# INGRES REFERENCE MANUAL — VERSION 6.1

by

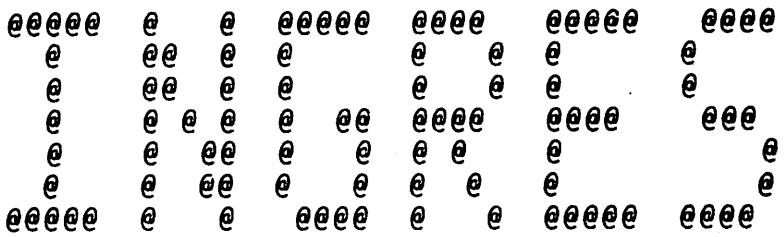
K. Youssefi, N. Whyte, M. Ubell, D. Ries,
P. Hawthorn, B. Epstein, R. Berman and E. Allman

Memorandum No. ERL-M579

20 December 1977 (revised)

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720



VERSION 6.1

R E F E R E N C E  M A N U A L

12/20/77

by

Karel Youssefi
Nick Whyte
Mike Ubell
Dan Ries
Paula Hawthorn
Bob Epstein
Rick Berman
Eric Allman

This manual is a reference manual for the INGRES data base system. It documents the use of INGRES in a very terse manner. For learning how to use INGRES refer to the document called "A Tutorial on INGRES".

This manual is subdivided into four parts:

Quel: describes the commands and features which are used inside of INGRES.

Unix: describes the INGRES programs which are executable as unix commands.

Files: describes some of the important files used by INGRES.

Error: lists all the user generatable error messages along with some elaboration as to what they mean or what we think they mean.

Each entry in this manual has one or more of the following sections:

NAME section

This section repeats the name of the entry and gives an indication of its purpose.

SYNOPSIS section

This section indicates the form of the command (statement). The conventions which are used are as follows:

Upper case names are used to indicate reserved words. When entering actual INGRES commands, these words need not be typed in upper case.

Lower case words indicate generic types of information which must be supplied by the user. In the DESCRIPTION section the legal values for these names are described.

Square brakets ( [] ) indicate that the enclosed item is optional.

Braces ( {} ) indicate an optional item which may be repeated.

DESCRIPTION section

This section gives a detailed description of the entry with references to the reserved words and generic names used in the SYNOPSIS section.

EXAMPLE section

This section gives one or more examples of the use of the entry. Most of these examples are based on the two relations:

emp(name,sal,mgr,bdate)
and
newemp(name,sal,age)

SEE ALSO section
This section gives the names of entries in the manual which are closely related to the current entry.

DIAGNOSTICS section
This section describes error messages and warning diagnostics specific to the particular command which may not be completely transparent to the user.

BUGS section
This section indicates known bugs or deficiencies in the command.

To start using INGRES you must be entered as an INGRES user. The INGRES administrator does this by entering you into the "users" file (see users(files)). See the section on ingres(unix) to start using INGRES. The sections on quel(quel) and monitor(quel) will direct you in how to use INGRES.

ACKNOWLEDGEMENTS

We would like to acknowledge the people who have worked on INGRES in the past.

William Zook, Peter Rubinstein, Peter Kreps, Gerald Held, James Ford.

NAME

append - append tuples to a relation

SYNOPSIS

APPEND [TO] relname (target_list) [WHERE qual]

DESCRIPTION

Append adds tuples which satisfy the qualification to relname. Relname must be the name of an existing relation. The target_list specifies the values of the attributes to be appended to relname. The domains may be listed in any order. Attributes of the result relation which do not appear in the target_list as result_attnames (either explicitly or by default) are assigned default values of 0 for numeric attributes or blank for character attributes.

Values or expressions of any numeric type may be used to set the value of a numeric type domain, with conversion to the result domain type taking place. Numeric values cannot be directly assigned to character domains. Conversion from numeric to character can be done using the "ascii" operator (see quel). Character values cannot be directly assigned to numeric domains. Use the "int1", "int2", etc. functions to convert character values to numeric (see quel(quel)).

The keyword "ALL" can be used when it is desired to append all domains of a relation.

EXAMPLE

```
/* Make new employee Jones work for Smith */
   range of n is newemp
   append to emp(n.name,n.sal,mgr="Smith",bdate=1975-n.age)
                where n.name = "Jones"
/* Append the newemp1 relation to newemp */
   range of n1 is newemp1
   append to newemp(n1.all)
```

SEE ALSO

quel(quel), retrieve(quel), copy(quel)

DIAGNOSTICS

Use of a numeric type expression to set a character type domain or vice versa will produce diagnostics.

BUGS

Duplicate tuples appended to a relation stored as a "paged-heap" (unkeyed, unstructured) are not removed.

NAME

copy - copy data into/from a relation from/into a UNIX file.

SYNOPSIS

COPY relname ({domname = format {, domname = format }} )
direction "filename"

DESCRIPTION

Copy moves data between INGRES relations and standard UNIX files. Relname is the name of an existing relation. In general domname identifies a domain in relname. Format indicates what format the UNIX file should have for the corresponding domain. Direction is either "into" or "from". Filename is the full UNIX pathname of the file.

On a copy "from" a file to a relation, the relation cannot have a secondary index, it must be owned by you, and it must be updatable (not a secondary index or system relation).

The formats allowed by copy are:

i1,i2,i4 - The data is stored as an integer of length 1, 2, or 4 bytes in the UNIX file.

f4,f8 - The data is stored as a floating point number (either single or double precision) in the UNIX file.

c1,c2,...,c255 - The data is stored as a fixed length string of characters.

c0 - Variable length character string.

d0,d1,...,d255 - Dummy domain.

Corresponding domains in the relation and the UNIX file do not have to be the same type or length. Copy will convert as necessary. When converting anything except character to character, copy checks for overflow. When converting from character to character, copy will blank pad or truncate on the right as necessary.

The domains should be ordered according to the way they should appear in the UNIX file. Domains are matched according to name, thus the order of the domains in the relation and in the UNIX file does not have to be the same.

Copy also provides for variable length strings and dummy domains. The action taken depends on whether it is a copy "into" or a copy "from". Delimitors for variable length strings and for dummy domains can be selected from the list of:

nl - new line character
tab - tab character
sp - space
nul or null - null character
comma - comma
colon - colon
dash - dash
lparen - left parenthesis
rparen - right parenthesis
x - any single character 'x'

The special meaning of any delimitor can be turned off by preceeding the delimitor with a "\".

When the direction is FROM, copy appends data into the relation FROM the UNIX file. Domains in the INGRES relation which are not assigned values from the UNIX file are assigned the default value of zero for numeric domains, and blank for character domains. When copying in this direction the following special meanings apply:

c0delim - The data in the Unix file is a variable length character string terminated by the delimitor "delim". If "delim" is missing then the first comma, tab, or newline encountered will terminate the string. The delimitor is not copied.

For example:
pnum=c0 string ending in comma, tab, or nl.
pnum=c0nl - string ending in nl.
pnum=c0sp - string ending in space.
pnum=c0Z - string ending in the character 'Z'.

A delimitor can be escaped by preceeding it with a "\". For example, using name = c0, the string "Blow\, Joe," will be accepted into the domain as "Blow, Joe".

d0delim - The data in the Unix file is a variable length character string delimited by "delim". The string is read and discarded. The delimitor rules are identical for "c0" and "d0". The domain name is ignored.

d1,d2,...,d255 - The data in the Unix file is a fixed length character string. The string is read and discarded. The domain name is ignored.

When the direction is INTO, copy transfers data INTO the UNIX file from the relation. If the file already existed, it is truncated to zero length before copying begins. When copying in this direction, the following special meanings apply:

c0 - The domain value is converted to a fixed length character string and writted into the Unix file. For character domains, the length will be the same as the domain length. For numeric domains, the standard INGRES conversions will take place as specified by the "-i", "-f", and "-c" flags (see ingres(unix)).

c0delim - The domain will be converted according to the rules for c0 above. The one character delimitor will be inserted immediately after the domain.

d1,d2,...,d255 - The domain name is taken to be the name of the delimitor. It is written into the Unix file 1 times for d1, 2 time for d2, etc.

d0 - This format is ignored on a copy "into".

d0delim - The "delim" is written into the file. The domain name is ignored.

If no domains appear in the copy command (i.e. copy relname () into/from "filename") then COPY automatically does a "bulk" copy of all domains using the order and format of the domains in the relation. This is provided as a convenient shorthand notation for copying and restoring entire relations.

EXAMPLE

```
/* Copy data into the emp relation */
   copy emp(name=c10,sal=f4,bdate=i2,mgr=c10,xxx=d1)
                from "/mnt/me/myfile"

/* Copy employee names and their salaries into a file */
   copy emp(name=c0,comma=d1,sal=c0,nl=d1)
                into "/mnt/you/yourfile"
```

SEE ALSO

append(quel), create(quel), ingres(unix)

DIAGNOSTICS

There are many possible user errors. They attempt to be self-explanatory. In addition several "warnings" can appear after a copy is complete. On a copy "from" a warning is issued if a duplicate tuple is found, if a character domain contains non-printing (ie. control characters) characters, and if a domain read using the "c0" format had to be truncated.

BUGS

Copy stops operation at the first error.

When specifying "filename", the entire UNIX directory pathname must be provided, since INGRES operates out of a different directory than the user's working directory at the time INGRES is invoked.

NAME

create - create a new relation

SYNOPSIS

CREATE relname (domname = format {, domname = format } )

DESCRIPTION

Create will enter a new relation into the data base. The relation will be "owned" by the user and will be set to expire after seven days. The name of the relation is relname and the domains are named domname1, domname2, etc. The domains are created with the type specified by format. Formats are of the form Xn where X is a character (i, f, or c) and n is an integer (1 to 255). The type of the domain may be integer (i), floating (f), or character (c). and the length of the domain may be 1,2 or 4 for integers; 4 or 8 for floating; or 1 to 255 for characters.

The relation is created as a paged-heap with no data initially in it.

A relation can have no more than 49 domains. A relation cannot have the same name as a system relation.

EXAMPLE

```
/* Create relation emp with domains name, sal and bdate */
   create emp(name = c10, salary = f4, bdate = i2)
```

SEE ALSO

save(quel), destroy(quel), copy(quel), append(quel)

DIAGNOSTICS

If the user already owns a relation with the same name as relname, a diagnostic is issued and the create is aborted.

BUGS

NAME

delete - delete tuples from a relation

SYNOPSIS

DELETE tuple_variable [WHERE qual]

DESCRIPTION

Delete removes tuples which satisfy the qualification from the relation that they belong to. The tuple_variable must have been declared in a range statement to range over an existing relation. Note that delete does not have a target_list. Also note that the delete command requires a tuple variable from a range statement and not the actual relation name. If the qualification is not given, the effect is to delete all tuples in the relation. The result is a valid, but empty relation.

EXAMPLE

```
/* Remove all employees who make over $30,000 */
   range of e is emp
   delete e where e.sal > 30000
```

SEE ALSO

quel(quel), range(quel), destroy(quel)

DIAGNOSTICS

BUGS

NAME

destroy - destroy existing relation(s)

SYNOPSIS

DESTROY relname {, relname}

DESCRIPTION

Destroy removes the relations from the data base. Only the relation owner may destroy a relation. A relation may be emptied of tuples but not destroyed using the delete statement or the modify statement.

If the relation being destroyed has secondary indices on it, the secondary indices are also destroyed. Destruction of just a secondary indice does not affect the primary relation it indexes.

EXAMPLE

```
/* Destroy the emp relation */
   destroy emp
   destroy emp, parts
```

SEE ALSO

create(quel), delete(quel), index(quel), modify(quel)

DIAGNOSTICS

BUGS

NAME

help - get information about how to use INGRES

SYNOPSIS

HELP [relname]["section"] {,relname}{,"section"}

DESCRIPTION

HELP may be used to obtain information about any section of this manual, the content of the current data base, or a specific relation in the data base. The legal forms are as follow:

HELP "section" - Produces a copy of the specified section of the INGRES Reference Manual, and prints it on the standard output device.

HELP - Gives information about all relations that exist in the current database.

HELP relname - Gives information about the specified relations.

HELP "" - Gives the table of contents.

EXAMPLE

```
help
help "help"    /* prints this page of the manual */
help "quel"
help emp
help emp, parts, "help", supply
```

SEE ALSO

DIAGNOSTICS

Cannot find manual section " ... "
Relation " ... " not found

BUGS

Alphabetics appearing within the section name must be in lower-case to be recognized.

NAME

index - create a secondary index on an existing relation.

SYNOPSIS

INDEX ON relname IS indexname (domain1 { ,domain2 } )

DESCRIPTION

Index is used to create secondary indices on existing relations in order to make retrieval and update with secondary keys more efficient. The secondary key is constructed from relname domains 1, 2,...,6 in the order given. Only the owner of a relation is allowed to create secondary indices on that relation.

In order to maintain the integrity of the index, users will NOT be allowed to directly update secondary indices. However, whenever a primary relation is changed, its secondary indices will be automatically updated by the system. Secondary indices may be modified to further increase the access efficiency of the primary relation. When an index is first created, it is automatically modified to an ISAM storage structure on all its domains. If this structure is undesirable, the user may override the default ISAM structure by using the "-n" switch (see ingres), or by entering a modify command directly.

If a modify or destroy command is used on relname all secondary indices on relname are destroyed.

Secondary indices on other indices, or on system relations are forbidden.

EXAMPLE

```
/* Create a secondary index called "x" on relation "emp" */
   index on emp is x(mgr,sal)
```

SEE ALSO

copy(quel), destroy(quel), modify(quel)

BUGS

At most 6 domains may appear in the key.

The copy command cannot be used to copy into a relation which has secondary indices.

The default structure ISAM is a poor choice for an index unless the range of retrieval is small.

NAME

modify - convert the storage structure of a relation

SYNOPSIS

MODIFY relname TO storage-structure [ON key1 {, key2}]
[WHERE [FILLFACTOR = n] [,MINPAGES = n] [,MAXPAGES = n]]

DESCRIPTION

Relname is modified to the specified storage structure. Only the owner of a relation can modify that relation. This command is used to increase performance when using large or frequently referenced relations. The storage structures are specified as follows:

isam - indexed sequential storage structure
cisam - compressed isam
hash - random hah storage structure
chash - compressed hash
heap - unkeyed and unstructured
cheap - compressed heap
heapsort - heap with tuples sorted and duplicates removed
cheapsort - compressed heapsort
truncated - heap with all tuples deleted

The paper "Creating and Maintaining a Database in INGRES" (ERL Memo M77-71) discusses how to select storage structures based on how the relation is used.

The current compression algorithm only suppresses trailing blanks in character fields. A more effective compression scheme may be possible, but tradeoffs between that and a larger and slower compression algorithm are not clear.

If the ON phrase is omitted when modifying to isam, cisam, hash or chash, the relation will automatically be keyed on the first domain. When modifying to heap or cheap the ON phrase must be omitted. When modifying to heapsort or cheapsort the ON phrase is optional.

When a relation is being sorted (isam, cisam, heapsort and cheapsort), the primary sort keys will be those specified in the ON phrase (if any). The first key after the ON phrase will be the most significant sort key and each successive key specified will be the next most significant sort key. An domains not specified in the ON phrase will be used as least significant sort keys in domain number sequence.

FILLFACTOR specifies the percentage (n can vary from 1 to 100) of each primary data page that should be filled with tuples under ideal conditions. FILLFACTOR may be used with isam, cisam, hash and chash. Care should be taken when using large fillfactors since a non-uniform distribution of key values could cause overflow pages to be created, and

thus degrade access performance for the relation.

MINPAGES specifies the minimum number of primary pages a hash or chash relation must have. MAXPAGES specifies the maximum number of primary pages a hash or chash relation may have. MINPAGES and MAXPAGES must be at least one. If both MINPAGES and MAXPAGES are specified in a modify, MINPAGES cannot exceed MAXPAGES.

Default values for FILLFACTOR, MINPAGES and MAXPAGES are as follows:

| | FILLFACTOR | MINPAGES | MAXPAGES |
|---|---|---|---|
| hash | 50 | 10 | no limit |
| chash | 75 | 1 | no limit |
| isam | 80 | NA | NA |
| cisam | 100 | NA | NA |

EXAMPLES

```
/* modify the emp relation to an indexed
       sequential storage structure with
       "name" as the keyed domain */

modify emp to isam on name

/* if "name" is the first domain of the emp relation,
       the same result can be achieved by */

modify emp to isam

/* do the same modify but request a 60  occupancy
       on all primary pages */

modify emp to isam on name where fillfactor = 60

/* modify the supply relation to compressed hash
       storage structure with "num" and "quan"
       as keyed domains */

modify supply to chash on num, quan

/* now the same modify but also request 75  occupancy
       on all primary, a minimum of 7 primary pages
       pages and a maximum of 43 primary pages */

modify supply to chash on num, quan
       where fillfactor = 75, minpages = 7,
       maxpages = 43

/* again the same modify but only request a minimum
       of 16 primary pages */

modify supply to chash on num, quan
       where minpages = 16
```

```
    /* modify parts to a heap storage structure */

    modify parts to heap

    /* modify parts to a heap again, but have tuples
          sorted on "pnum" domain and have any duplicate
          tuples removed */

    modify parts to heapsort on pnum
```

SEE ALSO

sysmod(unix)

NAME

monitor - interactive terminal monitor

DESCRIPTION

The interactive terminal monitor is the primary front end to INGRES. It provides the ability to formulate a query and review it before issuing it to INGRES. If changes must be made, one of the UNIX text editors may be called to edit the query buffer.

Messages and Prompts. The terminal monitor gives a variety of messages to keep the user informed of the status of the monitor and the query buffer.

As the user logs in, a login message is printed. This typically tells the version number and the login time. It is followed by the dayfile, which gives information pertinant to users.

When INGRES is ready to accept input, the message "go" is printed. This means that the query buffer is empty. The message "continue" means that there is information in the query buffer. After a \go command the query buffer is automatically cleared if another query is typed in, unless a command which affects the query buffer is typed first. These commands are \append, \edit, \print, and \go. For example, typing

```
help parts
\go
print parts
```

results in the query buffer containing

```
print parts
```

whereas

```
help parts
\go
\print
print parts
```

results in the query buffer containing

```
help parts
print parts
```

An asterisk is printed at the beginning of each line when the monitor is waiting for the user to type input.

Commands. There are a number of commands which may be en-

tered by the user to affect the query buffer or the user's environment. They are all preceeded by a backslash ("\"), and all are executed immediately (rather than at execution time like queries).

Some commands may take a filename, which is defined as the first significant character after the end of the command until the end of the line. These commands may have no other commands on the line with them. Commands which do not take a filename may be stacked on the line; for example

```
\date\go\date
```

will give the time before and after execution of the current query buffer.

\r
\reset — Erase the entire query (reset the query buffer). The former contents of the buffer are irretrieveably lost.

\p
\print — Print the current query. The contents of the buffer are printed on the user's terminal.

\e
\ed
\edit
\editor — Enter the UNIX text editor (see ED in the UNIX Programmer's Manual); use the ED command 'w' followed by 'q' to return to the INGRES monitor. If a filename is given, the editor is called with that file instead of the query buffer. The monitor first tries to call the program /usr/bin/ingres_ed, and if that fails, calls /bin/ed. It is important that you do not use the "e" command inside the editor; if you do the (obscure) name of the query buffer will be forgotten.

\g
\go — Process the current query. The contents of the buffer are transmitted to the parser and run.

\a
\append — Append to the query buffer. Typing \a after completion of a query will override the auto-clear feature and guarantees that the query buffer will not be reset.

\time
\date — Print out the current time of day.

\s

\sh
\shell Escape to the UNIX shell. Typing a control-d will cause you to exit the shell and return to the INGRES monitor. If there is a filename specified, that filename is taken as a shell file which is run with the query buffer as the parameter "$1". If no filename is given, an interactive shell is forked. The monitor first tries to call /usr/bin/ingres_sh. If that fails, it calls /bin/sh.

\q
\quit Exit from INGRES.

\cd
\chdir Change the working directory of the monitor to the named directory.

\i
\include
\read Switch input to the named file. Backslash characters in the file will be processed as read.

\w
\write Write the contents of the query buffer to the named file.

\<any other character>
Ignore any possible special meaning of character following '\'. This allows the '\' to be input as a literal character. (See also quel(quel) - strings). It is important to note that backslash escapes are sometimes eaten up by the macro processor also; in general, send two backslashes if you want a backslash sent (even this is too simplistic [sigh] - try to avoid using backslashes at all).

Macros. Version 6.1 of INGRES introduces a powerful macro facility which is still experimental. When additional user tests are completed and the design of the macros frozen, they will be released.

It will be possible to define macros which will allow abbreviations for keywords and whole commands. For example, macros can be defined which will expand

```
r p parts
ret (p.all) w p.pnum = 5
```

to

```
range of p is parts
```

```
        retrieve (p.all) where p.pnum = 5
```

or expand

```
        get all from parts
```

to

```
        range of x is parts
        retrieve (x.all)
```

When macros are fully implemented, several commands to support them will appear. Also, the macros will superceed certain other features; notably, the ingres_ed and ingres_sh features will be replaced by a feature to allow selection of editor and shell on a per-user basis.

Initialization. At initialization (login) time a number of initializations take place. First, a macro called "{pathname}" is defined which expands to the pathname of the INGRES subtree (normally "/mnt/ingres"); it is used by system routines such as demodb. Second, the initialization file .../files/startup is read. This file is intended to define system-dependent parameters, such as the default editor and shell. Third, a user dependent initialization file, specified by a field in the users file, is read and executed. This might be used to define certain macros, execute common range statements, and soforth. Finally, control is turned over to the user's terminal.

An interrupt while executing either of the initialization files restarts execution of that step.

Flags. Certain flags may be included on the command line to INGRES which affect the operation of the terminal monitor. The -a flag disables the autoclear function. This means that the query buffer will never be automatically cleared; equivalently, it is as though a \append command were inserted after every \go. Note that this means that the user must explicitly clear the query buffer using \reset after every query. The -d flag turns off the printing of the dayfile. The -s flag turns off printing of all messages (except errors) from the monitor, including the login and logout messages, the dayfile, and prompts. It is used for executing "canned queries", that is, queries redirected from files.

SEE ALSO

ingres(unix)
quel(quel)

DIAGNOSTICS

| | |
|---|---|
| go | You may begin a fresh query. |
| continue | The previous query is finished and you are back in the monitor. |
| Executing . . . | The query is being processed by INGRES. |
| >>ed | You have entered the UNIX text editor. |
| >>sh | You have escaped to the UNIX shell. |
| Funny character nnn | converted to blank<br>INGRES maps non-printing ASCII characters into blanks; this message indicates that one such conversion has just been made. |

BUGS

NAME

print - print relation(s)

SYNOPSIS

PRINT relname {, relname}

DESCRIPTION

Print displays each relation specified in entirety on the terminal (standard output). The formats for various types of domains can be defined by the use of switches when INGRES is invoked. Domain names are truncated to fit into the specified width. See ingres for details.

EXAMPLE

```
/* Print the emp relation */
   print emp
   print emp, parts
```

SEE ALSO

ingres(unix), retrieve(quel), printr(unix)

DIAGNOSTICS

BUGS

Print does not handle long lines of output correctly - no wrap around.

Print should have more formating features to make printouts more readable.

Print should have an option to print on the line printer.

NAME

quel - QUEry Language for INGRES

DESCRIPTION

The following is a description of the general syntax of QUEL. Individual QUEL statements and commands are treated separately in the document; this section describes the syntactic classes from which the constituent parts of QUEL statements are drawn.

1. Comments

A comment is an arbitrary sequence of characters bounded on the left by "/*" and on the right by "*/" :

```
/* This is a comment */
```

2. Names

Names in QUEL are sequences of no more than 12 alphanumeric characters, starting with an alphabetic. Underscore (_) is considered an alphabetic. All upper-case alphabetics appearing anywhere except in strings are automatically and silently mapped into their lower-case counterparts.

3. Keywords

The following identifiers are reserved for use as keywords and may not be used otherwise:

| | | |
|---|---|---|
| abs | all | and |
| any | append | ascii |
| atan | avg | avgu |
| by | concat | copy |
| cos | count | countu |
| create | delete | destroy |
| exp | float4 | float8 |
| from | gamma | help |
| in | index | int1 |
| int2 | int4 | into |
| is | log | max |
| min | mod | modify |
| not | of | on |
| onto | or | print |
| range | replace | retrieve |
| save | sin | sqrt |
| sum | sumu | to |
| until | where | |

## 4. Constants

There are three types of constants, corresponding to the three data types available in QUEL for data storage.

### 4.1. String constants

Strings in QUEL are sequences of no more than 255 arbitrary ASCII characters bounded by double quotes ( " " ). Upper case alphabetics within strings are accepted literally. Also, in order to imbed quotes within strings, it is necessary to prefix them with '\' . The same convention applies to '\' itself.

Only printing characters are allowed within strings. Non-printing characters (i.e. control characters) are converted to blanks.

### 4.2. Integer constants

Integer constants in QUEL range from -2,147,483,647 to +2,147,483,647. Integer constants beyond that range will be converted to floating point. If the integer is greater than -32,768 and less than 32,768 then it will be left as a two byte integer. Otherwise it is converted to a four byte integer.

### 4.3. Floating point constants

Floating constants consist of an integer part, a decimal point, and a fraction part. Scientific notation is not presently accepted. They are taken to be double-precision quantities with a range of approximately -10**38 to 10**38 and a precision of 17 decimal digits.

## 5. Attributes

An attribute is a construction of the form:

variable.domain

Variable identifies a particular relation and can be thought of as standing for the rows or tuples of that relation. A variable is associated with a relation by means of a RANGE statement. Domain is the name of one of the columns of the relation over which the variable ranges. Together they make up an attribute which represents values of the named domain.

## 6. Arithmetic operators

Arithmetic operators take numeric type expressions as operands. Unary operators group right to left; binary operators group left to right. The operators (in order of

descending precedence) are:

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

Parentheses may be used for arbitrary grouping. Overflow and divide by zero are NOT checked on integer operations. Overflow, underflow, and divide by zero on floating point numbers are checked only if the appropriate machine hardware exists and has been enabled. Arithmetic overflow and divide by zero are not checked on integer operations. Floating point operations are checked for overflow, underflow, and divide by zero only if the appropriate machine hardware exists and has been enabled.

7. Expressions (a_expr)

An expression is one of the following:

constant
attribute
functional expression
aggregate or aggregate function
a combination of numeric expressions and arithmetic operators

For the purposes of this document, an arbitrary expression will be refered to by the name "a_expr".

8. Formats

Every a_expr has a format denoted by a letter (c, i, or f for character, integer, or floating data types respectively) and a number indicating the number of bytes of storage occupied. Formats currently supported are listed below. The ranges of numeric types are indicated in parentheses.

c1 - c255 character data of length 1-255 characters

| | | |
|---|---|---|
| i1 | 1-byte integer | ( -128 to +127) |
| i2 | 2-byte integer | (-32768 to +32767) |
| i4 | 4-byte integer | (-2,147,483,648 to +2,147,483,647) |
| f4 | 4-byte floating | (-10**38 to +10**38, 7 decimal digit precision) |
| f8 | 8-byte floating | (-10**38 to +10**38, 17 decimal digit precision) |

One numeric format can be converted to or substituted for any other numeric format.

9. Type Conversion.

When operating on two numeric domains of different types, INGRES converts as necessary to make the types identical.

When operating on an integer and a floating point number, the integer is converted to a floating point number before the operation. When operating on two integers of different sizes, the smaller is converted to the size of the larger. When operating on two floating point number of different size, the larger is converted to the smaller.

The following table summarizes the possible combinations:

| | i1 | i2 | i4 | f4 | f8 |
|---|---|---|---|---|---|
| i1 - | i1 | i2 | i4 | f4 | f8 |
| i2 - | i2 | i2 | i4 | f4 | f8 |
| i4 - | i4 | i4 | i4 | f4 | f8 |
| f4 - | f4 | f4 | f4 | f4 | f4 |
| f8 - | f8 | f8 | f8 | f4 | f8 |

INGRES provides five type conversion operators specifically for overriding the default actions. The operators are:

```
int1(a_expr)   result type i1
int2(a_expr)   result type i2
int4(a_expr)   result type i4
float4(a_expr) result type f4
float8(a_expr) result type f8
```

The type conversion operators convert their argument a_expr to the requested type. A_expr can be anything including character. If a character value cannot be converted, an error occures and processing is halted. This can happen only if the syntax of the character value is incorrect.

Overflow is not checked on conversion.

10. Target_list

A target list is a parenthesized, comma separated list of one or more elements , each of which must be of one of the following forms:

a) result_attname IS a_expr

Result_attname is the name of the attribute to be created (or an already existing attribute name in the case of update statements.) The equal sign ("=") may be used interchangeably with IS. In the case where expr is anything other than a single attri-

bute, this form must be used to assign a result name to the expression.

b) attribute

In the case of a RETRIEVE, the resultant domain will acquire the same name as that of the attribute being retrieved. In the case of update statements (APPEND, REPLACE), the relation being updated must have a domain with exactly that name.

Inside the target list the keyword "all" can be used to represent all domains. For example:

```
range of e is employee
retrieve (e.all) where e.salary > 10000
```

will retrieve all domains of employee for those tuples which satisfy the qualification. "All" can be used in the target list of a retrieve or an append. The domains will be inserted in their "create" order; that is, the same order they were listed in the create statement.

11. Comparison operators

Comparison operators take arbitrary expressions as operands.

```
<   (less than)
<=  (less than or equal)
>   (greater than)
>=  (greater than or equal)
=   (equal to)
!=  (not equal to)
```

They are all of equal precedence. When comparisons are made on character attributes, all blanks are ignored.

12. Logical operators

Logical operators take clauses as operands and group left-to-right:

```
not  (logical NOT; negation)
and  (logical AND; conjunction)
or   (logical OR; disjunction)
```

NOT has the highest precedence of the three. AND and OR have equal precedence. Parentheses may be used for arbitrary grouping.

13. Qualification (qual)

A qualification consists of any number of clauses connected

by logical operators. A clause is a pair of expressions connected by a comparison operator:

```
a_expr  comparison_operator  a_expr
```

Parentheses may be used for arbitrary grouping. A qualification may thus be:

```
clause
NOT qual
qual OR qual
qual AND qual
(qual)
```

## 14. Functional expressions

A functional expression consists of a function name followed by a parenthesized (list of) operand(s). Functional expressions can be nested to any level. In the following list of functions supported (n) represents an arbitrary numeric type expression. The format of the result is indicated on the right.

| | |
|---|---|
| abs(n) | same as n (absolute value) |
| ascii(n) | character string (converts numeric to character) |
| atan(n) | f8 (arctangent) |
| concat(a,b) | character (character concatenation. see 16.2) |
| cos(n) | f8 (cosine) |
| exp(n) | f8 (exponential of n) |
| gamma(n) | f8 (log gamma) |
| log(n) | f8 (natural logarithm) |
| mod(n,b) | same as b (n modulo b. n and b must be i1, i2, or i4) |
| sin(n) | f8 (sine) |
| sqrt(n) | f8 (square root) |

## 15. Aggregate expressions

Aggregate expressions provide a way to aggregate a computed expression over a set of tuples.

### 15.1. Aggregation operators

The definitions of the aggregates are listed below.

```
count  -  (i4) count of occurrences
countu -  (i4) count of unique occurrences
sum    -  summation
sumu   -  summation of unique values
avg    -  (f8) average (sum/count)
avgu   -  (f8) unique average (sumu/countu)
```

max - maximum
min - minimum
any - (i2) value is 1 if any tuples satisfy the qualification, else it is 0

15.2. Simple aggregate

aggregation_operator ( a_expr [ WHERE qual] )

A simple aggregate evaluates to a single scalar value. A_expr is aggregated over the set of tuples satisfying the qualification (or all tuples in the range of the expression if no qualification is present). Operators sum and avg require numeric type a_expr; count, any, max and min permit a character type attribute as well as numeric type a-expr.

Simple aggregates are completely local. That is, they are logically removed from the query, processed separately, and replaced by their scalar value.

15.3. "any" aggregate

It is sometimes useful to know if any tuples satisfy a particular qualification. This can be done by using the aggregate "count" and checking whether it is zero or non-zero. Using "any" instead of "count" is more efficient since processing is stopped if possible the first time a tuple satisfies a qualification. "Any" returns 1 if the qualification is true and 0 otherwise.

15.4. Aggregate functions

aggregation_operator ( a_expr BY by_domain
{, by_domain} [WHERE qual ] )

Aggregate functions are extensions of simple aggregates. The BY operator groups (i.e. partitions) the set of qualifying tuples by by_domain values. For more than one by_domain, the values grouped by are the concatenation of individual by_domain values. A_expr is as in simple aggregates. The aggregate function evaluates to a set of aggregate results, one for each partition into which the set of qualifying tuples has been grouped. The aggregate value used during evaluation of the query is the value associated with the partition into which the tuple currently being processed would fall.

Unlike simple aggregates, aggregate functions are not completely local. The "by-list", which differentiates aggregate functions from simple aggregates, is global to the query. Domains in the "by list" are automatically linked to the other domains in the query which are in

the same relation.

Example:
```
/* retrieve the average salary for the employees working
for each manager */
range of e is employee
retrieve (e.manager, avesal=avg(e.salary by e.manager))
```

15.5 Aggregates on Unique Values.

It is occasionally necessary to aggregate on unique values of an expression. The "avgu", "sumu", and "countu" all remove duplicate values before performing the aggregation. For example:

```
count(e.manager)
```

would tell you how many occurrences of e.manager exist. But

```
countu(e.manager)
```

would tell you how many unique values of e.manager exist.

16. Special character operators

There are three special features which are particular to character domains.

16.1 Pattern matching characters

There are four characters which take on special meaning when used in character constants (strings):

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

These characters can be used in any combination to form a variety of tests. For example:

```
where e.name = "*"    matches any name.
where e.name = "E*"   matches any name starting with
          "E".
where e.name = "*ein"  matches all names ending
          with "ein"
where e.name = "*[aeiou]*"  matches any name with
          at least one vowel.
where e.name = "Smith?"  matches any six character
          name starting with "Smith".
where e.name = "[A-J]*"  matches any name starting
          with A,B,..,J.
```

The special meaning of the pattern matching characters

can be disabled by preceding them with a "\". Thus "*" refers to the character "*". When the special characters appear in the target list they must be escaped. For example:

```
title = "\*\*\* ingres \*\*\*"
```

is the correct way to assign the string "*** ingres ***" to the domain "title".

16.2 Concatenation

There is a concatenation operator which can form one character string from two. Its syntax is "concat(field1, field2)". The size of the new character string is the sum of the sizes of the original two. Trailing blanks are trimmed from the first field, the second field is concatenated and the remainder is blank padded. Concat can be arbitrarily nested inside other concats. For example:

```
name = concat(concat(x.lastname, ","), x.firstname)
```

will concatenate x.lastname with a comma and then concatenate x.firstname to that.

16.3 Ascii (numeric to character translation)

The ascii function can be used to convert a numeric field to its character representation. This can be useful when it is desired to compare a numeric value with a character value. For example:

```
retrieve ( ... )
        where x.chardomain = ascii(x.numdomain)
```

Ascii can be applied to a character value. The result is simply the character value unchanged. The conversion format is determined by the printing format (see ingres(unix)).

SEE ALSO

range(quel), retrieve(quel), append(quel), delete(quel), replace(quel), ingres(unix)

BUGS

The maximum number of variables which can appear in one query is 6.

Scientific notation cannot be used for floating point constants.

Numeric overflow, underflow, and divide by zero is not detected.

When converting numeric to numeric overflow is not checked.

NAME

range - declare a variable to range over a relation

SYNOPSIS

RANGE OF variable IS relname

DESCRIPTION

Range is used to declare variables which will be used in subsequent QUEL statements. The variable is associated with the relation specified by relname. When the variable is used in subsequent statements it will refer to a tuple in the named relation. A range declaration remains in effect for an entire INGRES session (until exit from INGRES), until the variable is redeclared by a subsequent range statement, or until the relation is removed with the destroy command.

EXAMPLE

```
/* Declare tuple variable e to range over relation emp */
   range of e is emp
```

SEE ALSO

quel(quel), destroy(quel)

DIAGNOSTICS

BUGS

Only 10 variable declarations may be in effect at any time. After the 10th range statement, the least recently referenced variable is re-used for the next range statement.

NAME

replace - replace values of domains in a relation

SYNOPSIS

REPLACE tuple_variable (target_list) [WHERE qual]

DESCRIPTION

Replace changes the values of the domains specified in the target_list for all tuples which satisfy the qualification. The tuple_variable must have been declared to range over the relation which is to be modified. Note that a tuple variable is required and not the relation name. Only domains which are to be modified need appear in the target_list. These domains must be specified as result_attnames in the target_list either explicitly or by default (see QUEL).

Numeric domains may be replaced by values of any numeric type (with the exception noted below). Replacement values will be converted to the type of the result domain.

EXAMPLE

```
/* Give all employees who work for Smith a 10 raise */
   range of e is emp
   replace e(sal = 1.1*e.sal) where e.mgr = "Smith"
```

SEE ALSO

quel, range

DIAGNOSTICS

Use of a numeric type expression to replace a character type domain or vice versa will produce diagnostics.

BUGS

NAME

retrieve - retrieve tuples from a relation

SYNOPSIS

RETRIEVE [[INTO] relname] (target_list) [WHERE qual]

DESCRIPTION

Retrieve will get all tuples which satisfy the qualification and either display them on the terminal (standard output) or store them in a new relation.

If a relname is specified, the result of the query will be stored in a new relation with the indicated name. A relation with this name owned by the user must not already exist. The current user will be the owner of the new relation. The relation will have domain names as specified in the target_list result_attnames. The new relation will be saved on the system for seven (7) days unless explicitly saved by the user until a later date.

The keyword "ALL" can be used when it is desired to retrieve all domains.

If no result relname is specified then the result of the query will be displayed on the terminal and will not be saved. Duplicate tuples are not removed when the result is displayed on the terminal.

The format in which domains are printed can be defined at the time ingres is invoked (see ingres(unix)).

If a result relation is specified then the default procedure is to modify the result relation to an CHEAPSORT storage structure removing duplicate tuples in the process.

If the default CHEAPSORT structure is not desired, the user can override this at the time INGRES is invoked by using the "-r" switch (see ingres).

EXAMPLE

```
/* Find all employees who make more than their manager */
   range of e is emp
   range of m is emp
   retrieve (e.name) where e.mgr = m.name
           and e.sal > m.sal
/* Retrieve all domains for those who make more
       than the average salary */
   retrieve into temp (e.all) where e.sal > avg(e.sal)
```

SEE ALSO
modify(quel), quel(quel), range(quel), save(quel), ingres(unix)

DIAGNOSTICS

BUGS

NAME

save - save a relation until a date.

SYNOPSIS

SAVE relname UNTIL month day year

DESCRIPTION

Save is used to keep relations beyond the default 7 day life span.

Month can be an integer from one through twelve, or the name of the month, either abbreviated or spelled out.

Only the owner of a relation can save that relation. There is an INGRES process which typically removes a relation immediately after its expiration date has passed.

The actual program which destroys relations is called purge. It is not automatically run. It is a local decision when expired relations are removed.

System relations have no expiration date.

EXAMPLE

```
/* Save the emp relation until the end of February 1987 */
   save emp until feb 28 1987
```

SEE ALSO

retrieve(quel), create(quel), purge(unix)

DIAGNOSTICS

bad month, day, or year
not the owner
system relation

BUGS

NAME

copydb - create batch files to copy out a data base and restore it.

SYNOPSIS

copydb [-uxx] database full-path-name-of-directory {relation}

DESCRIPTION

Copydb creates two files in the directory. Copy.out, which contains Quel instuctions which will copy all relations owned by the user into files in the named directory, and copy.in, which contains instructions to copy the files into relations, create indexes and do modifies. The files will have the same names as the relations with the users Ingres id tacked on the end. (The directory MUST NOT be the same as the data base directory as the files have the same names as the relation files.) The -u flag may be used to run copydb with a different user id. (The fact that copydb creates the copy files does not imply that the user can necessarily access the specified relation). If relation names are specified only those relations will be included in the copy files.

Copydb is written in Equel and will access the database in the usual manner. It DOES NOT have to run as INGRES.

EXAMPLE

```
chdir /mnt/mydir
copydb db /mnt/mydir/backup
ingres db <backup/copy.out
tp r1 backup
rm -r backup

tp x1
ingres db <backup/copy.in
```

DIAGNOSTICS

The copydb program will give self-explanitory diagnostics. If "too many indexes" is reported it means that more than ten indexes have been specified on one relation. The constant can be increased and the program recompiled. Other limits are set to the system limits.

BUGS

Copydb assumes that indexes which are Isam do not need to be remodified. Copydb cannot tell if the relation was modified with a fillfactor or minpages specification. The copy.in file may be edited to reflect this.

NAME

creatdb - create a new data base

SYNOPSIS

creatdb [-uxx] [+-c] [-e] [-m] dbname

DESCRIPTION

Creatdb creates a new INGRES data base, complete with the necessary directory structure and system relations. The data base administrator (DBA) for the new data base is set to the identity of the user who issued the CREATDB command or the user with code "xx". All system relations are owned by the DBA. Only the INGRES user may use the -u flag.

The -c flag is used if the data base is for a single user and no concurrency control is needed. Note that even if users of the same data base only update their private relations, they still share the same system catalogs and thus require concurrency control. Therefore, the -c flag must only be used for a truly single user data base.

The -e flag indicates that the data base already exits and CREATDB is being used to modify the concurrency control option as specified by +-c. In this case +c will cause concurrency control to be enforced. The -c flag will turn concurrency control off.

The INGRES superuser must authorize a user to execute the creatdb command by setting the 000001 bit in the status field of the users file for that user.

The -m flag is used when the Unix directory for the new database already exists. This is useful when a database will be on a separate mounted Unix file system.

EXAMPLE

```
creatdb demo
creatdb -uav erics_db
```

SEE ALSO

demodb(unix)
destroydb(unix)
users(files)

DIAGNOSTICS

illegal database name -- database name is not a legal name, i.e., it is more than 14 characters long, or it begins with a non-alphabetic character, or it has a non-alphanumeric character in it.

database already exists -- the database name you have specified has already been used.

you may not access this database -- this database name is

not permitted to you based on the contents of the users file.

you are not allowed this command -- the status entry in the users file does not have creatdb permission set.

NAME

destroydb - destroy an existing database

SYNOPSIS

destroydb [-s] [-m] dbname

DESCRIPTION

Destroydb will remove all reference to an existing database. The directory of the database and all files in that directory will be removed.

To execute this command the current user must be the database administrator for the database in question, or must be the INGRES superuser and have the -s flag stated.

The -m flag causes destroydb not to remove the Unix directory. This is useful when the directory is a separate mounted Unix file system.

EXAMPLE

```
destroydb demo
destroydb -s erics_db
```

SEE ALSO

creatdb(unix)

DIAGNOSTICS

invalid dbname -- the database name specified is not a valid name.

you may not reference this database -- the database may exist, but you do not have permission to do anything with it.

you may not use the -s flag -- you have tried to use the -s flag, but you are not the INGRES superuser.

you are not the dba -- someone else created this database.

database does not exist -- this database does not exist.

NAME

equel - Embedded QUEL interface to C

SYNOPSIS

equel [-d] file.q ...

DESCRIPTION

Equel provides the user with a method of interfacing the general purpose programming language "C" with INGRES. It consists of the equel precompiler and the equel object library.

The precompiler is invoked with the statement:

```
equel [-d] file.q ...
```

where file.q is the source input file name, which must end with ".q". The output is written to the file "file.c". The -d flag causes additional debug information to be printed at run time, see below. As many files as wished may be specified. The output files may than be compiled using the C compiler:

```
cc file.c ... -lq
```

The "-lq" requests the use of the equel object library.

All equel routines and globals begin with the characters "II", and so all globals variables and procedure names of the form IIxxx are reserved for use by equel.

Equel commands are indicated by lines which begin with a double pound sign ("##"). Other lines are simply copied as is. All normal INGRES commands may be used in equel and have the same effect as if invoked through the interactive terminal monitor. Only retrieve commands with no result relation specified have a different syntax and meaning. Also, the following equel commands are permissable.

## ingres [ingres flags] data_base_name

This command has the same effect as invoking INGRES from the UNIX shell. It is not permissible to execute this command twice without an intervening ## exit. Each flag should be enclosed in quotes to avoid confusion in the Equel parser:

```
## ingres "-f4f10.2" "-i212" demo
```

## exit

Exit simply exits from INGRES. It is equivalent to the \q command to the teletype monitor.

```
## int C-variable {, C-variable} ;
## long C-variable {, C-variable} ;
## float C-variable {, C-variable} ;
## double C-variable {, C-variable} ;
## char *C-variable {, C-variable} ;
```

## char C-variable[integer] {, C-variable[integer]} ;

These commands all declare C-variables. They may be used as normal variables in the C language, and may also be used in equel statements. The variables are global with respect to equel, but obey the normal C scope rules. All variables must be declared before being used. Anywhere a constant may appear in an INGRES command, a C-variable may appear. The value of the C-variable is substituted at execution time. If a variable of one of the char types is used almost anywhere in a equel statement, the contents of that variable is used at object time. For example:

```
## char *dbname;
dbname = "demo";
## ingres dbname
```

will cause INGRES to be invoked with data base "demo".

The format of retrieve without a result relation is modified to:

```
## retrieve (C-variable=a_fcn {, C-variable=a_fcn} )
##     [WHERE qual] {
C-code
## }
```

This statement causes the "C-code" to be executed once for each tuple retrieved, with the "C-variable"s set appropriately. Numeric values of any type are converted as necessary. No conversion is done between numeric and charactor values. (The normal Ingres ascii function may be used for this purpose.)

INGRES and run-time EQUEL errors cause the routine IIerror to be called, with the error number and the parameters to the error in an array of string pointers as in a C language main routine. The error number will be printed with the error parameters. In addition if the "-d" flag was set the file name and line number of the error will be printed. This information is useful in debugging but can take up process space. If the error was from INGRES (error number >= 2000) then IIerrflag is set so that retrieves will terminate properly. The user may write an IIerror routine to look up the messages in .../files/error?, or do other tasks.

Interrups are caught by equel, if they are not being ignored. This insures that the rest of Ingres is in sync with the Equel process. There is a function pointer, IIinterrupt, which points to a function to call after the interrupt is caught, the user may use this to service the interupt. It is initialized to "exit" and is called with -1 as its argument. For example:

```
extern int      *IIinterrupt;
extern          reset();

setexit();
```

```
        IIinterrupt = &reset();
        mainloop();
```

SEE ALSO

.../demo/equeltut.q
C manual
ingres(UNIX)
quel(QUEL)

FILES

.../files/error? can be used by the user to decifer Ingres error numbers. /lib/libq.a run time library.

BUGS

The C-code embedded in the tuple-by-tuple retrieve operation may not contain additional QUEL statements or recursive invocations of INGRES.

There is no way to specify an i1 format c-variable.

Single QUEL statements can expand to more than one c statements. Equel should bracket the expression with braces.

NAME

geoquel - GEO-QUEL data display system

SYNOPSIS

geoquel [<flags>] dbname

DESCRIPTION

This is the UNIX command which is used to invoke GEO-QUEL. Dbname is the name of an existing data base.

The format of the graphic output depends upon the type of terminal in use. GEO-QUEL will look up the terminal type at login time and produce output appropriate for that terminal. If the terminal in use is incapable of drawing graphic output then a display list is generated for a Tek 4014 but will only be displayed if the results are saved with SAVEMAP and then re-displayed.

The optional <flags> may be combinations of thsse:

-s Don't print any of the monitor messages, including prompts. This is inclusive of the dayfile.
-d Don't print the dayfile.
-a Disable the autoclear function in the terminal monitor.
-tT Set the terminal type to T. T can be gt40, gt42, 4014 for DEC's GT40-GT42, and Tek's 4014.

EXAMPLE

```
geoquel demo
geoquel -d demo
geoquel -s demo < batchfile
```

SEE ALSO

GEO-QUEL reference manual

DIAGNOSTICS

The diagnostics produced by GEO-QUEL are intended to be self-explanatory. Occasional messages may be produced by INGRES; for an explanation of these messages see the INGRES system documentation.

NAME

ingres - INGRES relational data base management system

SYNOPSIS

ingres [<flags>] dbname [process_table]

DESCRIPTION

This is the UNIX command which is used to invoke INGRES. Dbname is the name of an existing data base. The optional flags have the following meanings (a "+-" means the flag may be stated "+x" to set option x or "-x" to clear option x. "-" alone means that "-x" must be stated to get the x function):

+-U: Enable/disable direct update of the system relations and secondary indicies. You must have the 000004 bit in the status field of the users file set for this flag to be accepted. This option is provided for system debugging and is strongly discouraged for normal use.

-uXX: Pretend you are the user with code XX (found in the users file). This may only be used by the DBA for the database or by the INGRES superuser.

-cN: Set the minimum field width for printing character domains to N. The default is 6.

-iLN: Set integer output field width to N. L may be 1, 2, or 4 for i1's, i2's, or i4's repectively.

-fLXM.N: Set floating point output field width to M characters with N decimal places. L may be 4 or 8 to apply to f4's or f8's respectively. X may be e, E, f, F, g, G, n, or N to specify an output format. E is exponential form, F is floating point form, and G and N are identical to F unless the number is too big to fit in that field, when it is output in E format. G format guarantees decimal point alignment; N does not. The default format for both is n10.3.

-vX: Set the column seperator for retrieves to the terminal and print commands to be X. The default is vertical bar.

-rM: Set modify mode on the RETRIEVE INTO command to M. M may be "isam", "cisam", "hash", "chash", "heap", "cheap", "heapsort", or "cheapsort", for ISAM, compressed ISAM, hash table, compressed hash table, heap, compressed heap, sorted heap, or compressed sorted heap. The default is "cheapsort".

-nM: Set modify mode on the INDEX command to M. M can take the same values as the "-r" flag above. Default is isam.

+-a: Set/clear the autoclear option in the terminal monitor. It defaults to set.

+-b: Set/reset batch update. Users must the 000002 bit set in the status field of the users file to clear this flag. This flag is normally set. When clear, queries will run slightly faster, but no recovery can take place. Queries which update a secondary index automatically set this flag for that query only.

+-d: Print/don't print the dayfile. Normally set.

+-s: Print/don't print any of the monitor messages, including prompts. This flags is normally set. If cleared, it also clears the -d flag.

+-w: Wait/don't wait for the database. If the +w flag is present, INGRES will wait if certain processes are running (purge,restore, and/or sysmod) on the given data base. Upon completion of those processes INGRES will proceed. If the -w flag is present, a message is returned and execution stopped if the data base is not available. If the +-w flag is omitted and the data base is unavailable, the error message is returned if INGRES is running in forground( more precisly if the standard input is from a terminal). Otherwise the wait option is invoked.

Process_table is the pathname of a UNIX file which may be used to specify the run-time configuration of INGRES. This feature is intended for use in system maintenance only, and its unenlightened use by the user community is strongly discouraged.

Note: It is possible to run the monitor as a batch-processing interface using the '<', '>' and '|' operators of the UNIX shell, provided the input file is in proper monitor-format.

EXAMPLE

```
ingres demo
ingres -d demo
ingres -s demo < batchfile
ingres -f4g12.2 -i13 +b -rhash demo
```

SEE ALSO

monitor(quel)
users(files)

DIAGNOSTICS

too many options to ingres -- you have stated too many flags as ingres options.

invalid user -- in using the "-u" flag, you have given a user code which does not exist.

bad flag format -- you have stated a flag in a format which is not intelligible, or a bad flag entirely.

too many parameters to ingres -- you have given a database name, a process table name, and "something else" which ingres doesn't know what to do with.

no database name specified -- you have failed to give a database name.

improper database name -- the database name is not legal.

you may not access this database -- according to the users file, you do not have permission to enter this database.

you are not authorized to use these flags -- one or more of the flags you have specified require a special permission which you do not have.

cannot access database -- the database which you have specified does not exist.

you may not use the -u flag -- you are not the DBA for the database specified.

NAME

printr - print relations

SYNOPSIS

printr [<flags>] database relation {relation}

DESCRIPTION

Printr prints the named relation(s) out of the database specified, exactly like the PRINT command.

Flags accepted are -u, +-w, -c, -i, -f, and -v. Their meanings are identical to the meanings of the same flags in INGRES.

SEE ALSO

ingres(unix)
print(quel)

DIAGNOSTICS

bad flag -- you have specified a flag which is not legal or is in bad format.
you may not access database -- this database is prohibited to you based on status information in the users file.
cannot access database -- the database does not exist.

NAME

purge - destroy all expired and temporary relations

SYNOPSIS

purge [-f] [-a] [-p] [-s] [-w] [+w] {database}

DESCRIPTION

Purge goes through named databases and deletes system temporary relations and miscellaneous files. If the -p flag is stated, it also deletes expired user relations.

You must be the DBA to execute this command, or state the -s flag and be the INGRES superuser.

If the -f flag is present all unrecognized files are deleted otherwise their existence is just reported. If no database names are specified, all databases for which you are the DBA are purged. If the -s flag is stated, all databases are purged if no specific databases are mentioned. If the -a flag is stated, you are asked before each database whether or not you want to purge that database. A response beginning "y" is yes; anything else is no.

If the data base is being used while Purge is working errors may occur, so Purge will lock the data base while it is being processed. If a data base is busy Purge will report this and go on to the next data base, if any. If standard input is not a terminal Purge will wait for the data base to be free. If -w flag is stated Purge will not wait, regardless of standard input. The +w flag causes Purge to always wait.

EXAMPLES

```
purge -p +w tempdata
purge -a -f
```

SEE ALSO

save(quel)
restore(unix)

DIAGNOSTICS

who are you? -- you are not entered into the users file.
not ingres superuser -- you have tried to use the -s flag but you are not the INGRES superuser.
you are not the dba -- you have tried to purge a database for which you are not the DBA.
cannot access database -- the database does not exist.

BUGS

NAME

restore - recover from an INGRES or UNIX crash.

SYNOPSIS

restore [-a] [-s] [-w] [+w] {database}

DESCRIPTION

Restore is used to restore a data base after an INGRES or UNIX crash. It should always be run after any abnormal termination to ensure the integrity of the data base.

In order to run restore, you must be the DBA for the database you are restoring or the INGRES superuser and specify the -s flag.

If no databases are specified then all databases for which you are the DBA are restored. If the -s flag is specified then all databases are restored.

If the -a flag is specified you will be asked before restore takes any serious actions. It is advisable to use this flag if you suspect the database is in bad shape. Using /dev/null as input with the -a flag will provide a report of problems in the data base. If there were no errors while restoring a database, purge will be called, with the same flags that were given to restore, to remove unwanted files and system temporaries. Restore may be called with the -f and/or -p flags for purge. Unrecognized files and expired relations are not removed unless the proper flags are given. In the case of an incomplete destroy, create or index restore will not delete files for partially created or destroyed relations. Purge must be called with the -f flag to accomplish this.

Restore locks the data base while it is being processed. If a data base is busy Restore will report this and go on to the next data base. If standard input is not a terminal Restore will wait for the data base to be free. If the -w flag is set Restore will not wait regardless of standard input. If +w is set it will always wait.

Restore can recover a database from an update which had finished filling the batch file. Updates which did not make it to this stage should be rerun. Similarly modifies which have finished recreating the relation will be completed (the relation relation and attribute relations will be updated). If a destroy was in progress it will be carried to completion, while a create will almost always be backed out. Destroying a relation with an index should destroy the index so restore may report that a secondary relation has been found with no primary.

EXAMPLE

```
restore -f demo
restore -a grants < /dev/null
```

DIAGNOSTICS

All diagnostics are followed by a tuple from the attribute, relation or indexes relations.

"No relation for attribute(s):" the attributes listed have no corresponding entry in the relation relation

"No primary relation for index:" the tuple printed is the relation tuple for a secondary index for which there is no primary relation. The primary probably was destroyed the secondary will be.

"No indexes entry for primary relation:" the tuple is for a primary relation, the relindxd domain will be set to zero. This is the product of an incomplete destroy.

"No indexes entry for index:" the tuple is for a secondary index, the index will be destroyed. This is the product of an incomplete destroy.

"RELNAME is index for:" an index has been found for a primary which is not marked as indexed. The primary will be so marked. This is probably the product of an incomplete index command. The index will have been created properly but not modified.

"No file for:" There is no data for this relation tuple, the tuple will be deleted. If, under the -a option, the tuple is not deleted purge will not be called.

"No secondary index for indexes entry:" An entry has been found in the indexes relation for which the secondary index does not exist (no relation relation tuple). The entry will be deleted.

SEE

purge(unix)

NAME

sysmod - modify system relations to predetermined storage structures.

SYNOPSIS

SYSMOD [-s] [-w] [+w] dbname {RELATION} {ATTRIBUTE} {INDEXES}

DESCRIPTION

SYSMOD will modify the relation, attribute and indexes relations to hash unless at least one of the RELATION, ATTRIBUTE or INDEXES parameters are given, in which case only those relations given as parameters are modified. The system relations are modified to gain maximum access performance when running INGRES. The user must be the data base administrator for the specified database, or be the INGRES superuser and have the -s flag stated.

SYSMOD should be run on a data base when it is first created and periodically thereafter to maintain peak performance. If many relations and secondary indices are created and/or destroyed, SYSMOD should be run more often.

If the data base is being used while Sysmod is running, errors will occur. Therefore, Sysmod will lock the data base while it is being processed. If the data base is busy, Sysmod will report this. If standard input is not a terminal Sysmod will wait for the data base to be free. If -w flag is stated Sysmod will not wait, regardless of standard input. The +w flag causes Sysmod to always wait.

The system relations are modified to hash; the relation relation is keyed on the first domain, the indexes and attribute relations are keyed on the first two domains. The relation and attribute relations have the minpages option set at 10, the indexes relation has a minpages value of 5.

SEE ALSO

modify(quel)

NAME

usersetup - setup users file

SYNOPSIS

.../bin/usersetup [pathname]
(normally executable by INGRES super-user only)

DESCRIPTION

The /etc/passwd file is read and reformatted to become the INGRES users file, stored into .../files/users. If pathname is specified, it replaces "...".

After running usersetup, the users file must be editted. Any users who are to be authorized to execute the creatdb or demodb commands must have the 000001 bit in the fifth field of the file set. Any users who are to be permitted to override batch update with the -b flag must have the 000002 bit set. Users who are to be permitted to use the -U flag to directly update system relations and secondary indicies must have the 000004 bit set. To disable a user from executing INGRES entirely, completely remove her line from the users file.

As UNIX users are added or deleted, the users file will need to be editted to reflect the changes. For deleted users, it is only necessary to delete the line for that user from the users file. To add a user, you must assign that user a code in the form "aa" and enter a line in the users file in the form:

name:cc:uid:gid:status:flags:proctab:::databases

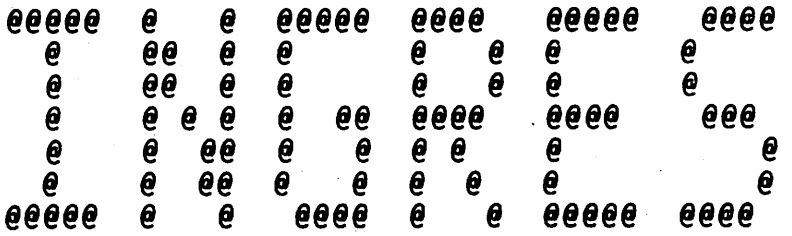
where name is the user name (taken from the first field of the /etc/passwd file entry for this user), cc is the user code assigned, which must be exactly two characters long and must not be the same as any other existing user codes, uid and gid are the user and group ids (taken from the third and fourth fields in the /etc/passwd entry), status is the status bits for this user, normally 000000, flags are the default flags for INGRES (on a per-user basis), proctab is the default process table for this user (which defaults to "=proctab6.0"), and databases is a list of the databases this user may enter. If null, she may enter all databases. If the first character is a dash ("-"), the field is a comma seperated list of databases which she may not enter. Otherwise, it is a list of databases which she may enter.

The databases field includes the names of databases which may be created. When a database is created, there is no way of protecting access to it, except by contacting the INGRES superuser.

Usersetup may be executed only once, to initially create the users file.

SEE ALSO
ingres(unix)
passwd(V)
users(files)

BUGS
It should be able to bring the users file up to date.



# S U P P O R T  F I L E S

Version 6

12/20/77

by

Eric Allman

NAME

INGRES support files

DESCRIPTION

The directory .../files contains a number of files necessary to run INGRES and its support software. This documentation describes the format and use of these files.

NAME

.../files/dayfile6.1 - INGRES login message

DESCRIPTION

The contents of dayfile reflect user information of general system interest, and is more or less analogous to /etc/motd in UNIX. The file has no set format; it is simply copied at login time to the standard output device by the monitor if the -s or -d options have not been requested. Moreover the dayfile is not mandatory, and its absence will not generate errors of any sort; the same is true when the dayfile is present but not readable.

NAME

.../files/dbtemplate - database template

DESCRIPTION

This routine contains the template for a database used by creatdb. It has a set of entries for each relation to be created in the database. The sets of entries are seperated by a blank line. Two blank lines or an end of file terminate the file.

The first line of each set of entries is the name of the relation. The second through last lines contain the domain type, a tab, and the domain name.

The first set of entries must be for the relation catalog, and the second set must be for the attribute catalog.

EXAMPLE

```
relation
c12     relid
c2      relowner
i1      relspec
i1      relindxd

attribute
c12     attrelid
c2      attowner
c12     attname

(other relation descriptors)
```

SEE ALSO

creatdb(UNIX)

NAME

.../files/error? - files with INGRES errors

DESCRIPTION

These files contain the INGRES error messages. There is one file for each thousands digit; hence, error number 2313 will should be in file error2.

Each file consists of a sequence of error messages with associated error numbers. When an error enters process one, the appropriate file is scanned for the correct error number. If found, the message is printed; otherwise, the first message parameter is printed.

Each message has the format <errnum> <TAB> <message> <tilde>. Messages are terminated by the tilde character ("~"). The message is scanned before printing. If the sequence "%n" is encountered (where n is a digit from 0 to 9), parameter n is substituted, where %0 is the first parameter.

The parameters can be in any order. For example, an error message can reference %2 before it references %0.

SEE ALSO

error(UTIL)

EXAMPLE

```
1003   line %0, bad database name %1~
1005   In the purge of %1,
a bad %0 caused execution to halt~
1006   No process, try again.~
```

NAME

pipes.h - interprocess pipe format

SYNOPSIS

```
# include       ".../source/pipes.h"
```

DESCRIPTION

Pipes.h is the header file for using rdpipe, wrpipe, and proc_error. It contains the defined constants for the buffer size, the statuses of the blocks being passed, and the struct for piping data from one process to another.

There are four statuses with which a block can be passed, NORM_STAT (normal status), LAST_STAT (last block of command or message), SYNC_STAT (for synchronizing delete signals), and ERR_STAT (for synchronizing the processes in the event of an error).

As for the struct, there are 8 bytes (HDRSIZ) of header info, a 120 byte (PBUFSIZ) buffer, and an index for the next available spot in the buffer. Upon writing this struct only HDRSIZ + PBUFSIZ bytes are written. In the header there are two bytes, exec_id and func_id, that are used by the programs, a full word for the error number, err_id, a byte for the status of the current block, hdrstat, and a byte with the number of signficant characters in the buffer.

SEE ALSO

rdpipe(UTIL)
wrpipe(UTIL)
proc_error(UTIL)
proctab(FILES)

NAME

.../files/proctab6.1

DESCRIPTION

The contents of .../files/proctab6.1 describe the internal configuration of its object code programs to the INGRES control system. Each line (where 'line' is defined in the text-editor sense) has a special meaning. Lines 1 through 5, respectively, are full UNIX pathnames of those sites wherein the initial overlay program, decomposition process, one variable query processor, parser, and terminal monitor may be found. The sixth line is the full pathname of the sort program used in modify to ISAM. Subsequent lines are of the form

cmdname:ef

where cmdname is the name of a valid QUEL command, e is the code character of the process which handles the command. The current code characters are:

* ovqp

$ decomp (for historical reasons)

Anything else is taken to be an overlay (ie. DBU). The code character is appended to the last character of the pathname for the overlay process. F is the index of the associated command in the calling sequence for the process. Hence, a line of the form

help:a0

would signify to INGRES that the help command may be found in /mnt/ingres/bin/overlaya, indexed by argument number 0. The maximum number of such lines is regulated by the manifest constant MAXPROC, defined in the parser.

NAME

.../files/startup - INGRES startup file

DESCRIPTION

This file is read by the monitor at login time. It is read before the user startup file specified in the users file. The primary purpose is to define a new editor and/or shell to call with the \e or \s commands.

SEE ALSO

monitor(quel)
users(files)

NAME

.../files/users - INGRES user codes and parameters

DESCRIPTION

This file contains the user information in fields seperated by colons. The fields are as follows:

* User name, taken directly from /etc/passwd file.
* User code, assigned by the ingres super-user. It must be a unique two character code.
* UNIX user id. This MUST match the entry in the /etc/passwd file.
* UNIX group id. Same comment applies.
* Status word in octal. Bit values are:
  000001 creatdb permission
  000002 permits batch update override
  000004 permits update of system catalogs
  100000 ingres superuser
* A list of flags automatically set for this user.
* The process table to use for this user.
* An initialization file to read be read by the monitor at login time.
* Unassigned.
* Comma seperated list of databases. If this list is null, the user may enter any database. If it begins with a '-', the user may enter any database except the named databases. Otherwise, the user may only enter the named databases.

EXAMPLE

```
ingres:aa:5:2:177777:-d:=special:/mnt/ingres/ingres.init::
guest:ah:35:1:000000:::::demo,guest
```

SEE ALSO

initucode(UTIL)

# THE INGRES ERROR MESSAGES

This document describes the error returns which are possible from the INGRES data base system and gives an explanation of the probable reason for their occurrence. In all cases the errors are numbered nxxx where n indicates the source of the error. Basically,

1 = equel preprocessor

2 = parser

4 = decomposition and one variable query processor

5 = data base utilities

For a description of these routines the reader is referred to "The Design and Implementation of INGRES". The xxx in an error number is an arbitrary identifier.

The error messages are stored in the file .../files/errorn, where n is defined as above. The format of these files is the error number, a tab character, the message to be printed, and the tilde character ("~") to delimit the message.

In addition many error messages have "%i" in their body where "i"

is a digit interpreted as an offset into a list of parameters returned by the source of the error. This indicates that a parameter will be inserted by the error handler into the error return. In most cases this parameter will be self explanatory in meaning.

Where the error message is thought to be completely self explanatory, no additional description is provided.

## EQUEL ERRORS

1000 In domain %0 numeric retrieved into char field.~

Equel does not support conversion at run-time of numeric data from the data base to character string representation. Hence, if you attempt to assign a domain of numeric type to a C-variable of type character string, you will get this error message. To convert numerics to characters use the "ascii" function in QUEL.

1001 Numeric overflow during retrieve on domain %0.~

You will get this error if you attempt to assign a numeric data base domain to a C-variable of a numeric type but with a shorter length. In this case the conversion routines may generate an overflow. For example, this error will result from an attempt to retrieve a large floating point number into a C-variable of type integer.

1002 In domain %0, charactor retrieved into numeric variable.~

This error is the converse of error 1000.

## PARSER ERRORS

2100 line %0, Attribute '%1' not in relation '%2'~

This indicates that in a given line of the executed workspace the indicated attribute name is not a domain in the indicated relation.

2101 line %0, Logical operations are not allowed in the target list~

This indicates that a logical operator (and, or, not) is used as part of the target list of a QUEL statement. This is not allowed.

2102 line %0, No operations on characters allowed~

This indicates that an attempt was made to perform an arithmetic operation on a character domain or character constant. For example, "character" * 2 is not an allowed operation.

2103 line %0, Function type does not match type of attribute '%1'~

This error will be returned if a function expecting numeric data is given a character string or vice versa. For example, it is illegal to take the SIN of a character domain.

2105 line %0, You must rename 'tid' when it appears in the target list~

The reserved symbol "tid" (tuple identifier) cannot be the name of a domain. Consequently, it must be renamed when used as part of the target list. Tid is used for debugging system code.

2106 line %0, Data base utility command buffer overflow~

This error will result if a utility command is too long for the buffer space allocated to it in the parser. You must shorten the command or recompile the parser.

2107 line %0, You are not allowed to update this relation: %1~

This error will be returned if you attempt to update any system relation or secondary index directly in QUEL (such as the RELATION relation). Such operations which compromise the integrity of the data base are not allowed.

2108 line %0, Invalid result relation for APPEND '%1'~

This error message will occur if you execute an append command to a relation that does not exist, or that you cannot access. For example, append to junk( ... ) will fail if junk does not exist.

2109 line %0, Variable '%1' not declared in RANGE statement~

Here, a symbol was used in a QUEL expression in a place where a tuple variable was expected and this symbol was not defined via a RANGE statement.

2111 line %0, Too many attributes in key for INDEX~

A secondary index may have no more than 6 keys.

2117 line %0, Invalid relation name '%1' in RANGE statement~

You are declaring a tuple variable which ranges over a relation which does not exist.

2118 line %0, Out of space in query tree - Query too long~

You have the misfortune of creating a query which is too long for the parser to digest. The only options are to shorten the query or recompile the parser to have more buffer space for the query tree.

2119 line %0, MOD operator not defined for floating point or character attributes~

2120 line %0, no pattern match operators allowed in the target list~

2121 line %0, Only character type domains are allowed in CONCAT operator~

2123 line %0, '%1.all' not defined for replace~

This message should be self explanitory.

2500 syntax error on line %0
last symbol read was: %1~

A 2500 error is reported by the parser if it cannot otherwise classify the error. One common way to obtain this error is to omit the required parentheses around the target list. The parser reports the last symbol which was obtained from the scanner. Sometimes, the last symbol is far ahead of the actual place where the error occurred. The string "EOF" is used for the last symbol when the

parser has read past the query.

2501 The token '%1', on line %0, cannot follow a %2 command, therefore the command was not executed~

Indicates that the symbol following a RETRIEVE, APPEND, REPLACE, or DELETE command was not the start of a new command or the end of the query buffer. In general, this error catches misspelled keywords in the qualification, however, a legal command followed by a misspelled command name will not be run.

2700 line %0, non-terminated string~

You have omitted the required string terminator (") .

2701 line %0, string too long~

Somehow, you have had the persistence or misfortune to enter a character string constant longer than 255 characters.

2702 line %0, invalid operator~

You have entered a character which is not alphanumeric, but which is not a defined operator, for example, "?".

2703 line %0, Name too long '%1'~

In INGRES relation names and domain names are limited to 12 characters.

2704 line %0, Out of space in symbol table - Query too long~

2705 line %0, non-terminated comment~

2707 line %0, bad floating constant: %1~

Your floating constant is too large or too small. However, this error is not currently checked.

2808 line %0, control character passed in pre-converted string~

In EQUEL a control character became embedded in a string and was not caught until the scanner was processing it.

## OVQP ERRORS

4100 OVQP query list overflowed~

This error is produced in the unlikely event that the internal form of your interaction requires more space in the one variable query processor than has been allocated for a query buffer. There is not much you can do except shorten your interaction or recompile OVQP with a larger query buffer.

4101 numeric operation using char and numeric domains not allowed~

Occasionally, you will be notified by OVQP of such a type mismatch on arithmetic operations. This only happens if the parser has not recognized the problem.

4102 unary operators are not allowed on character values~

This error will be produced if you use an expression such as "+ "string""

4103 binary operators cannot accept combinations of char and numeric fields~

4104 cannot use aggregate operator "sum" on character domains~

4105 cannot use aggregate operator "avg" on character domains~

4106 the interpreters stack overflowed -- query too long~

4107 the buffer for ASCII and CONCAT commands overflowed~

4108 cannot use arithmetic operators on two character fields~

4109 cannot use numeric values with CONCAT operator~

4110 floating point exception occurred.~

If you have floating point hardware instead of the floating point software interpreter, you will get this error upon a floating point exception (underflow or overflow). Since the software interpreter ignores such exceptions, this error is only possible with floating point hardware.

4111 character value cannot be converted to numeric due to incorrect syntax.~

When using int1, int2, int4, float4, or float8 to convert a character to value to a numeric value, the character

value must have the proper syntax. This error will occure if the character value contained non-numeric characters.

4199 you must convert your 6.0 secondary index before running this query!~

The internal format of secondary indices was changed between versions 6.0 and 6.1 of INGRES. Before deciding to use a secondary index OVQP checks that it is not a 6.0 index. The solution is to destroy the secondary index and recreate it.

## DECOMP ERRORS

4602 query involves too many relations to create aggregate function intermediate result.~

In the processing of aggreagte functions it is usually necessary to create an intermediate relation for EACH aggregate function. Since each relation requires a separate file a situation can be created where decomp requires more open files than are available in your UNIX installation. You must either break the interaction apart and process the aggregate functions separately or you must recompile UNIX to support more open files per process.

4603 Query involves too many relations for available file descriptors~

Same general problem as 4602. The only solution is to reduce the number of variables in your interaction by breaking it apart or recompile UNIX.

4610 Query too long for available buffer space (qbufsize).~

This will happen if the internal form of the interaction processed by decomp is too long for the available buffer space. You must either shorten your interaction or recompile decomp. The name in parenthesis gives the internal name of which buffer was too small.

4612 Query too long for available buffer space (sqsize).~

4613 Query too long for available buffer space (stacksiz)~

4614 Query too long for available buffer space (agbuf).~

## DATA BASE UTILITIES ERRORS

5001 PRINT: bad relation name %0~

You are trying to print a relation which doesn't exist.

5102 CREATE: duplicate relation name %0~

You are trying to create a relation which already exists.

5103 CREATE: %0 is a system relation~

You cannot create a relation with the same name as a system relation. The system depends on the fact that the system relations are unique.

5104 CREATE %0: invalid attribute name %1~

This will happen if you try to create a relation with an attribute longer than 12 characters.

5105 CREATE %0: duplicate attribute name %1~

Attribute names in a relation must be unique. You are trying to create one with a duplicated name.

5106 CREATE %0: invalid attribute format "%2" on attribute %1~

The allowed formats for a domain are c1-c255, i1, i2, i4, f4 and f8. Any other format will generate this error.

5107 CREATE %0: excessive domain count on attribute %1~

A relation cannot have more than 49 domains. The origin of this magic number is obscure. This is very difficult to change.

5108 CREATE %0: excessive relation width on attribute %1~

The maximum number of bytes allowed in a tuple is 498. This results from the decision that a tuple must fit on one UNIX "page". Assorted pointers require the 14 bytes which separates 498 from 512. This "magic number" is very hard to change.

5201 DESTROY: %0 is a system relation~

The system would immediately stop working if you were allowed to do this.

5202 DESTROY: %0 does not exist or is not owned by you~

5300 INDEX: primary relation does not exist~

5301 INDEX: more than maximum number of domains~

A secondary index can be created on at most six domains.

5302 INDEX: invalid domain %0~

5303 INDEX: relation %0 not owned by you~

5304 INDEX: relation %0 is already an index~

5305 INDEX: relation %0 is a system relation~ Secondary indices cannot be created on system relations.

5401 HELP: relation %0 does not exist~

5402 HELP: cannot find manual section "%0"~

Either the desired manual section does not exist, or your system does not have any on-line documentation.

5500 MODIFY: relation %0 does not exist~

5501 MODIFY: you do not own relation %0~

5502 MODIFY %0: you may not provide keys on a heap~

5503 MODIFY %0: too many keys provided~

5504 MODIFY %0: cannot modify system relation~

5507 MODIFY %0: duplicate key "%1"~

5508 MODIFY %0: key width (%1) too large for isam~

When modifing a relation to isam, the sum of the width of the key fields cannot exceed 245 bytes.

5510 MODIFY %0: bad storage structure "%1"~

The valid storage structure names are heap, cheap, isam, cisam, hash, and chash.

5511 MODIFY %0: bad attribute name "%1"~

5512 MODIFY %0: "%1" not allowed or specified more than once~

5513 MODIFY %0: fillfactor value %1 out of bounds~

5514 MODIFY %0: minpages value %1 out of bounds~

5515 MODIFY %0: "%1" should be "fillfactor" or "minpages"~

5516 MODIFY %0: maxpages value %1 out of bounds~

5517 MODIFY %0: minpages value exceeds maxpages value~

5600 SAVE: cannot save system catalog "%0"~

System relations have no save date and are guaranteed to stay for the lifetime of the data base.

5601 SAVE: bad month "%0"~ This was a bad month for INGRES.

5602 SAVE: bad day "%0"~

5603 SAVE: bad year "%0"~

5604 SAVE: relation %0 does not exist or is not owned by you~

5800 COPY: relation %0 doesn't exist~

5801 COPY: attribute %0 in relation %1 doesn't exist or it has been listed twice~

5803 COPY: too many attributes~

Each dummy domain and real domain listed in the copy statement count as one attribute. The limit is 150 attributes.

5804 COPY: bad length for attribute %0. Length="%1"~

5805 COPY: can't open file %0~

On a copy "from", the file is not readable by the user.

5806 COPY: can't create file %0~

On a copy "into", the file is not creatable by the user. This is usually caused by the user not having write permission in the specified directory.

5807 COPY: unrecognizable dummy domain "%0"~

On a copy "into", a dummy domain name is used to insert certain characters into the unix file. The domain name given is not valid.

5808 COPY: domain %0 size too small for conversion. There were %2 tuples seccessfully copied from %3 into %4~

When doing any copy except character to character, copy checks that the field is large enough to hold the value being copied.

5809 COPY: bad input string for domain %0. Input was "%1". There were %2 tuples successfully copied from %3 into %4~

This occurs when converting character strings to integers or floating point numbers. The character string contains something other than numeric characters (0-9,+,-,blank,etc.).

5810 COPY: unexpected end of file while filling domain %0. There were %1 tuples successfully copied from %2 into %3~

5811 COPY: bad type for attribute %0. Type="%1"~

The only accepted types are i, f, c, and d.

5812 COPY: The relation "%0" has a secondary index. The index(es) must be destroyed before doing a copy "from"~

Copy cannot update secondary indices. Therefore, a copy "from" cannot be done on an indexed relation.

5813 COPY: You are not allowed to update the relation %0~

You cannot copy into a system relation or secondary index.

5814 COPY: You do not own the relation %0.~

You cannot use copy to update a relation which you do not own. A copy "into" is allowed but a copy "from" is not.

5815 COPY: An unterminated "c0" field occurred while filling domain %0. There were %1 tuples successfully copied from %2 into %3~

A string read on a copy "from" using the "co" option cannot be longer than 1024 characters.

5816 COPY: The full pathname must be specified for the file %0~

The file name for copy must start with a "/".

5817 COPY: The maximum width of the output file cannot exceed 1024 bytes per tuple~

The amount of data to be output to the file for each tuple exceeds 1024. This usually happens only if a format

was mistyped or a lot of large dummy domains were specified.