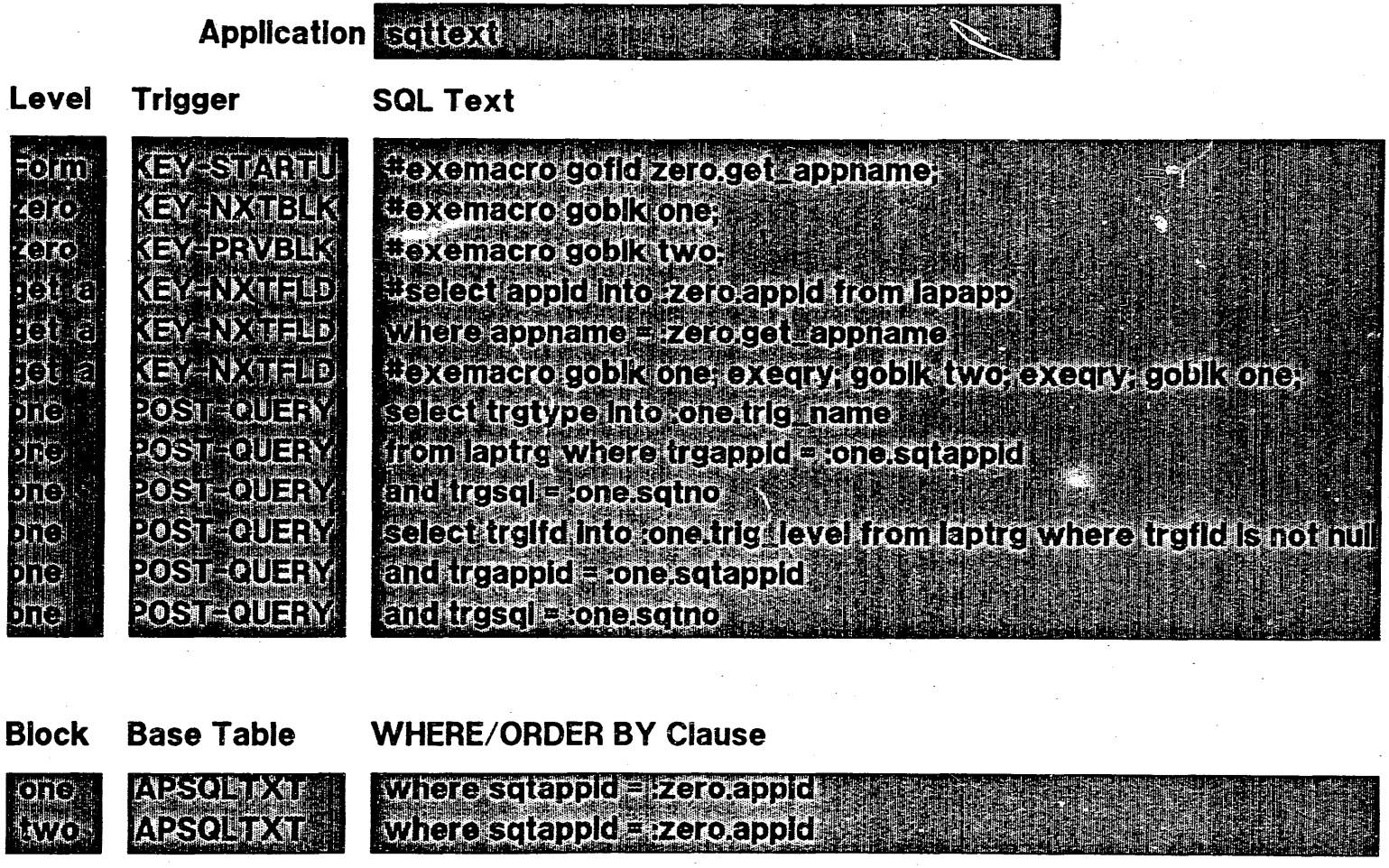

Fig. 3.1. Sample SQTTEXT screen.

3.5 SCREEN NAVIGATION

The following commands can be used to navigate through the SQTTEXT screen.

• Move forward between blocks.	Press the NEXT BLOCK key.
• Move backward between blocks.	Press the PREVIOUS BLOCK key.
• Scroll forward through a scrolling block.	Press the NEXT RECORD key.
• Scroll backward through a scrolling block.	Press the PREVIOUS RECORD key.
• Scroll through the SQL text.	Press the left and right arrow keys.

3.6 VIEWING SQL TEXT

Entering the name of the SQL*Forms application in the field labelled "Application" causes all the SQL text for an application to be displayed on the screen in two scrolling blocks. To view another application, the user should perform the following steps in the order shown.

• Move to the field labelled "Application".	Press the NEXT BLOCK key until the cursor returns to the field labelled "Application".
• Choose a new application.	Type the name of the new application to be viewed.

3.7 SEARCHING FOR SPECIFIC SQL TEXT

To search for particular SQL text, the user should perform the following steps in the order shown.

• Enter the query mode.	Press the ENTER QUERY key.
• Enter the text string to be searched for.	Type the text string that will serve as the object of the search into the *Level*, *Trigger*, and/or *SQL Text* fields associated with the upper block and/or the *Block*, *Base Table*, and *SQL Text* fields associated with the lower block.
• Start the query.	Press the EXECUTE QUERY key.

The following are examples of how to perform some typical queries within the SQTTEXT application.

• Display all SQL text for the KEY-STARTUP trigger.	Type "KEY-STARTUP" in the *Trigger* field.
• Display all the SQL text containing the string "goblk" in the upper block.	Type "%goblk%" in the *SQL Text* field in the upper block.

(The "%" symbols indicate that the string "goblk" can occur at any position within a column).

- Display the contents of the *Base Table* and *SQL Text* fields. — Type "one" in the *Block* field on the lower block.
- Display all the SQL text containing "exemacro" in the upper block. — Type "%exemacro%" in the *SQL Text* field in the upper block. This is can be used to check for the proper punctuation for "exemacro" commands.

3.8 EDITING SQL TEXT

To change the SQL text displayed on the screen, the user should perform the following steps in the order shown.

- Enter the new SQL text. — Type the desired text in the *SQL Text* fields.
- Store the change in the database. — Press the COMMIT key.

The user cannot add or delete SQL text rows . Also, the user cannot edit the *Level*, *Trigger*, *Block*, or *Base Table* fields, because they serve as keys to other IAP tables.

If the user changes any SQL text through the SQTTEXT application, a new .inp file must be generated. For this reason, if the user intends to edit the SQL text instead of just view it, SQTTEXT should be run from within SQL*Forms. To generate a new .inp file from within SQL*Forms, the user should perform the following steps in the order shown.

- Exit the SQTTEXT screen. — Press the EXIT/CANCEL key. Exiting this screen will return the user to the CHOOSE FORM screen. The name of the application being examined will be displayed in the *Name* field.
- Choose the GENERATE option. — Move to the GENERATE option and press the ACCEPT key.
- Accept the name of the file to be generated. — Press the ACCEPT key.
- Exit SQL*Forms. — Press the EXIT/CANCEL key.

4. CONCLUSIONS

The SQTTEXT application can be used to develop, debug, and maintain SQL*Forms applications. It aids these processes by making it possible to display and edit on one screen all the SQL text associated with an application – a capability not provided within SQL*Forms. SQTTEXT provides this capability by using information stored in the ORACLE IAP tables.

SQTTEXT was developed to aid the design and prototyping of the WHIST-MOD system dictionary, but it can be applied to any SQL*Forms programming efforts. It is a powerful tool in the hands of experienced ORACLE programmers. SQTTEXT should be used for editing *ONLY* by programmers who are comfortable with SQL*Forms and familiar with the ORACLE IAP tables. Less experienced programmers can safely use SQTTEXT to view an application.

APPENDIX A: The IAP Tables

A.1 IAPAPP TABLE

This table contains application identification information.

Name	Information	Null?	Type
APPID	Form ID	NOT NULL	NUMBER
APPOWNER	Form creator		CHAR(30)
APPNAME	Form name	NOT NULL	CHAR(30)
APPTITLE	Form title		CHAR(80)
APPWKSIZE	Context size		NUMBER
APPVAUNIT	Validation unit		CHAR(6)

A.2 IAPBLK TABLE

This table contains block identification information and attributes.

Name	Information	Null?	Type
BLKAPPID	Link to form ID	NOT NULL	NUMBER
BLKNAME	Block name	NOT NULL	CHAR(30)
BLKDESC	Block description		CHAR(60)
BLKHIDE	Display in block menu		CHAR(1)
BLKSEQ	Block sequence number		NUMBER
BLKUNQKEY	Check for unique key		CHAR(1)
BLKCTRL	Whether block has base table		CHAR(1)
BLKTOWNER	Table owner		CHAR(30)
BLKTNAME	Table name		CHAR(30)
BLKNOREC	Rows displayed		NUMBER
BLKNOBUF	Rows buffered		NUMBER
BLKBLIN	Base line		NUMBER
BLKLNRC	Lines per row		NUMBER
BLKOBYSQL	Link to IAPSQLTXT table		NUMBER

A.3 IAPCOMMENT TABLE

This table contains all of the comment text for an application.

Name	Information	Null?	Type
CMTAPPID	Link to form ID	NOT NULL	NUMBER
CMTBLK	Block name		CHAR(30)
CMTFLD	Field name		CHAR(30)
CMTTRGTYP	Trigger name		CHAR(30)
CMTTRGSEQ	Trigger step number		NUMBER
CMTLINE	Line number		NUMBER
CMTTEXT	Comment text		CHAR(80)

A.4 IAPFLD TABLE

This table contains information on fields and their attributes.

Name	Information	Null?	Type
FLDAPPID	Link to form ID	NOT NULL	NUMBER
FLDBLK	Block name	NOT NULL	CHAR(30)
FLDNAME	Field name	NOT NULL	CHAR(30)
FLDSEQ	Sequence number		NUMBER
FLDTYPE	Data type		CHAR(7)
FLDLEN	Field length		NUMBER
FLDDLEN	Display length		NUMBER
FLDQLEN	Query length		NUMBER
FLDBTAB	Database field		CHAR(1)
FLDKEY	Primary key		CHAR(1)
FLDCKBLK	Copy field block		CHAR(30)
FLDCKFLD	Copy field		CHAR(30)
FLDDFLT	Default value		CHAR(80)
FLDDISP	Field is displayed on screen		CHAR(1)
FLDPAGE	Screen page		NUMBER
FLDLINE	Screen line		NUMBER
FLDCOL	Screen column		NUMBER
FLDPROMPT	Prompt		CHAR(80)
FLDPRABOV	Prompt above field		CHAR(1)
FLDPRRPT	Prompt beside field		CHAR(1)
FLDENTER	Input allowed		CHAR(1)

FLDQUERY	Query allowed	CHAR(1)
FLDUPDATE	Update allowed	CHAR(1)
FLDUPDNUL	Update if null	CHAR(1)
FLDMAND	Mandatory	CHAR(1)
FLDFIXED	Fixed length	CHAR(1)
FLDSKIP	Autoskip	CHAR(1)
FLDHIDE	Field is hidden	CHAR(1)
FLDAUTOHLP	Automatic help	CHAR(1)
FLDUPPER	Uppercase	CHAR(1)
FLDLOVT	Table for list of values	CHAR(61)
FLDLOVC	Column for list of values	CHAR(30)
FLDLOW	Low range value	CHAR(30)
FLDHI	High range value	CHAR(30)
FLDHELP	Help text	CHAR(80)

A.5 IAPMAP TABLE

This table stores all the boilerplate and graphics for a form.

Name	Information	Null?	Type
MAPAPPID	Link to form ID	NOT NULL	NUMBER
MAPPAGE	Page number	NOT NULL	NUMBER
MAPLINE	Line number		NUMBER
MAPGRPH	Text or graphics characters		CHAR(1)
MAPTEXT	Boilerplate text		CHAR(132)

A.6 IAPSQLTXT TABLE

This table contains the text of all SQL statements for a form, including the default WHERE/ORDER BY clauses.

Name	Information	Null?	Type
SQTAPPID	Link to form ID	NOT NULL	NUMBER
SQTNO	SQL text identifier	NOT NULL	NUMBER
SQTLINE	Line number		NUMBER
SQTTEXT	SQL text		CHAR(80)

A.7 IAPTRG TABLE

This table contains information on the location and attributes of trigger steps in an application.

Name	Information	Null?	Type
TRGAPPID	Link to form ID	NOT NULL	NUMBER
TRGBLK	Block name		CHAR(30)
TRGFLD	Field name		CHAR(30)
TRGTYPE	Trigger name	NOT NULL	CHAR(30)
TRGSEQ	Trigger step number		NUMBER
TRGLABEL	Trigger step label		CHAR(30)
TRGSQL	Link to IAPSQLTXT table		NUMBER
TRGCURS	Separate cursor		CHAR(1)
TRGMVE	Abort when step fails		CHAR(1)
TRGINV	Reverse return code		CHAR(1)
TRGROLL	Return success if aborted		CHAR(1)
TRGSLAB	Success label		CHAR(30)
TRGFLAB	Failure label		CHAR(30)
TRGMSG	Message if step fails		CHAR(80)

A.8 IAPTRIGGER TABLE

This table contains information on the location of triggers in an application.

Name	Information	Null?	Type
TRIGAPPID	Link to form ID	NOT NULL	NUMBER
TRIGBLK	Block name		CHAR(30)
TRIGFLD	Field name		CHAR(30)
TRIGTYPE	Trigger name	NOT NULL	CHAR(30)
TRIGDESC	Description		CHAR(20)
TRIGHIDE	Include in SHOW KEYS		CHAR(1)

APPENDIX B: The .inp File

```
; Generated by SQL*Forms Version 3682.32.9 on Mon Aug 13 15:00:38 1990.
; Application owner is TRICIA.  Application name is sqttext.
; (Application ID is 43)
; ------------------------------------------------
;;This application allows a designer to view all the SQL text associated with
;;an application on one screen.
;Application Title :
SQTTEXT
;ORACLE workspace size :

;;This trigger gets the application name from the user.
;Block name / Description :
**KEY-STARTUP
;SQL>
#exemacro gofld zero.get_appname;

;Message if value not found :

;Must value exist Y/N :
Y
;;This block contains the field where the user enters the name of the
;;application to be examined.
*;Block name / Description :
zero/
;Table name :
*
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
1 / 1
;;This trigger moves the cursor to the next block when the NXTBLK key is
;;pressed.
;Field name :
*KEY-NXTBLK
;SQL>
#exemacro goblk one;

;Message if value not found :

;Must value exist Y/N :
Y
;;This trigger moves the cursor to the previous block when the PRVBLK key is
;;pressed.
;Field name :
*KEY-PRVBLK
;SQL>
#exemacro goblk two;
```

;Message if value not found :

;Must value exist Y/N :
Y
;Field name :
blkobysql
;Type of field :
NUMBER
;Length of field / Display length / Query length :
5 / 5 / 5
;Is this field in the base table Y/N :
N
;Default value :

;Page :

;SQL>

;Field name :
appid
;Type of field :
NUMBER
;Length of field / Display length / Query length :
5 / 5 / 5
;Is this field in the base table Y/N :
N
;Default value :

;Page :

;SQL>

;;This field serves as the place for the user to enter the name of the
;;application to be examined.
;Field name :
get_appname
;Type of field :
CHAR
;Length of field / Display length / Query length :
30 / 30 / 30
;Is this field in the base table Y/N :
N
;Default value :

;Page :
1
;Line :
1

```
;Column :
32
;Prompt :

;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;;This trigger finds the IAP application ID number for the application name
;;entered by the user.
;SQL>
**KEY-NXTFLD
/
;SQL>
select appid into :zero.appid from iapapp
where appname = :zero.get_appname
/
;Message if value not found :

;Must value exist Y/N :
Y
#exemacro goblk one; exeqry; goblk two; exeqry; goblk one;

;Message if value not found :

;Must value exist Y/N :
Y
;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 :

;Lowest value :

;Highest value :

;Field name :

;;This block contains the trigger level, trigger name, and SQL text associated
;;with each trigger.
;Block name / Description :
*one/one
;Enter default WHERE and ORDER BY clause :
```

```
where sqtappid = :zero.appid

;Table name :
IAPSQLTXT
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
12 / 12
;Base crt line ?
4
;How many physical lines per record ?
1
;;This trigger retrieves the trigger level and trigger name for each line of
;;SQL text associated with a trigger.
;Field name :
*POST-QUERY
;SQL>
select trgtype into :one.trig_name
from iaptrg where trgappid=:one.sqtappid
and trgsql=:one.sqtno
/
;Message if value not found :

;Must value exist Y/N :
Y
select trgfld into :one.trig_level from iaptrg where trgfld is not null
and trgappid = :one.sqtappid
and trgsql = :one.sqtno
/
;Message if value not found :
$Done $
;Must value exist Y/N :
N
select trgblk into :one.trig_level from iaptrg where trgblk is not null
and trgappid = :one.sqtappid
and trgsql = :one.sqtno
/
;Message if value not found :
$Done $
;Must value exist Y/N :
N
select 'Form' into :one.trig_level from iaptrg where trgblk is null
and trgfld is null and trgappid = :one.sqtappid
and trgsql = :one.sqtno
/
;Message if value not found :

;Must value exist Y/N :
```

```
Y
$Done
#exemacro null;

;Message if value not found :

;Must value exist Y/N :
Y
;;This trigger moves the cursor to the next block when the NXTBLK key is
;;pressed.
;Field name :
*KEY-NXTBLK
;SQL>
#exemacro goblk two;

;Message if value not found :

;Must value exist Y/N :
Y
;;This trigger moves the cursor to the next record (scrolls down one record)
;;when the NXTREC key is pressed.
;Field name :
*KEY-NXTREC
;SQL>
#exemacro nxtrec;
/
;Message if value not found :

;Must value exist Y/N :
N
#exemacro case one.sqttext is
when '' then prvrec;
end case;

;Message if value not found :

;Must value exist Y/N :
Y
;;This trigger moves the cursor to the previous block when the PRVBLK key is
;;pressed.
;Field name :
*KEY-PRVBLK
;SQL>
#exemacro goblk zero;

;Message if value not found :

;Must value exist Y/N :
```

Y
;;This trigger moves the cursor to the previous record (scrolls up one record)
;;when the PRVREC key is pressed.
;Field name :
*KEY-PRVREC
;SQL>
#exemacro prvrec;

;Message if value not found :

;Must value exist Y/N :
Y
;Field name :
sqtno
;Type of field :
NUMBER
;Length of field / Display length / Query length :
5 / 5 / 5
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :

;SQL>

;Field name :
trgfld
;Type of field :
CHAR
;Length of field / Display length / Query length :
30 / 30 / 30
;Is this field in the base table Y/N :
N
;Default value :

;Page :

;SQL>

;Field name :
appname
;Type of field :
CHAR
;Length of field / Display length / Query length :
30 / 30 / 30

```
;Is this field in the base table Y/N :
N
;Default value :

;Page :

;SQL>

;;This field indicates the level of trigger associated with each line of SQL
;;text:
;;      1. Form level trigger  - field contains the word 'Form'
;;      2. Block level trigger - field contains block name
;;      3. Field level trigger - field contains field name
;Field name :
trig_level
;Type of field :
CHAR
;Length of field / Display length / Query length :
30 / 5 / 30
;Is this field in the base table Y/N :
N
;Default value :

;Page :
1
;Line :
1
;Column :
1
;Prompt :

;Allow field to be entered Y/N :
*N
;Allow entry of query condition Y/N :
Y
;SQL>

;;This field displays the name of the trigger associated with each line of SQL
;;text.
;Field name :
trig_name
;Type of field :
CHAR
;Length of field / Display length / Query length :
30 / 10 / 30
;Is this field in the base table Y/N :
N
;Default value :
```

;Page :
1
;Line :
1
;Column :
7
;Prompt :

;Allow field to be entered Y/N :
*N
;Allow entry of query condition Y/N :
Y
;SQL>

;;This field contains the SQL text associated with each trigger.
;Field name :
sqttext
;Type of field :
CHAR
;Length of field / Display length / Query length :
80 / 63 / 80
;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 :
1
;Column :
18
;Prompt :

;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 value for : SQTTEXT
;Lowest value :

;Highest value :

;Field name :
sqtappid
;Type of field :
CHAR
;Length of field / Display length / Query length :
3 / 3 / 3
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :

;SQL>

;Field name :

;;This block contains the block name, base table name, and SQL text associated
;;with each WHERE/ORDER BY clause.
;Block name / Description :
*two/
;Enter default WHERE and ORDER BY clause :
where sqtappid = :zero.appid

;Table name :
IAPSQLTXT
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
4 / 4
;Base crt line ?
18
;How many physical lines per record ?
1
;;This trigger retrieves the block name and trigger name.
;Field name :
*POST-QUERY
;SQL>
select blkname into :two.block_name
from iapblk
```

```
where blkappid = :two.sqtappid
and blkobysql = :two.sqtno
/
;Message if value not found :

;Must value exist Y/N :
Y
select blktname into :two.base_table
from iapblk
where blkappid = :two.sqtappid
and blkobysql = :two.sqtno

;Message if value not found :

;Must value exist Y/N :
Y
;;This trigger moves the cursor to the next block when the NXTBLK key is
;;pressed.
;Field name :
*KEY-NXTBLK
;SQL>
#exemacro goblk zero;

;Message if value not found :

;Must value exist Y/N :
Y
;;This trigger moves the cursor to the next record (scrolls down one record)
;;when the NXTREC key is pressed.
;Field name :
*KEY-NXTREC
;SQL>
#exemacro nxtrec;
/
;Message if value not found :

;Must value exist Y/N :
N
#exemacro case two.sqttext is
when '' then prvrec;
end case;

;Message if value not found :

;Must value exist Y/N :
Y
;;This trigger moves the cursor to the previous block when the PRVBLK key is
;;pressed.
```

```
;Field name :
*KEY-PRVBLK
;SQL>
#exemacro goblk one;

;Message if value not found :

;Must value exist Y/N :
Y
;;This trigger moves the cursor to the previous record (scrolls up one record)
;;when the PRVREC key is pressed.
;Field name :
*KEY-PRVREC
;SQL>
#exemacro prvrec;

;Message if value not found :

;Must value exist Y/N :
Y
;;This trigger displays the block name for each line of SQL text in the
;;WHERE/ORDER BY clause.
;Field name :
block_name
;Type of field :
CHAR
;Length of field / Display length / Query length :
30 / 5 / 30
;Is this field in the base table Y/N :
N
;Default value :

;Page :
1
;Line :
1
;Column :
1
;Prompt :

;Allow field to be entered Y/N :
*N
;Allow entry of query condition Y/N :
Y
;SQL>

;Field name :
blkname
```

;Type of field :
CHAR
;Length of field / Display length / Query length :
30 / 30 / 30
;Is this field in the base table Y/N :
N
;Default value :

;Page :

;SQL>

;Field name :
sqtno
;Type of field :
CHAR
;Length of field / Display length / Query length :
5 / 5 / 5
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :

;SQL>

;Field name :
blktname
;Type of field :
CHAR
;Length of field / Display length / Query length :
30 / 30 / 30
;Is this field in the base table Y/N :
N
;Default value :

;Page :

;SQL>

;;This field displays the name of the base table for each line of SQL text
;;in the WHERE/ORDER BY clause.
;Field name :
base_table
;Type of field :
CHAR

```
;Length of field / Display length / Query length :
30 / 10 / 30
;Is this field in the base table Y/N :
N
;Default value :

;Page :
1
;Line :
1
;Column :
7
;Prompt :

;Allow field to be entered Y/N :
*N
;Allow entry of query condition Y/N :
Y
;SQL>

;;This field contains the SQL text contained within each WHERE/ORDER BY
;;clause.
;Field name :
sqttext
;Type of field :
CHAR
;Length of field / Display length / Query length :
80 / 63 / 80
;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 :
1
;Column :
18
;Prompt :

;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 :

;Lowest value :

;Highest value :

;Field name :
sqtappid
;Type of field :
CHAR
;Length of field / Display length / Query length :
3 / 3 / 3
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :

;SQL>

;Field name :

;Block name / Description :

                Application
%LINE
3
Level Trigger    SQL Text
%LINE
17
Block Base Table WHERE/ORDER BY Clause
%END
```

-END-

DATE FILMED

11/05/90