DEPARTMENT OF THE INTERIOR

U. S. GEOLOGICAL SURVEY

User's guide to ESM: a database for strong-motion information

Version 1

by
April Converse[1]
01 March 87

Open-File Report 87-160

This report is preliminary and has not been reviewed for conformity with U. S. Geological Survey editorial standards. Any use of trade names is for descriptive purposes only and does not imply endorsement by the USGS.

[1] U.S. Geological Survey, Menlo Park, California, USA.

Open-File Reports are Distributed by:

Open-File Services Section
Branch of Distribution
U. S. Geological Survey
Box 25425, Federal Center
Denver, Colorado 80225

(303) 234-5888

ESM
Table of Contents

Table of Contents, Continued

PREFACE

This report describes ESM, a database that contains descriptions of the strong-motion records acquired from the permanent network of strong-motion instruments maintained by the U.S. Geological Survey (USGS). The database also contains information about the earthquakes that produced the records, information about the recording sites, and information about the recording instruments.

The report is primarily a user's guide for members of the USGS, but it is published as an open file report to inform organizations outside the USGS about the database. The report introduces new users to the database and serves as a reference manual for experienced users.

The database content and support software are rather rudimentary at present. They may evolve substantially in the future and, if so, new versions of this report will become available as the database development progresses.

01 March 1987

April Converse
U. S. Geological Survey
Mail Stop 977
345 Middlefield Road
Menlo Park, CA, 94025

telephone: (415) 323-8111
extension 2881

or FTS 467-2881

ACKNOWLEDGMENTS

The database described in this report uses the INGRES database management and support software provided by Relational Technology, Inc; 1080 Marina Village Parkway; Alameda, California 94501. Portions of Chapters 7 and 8 and most of Appendixes A, B, and C of this report were copied or paraphrased from the INGRES documentation with kind permission from Relational Technology.

Additional software and general guidance was provided by Jon Stedman of Software Mechanics; 712 Fulton Street; Redwood City, California 94062.

CHAPTER 1

Introduction

1.1 Overview

The ESM database provides descriptions of strong-motion accelerogram records and the circumstances in which the records were recorded. The accelerograms described in the database are those that have been retrieved from the permanent network of strong-motion recorders operated by the U.S. Geological Survey (USGS). The database also includes information about the level of processing and analysis that has been performed on the records, information about each earthquake that triggered recorded motion, information about the sites at which the motion was recorded, and information about the recording instruments. The database information is updated by members of the USGS as new information is gathered.

The database is organized, maintained, and accessed using INGRES, a general-purpose relational database management software package provided by Relational Technology, Inc. Most of the information in the ESM database was transfered from its predecessor, the SMIRS database, which was dismantled in March 1986.

This report describes the contents of the database and methods that can be used to retrieve information from it. A separate report, the "ESM Update, Maintenance, and Development Guide", is available to those who are involved in changing the database contents.

Among other things, ESM is used by members of the USGS to gather and distribute information about USGS strong-motion records to other organizations. To request information from outside the Survey, write to:

ES&G Data Project (ESM inquiry)
US Geological Survey, Mail Stop 977
345 Middlefield Road
Menlo Park, CA 94025

or telephone Chuck Mueller, John Watson, April Converse, or Gerald Brady at (415) 323-8111.

People who would prefer to browse through the ESM database themselves rather than mailing in their requests may visit the Menlo Park USGS center and use ESM directly. To arrange a visit, write or telephone the BESG data project at the address or telephone number shown above.

1.2 Topics of Database Information

The ESM information is arranged in 12 tables. The tables are summarized in chapter 2 and are described in detail in chapter 3. The name of each table is meant to indicate the sort of item that is described in each of its rows. The table names are, or are abbreviations of: *events*, *records*, record-*traces*, record-*analyses*, *stations*, *arrays*-of-stations, other-station-*numbers*, recording-*instruments*, *transducers*-attached-to-a-recorder, transducer-*calibrations*, *annotation*, and *doc*umentation. Throughout this report, italics (*italics*) are used to denote table names and other words or phrases that might occur in a dialog between a user and the computer.

A seismic "event" is an earthquake, or a similar earth vibration caused, for example, by an explosion. Instruments in the USGS strong-motion network record the acceleration (or in some cases the displacement) they undergo during an event. These instruments do not record continuously; they are triggered by motion that is strong enough to be of interest in the study of the behavior of structures during seismic events. In the database, each row in the *events* table describes a single event.

A "transducer" is the device that detects acceleration and transmits the signal to a recorder. Some types of strong-motion instruments have transducers that are situated in locations remote from their recorder, but most of the instruments in the network at the present time consist of a recorder and its transducers in a single unit. Most transducers are placed in groups of three to record orthogonal components of motion. In the database, each row in the *transducers* table describes a single transducer.

A "recorder" is the "instrument" that records the acceleration on paper, film, magnetic tape, or computer memory. In the database, each row in the recording-*instruments* table describes a single recorder.

A "record" is the paper, film, magnetic tape or computer disk on which an event has been recorded. The physical record made by remote transducers may contain traces from several locations. In the database, each row in the *records* table describes a single record.

Several stages of data "analysis" are routinely performed on the digitized traces from significant records. The

unprocessed digitized traces and the results from the various processing steps are not stored within the database, but their existence and availablilty are indicated in the *analyses* table. Each row in the *analyses* table describes one set of data processing procedures: the digitization, computer processing, and archival of all the data traces on a single record. Since a record may have been digitized more than once and the data processing may have been performed more than once with different methods, there may be several rows in the *analyses* table for any one record.

A recording "station" is an entire location or general site containing strong-motion recording instruments. Each station in the network has been assigned a unique four-digit station number, and many of the stations have also been assigned a three-letter alphabetic code. In the database, each row in the *stations* table describes either a single station, or if the station is composed of several substations (few of them are), a single substation.

A "substation" is a subdivision of a station. Normally, it is a structure or site adjacent to the parent station. It may be anything that houses strong-motion instruments: an instrumented building, bridge, or dam; or a lone instrument shelter. Most stations consist of a single substation, but those stations having instruments in several structures with different characteristics are said to be composed of several substations.

An "array" consists of a group of recorders designed to provide data on a specific aspect of seismic engineering interest. An array may consist of closely situated recorders, all of which may be considered to be in the same station, or an array may span distances of up to several hundred kilometers, comprising many separate stations. The instruments in an array may or may not be interconnected for synchronous operation. In the database, each row in the *arrays* table indicates a single station within an array.

1.3 Column Notation and Range Variables

In QUEL, the INGRES query language, a specific column of a specific table is denoted with an expression like *e.edate*. A table identifier (e.g., *e*) called a "range variable" is followed by a column name (e.g., *edate*) and the two are separated by a dot. The range variable may be a table name (e.g., *events*) or it may be an arbitrary, user-defined name (such as *e*) that refers to a specific table.

In addition to identifying a specific table, range variables are used within many QUEL statements rather like subscripts in mathematical notation. They are used as row

markers that range over the table specified in a *range* statement. For example, the QUEL range statement *range of e is events* defines a range variable *e* that could subsequently be used in the following QUEL *retrieve* statement:

```
retrieve (e.all) where e.edate >= "01-jan-1984"
                   and e.edate <  "01-jan-1985"
```

This *retrieve* statement requests INGRES to display all rows in the *events* table that contain an event date in 1984. In processing the *retrieve*, INGRES would consider each row in the events table in turn to see whether it satisfied the qualification following the word *where*. If there were no *where* clause in the statement (i.e., *retrieve (e.all)* and nothing else), then the entire *events* table would be displayed.

Although range variables are used primarily in the QUEL language described in Chapter 8, Chapter 10, and Appendix B, they are also used elsewhere in this report to denote the various tables and columns in a syntax consistent with that used in QUEL.

The ESM table names and column names were assigned when the database tables were designed. Row marker names, however, are assigned by the QUEL user as needed with the use of a *range* statement. Although the user may choose any name as a range variable name, one name is used for each table thoughout this report. For brevity, the range variable names used in this report are shorter than their corresponding table names. These are the range statements that are assumed to be in effect throughout this report:

```
range of  e  is events
range of  r  is records
range of  rt is traces
range of  ra is analyses
range of  s  is stations
range of  s  is stations
range of  array is arrays
range of  n  is numbers
range of  i  is instruments
range of  t  is transducers
range of  c  is calibrations
range of  a  is annotation
range of doc is doc
range of table is  <any table in general>
```

QUEL users must take care not to use two range variable names that refer to the same table as though they were identical. The are not identical, for each range variable ranges over a table independently of any other range variables associated with that table. Use of several range variables that refer to the same table is necessary in some complex queries, but such use often results in a very serious error which is

called a "disjoint query" in the INGRES documentation. For more information, refer to the discussion of range variables and "disjoint queries" in Chapter 8, section 8.2.2.

1.4 Join Fields

"Join fields" are used in relational database arrangements to serve as the link between parent and subordinate information that resides in two separate tables. The join fields are similar columns that occur in two tables and allow a row in one table to be matched to a row or several rows in another table.

Consider, for example, the strong-motion records that are described in ESM. On each physical record are 3 or more data traces and each record may have been processed several times with different methods or, as is usually the case, not at all. To allow any number of traces and any number of analyses to be associated with any one record, the record description information is contained in three separate tables in ESM. Each record description in the database consists of one row from the *records* table that contains information about the record as a whole; three or more rows from the record-*traces* table, each of which contain information pertaining to a single trace on the record; and possibly several (but usually no) rows from the *analyses* table, each of which contains information about the digitization and data processing done on the record. The association between a row in the *records* table and the related rows in the record-*traces* table (or the record-*analyses* table) is made by "joining" the two tables. The "*edate*", "*ed_flag*", "*eid*", "*rtype*", and "*rsn*" columns are joined, and all these columns must occur in the record-*traces* table (and the record-*analyses* table) in addition to the *records* table.

Special join fields named "*tag*" are used in ESM as links from various data tables to the *annotation* table. The tags are integer numbers that have no meaning or purpose other than this joining function. (Note that the *records*, *traces*, and *analyses* tables could have been arranged to use the tag fields for their joins also. Future versions of the database may be rearranged to do so.)

1.5 Annotation and Documentation within the Database

Comments, remarks, and other free-form text may be linked to any row in the *events*, *stations*, *records* and *instruments* tables. The text is kept in a separate table, the *annotation* table; it contains one line of text in each row. Several, many, or no lines of text from the *annotation* table may be linked to any one row in the *events*, *stations*, *records* or *instruments* tables. The *tag* field in the data table is joined to the *parent_*

tag field in the *annotation* table to link a row in the data table to the appropriate rows in the *annotation* table.

The "*type*" column in the annotation table is used to indicate which type of annotation is given in each row. The four most commonly occurring types of annotation are names, references, comments, and x-comments (*type* = "*n*", "*r*", "*c*" and "*x*"). Names include event names and station names; references indicate published reports; comments contain text that might appear in a printed report; and x-comments are comments that contain temporary or messy notes that should be excluded from a printed report. Another type of annotation, "*j*" for junk, includes problems that should be resolved as the database information is cleaned up, things like old-SMIRS information that the old-SMIRS-to-ESM translating program couldn't fit well into the ESM tables and that will require manual cleanup attention.

The *doc*umentation table is similar to the *annotation* table in that it too contains one line of text in each of its rows. The information in the *doc*umentation table is more general than that in the *annotation* table, however. It provides general announcements about the database and a dictionary of all the codes used in ESM.

1.6 Storage Formats for Data Types

Values in any one column are all stored in the same format. The following formats are used for the various columns in ESM tables:

notation	type	range
c1 - c255	character	A string of 1 to 255 characters. Blanks are ignored in character string comparisons.
text(1) - text(2000)	text	A string of 1 to 2000 ASCII characters. Blanks are significant in text string comparisons.
i1	1-byte integer	-127 to +127
i2	2-byte integer	-32767 to +32767
i4	4-byte integer	- to +2,147,483,647
f4	4-byte floating	- to + 10**38, with 7-digit precision
date	date (12 bytes)	1-jan-1582 to 31-dec-2382.

There are two other INGRES formats too, f8 and money, but they won't be used in the ESM database.

The table above was copied from Section 1.2.7 of the "INGRES Reference Manual".

1.7 Null Values

When information is unknown or has not been entered into the database it usually appears as a blank (for character data) or a zero (for numeric data). Latitudes, longitudes, and dates, however, use special values to indicate "not known".

1.7.1 Latitudes and Longitudes

The value *-303.0303* appears in latitude or longitude colums when necessary to indicate that the actual value is unknown.

1.7.2 Dates

The dates in ESM, event dates for example, are represented in ESM with two fields: an INGRES date field plus a one-character flag. The flag is blank (" "), a number sign ("#") or a dash ("-"). The "#" will appear in the flag alongside dates that are not precisely known; comments in the annotation table often indicate the time interval in which the actual date is thought to have occurred. The "-" occurs in the flag column alongside dates that are not known at all and the corresponding date fields contain *03-mar-1630*, or for removal dates, *03-mar-2030*.

1.8 Values that Do Not Fit in the Appropriate Column

Character or text values sometimes occur that are too long to fit in their field. For these values, as many leading characters as will fit, less one, are stored in the field and the last character in the field is set to "/" to indicate that the complete value can be found in the *annotation* table. The over-long character values usually have come from old-SMIRS values that were typographic errors. Such problems should eventually be cleaned up manually.

1.9 Future Development

The content of ESM is, at present, a reformatted version of the no-longer-operating database named SMIRS, hereafter referred to as old-SMIRS. ESM also contains, in addition to the old-SMIRS information, instrument descriptions that were loaded into ESM from a separate file, referred to in this report as the

"LLL" file, that was created by the technicians who maintain the strong-motion instruments. The old-SMIRS and LLL information was transfered to ESM using a set of computer programs. Since the old-SMIRS information was not in tables as is required for a relational database like ESM and since old-SMIRS had a far more flexible structure than does ESM, some of the information did not fit in the new arrangement well. Manual attention should be given to the database contents to resolve those transfer problems. In addition to cleaning up problems that arose from the effort to move information from the hierarchical old-SMIRS structure into the ESM tables, problems that existed in old-SMIRS should also be cleaned up as should problems of inconsistency between the old-SMIRS information and the LLL information.

New information may be added to ESM that was not in old-SMIRS. For instance, the database may be expanded to provide detailed reference to the location of the computer files that contain x-y coordinates of the waveforms from those records that are significant enough to have been digitized and processed with the computer. The waveform files are archived for dissemination purposes in Denver, Colorado, now, but they eventually may be brought to Menlo Park and be cataloged in the ESM database.

Many tasks that would be required to modify ESM, as it exists now, into a more useful database are listed in several of the chapters in the "ESM Update, Maintenance, and Development Guide" (it is available from April Converse). Cleanup and completion problems are also mentioned along with the descriptions of the ESM tables in chapter 3 of this report.

ESM could be improved in many ways and could evolve substantially in the future. Few improvements will be made, however, unless prospective users make their needs known to those who allocate limited USGS resources to such efforts. If you wish to request that improvements be made to ESM or to other USGS techniques for distributing strong-motion information, write to the BESG data project at the address given on page 1.

If the structure of the new database evolves substantially, a new leading letter will be assigned as the first character of the database name, so ESM may evolve into F, G, and HSM.

New versions of this report will become available if the database and its support software evolve, so readers should take note of the date and version number of the report they are using. Note too, the date at the top of each page. Notices announcing new versions of the report will be entered into the database itself, in the *doc*umentation table, whenever new versions of the report are available.

Whenever substantial changes are made in the database, in its support software, or in its reference reports, a notice will be added to the VAX disk file at *PUB2:[ESM.MAINT]NEWS.TXT*. The

contents of that file are loaded into the database as *doc*umentation table rows with *doc.kind = news*. Users should refer periodically to the *NEWS.TXT* file or the *doc*umentation table in the database to keep abreast of new developments.

CHAPTER 2

Summary Diagram of Table and Column Names

2.1 Overview

The dotted boxes shown below represent columns in the tables in the ESM database. Each table can contain any number of rows. The format specification shown in each box indicates the format of the data in the column (See Chapter 1, Section 1.6 for more about formats). The column headings at the top of the boxes are the names of the data items stored in the associated columns. Refer to chapter 3 for a general description of each table and for detailed description of the contents of specific columns. Refer to appendix D for a comparison of this arrangement to that used in old-SMIRS.

Columns of the same name in different tables will often be used as "join fields" to join (or cross-reference through) two or more tables. Note, however, that the tables can be joined on any pair of columns having similar formats, not just those columns having the same names.

Equal signs (===) are shown along the bottom of boxes in subordinate tables where information has been duplicated from a parent to a subordinate table in order to link appropriate rows of the tables together. The letter shown among the equal signs indicates the table from which the information is duplicated. An asterisk (*) is shown at the bottom of boxes where information that was not in old-SMIRS will have to be added, somehow, to ESM. A plus sign (+) indicates information that was in old-SMIRS in one form or another, but that must be reformatted or given careful completion or cleanup attention before it would fit into ESM. Columns that may be removed from the database eventually, once cleanup and completion are finished, are shown with "xxx" along the bottom.

Although they are not shown here, the first version of the database will have two additional columns, *bdmsid* (i2) and *proof_read* (c3), tacked onto the end of each table. These two columns will be used during cleanup activities and may be removed from all the tables after the cleanup process is complete.

To simplify the appearance of the diagrams, the date flags (see Chapter 1, Section 1.7.2) are not shown as separate columns on the diagrams. The date flag names are: *ed_flag*, *ind_flag*, *rmd_flag*, and *cd_flag*, depending on whether they occur alongside an *edate* (event date), an *in_date* (installation date), a *rm_date* (removal date), or a *cdate* (calibration date).

2.2 ESM Event Information

Events Table (*range of e is events*)

edate	*eid*	*time*	*time_zone*	*lat*	*long*	*mag*	*depth*	*tag*
c1,date	c1	c9 +	c3 xxxxxx	f4	f4	f4	f4	i4

2.3 ESM Record Information

Records Table (*range of r is records*)

edate	*eid*	*rtype*	*rsn*	*s_trig*	*epi_d*	*own_ agency*	*tag*
cl,date	cl	c7	i2	f4	f4	c5	i4
					xxxxxxxx	xxxxxxx	

Record Traces Table (*range of rt is traces*)

edate	*eid*	*rtype*	*rsn*	*channel*	*stn_ number*	*sub_stn*	*peak*	*sm_dur*	*tag*
cl,date	cl	c7	i2	i2	i2	cl	f4	f4	i4
====r====	==r==	===r==	==r==			xxxxxxx	+	+	

Record Analyses Table (*range of ra is analyses*)

edate	*eid*	*rtype*	*rsn*	*seconds*	*dig_ agency*	*dp_ agency*	*arch_ agency*	*archived _id*	*tag*
cl,date	cl	c7	i2	f4	c5	c5	c5	c5	i4
====r====	==r==	===r==	==r===				*	*	

2.4 ESM Recording Site Information

Stations Table (*range of s is stations*)

stn_ number	*stn_ code*	*sub _stn*	*in_ date*	*rm_ date*	*state*	*region*	*lat*	*long*	*str_ code*	*hms*	*tag*
i2	c3	c1	c1,date	c1,date	c11	i1 *	f4	f4	c2	c1 *	i4

Arrays Table (*range of array is arrays*)

array_ name	*stn_ number*	*sub_stn*	*tag*
c12	i2	c1	i4

Numbers Table (*range of n is numbers*)

stn_ number	*agency*	*other_ number*
i2	c5	i2

2.5 ESM Recording Instrument Information

Recording Instruments Table (*range of i is instruments*)

rtype	*rsn*	*owner*	*stn_number*	*in_date*	*rm_date*	*tag*
c7	i2	c5	i2	c1,date	c1,date	i4

Transducers Table (*range of t is transducers*)

stn_number	*sub_stn*	*rsn rtype*	*rsn channel*	*ttype lvt*	*direction*	*loc_code*	*loc_desc*	*in_date*	*rm_date*	*tag*
i2	c1	c7	i2	i1	c3 c1	c5	c1	t30	c1,date	c1,date	i4
===s==	===s==	===i==	===i==		xxxx						

Transducer Calibrations Table (*range of c is calibrations*)

rtype	*rsn*	*channel*	*tsn*	*cdate*	*sens*	*period*	*damping*	*tag*
c7	i2	i1	i2	c1,date	f4	f4	f4	i4
===i==	===i==		===t==					

2.6 ESM Text

Annotation Table (*range of a is annotation*)

parent_tag	*type*	*ln*	*text (=one line of characters)*
i4	c1	i2	t65

Documentation (*range of doc is doc*)

kind	*code*	*ln*	*text*
c12 *	c12 *	i2 *	t65 *

2.7 Temporary Tables That Will Be Removed During Cleanup

There are several tables that contain alternative versions of parts of the primary tables. These other versions occur because much of the station, transducer, and instrument information was loaded from two different sources, old-SMIRS and the LLL file. The alternative rows will be compared to the corresponding rows in the appropriate primary table during cleanup, and differences will be resolved.

tablename	format	comments
morestn	*stations*	contains station descriptions that came from the LLL file for stations that had already been loaded into the *stations* table from old-SMIRS.
moreins	*instruments*	contains instrument descriptions that came from old-SMIRS for stations that had already had info loaded into the *instruments* table from the LLL file.
oldtrans	*transducers*	contains all the transducer descriptions that were originally loaded from old-SMIRS and from the LLL file. A copy of this table was cleaned up manually to create the *transducers* table.
delannot	i + *annotation*	contains annotation that came from the LLL file and which can probably be deleted. the first column in the table is the number of the station to which the annotation belongs.
moreagency	*documentation*	Agency code descriptions for agency codes that aren't used elsewhere in ESM.

CHAPTER 3

Glossary of Table and Column Names

A glossary of all the table and column names defined for the ESM database is given in this Chapter. Included in the glossary are many informal notes about inconsistencies and other problems with the database content. These notes are intended primarily for those who will be involved in the cleanup effort, but the notes may also help other users interpret what they find in the database.

3.1 Columns that Occur in all the ESM Tables

There are three columns that occur in all the tables in the database except for the *annotation* and *documentation* tables. These three columns are named "*tag*", "*bdmsid*", and "*proof_read*". The *bdmsid* and *proof_read* columns are intended for use during the manual cleanup process and their contents are only useful to those involved in the cleanup or other maintenance activities.

table.tag

The annotation tag, or "*tag*", column occurs at the end of every row in almost every ESM table. It is either zero or a number that uniquely identifies a specific row. The number is used to link the row to the appropriate rows in the *annotation* table. Data table rows that have no annotation associated with them carry a *tag* value of zero. Thus, a non-zero *tag* serves as a flag to remind the user to look for the annotation in addition to providing the actual link for retrieving the annotation.

Although there is a *tag* column in almost all the tables of the present version of the database, future versions will have annotation *tag* columns in the *events*, *records*, *stations*, and *instruments* tables only. Any annotation that the old-SMIRS-to-ESM translator attached to one of the subordinate tables has been, or will be, moved to the appropriate parent table rows during the manual cleanup process. At the time this is written, 20jul86, the only subordinate tables that still have some non-

zero annotation tags are the record-*traces* and the record-*analyses* tables; their annotation will eventually be moved to the *records* table.

table.bdmsid

The *table.bdmsid* values indicate where the rest of the information in the row came from. These values will be used during the database cleanup process. They allow those involved to refer back to the appropriate old-SMIRS entry or the appropriate section of the LLL file when trying to resolve problems with entries that the translating programs didn't handle well. This *bdmsid* column will be removed from the tables in the future, once the manual cleanup process is complete.

If a *table.bdmsid* value is positive, the information came from old-SMIRS and the number is the "bdmsid" value that the old-SMIRS database assigned to identify uniquely each of its root-level entries. If a *bdmsid* value is negative, the information came from the LLL file and the *bdmsid* number indicates a line number in the LLL file. If a *bdmsid* value is zero, the information was added to the database after the old-SMIRS-to-ESM transfer.

table.proof_read

This column is set aside for use by those involved in maintaining the database. A set of rows can be marked with something special so they can be retrieved and possibly modified as a group. For instance, when someone adds new information to the database, they may want to mark each of the rows added as "NEW" in the *proof_read* column. Then all the "NEW" information can be selected and sent through the "BASIC" report generator (see Chapter 9).

The LLL-file-to-ESM translator assigned "*LLL*" as the *proof_read* value in all the rows it generated. The old-SMIRS-to-ESM translator assigned "*A00*" as the *proof_read* value in all the rows it generated except for those in the *stations* and *transducers* tables. It assigned "*SSS*" to *stations.proof_read* and "*SSS*", "*RRR*", or "*III*" to *transducers.proof_read*, depending on whether the transducer information came from an old-SMIRS stations entry, a record entry, or a recorder entry.

As the manual database cleanup process proceeds, the *proof_read* values originally assigned by the translating programs will be modified to allow us to keep track of which rows have received which level of attention. The following *s.proof_read* values have been used in the early efforts to clean up the stations table:

SSS the row has not been changed (or only slightly) since it came from old-SMIRS.

LLL the row has not been changed (or only slightly) since it came from the LLL file.

A01 to *A09* someone has dabbled with the contents of the row, but just because one or two fields have been corrected doesn't mean the others are OK.

A10 The contents of the row are probably OK, but no one has checked it against any information other than old-SMIRS or the LLL file.

A20 The info has been checked against the files kept by the maintenance technicians, but it would be a good idea if someone else reviewed it all.

A50 The row has received all the first-pass cleanup attention it's ever going to get.

A51 Well, something came up, and the row was corrected again.

And, if we ever find time to do a second-pass cleanup, we may use:

B01 The row has received some second-pass correction.

B50 The row has been thoroughly proof-read twice.

B51 And corrected again.

3.2 Events Table (*range of e is events*)

Each row in the *events* table describes an earthquake or similar vibration that has triggered one or more of the strong-motion recorders in the permanent network.

The information in any row from the *events* table is incomplete without the associated event name(s), remarks, and comments from the *annotation* table.

The column name (underlined), INGRES format (in parenthesis), and description of the column contents is given in the list that follows for each field in an arbitrarily selected *event* table row marked as "*e*".

e.edate (c1,date) date of the event recorded.

Note that this is not just one column, but two, as are all the date columns in ESM. They are named "*ed_flag*" and "*edate*". In the first column (*ed_flag*) is a blank, a dash (-), or a number symbol (#); and in the second column (*edate*) is a date (an INGRES format date). The number symbol, if one occurs in the *ed_flag* column, indicates that the date is not precisely known. A dash in the *ed_flag* column indicates that the date is entirely unknown; the corresponding *edate* value given for unknown dates is *03-mar-1630*, or for unknown removal dates, *03-mar-2030*.

Cleanup or completion considerations:

- In old-SMIRS, this date was originally meant to be the local date, not the GMT date. Some dates may need to be changed when the times, *e.time*, are changed to GMT (Greenwich Mean Time).

- INGRES dates include a time field in addition to the date field, but the time field in INGRES dates will not be used in ESM. Time information is stored separately, in *e.time*. This is done so that *e.date* can be used as a join field with all the other date columns in ESM. It is done, also, because the time information in *e.time* would require a lot of manual cleanup before INGRES would accept it as part of a date value.

- There were 290 events entries in old-SMIRS for which there were two dates given, the two dates usually indicating the two technician visits surrounding the event. How shall these be represented in ESM? The old-SMIRS-to-ESM

translator just transferred the first date to ESM, put a # in the adjoining *ed_flag* column, and put the second date into comments.

e.eid (c1) event identifier, usually blank.

This column contains a blank or a unique identifier ("*a*", "*b*", "*c*", ...) used to distinguish among several events recorded on the same day. The column will be blank if there were only one event that day. Note that these identifiers are used, rather than the time of the event, because in some cases the time is not known, only the relative order of the events.

e.time (c9) time of the event.

The format used for time in old-SMIRS was 4-digit military time plus a 3-character suffix: 2 digits for the hour followed by 2 digits for the minute followed by a 3-character time-zone abbreviation. Example: 1643pst. Although this was a single data element in old-SMIRS, the time and time zone are stored in ESM as separate columns, *e.time* and *e.time_zone*. Once all the ESM times are converted to GMT, the time zone column will be removed.

Cleanup or completion considerations:

- The *e.time* column would only need to be 4 characters wide if all the old-SMIRS time values had consistently followed the format intended for them. Of course they do not, so I allowed a little extra space so that times that were slightly over-long wouldn't get thrown into the ESM j-comments. (The j-comments are temporary cleanup comments.)

- We will probably want more precision than 4 digits in the ultimate database, and we will probably want all times to be given in GMT. (Gerry says yes to both those "probably"s.) Chuck is inclined to use epocal time, like that used in the CSS database, in our ultimate database. I think it is OK to wait and worry about converting from old-SMIRS times to epocal time until after the more awkward cleanup problems have been taken care of, though.

- About GMT (Greenwich Mean Time) and its equivalent, UTC (Coordinated Universal Time) time: The current USGS notation is to call it GMT time, not the UTC that was in use several years ago. Jo says she now gives the time in GMT when adding new info to old-SMIRS. There are many old-SMIRS time values that were given in local time, though, as indicated in the time zone suffix. These are values Fritz (or whoever) loaded back in the days when we had agreed to use local time. What shall we do with those times that aren't UTC, correct them all manually? If so, beware of day-light savings time offsets! Also, realize that the

date may need to be changed a day when the corresponding time is converted.

Of the 1075 event entries in old-SMIRS, only 881 had any time given at all. Of those, there were 20 time values that had no suffix at all. Fritz and I originally planned that a missing suffix would indicate that it was the local time zone at the epicenter, but Fritz may have changed conventions and let missing suffix indicate UTC. Must check somehow.

The following time suffixes occur in old-SMIRS:

pst	434
utc	145
UTC	161
gmt	28
mst	12
pdt	18
gct	22
cst	6
ast	20
est	6
hst	8
mdt	1

The translator changed all "utc" and "UTC" time zones to "gmt" (lower case). It changed all the other time zones to upper case so they will attract attention to the fact that they need to be corrected. The 20 times having a blank time zone in old-SMIRS were assigned a time zone of "???" by the translator. The many old-SMIRS entries that had no time given at all have, in ESM, a blank *e.time_zone* as well as blank *e.time*.

- There were only 4 entries in old-SMIRS that contained more than one E-TIME value. In these 4 entries, the two values may be:
  1) typographical error;
  2) range within which the actual time is thought to have occurred;
  3) one time for local time, the other for UTC (Gerry's guess, if they differ by an integral number of hours).

  We must check more carefully during the cleanup process. The old-SMIRS-to-ESM translator just threw those four E-TIME.2 values into j-comments.

e.time zone (c3) time zone

See the time zone discussion with *e.time*, above.

e.lat (f4) latitude of the epicenter.

Latitudes are given in decimal degrees and range from -90.0

to +90.0. Positive latitudes are north of the equator; negative are south. A value of -303.0303 indicates that the latitude is unknown.

e.long (f4) longitude of the epicenter.

Longitudes are given in decimal degrees and range from -180.0 to +180.0. Positive longitudes are east of the Greenwich meridian; negative are west. A value of -303.0303 indicates that the longitude is unknown.

Cleanup or completion considerations:

It would be nice if latitude and longitude values could be stored as numbers, but displayed with a "N", "S", "E" or "W" suffix to indicate direction, as was done in old-SMIRS. Old-SMIRS wasn't smart enough to handle the discontinuity between 180E and 180W in range searches, but it would be nice if ESM could. Gerry prefers that the suffixes be in upper case rather than in lower case if we can have them. And Chuck doesn't want the suffixes at all.

e.mag (f4) magnitude of the event, M_L.

Cleanup or completion considerations:

- There are only 3 old-SMIRS entries where several of these are given. The old-SMIRS-to-ESM translator just put the three E-MAG.2 values into j-comments.
- Gerry's comment: "Eventually, as the years go by even the eng'g community will learn some new mag types. For the time being, we can stick with Richter (=local), M_L."

e.depth (f4) focal depth, km.

This column is rarely filled. There are only 153 old-SMIRS event entries that contain a depth value.

Miscellaneous notes about the events table:

There were 1075 events entries in old-SMIRS on 15oct84. Of those, 451 entries had more than one event-name, 39 of which had 3 names, and one had four names. To allow for the multiple names, the event names are not stored in the event table, but in the *annotation* table, with *a.kind="n"*.

Fritz, Gerry, and I allowed space for two data items in old-SMIRS event entries that probably need not go into ESM: E-MAX-MMI and E-MAG-TYPE.

E-MAX-MMI was the maximum intensity of the event, Modified Mercalli intensity scale. There were 483 entries in old-SMIRS in which a E-MAX-MMI value was given, 4 of which had two of

them. At one time Gerry said there is now no point in having E-MAX-MMI in the database. He says that although it's the only way that East Coast EQs are measured, the values are useful to us only if we have a record from the epicentral area. The translator put them into remarks.

E-MAG-TYPE was meant to indicate the type of magnitude given in E-MAG. Although there was space for it in old-SMIRS, I don't think it was ever used. (See Gerry's remarks with e.mag.)

3.3 Record Information Tables

The *records* table and its two subordinates, the *traces* table and *analyses* table, describe the strong-motion records acquired by the permanent strong-motion network.

There will be a row in the *records* table for each record retrieved from the network, but since only the significant records are digitized and analysed, most *record* table rows will not have any subordinate *analyses* table rows associated with them. If a record has been digitized more than once and/or data processing analyses performed more than once, however, there will be several rows in the *analyses* table for that record.

There will be at least one row in the *traces* table for each row in the *records* table in the first version of ESM. Once the information in the database is completed, however, there will be three or more rows in the *traces* table corresponding to each single row in the *records* table. These three or more rows will describe the three or more data traces on the record.

3.3.1 Records Table (*range of r is records*)

Each row in the *records* table describes characteristics of one strong-motion record. Most of the records presently described in ESM are three-component analog records on 75 mm. photographic film. Some of the older records were recorded on paper, however, and some recent records (many more are expected in the future) were recorded on magnetic tape or computer disks.

The information in any row from the *records* table is incomplete without the associated rows from the *analyses* and *traces* tables as well as remarks and comments from the *annotation* table.

The column name (underlined), INGRES format (in parenthesis), and description of the column contents is given in the list that follows for each field in an arbitrarily selected *record* table row marked as "*r*".

r.edate (c1,date) event date

Date of the event recorded. See *e.edate*.

Cleanup or completion considerations:
There are 329 entries in old-SMIRS in which this occurs

twice. The translator just threw the second date into the j-comments and set an "#" into *date_flag*.

r.eid (c1) event id, usually blank.

See *e.eid*.

r.rtype (c7) recorder type code.

"RFT" and "SMA-1" are examples of frequently occurring recorder type codes. All the recorder type codes are explained in the *doc*umentation table rows having *doc.kind= "recorder"*.

Cleanup or completion considerations:
Some recorder type codes in the database have a trailing "x", "y", or whatever (trailer in lower case, rest of the code in upper case, as in "RFTx"). These indicate that some cleanup attention is required. The trailing character was added, in fact, during cleanup, but not to correct the row, only to mark it as needing more attention.

r.rsn (i2) recorder serial number.

The recorder type code and serial number columns, *rtype* and *rsn*, occur in several tables: *records*,, record-*traces*, record-*analyses*, *instruments*, *transducers*, and *calibrations*. These fields can be used as "join fields" to acquire information from the various tables about a specific instrument.

Cleanup or completion considerations:
- The serial numbers often appear as negative numbers, and these are used to indicate that cleanup attention is required. Some of the reasons for the negative serial numbers are:
  - The serial number was often not given in old-SMIRS record descriptions. Just to distinguish one record from another, blank serial numbers were replaced by minus the appropriate station number in an early cleanup effort.
  - There were quite a few STD/1 recorders listed in the LLL file. Since they probably didn't all refer to the same instrument, the LLL-to-ESM translator reset the serial numbers to -1, -2, -3, etc.
- In old-SMIRS, the recorder type and serial number were stored together as a single data element named R-RR-ID. The format for R-RR-ID was to have the recorder type code (usually 3 characters) followed by a colon (:) followed by the recorder serial number. Example: "SMA:1234". In a very few cases (typos) "=", "-", and "*" were used rather than the colon. In some cases, no serial number was given;

in others no recorder type code (some of these may account for the multiply-occurring R-RR-IDs: typos where typist hit ";" rather than ":", creating a new occurrence of R-RR-ID). We must look into it more carefully during cleanup.

- All the serial numbers I saw in old-SMIRS were numeric, shall we make it a numeric field? (Yes, says Gerry. But what about future instruments? says Chuck.)

>>- R-RR-ID occurs in most (4131) of the old-SMIRS RECORDS entries; twice in 12 of those entries, 3 times in 5 of them, and 4 times in 2 of them. Translator just threw all but the first occurrence into the j-comments.
During cleanup, we must investigate carefully what those multiply-occurring R-RR-IDs mean. I hope they are just trash, but probably not. The problem is that Fritz decided that in old-SMIRS, a "record" would consist of those traces recorded by transducers situated at the same level in the same structure, regardless of whether or not those transducers were attached to the same recorder. In ESM, though, I think we'll let a "record", as described in the *records* table, be the same entity as the physical records we retrieve from the field. The traces in such a record are arranged just as they were recorded onto the recording medium, retrieved from the field, archived downstairs, and processed with the computer according to Pete and Barry's current conventions. Such a "record" usually , but not necessarily, involves traces from three orthogonal transducers, all situated within the same recording instrument. But things are not always so simple: transducers attached to the newer remote recorders may be widely scattered and need not be arranged in orthogonal triplets.

r.s trig (f4) S minus trigger interval, in seconds.

This value indicates the time lapse between the beginning of the record and the estimated arrival of the S-wave.

Cleanup or completion considerations:
This occurs in less than a fourth (638) of the old-SMIRS entries. (That's OK, says Gerry, S-wave often not identifiable.) It occurs twice in two of them and 6 times (!) in one of them.

r.epi d (f4) Distance from epicenter to recording site, km.

Cleanup or completion considerations:
- Rather than being stored in the database, this value should be recalculated each time it is wanted from the lat/long of the station and of the epicenter. R-EPI-D was never intended to be a permanent component of old-SMIRS

either, it was included only until the time when I could add software to calculate it. Well, I never had time to add the software to old-SMIRS, but it should be relatively easy to do with an INGRES program or "terminal monitor macro." Gerry wants us to be sure the calculation will work in searches too.

- R-EPI-D occurs in about three-fourths (2966) of the old-SMIRS records entries, and it occurs twice in one of those (probably a typo?).

r.own agency (c5)

An agency code. All the agency codes are explained in the documentation table rows having *doc.kind= "agency"*.

Cleanup or completion considerations:

- I'm not sure what this code represents! Examples of frequently occurring codes are: "USGS", "CDMG", "VA". I had thought it was meant to indicate the agency that stores the original analog record, but Gerry noted that: "Seems from the code values that confusion exists. USGS stores the originals of nearly all records in SMIRS. Looks like the codes represent the owner of the recorder." Well, these codes don't seem to match the codes in *i.owner* very well, so what are they?? Maybe we should check how well they match *i.owner* once again: I may have done the first check before I cleaned up the codes. Once we figure out what this code represents, we should consider (with Ed and Dick) whether we cannot just remove it from the database.

- R-AGENCY occurred in about a quarter (972) of the old-SMIRS record entries. It occurred twice in two entries and three times in another entry. The multiple occurrances are probably typos who's values were meant to be assigned to other data elements.

Miscellaneous notes about the *records* table:

There were 4181 records entries in old-SMIRS on 15oct84.

This table corresponds to the root level of the old-SMIRS RECORDS dataset. Note, however, that much of the information that had been in the old-SMIRS root-level RECORDS entries has either been moved to the ESM record *traces* table or duplicated there. The only columns in the *records* table that aren't also in the record *traces* table are *r.s_trig*, *r.epi_d*, *r.own agency*, and maybe annotation (via *r.tag*) that applies to the record as a whole, not to an individual trace. (We could just move those 3 columns to the record *traces* table -- means duplicating each value about three times -- and forget this table. Gerry says he prefers them separate, though, and I think

I do too.)

The transducer location value, R-TRANS-L, that had been stored in old-SMIRS records entries has been moved to the *transducers* table (at *t.loc_desc*) in ESM.

Fritz and I allowed space for two data items in old-SMIRS record entries that probably need not go into ESM: R-MMI and R-LENGTH.

R-MMI: Event intensity at the recording site, Modified Mercalli Intensity scale. Although there was space for this in old-SMIRS, it was never used. If we do decide that we want such in ESM, though, it would be nice to have them stored as numbers (so range searches will work) but displayed as Roman Numerals, as is done in old-SMIRS for E-MAX-MMI.
Gerry: "This is a carry-over from the statistical look at MMI vs. Accln of 1970's. Not available at the same time as rest of data. Suggest cancel."

R-LENGTH: Total length of the record, in seconds. Jo says she hasn't been putting this into old-SMIRS. I only found 3 of them there: the translator just put them into comments. It is D-LENGTH, the length of the record that was digitized, that will be stored (as *ra.seconds*) in ESM.

3.3.2 Traces Table (*range of rt is traces*)

Each row in the *traces* table describes either the characteristics of one of the data traces on a strong-motion record, or if *rt.channel =0*, the most significant of all the traces on a record. Ideally, once the cleanup and completion process has been finished, there will usually be three (or however many data traces there were on the record, if not 3) rows in this table corresponding to every one row in the *records* table.

Each row in this table is subordinate to a row in the *records* table.

The column name (underlined), INGRES format (in parenthesis), and description of the column contents is given in the list that follows for each field in an arbitrarily selected record *traces* table row marked as *rt*.

rt.rdate, *rt.eid*, *rt.rtype*, *rt.rsn*

These are join fields that link this row to the appropriate row in the *records* table.

rt.channel (i2) channel number or trace sequence number.

This number indicates which of the 3 (or however many) data traces on the record is described in this row. In the uncorrected version of ESM that came directly from the old-SMIRS-to-ESM translating program, this value was 0 to indicate that we don't have information about the separate traces yet and that the info in the row represents the most significant of all the traces on the record. Information about the separate traces will have to be added manually later, since the information was not available in old-SMIRS.

On film records, the trace sequence numbers start with "1" at the top of the record (with emulsion side up and the record progressing from left to right). Time traces and fixed reference traces are not counted. This number also corresponds to the manufacturer's channel number.

rt.stn number (i2) station number.

See *s.stn_number*.

The station number given in this table refers to the station where a transducer is located. To find the station

where the recorder is situated, for those few situations where a recorder is located in one station and some of its transducers in another, one would look in the *instruments* table.

Cleanup or completion considerations:
There were 40 entries in old-SMIRS in which R-STN occurred twice. What does that mean? They look like station numbers, not typos: R-STN.2 =0312 in 5 of those entries, =0942 in 22 of them, 0958 in 9 of them and =142, 269, 285, and 5106 in the others.
(From Gerry: 312 a dupe for 136
942 El Centro #6,
958 " #8,
269 see 585)

rt.sub stn (c1) Substation identification, usually blank.

See *s.sub_stn*.

Cleanup or completion considerations:
Since a trace's substation id. can be learned from the appropriate transducer table row, this column should probably be removed from the database once the *transducers* table is cleaned up.

rt.peak (f4) Peak value.

Approximate maximum acceleration recorded, in g's. Velocity records may one day be described in this database also. For velocity records, this value would represent the maximum velocity recorded.

The peak value is sometimes shown as a negative number. This indicates that the peak is some number less than -*rt.peak*. These values should be translated accordingly in any printed report. That is, a peak value occurring in the database as -.05 should be shown in a report as "< .05".

Cleanup or completion considerations:
- Note that old-SMIRS gave at most one value for all the traces on the record. ESM will need to carry a value for each trace (all 3 or all 11 or all whatever) if the info is going to be used to generate "Table 1" in the Strong Motion Program Circular.
- R-PEAK-A occurred in about half (2021) of the old-SMIRS entries. And I don't know what it means, but it occurred twice in 19 of those entries, 3 times in 15 of them, 6 times in two of them and 7 times in one of them. (Gerry's note: "Efforts to enter a peak for each trace?" Pete's note: "I seem to remember doing this for awhile.")

rt.sm_dur (f4) Strong-motion duration, in seconds.

This value indicates the approximate time interval between the first and last peaks greater than 0.1g, in seconds.

Cleanup or completion considerations:

- Old-SMIRS gave at most one value for each record, the largest such interval in all the traces on the record. ESM will need to carry a value for each trace if the info is going to be used to generate "Table 1" in the Strong Motion Program Circular.

- *Rt.sm_dur* had been named R-10TH-G in old-SMIRS. From the old-SMIRS manual: "If there was just a single peak on the record, R.10TH-G will be present in the entry, but will carry no value." Well, a data item can't be present but carry no value in an INGRES database, so shall we use value=-1 to indicate a single peak? Jo says she hasn't been using null values to indicate single peaks anyway. She just leaves out R-10TH-G and makes a remark about the single peak in the comments (or was it the remarks?). Null-valued R-10TH-G only occurred twice in old-SMIRS. Gerry: "Let's scale the duration of the peak, like 0.005 sec. That's clearly one peak."

- R-10TH-G occurred in only 177 of the old-SMIRS entries. It occurred twice in 5 of those entries and three times in 4 of them.

Miscellaneous notes about the *traces* table:

Text describing the location of the transducer that recorded the trace indicated in a row (*rt*) of the traces table can be found in the *t.loc_desc* field of the *transducer* table row (*t*) having *t.channel = rt.channel and t.rtype = rt.rtype and t.rsn = rt.rsn and t.stn_number = rt.stn_number*.

The *traces* table is new and has no counterpart in old-SMIRS although many of its columns come from items that had been stored in the root level of old-SMIRS records entries. For every one row in the *records* table, though, there should be three (or however many data traces there were on the record, usually 3) rows in this table, with much of the info the same in all three.

The old-SMIRS-to-ESM translating program generated just one row in the *traces* table for each record since there was no more information than that in old-SMIRS. Old-SMIRS only carries (if any) a single peak-acceleration and a single strong-motion duration value for each record, the largest of all such values on the record. But we must have the peak acceleration and

strong-motion duration values available for all three (or however many) traces if we are going to be able to generate the "Table 1" information from this database.

The station and substation information (*rt.stn_number*, and *rt.sub_stn*) are given in this table rather than in the records table to accommodate records taken from remote recorders attached to transducers that are scattered all over, possibly in different stations and sub-stations. Unfortunately, that means the same station and sub-station values must be repeated in all 3 *traces* table rows associated with any of the many records taken by the standard recorders having 3 orthogonal transducers contained in one box together with the recorder. The station number given in this table is the station where a transducer is located. To find the station where the recorder is situated, one would look in the *instruments* table.

3.3.3 Analyses Table (*range of ra is analyses*)

Each row in the *analyses* table describes the digitization, data processing, and whereabouts of the digitized data traces on a record. Each row in this table is subordinate to a row in the *records* table.

The column name (underlined), INGRES format (in parenthesis), and description of the column contents is given in the list that follows for each field in an arbitrarily selected record *analyses* table row marked as *ra*.

ra.edate, *ra.eid*, *ra.rtype*, *ra.rsn*

These are join fields that link this row to the appropriate row in the *records* table.

ra.seconds (f4) length of record processed, in seconds.

Cleanup or completion considerations:
- Why did old-SMIRS store it as character?? all the values I saw were numeric. Should check more carefully, though. Gerry: "should be".
- D-LENGTH occurred in 188 of the old-SMIRS sub-entries. It did not occur more than once in any of them.

ra.dig agency (c5) digitizing agency.

This code indicates the digitizing agency, organization, or method. The code should be one of the following:

"CDMG"	California Division of Mines and Geology.
"CIT"	California Institute of Technology,
"IOMGS"	IOM/TOWILL digitiztion done for the USGS,
"USC"	University of Southern California, and
"USGS"	U.S. Geological Survey,

Cleanup or completion considerations:
- The following codes occurred as D-RAW in old-SMIRS: "CIT", "IOM", "USC", and (a typo) "288".
- D-RAW occurred in only 226 of the analysis old-SMIRS sub-entries. I assume it was given in just the first of the multiply-occurring sub-entries.

ra.dp agency (c5) data processing agency.

This code indicates the agency that performed the

analysis. The code should be one of the following:
"CDMG" California Division of Mines and Geology.
"CIT" California Institute of Technology,
"USC" University of Southern California, and
"USGS" U.S. Geological Survey,

Cleanup or completion considerations:
- The following codes occurred as D-AGENCY in old-SMIRS: "EDIS", "USGS", and "USC".
- This code should be the analyzing agency, not the archiving agency. Old-SMIRS just had one code, D-AGENCY, assuming that same agency would both analyse and distribute the data. The old-SMIRS-to-ESM translator put the D-AGENCY values into *ra.dp_agency*, but those that are "EDIS" should be moved to *ra.arch_agency*. I can't remember whether or not I've already done this, but even if so, these two agency columns should receive careful attention during the cleanup and completion effort.
- D-AGENCY occurred in 404 of the old-SMIRS sub-entries. It occurred twice in 180 of those. (total of 584 occurrences.) Most, or maybe all, of those second occurrences probably indicate a second processing at USC.

ra.arch agency (c5) archiving agency.

This column will be blank in the first version of ESM.

ra.archived id (c5) archive identification.

We have no idea, yet, how the location of digitized waveform files will be noted in the database. I just included this empty column to serve as a reminder that such information is important and must be added eventually.

Annotation that may be Associated with any Row in the *Analyses* Table

There were two data elements in the old-SMIRS analyses sub-entries that the old-SMIRS-to-ESM translator just threw into the *annotation* table: D-FREQ and D-STAGE. Although this annotation is presently (20jul86) tagged to *analyses* table rows, it will probably be moved during the cleanup process to the parent *record* table rows.

DFREQ, the frequency band used in the stage 2 filtering process, was transfered as comments. Values occurring for D-FREQ in old-SMIRS are something like ".05-.07, 25-27". The numbers indicate the transition bands used in the low- and high-cut filters. D-FREQ occurred in 321 of the old-SMIRS sub-entries. It occurred 3 times in 78 of those sub-entries, a

total of 477 occurrences. (The multiple occurrences indicate the frequency band for each trace on the record, says Gerry. USC doesn't use the same band for all traces on the same record as we do.) D-FREQ was given as character information and not in a convenient form for searching, but perhaps someday in the future we should clean up the information and move it from the *annotation* table to numeric columns in the *analyses* table. It may be sufficient to store just two numbers that represent the corner frequencies (i.e., .07 and 25.) rather than the four numbers required to indicate the corner frequencies and roll-off bands. In the future, we should also add comments that indicate the filter type and roll-off order too.

D-STAGE values were transfered to ESM as comments. They are not very important and should perhaps be deleted altogether. (Yes, delete them, says Gerry.) D-STAGE was meant to indicate the highest stage of routine analysis that has been performed on the digitized data. It occurred in 402 of the old-SMIRS sub-entries. It occurred three times in one of them, but all three were typos, intended for D-FREQ. The values given to D-STAGE were integers, 1 through 4.

1 => scaled, uncorrected data
2 => Instrument corrected, filtered data
3 => response spectra
4 => Fourier spectra.

Miscellaneous notes about the *analyses* table:

There are two new columns in this table that didn't occur in old-SMIRS: *arch_agency* and *archive_id*. We might use these columns to store archiving information. For consistency with the ES&GDB, we may want to consider keeping the *archive_id* column in the record-*traces* table, rather than the *analyses* table, though.

There were 409 analysis sub-entries in old-SMIRS on 15oct84. 182 of these were second occurrences, so there were only 227 record entries that had analysis sub-entries attached. Why so many repeats? Gerry: "USC did a whole batch over again."

3.4 Recording Site Information

The *stations*, *arrays*, and *numbers* tables contain information about the recording sites in the permanent strong-motion network.

Each recording site has been assigned a station number. There may be any number of near-by instrumented structures and free-field sites collected together as a single station. The assignment of station numbers to recording sites was made somewhat arbitrarily, depending on the way a specific recording site was organized. For example, the instrumentation associated with a dam site may be scattered over several miles, and it may or may not all be regarded as a single station, depending on how the maintenance technicians regard the instruments. Usually, however, each instrumented structure is a single station. In those few cases where there are more than one instrumented structure or free-field site within a station, a "substation" identification (*a*, *b*, *c*, ...) is used to distinguish among them in the ESM database.

A structure may contain any number of transducers and any number of recorders. The transducers are usually connected to their nearest recorder, but they need not be. A transducer residing in a substation may be attached to a remote recorder that resides in a different substation or, rarely, in a different station. For this reason, the information about the recording instruments that are, or ever have been, in a station is not included in the *stations* table, but is stored separately, in the *instruments*, *transducers* and *calibrations* tables.

An array is a collection of recording instruments that is usually larger than is a station, possibly spanning many miles. An array may cover quite a small area, however. For example, the Differential Array at El Centro is only a few hundred meters long. What makes this collection of instruments an "array" is its precise layout and consideration of common timing. A station or a substation may be a member of more than one array. The list of stations and substations within each array is kept in the *arrays* table.

The *numbers* table is used to keep track of various alternative station numbers.

3.4.1 Stations Table (*range of s is stations*)

Each row in this table describes a recording site in the permanent strong-motion network.

Those few stations that involve more than one substation will have more than one row in the table: one row for each substation, with the station information (*s.stn, s.state*) repeated for all the substations.

The information in a row from the *stations* table is incomplete without the associated station names, structure descriptions, remarks, and comments from the *annotation* table.

The column name (underlined), INGRES format (in parenthesis), and description of the column contents is given in the list that follows for each field in an arbitrarily selected *stations* table row marked as "*s*".

s.stn number (i2) Station number, 1 to 9999.

Cleanup or completion considerations:
There are 4 stations in old-SMIRS that carry two station numbers. The second station numbers in those cases are 263, 320, 1364, and 1910. What does it mean to have two station numbers? Jo suggests that they come from two stations that have been combined into one. Jo's 1980 station list report says:

263: "same as 475"
320: "see 1437*"
1364 "see 1035"
1910 (no entry)

s.stn code (c3) 3-character station code

These codes were assigned (by Fritz Matthiesen) for the convenience of other Branches of the Survey that prefer to refer to stations with 3-letter codes rather than with the numeric station numbers in use in the Seismic Engineering group.

s.sub stn (c1) Substation identifier, usually blank.

The substation identifiers are usually blank, but if not blank, a single character. The identifiers were arbitrarily assigned by the translating programs and are used to distinguish among the several *stations* table rows describing one station that has more than one sub-station. There were very few such stations in old-SMIRS: only 35 of them.

s.in date (c1, date) Installation date.

This value is not given for very many of the stations: only 281 of them at the time old-SMIRS was translated into ESM. When the installation date is unknown, it appears as *-03-mar-1630*.

s.rm date (c1, date) Removal date.

If the station is still in the network, the *s.rm date* will be *-03-mar-2030*. Many removal dates are shown as *#01-jan-1986*: these are stations for which the old-SMIRS comments had indicated that the station was discontinued, but for which no removal date was given.

s.state (c11) State or country.

The state codes are the 2-character zip-code abbreviations used for states within the USA. Names of foreign countries are spelled out completely. All the state codes are explained in the *doc*umentation table rows having *doc.kind ="state"*.

s.region (i1) region code

The region codes are described in the *doc*umentation table rows having *doc.kind = region*.

The region codes in the database at this time (20jul86) are definitely out-of-date and of no use to anyone (they were transferred from the LLL file). In the future, however, new region codes will probably be assigned so the regions indicated in the ESM database will be the same as the regions indicated in the ES&GDB.

s.lat (f4) Latitude of the substation.

See *e.lat*

s.long (f4) Longitude of the substation.

See *e.long*

s.str code (c2) Structure class code.

All the structure class codes are explained in the documentation table rows having *doc.kind= "structure"*. The list of codes Fritz gave to Jo are:

GL= ground level
IS= instrument shelter
BL= building
DM= dam
BR= bridge
PP= pumping plant
NP= nuclear plant
TN= tunnel
GB= ground building
GS= ground shelter
GD= ground dam
BB= basement of building
BV= buried vault
HB= highway bridge (although this code should be used in future station descriptions, it never occurred in old-SMIRS. All the others did.)

Note that each of these brief structure codes may be supplemented with more information in the annotation associated with the station in question. The annotation types "z" (size of structure) and "b" (construction description) provide structural information.

Cleanup or completion considerations:
Here are all the SS-STR-C values that occurred in old-SMIRS that aren't on the above list: "GV", "DH", "MN (or was that NN?), "BP", "XX", "X" (originally intended to mean "other"), "gs" (should be GS?), "instr shltr" (should be IS?), "1-story bldg" (a typo meant for SS-STR-S?).

s.hms (c1) hard, medium, or soft

This value will indicate, very generally, with a "h", "m" or "s", what the geology underlying the site is like. More detailed description of the underlying geology is given in the *annotation* table, with *a.type = "g"*.

Cleanup or completion considerations:
- This column is completely empty now (01mar87). It should be filled in when the geology information in the *annotation* table (*a.type = "g"*) is cleaned up.

- Chuck Mueller suggests that rather than using "hard", "medium" and "soft" as the general geological categories we use the "A" through "F" categories Ken Campbel used in his BSSA article of December 1981:

A	Recent alluvium	Holocene Age soil deposits with rock >= 10m deep.
B	Pleistocene deposits	Pleistocene Age soil deposits with rock >= 10m deep.
C	Soft rock	Sedimentary rock, soft volcanics, and soft metasedimentary rock.

D	Hard rock	Crystalline rock, hard volcanics, and hard metasedimentary rock.
E	Shallow soil deposits	Holocene or Pleistocene Age soil deposits < 10m deep overlying soft or hard rock
F	Soft soil deposits	Extremely soft or loose Holocene Age soils such as beach sand or recent floodplain, lake, swamp, estuarine, and delta deposits.

Annotation associated with station table rows

There are several types of annotation rows that may be linked to a row in the *stations* table. The *a.type* value for such rows may be *"n"*, *"a"*, *"r"*, *"c"*, *"z"*, *"b,"* *"s"*, and *"g"* (plus the temporary kinds used during cleanup: *"j"*, etc.). The lines having *a.type* = *"n"*, *"r"*, and *"c"* have the same meaning as they do for all the other ESM tables: names, remarks, and comments. The *"a"*, *"z"*, *"b"*, *"s"*, and *"g"* annotation rows only occur for stations.

a.type = "a" Address.

The *a.text* field for these annotation table rows contain the address of the station. The address information transferred from old-SMIRS and from the LLL file was indistinguishable to the transfer program from "name" information. Thus, the addresses for stations that have not received any cleanup attention have annotation type = "n", while addresses for stations that have been cleaned up have annotation type = "a". Many stations do not have addresses and therefore have no address information at all.

a.type = "z" Size of the structure.

Here are some of the size description values transfered from old-SMIRS into *a.text* fields in the *annotation* table: "1-story bldg", "small instr shltr", "8' instr shelter", "9' high instr shelter", "26-story", "10-level garage", "earth dam", "power plant", "RR Bridge".

Cleanup or completion considerations:

- SS-STR-S occurred in 1277 of the 1719 structures sub-entries in old-SMIRS. It occurred twice in two of these sub-entries. I wonder if they are the same two sub-entries that had two SS-STR-C values?

- Since this info will be used in the "station list" report, it should be phrased and formatted in a consistent way. Once such cleanup has been performed, the *a.text* values

should probably be moved from the annotation table back into the stations table. Here are some examples Gerry made of the type of size information we'd like to have for some of the structure class codes:

s.str_code	*a.text where a.kind = "z"*
BL	26 stories
DM	120' high
BR	500' long
BB	26 stories
BV	20' deep
HB	500' long

a.type = "b" construction description

The *a.text* fields of these *annotation* table rows contain a short description of the type of construction, to supplement *s.str_code* and *a.kind ="z"*. Here are some of the construction description values that occurred in old-SMIRS: "steel frame", "prefab metal bldg", "earth dam", "concrete", "reinf conc shear wall", "stl brcd frm", "rc mom frm".

Cleanup or completion considerations:
- SS-STR-D occurred in 1277 of the structures sub-entries in old-SMIRS. It occurred twice in 43 of these, three times in 2 of them.
- Since this info will be used in the "station list" report, it should be phrased and formatted in a consistent way. Once such cleanup has been performed, the construction description values should probably be moved from the annotation table back into the stations table.

a.type = "s" source of information.

The *a.text* field in these annotation rows usually contain an agency code, very often "CDMG".

Cleanup or completion considerations:
This may merely indicate the owner of the recorder, and if so, it can be removed from the station's annotation because the information is kept as *i.owner* in the *instruments* table. We must check before deleting these, though! I tried it before any other cleanup had occurred and only 298 out of the 1670 source values matched the appropriate *i.owner* values. That may be because the agency codes were inconsistently expressed (should be cleaned up by now) or because the source values mean something other than *i.owner* does.

a.type = "g" geology

The *a.text* field in these annotation rows contain a code or comment indicating the site geology in more detail than is (or will be) given in *s.hms*.

The text is sometimes in English (i.e., "alluvium"), and other times is a string of two-character codes separated by colons. The in-english text indicates surface geology, the two-character codes indicate the geology at depth. Here are some examples of the uncoded geology descriptions that occurred in old-SMIRS as SS-GEOL-C:

"volcanic rk",
"alluvium >23m",
"alluvium",
"sndstn & cnglmrt",
"sandstone",
"granite",
"granitics",
"siltstone",
"conglomerate".

Here are some examples of coded SS-GEOL-C values found in old-SMIRS:

"AL84:SSU:FR"
"UU80"
"AL:SSS"
"Z58:CL12:GS15:USI56"
"CL2:SC12:ZC14:GS16:MF30"

The geology codes were devised by Barry Silverstein in association with Gerry Brady. The codes were originally presented in five USGS Open-File Reports titled "Geologic Description of Selected Strong-Motion Accelerograph Sites, Part I" through "... Part V" by B.L. Silverstein (USGS OFR # 78-1005, 79-428, 79-1619, 80-473, and 80-1140). Each of these reports describes the geology underlying from 10 to 20 stations. Barry also loaded the coded geologic descriptions that he presented in the five Open-File Reports into old-SMIRS. The geologic codes presented in the first of these reports are somewhat different than the codes in the fifth report. Those codes shown in the list below that are given in the first report but not in the last report or vice-versa are prefaced by a (1) or a (5) to indicate which report it came from.

a) Class Code:

(1)	ALV	Alluvium
(1)	IGN	Igneous Rock
(1)	MET	Metamorphic Rock
(1)	MIX	Mixture of Rock Types
(1)	SED	Sedimentary Rock

b) Rock Symbols:

(1)	AC	Coarse Alluvium

	AF	Artificial Fill
(1)	AL	Alluvium (replaced by NF, AF, CN, CA, and SO)
	AG	Agglomerate
	AM	Amphibolite
	AN	Andesite
	BA	Basalt
	BR	Breccia
(5)	CA	Coarse Artificial Fill
	CG	Conglomerate
	CH	Chert
(1)	CI	Combination Igneous
(1)	CM	Combination Metamorphics
(1)	CR	Combination Rock Types
(1)	CS	Combination Sedimentary
(5)	CN	Coarse Natural Fill
	CY	Claystone
	DI	Diorite
	DK	Dikes or Sills
	DO	Dolomite
	FR	Franciscan Rocks
	GB	Gabbro
	GD	Granodiorite
	GN	Gneiss
(5)	GO	Gouge
	GR	Granite
	GS	Greenstone
	HF	Hornfels
	LF	Lava Flows
	LS	Limestone
	MR	Marble
	MS	Mudstone
	MZ	Monzonite
(5)	NF	Natural Fill
(1)	OB	Obsidian
	PH	Phyllite
(1)	PU	Pumice
	QM	Quartz Monzonite
	QZ	Quartzite
	RY	Rhyolite
	SC	Schist
	SH	Shale
	SI	Siltstone
	SL	Slate
(5)	SO	Soil
	SP	Serpentinite
	SS	Sandstone
	SY	Syenite
	TF	Tuff
(5)	UB	Undifferentiated Basement
(5)	UD	Unconsolidated Deposits
(5)	UI	Undifferentiated Igneous Rocks
(5)	UM	Undifferentiated Metamorphic Rocks
(5)	UP	Unconsolidated Pyroclastic Rocks
(5)	US	Undifferentiated Sedimentary Rocks

VA Volcanic Ash
VG Volcanic Glass
XB Crystaline Basement

c) Descriptive Symbols

C Consolidated
D Deep
F Fractured, Sheared, or Jointed
I Interbedded
L Layered or Stratified
M Massive
P Permafrost
S Semiconsolidated
U Unconsolidated
V Veneer
W Weathered
(1) X Crystalline Basement

d) Other Symbols

: Overlying
+ And
. Decimal
K x1000 (multiplier)

Here follows an explanantion, copied from USGS Open-File Report # 78-1005 (the first of the five reports), of how the geology codes are expressed:

> The first three characters are a class code that broadly describes the surface materials and is intended for those interested only in general geology. Rock types are given by two letter abbreviations. A third letter describes the condition of the rock. For example, SH indicates shale, and SHF would indicate a fractured, sheared or jointed shale. Lack of a descriptive term generally implies that its condition is not known. The abbreviations are not standardized to a particular nomencalture.
>
> A number following the rock type indicates the depth of the rock, i.e. AL914:CS1067 means there is alluvium to 914 meters and a combination of sedimentary rocks from 914 to 1067 meters.
>
> The letter K following a number is a 1000 times multiplier, e.g. 4K equals 4000 meters. In some cases a number is not given. This indicates that the actual depth is not known but evidence suggests that a particular rock is present. The colon, not including the one immediately after the three letter general description, may be translated literally as "overlying". A plus sign between two (or more) rock names means both (or all) are present. Example:

ALV:AL84:SSU:SH+SC
ALV: => alluvium at surface.
(General surface geology.)
AL84 => alluvium to 84m.
:SSU => overlying unconsolidated sandstone to unknown depth.
:SH+SC => overlying shale and schist to an unknown depth.

Cleanup or completion considerations:

- SS-GEOL-C occurred in 768 of the substations in old-SMIRS. It occurred twice in 696 of these, three times in 116, four times in 15 and five times in 1. What are the multiple occurrences all about? (Let Gerry Brady and Barry Silverstein resolve these.)

- When they reviewed this report, Chuck and John both thought the geology codes were unnecessarily complicated and I (April) do too. Why were they coded in the first place since they were loaded into old-SMIRS in a way that prevented all but the first code in a string to be used for retrieval? If we want this much detail in the database and it's not going to be used as the basis for retrieval, why not have it in English?

Miscellaneous about stations

There were 1696 stations entries in old-SMIRS on 15oct84.

Fritz and I allowed space for two data items in the old-SMIRS station entries that probably need not go into ESM: SS-SHEAR-V and ST-LOC.

SS-SHEAR-V	Near-surface shear wave velocity. Although there was a space for this in old-SMIRS, it was never entered.
ST-LOC	Station location. This data element had been removed from the old-SMIRS database a long while ago. Its values were loaded as new occurrences of the ST-NAME data element.

3.4.2 Arrays Table (*range of array is arrays*)

This table provides a list of the station/substations within the various arrays and within other groups of nearby or otherwise related stations.

The column name (underlined), INGRES format (in parenthesis), and description of the column contents is given in the list that follows for each field in an arbitrarily selected *arrays* table row marked as "*array*".

array.array_name (t15) Name of an array.

The recognized array names are explained in the documentation table rows having *doc.kind = "array"*. Here are the array names that existed at the time old-SMIRS was transfered to ESM: "APEEL", "Bear Valley", "Gilroy", "Cholame", "Lake Hughes", "El Centro", "Leona Valley", "Maricopa", "Hemet", "Oroville" and "Differential". ("Differential", however, should get changed to "El Centro Differential" and "Hollister Differential".)

array.stn_number (i2) Station number.

See *s.stn_number*.

array.sub_stn (c1) sub-station identification.

See *s.sub_stn*.

3.4.3 Numbers Table (*range of n is numbers*)

This table indicates numbers that have been used to identify the various stations other than those station numbers used in *s.stn_number* and elsewhere in the ESM database. These alternative numbers may be numbers used at one time by the USGS or they may be numbers used by other agencies, such as CDMG.

n.stn number (i2) Station number.

This is a station number used elsewhere in ESM.

n.agency (c5) Agency code.

This indicates the agency that uses, or has used in the past, the number shown in the *n.other_number* column for the same station that is identified in ESM by the number shown in the *n.stn_number* column.

n.other number (i2) station number.

This is another number that has been used to identify the station that is identified in ESM as *n.stn_number*.

3.5 Recording Instrument Information

The recording-*instruments* table and its subordinates, the *transducers*, and transducer-*calibrations* tables, all contain information about the recording instruments that are in (or ever have been in) the permanent strong-motion network.

3.5.1 Instruments (*range of i is instruments*)

A row in the *instruments* table simply identifies a recorder, its owner, and the time interval during which the recorder was in place in a specific station. The different stations in which a recorder may have resided in the past will each be described in a separate row in the table. More information about the transducers attached to a recorder is given in the *transducers* table and in the transducer-*calibrations* table.

Most of the information in this table came from the LLL file, not from old_SMIRS. The old_SMIRS version of this table can be found in the separate *moreins* table which is intended only for use during the cleanup process and which will be removed from the database eventually.

The column name (underlined), INGRES format (in parentheses), and description of the column contents is given in the list that follows for each field in an arbitrarily selected *instruments* table row marked as "*i*".

i.rtype (c7) recorder type code.

See *r.rtype*.

Cleanup or completion considerations:
RR-ID occurred twice in 6 of the old-SMIRS recorder entries. What does that mean?

i.rsn (i2) recorder serial number

See *r.rsn*.

The pair of data elements *i.rtype* and *i.rsn* can be cross-referenced, or "joined" to *r.rtype* and *r.rsn* to learn whether any significant records have been taken by this instrument.

i.owner (c5) recorder owner code.

The recognized agency codes are explained in the documentation table rows having *doc.kind = "agency"*.

i.stn number (i2) station number.

This value indicates the station in which the recorder is installed. See *s.stn_number*.

i.in date (c1,date) installation date.

This is the date the recorder was installed at station *i.stn_number*

i.rm date (c1,date) removal date.

Date the recorder was removed from this station, if it has been removed. If the recorder has not been removed, *i.rm_date = #03-mar-2030*.

3.5.2 Transducers Table (*range of t is transducers*)

Each row in the *transducers* table should describe a single motion-sensing device. At present, however, the table is incomplete and some of the rows loaded by the old-SMIRS-to-ESM translator actually describe a set of 3 sensors, arranged to record orthogonal components of motion. As with the record-*traces* table, the channel number is used to indicate whether a row describes a single transducer (*t.channel* > 0) or the most significant (that with the highest peak) of a set of transducers (*t.channel* = 0).

To complicate matters further, there are several different versions of most transducer description rows: one came from old-SMIRS recorders (*t.proof_read* = *"III"*), one from old-SMIRS stations (*t.proof_read* = *"SSS"*), others from old-SMIRS records (*t.proof_read* = *"RRR"*), and yet another from the LLL file (*t.proof_read* = *"LLL"*). All these transducer descriptions were loaded into a single table, the *oldtrans* table, and the first version of the *transducers* table was made from a copy of *oldtrans*. Next, a very rough cleanup pass was made over the *transducers* table to remove rows that appeared to duplicate, to some extent, rows that came from another source. The versions coming from the LLL file were those that were most often left in the *transducers* table, since they are more complete than the other versions. When *transducer* rows from several sources contradicted each other, both versions were (usually!) left in the table. All this means that careful attention must be given to the *transducers* table during the cleanup process. The *oldtrans* table remains in the database to help with this task.

The column name (underlined), INGRES format (in parenthesis), and description of the column contents is given in the list that follows for each field in an arbitrarily selected *transducers* table row marked as "*t*".

t.stn number and *t.sub stn* are used as join fields with the *stations* table

Cleanup or completion considerations:
Some of the station numbers currently (20jul86) in the *transducers* table are negative numbers. These are merely used as a flag to indicate that cleanup is required for that row. The problem is not necessarily in the station number, other than that it has had its sign changed.

t.rtype and *t.rsn* are used as join fields with the *records*, *traces*, *analyses*, *instruments*, and *calibrations* tables.

t.channel (i1) channel or trace sequence number.

This number indicates the relative location of the trace recorded by this transducer on a physical record. For film records, trace location numbers start at the top of the record (with emulsion side up and the record progressing from left to right). Time traces and fixed reference traces are not counted. This number also corresponds to the manufacturer's channel number.

t.channel = 0 indicates that the rest of the information in the row represents the most significant values of all the traces on the record, not just one trace. In the present version of the database (29may86), the *transducers* table rows that came from the "LLL" file usually have *t.channel* > 0, and those that came from old-SMIRS have *t.channel* = 0, or < 0. The rows with *t.channel* < 0 represent a single channel, but which channel is which is uncertain (more cleanup required!).

t.ttype (c3) transducer type, usually "acc".

This field indicates the type of transducer. "*acc*" indicates one of the several types of mechanical transducers; "*fba*" indicates a force-balance accelerometer (these are used in digital recorders and others); "*cdm*" indicates a Carder displacement meter; and "*dm*" indicates a displacement meter (larger than a cdm).

t.lvt (c1) transducer identification.

This character is assigned to transducers that are arranged in 3-component packages. It is usually a "*l*", "*v*", or "*t*" to indicate whether the transducer senses motion in the longitudinal direction of its container (largely a manufacturer's definition), vertical, or transverse to the "*l*" direction.

Cleanup or completion considerations:

- This column may be deleted in the future, since its information is better expressed in the *t.direction* column. Does anyone see any reason for keeping it? Yes, says Gerry, "L, V, T" is historical and had better stay.
- Some of the *lvt* values given in the LLL file were "*a*" and "*b*". What do they mean? (Ask Dennis Johnson.)

t.direction (c5) orientation of the transducer.

Transducer orientations are given as the direction of ground acceleration for a positive trace, in azimuthal notation: 001 to 360 degrees from north in the clockwise

direction; or "UP" or "DOWN".

t.loc code (c1) transducer location code.

Cleanup or completion considerations:
- Very few transducer location codes are given in the database yet (20jul86). There were only 72 given in all of old-SMIRS and some of them were lost during the first cleanup pass on the *transducers* table. So these must be assigned almost entirely during the cleanup/completion process. They can probably be determined in most cases from the infomation given in *t.loc_desc*, once that is cleaned up.
- The transducer location codes suggested in the old-SMIRS manual were
  - G Ground level in an instrument shelter (approximately free-field.),
  - B Basement or ground level within a structure,
  - S Above ground level in a structure,
  - V Instrument Vault, presumed below ground, and
  - X other.

  The only T-LOC-CODE codes that actually occur in old-SMIRS are: "B", "b", "G", and "S". Whatever codes we end up using should be explained in the documentation table.

t.loc desc (t30) location description.

A short transducer location description, in English, that supplements *t.loc_code*.

Cleanup or completion considerations:
- These descriptions may go into the annotation table someday, once the conflicting descriptions from the 4 different sources are resolved and the *t.loc_code* values are filled in accordingly.

t.in date (c1,date) Date the transducer was installed.

t.rm date (c1,date) Date the transducer was removed.

In the first version of ESM, the transducer instalation and removal dates are the same as the recorder installation and removal dates, since only one date was available from old-SMIRS or from the LLL file. The two sets of dates are separate in ESM, however, to provide for the fact that the transducers are sometimes rearranged and replaced even though the recorder itself is not moved. This is not a common occurrence with the "permanently" installed analog recorders, but it will be common with portable, digital recorders.

Miscellaneous Notes about the Transducer Table:

During early cleanup activity, many of the transducer table rows have been altered, not to correct them, but merely to flag some problem that needs fixing during subsequent cleanup. These alterations were made to distinquish one *transducers* row from another when the information given was clearly in error or incomplete. Some of the temporary cleanup alterations that have been made are:

- *t.stn_number* = - the value originally given. This was done primarily when it appeared that the station number given was wrong. But it was also done when it wasn't clear what was wrong with the row, it just didn't match the other *oldtrans* rows for the station indicated: the problem could be a miss-typed recorder type or serial number just as easily as a miss-typed station number.
- *t.rsn* = - the value originaly given. The recorder serial numbers were (or should have been!) changed in the appropriate *records*, *traces*, and *analyses* table rows too.
- *t.rtype* = original value with an "x", "y", or ... tacked onto the end, as in "STDx". The recorder types were (or should have been!) changed in the appropriate *records*, *traces*, and *analyses* table rows too.

Chuck says it would be useful to have the transducer serial number in this table. The serial numbers are important, he says, for the GEOS-type instrumentation. But we rarely have the transducer serial numbers for the SMA-type instruments. Those we do have came from the LLL file and are now kept in the calibration table. Gerry tells me that the technicians (SMA-type) don't often keep track of the transducer serial numbers. We must learn more by talking it over with Gerry, Dennis, and Ed. It certainly would be nice to be able to use the transducer serial number as a join field between the *transducers* and *calibrations* tables.

3.5.3 Transducer Calibrations Table
(*range of c is calibrations*)

Each row in the *calibrations* table describes the calibration constants determined for a transducer on a specific date.

This information was almost entirely missing in old-SMIRS, so the translator loaded it from the LLL file. There were only 6 calibration subentries in all of old-SMIRS. They occurred as 3 pairs, 2 subentries linked to each of 3 transducers. We should visually check whether or not those six old-SMIRS entries correspond to what was loaded from the LLL file, but even if they don't, the LLL version is probably the correct one.

c.rtype, and *c.rsn* are used as join fields to *i.rtype* and *i.rsn*; *t.rtype* and *t.rsn*; and *r.rtype* and *r.rsn*.

c.channel is used as a join field with *t.channel*.

c.tsn (i2) Transducer serial number.

If the value was given in the LLL file as a roman numeral rather than a decimal number, c.tsn will be shown in ESM as a number less than zero.

c.cdate (c1,date) calibration date.

c.sens (f4) sensitivity.

This value indicates an accelerometer's sensitivity (cm/g), or a displacement meter's magnification (dimensionless).

c.period (f4) Transducer period, seconds.

Cleanup or completion considerations:
Dennis Johnson has asked that these be converted to natural frequencies (= *1.0/c.period*) and renamed to *c.nfreq*.

c.damping (f4) damping, fraction.

This value indicates the damping in the transducer, as a fraction of critical damping.

3.6 Text Information

Comments, remarks, and other free-form text may be linked to any row in the *events, stations, records* and *instruments* tables. The text is kept in a separate table, the *annotation* table; it contains one line of text in each row. Several, many, or no lines of text from the *annotation* table may be linked to any one row in the *events, stations, records* or *instruments* table. The *tag* field in the data table is joined to the *parent_tag* field in the *annotation* table to link a row in the data table to the appropriate rows in the *annotation* table.

The "*type*" column in the *annotation* table is used to indicate which type of annotation is given in each row. The four most commonly occurring types of annotation are names, references, comments, and x-comments (*type* = *"n"*, *"r"*, *"c"* and *"x"*). Names include event names and station names; references indicate published reports; comments contain text that might appear in a printed report; and x-comments are comments that contain temporary or messy notes that should be excluded from a printed report. Another type of annotation, *"j"* for junk, includes problems that should be resolved as the database information is cleaned up, things like old-SMIRS information that the old-SMIRS-to-ESM translating program couldn't fit well into the ESM tables and that will require manual cleanup attention.

The *doc*umentation table is similar to the *annotation* table in that it too contains one line of text in each of its rows. The information in the *doc*umentation table is more general than that in the *annotation* table, however. It provides general announcements about the database and a dictionary of all the codes used in ESM.

3.6.1 Annotation Table (*range of a is annotation*)

This table contains lines of text used as comments, remarks, names, or other free-form information associated with the *events, stations, records* and *instruments* tables.

a.parent_tag (i2) parent tag.

This tag links the text in this row to the appropriate row in another table when *a.parent_tag= table.tag*.

a.type (c1) type of annotation indicator.

type code	type of annotation
n	name
r	references
c	comments
x	temporary or messy comments
a	address (used only with the *stations* table)
z	size of structure (used only with *stations* table rows)
b	construction description (used only with *stations* table rows)
m	maintenance notes (used only with *stations* and *instruments* tables. In fact, there are really none at all yet - 20jul86 - and may never be.)
g	geology class code. (used only with the *stations* table). These will need a lot of manual clean-up attention.
s	source of information. (used only with the *stations* table). These are agency codes, but I am not sure what they mean. We must find out during cleanup.

And there will be several annotation categories that will come and go as the cleanup process progresses. At present (20jul86) there is only one, *j* for junk:

j	Miscellaneous values that the translator couldn't otherwise fit into the tables. Included in this category will be values that were too long to fit in their alloted column width and second (or greater) occurrences.

a.ln (i2) line number.

This number sequences the several lines of text of the same type that may be linked to the same parent.

a.text (t65) One line of text.

3.6.2 Documentation Table (*range of doc is doc*)

This table contains more general documentation than that kept in the *annotation* table. The table contains an explanation of the various codes used in the other tables and also contains some very general sorts of informatation such as "*news*".

d.kind (c12) kind of documentation identifier.

kind of documentation given in the remarks associated with this entry. ="*news*", "*agency*", "*array*", *geology*", "*recorder*", "*region*", "*state*", "*structure*", "*trans_loc*", etc., etc.

d.code (c11) code

A description of this code is given in the *d.text* field. The code is one that may occur elsewhere in the database.

d.ln (i2) line number.

d.text (t65) one line of text.

3.7 Temporary Tables

There are several tables that contain alternative versions of parts of the primary tables. These other versions occur because much of the station, transducer, and instrument information was loaded from two different sources, old-SMIRS and the LLL file. The alternative rows will be compared to the corresponding rows in the appropriate primary table during cleanup, differences will be resolved, and the temporary tables removed. The tables are *morestn, moreins, oldtrans, delannot,* and *moreagency*. They are described in the summary chapter at section 2.7

CHAPTER 4

SMSM: A Small, Sample Version of ESM

There is a small version of ESM named SMSM (for small smirs) that can be used for testing purposes. The SMSM database was established primarily as a learning tool for those few people who will be involved in changing the contents of the ESM database or in developing its software, but SMSM can also be used by those who only want to learn how to retrieve information. Experimenting with the small version of the database is preferable to experimenting with the full version, first, because you can feel confident that you won't ruin anything important if you make a mistake while learning how to change the contents of the database, and second, because the machine will respond more quickly (for you and for other people using the machine at the same time) if you are dealing with a small database. It is quite annoying to other VAX users when INGRES degrades the response time for their computer use, so it would be best, for other's sakes as well as your own, to use the small, more quickly processed, version of the database when you are just experimenting with database manipulating techniques.

The ESM/SMSM database administrator (John Watson?) can regenerate the SMSM database easily whenever a user has altered or destroyed any of its content. If you do alter the contents of SMSM, please remember to inform the database administrator.

SMSM contains about 34 stations, 100 events, and 250 records. Among the stations are:

Isabella Dam (an example of a station with several substations)
Bulk Mail Facility in Richmond (an example of a remote recorder having transducers in two separate buildings)
Hollister Differential Array
Long Valley Dam
Martis Creek Dam
Coalinga Pleasant Valley Pump Plant
APEEL Array
UCSB station
Pacoima Dam

After the station information for SMSM was collected, all the record, event, and recorder information associated with those stations was added to the collection.

This is a rather haphazard collection: most of the stations were selected because they did not translate from old-SMIRS into the ESM structure conveniently. This small collection of problem entries was used to consider how similar problems will be resolved throughout the database during the cleanup effort. Thus, the information in SMSM will require more manual cleanup than will the majority of ESM information.

Although the SMSM database will normally be a subset of ESM, it will on some occasions be a copy of the entire ESM database. This will happen when the database administrator or other database developers are experimenting with database programs and techniques that require the full database rather than a subset. If you wish to learn whether SMSM is a subset of ESM or a copy of it, you can compare the number of rows in the stations table in the two databases via the *helpr* command as follows:

```
$ HELPR SMSM stations
$ HELPR ESM  stations
```

CHAPTER 5

Format of Results from ESM Queries

Selected portions of the ESM information can be displayed in three ways: in forms, in terminal monitor displays, and in reports.

Forms are used with computer terminals having CRT screens as their display medium. The forms are manipulated with the INGRES software package named QBF, or Query-By-Forms. The forms that have been established specifically for use with the ESM database are available to the user through the *$ESMFORMS* command described subsequently in Chapters 6 and 7.

The INGRES terminal monitor provides access to the INGRES query language, or QUEL. QUEL provides a much wider range of possible operations than does Query-By-Forms, and it can be used with printer terminals as well as with video terminals, but it is more complicated to use than Query-By-Forms. The *$ESMQUEL* command provides access to the ESM information via QUEL and the INGRES terminal monitor. Instructions for using these are given in Chapters 6 and 8.

The reports are formatted disk files that can be copied to a variety of display devices, but primarily to a line-printer. The reports that have been established specifically for the ESM database are generated through the *$ESMRPT* command that is described in Chapters 6 and 9.

Three fundamental formats in which ESM information can be displayed are named "*onetbl*", "*anntbl*", and "*basic*". Each format is available as a form and as a report layout. Other, more specialized formats will probably be developed for reports in the future (See Chapter 9).

The *onetbl*, or "one table", format shows just one table at a time. The format is very similar to that provided with the INGRES/QUEL *print* command, except that some unneccessary space that is given in some of the columns, particularly the date columns, in the *print* format has been removed in the *onetbl* format. Thus, the *onetbl* display of very wide tables like the *transducers* table have been made to fit across a 132-column screen or printer sheet, while *print* results will not. The

onetbl forms and reports are available for any of the 12 permanent tables in ESM, and with some restrictions, *onetbl* forms and reports also work with almost any arbitrarily constructed user-created table too.

The *anntbl*, or "annotated table", format shows rows from a specified table plus any appropriate *annotation* table rows. *anntbl* forms and reports work for the *events*, *records*, *analyses*, *stations*, and *instruments* tables.

The *basic* format displays closely related tables together. It shows rows from a specified table plus any appropriate rows from tables subordinate to the table specified. For instance, a *basic* report for a row from the *records* table will show the *records* table row along with the record-*traces*, record-*analyses*, *transducers*, and *annotation* table rows associated with the record in question. The *basic* reports work for the *events*, *records*, *stations*, and *instruments* tables, but none of the *basic* forms work yet. A very preliminary version of the basic/*stations* form is available for illustration, however. Each of the four *basic* layouts has a different set of subordinate tables although *annotation* rows appear in all four *basics* and *transducers* rows appear in all but the basic/*events*. Note that the basic/*events* layout is the same as the anntbl/*events* layout since *annotation* is the only table that is subordinate to the *events* table.

Figures 5-1, 5-2, and 5-3 illustrate the three formats when used with selected portions of the *stations* table. Figure 5-1 shows an arbitrarily selected group of *stations* table rows in *onetbl* format; Figure 5-2 shows two stations in *anntbl* format; and figure 5-3 shows one station in *basic* format.

stn_nu	stn	s	i	in_date	w	rm_date	state	region	lat	long	st	h	tag	bdmsid	pro
117	ELC	#		1-jan-1933	-	3-mar-2030	CA	1	32.794	-115.549	GB		4432	38	A10
133	HSB	#		1-jan-1933		25-jul-1975	CA	1	34.089	-118.338	BL		4487	86	A10
135	PEL	#		1-jan-1933		16-apr-1973	CA	1	34.090	-118.340	GS		5824	5022	A10
241				15-feb-1967	-	3-mar-2030	CA	1	34.220	-118.470	BL		4510	107	A10
287	SOD			23-may-1968	-	3-mar-2030	CA	1	34.160	-117.680	DM		4582	171	A10
288	VRN	#		1-jul-1932		25-jul-1975	CA	1	34.000	-118.200	BB		4605	189	A10
634	NWK	a		3-mar-1630	-	3-mar-2030	CA	1	33.920	-118.070	BL		5280	4557	A10
634	NWK	b		3-mar-1630	-	3-mar-2030	CA	1	33.920	-118.070	GS		5281	4557	A10
707	LMR			26-apr-1973	-	3-mar-2030	CA	1	33.852	-117.440			5140	4435	A10
895	BHA	a		13-mar-1974	#	1-jan-1986	CA	1	34.008	-118.361	GS		5527	4771	A10
895	BHA	b		11-aug-1974	-	3-mar-2030	CA	1	34.008	-118.361	TN		5529	4771	A10
942	EO6			13-feb-1975	#	12-feb-1980	CA	1	32.840	-115.490	GB		4779	3916	A10
958	EO8			13-mar-1975	#	12-feb-1980	CA	1	32.810	115.530	GB		4838	3961	A10
963				9-may-1975	-	3-mar-2030	CA	1	34.190	-118.310	RL		5011	4180	A10
1010	BKD	a		10-jan-1968	-	3-mar-2030	CA	2	39.910	-122.340	DM		4781	-4289	A10
1010	BKD	b		10-jan-1968	-	3-mar-2030	CA	2	39.820	-122.340	DM		4783	-4293	A10
1010	BKD	c		13-mar-1974	-	3-mar-2030	CA	2	39.820	-122.340	GS		8065	-5635	A10
1035	ISD	a		19-jan-1968	-	3-mar-2030	CA	2	35.646	-118.481	DM		4643	222	A20
1035	ISD	b		19-jan-1968	-	3-mar-2030	CA	2	35.645	-118.479	DM		8090	222	A20
1035	ISD	c		19-jan-1968	-	3-mar-2030	CA	2	35.644	-118.483			8091	222	A20
1035	ISD	d		19-jan-1968	-	3-mar-2030	CA	2	35.643	-118.472	DM		8092	222	A20
1035	ISD	e		19-jan-1968	-	3-mar-2030	CA	2	35.642	-118.470	DM		8093	0	A20
1035	ISD	f		19-jan-1968	-	3-mar-2030	CA	2	35.642	-118.469	DM		8408	0	A20
1065	ALX			12-nov-1934		30-sep-1976	CA	2	37.790	-122.400	BL		4675	248	A10
1084	SLD	1		10-may-1968	-	3-mar-2030	CA	2	37.070	-121.080	DM		8140	-4613	A10
1084	SLD	2		10-may-1968	-	3-mar-2030	CA	2	37.070	-121.080	DM		8139	0	A10
1084	SLD	3		10-may-1968	-	3-mar-2030	CA	2	37.060	-121.080	DM		8141	-4617	A10
1084	SLD	4		10-may-1968	-	3-mar-2030	CA	2	37.060	-121.070	DM		8142	0	A10
1084	SLD	5		10-may-1968	-	3-mar-2030	CA	2	37.060	-121.070	DM		8143	-4625	A10
1084	SLD	6		2-jul-1968	-	3-mar-2030	CA	2	37.060	-121.070	DM		8144	-4629	A10
1098	TMD	a		17-jan-1968	-	3-mar-2030	CA	2	36.410	-119.000	DM		4685	258	A10
1098	TMD	b		17-jan-1968	-	3-mar-2030	CA	2	36.420	-119.000	DM		8153	258	A10
1098	TMD	c		13-mar-1974	-	3-mar-2030	CA	2	36.410	-119.000			8412	-5707	A10
1133	MKD	a		8-dec-1971	-	3-mar-2030	CA	2	39.326	-120.113	DM		4786	3921	A10
1133	MKD	b		8-dec-1971	-	3-mar-2030	CA	2	39.326	-120.113	GS		4788	3921	A10
1135	AUB			8-dec-1971	-	3-mar-2030	CA	2	38.950	-120.970	GS		4973	4144	A10
1140	ORD			29-jan-1970	-	3-mar-2030	CA	2	39.550	-121.480	DM		4789	3922	A10
1168				1-dec-1970	-	3-mar-2030	CA	2	37.790	-122.400	BL		5367	4633	A10
1216	WSD			2-may-1972	-	3-mar-2030	CA	2	38.720	-123.004	DM		5456	4715	A10
1247	PTR			8-sep-1972	#	1-jan-1986	CA	2	38.040	-122.800	GS		5315	4589	A10
1248	WPS			13-sep-1972	-	3-mar-2030	CA	2	35.750	-118.100	GS		5452	4714	A10
1249	CPM	a		28-jul-1978	-	3-mar-2030	CA	2	40.349	-124.354	GS		4659	232	A10
1249	CPM	b		20-sep-1972	-	3-mar-2030	CA	2	40.349	-124.354	IS		4658	232	A10
1420	SG4	a		15-may-1975	-	3-mar-2030	CA	2	36.760	-121.410	TN		5343	4614	A10
1420	SG4	b		15-may-1975	-	3-mar-2030	CA	2	36.760	-121.410	GS		5344	4614	A10
1439	FBM	a		18-dec-1975	-	3-mar-2030	CA	2	37.900	-122.320	RL		5684	4887	A10
1439	FBM	b		18-dec-1975	-	3-mar-2030	CA	2	37.900	-122.320	GR		5687	4887	A10
1439	FBM	c		18-dec-1975	-	3-mar-2030	CA	2	37.900	-122.320	GB		5689	4887	A10
1465	SID			21-may-1976	-	3-mar-2030	CA	2	35.800	-120.870	DM		4808	3938	A10
1484	LSD	a	-	3-mar-1630	-	3-mar-2030	CA	0	36.062	-118.924	DM		5941	6902	A10
1484	LSD	b	-	3-mar-1630	-	3-mar-2030	CA	0	36.061	-118.920	DM		0	6902	A10
1484	LSD	c	-	3-mar-1630	-	3-mar-2030	CA	0	36.060	-118.915	DM		0	6902	A10
1484	LSD	d	-	3-mar-1630	-	3-mar-2030	CA	0	36.059	-118.913	DM		0	6902	A10
1484	LSD	e	-	3-mar-1630	-	3-mar-2030	CA	0	36.059	-118.923	DM		0	6902	A10
1499	PJH	a	-	3-mar-1630	-	3-mar-2030	CA	0	37.823	-122.233	BL		6034	6987	A10
1499	PJH	b	-	3-mar-1630	-	3-mar-2030	CA	0	37.823	-122.233	GS		6036	6987	A10
1508	SSC	a	-	3-mar-1630	-	3-mar-2030	CA	0	37.353	-121.903	BL		6073	7018	A10
1508	SSC	b	-	3-mar-1630	-	3-mar-2030	CA	0	37.355	-121.905	GS		6075	7018	A10
1519	RMD	a	-	3-mar-1630	-	3-mar-2030	CA	0	37.949	-120.524	DM		5997	6953	A10
1519	RMD	b	-	3-mar-1630	-	3-mar-2030	CA	0	37.949	-120.524	GS		5999	6953	A10
1528	WMD	a	-	3-mar-1630	-	3-mar-2030	CA	0	37.370	-118.987	DM		6106	7043	A10
1528	WMD	b	-	3-mar-1630	-	3-mar-2030	CA	0	37.356	-118.988	GS		6107	7043	A10
2002			-	3-mar-1630		22-apr-1975	NV	3	36.020	-114.740	DM		4692	263	A10
2151	LPD			24-apr-1973	-	3-mar-2030	OR	4	43.920	-122.750	DM		5595	4820	A10
2173	SBM			17-oct-1975	#	1-jan-1986	WA	4	47.290	-122.320	RL		5680	4886	A10

Figure 5-1
Stations in *ONETBL* format

STATIONS table row:

stn_nu	stn	s	i	in_date	r	rm_date	state	region	lat	long	st	hme	tag	bdmeid	pro
1035	ISD	a		19-jan-1968	-	3-mar-2030	CA		2	35.646	-188.481	DM		4643		222	A20

	parent	t	ln	text	pro
Names:	4643	n	1	Isabella Dam	A00
Address:	4643	a	1	Box 977; Lake Isabella, CA. Phone: (714) 379-2742	A01
Construct:	4643	b	1	earth dam	A00
Geology:	4643	g	1	RK	A00
	4643	g	2	granite	A00
Source:	4643	e	1	ACOE	A00
X-com:	4643	x	1	There are two dams at this site: a main dam and an auxiliary dam.	A01
	4643	x	2	subestation a = main dam, crest	A01
	4643	x	3	subestation b = main dam, lower spillway gallery	A01
	4643	x	4	subestation c = main dam, toe at downstream campground	A01
	4643	x	5	subestation d = auxiliary dam, right abutment	A01
	4643	x	6	subestation e = auxiliary dam, crest	A01
	4643	x	7	subestation f = auxiliary dam, lower tower	A01

STATIONS table row:

stn_nu	stn	s	i	in_date	r	rm_date	state	region	lat	long	st	hme	tag	bdmeid	pro
2002			-	3-mar-1630		22-apr-1975	NV		3	36.020	-114.740	DM		4692		263	A10

	parent	t	ln	text	pro
Names:	4692	n	1	Hoover Dam	A00
	4692	n	2	formerly Boulder Dam	A01
Address:	4692	a	1	Boulder City, Nevada	A01
Construct:	4692	b	1	concrete dam	A00
Geology:	4692	g	1	RK	A00
	4692	g	2	basalt	A00
Source:	4692	e	1	USBR	A00

Figure 5-2
Two Stations in *ANNTBL* format

STATIONS table row:

stn_nu	esn	e	lin_date	l=	lm_date	state	region	lat	long	lat	hme	tag	bdmeid	pro
1035	ISD	a	19-jan-1968	-1	3-mar-2030	CA	21	35.6466	-118.481	DM		4863	222	A 20
1035	ISD	b	19-jan-1968	-1	3-mar-2030	CA	21	35.6451	-118.4791	DM		8090	222	A 20
1035	ISD	c	19-jan-1968	-1	3-mar-2030	CA	21	35.6464	-118.483			8091	222	A 20
1035	ISD	d	19-jan-1968	-1	3-mar-2030	CA	21	35.6431	-118.4721	DM		8092	222	A 20
1035	ISD	e	19-jan-1968	-1	3-mar-2030	CA	21	35.6421	-118.4701	DM		8093	01	A 20
1036	ISD	f	19-jan-1968	-1	3-mar-2030	CA	21	35.6421	-118.4691	DM		8408	01	A 20

	parent	t	ln	text	pro
Names:	4863	n	1	Isabella Dam	A00
Address:	4863	a	1	Box 877; Lake Isabella, CA. Phone: '714) 379-2742	A01
Construct:	4863	b	1	earth dam	A00
Geology:	4863	g	1	RK	A00
	4863	g	2	granite	A00
Source:	4863	e	1	ACOE	A00
m-com:	4863	z	1	There are two dams at this site: a main dam and an auxiliary dam.	A01
	4863	z	2	substation a = main dam, crest	A01
	4863	z	3	substation b = main dam, lower spillway gallery	A01
	4863	z	4	substation c = main dam, toe at downstream campground	A01
	4863	z	5	substation d = auxiliary dam, right abutment	A01
	4863	z	6	substation e = auxiliary dam, crest	A01
	4863	z	7	substation f = auxiliary dam, lower tower	A01

Names:	8090	n	1	Isabella Dam, substation b = main dam, lower spillway gallery	A01

Names:	8091	n	1	Isabella Dam, substation c: downstream campground	A01

Names:	8092	n	1	Isabella Dam, substation d: auxiliary dam, right abutment	A01

Names:	8093	n	1	Isabella Dam, substation e: auxiliary dam, crest	A01

Names:	8408	n	1	Isabella Dam, substation f: auxiliary dam, lower tower	A01

TRANSDUCERS in the Station:

stn_nu	s	type	wn	channel	tty	l	direc	l	loc_desc	t	lin_date	l=	lm_date	tag	bdmeid	pmc
1035	a	RFT-250	109	1	acc	l	184		main dam, crest		19-jan-1968	-1	3-mar-2030	01	-4393	LLL
1035	a	RFT-250	109	2	acc	v	up		main dam, crest		19-jan-1968	-1	3-mar-2030	01	-4393	LLL
1035	a	RFT-250	109	3	acc	t	104		main dam, crest		19-jan-1968	-1	3-mar-2030	01	-4393	LLL
1035	b	RFT-250	108	1	acc	l	184		main dam, spillway gallery		19-jan-1968	-1	3-mar-2030	01	-4389	LLL
1035	b	RFT-250	108	2	acc	v	up		main dam, spillway gallery		19-jan-1968	-1	3-mar-2030	01	-4389	LLL
1035	b	RFT-250	108	3	acc	t	104		main dam, spillway gallery		19-jan-1968	-1	3-mar-2030	01	-4389	LLL
1035	c	RFT-350	588	1	acc	l	184		toe at downstream campground		15-mar-1974	-1	3-mar-2030	01	-5691	LLL
1035	c	RFT-350	588	2	acc	v	up		toe at downstream campground		15-mar-1974	-1	3-mar-2030	01	-5691	LLL
1035	c	RFT-350	588	3	acc	t	104		toe at downstream campground		15-mar-1974	-1	3-mar-2030	01	-5691	LLL
1035	d	RFT-250	112	1	acc	l	184		auxiliary dam, right abutment		19-jan-1968	-1	3-mar-2030	01	-4405	LLL
1035	d	RFT-250	112	2	acc	v	up		auxiliary dam, right abutment		19-jan-1968	-1	3-mar-2030	01	-4405	LLL
1035	d	RFT-250	112	3	acc	t	104		auxiliary dam, right abutment		19-jan-1968	-1	3-mar-2030	01	-4405	LLL
1035	e	RFT-250	110	1	acc	l	184		auxiliary dam-right crest		19-jan-1968	-1	3-mar-2030	01	-4397	LLL
1035	e	RFT-250	110	2	acc	v	up		auxiliary dam-right crest		19-jan-1968	-1	3-mar-2030	01	-4397	LLL
1035	e	RFT-250	110	3	acc	t	104		auxiliary dam-right crest		19-jan-1968	-1	3-mar-2030	01	-4397	LLL
1035	f	RFT-250	111	1	acc	l	184		auxiliary dam-tower base		19-jan-1968	-1	3-mar-2030	01	-4401	LLL
1035	f	RFT-250	111	2	acc	v	up		auxiliary dam-tower base		19-jan-1968	-1	3-mar-2030	01	-4401	LLL
1035	f	RFT-250	111	3	acc	t	104		auxiliary dam-tower base		19-jan-1968	-1	3-mar-2030	01	-4401	LLL

NUMBERS used by other Agencies:

stn_nu	agenc	other_
1035	USGS	1036
1035	USGS	1037
1035	USGS	1038
1035	USGS	1039
1035	USGS	1384

ARRAYS to which the station belongs:

array_name	stn_nu	e	tag	bdmeid	pro
(NO DATA)					

Figure 5-3
One Station in *BASIC* format

CHAPTER 6

How to Retrieve Information from the Database

6.1 Overview

INGRES provides two methods for addressing your retrieval requests to its database management software. The easiest method is to use the QBF, or Query-By-Forms, software, but the more general and more powerful way is to use the QUEL, or Query Language, software.

Query-By-Forms and the user communicate through forms and menus displayed on a CRT screen. At each stage in the process, a form is displayed and a message at the bottom of the screen gives a menu of options, a prompt, or some other similar text. Although Query-By-Forms provides a limited range of operations at any one time, it is highly visual and easy to use without much instruction. Query-By-Forms can only be used, of course, by users having video displays on their terminals. Users having printing terminals, like the TI-700s, must use QUEL rather than Query-By-Forms.

QUEL provides a much wider range of possible operations than does Query-By-Forms, but it is more difficult to learn to use. Instructions for using Query-By-Forms are given in Chapter 7 and instructions for using QUEL are given in Chapter 8. This chapter presents information useful to all ESM users, regardless of whether they are using Query-By-Forms or QUEL.

6.2 Getting Started

The ESM database resides on a VAX 11/750 computer at the USGS offices in Menlo Park, California. If you wish to retrieve information from the database, you must either have an account on the machine or know the login sequence for one of the visitor's accounts there. Any of the people associated with the ES&G data project (names, address, and telephone number are given in Chapter 1, page 1) can let you know the account name and current password of one of the visitor's accounts.

Once a user has logged onto the machine, the VAX operating

system (it is named "VMS") will display a dollar sign ($) at the terminal as a prompt to let the user know that the machine is ready and waiting for the user to type something. The dollar sign will appear whenever the operating system is ready to accept a user's input line.

There are many ways to access an INGRES database. The simplest method for accessing ESM is to type *"ESMFORMS* in response to the *$*-sign prompt. You must have defined the *ESMFORMS* symbol beforehand, though, and this is done by invoking the pre-set commands and symbol definitions in the file at *PUB2:[ESM]ESM.DEF*. This will have been done for you automatically when you logged onto the machine if you are using one of the visitor's accounts. If you intend to use ESM from your own account, however, you must type *"@PUB2:[ESM]ESM.DEF"* yourself, before using the ESM (or SMSM) database. The commands in the *ESM.DEF* file do many things other than defining the *ESMFORMS* symbol: all the instructions that follow in this report assume that the *ESM.DEF* commands have been invoked. The best way to ensure that this always happens is to insert a line containing *"$@PUB2:[ESM]ESM.DEF"* into the *LOGIN.COM* file in your home directory. (Create a *LOGIN.COM* file there containing nothing but that one line if you do not already have a *LOGIN.COM* file.)

6.3 Terminal Types

INGRES needs to know which type of terminal it is communicating with. The start-up definitions in the *ESM.DEF* file are arranged as though the user were at a DEC VT100 terminal or a compatible terminal like a VT220. Those users who are using a different type of terminal must indicate their terminal type when invoking the *ESM.DEF* definitions. Instead of the

$ @PUB2:[ESM]ESM.DEF

line (explained above in section 6.2), these users should type, or include in their *LOGIN.COM* files, the longer:

$ @PUB2:[ESM]ESM.DEF **termname**

Where **termname** is one of the many terminal names shown in appendix C of the "INGRES Terminal User's Guide" (in Volumne 4 of the INGRES documentation set.)

6.4 VAX Commands

Use the *$ESMFORMS* or *$ESMQUEL* command to get started. *$ESMFORMS* provides a menu giving you the option to use the many

forms that have been established for the ESM database. *$ESMQUEL* connects you to the ESM database via the INGRES terminal Monitor, the software that allows users to construct and submit QUEL commands from a terminal. The SMall SMirs counterparts to these two ESM commands are *$SMSMFORMS* and *$SMSMQUEL*.

The *$ESMFORMS*, *$ESMQUEL*, and *$LOGOUT* commands are the only three VAX operating system level commands a user must know of in order to address queries to the ESM database. The first two initiate the user's interaction with INGRES and the third ends the user's terminal session.

Many users will want to make use of other commands, however, and there are 3 sets of commands to choose from. One set of commands is provided by the VAX/VMS operating system and they are described in manuals provided by Digital Equipment Corporation.

Another set of commands is provided by the INGRES software. Knowlege of these commands is not required for those who wish to address queries to the ESM or SMSM databases, but it is required of those who are involved in maintaining and updating the databases. INGRES is well documented in a set of 4 manuals provided by Relational Technology Incorporated. Most of the information in those manuals that is pertinent to query users has been copied or paraphrased in this report.

Other commands that can be used at VAX command level (in answer to the $ prompt) are made available to the user when the *ESM.DEF* file is invoked. These commands are not provided by the VAX operating system or by INGRES, but are specifically tailored for use with the ESM and SMSM databases. The most important of these are *$ESMFORMS* and *$ESMQUEL*, but there are others as well.

The list that follows shows all the commands enabled by the *ESM.DEF* file. Some of the commands in the list require that the user type some arguments after the command name. The argument names are shown in bold type; when used, the user should type something meaningful in place of the argument name, "*arrays*" rather than "**table**", for instance. Any of the following commands that require arguments will display a brief message describing the required arguments if the user types the command without arguments.

command	purpose/use
$ESMFORMS **format table**	Access the ESM database through forms and the INGRES Query-By-Forms software. The **format** specified on this command line may be "onetbl", "anntbl", or "basic" (see Chapter 5 for description of these formats). Instructions for using the forms and the *$ESMFORMS* command are given in chapter 7.

$ESMFORMS may be abbreviated as *$ESMF*.

$ESMQUEL Access the ESM database via the INGRES Terminal Monitor and the QUEL query language. Instructions for using the Terminal Monitor and QUEL are given in chapter 8. Note that the user may want to use the *$S80* or *$S132* command (see below) before using this command.
$ESMQUEL may be abbreviated as *$ESMQ*.

$ESMQUEL **<disk-file** request that ESMQUEL read its commands from the named disk file rather than from the user's terminal. Note that in accordance with INGRES command conventions, the input disk file name must be prefaced with a less-than sign (<) and there should be no blanks between it and the file name.

$ESMRPT **format report-table primary-table** [**disk-file**] [NLPP=**#**]
Generate reports from ESM data. This command requires several arguments, see Chapter 9. *$ESMRPT* may be abbreviated as *$ESMR*.

$SMSMFORMS **format table**
Counterpart of the *$ESMFORMS* command for the SMSM database. *$SMSMFORMS* may be abbreviated as *$SMSMF*.

$SMSMQUEL Counterpart of the *$ESMQUEL* command for the SMSM database. *$SMSMQUEL* may be abbreviated as *$SMSMQ*.

$SMSMRPT **format report-table primary-table** [**disk-file**] [NLPP=**#**]
Counterpart of the *$ESMRPT* command for the SMSM database. *$SMSMRPT* may be abbreviated as *$SMSMR*.

$S132 This command sets the number of characters that may be seen across a VT100 terminal screen to 132.
$S132 is invoked from the *$ESMFORMS* command, so a user need not type the *$S132* before typing *$ESMFORMS*. *$S132* is not invoked automatically from the *$ESMQUEL* command, however, so a user wishing to see 132 characters across the screen in a QUEL session should type *$S132* before typing *$ESMQUEL*.

$S80 This command sets the number of characters that may be seen across a VT100 terminal screen to 80. This is the default character width setting.

6.3 Control Keys

You can interrupt a command by typing control-c. Such an interruption is useful in cases when you have entered a command and you want to stop it. For instance, if you've typed a QUEL *print* command for a very long table, you can use control-c to stop the display before the entire table is printed.

Various other functions can be performed by using other control keys. To use a control key, press the CONTROL or CTRL key an hold it down while you press the specified other key. For example, to enter control-c, hold the CTRL key down and then press the c key.

Many control keys are used to move the cursor around in the forms presented with the *$ESMFORMS* command. Those keys are described in Chapter 7.

One control key you should try to remember not to use while using INGRES or any of the ESM/SMSM commands is control-y. Control-y will interrupt whatever command is going at the time, just as control-c would do, but control-y does not allow INGRES to perform any wrapup functions before the interruption. If you should inadvertently use control-y rather than control-c, just type *continue* in response to the first *$*-prompt thereafter. The interrupted process will resume and you may then type the control-c. If you or anyone else should inadvertently interrupt an INGRES session with control-y, a diagnostic message will appear the next time someone tries to use the ESM (or SMSM) database. The message will tell you, if you are that someone, to type *$RESTOREDB ESM* (or *$RESTOREDB SMSM*) and you should, in fact, do so.

Another control key that is sometimes useful while using ESM, SMSM, and/or INGRES is control-t. This momentarily interrupts the terminal output to display a line of information about the current process. If the computer seems to be taking a long while to respond to your requests, you may wonder if it is because the computer is very busy and is not allocating many resources to your request or if it is because you have given it a very complicated request. Several control-t's can inform you, without interrupting your request, whether or not your request is receiving much attention from the computer.

Table 1-1, page 1-6, of the "VAX/VMS DCL Concepts" manual lists many more of the the control keys and describes their function.

CHAPTER 7

Query by Forms

7.1 Overview

The INGRES Query-by-Forms, or QBF, software is a visually oriented, forms-driven interface between the INGRES database management software and the terminal user. Query-By-Forms and the user communicate through forms and menus displayed on a CRT screen. At each stage in the process, a form is displayed and a message at the bottom of the screen gives a menu of options, a prompt, or some other similar text. Although Query-By-Forms provides a limited range of operations at any one time, it is highly visual and easy to use without much instruction. Query-By-Forms enables the user to retrieve information from databases without requiring knowledge of QUEL, the INGRES Query Language described in Chapter 8.

7.2 VAX $-level Commands

Use the *$ESMFORMS* (or *$SMSMFORMS*) command to get started. This command enables users to access the ESM (or SMSM) database using Query-By-Forms with forms and support software that have been specifically tailored to the ESM and SMSM databases. The following variations of the ESMFORMS command are available for accessing the ESM/SMSM forms:

$ESMFORMS
$ESMFORMS **format** **table**
$SMSMFORMS
$SMSMFORMS **format** **table**

$ESMUPDATE
$ESMUPDATE **format** **table**
$SMSMUPDATE
$SMSMUPDATE **format** **table**

Where **format** is *"onetbl"*, *"anntbl"*, or *"basic"* and **table** is the name of one of the tables in the database. Those users who dislike making extra keystrokes may abbreviate *$ESMFORMS* to *$ESMF*, *$SMSMFORMS* to *$SMSMF*, *$ESMUPDATE* to *$ESMU*, and

$SMSMUPDATE to *$SMSMU*. The formats *"onetbl"*, *"anntbl"*, and *"basic"* may be abbreviated to *"one"*, *"ann"*, and *"bas"*, or even more briefly, to *"o"*, *"a"*, and *"b"*.

The $ESM___ commands access the ESM database while the $SMSM___ commands access the SMSM database. The "FORMS" in the *$ESMFORMS* and *$SMSMFORMS* commands indicates that the user wants to make queries to the database but not to make any changes in the database contents. The "UPDATE" in the *$ESMUPDATE* and *$SMSMUPDATE* commands indicates that the user may decide to change the database contents in addition to making queries. The *$ESMUPDATE* command is not available to all users, but *$SMSMUPDATE* is available to anyone who wants to practice using database updating techniques.

The forms commands, when typed without the **format** and **table** parameters on the command line, should eventually present the user with a menu of options from which the user can select the table he wants to see and the format (*onetbl*, *anntbl*, or *basic*) in which he wants it displayed. At present, however, the commands do not work as intended, since all the required software has not been written yet. All the commands do at present when given without input parameters is inform the user what input parameters are required.

7.3 The ESM/SMSM Forms

Once a user has typed the appropriate command, a blank form will appear on the screen. Figure 7-1 illustrates one of the forms, the *anntbl* form for *stations* data. Note the menu, the list of options (*"Go Blank LastQuery Order Help Quit:"*) at the bottom of the form.

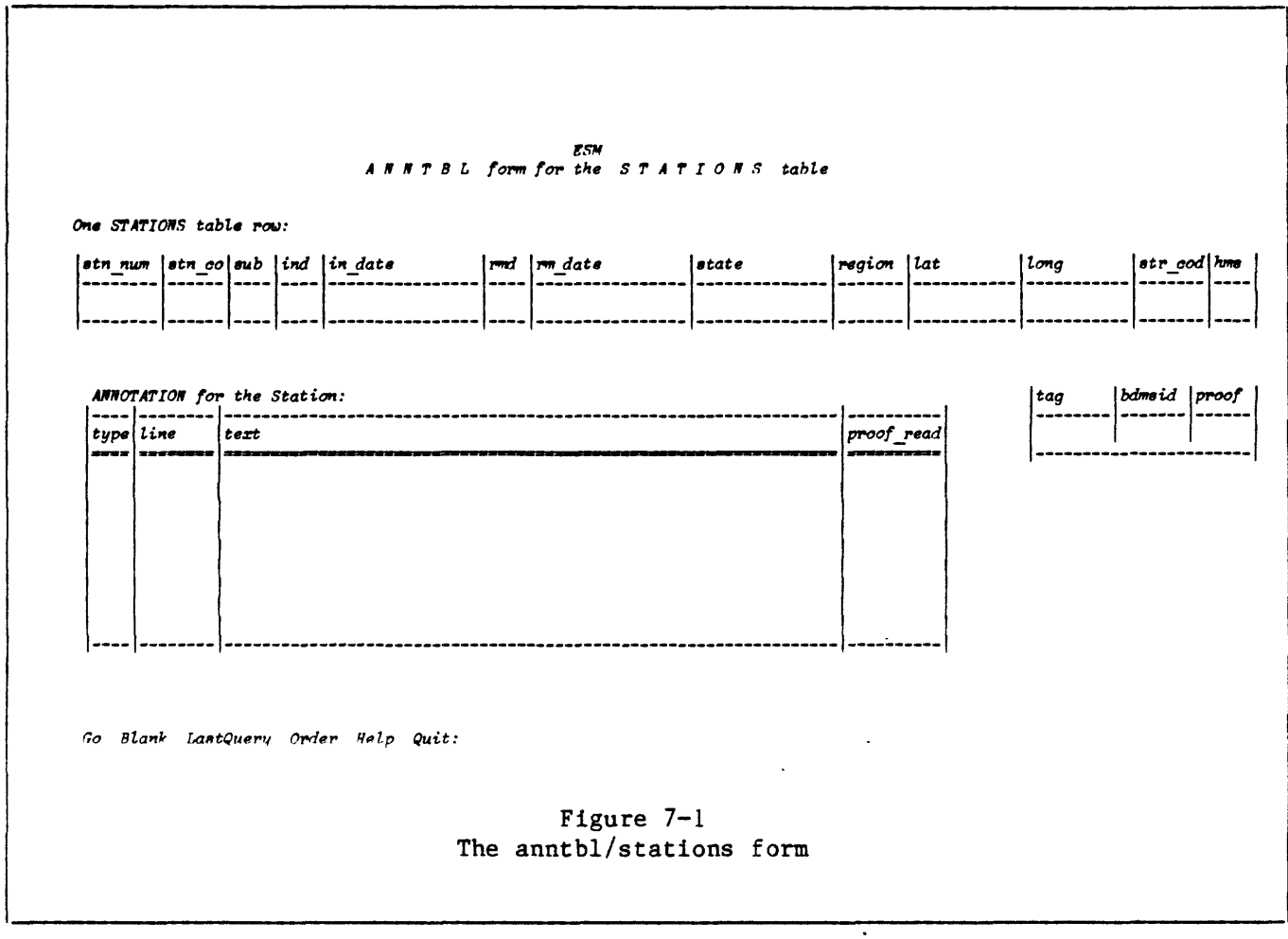


Figure 7-1
The anntbl/stations form

Immediately after a user types *$ESMFORMS anntbl stations*, the blank form illustrated in figure 7-1 will appear on the screen. The cursor will be resting on the first field, ready for the user to either type something into that field, to move the cursor to another field, or to move the cursor down to the menu at the bottom of the screen.

You may fill in various fields of the blank form to indicate which information you want to see. For instance, if you wished to see the *anntbl* display for station 2002, you would type "2002" in the *stn_number* field on the form, then select the menu option "go" to ask INGRES to fill out the rest of the form. To do so, of course, you must know how to move the cursor around to different areas of the form. There are a variety of ways to do this (see section 7.6), but the most straightforward method is to use the "nextfield" and "menu" keys. Each time you press the "nextfield" key, the cursor will move into a new field which you may, or may not, choose to fill in. The "menu" key is used to move the cursor out of the data fields and down to the

menu at the bottom of the screen where you can select one of the menu items shown (by typing "go", for instance).

The "nextfield" key is the TAB key on most terminals, but on VT100 compatible terminals, it is the ,-key on the keypad that sits to the right-hand side of the standard keyboard. (The ,-key is in the middle row of the keypad, at the right-most edge.) The "menu" key is the one labeled ESC or ESCAPE on most terminals, but on VT100 compatible terminals, it is the PF1-key on the keypad (upper left-most keypad key).

The menu list may be so long on some forms and on some terminals that it will not fit across the screen. If you wish to see another part of such a list, press the "menu" (ESC or PF1) key twice. The forms, too, can be too wide to fit across the screens on some terminals. If you wish to see a part of a form that is not currently on your screen, press the "nextfield" (TAB or keypad-,) repeatedly, until the portion of the form you wish to see has scrolled onto the screen.

7.4 Menus[1]

Query-By-Forms, as used by the *$ESMFORMS* command, has two states, the "Query" state and the "Go" state. While in the Query state, the user specifies a query by filling in fields on the form. Then the user may request that Query-By-Forms enter the "Go" state and retrieve the specified rows. In the Go state, the user can browse the rows Query-By-Forms retrieved. After browsing some or all of the retrieved rows, the user can then return to the Query state to specify a new query.

The layout of the form remains the same regardless of whether Query-By-Forms is in Query or in Go state, but the menu at the bottom of the form is different for the two states. The Query state menu appears as follows:

Go Blank LastQuery Order Help End:

This menu presents the user with the following choices:

option	function:
Go	Execute the query as specified on the form.
Blank	Clear the form af all input information in the fields.
LastQuery	Place the contents of the previously run query

[1] Most of the information in this section and the next section was copied or paraphrased from the "INGRES Query-By-Forms User's Guide".

	on the form for review or editing.
Order	Establish the ordering of rows returned by the query. If user selects this option, a blank form will appear. User should then type a number (1, 2, 3, ...) into each field by which the results should be sorted. The number 1 indicates the first sort field, the number 2 the second sort field, and so forth.
Help	Get help about this form or about the various special keys that are currently in effect.
End	Return to the main menu (if there is one) or to the $-prompt.

The options provided in the Go state menu vary slightly, depending on the format the user selected, but it will appear somewhat as follows:

NextMaster Query Help End:

This menu presents the user with the following choices:

option	function:
NextMaster	Retrieve the next screen of data.
Query	Return from Go state back to Query state.
Help	same as above.
End	same as above.

Once you have pressed the "menu" (ESC or PF1) key to position the cursor immediately behind the colon (:) on the menu line, you can indicate your choice by typing the name of the option and then pressing the RETURN key. You need not type the entire name: the first letter or first few letters will suffice. When you type the name (or abbreviation) of a menu option, you can do so in upper or lower case.

7.5 Queries Based on Inequalities and Partial Matches

Queries based on equality are expressed by simply entering values into one or more of the fields of a form. You can also specify queries based on inequality merely by preceding a value in the field with one of the comparison symbols <, <=, >, >=, or *!=*.

symbol	meaning
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal
!=	not equal
=	equal

Fields containing only values with no comparison operator are assumed to have an = operator.

You can also express multiple terms for comparison within a field. Query-By-Forms supports the logical operators *and* and *or* as well as the use of parentheses within a field. For instance, the *stn_number* field in the form shown in Figure 7-1 could accept the following:

>7000 or (!=1035 and < 2000)

The *in_date* field could accept:

>= 1-jun-1985 and <=28-jun-1985

Even though a field's display window is delimited on the screen, a qualification expression such as the two examples above can exceed the field length. When you enter a long qualification into a field, the qualification string scrolls within the field display window. You can move forward and backward within the qualification string by using the arrow keys or other appropriate cursor movement keystrokes (see next section) so that you can modify the characters.

Query-By-Forms supports the use of wild-card characters (*, *?*, *[...]*) for partial pattern matching. The wild card characters are described in Appendix B, section B.3.6.3, but in brief:

*	matches any string of zero or more characters
?	matches any single character
[...]	matches any of the characters in the brackets.

For example, the *text* field in the *annotation* table illustrated in Figure 7-1 could accept:

*"*dam*" or "*Dam*"*

to retrieve any stations having the word "dam" anywhere in any of its annotation. Searching through the *text* column in the *annotation* table as illustrated above is very time consuming, however. It is best to avoid doing so if the required information can be retrieved based on the fields in the parent table (*stations* in this case). In addition to being time-consuming, searches based on annotation values do not return as much information as those based on the parent table. In the first case, the *anntbl* display for any one station would show a complete *stations* table row, but only those *annotation* table rows for that station that had the word "dam" somewhere in the *text* field. In the second case, all *annotation* table rows associated with the parent *stations* table row would be displayed.

7.6 Cursor-moving Keys and Menu Option Keys[2]

Although the "nextfield" key (TAB or keypad-,) and "menu" key (=ESC or PF1) are the only special keys that are needed to make use of the forms, other special keys are available that can minimize the number of keystrokes the user must make. Many of the special keys are control keys (see Chapter 6, section 6.3). On VT100 terminals, the arrow keys that sit above the standard keyboard and the 18 keys in the keypad at the right of the standard keyboard also have special meaning with the ESM forms.

When the *ESM.DEF* set-up definitions are invoked (see Chapter 6), one of two sets of special keys are defined for use with *$ESMFORMS* and related commands. One set is implemented for the VT100 terminals that are commonly used with VAX machines, the other, more limited, set is implemented for other types of terminals. Other sets of key definitions may be added in the future. These two sets of key definitions are somewhat different than the definitions made by default by the INGRES system. If you are using several INGRES databases, you may prefer to keep the default key definitions. To revert to the INGRES defaults, insert

$DEFINE INGRES_KEYS "SYS_INGRES:[INGRES.FILES]VT100F.MAP"

or

$DEFINE INGRES_KEYS "SYS_INGRES:[INGRES.FILES]FRS.MAP"

immediately after the *$ @PUB2:[ESM]ESM.DEF* line in your *LOGIN.COM* file. The first definition, *VT100F.MAP*, is for VT100 terminals, the second, *FRS.MAP*, is for all other terminals.

The user can learn which special keys are currently defined for his terminal by selecting the "help" option included in any of the forms menus and then selecting the "keys" option. A list of key names and their corresponding functions will then be displayed. The ESM special key definitions are also presented below in tables 7-2, 7-3, and 7-4.

Some of the special keys are used as abbreviations for menu options. For example, on VT100 terminals the PF2 key on the keypad is equivalent to the "help" menu option. When a special key may be used to select a menu option, that special key is noted in parenthesis after the option on the menu line. The

---

[2] Much of the information in this section has been extracted from the "INGRES Terminal User's Guide."

line

Go Blank LastQuery Order Help Quit:

would be the first menu line to appear on the screen after a user typed *$ESMFORMS onetbl stations* if the user were at a terminal having no special menu keys defined for it. For a user at a VT100 terminal, however, that line will appear as:

Go(Enter) Blank(2) LastQuery(3) Order(4) Help(PF2) Quit(PF3):

Using one of the special keys noted in parentheses requires only one key-stroke as opposed to the three key-strokes that would be required to make the same request using the "menu" key. For instance, typing the "help" key (PF2 on a VT100) is equivalent to typing "menu" (PF1 or ESC), then H, then RETURN. Replacing three keystrokes with a single keystroke doesn't sound like much of an advantage in the abstract, but in practice it makes the typing much more convenient.

The menu keys are indicated in an abbreviated notation in the parentheses on the menu lines. Note that when a single number is shown (i.e. "(4)"), it indicates the VT100 keypad key marked with that number, not the key on the standard keyboard. Note too, that control keys (i.e., control-G) are noted with a preface of "^" (i.e., "(^G)") rather than spelling out the word "control-" as is done in this report.

7.6.1 Special Keys on VT100 Terminals

On VT100 terminals and VT100 compatible terminals (this includes VT220s), the arrow keys that sit above the standard keyboard can be used to move the cursor right or left, up or down. These terminals also have a set of 18 function keys that sit in a keypad at the right of the standard keyboard. All of the keypad keys have been given special meaning for use with the ESM and SMSM databases: They are illustrated in Figure 7-2.

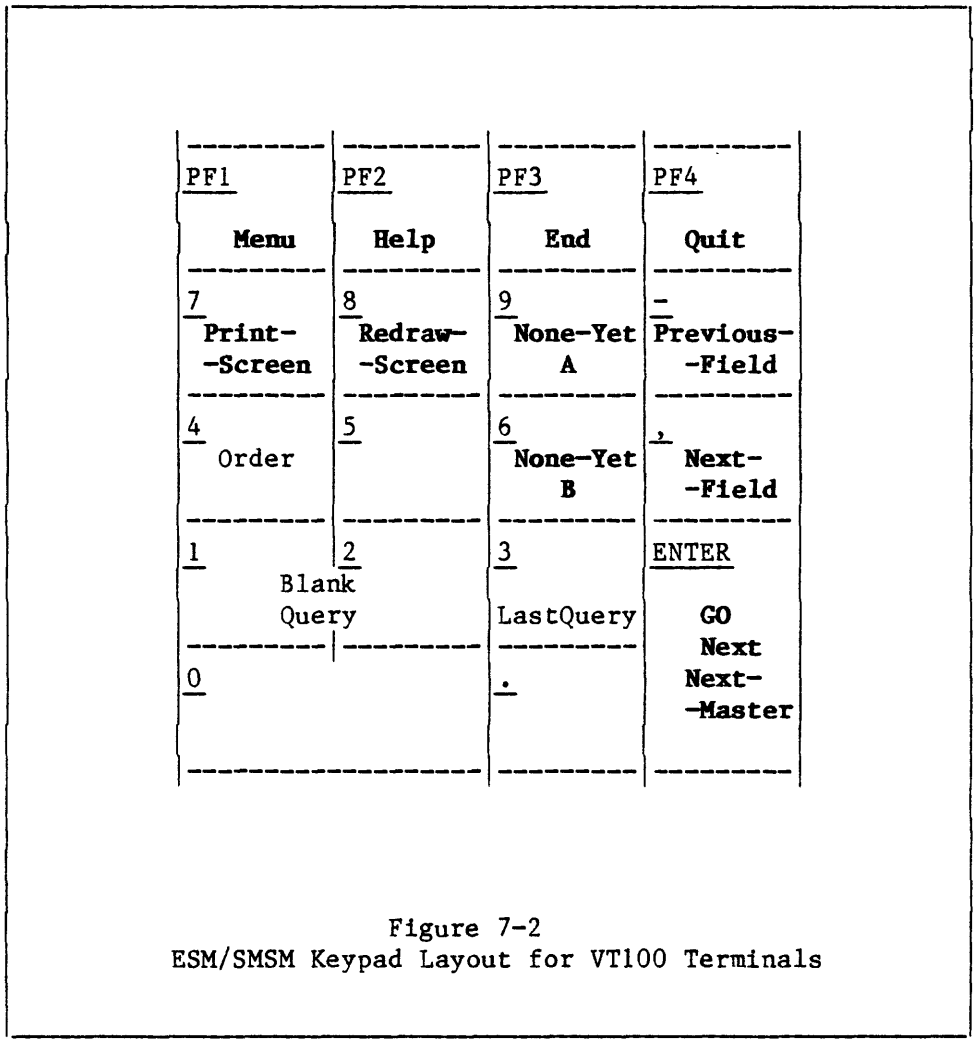


Figure 7-2
ESM/SMSM Keypad Layout for VT100 Terminals

Here follows a brief description of the function of each key shown in figure 7-2. Those keys indicated with a leading asterisk (*) are defined differently for ESM forms than they are in the default INGRES key definitions.

Keypad Label	Function Name	Function
PF1	Menu	Move cursor to the end of the menu line. If this key is pressed twice in succession when the menu line is too long to fit across the screen, a new portion of the menu line will scroll onto the screen.
PF2	Help	Get help about the current form or the current special keys.
PF3	End	End current form and return to previous form. If already at the top-most form (as is usually the case in retrieve-

		only activities), "End" works just as "Quit" would.
PF4	Quit	End forms activity, returning to the $- prompt.
* -	PreviousField	Move cursor to the previous field.
* ,	NextField	Move cursor to the next field. This moves the cursor to the first field to the right of, or down from, the current cursor position. Note that the arrow keys cannot be used in place of this key to move the cursor from one field to another, they can only move right and left within the current field or up and down within a table field.
ENTER	GO	Go, execute the query specified on the form, or execute current function.
	NextMaster Next	Retrieve the next screen full of data. (On some menus, this appears as "Next" rather than as "NextMaster".)
.	Undo/Forget	(This key is only used in update/maintenance activities.)
0	Save	(This key is only used in update/maintenance activities.)
1	Menuitem1	See Menuitem 5.
2	Menuitem2	"
3	Menuitem3	"
4	Menuitem4	"
5	Menuitem5	Some menu options, like "Blank" and "Query", are not always associated with the same key. They move around depending on their sequence in the menu list. The moving menu options used in retrieve-only activities are the following:
	Blank	Clear the form of all input information in the fields.
	Order	Specify the ordering of rows to be returned by the query.
	LastQuery	Place the contents of the previously run query on the form for review or editing.
	Query	Exit the "Go" state and return to "Query" state.
* 6	None-Yet-B	This key is undefined at present and is reserved for future use. See None-Yet-A.
* 7	PrintScreen	"Print" (copy really) the contents of the current screen to a disk file. User will be prompted for the name of the disk file to recieve the copy. Note that if a disk file of that name

		already exists, the copy will be appended to whatever is already in the file. So if you are using one of the visitors accounts, use file names that are unique and based on your name, *SueJones.tmp*, for example.
* 8	RedrawScreen	Redraw the screen. This is useful if your current form gets distorted by a message from some other user or by noise in the transmission lines between your terminal and the computer.
* 9	None-Yet-A	This key, like the None-Yet-B key, is undefined at present and is reserved for future use. The two keys will be used with the BASIC forms, when and if those forms are ever constructed, to move the cursor from one table field to another.

There are other special keys on the VT100 terminals in addition to those on the keypad shown in Table 7-2. These additional keys are shown in Table 7-3. Of these, the cursor moving arrows, the DELETE key, control-N, and control-E are the most useful. In Table 7-3, the keys that are defined differently for ESM/SMSM forms than they are in the default INGRES key definitions are shown with a leading asterisk (*). Note that Table 7-3 does not show the control keys like control-C and control-T that are recognized by the VAX operating system rather than by INGRES (See Chapter 6, section 6.3 for those keys.)

arrow keys:

->	Move left one character in a field.
<-	Move right one character in a field.
↓	Move down one line.
↑	Move up one line.

keys on the standard keyboard:

RETURN	Move to next field, clearing everything to the right of the cursor in the current field.
DELETE or DEL or RUBOUT	Delete the character immediately to the left of the cursor.
control-D	Delete the character under the cursor.
control-N	Newrow: move cursor to first column of next row in a table field. If cursor is not in a table field, then control-N works as "NextField".
control-E	Switch between overstrike and insert typing mode
* control-I	Scroll up: display the previous screen or set of rows in a table field.
* control-P	Scroll down: display the next screen or set of rows in a table field.
control-L	Scroll page or form left.
control-H	Scroll page or form right.
control-U	Nextword: move up one word within field.
control-R	Previousword: move back one word within field.

Keys that don't seem to work as the "INGRES Terminal User's Guide" says they should, but may eventually work, once the bugs are worked out:

control-K	Top: move cursor to top of form or table field.
control-J	Bottom: move cursor to bottom of form or table field.
control-F	Find: Search current column of table field for a specified string.
control-X	Clear out the rest of the field.

Keys used only in update/maintenance activities:

control-A	Auto-duplicate value while in fill mode. Unfortunately, this key doesn't work in table fields. It duplicates the same value in the current field as the field contained in the last screen.
control-V	Start the default text editor on the field.

Table 7-3
ESM forms manipulating keys on VT100 Terminals

7.6.2 Special Keys on Terminals Other than VT100s

If you are not using a VT100 terminal, the arrow and keypad keys (even if the terminal has such) will not work as they do on the VT100s. Some control keys and other special keys can be used to simplify the typing, though. These keys are indicated in Table 7-4.

ESC	Menu: move cursor immediately to the right of the ":" that follows the menu.
TAB	Move the cursor to the next field. This moves the cursor to the first field to the right of, or down from, the current cursor position.
control-I	move cursor to next field (=TAB)
control-P	move cursor to previous field (This is like a backwards TAB.)
control-N	New row: move to the first column of the next row in a table field. If cursor is not in a table field, control-N works as does TAB.
control-H	Move left one character in a field. (equivalent to the left arrow on VT100s)
control-L	Move right one character in a field.
control-J	Move down one line.
control-K	Move up one line.
control-B	Move cursor to next word
control-R	Move cursor to previous word
control-X	Print screen. (This doesn't work -- yet. It may in the future.)
control-W	Redraw the screen
control-E	Switch between overstrike and insert typing modes
RETURN	Move to next field, clearing everything to the right of the cursor
DELETE or DEL or RUBOUT	Delete the character to the left of cursor.
control-D	Delete the character under the cursor
control-F	scroll down on the form (by a full page)
control-G	scroll up on the form (by a full page)
control-O	scroll left on a form
control-U	scroll right on a form
control-A	duplicate the same value in the current field as it had in the last screen (unfortunately, this doesn't work in table fields).
control-V	Start the default text editor on the field.

Table 7-4
Forms Manipulating Keys for other than VT100 terminals

CHAPTER 8

Introduction to QUEL, the INGRES Query Language

8.1 Overview

QUEL, the INGRES Query Language, provides a rich set of commands that allow a user to specify how data in an INGRES database shall be manipulated. INGRES provides several interfaces between the user and QUEL but the most direct interface for the terminal user is the INGRES Terminal Monitor. (The EQUEL preprocessors are other such interfaces, but they are used as the interface between QUEL and various programming languages like FORTRAN.) Two sets of commands are presented in this Chapter: QUEL commands and Terminal Monitor commands. The QUEL commands dictate how the data in the database shall be manipulated while the Terminal Monitor commands manipulate QUEL commands.

Those QUEL and Terminal Monitor commands used for retrieval requests are briefly presented in this chapter and are described in detail in Appendixes B and C. Many examples of QUEL usage with the ESM database are given in Chapter 10.

ESM users may not need to read the detailed instructions in Appendixes B and C very carefully in order to learn how to address their retrieval requests to the ESM database, for the examples in Chapter 10 may be sufficiently self-explanatory to make the detailed instructions unnecessary. On the other hand, those users who wish to see even more detail than is given in the appendixes to this report may refer to the "INGRES Reference Manual" and the "INGRES/QUEL Self-Instruction Guide" (both manuals are in the first three-ring binder belonging to the 4-volume set of INGRES documentation).

8.2 QUEL commands

The essential QUEL commands used for retrieval are *range* and *retrieve*. Here is an example of a retrieval request:

```
range of r is records
retrieve (r.all)
   where r.edate >= "01-jan-1984"
     and r.edate  < "01-jan-1985"
```

These two QUEL statements will retrieve all columns of all the rows in the *records* table that contain an event date in 1984.

The *print*, *help*, *save*, *delete*, *destroy*, *replace*, *append*, *modify* and *define view* commands are also used occasionally in retrieval requests (with roughly that order of importance). The *save*, *delete*, *destroy*, *replace*, *append* and *modify* commands, however, are ordinarily used only on temporary tables created by the user (with a *retrieve into* command). INGRES allows only a few users who have special privileges to use *save*, *delete*, *destroy*, *replace*, *append* or *modify* on the 12 permanent ESM tables.

Here is a brief description of what each of the commands mentioned above does:

QUEL command name	function
range	The *range* statement assigns a range variable name to a specific table.
retrieve	The *retrieve* command fetches all rows that satisfy the qualification phrase, qual, and either displays them on the terminal (the default) or, when the *retrieve into* form of the command is used, stores them in a new table. The results can be sorted by the values of one or more columns. A lot of information about qual phrases is given in Appendix B.
print	The *print* command displays the contents of the table(s) specified.
help	The *help* command is used to display information about the contents of the database, about specific tables in the database, about INGRES features, or information about protections, permissions, and view definitons.
save	The *save* command is used to preserve user-created tables (those created with *retrieve into* commands) until the specified expiration date. Only the owner of a table can *save* that table.

delete	The *delete* command removes rows that satisfy the qualification, qual, from a table.
destroy	The *destroy* command removes tables from the database. Only the table owner may destroy a table.
replace	The *replace* command replaces values of specified columns in specified rows of a table.
append	The *append* command adds new rows to a table.
modify	The *modify* command will remove duplicate rows from a table and establish an index to the table so that subsequent *retrieve*-s on that table may execute efficiently. When the access method specified in a *modify* statement is *isam*, as is suggested throughout this report, the table is sorted so results from subsequent *print* statements will appear in the same order as specified for the keys in the *modify* statement.
define view	The *define view* command establishes a virtual (i.e., not actual) table that is made up of sections of existing tables. Once defined, a view-name can be used in place of a table-name in *range* and *retrieve* statements. The syntax of the *define view* statement is nearly identical to the *retrieve into* command.

The syntax for these few, frequently-used QUEL commands is summarized below, in table 8-1; the syntax for all the QUEL commands is summarized in Appendix A; and the syntax for the table 8-1 commands is described in detail in Appendix B.

8.2.1 WARNINGS!!!

Many users may not need to read Appendix B in order to be able to construct their own QUEL commands, for the summary in table 8-1 and the examples in Chapter 10 may provide adequate instruction. All users should be aware of some of the warnings that are buried in Appendix B, however, so they are repeated here.

- The ESM database administrator may remove tables users have created with *retrieve into* commands after a week's time unless the user has issued *save* statements for the tables. It is good courtesy to *delete* your own tables once you no longer need them, for if you leave them as

clutter in the database, the database administrator must go to extra bother to remove them.

- It is good practice to use "*modify* table-name *to isam* ..." statements on user-created tables that will subsequently be used in *retrieve* statements. Without appropriate indexes, INGRES can, in some situations, thrash around for an indefinitely long time in an effort to satisfy a *retrieve* that threads through several tables.

- Do not use *append* statements on any tables other than those in "heap" structure. There will be no diagnostic, but the *append* just won't happen. (This bug may have disappeared in recent INGRES versions). Tables created with *retrieve into* commands have a heap structure by default, but once a table has been *modify*-ed to a different structure, it should be *modify*-ed back to heap before any *append*s are made to it. Use: "*modify* table-name *to heap*".

- The *retrieve into* command will allow you to construct a table having several columns of the same name, but you should avoid doing so. Such a table cannot be manipulated well in subsequent commands and causes maintenance problems for the database administrator. See appendix B, section B.2.2, for an example of how to avoid this problem.

- Do not mix default and explicitly declared range variables within a single QUEL statement. See the discussion of "disjoint queries" and range variables in the next section, the warinings about the *delete* statement in Appendix B, section B.2.6, and the warnings about the *replace* statement in Appendix B, section B.2.8.

  Don't: *delete e where events.edate > "01-jan-1990"*

  Do: *delete events where events.edate > "01-jan-1990"*
  or: *delete e where e.edate > "01-jan-1990"*

8.2.2 Syntax of Frequently used QUEL Commands

The syntax of the QUEL commands listed above is described in detail in Appendix B and is summarized in this section in table 8-1.

Within table 8-1, words shown in italics are keywords recognized by QUEL and words shown in regular type represent words or phrases the user must provide. The phrases shown underlined and in bold type (**bold**) indicate user-supplied components of the commands that are explained below table 8-1 or in appendix B. For instance, there is no need to explain what

the "table-name" component of many QUEL statements is, but an explanation is required for the "**sort-list**" component. The vertical bar (|) indicates a choice, the square brackets ([]) indicate that the enclosed material is optional, and the curly brackets ({}) indicate any number (including zero) occurrences of the enclosed material. Note that you do not type the bar or brackets while constructing an actual command. Some of the commands are shown in table 8-1 in a more simple form than they are shown in Appendix A where infrequently used options are also shown.

range of **range-variable** *is* table-name

retrieve [*into* table-name|*unique*] (**target-list**)
[*where* **qual**] [*sort by* **sort-list**]

print table-name {, table-name}

help [*view*|*permit*|*integrity*] [table-name {,table-name}]

save table-name *until* month day year

delete **range-variable** [*where* **qual**]

destroy table-name

replace **range-variable** (**target-list**) [*where* **qual**]

append to table-name (**target-list**) [*where* **qual**]

modify table-name *to isam* [*on* **sort-list**]

define view view-name (**target-list**) [*where* **qual**]

/* comments */

Table 8-1
QUEL Retrieval Commands

Names

All the names that a user chooses, such as range-variable, table-name, and view-name, should be 9 characters long or less.

range-variable

A range variable is a named variable that identifies a specific table or a set of rows within a table. The name of a table can be used as a range variable for that table, or an arbitrary and different range variable name may be assigned to a specific table with the *range* statement.

Subsequent to their declaration in a *range* statement, range variables are used within other QUEL statements as row markers that range over the table named in the range statement. For example, the statement

range of e is events

defines a range variable *e* that will take as its values rows in the table named *events* when used within QUEL statements like *retrieve*, *replace*, and *delete*. Within these statements, a specific row or set of rows in the *events* table may be associated with the range variable *e* through a qual (qualification) phrase. An example:

```
retrieve (e.all) where e.edate >= "01-jan-1984"
                   and e.edate <  "01-jan-1985"
```

In this statement, the range variable *e* ranges over just those rows in the events table that have an event date in 1984. If the *where* and the qualification phrase that follows it were not included, as in *retrieve (e.all)*, then *e* would range over all the rows in the events table.

Once declared, a *range* statement remains in effect until:

- The end of the QUEL session.
- The variable is redeclared by a subsequent *range* statement.
- The table is removed with the *destroy* command.
- More than 9 other range variables have been used or declared since the last use of the range variable in question.

Only ten range variables may be in effect at any time. After the tenth *range* statement, the least recently referenced range variable is supplanted by the next *range* statement. This limit of ten includes both default and explicitly declared range variables.

Range variables allow the QUEL user to treat one table as though it were several separate but identical tables. This is because each range variable assigned to a given table ranges over that table independently of any other range variables assigned to that same table. Use of several range variables that refer to the same table is necessary in some complex

queries, but such use often results in a very serious error that is called a "disjoint query" in the INGRES documentation. Disjoint queries usually arise when a user inadvertantly uses default and explicitly declared range variables (e.g. *events* and *e*) as though they were identical. Here, for example, is a *delete* statement that would delete all rows in the *events* table, not just those rows that have an event date of 01-jan-84:

```
    range of e is events
    delete e where events.edate = "01-jan-1984"          (Don't!)
or  delete events where e.edate = "01-jan-1984"          (Don't!)
```

This is the way it should be expressed:

```
    delete e      where      e.edate = "01-jan-1984"     (Yes!)
or  delete events where events.edate = "01-jan-1984"     (Yes!)
```

Most ESM database users are not permited to *delete* rows from tables other than those they created for themselves, so the *events* table and all the other permanent ESM tables are not really vulnerable to the problem just illustrated (except through mistakes made by the database administrator!), but user-created tables can easily be destroyed this way. Here is another example of the problem, in a *retrieve* statement this time:

```
    retrieve (e.ed_flag, events.edate)                   (Don't!)
or  retrieve (events.ed_flag, e.edate)                   (Don't!)
```

Either of the two *retrieve*s above would return the entire cartesian product of the events table with itself, not merely the *ed_flag* and *edate* values from all rows in the table. It should be done as follows:

```
    retrieve (     e.ed_flag,      e.edate)              (Yes!)
or  retrieve (events.ed_flag, events.edate)              (Yes!)
```

Two different range variables that refer to the same table should only be used within the same QUEL statement when the user genuinely wishes to join a table to itself. For example, the following *retrieve* statement would create a table containing a copy of those rows from the *stations* table that describe stations that were installed before station 1035 was installed.

```
range of s is stations
range of q is stations
retrieve into mytable (s.all)
   where s.in_date < q.indate
     and q.stn_number = 1035
```

Target-list

The target-list specifies which columns should be affected in an *append*, *retrieve*, or *replace* statement. The column names may be listed in any order; they are separated from one another by commas; and the list is enclosed within required parentheses. Each item in a target-list can be one of the following:

range-variable.*all*
range-variable.columnname
result-columnname = expression

The keyword "*all*" is an abbreviation for a list of all the columns of the table referenced by the range variable. For instance, if *e* is a range variable for the *events* table, *(e.all)* is equivalent to *(e.ed_flag,e.edate,e.eid,e,time,e.time zone,e.lat,e.long,e.mag,e.depth,e.tag,e.bdmsid,e.proof_read)*.

Each of the other two items specifies a single column: "range-variable.columnname" explicitly names an existing column, and "result-columnname = expression" creates a new column whose type accommodates the expression. See Appendix B, section B.3 for more information about QUEL expressions.

Some examples:
(e.edate)
(e.edate, newcolumn = "XYZ", e.elong)
(e.lat,e.long, s.lat,s.long, r.epi_d, rt.all)
(newvalue = max(a.ln))

Sort-list = columnname[:**sortorder**][{, columnname[:**sortorder**]}]

The sort list specifies the column(s) by which the results from a *retrieve* or *modify* request shall be sorted. If more than one column name is specified, they must be separated by commas. If more than one column name is specified in the sort-list, rows are sorted on the basis of the first column specified, and within values of that column, on the basis of values of the second column specified, and so forth. By default, rows are sorted in ascending (1,2,3,... a,b,c ...) order.

In *retrieve* statements (but not in most *modify* statements), you can override the default sort order by specifying a sortorder of *d* or *descending*. For completeness, you can also specify *a* or *ascending* for sortorder even though that is the default.

Qual

The qualification phrase, qual, follows the keyword *where* and identifies specific rows in a table. In QUEL, expressions and qualifications are recursively defined as follows:

A qualification, qual, is one of the following:

(qual)
not qual
qual *and*|*or* qual
expr comparison-operator expr

An expression, expr, is one of the following:

a constant
a column
(expr)
expr arithmetic-operator expr
function (expr)
aggregate-operator (expr
[*by* expr {,expr}]
[*where* qual])

Each of the qualification and expression components that are underlined above are explained in detail in Appendix B, section B.3. Here, however, are a few examples:

```
where s.lat > 50.0 and (s.state = "CA" or s.state = "NV")
where  1.0/c.period  > temptable.frequency
where  length(q.text) = max (length(q.text))
```

(note in the above example that *length* and *max* are functions. See Appendix B, section B.3.5)

```
where e.edate = rt.edate
  and e.eid = rt.eid  and  e.ed_flag = rt.ed_flag
  and rt.stn_number = s.stn_number
  and s.state = "ID"
```

8.3 Terminal Monitor Commands

The terminal monitor is an interface between the QUEL user and INGRES. The terminal monitor assists the user in entering and running QUEL commands from a terminal, and it allows the user to write, review, and edit QUEL commands prior to executing them.

The terminal monitor maintains a workspace that can be thought of as a blackboard. You enter QUEL commands into the workspace, examine them and, if necessary, change them. When you think the command is correct, you can tell the terminal monitor to pass your workspace to INGRES for execution. The terminal monitor then receives the results of your command(s) from INGRES and displays them on your terminal.

Whenever the terminal monitor is waiting for the user to type something, it prints an asterisk at the beginning of the current line. It also displays messages to inform you of its status. When it displays

```
go
*
```

or

```
continue
*
```

the terminal monitor is ready to accept commands. When the terminal monitor prints "*continue*," the workspace contains something, and when it prints "*go*," the workspace is empty. When it displays

```
Executing ...
```

the terminal monitor is telling you that INGRES is executing your command(s).

The terminal monitor supports a variety of commands that facilitate your session with INGRES, but there are eight basic commands with which you should be familiar. These eight commands appear in Table 8-2 below:

command	meaning
\r	*reset*/erase the workspace
\p	*print* the workspace
\e	invoke *editor* on the workspace
\g	execute query (*go*)
\a	*append* to the workspace
\q	*quit*/exit INGRES
\i filename	read/*include* the named file
script filename	log query and response to named file

Table 8-2
Terminal Monitor Commands

Each of the one-letter commands shown in table 8-2 can be entered in its one-letter form or with its full name. All terminal monitor commands require the backslash character.

For more information about each of these commands, refer to Appendix C. For examples of their use, refer to Chapter 10.

CHAPTER 9

Reports

9.1 Introduction

The format in which QUEL displays its results is quite inflexible, but QUEL results may be used as input to the *$ESMRPT* command to generate reports that display the information in various formats. The *$ESMRPT* command generates reports from data in the ESM database and the *$SMSMRPT* command generates similar reports from data in the SMSM database. The report formats available at present are named *"onetbl"*, *"anntbl"*, *"basic"*, and *"stnfix"* and other formats should be added to the *$ESMRPT/$SMSMRPT* capabilities in the future. Users may also create their own report generators using the INGRES Report-Writer sub-system. Instructions for creating new report layouts are given in the "INGRES/REPORTS: Report-Writer Reference Manual" (in Volume 3 of the INGRES documentation set).

To use one of the existing report generators, *basic* for instance, one first collects the information to be reported on into a temporary table using the QUEL *retrieve into* command, then runs the *$ESMRPT* command, naming on the *$ESMRPT* command line the name of the table to use for the report. Here, for example, follows a print-out of commands that could be used to generate the *basic* report for those stations located in the vicinity of the USGS offices in Menlo Park. The report will reside in a disk file named *tmp.tmp*.

```
$set verify
$!  this file is stored at PUB2:[ESM.EXAMPLES]EXAMPLE1.BAT
$ESMQUEL
  range of s is stations  \g
  retrieve into temptable (s.all)
           where s.lat  > 37.0   and s.lat  < 38.0
             and s.long >-122.0  and s.long <-120.0
  \g
  print temptable \g
  \q
$ESMRPT basic temptable stations tmp.tmp
$ESMQUEL
  destroy temptable  \g
  \q
```

These commands reside in the file at *pub2:[esm.esamples]example1.bat*. To request that the commands in that file be executed in interactive mode, a user could simply type:

```
$ @pub2[esm.examples]example1.bat
```

Or, preferably, one could run it in batch mode late at night by typing:

```
$submit/after=23:30 pub2:[esm.examples]example1.bat
```

Notice the *$ESMRPT* command line in the example above. Following the command name *ESMRPT*, the user must type, in order, the report format name (*basic* in this case), the report table name (*temptable* in this case), and then the name of the primary database table of which the report table is a subset (*stations* in this case). Following these required arguments, the sample *$ESMRPT* command above also specified an optional output file name (*tmp.tmp* in this case).

In general, the *$ESMRPT* command line has the form:

$ESMRPT **format report-table primary-table** [**disk-file**] [*nlpp=#*]

Where:

format = "*onetbl*", "*anntbl*", "*basic*", or "*stnfix*". The first three formats may be abbreviated to "*one*", "*ann*", or "*bas*", or even more briefly to "*o*", "*a*", or "*b*". Other formats should be added in the future.

report-table = the name of the user-created table that contains the information from which the report will be generated.

Except when format = "*onetbl*", **report-table** must be a copy of a subset of one of the 12 permanent tables, or at least have the same structure and column names as one of the 12 permanent tables. The report table may be wider than the permanent table, however, and contain user-constructed columns to the right of the columns copied from the permanent table. These extra columns may be used for sorting the table into the order the user prefers.

The sort order that exists in the report table will be preserved in the generated report.

When format = "*onetbl*", the report table may be any arbitrarily-structured, user-constructed table.

Rather than being a user-created subset of one

of the permanent tables, report-table may actually be one of the permanent tables, but that would make for an overly-long report!

primary-table = the name of one of the 12 permanent tables. This parameter informs the *$ESMRPT* command of the structure of report-table. When **report-format** is "*onetbl*", the report-table need not have the same format as one of the 12 permanent tables, in which case this parameter may be specified as "*none*".

disk-file This is an optional parameter that indicates the name of the disk file that will receive the report. If this parameter is missing, the report will be displayed at an interactive user's terminal (not very useful!), or written to a batch job's log file.

The output disk file (*tmp.tmp* in the example above) may be sent to the line-printer (*$print tmp.tmp*), displayed at the terminal (*$type tmp.tmp*), edited (*$edt tmp.tmp*), and manipulated and displayed in various other ways.

nlpp=# This optional and not (yet) very useful parameter indicates the number of lines per page to be used in the report. If this parameter is omitted, there will be no page-breaks or carriage-control characters in the output file. Since very little thought has been given to how we might want the trailing and heading text to appear at the top and bottom of each page, it is best to omit this parameter. The report paging may be improved in the future, however.

9.2 Report Formats

One other format, in addition to the fundamental *onetbl*, *anntbl*, and *basic* formats described in chapter 5 can be generated through the *$ESMRPT* command:

stnfix This is a combination of the basic/*stations* and the basic/*instruments* with some of the temporary cleanup tables (like *moreins*) added. This format is intended for use by those who are involved in the data cleanup effort.

Other formats may be added in the future:

stnlst This report generator should present station information in a format similar to that used in the "Station List" open file report. Once the

information in the database is proof-read and completed, We could use the *stnlst* reporter to generate updated versions of that report.

table1 This report generator should present record information in a format similar to that used in table 1 in the "Strong-Motion Program Report" circulars.

mail This should create the relatively unformatted print-outs to be mailed out in response to inquiries from people outside the Survey. The *mail* report should be rather like the *basic* report, but things should be labeled a little more nicely. And some of the book-keeping information, like *bdmsid* and *proof_read*, could be omitted. (We will use *basic* until such time as *mail* becomes available.)

allrecord It would probably be nice to have a report similar to the basic/*records* report which also showed all the event, station, and instrument information for each record too.

9.3 Warnings

For some of the reports, the *$ESMRPT* and *$SMSMRPT* commands create and read a temporary file on the scratch disk (located at *sys1:[scratch]*). This can lead to two problems:

1) The scratch disk may be too full to hold the file that *$ESMRPT* needs to create to do its work. If this occurs, either ask the VAX system manager (Howard Bundock) to clean out the scratch disk for you, wait until someone else complains (they will!), or wait until the scratch space is cleaned out in one of the automatic routine cleanup passes that occur daily and weekly. You can learn how many blocks are currently available on the scratch disk by typing "*$show dev scr:*".

2) There is a slight chance that several concurrently running reports may collide: one report generating process may write over the scratch file another wrote. If this happens (it is very unlikely: there is only a very short interval in which it can happen), simply rerun your report again.

CHAPTER 10

Examples

10.1 Overview

This chapter presents some samples of QUEL retrieval requests along with their supporting terminal monitor and VAX commands.

Many of the examples in this chapter illustrate how to retrieve information that is not easily requested through the ESM forms. These are queries that must thread through several tables such as when finding the *events* that triggered *records* in *stations* having specific characteristics.

If we find that some of these cross-table queries occur frequently enough, we may eventually set up some more elegant, simple to use, forms-based mechanisms for responding to them. Nevertheless, it is important to be able to retrieve information using QUEL, for there will always be occasions when ad-hoc queries arise that were not anticipated when the forms were designed. It would be a waste of our limited resources to try to set up forms that will accommodate all queries.

QUEL must also be used to construct the temporary tables that the *$ESMRPT* report-generating command requires as input. (See Chapter 9.)

While constructing QUEL commands for ESM, it's best to have a copy of the ESM summary diagram on-hand (unless you have a fantastic memory) in order to remember the names of all the tables and their columns. That diagram is shown in Chapter 2. It is also helpful to have a copy of Appendix A on-hand while working at a terminal.

All the examples shown in this Chapter are also kept on the VAX disk in the directory at *pub2:[esm.examples]*. Users are encouraged to make copies of these files and modify them for their own use.

10.2 Sample 1

The first sample is quite uncomplicated, for it deals with just a single column from a single table. The sample can be used to learn the various kinds of documentation available from the *doc* table. Here is a print-out of the *sample1.iql* file:

```
/*  this is PUB2:[ESM.EXAMPLES]SAMPLE1.IQL    */
   retrieve  unique (doc.kind) sort by kind
\p
```

Note the comment line in that file. QUEL comments are bounded on the left with "/*" and on the right with "*/". A user wishing to execute the commands in that file could use the *\r*, *\i* and *\g* terminal monitor commands, as follows:

```
*\r
*\i pub2:[esm.examples]sample1.iql
*\g
```

Here follows a print-out of what the screen would look like as the user typed the *$ESMQUEL* and those three lines. The characters that would be typed by the user are underlined and shown in bold type (**bold**); all the other characters are those that would be displayed by the computer.

```
$ ESMQUEL
INGRES VAX Version 3.0/23c (vax.vms/02) login
Copyright (c) 1985, Relational Technology Inc.
INGRES 3.0 Production Version

go
* \r
go
* \i pub2:[esm.examples]sample1.iql

/*    this is pub2:[esm.examples]sample1.iql       */
retrieve unique (doc.kind) sort by kind

continue
* \g

Executing . . .

|kind        |
|------------|
|agency      |
|array       |
|geology     |
|news        |
|recorder    |
```

```
|region      |
|state       |
|structure   |
|trans-loc   |
|------------|
(9 rows)

continue
* \g

INGRES VAX Version 3.0 logout 10-JUN-1986 14:47:40
Copyright (c) 1985, Relational Technology Inc.
goodbye -- come again
$
```

10.3 Sample 2

The second sample can be used to display news items that are in the *doc*umentation table. This is a rather useful sample, for it is actually good practice to run it periodically (or its *$ESMFORMS* forms equivalent) to see if any changes have been made to the database that you are not yet aware of. Here is a print-out of the *sample2.iql* file:

```
/*  this is PUB2:[ESM.EXAMPLES]SAMPLE2.IQL    */
\script tmp.tmp
range of d is doc
retrieve (d.code, d.text)
   where d.kind = "news"
\p
\g
```

Also, since the \g command is in the *sample2.iql* file, users wishing to execute the commands in the file do not need to type the \g, they would only need to type the \r and \i commands, as follows:

```
*\r
*\i pub2:[esm.examples]sample2.iql
```

Since the \g command is in the *sample2.iql* file, sample2 can also be invoked from VAX command level, as well as from the terminal monitor, as follows:

```
$smsmquel <pub2:[esm.examples]sample2.iql
```

The '<' symbol on that *$smsmquel* command line indicates that the terminal monitor should take its commands from the file named after the '<' rather than from the user's terminal.

10.4 Sample 3

The third sample is more complicated. It shows commands that could be used to retrieve all the stations that are located on or near dams. Ideally, one would only need to retrieve those *stations* table rows having "DM" as their *str_code* value, but since the *str_code* column, like many others in the database, is not completely filled in yet, more investigation is required. In this example, not only are stations with *s.str_code = "DM"* retrieved, but also any stations having the word "dam" anywhere in their annotation. And stations having "ACOE" (for Army Corps of Engineers) in their source annotation. And stations having "ACOE" as the instrument owner. And stations containing transducers with the *t.loc_desc* suggesting a dam site.

```
$on control_Y then goto stopit
$on warning  then goto stopit
$set verify
$!
$!  this is PUB2:[ESM.EXAMPLES]DAMS.BAT.
$!
$!  Find all the stations that have anything to do with
$!    dams.
$!  This is used as sample #3 in Chapter 10 of the ESM
$!    user's Guide.
$!  It takes 30 seconds of CPU time and 2 hours of elapsed
$!    time.
$!
$SET DEF SYS1:[SCRATCH]
$ESMQUEL
/*

    First, get numbers of all stations that have structure
      code = "DM" into a temporary table named "myname"

 */
destroy myname \g
range of s is stations \g
retrieve into myname (s.stn_number)
   where s.str_code = "DM"
\g
/*

   Next, all stations that have "dam" anywhere in their
     annotation:

 */
range of a is annotation \g
append to myname (s.stn_number)
   where   (   a.text = "*dam*"
```

```
            or a.text = "*Dam*")
      and   s.tag  = a.parent_tag
\g
/*

   All stations having "ACOE" (Army Corps of Engineers) as
      source annotation:

 */
append to myname (s.stn_number) where
            a.text = "*ACOE*"
      and   a.type = "s"
      and   s.tag  = a.parent_tag
\g
/*

   All stations having "ACOE" as the instrument owner:

 */
range of i is instruments \g
append to myname (i.stn_number)
   where i.owner = "ACOE"
\g
/*

   All stations containing transducers with the location
     description suggesting a dam site.

  */
range of t is transducers \g
append to myname (t.stn_number) where
            t.loc_desc = "*dam*"
      or    t.loc_desc = "*Dam*"
      or    t.loc_desc = "*abut*"
      or    t.loc_desc = "*Abut*"
      or    t.loc_desc = "*crest*"
      or    t.loc_desc = "*Crest*"
      or    t.loc_desc = "*allery*"
\g
/*

   remove duplicate numbers from the station number list:

 */
help myname \g
modify myname to isam on stn_number   \g
/*

Now list all the stations table rows that match the
  numbers left in the temporary (myname) table:

*/
range of q is myname   \g
retrieve (s.all) where  s.stn_number = q.stn_number  \g
```

```
destroy myname \g
/*   */
\q
$EXIT      != normal end
$STOPIT:
$write sys$output "Stopping -.bat due to error or cntrl-Y"
$STOP      != error end
```

There were 266 stations table rows retrieved by this example on 17jun86; 206 of them had *s.str_code = "DM"*.

Some of the techniques used in this example should be noted:

- User-Created tables

  INGRES will not allow you to create a table if a table of the same name already exists, even if the pre-existing table is exclusively owned by another user. The "myname" table created with the *retrieve into myname ...* statement above must not already exist in the database if the *retrieve* is to work. It is good practice for each user to select user-created table names that are unique and unlikely to be selected by other users. The user's own name and variations thereof are good choices for temporary table names. When choosing a name, make sure it is less than 9 characters long: INGRES will accept longer names, but it only recognizes the first 9 characters.

  User-created tables will stay in the database for at least a week after they are created, so it is good practice to destroy your temporary tables when you are through using them. It is also good practice to use the *destroy* <tablename> statement just before creating a table, just in case you created a table of the same name in an earlier session and forgot to *destroy* it thereafter. INGRES will not allow you to *destroy* a table unless you are the owner of that table or you have special privileges.

- Annotation searches

  Searching through the *annotation* table as is done in several of the steps in the above example is very time consuming. This example took several hours of elapsed time and 30 seconds of CPU time when it was run during the day with a normal load on the machine. Since it is so time-consuming, it is best to run something like this as a batch job, and even better to run it as a batch job late at night when there won't be any interactive users.

- Wild-card characters

  Notice the wild-card character, "*", used in some of the string searches. The "*" matches any string of characters. For more information about this and other

wild-card characters, see Appendix B, section B.3.6.3.

- Upper and Lower Case Characters

  Care must be taken in specifying upper or lower case in string searches. For example, the agency code "ACOE" was given in the example above in capital letters since that is the way all agency codes occur in the database. No matches would be found for the string "acoe".

  Other strings do not occur as systematically as the agency codes which are always in upper case. Upper and lower cases occur rather haphazardly in many of the *annotation* table rows. Consequently, the user must make allowance for a word occurring with or without its first letter capitalized. If a word is rather long, like "gallery", the first letter can be omitted in the search string, as in *"*allery*"*, but if a word is short or otherwise indefinite without its first character, it should be specified in two forms, one with the capitalized first letter and one without, as in "*Dam*" and "*dam*".

- *Modify* Statements

  The *modify* statement in the above example was used to remove duplicate rows from the table and to establish an index to the table so the *retrieve* statement that follows may execute efficiently. It is always good practice to *modify* user-created tables that will subsequently be used in *retrieve*s: without appropriate indexes, INGRES can, in some situations, thrash around for an indefinitely long time in an effort to satisfy a *retrieve* that threads through several tables.

  When the access method specified in a *modify* statement is *isam*, as it is in this example, the table is sorted so results from subsequent *print* statements will appear in the same order as specified for the indexes in the *modify*.

10.5 Sample 4

The fourth example shows another time-consuming request that is best perfomed in batch mode. Although it takes only 60 seconds of CPU time, it can take several hours of elapsed time if run during the day when, as usual, other people are also using the machine. Suppose a user is suspicious that there is more than one station in California named Anderson Dam or something similar (there are four). The commands listed below could be used to retrieve all the stations named "Anderson" or "Andersen" in California, plus all the records taken at those stations and all the events recorded. The commands in this example display the resulting information through the *basic*, *stnfix*, and *onetbl* report generators.

```
$on control_Y then goto stopit
$on warning   then goto stopit
$set verify
$!
$!  This is PUB2:[ESM.EXAMPLES]ANDERSON.BAT
$!
$!  Use it to learn about any stations named "Anderson"
$!    (or ..."sen") in California.
$!
$SET DEF SYS1:[SCRATCH]
$ESMQUEL
/*

    Get stations into mynameS:

*/
destroy mynames \g
range of s is stations \g
range of a is annotation \g
retrieve into mynames (s.all)
         where  s.tag = a.parent_tag
           and  s.state = "CA"
           and  (   a.text = "*nderson*"
                 or a.text = "*ndersen*")
\g
modify mynames to isam on stn_number \g
print mynames \g
/*

    Get records into mynameR:

*/
destroy myname \g
range of rt is traces  \g
range of q is mynames  \g
retrieve into myname (rt.all)
```

```
        where (rt.stn_number = q.stn_number)
\g
modify myname to isam on edate,eid,rtype,rsn  \g
/*  */
range of q is myname \g
range of r is records   \g
destroy mynamer   \g
retrieve into mynamer (r.all,q.stn_number)
     where r.edate = q.edate
       and r.ed_flag = q.ed_flag
       and r.eid   = q.eid
       and r.rtype = q.rtype
       and r.rsn   = q.rsn
\g
modify mynamer to isam on edate, eid, rtype,rsn  \g
/*

    now get event info into mynameE:

*/
destroy mynamee \g
range of q is mynamer \g
range of e is events  \g
retrieve into mynamee (e.all)
     where e.edate = q.edate
       and e.ed_flag = q.ed_flag
       and e.eid  = q.eid
      sort by edate,eid
\g
\q
$!!
$!   Now make reports:
$!!
$ESMRPT ONETBL myname traces
$ESMRPT STNFIX mynames stations
$ESMRPT BASIC mynames stations
$ESMRPT BASIC mynamer records
$ESMRPT BASIC mynamee  events
$!!
$!   Cleanup:
$!!
$ESMQUEL
destroy myname   \g
destroy mynames  \g
destroy mynamer  \g
destroy mynamee  \g
\q
$!!
$!   Done
$!!
$EXIT    != normal end
$STOPIT:
$write sys$output "Stopping -.bat due to error or cntrl-Y"
$STOP    != error end
```

10.6 Retrieve Examples

The *retrieve* command is the most important component of QUEL. More examples of *retrieve* statements follow below. Note, however, that some of the examples would not yield very complete results since many of the database columns are not completely filled in yet.

1) Find all records from events --
   1a) of particular magnitude (e.g., 6.5 to 7.5):

```
retrieve into temptable (r.all)
   where r.edate = e.edate and r.eid = e.eid
  and   r.ed_flag = e.ed_flag
   and   e.mag>= 6.5 and e.mag <=7.5
\g
```

   1b) with epicenter within a given lat/long rectangle:

```
retrieve (r.all)
  where r.edate = e.edate and r.eid = e.eid
    and r.ed_flag = e.ed_flag
    and e.lat > 38.0 and e.lat < 40.0
    and e.long> -122.0 and e.long < -120.0
\g
```

2) Find all records from stations --
   2a) within a particular state (Idaho, in this example):

```
retrieve unique (rt.ed_flag,rt.edate,rt.eid,
                 rt.rtype,rt.rsn,rt.stn_number,rt.sub_stn)
  where rt.stn_number = s.stn_number
    and s.state = "ID"
   sort by edate
\g
```

   2b) within a given lat/long rectangle:

```
retrieve unique (rt.ed_flag,rt.edate,rt.eid,
                 rt.rtype,rt.rsn,rt.stn_number,rt.sub_stn)
  where rt.stn_number = s.stn_number
    and s.lat > 38.0 and s.lat < 40.0
    and s.long> -122.0 and s.long < -120.0
\g
```

2c) of a particular type of structure:

```
retrieve unique (rt.ed_flag,rt.edate,rt.eid,
                 rt.rtype,rt.rsn,rt.stn_number)
where rt.stn_number = s.stn_number and s.str_code = "DM"
\g
```

2d) having particular geologic site conditions:

```
retrieve unique (rt.ed_flag,rt.edate,rt.eid,
                 rt.rtype,rt.rsn,rt.stn_number,rt.sub_stn)
  where rt.stn_number = s.stn_number
    and s.state = "NV"
    and s.tag = a.parent_tag
    and a.type = "g"  and  a.text = "*lluvium*"
\g
```

A better way to do this, since user is probably going to want to see the station info as well as the record info, would be:

```
retrieve into myname (s.all)
   where s.state = "NV"
   and   s.tag = a.parent_tag
   and   a.type = "g" and a.text = "*lluvium*"
\g
modify myname to isam on stn_number \g
print myname  \g

range of q is myname \g
retrieve unique (rt.ed_flag,rt.edate,rt.eid,
                 rt.rtype,rt.rsn,rt.stn_number,rt.sub_stn)
   where rt.stn_number = q.stn_number
   sort by edate
\g
```

3) Find all events that produced records --
3a) with a particular peak acceleration:

```
retrieve unique (e.all)
   where e.edate = rt.edate  and  e.eid=rt.eid
   and   e.ed_flag = rt.ed_flag
   and  rt.peak >= 0.4
  sort by edate
\g
```

3b) ... and which records were subsequently digitized:

```
   ...
   and   ra.edate = rt.edate  and  ra.eid=rt.eid
   and   ra.ed_flag = rt.ed_flag
   and   ra.rtype   = rt.rtype
```

```
   and   ra.rsn      = rt.rsn
   ...
```

4) Find all events that triggered records in stations --
   4a) in a particular state:

```
retrieve  (e.all)
   where e.edate = rt.edate
     and e.eid  = rt.eid  and  e.ed_flag = rt.ed_flag
     and rt.stn_number = s.stn_number
     and s.state = "ID"
    sort by edate
\g
```

   4b) of a particular structure type:

```
        ... and s.str_code = "DM"
```

5) Find all stations that have produced records from events --
   5a) having epicenters within a particular lat/long rectangle:

```
retrieve  (s.all)
   where s.stn_number = rt.stn_number
     and rt.edate = e.edate
     and rt.eid = e.eid  and  rt.ed_flag = e.ed_flag
     and e.lat > 38.0 and e.lat < 40.0
     and e.long> -122.0 and e.long < -120.0
    sort by stn_number
\g
```

   5b) ... and having a particular magnitude:

```
        ...
        and e.mag >=5.0 and e.mag <=6.0
        ...
```

6) Find all stations that have produced records --
   6a) at a particular epicentral distance:

```
retrieve (s.all)
   where s.stn_number = rt.stn_number
   and  rt.edate =r.edate
   and  rt.eid =  r.eid  and  rt.ed_flag =r.ed_flag
   and   r.epi_d <= 10.0  and r.epi_d > 0.0
   sort by stn_number
\g
```

Note that to satisfy this request, we must either keep

the epicentral distance in the *records* table, as is done now, or we must add a new function to the ESM support software that will calculate epicentral distance on the fly. If we had such a function, the phrase *and r.epi_d <= 20.0* shown above would be replaced with one like:

```
   and new-function(e.lat,e.long,s.lat,s.long)  <= 20.0
```

6b) with a particular peak acceleration:

```
retrieve (s.all)
   where s.stn_number = rt.stn_number and rt.peak >= 0.6
\g
```

7) Find instrument information about a specific record (one taken for the San Fernando EQ by the RFT/108 recorder which was situated at the Isabella dam at the time), especially the period, damping, and sensitivity for all 3 traces on that record:

```
retrieve (c.all)
   where c.rtype =  "RFT*"  and c.rsn = 108
     and c.cdate <  "09-nov-1971"
    sort by cdate
\g
```

And things can get even more complicated than in the examples just shown by requesting them in combination with one another.

10.7 Other Examples

A collection of sample files is maintained in the VAX directory at *pub2:[esm.examples]*. The collection includes copies of all the examples shown in this Chapter and more. More examples should be added to the collection as time passes. The sample files there have suffixes of *.iql* or *.bat*. The suffix "*.iql*" indicates that the file contains QUEL and terminal monitor commands: iql for ingres query language. These *.iql* files can be included in a user's *$ESMQUEL* or *$SMSMQUEL* session by typing:

```
\i pub2:[esm.examples]whatever.iql
```

The suffix "*.bat*" indicates that the file contains VAX commands in addition to QUEL commands. The *.bat* files can be executed in batch mode or in indirect interactive mode. To request that a *.bat* file be run in batch mode late at night, type:

```
$ submit/after=23:30    pub2:[esm.examples]whatever.bat
```

To run one in indirect interactive mode, use:

```
$ @pub2:[esm.examples]<whatever>.bat
```

Users are encouraged to make copies of the files in *pub2:[esm.examples]* and modify them for their own use. A table of contents for all the important sample files in that directory is in the file at *pub2:[esm.examples]examples.nts*. Copies of all the sample files have been collected together in a form suitable for printing in the file at *pub2:[esm.examples]examples.all*

APPENDIX A:

Summary of Commands

Brief summaries of the various commands discussed in this report are presented in this appendix. Included in these summaries are some QUEL and Terminal Monitor commands that are not discussed elswhere in this manual: these are commands that only need to be used by those involved in database administration.

The following notations are used in the command summaries:

notation	meaning
A|B	A or B
[A]	A is optional
{A}	Zero or more occurrences of A
italics	Words printed in italics are key words having special meaning to INGRES or to the VAX operating system.
normal type	Words shown in the command examples that are not in italics represent phrases that the user must provide.
whatever	Bold, underlined phrases represent phrases that the user must provide and that are described in more detail below.

A.1 To Get Started:

$ @pub2:[esm]esm.def
or: *$ @pub2:[esm]esm.def* termname

For more information about the *esm.def* file and its use, refer to Chapter 6.

A.2 ESM Commands

For more information about these commands, refer to Chapters 6, 7, and 9.

$esmforms format table
$esmupdate format table
$esmquel
$esmquel <disk-file
$esmrpt format report-table primary-table [disk-file] [*nlpp=#*]

$smsmforms format table
$smsmupdate format table
$smsmquel
$smsmquel <disk-file
$smsmrpt format report-table primary-table [disk-file] [*nlpp=#*]

$s132
$s80

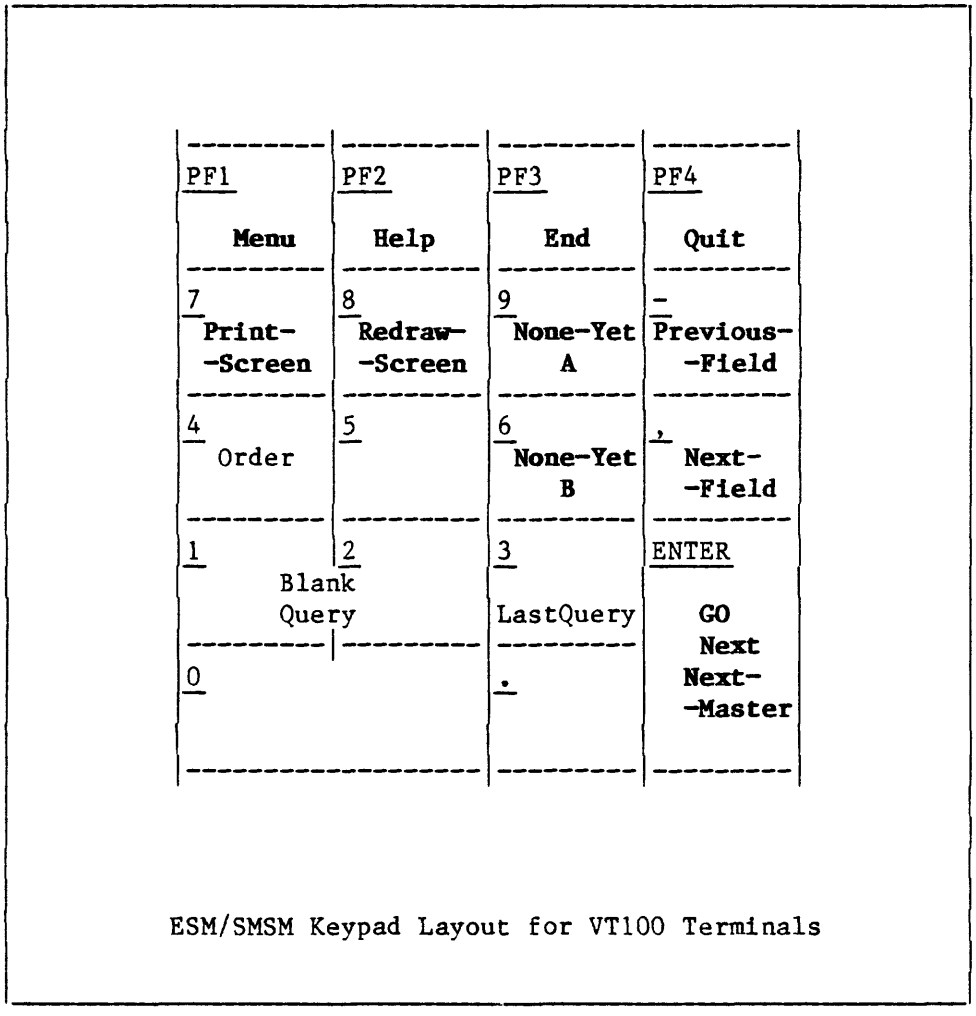
A.3 Control Keys

For more information about the control keys, refer to Chapter 6, section 6.3, and to Chapter 7, sections 7.6.1 and 7.6.2.

control-c (interrupt the current process)
control-y (don't use this with database processes !!!)
control-t (display process information)

A.4 Forms Manipulating Keys on The VT100 Keypad

For more information about the function of the various keys in the keypad on VT100 terminals, refer to Chapter 7, section 7.6.1.



ESM/SMSM Keypad Layout for VT100 Terminals

A.5 Terminal Monitor Commands[1]

For more information about these commands, refer to Chapter 8, section 8.3; Appendix C; or the INGRES Reference Manual.

\r	reset/erase query buffer
\p	print query buffer
\l	print query buffer after macro expansion
\v	evaluate macros
\e	invoke editor on query buffer
\g	execute query (go)
\a	append to query buffer
\time	print time and date
\date	print time and date
\s	invoke command executive
\q	quit/exit INGRES
\cd dir_name	change default directory
\i filename	read/include the named file
\w filename	write query buffer to named file
\branch	transfer control to *\mark*
\mark	label for *\branch*
\script filename	log query and response to named file.

A.6 QUEL Commands

For more information about these commands, refer to Chapter 8, Appendix B, or the INGRES Reference Manual.

A.6.1 QUEL Data Manipulation Commands

abort [*to* savepoint-name]
append [*to*] tablename (**target-list**) [*where* **qual**]
begin transaction
define view view-name (**target-list**) [*where* **qual**]
delete range-var [*where* **qual**]
end transaction
help [*view*|*permit*|*integrity*] [tablename {,tablename}]
print tablename {,tablename}
range of range-var *is* tablename {, range-var *is* tablename}
replace range-var (**target-list**) [*where* **qual**]
retrieve [*into* tablename|*unique*] (**target-list**)
[*where* **qual**] [*sort* [*by*]] **sort-list**
savepoint savepoint-name
/* comment */

---

[1] Most of the information that follows in this appendix was copied from the INGRES quick-reference card provided along with the 4-volume INGRES documentation set.

A.6.2 QUEL Database Administration Commands

copy tablename (columnname=format
{,columnname=format}) *from*|*into* "filename[,type]"
create tablename (columnname=format{,columnname=format})
define integrity on range-var *is* **qual**
define permit oplist *on*|*of*|*to* range-var [(columnname
{,columnname})] *to* username [*at* terminal] [*from* time
to time] [*on* day *to* day] [*where* **qual**]
destroy [*permit*|*integrity*] tablename [integer
{,integer}|*all*]
index on tablename *is* indexname (columnname
{,columnname})
modify tablename *to* **storage-structure** [*on* **sort-list**]
[*where* [*fillfactor*=n] [, *minpages* =n] [, *maxpages*=n]]
save tablename *until* month day year
set [*journaling*|*nojournaling*] [*on* tablename]
set [*joinop*|*nojoinop*]
set ret_into "[*heap*|*cheap*|*heapsort*|*cheapsort*|
hash|*chash*|*isam*|*cisam*]"
set [*nodeadlock*|*deadlock*]

Target-List

The target list specifies which columns should be affected in an *append*, *retrieve*, or *replace* statement. The column names may be listed in any order and are separated from one another by commas. See Chapter 8, section 8.2.2 for more information.

Sort-list = columnname[:**sortorder**] [{,columnname[:**sortorder**]}]

The sort list specifies the column(s) by which the results from a *retrieve* or *modify* request shall be sorted. If more than one column name is specified, they must be separated by commas. By default, rows are sorted in ascending (1,2,3,... a,b,c ...) order. You can override this default by specifying a **sortorder** of *d* or *descending*. For completeness, you can also specify *a* or *ascending* for **sortorder** even though that is the default.

Note that in *modify* statements, a **sort-list** should not be given for those structures (*heap*, for instance) for which sorting would be inappropriate.

Storage-structure

isam	indexed sequential
cisam	compressed isam
hash	random hash
chash	compressed hash
heap	unkeyed and unstructured
cheap	compresed heap
heapsort	sorted heap with no duplicates
cheapsort	compresed heapsort
btree	b-tree structure
cbtree	compressed b-tree structure

Qual

The qualification phrase identifies specific rows. See section B.3 of Appendix B, or for even more information, see section 1.4 of the INGRES Reference Manual.

A.6.3 QUEL Partial Pattern Matching (used within qual expressions)

*	matches any string of zero or more characters
?	matches any single character
[a-z,$]	matches any characters in brackets

A.6.4 QUEL Aggregate Functions (used within qual expressions)

count(...)	count occurrences
countu(...)	count unique occurrences
sum(...)	sum
sumu(...)	sum unique values
avg(...)	average
avgu(...)	average unique values
max(...)	maximum
min(...)	minimum
any(...)	value is 1 if any rows qualify, else 0

A.6.5 Functions (used within qual expressions)

abs(n)	absolute value of *n*
ascii(n)	numeric to character conversion
atan(n)	arctangent of *n*
concat(a,b)	concatenates *a* and *b*
cos(n)	cosine of *n*
date(expr)	converts character or text string to date data type
dow(expr)	reports day of week of given date
exp(n)	exponential of *n*
float4(n)	converts *n* to f4
float8(n)	converts *n* to f8
int1(n)	converts *n* to i1
int2(n)	converts *n* to i2
int4(n)	converts *n* to i4
interval(u,d)	converts date interval of floating-point number of user-specified time units
left(c,n)	returns *n* leftmost characters of string *c*
length(c)	returns length of string *c*, not counting trailing blanks
locate(c1,c2)	returns the location of the first occurrence of *c2* within *c1*
log(n)	natural logarithm of *n*
lowercase(c)	converts all upper case characters in string *c* to lower case
mod(n,b)	modulo
money(n)	converts character, float, or integer to money data type
right(c,n)	returns *n* rightmost characters of string *c*
shift(c,n)	shifts string *c* *n* places to the right if *n*>0 and to the left if *n*<0
sin(n)	sine
size(c)	returns the declared size of *c* without removing trailing blanks
squeeze(c1)	compresses white space
sqrt(n)	square root of *n*
text(expr)	converts expr to text string
trim(c1)	returns *c1* without trailing blanks
_bintim(0)	current time (internal format)
_time(n)	convert internal time to hh:mm
_date(n)	convert internal date to dd-mmm-yy

A.6.6 Symbolic Constants (used within qual expressions)

dba	dba code for database
usercode	current user's code
username	current user's name
_version	INGRES version number

APPENDIX B

QUEL Syntax[1]

B.1 Overview

This appendix explains the QUEL (INGRES Query Language) statements that were introduced in Chapter 8. Section B.2 presents the *range*, *retrieve*, *print*, *help*, *save*, *delete*, *destroy*, *replace*, *append*, *modify* and *define view* statements. All of these QUEL statements are summarized in Chapter 8, Table 8-1. Other QUEL statements that are used only for database administration and that should have no utility for the general user are summarized, along with the Table 8-1 statements, in Appendix A, Sections A.6.1 and A.6.2. Those who do need to make use of the database administration commands can refer to the INGRES Reference Manual for more explanation than is given in Appendix A.

The QUEL statements presented in this appendix are given in roughly the order of their importance. Many users will only need to make use of the *range*, *retrieve*, *print* and *help* statements, for most of the other statements (*save*, *delete*, *destroy*, *replace*, *append* and *modify*) are ordinarily used only on temporary tables created by the user (with a *retrieve into* command). INGRES allows only a few users who have special administrative privileges to use *save*, *delete*, *destroy*, *replace*, *append*, or *modify* on the 12 permanent ESM tables.

Section 8.3 presents the many possible components of the qualification phrase that can be included in the *retrieve*, *delete*, *replace*, *append*, and *define view* statements. The qualification phrases are shown as **qual** in the summaries given in Table 8-1 and Appendix A.

---

[1]Most of the information in this appendix has been copied or paraphrased from various portions of the INGRES documentation, most of it from the INGRES Reference Manual.

B.2 Frequently-Used QUEL Commands

B.2.1 The *range* statement and range variables

Syntax: *range of* range-variable *is* tablename

The range statement assigns a range variable name to a specific table. Example: *range of e is events*. Range variables, or "row markers" as they are sometimes called in the INGRES documentation, are used in qual qualification phrases (see section B.3) to mark a specific row or set of rows in a table. An expression like range-variable.columnname is used within qual phrases to denote the intersection of a single row and a single column in a table.

The table names themselves can be used as range variable names, in which case a *range* statement is not needed, but it is often convenient to assign a shorter range variable name with the *range* statement. For instance, it is easier to type "e" than to type "events" to specify rows in the events table in an expression such as:

```
range of e is events
retrieve (e.all)
   where  e.edate >= "01-jan-1984"
     and  e.edate  < "01-jan-1985"
     and  e.lat > 40.0 and e.lat <50.0
     and  e.long > -122.0 and e.long < -120.0
```

Another benefit of *range* statements is that you can declare more than one range variable for a table. Although this can be a benefit and in fact necessary for some complex queries, it can also be the source of confusing or disastrous results. If two range variables are assigned to the same table, they each refer to a different copy of the table. For example, the pair of statements below returns the entire cartesian product of two copies of the *events* table, not merely the *edate* and *elat* values from each row:

```
range of e is events
retrieve (e.edate, events.lat)
```

Other examples of the possible misuse of several range variables that refer to the same table are given in section B.2.6 in the discussion of the *delete* statement and in Chapter 8, Section 8.2.2 in the discussion of *range* variables and "disjoint queries".

The definition of a range variable can be changed at any time: *range of e is events* then: *range of e is mytable* leaves the range variable *e* assigned to the user-created table

named *mytable*.

Once declared, a *range* statement remains in effect until:

- The end of the QUEL session.
- The variable is redeclared by a subsequent *range* statement.
- The table is removed with the *destroy* command.
- More than 9 other range variables have been used or declared since the last use of the range variable in question.

Only ten range variables may be in effect at any time. After the tenth *range* statement, the least recently referenced range variable is supplanted by the next *range* statement. This limit of ten includes both default and explicitly declared range variables.

B.2.2 *retrieve*

Syntax: *retrieve* [*into* tablename|*unique*] (target-list)
[*where* qual] [*sort by* sort-list]

The *retrieve* command fetches all rows that satisfy the qualification phrase, qual (see section B.3), and either displays them on the terminal (the default) or stores them in a new table. The results can be sorted by the values of one or more columns.

If the "*into* tablename" is specified, the result of the query is stored in a new table having the indicated name and owned exclusively by the user who invoked the *retrieve into*. A table of the same name must not already exist. The new table will have column names as specified in the target-list. The table will remain in the database for at least a week. It can be saved for an even longer time through a *save* statement. (See section B.2.5)

If the "*unique*" is given in a *retrieve* statement, rows are sorted on all the columns in the target-list (beginning with the first column), and duplicate rows are removed before being displayed.

If no "*into* tablename" clause is specified, then the result of the query is displayed on the terminal and is not saved in any table. Unless the key word "*unique*" is specified or a "*sort by*" clause is given, duplicate rows are not removed when the result is displayed on the terminal.

Some *retrieve* examples:

```
retrieve (e.edate)
retrieve (e.edate) where e.lat > 50.0
retrieve unique (e.edate) where e.lat > 50.0
retrieve (e.edate,e.lat) where e.lat > 50. sort by edate,elat
retrieve into mytable (e.all) where e.lat > 50.0
retrieve (value = max (a.ln))
retrieve (q.text) where length(q.text) = max(length(q.text))
retrieve unique (e.all) where
   where e.edate = rt.edate
     and e.eid = rt.eid  and  e.ed_flag = rt.ed_flag
     and rt.stn_number = s.stn_number
     and s.state = "ID"
    sort by edate
```

Many more *retrieve* examples are given in Chapter 10.

WARNINGS about user-owned tables created via the *retrieve into* statement:

- The ESM database administrator may remove user-created tables after a week's time unless the user has issued *save* statements (see section B.2.5) for the tables. It is good courtesy to *delete* (see section B.2.6) your own tables once you no longer need them, for if you leave them as clutter in the database, the database administrator must go to extra bother to remove them.

- It is good practice to use "*modify* table-name *to isam* ..." statements (see section B.2.10) on user-created tables that will subsequently be used in *retrieve* statements. Without appropriate indexes, INGRES can, in some situations, thrash around for an indefinitely long time in an effort to satisfy a *retrieve* that threads through several tables.

- The *retrieve into* statement will allow you to construct a table having several columns of the same name, but you should avoid doing so. Such a table cannot be manipulated well in subsequent commands and causes maintenance problems for the database administrator. For example, one could, but should not, create a table having two columns named *oldname* with the following statement:

```
retrieve into mytable
   (y.oldname, z.oldname)                ( <<< Don't!)
   where ...
```

The correct way to do this would be to give one of the columns a new name, *newname* for instance:

```
retrieve into mytable
   (newname = y.oldname, z.oldname)      (Yes!)
   where ...
```

B.2.3 *print*

Syntax: *print* tablename {,tablename}

The *print* command displays the contents of each table specified at your terminal.

B.2.4 *help*

Syntax: *help* [*view*|*permit*|*integrity*] [tablename {,tablename}]

The *help* command is used to print information about the contents of the database, about specific tables in the database, about INGRES features, or information about protections, permissions, or view definitions. INGRES/QUEL will recognize the following forms of the help command:

help	Lists all user tables (i.e., all tables other than those the INGRES system uses to keep track of its own machinations) that exist in the current database.
help tablename	Gives information about the specified table(s).
help all	Equivalent to "*help* tablename" for all the (user) tables.
help view viewname	Prints view definition of specified view(s).
help permit tablename	Prints permissions on specified table(s).
help integrity tablename	Prints current integrity constraints on specified tables.

B.2.5 *save*

Syntax: *save* tablename *until* month day year

The *save* command is used to preserve tables until the given expiration date. Only the owner of a table can *save* that table.

The month can be an integer from 1 through 12, or the name of the month, either abbreviated or spelled out. Day is simply the day of the month, and year is the fully specified year (i.e., 1982 or 1999).

Example: *save mytable until feb 28 1987*

B.2.6 *delete*

Syntax: *delete* range-variable [*where* qual]

The *delete* command removes rows that satisfy the qualification, qual, from the table referred to by range-variable. The range-variable must have been declared to range over an existing table in a previous *range* statement, or it must be the default range-variable name provided by INGRES (= the table name).

If the qualification, qual, is not given in a *delete* command, the command deletes all rows in the table. The result is a valid but empty table.

To delete rows from a table, you must either be its owner or have *delete* permission on the table.

Here are two examples of *delete* statements:

```
delete mytable
delete e where e.proof_read = "TMP"
           or e.proof_read = "ZZZ"
           or e.proof_read = "zzz"
```

BEWARE:
Do not mix default and explicitly declared range variables in a *delete* statement. Mixing range variables usually results in a disjoint query, whose results are never what you want, and often ruinous to valuable data. Suppose user has created a table named *mytable* that contains a subset of the *events* table. Here is an example of a *delete* that would empty the entire *mytable* table even though that is probably not what the user intends:

```
range of q is mytable
delete mytable where q.edate < "01-jan-1980"      (Don't!)
```

Or equivalently:

```
delete q where mytable.edate < "01-jan-1980"      (Don't!)
```

Here is the way it should be expressed:

```
      delete q where q.edate < "01-jan-1980"            (Yes!)
Or:   delete mytable where mytable.edate < "01-jan-1980"  (Yes!)
```

B.2.7 *destroy*

Syntax: *destroy* tablename

The *destroy* command removes tables from the database. Only the table owner may *destroy* a table. Note that a table can be emptied of rows, but not destroyed, by using the *delete* command.

Example: *destroy mytable*

B.2.8 *replace*

Syntax: *replace* range-variable (target-list) [*where* qual]

The *replace* command replaces values of columns in a table. It changes the values of the specified columns in the target-list for all rows in the table referred to by range-variable that satisfy the qualification, qual. Only columns to be modified need appear in the target-list.

Numeric columns may be replaced by values of any numeric type. Replacement values are converted to the type of the result columns.

Only the owner of a table, or a user with *replace* permission on the table, can replace values.

For example, here follow two *replace* statements that were used in the early completion effort on the *transducers* table.

```
replace t (ttype = "acc") where t.ttype = " "

replace t (sub_stn = q.new_substn)
  where q.stn_number = t.stn_number
    and q.old_substn = t.sub_stn
    and q.rmd_flag   = t.rmd_flag
    and q.rm_date    = t.rm_date
```

WARNING:
Do not mix explicitly declared range variables with those that INGRES provides by default. Mixing range variables usually results in a disjoint query, which can seriously corrupt your data. (See the similar warning at section B.2.6 with the *delete* statement.)

B.2.9 *append*

Syntax: *append to* tablename (target-list) [*where* qual]

The *append* command adds rows to a table that satisfy the qualification, qual. The target-list specifies the columns to be appended to the table. The columns may be listed in any order. Columns of the result table which do not appear in the target-list are assigned default values of 0 (for numeric columns) or blank (for character columns).

Expressions of any numeric type or value may be used to set the value of a numeric type column. Conversion to the result column type takes place. Numeric values cannot be directly assigned to character columns, however: Conversion from numeric to character can be done by using the *ascii* operator or the *text* function (see section B.3). Character values cannot be directly assigned to numeric columns. Use the *int1*, *int2*, etc. functions to convert character values to numeric (see section B.3).

The keyword "*all*" can be used when you want to *append* all columns of one table, X, to another table, Y, but only when the column names of table X are the same as the column names of table Y. Example: *append to Y (X.all)*

An *append* may only be used by the owner of the table or a user with *append* permission on the given table.

Examples:
Simply add a row to an existing table:

```
append to arrays(array_name = "mugwump", stn_number= 9999)
```

Duplicate the text in an existing set of *annotation* rows into a new set of *annotation* table rows having a new *parent_tag* value:

```
range of a is annotation
append to annotation
        (parent_tag = 1234, a.type, a.ln, a.text)
        where a.parent_tag = 5678
```

Append the entire *mytable2* table onto *mytable1*. (Note that the two tables must have the same structure in this case):

```
range of q is mytable2
append to mytable1 (q.all)
```

WARNING:
Do not use *append* statements on any tables other than those

in "heap" structure. There will be no diagnostic, but the *append* just won't happen. (This bug may have disappeared since INGRES version 4.0 was installed.) Tables created with *retrieve into* statements have a heap structure by default, but once a table has been *modify*-ed to a different structure, it should be *modify*-ed back to heap before any *appends* are made to it. Use: "*modify* table-name *to heap*".

B.2.10 *modify*

Syntax: *modify* tablename *to isam* [*on* sort-list]

The *modify* command will sort a table, remove duplicate rows from the table, and establish an index to the table so any subsequent *retrieves* on that table can execute efficiently. It is always good practice to *modify* temporary tables that will subsequently be used in *retrieves*: without appropriate indexes, INGRES can, in some situations, thrash around for an indefinitely long time in an effort to satisfy a *retrieve* that threads through several tables.

The keyword "*isam*" is only one of several storage structures that can be specified in a *modify* statement. Other storage structures are available (see Appendix A or the INGRES Reference Manual), but *isam* is probably the best choice for use with user-created temporary tables.

Only the owner of a table may *modify* that table.

Here is an example of a *modify* statement: *modify events to isam on edate, eid*

For more examples of *modify* statements, refer to the file at *PUB2:[ESM.MAINT]INDEXES.IQL*. It contains all the *modify* statements that are applied to the 12 permanent tables in the ESM database.

B.2.11 *define view*

Syntax: *define view* view-name (target-list) [*where* qual]

The *define view* command establishes a virtual (i.e., not actual) table that is made up of sections of existing tables. Once defined, a view-name can be used in place of a table-name in *range* and *retrieve* statements. The sytax of the *view* statement is nearly identical to the *retrieve into* command.

In INGRES, tables that are created with a *retrieve into* command have direct representations in storage. These tables are called "base tables". In contrast to a base table, a view is not directly represented in storage. Instead, a view is defined in terms of base tables, and that definition is stored by the INGRES system. When you issue a query on a view, INGRES modifies the query so that it actually works on the base tables.

An example:

```
define myview (e.lat,e.long,r.all)
        where e.edate   = r.edate
          and e.ed_flag = r.ed_flag
          and e.eid     = r.eid
          and r.epi_d   < 20.0
```

B.3 QUEL Expressions and Qualifications[2]

In QUEL, phrases called expressions are components of other QUEL phrases called target-lists (explained in Chapter 8, Section 8.2.2) and qualifications (explained in this section). QUEL expressions and qualifications are recursively defined as follows:

An expression, expr, is one of the following:
- a constant
- a column
- (expr)
- expr arithmetic-operator expr
- function (expr)
- aggregate-operator (expr
  [*by* expr {,expr}]
  [*where* qual])

A qualification, qual, is one of the following:
- (qual)
- *not* qual
- qual *and*|*or* qual
- expr comparison-operator expr

Each of the qualification and expression components that are underlined above are explained in detail in the subsections that follow.

B.3.1 Constants

Constants are valid expressions. Examples:

"the quick brown fox"
1209
7.77
1.4e-23
"01-jan-1932"

---

[2] All the material in this section, except for minor changes, was copied from section 1.4 of the INGRES Reference Manual.

B.3.2 Columns

An actual column, properly denoted, is also an expression. Example: *e.edate*

B.3.3 Parentheses

An expression can be enclosed in parentheses, such as *(e.edate)* or *("the quick brown fox")*, without affecting its meaning.

B.3.4 Arithmetic Operations

B.3.4.1 Arithmetic Operators

Expressions of numeric types can be combined arithmetically to produce other expressions. INGRES supports the following arithmetic-operators (in descending order of precedence):

+,-	plus, minus (unary)
**	exponentiation
*,/	multiplication, division
+,-	addition, subtraction (binary)

Unary operators group from right to left, and binary operators group from left to right.

Parentheses can force the desired order of precedence. For example,

(s.region + s.lat) / 15

is a (nonsensical) expression in which the precedence is unambiguous.

The + operator can also be used to concatenate character or text strings. For example,

"This " + "is " + "a " + "test."

gives the value

"This is a test."

When used in this fashion, the + operator behaves exactly as the *concat* function described in Section B.3.5.2 of this appendix.

B.3.4.2 Arithmetic Operations on Dates

INGRES supports a limited set of arithmetic operations on items of the date data type:

Addition

date interval	+	date interval	->	date interval
date interval	+	absolute date	->	absolute

Subtraction

date interval	-	date interval	->	date interval
absolute date	-	absolute date	->	date interval
absolute date	-	absolute date	->	absolute date

An absolute date is an expression like *"01-july-86"*, and although none are used in any of the 12 permanent tables in ESM, INGRES also allows dates that include the time of day as well, as in *"01-july-1986 12:32:48"*. Date intervals, like the time of day, are not used in any of the 12 permanent ESM tables either, but users may want to use them in their QUEL expressions; they are expressions like *"23:38 hours"* or *"-800 years"*.

INGRES does not support multiplication or division of date values.

INGRES also enables you to convert date constants into numbers of days relative to an absolute date. For example, to convert today's date to the number of days since January 1, 1900, use the following command:

```
retrieve (num_days =
        int4(interval("days", "today" - date("1/1/00"))))
```

To convert back, use the following command:

```
retrieve (todays_date =
        (date("1/1/00") + concat(ascii(num_days), "days")))
```

where *num_days* is the number of days added to the date constant; and *init4*, *interval*, *date*, *concat*, and *ascii* are functions that are explained in section B.3.4.5.

B.3.4.3 Numeric Type Conversion

When two numeric expressions are combined, INGRES converts as necessary to make the storage formats (i.e., data types and widths) identical. The resulting expression then has the same storage format.

When INGRES operates on an integer and a floating point number, the integer is converted to a floating point number before the operation. When INGRES operates on two integers of different sizes, the smaller is converted to the size of the larger. When operating on two floating point numbers of different sizes, INGRES converts the larger to the size of the smaller number. (Note, however, that only one size floating point number, f4, is used in any of the 12 permanent ESM tables.) For example, in the (nonsensical) expression

(s.lat + s.region) / 15

the first operator (+) combines an f4 expression (*s.lat*) with an i1 expression (*s.region*). The result is f4. The second operator (/) combines the f4 expression with an i2 constant (15), resulting in an f4 expression.

Note that while *(s.lat + s.region) / 15* produces an f4 expression, *float8((s.lat + s.region) / 15)* produces an f8 expression. For information about explicit type conversion functions, refer to section B.3.4.5, below.

B.3.4.4 Default Character and Text Type Conversion

Whenever a string of type character is put into a column defined as type text, all the string's trailing blanks are removed. Conversely, whenever a string of type text is put into a column defined as type character, the string is padded with blanks to fill out the column's defined width, if necessary.

B.3.4.5 Explicit Type Conversion Functions

In addition to INGRES's default type conversions (see section B.3.4.3), many explicit type conversion functions are available. The following explicit type conversion functions can be used:

Name	Operand Type	Result	Description
ascii(expr)	any	c	Converts any value to character string.
date(expr)	c, text	date	Converts character or text string to internal date representation
dow(expr)	date	c	Converts absolute date into its day of week (e.g.,"Monday," "Tuesday")
float4(expr)	any except date	f4	Converts non-date expression to an f4 constant. Note that

			expr must be a number, although the number can be stored as any of the valid data types, including character and text, as in *float8("40")* in which "40" is a number stored as a text string.
float8(expr)	any except date	f8	Converts non-date expression to an f8 constant
int1(expr)	any except date	i1	Converts non-date expression to an i1 constant
int2(expr)	any except date	i2	Converts non-date expression to an i2 constant
int4(expr)	any except date	i4	Converts non-date expression to an i4 constant
money(expr)	any except date	money	Converts non-date expression to internal money representation.
text(expr)	any	text	Converts any value to a text string. This function removes trailing blanks, if any, from character string expressions.

The ascii function, *ascii(*n*)*, is especially useful. It converts a number n to its character representation. This is useful for comparing a numeric column with a character string. For example, the conditions

s.region = 4

and

ascii(s.region) ="4"

are identical. Note that the reverse operation, converting numbers stored as characters into numeric data types, is supported in the numeric type conversion functions, described in Section B.3.5.1, below.

B.3.5 Built-in Functions

A function is denoted by a function name, followed by a parenthesized operand (or list of operands). When expressions of a valid type are substituted for the operands, the result is again an expression. Functions can be nested to any level.

B.3.5.1 Numeric Functions

In addition to the type conversion functions (see section B.3.4.5 above), the following functions constitute type-to-type operators:

Name	Result Format	Description
abs(n)	i, f, or money	absolute value of n
atan(n)	f8	arctangent of n
cos(n)	f8	cosine of n
exp(n)	f8	exponential of n
log(n)	f8	natural logarithm of n
mod(n,b)	i	n, modulo b. n and b must be i1, i2 or i4
sin(n)	f8	sine of n
sqt(n)	f8	square root of n

For example, *exp(s.lat)* gives the exponential of *s.lat* as an f8 expression.

B.3.5.2 String Functions

The following functions operate on character or text data. The expressions "c1" and "c2" represent arguments for the various functions. They can represent character or text strings, except where noted. The expressions "len" and "nshift" represent integer arguments.

Name	Result Format	Description
concat(c1,c2)	c or text	Concatenates one string to another. The result size is the sum of the sizes of the two arguments.

		Blanks are trimmed from the first string if it is a character string, and the second string is added. If the result is a character string, it is padded to achieve the proper length. The result format is "text" if both parameters are text, and "c" otherwise.
left(c1,len)	c or text	Returns the leftmost "len" characters of "c1". If the result is a character string, it is the same length as "c1", padded with blanks. The result format is the same as "c1".
length(c1)	i2	If "c1" is a character string, returns the length of "c1" without trailing blanks. If "c1" is a text string, returns the number of characters actually in "c1".
locate(c1,c2)	i2	Returns the location of the first occurrence of "c2" within "c1", including trailing blanks from "c2." The location is in the range 1 to *size*(c1). If "c2" is not found, the function returns *size*(c1) + 1.
lowercase(c1)	c	Converts all upper case characters in "c1" to lower case.
pad(c1)	text	Returns "c1" with trailing blanks appended to "c1"; for instance, if "c1" is a text string that could hold fifty characters but only has two characters, then "pad(c1)" appends 48 trailing blanks to "c1" to form the result.
right(c1,len)	c or text	Returns the rightmost "len" characters of "c1". Trailing blanks are not removed first. If "c1" is a character string, the result is padded to the same length as "c1". If "c1" is a text string, no padding occurs. The result format is the same as "c1".
shift(c1,nshift)	c or text	Shifts the string "nshift" places to the right if "nshift" > 0 and to the left if "nshift" < 0. If "c1" is a character string, the result is

		padded with blanks to the length of "c1". If "c1" is a text string, no padding occurs. The result format is the same as "c1".
size(c1)	i2	Returns the declared size of "c1" without removal of trailing blanks.
squeeze(c1)	text	Compresses white space. White space is defined as any sequence of blanks, newlines (line feeds), carriage returns, horizontals tabs and form feeds (vertical tabs). Trims white space from the beginning and end of the string, and replaces all other white space with single blanks. This function is useful for comparisons. The value for "c1" must be a string of text data type (not character data type). The result is the same length as the argument.
trim(c1)	text	Returns "c1" without trailing blanks. The result has the same length as "c1". This function is useful for converting character strings to text strings.
uppercase(c1)	c	Converts all lower case characters in "c1" to upper case.

The character functions can be arbitrarily nested to achieve other string functions. For example,

left(right(x.name, size(x.name) - 1), 3)

returns the substring of *x.name* from character positions 2 through 4. In fact, this function is contained in the INGRES Terminal Monitor macro *substring*, and can be used as follows:

substring(c1,pos1,pos2)	c	Returns the characters in "c1" from character positions "pos1" to "pos2." The result is blank filled and is the length of "c1".

Note that the *substring* function is only available in the INGRES Terminal Monitor and cannot be invoked from an EQUEL program or the INGRES-Report-Writer. (EQUEL programs and the INGRES-Report-Writer are beyond the scope of this report. Refer to the INGRES documentation for more information.)

You can also nest character functions within themselves.

For example,

concat(concat(x.lastname, ","), x.firstname)

concatenates *x.lastname* with a comma and then concatenates *x.firstname* with the first concatenation result. Note, however, that the same result can be achieved with the "+" operator:

x.lastname + "," + x.firstname

B.3.5.3 Date Functions

INGRES provides three functions that manipulate INGRES dates (like the *edate* column in the ESM *events* table). These functions take two arguments, date and unit. The date expression must be an absolute date and not a date interval. The unit expression is a quoted string that represents the part of the date to use in the function's calculation. Legal values for unit are:

second	*seconds sec secs*
minute	*minutes min mins*
hour	*hours hr hrs*
day	*days*
week	*weeks wk wks*
month	*months mo mos*
quarter	*quarters qtr qtrs*
year	*years yr yrs*

Name	Result Format	Description
date_trunc(unit,date)	date	Returns a date value that represents the input date truncated to the level of granularity expressed in the unit.
date_part(unit,date)	i4	Returns an integer representing one component of the input date. The unit parameter represents the desired component.
interval(unit,date)	f8	Converts a date interval into a floating-point constant in the units specified in the unit parameter.

By using the *date_trunc* function you can group all the dates within the same month or year, and so forth. For example:

x =date_trunc("month", date("23-oct-1985 12:33"))

returns "1-oct-1985" as its value and

```
x =date_trunc("month", date("23-oct-1985"))
```

returns "1-jan-1985" as its value. All truncation takes place in terms of calendar years and quarters ("1-jan", "1-apr", 1-jun" and "1-oct"). If you need to truncate in terms of a fiscal year, simply offset the calendar date by the number of months between the beginning of your fiscal year and the beginning of the next calendar year ("6 mos" for a fiscal year beginning September 1):

```
x =date_trunc("year", q.mydate + "6 mos") - "6 mos"
```

Monday constitutes the starting day for weeks. Note the beginning of a week for an early January date may fall into the previous year.

The *date_part* function is useful in aggregations (see section B.3.7) and in assuring correct ordering in complex date manipulation. For example,

```
x = date_part ("month", date("23-oct-1985"))
```

returns a value of 10 and

```
x = date_part ("day", date("23-oct-1985"))
```

returns a value of 23. Months are ordered with January set to month 1. Hours are set to a 24-hour clock. Quarters are numbered 1 through 4. Weeks return a number representing the number of the week since the beginning of the year in which the input date falls. Week 1 begins on the first Monday of the year. Dates before the first Monday of the year are considered to be in week 0.

Here is an example of the *interval* function:

```
retrieve into mytable (time = date("1 day 3 hrs"))
retrieve (z = interval ("hrs", mytable.time))
retrieve (z = interval ("days", mytable.time))
```

The second *retrieve* returns a value for *z* of 27.0 (hours) and the third *retrieve* returns 1.125 (days). Note: the *interval* function assumes that there are 30.44 days per month and 365.25 days per year when using the *mos*, *qtrs*, and *yrs* specifications.

B.3.5.4 System Functions

In addition to numeric, ascii and string functions, INGRES also supports symbolic functions that return system-related information. These functions include the following:

Name	Result Format	Description
_bintim(0)	i4	Returns the current time and date in an internal format, represented as the number of seconds since January 1, 1970
_time(arg)	c5	Converts an internal time, arg, as generated by *_bintim(0)*, into a character format for the time as hh:mm
_date(arg)	c9	Converts an internal time, as generated by *_bintim(0)*, into a character format for the date as dd-mmm-yy

For example, the following retrieval,

retrieve (time = _time(_bintim(0)), date = _date(_bintim(0)))

results in the following output:

time	*date*
13:06	*5-mar-1982*

B.3.5.5 Symbolic Constants

QUEL includes several additional constants to assist in performance measurements. These constants represent the total resources consumed by the INGRES process from the time INGRES was initiated to the completion of the last QUEL statement, and thus the constants provide cumulative, not incremental, data. QUEL statements can invoke these constants exactly as other constants.

Name	Result Format	Description
Performance constants:		
_cpu_ms	i4	cpu time, in milliseconds
_et_sec	i4	elapsed time, in seconds
_dio_cnt	i4	direct I/O requests
_bio_cnt	i4	buffered I/O requests

_pfault_cnt	i4	page faults
_ws_page	i4	shows current physical memory, in pages
_phys_page	i4	shows current virtual memory of a process in bytes

Other constants:

_version	c6	INGRES version number (e.g., "1.3/01")
dba	c6	INGRES user code of the database owner (from USERS.TXT file)
usercode	c6	INGRES user code of the user currently running INGRES
username	c12	INGRES user name of the user currently running INGRES

To use these constants, simply specify them in the target-list of a *retrieve* statement. Becasue they are constants, you must specify a label in order to use them. For example, the query

retrieve (cptime = _cpu_ms, version = _version)

results in the following display:

cptime	*version*
810	*1.3/01*

B.3.6 Qualification

The term "qualification" refers to a condition to be tested on each row of a table. Each of the following can function as a qualification when preceded by the key word *where*:

```
expr  comparison-operator  expr
not  qual
qual  or  qual
qual  and  qual
(qual)
```

Some examples illustrate qualification:

```
where e.lat <= 50.0
where not (e.lat <= 50.0)
where (e.lat <= 50) and (   r.epi_d <= 20.0
                         or r.rtype = "RFT-350"
                        )
where r.epi_d > avg(r.epi_d  by  r.own_agency)
```

B.3.6.1 Clauses and Comparison Operators

A clause is a qualification or a qualification component that has the value "true" or "false" and is expressed in the form:

```
expression  comparison-operator  expression
```

A comparison-operator is a binary operator that takes two expressions as operands. The expressions must both be numeric, character, text, money or date types, or one text string compared with one character string. The following comparison-operators are recognized in QUEL:

=	equal to
!=	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

Note that the comparison operator "not equal to" may also be indicated by <> or =.

All comparison-operators are of equal precedence. When

comparisons are made between character strings, all blanks are ignored. When comparisons are made between text strings, however, all blanks are significant.

A clause may be enclosed in parentheses without affecting its interpretation, as in the following examples:

```
(s.lat < 50.0)
(s.str_code != " ")
(s.stn_number > 100 * s.region)
```

B.3.6.2 Logical Operators

The following boolean logical operators are recognized in QUEL:

not	logical not;	negation
and	logical and;	conjunction
or	logical or;	disjunction

These operators take truth functions, that is, functions (like clauses) that return "true" or "false", as operands and return truth values once again.

Not has the highest precedence of the three operators; *and* and *or* have equal precedence. Parentheses may be used for arbitrary grouping. Logical operators group from left to right.

B.3.6.3 Partial Match Specification

QUEL supports special characters for use with the comparison-operators (in particular, the equals operator, =) to indicate partial matches of character string data. These characters allow the following partial match specifications:

*	matches any string of zero or more characters
?	matches any single character
[..]	matches any of the characters in the brackets

QUEL allows any of these special characters singly or in combination to specify partial match criteria:

x.name = ""*	matches any value in *x.name*.
x.name = "E"*	matches any value in *x.name* that begins with "E"
*x.name = "*ein"*	matches any value ending with "ein"
x.name = "[aeiou]*"*	matches any value with at least one vowel
x.name = "Br???"	matches any five-character value

	beginning with "Br"
x.name = "[A-J]"*	matches any value beginning with A, B, C, ..., J.
x.name = "[N-Z]???"	matches any four-character value beginning with N, O, P, ..., Z.

Note that blanks must not be used in bracketed expressions such as *"[A-J]*"* or *"[N-Z]???"*.

The special meaning of these characters can be disabled in a clause by preceding them with a blackslash character (\). Thus, * refers to the asterisk character. However, in an assignment (as opposed to a clause), the special characters do not perform a partial match specification, as in the following:

*newname = "**accountant**"*

Because the fragment above assigns a value "**accountant**" to the column *newname*, the asterisks need no escape treatment with the backslash. However, to retrieve the value so assigned requires the following syntax:

q.newname = "**accountant**"

B.3.7 Aggregation

The term "aggregation" refers to an operation on a collection of constants of a single data type. QUEL supports the following aggregation-operators:

Name	Result Format	Description
count	i4	Count of occurrences
countu	i4	Count of unique occurrences
sum	i4, f8, money	Summation
sumu	i4, f8, money	Summation of unique values
avg	f8, money	Average (sum/count)
avgu	f8, money	Unique average, = (sumu/countu)
max	c,i,f, date, text	Maximum value
min	c,i,f, date, text	Minimum value
any	i2	Value is 1 if any rows satisfy the expressed qualification; otherwise the value is 0.

B.3.7.1 Aggregate Expressions

An aggregate expression has the following syntax:

aggreggate-operator (expr [*where* qual])

where aggregate-operator denotes one of the operators shown above, expr an expression, and qual a qualification. An aggregate is a constant obtained by (1) evaluating the expression once for each assignment of its range variables that satisfy the qualification, and (2) applying the aggregate-operator to the collection of values so derived. For example, suppose that a table named "job" contains the following rows:

jid	*jtitle*	*salary*
1001	*salesman*	*4231.000*
1023	*accountant*	*1834.500*
2021	*clerk*	*1212.500*

Now, if *range of j is job*, then *sum(j.salary)* evaluates to 7278.000, while *sum(j.salary where j.salary > 1500)* evaluates to 6065.500.

In an aggregate the range variables enclosed within the expression are purely local to the aggregate and are not linked to the same range variables that may occur elsewhere in the QUEL statement.

B.3.7.2 Aggregate Functions

The syntax of an aggregate function is as follows:

aggregate-operator (expr *by* columnname {,columnname} [*where* qual])

The component "*by* columnname {,columnname} " is known as the "by list". For example,

avg(i.in_date by i.owner)

is an aggregate function.

The value of an aggregate function depends on the value(s) assigned to the columns in the *by* list. For each assignment of values to the columns in its *by* list, an aggregate function evaluates to the following aggregate:

aggregate-operator (expr [*where* qual] *and* columnname = value)

For example, consider the aggregate function

min (i.in_date by i.owner where i.rtype = "RFT")*

For the condition *i.owner = "ACOE"*, the function evaluates to the following aggregate:

min (i.in_date where i.owner = "ACOE" and i.rtype = "RFT")*

Unlike a simple aggregate, an aggregate function is not purely local. The range variables appearing in the *by* list are global variables. For example, the aggregate function

min (i.in_date by i.owner where i.rtype = "RFT")*

evaluates to a value for each row, *i*, in the *instruments* table that depends on the value in the *owner* column in that row. If the value in the *owner* column for a particular row, say i_1, is "CDMG", then the aggregate function evaluates to the constant produced by

min (i.in_date where i.owner = "CDMG" and i.rtype = "RFT")*

for the row i_1.

B.3.7.3 Multiple Variables

Several variables can appear within a single aggregation operator, such as

avg(j.salary by p.dept where p.job =j.jid)

In such cases the aggregation is interpreted as being performed on the "cartesian product" of the tables referenced by the variables. The cartesian product of two or more tables is the single table formed by taking all combinations of rows of the constituent tables. For example, suppose that "personnel" and "job" are two tables given by the following:

personnel *(range of p is personnel)*:

eno	dept	job
e1	101	1
e2	101	2
e3	102	1

job table *(range of j is job)*:

jid	salary
1	2000.000
2	3000.000

Then, the cartesian product is given by the following table:

eno	dept	job	jid	salary
e1	101	1	1	2000.000
e2	101	2	1	2000.000
e3	102	1	1	2000.000
e1	101	1	2	3000.000
e2	101	2	2	3000.000
e3	102	1	2	3000.000

The aggregate function

avg(j.salary by p.dept where p.job =j.jid)

evaluates to a unique value for each distinct *dept* value. For *dept =101*, its value is given by

avg(j.salary where p.dept =101 and p.job =j.jid)

which is the average of the *salary* column of the following rows:

eno	dept	job	jid	salary
e1	101	1	1	2000.000
e2	101	2	2	3000.000

Hence *avg(j.salary by p.dept where p.job =j.jid)* has a value of 2500 for *dept =101* and a value of 2000 for *dept =102*.

B.3.7.4 Nesting

An aggregate can be viewed as a constant (because it produces a single value), and an aggregate function as a column (because it produces a set of values of like type). As such, aggregation produces new expressions. The cycles

aggregation -> expression -> aggregation

and

aggregation -> expression -> qualification -> aggregation

are both permitted, and nesting to any level is allowed. For example, the following are all legal in QUEL:

```
avg(p.age by p.dept where p.age>avg(p.age))
avg(j.salary by p.dept where p.job=j.jid and
   p.age>avg(p.age))
avg((p.age-avg(p.age))**2)
```

B.3.7.5 The "*by* List"

As mentioned in section B.3.7.2, an aggregate function generates result values for each unique value in the *by* list. The set of values in the *by* list is computed independently of the *where* clause in the aggregate function. Under this procedure, which is the default, the aggregate processor includes each independently computed value in the *by* list within the result set, even if that *by* list value is not present in any row that meets the qualification in the aggregate function's *where* clause. In the final result of an aggregate function, the *by* list values for which no rows qualify are returned with an aggregate value of 0 or blank.

INGRES enables you to override this default inclusion of 0-valued aggregates. You may instead exclude these aggregate values for which no qualifying rows exist by means of the *set aggregate noproject* statement described in section 2.23 of the INGRES reference manual. Excluding these 0-valued aggregates by using these alternate semantics can result in faster processing

of aggregate functions. This is particularly true for aggregate functions that contain *where* clauses.

APPENDIX C:

Terminal Monitor Commands[1]

The terminal monitor is an interface between the QUEL user and INGRES. The terminal monitor assists the user in entering and running QUEL commands from a terminal, and it allows the user to write, review, and edit QUEL commands prior to executing them.

The terminal monitor maintains a workspace that can be thought of as a blackboard. You enter QUEL commands into the workspace, examine them and, if necessary, change them. When you think the command is correct, you can tell the terminal monitor to pass your workspace to INGRES for execution. The terminal monitor then receives the results of your command(s) from INGRES and displays them on your terminal.

Whenever the terminal monitor is waiting for the user to type something, it prints an asterisk at the beginning of the current line. It also displays messages to inform you of its status. When it displays

```
go
*
```

or

```
continue
*
```

the terminal monitor is ready to accept commands. When the terminal monitor prints "*continue*," the workspace contains something, and when it prints "*go*," the workspace is empty. When it displays

```
Executing ...
```

the terminal monitor is telling you that INGRES is executing your command(s).

The terminal monitor supports a variety of commands that facilitate your session with INGRES, but there are eight basic commands with which you should be familiar. These eight

---

[1]Most of the information in this appendix has been copied or paraphrased from Chapter 5 of the "INGRES Self-Instruction Guide".

commands appear in Chapter 8, Table 8-2 which is duplicated in Table C-1 below:

command	meaning
\r	*reset*/erase the workspace
\p	*print* the workspace
\e	invoke *editor* on the workspace
\g	execute query (*go*)
\a	*append* to the workspace
\q	*quit*/exit INGRES
\i filename	read/*include* the named file
script filename	log query and response to named file

Table C-1
Terminal Monitor Commands

Each of the one-letter commands shown in table C-1 can be entered in its one-letter form or with its full name. All terminal monitor commands require the backslash character.

The \g or \go Command

This most basic terminal monitor command passes the QUEL commands in the workspace to INGRES for execution. For instance, if you want to ask the names of all the tables in a database, you execute the *help* command as follows:

```
* help \g
```

or

```
* help
* \g
```

The QUEL command *help* lists information about the database. The terminal monitor command \g can appear anywhere after the QUEL command, *help*, either on the same line or on a different line. The \g command passes all of the commands in the workspace to INGRES, where they are executed in the order specified in the workspace. Thus you can execute any number of QUEL statements with one terminal monitor command.

When you type a QUEL command into the workspace, it stays there, even after the command is executed, until you enter new commands. This feature allows you to access those existing commands, and the terminal monitor offers a variety of functions that affect the existing workspace contents.

After a *\go* command, the query buffer is automatically cleared if another query is typed in -- unless a command that affects the query buffer is typed first. Commands that retain the query buffer contents are *\a*, *\e*, *\p*, *\v*, *\bell*, and *\nobell*. (*\v*, *\bell* and *\nobell* are not discussed elsewhere in this manual.)

For example, typing

```
help doc
\g
print doc
```

results in the query buffer containing

```
print doc
```

whereas,

```
help doc
\go
\p
print doc
```

results in the query buffer containing

```
help doc
print doc
```

Note that the *doc* table, like all the tables in ESM, is quite long. A user would probably get quite tired of seeing the entire table flash by on the screen if the *print doc* command shown above were actually executed. To stop the display of a print you're not really interested in watching to completion, type control-c (See Chapter 6, section 6.3).

The *\p* or *\print* Command

The *\p* command simply prints the contents of the workspace so you can see what is there. For instance, if you enter and execute the help command above, you can then review the command just executed by typing "*\p*":

```
* \p
help doc
print doc
continue
*
```

When you enter the *\p* command, the terminal monitor writes "*help doc*" then "*print doc*" (without the prompt character) then asks you to continue. If you enter a complex query or several queries, or if a query fails to execute properly, you can use the *\p* command to look at the workspace contents without losing them.

Note the difference between the QUEL *print* command and the terminal monitor *\p* command: the first prints one of the database tables, the second prints the contents of the terminal monitor's workspace.

The \a or *append* Command

The \a command allows you to append new QUEL statements to existing workspace contents. For instance, if you want to add a request for help about a specific table, such as the *events* table, you can do so with the following command:

```
* \a
  help events
```

When you execute these commands (with \g), the terminal monitor sends any previously existing commands as well as the new help command to INGRES. To verify that the new command is appended, use the \p command:

```
* \p
help doc
print doc
help events
continue
*
```

The \r or *reset* Command

The \r command enables you to reset, or clear, the contents of the workspace. Simply enter

```
* \r
```

and the terminal monitor responds with

```
go
*
```

Remember that the "*go*" response indicates that the workspace is empty. It can be useful to empty the workspace if you feel hopelessly lost with a current query.

The \i or *include* or *read* Command

The \i command allows you to include the contents of a named file into the workspace. This is useful for executing "canned" queries and scripts, thus saving you a lot of typing. The \i command requires that you name a file directly after "\i" on the command line, as in

```
* \i pub2:[esm.examples]sample1.iql
```

This command reads the contents of the file named *sample1.iql* from the directory at *PUB2:[ESM.EXAMPLES]* into the workspace. Obviously, the file must already exist. If the file does not

exist, the terminal monitor issues an appropriate error message. If no file is named with the *\i* command, the command has no effect.

The *\e* or *\edit* Command

The *\e* command invokes a text editor for use in editing your workspace contents. For instance, if you misspelled something you entered into the workspace, you could edit the workspace text rather than clearing the workspace and retyping it all again. This is particularly useful if a single error appears in a complex query or a collection of many queries in the workspace.

When you execute the *\e* command, the terminal monitor relinquishes its control of the communication with the user and passes that control to the EDT text editor. EDT is described in the VAX manual titled "VAX/VMS Volume 3A; Text Processing; EDT Reference". After typing *\e*, use EDT commands to change the workspace text. Then, when you are satisfied that the new version is correct, exit from EDT. INGRES will automatically return you to your current terminal monitor session with the workspace containing the edited text.

The *\script* Command

The *\script* command can be used to save your QUEL commands and their results to a disk file. The information displayed at the terminal is not affected: it will look the same regardless of whether scripting is on or off. The command acts as an on/off toggle to request whether or not a copy of what is appearing at the terminal shall be written to a disk file. When turning on the scripting function, a file name should be typed following the word *script*, as in:

```
\script tmp.tmp
```

If no file name is supplied, the script information is written to a file named "*script.ing*" in the user's current working directory.

The *\q* or *\quit* Command

The *\q* command enables you to quit INGRES and return to the VAX operating system level. Simply enter

```
* \q
```

and the INGRES log-out message will appear.

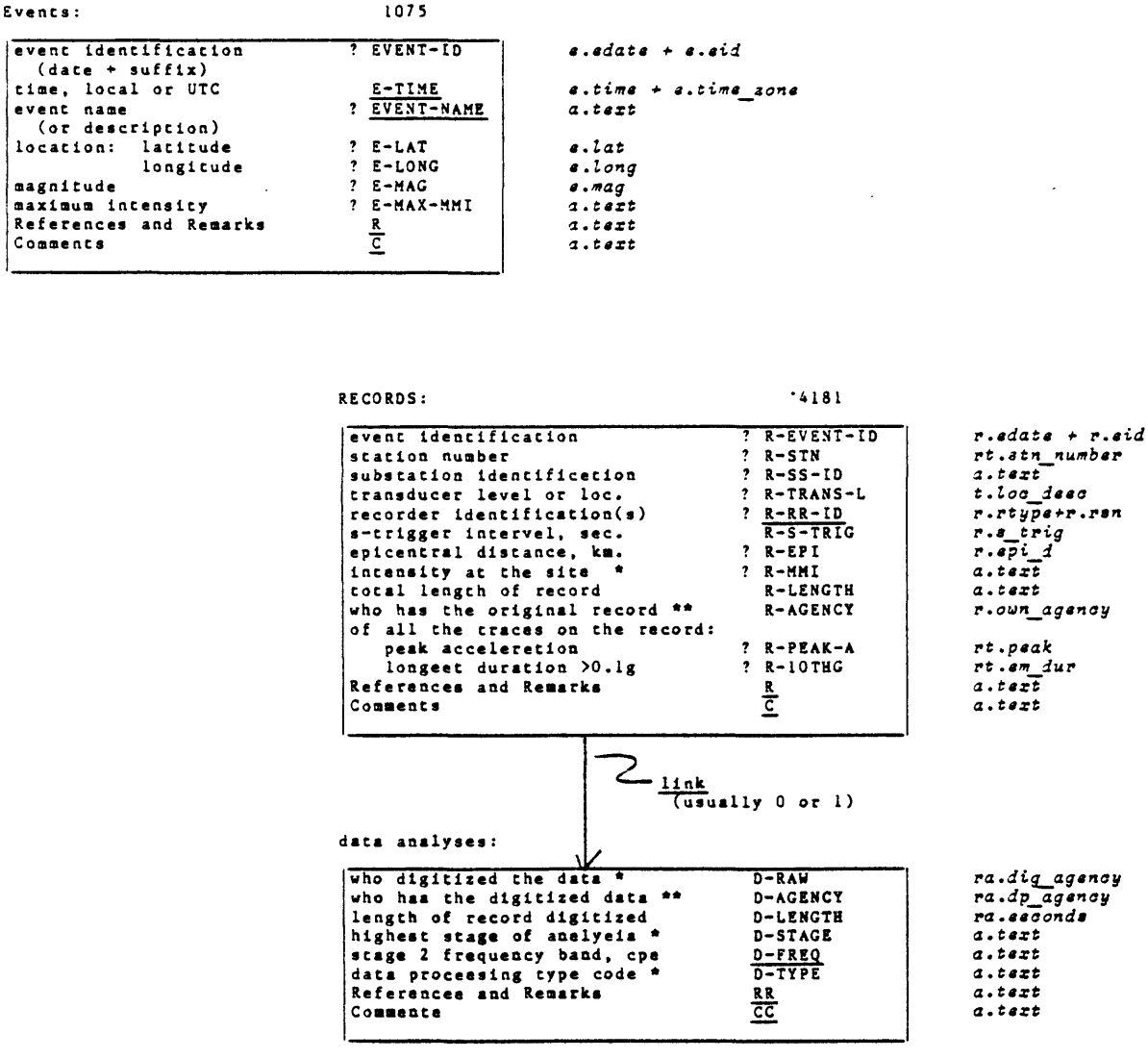
The terminal monitor also supports other commands which are

listed in appendix A and described in detail in Chapter 3 of the "INGRES Reference Manual". The eight commands presented here, however, represent the most commonly used commands in a typical INGRES/QUEL retrieval session.

APPENDIX D:

Old-SMIRS Summary Diagram Showing ESM Counterparts

Here follows a copy of the summary diagram from the old-SMIRS manual. The names of the ESM counterparts to each old-SMIRS data element are noted in italics along the right-hand margins of the diagram.



STATIONS: 1696

station number	?	ST-NUMBER	*s.stn_number*
station name (or street address)	?	ST-NAME	*a.text*
station location (or city)		ST-LOC	
state (or country) *	?	ST-STATE	*s.state*
nearby stations flag		ST-NEARBY	
References and Remarks		R	*a.text*
Comments		C	*a.text*

link
(often just one)

structures:

substation identification		SS-ID	*s.sub_stn*
late of installation	?	SS-IN-DATE	*s.in_date*
" removal	?	SS-RM-DATE	*s.rm_date*
location: latitude	?	SS-LAT	*s.lat*
longitude	?	SS-LONG	*s.long*
geology class code *	?	SS-GEOL-C	*a.text*
near surface shear wave vel.		SS-SHEAR-V	*a.text*
structure: class code *	?	SS-STR-C	*s.str_code*
size		SS-STR-S	*a.text*
short description		SS-STR-D	*a.text*
Remmarks		RR	*a.text*
Comments		CC	*a.text*

RECORDERS: 1594

recorder identification (**)	?	RR-ID	*i.rtype + i.rsn*
owner agency	?	RR-OWNER	*i.owner*
date of installation	?	RR-IN-DATE	*i.in_date*
" removal	?	RR-RM-DATE	*i.rm_date*
Remarks		R	*a.text*
Comments		C	*a.text*

link
(usually just one)

substations services:

station number	?	SSS-STN	*t.stn_number*
substation identifier		SSS-ID	*t.sub_stn*
Remarks		RR	*a.text*
Comments		CC	*a.text*

link2
(often 3, 5, 9, or 12)

transducers:

transducer serial number		T-SN		*c.tsn*
date of installation		T-IN-DATE	#	*i.in_date*
" removal		T-RM-DATE	#	*i.rm_date*
location: code *?	?	T-LOC-C	#	*t.loc_code*
short description		T-LOC-D	#	*t.loc_desc*
direction of acceleration for positive trace		T-ACCEL		*t.direction*
orientation		T-ORIENT		*(none)*
relative location on the physical record		T-TRACE		*t.channel*
Remarks		RRR		*a.text*
Comments		CCC		*a.text*

link3
(any number)

transducer calibrations:

calibration date	C-DATE	*c.cdate*
sensitivity, cm./g.	C-SENS	*c.sens*
period, sec.	C-PERIOD	*c.period*
damping, fraction of crit.	C-DAMPING	*c.damping*
Remarks	RRRR	*a.text*
Comments	CCCC	*a.text*

All the old-SMIRS information shown on this page is stored in the *doc*umentation table in ESM.

B. Dictionary data sets

B. 1 Recorder TYPES: 25

recorder type code (**)	?	REC-CODE
Remarks describing the recorder and its transducers.		R
Comments		C

B. 2 AGENCIES
(recorder owners and sources of information):

agency code (**)	?	AGENCY
Remarks about the agency, its full name, address and names of people or sub-organizations who can provide information.		R
Comments		C

B. 3 ARRAYS:

array name (**)	?	ARRAY
list of: stations and their substations in the array.	?	A-STN A-SUB-STN
Remarks		R
Comments		C

B. 4 groups of NEARBY stations:

a list of stations that are nearby one another.	?	NEARBY
Remarks		R
Comments		C

C. General information data sets

C. 1 Remarks about this data base and other sources of strong-motion information:

name of one of the categories of general information (e.g., "news", "commands"" "data processing"...)	?	INFO
Remarks		R

C. 2 Information about the other data sets:

data set name	DATASET
date to which the data in the named data set is current.	CURRENT
Remarks about the data set	R