

GRASS Drivers

- [CELL driver](#): Driver to redirect GRASS Monitor contents to CELL raster file at any size within the current GRASS database
- [PNG driver](#): Driver to redirect GRASS Monitor contents to external PNG file
- [HTMLMAP graphics driver](#): Driver to redirect vector maps to sensitive HTML maps
- [XDRIVER](#): GRASS Monitor driver – graphical map output in Monitor window

Other Commands

[help home](#), [database](#), [display](#), [drivers](#), [general](#), [grid3d](#), [imagery](#), [import](#), [misc](#), [models](#), [paint](#), [photo](#), [postscript](#), [raster](#), [scripts](#), [sites](#), [vector](#)





NAME

CELL driver – driver to redirect graphical output into a raster map
(*drivers*)

DESCRIPTION

The CELL driver can be used to redirect the output of the GRASS monitor into a raster map. Features are:

- storing graphical output into a file with better resolution than through screen grabbing.
- saving 3–dimensional output from d.3d or other display programs to a *D_cell* raster map.
- saving display overlays (raster/raster/vector/...) to a file for high resolution export with r.out.tiff or similar

The user will find a *D_cell* raster file in GRASS database after starting the driver and using d.rast, d.vect etc. and finally stopping the driver again. The *D_cell* raster file is written during closing procedure (this may require some time according to the selected resolution).

Then the user can export this file through r.out.tiff etc. for use in external mapping software.

Be aware that *D_cell* map is in a xy–projection which may differ to the current location's projection. So the *D_cell* map cannot be directly displayed on a GRASS monitor.

Usage in details

- Set the map size for the CELL driver:


```
export GRASS_WIDTH=xxx (default is 640)
export GRASS_HEIGHT=yyy (default is 480)
```

 If "export" command is not accepted by your shell, use "setenv" instead:


```
setenv GRASS_WIDTH xxx
setenv GRASS_HEIGHT yyy
```
- Start the CELL driver:


```
d.mon start=CELL
```

 Wait for "Ready" message (may take some time due to the file size selected)


```
d.mon select=CELL
```
- Now the user can work with the common display modules:


```
d.zzzz (d.rast, d.vect, d.3d, ...)
```

 Note that an eventually started GRASS monitor won't display anything as the output is redirected to *D_cell* file.
- Stop the CELL driver (and write *D_cell* file):


```
d.mon stop=CELL
```

 Note that this may require some time.
- To export the *D_cell* file, re–set the region:

```
g.region raster=D_cell  
Use the r.out.* export module.
```

NOTES

Note that the *D_cell* raster map is in xy-projection with simple xy-coordinates.

The resolution of the *D_cell* raster map is defined by the map extents. Use *g.region -p* to get the number of rows and cols and use the environment variables to set the *D_cell* size.

Note that the CELL driver is limited to 8bit colors.

SEE ALSO

[d.mon](#), [g.region](#), [htmlmapdriver](#), [pngdriver](#), [XDRIVER](#), [tcltkgrass cell-tiff script](#)

AUTHOR

CERL

Last changed: \$Date: 2002/01/25 05:45:32 \$

GRASS Drivers



NAME

XDRIVER – *GRASS Monitor* – driver to display GRASS maps in X11 environment
(*drivers*)

DESCRIPTION

The GRASS XDRIVER can be used to display GRASS maps in X11 environment.

Usage in details

- Set the map size for the XDRIVER:


```
export GRASS_WIDTH=xxx (default is 640)
export GRASS_HEIGHT=yyy (default is 480)
If "export" command is not accepted by your shell, use "setenv" instead:
setenv GRASS_WIDTH xxx
setenv GRASS_HEIGHT yyy
```
- Further variables:


```
XDRIVER_LEFT
  defines the left position of GRASS monitor
XDRIVER_TOP
  define the upper position of GRASS monitor
```

To control the XDRIVER, use the [d.mon](#) module.

The maximum color depth depends on the current X Server settings.

SEE ALSO

[g.region](#), [d.mon](#), [htmlmapdriver](#), [celldriver](#), [PNGdriver](#)

AUTHOR

CERL
various improvements from several authors.
Rewritten 2001 by Glynn Clements

Last changed: \$Date: 2003/02/09 12:07:23 \$



NAME

d.rast.pg – Generate or display a reclass map based on the unique values in a database column.
(GRASS–RDBMS Interface Display Program)

SYNOPSIS

d.rast.pg

d.rast.pg help

d.rast.pg *key=name* *tab=name* *col=name* [*lab=name*] [*where=name*] **input=name** [**output=name**]

ALTERNATE

d.rast.pg -s help

d.rast.pg -s *sql=filename* **input=name** [**output=name**]

DESCRIPTION

d.rast.pg displays a raster image in which each category represents a unique value in a database column. Values are retrieved from the current database based on a user defined SQL SELECT statement. The current database is identified by the GRASS environment variable \$PG_DBASE which is set using the *g.select.pg* GRASS–RDBMS interface tool. The output from this program is a set of GRASS reclass rules which are used to create or display a reclassified raster surface based on the attributes returned from the database and the raster data layer specified as the input map.

COMMAND LINE OPTIONS

Parameters:

key=databasecolumnname

Column corresponding to category values in raster map [input]

tab=databasetablename

Table containing [col]

col=databasecolumnname

Column to base reclass on

where=clause

Where clause for query (ie. where col='paved')

lab=name

Column to use as labels (optional)

input=map

Name of existing raster file

output=map

Name of new reclass map

ALTERNATE COMMAND LINE USAGE

The alternate command line structure is selected using the the *[-s]* flag on the command line. When using this flag the user must include the name of a text file on the command line as well. This file must include a complete, well formed SQL SELECT statement.

Flag:

-s

SQL statement is input from a prepared file

Parameters:

sql=filename

SQL statement specifying well formed selection criteria.

input=map

Name of an existing raster map

output=map

Name of a new reclass file

EXAMPLE 1:

```
d.rast.pg input=kur_rast_id output=soils2 tab=info_kuruma key=rec_id col=soils_id lab=soils_info.name
where="soils_info.type_id = soils_id"
```

RESULT:

This example will create a raster map *soils2* with a category for each unique value taken from the field "soils_id" from table *info_kuruma*, with label of value of the column "name" from the table *soils_info*. The raster map will be a reclassified data layer based on the input raster map *kur_rast_id*.

EXAMPLE 2:

```
d.rast.pg -s sql=raster.sql input=kur_rast_id output=soils3
```

raster.sql:

```
select rec_id, soils_info.type_id from info_kuruma where soils_info.name ~ 'barren' and soils_info.type_id =
soils_id
```

RESULT:

This example will create a raster map *soils3* with a category for each unique value taken from the column "type_id" from the table *soils_info*, with only 'barren' type. The raster map will be a reclassified data layer based on the input raster map *kur_rast_id*. Note that there must be two fields in the SELECT statement (these for the old and new values).

In this example, the input map *kur_rast_id* was prepared by **v.to.rast** from the vector are map *kuruma_id*, which is approximately 7,000 polygons with unique ids (corresponding to *rec_id* field in the table *info_kuruma*).

BUGS

None known.

NOTE

This program requires the Postgres database software. It uses the GRASS module *r.reclass* started from inside it.

SEE ALSO

[g.column.pg](#), [g.select.pg](#), [g.stats.pg](#), [g.table.pg](#), [d.site.pg](#), [d.vect.pg](#), [d.what.r.pg](#), [d.what.s.pg](#), [d.what.v.pg](#), [r.reclass.pg](#), [r.rescale.pg](#), [v.reclass.pg](#)

AUTHOR

Original Informix SQL–tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)



NAME

d.site.pg – Display select sites from a database query. (GRASS–RDBMS Interface Display Program)

SYNOPSIS

d.site.pg

d.site.pg help

d.site.pg [**sql**=file] **tab**=name **coordx**=name **coordy**=name [**cats**=name] [**where**=name] [**map**=name]
 [**plot**=color,icon,size]

ALTERNATE

d.vect.pg -s help

d.vect.pg -s sql=filename [**map**=name] [**plot**=color,icon,size]

DESCRIPTION

Query sites in a PostgreSQL database using an SQL file or by specifying conditions directly. Optionally, plot display or send output to a site_lists "map".

d.site.pg displays select point location information returned from a database query. The site locations returned from the database query are displayed as icons in the active frame. The user controls the color, size and icon used in the graphic output. As option, a site list of the database output is generated and placed in the current mapset/location.

COMMAND LINE OPTIONS

Parameters:

tab=*database tablename*

Table containing X, Y coordinate values.

coordx=*database_x_columnname*

Column containing x coordinate values.

coordy=*database_y_columnname*

Column containing y coordinate values.

where=*SQL whereclause*

SQL "where" clause which specifies the query criteria to be used in subsetting the database. The information specified in the where option must indicate the column(s) to be used, the operators to be used in the evaluation and the values which the data in the column will be evaluated against.

GRASS Drivers

For example, if you want to select only those records from the table well where the value for depth is either 58 or 75 the following could be entered:

well.depth = 58 or well.depth = 75

To select all wells in an area where the value for well is between 50 and 120:

well.depth > 50 and well.depth < 120

To select all wells of depth greater than 75 where the value for owner is not equal to SMITH:

well.depth > 75 and well.owner != 'SMITH'

If the database column used as the selection criteria is a character field then the associated value must be placed in quotes. To determine the data types associated with columns in the currently selected database use the *g.column.pg* command with the *-v* flag.

In addition to the operators presented in the examples above a range of relational operators including and, or, etc. are supported.

Queries more complex are best implemented using a prepared SQL file.

map=list

Name of sites list to output

cats=name

Name of the category field to be assigned in the new site list

plot=color,icon,size

Colors: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, black

Icon: diamond, box, plus, x

Size: 1-9

Default: gray, x, 3

sql=filename

SQL statements specifying well formed selection criteria

ALTERNATE COMMAND LINE USAGE

The alternate command line usage is provided to simplify the process of retrieving information from more than one table in the query criteria. The alternate command line structure is selected using the

the *[-s]* flag on the command line. When using this flag the user must include the name of a text file on the command line as well. This file must include a complete, well formed SQL SELECT statement specifying the query criteria to retrieve values which correspond to category values in a GRASS data layer.

Flag:

-s

SQL select statements are input from a prepared file

Parameters:

map=list

Name of sites list to output

plot=color,icon,size

Colors: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, black

Icon: diamond, box, plus, x

Size: 1-9

Default: gray, x, 3

sql=filename

SQL statements specifying well formed selection criteria

EXAMPLE:

1. *d.site.pg tab=izb coordx=xcoor coordy=ycoor where="how_big~'small'" plot=white,diamond,9*

Result: displays only 'small' huts in white color diamonds size 9

2. *d.site.pg -s sql=izb3.sql map=huts_good*

izb3.sql is:

```
select xcoor,ycoor,owner from izb where how_good ~ 'good' and izb_info.name=name and izb_info.rec_id=num
```

Result: displays only 'good' huts (and only huts which have 'name' coinciding with that taken from a table *izb_info*, linked on the hut number) and creates site map called *huts_good* with categories taken from the database field 'owner'

BUGS

None known.

NOTE

This program requires the Postgres database software.

SEE ALSO

[g.column.pg](#), [g.select.pg](#), [g.stats.pg](#), [g.table.pg](#), [d.rast.pg](#), [d.vect.pg](#), [d.what.r.pg](#), [d.what.s.pg](#), [d.what.v.pg](#),
[r.reclass.pg](#), [r.rescale.pg](#), [v.reclass.pg](#)

AUTHOR

Original Informix SQL-tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)



NAME

d.vect.pg – Display select vectors from an existing vector map. (GRASS–RDBMS Interface Display Program)

SYNOPSIS

d.vect.pg

d.vect.pg help

d.vect.pg [-f] **key=***name* **tab=***name* [**where=***name*] **map=***name* [**color=***name*]

ALTERNATE

d.vect.pg -s help

d.vect.pg -s sql=*filename* **map=***name* [**color=***name*]

DESCRIPTION

d.vect.pg displays select vectors from an existing vector map based on the unique values in a database column. Each row returned by a user constructed database query will be associated with a vector feature which is subsequently drawn on the graphics display in the active frame. The user can control the color of the vector draw by specifying a color on the command line.

COMMAND LINE OPTIONS

Flag:

-f

Fill polygons selected on the query criteria

Parameters:

key=*databasecolumnname*

Column in table "tab" of the currently selected database containing values corresponding to the vector maps category values. Table is designated on the command line by **tab=***tablename* and vector is designated on the command line by **map=***mapname*.

tab=*datatablename*

Table in the currently selected database containing a column which has values corresponding to vector category values in the map designated by **map=***map*.

where=*SQLwhereclause*

SQL "where" clause which specifies the query criteria to be used in subsetting the database. The field

names specified in the where option must indicate the column(s) to be used, the operators to be used in the evaluation and the values which the data in the column will be evaluated against.

If the database column used as the selection criteria is a character field then the associated value must be placed in quotes. To determine the data types associated with columns in the currently selected database use the *g.column.pg* command with the *-v* flag.

Queries which are more complex are best implemented using the *-s* flag and a prepared SQL file.

map=map

Name of an existing vector map with category values which correspond to values in a specified column in the currently selected database.

color=name

Color to draw vectors in

Colors: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, black

ALTERNATE COMMAND LINE USAGE

The alternate command line usage is provided to simplify the process of retrieving information from more than one table in the query criteria. The alternate command line structure is selected using the the *[-s]* flag on the command line. When using this flag the user must include the name of a text file on the command line as well. This file must include a complete, well formed SQL SELECT statement specifying the query criteria to retrieve values which correspond to category values in a GRASS data layer.

Flag:

-s

SQL select statements are input from a prepared file

-f

Fill polygons selected on the query criteria

Parameters:

sql=filename

SQL statements specifying well formed selection criteria.

map=name

Name of an existing vector map

color=name

Color to draw vectors in

Colors: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, black

EXAMPLE:

1. *d.vect.pg -f key=rec_id map=kuruma_id tab=info_kuruma where='type_id >32 and type_id < 38'*
color=red

Result: only polygons with type in 33–37 range would be displayed in red color..

2. *d.vect.pg -f -s sql=oak.sql map=kuruma_id*

oak.sql is:

select rec_id from info_kuruma where type_id > 32 and type_id <38;

Result: this would be the same as in the first example.

BUGS

none

NOTE

This program requires the Postgres database software.

SEE ALSO

[g.column.pg](#), [g.select.pg](#), [g.stats.pg](#), [g.table.pg](#), [d.rast.pg](#), [d.site.pg](#), [d.what.r.pg](#), [d.what.s.pg](#), [d.what.v.pg](#), [r.reclass.pg](#), [r.rescale.pg](#), [v.reclass.pg](#)

AUTHOR

Original Informix SQL-tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)



NAME

d.what.r.pg – Report database attributes associated with a raster category value at a specified location. (GRASS–RDBMS Attribute Interface Program)

SYNOPSIS

d.what.r.pg
d.what.r.pg help
d.what.r.pg map=*name* tab=*name* col=*name* [hv=*name*]

ALTERNATE

d.what.r.pg –s help
d.what.r.pg –s sql=*filename* map=*name*

DESCRIPTION

d.what.r.pg reports database attributes from the currently selected database associated with a raster category at a specific location on a raster map. The currently selected database is identified by the GRASS environment variable \$PG_DBASE which is set using the *g.select.pg* GRASS–RDBMS interface tool. If this environment variable is not set the program terminates with a message to the user. The current raster location is selected by the user with the mouse. If the alternate form of this command is selected with the *–s* flag the user has greater control over the manner in which the attribute information is displayed. Using the *–s* option the attributes from more than one table can be returned and displayed.

COMMAND LINE OPTIONS

Parameters:

map=*map*

Name of an existing raster map with category values linked to the currently selected database.

tab=*database tablename*

Table in the currently selected database containing a column associated with raster category values.

col=*database column name*

Column associated with raster category values.

hv

Type of database output – [h/v]

default: h

ALTERNATE COMMAND LINE USAGE

The alternate command line usage is provided to simplify the process of retrieving information from more than one table in the query criteria. The alternate command line structure is selected using the the `[-s]` flag on the command line. When using this flag the user must include the name of a text file on the command line as well. This file must include a complete, well formed SQL SELECT statement specifying the query criteria needed to retrieve values which correspond to category values in a GRASS data layer.

The user must also specify the name of the GRASS data layer containing these category values on the command line. The following examples illustrate the syntax which must be used in constructing a SELECT statement for use with the `-s` flag.

EXAMPLE:

1. *d.what.r.pg -s sql=raster.sql map=blag_forest_kur*

and raster.sql is:

```
select rec_id, type_id from info_kuruma where type_id = ?
```

Result:

Reports two fields from the table `info_kuruma` associated with the raster category value at the current mouse location the raster map.

2. *d.what.r.pg -s sql=raster2.sql map=blag_forest_kur*

and raster2.sql is:

```
select rec_id, type_id, census_info.type_good from info_kuruma where type_id = ? and rec_id = census_info.rec_id
```

Result:

Reports fields from two tables `info_kuruma` and `census_info` associated with the raster category value at the current mouse location the raster map.

3. *d.what.r.pg -s kur_rast_id sql=process.sql*

process.sql is:

```
update census_info set type_good = '1' where rec_id = ? and info_kuruma.type_id = 35 and info_kuruma.rec_id = rec_id;
```

Result: this would only change field of the `census_info` table for the clicked polygon or line, if the corresponding type taken from `info_kuruma` table is equal to 35.

Flag:

-s
SQL select statements are input from a prepared file.

Parameters:

sql=filename

SQL statements specifying well formed selection criteria.

map=map

Name of an existing raster map with category values linked to a database.

hv

Type of database output – [h/v]

default: h

BUGS

None known.

NOTE

This program requires the Postgres database software.

SEE ALSO

[g.column.pg](#), [g.select.pg](#), [g.stats.pg](#), [g.table.pg](#), [d.rast.pg](#), [d.site.pg](#), [d.vect.pg](#), [d.what.s.pg](#), [d.what.v.pg](#), [r.reclass.pg](#), [r.rescale.pg](#), [v.reclass.pg](#)

AUTHOR

Original Informix SQL–tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)



NAME

d.what.s.pg – Report database attributes associated with a specific point location. (GRASS–RDBMS Attribute Interface Program)

SYNOPSIS

d.what.s.pg

d.what.s.pg help

d.what.s.pg **tab**=name **xcol**=name **ycol**=name [**xpos**=value] [**ypos**=value] **distance**=value [**hv**=name]

ALTERNATE

d.what.s.pg **-s** help

d.what.s.pg [**-s**] [**sql**=file] **distance**=value [**hv**=name]

DESCRIPTION

d.what.s.pg reports database attributes in the currently selected database which are associated with a specific point location. The current database is identified by the GRASS environment variable \$PG_DBASE which is set using the *g.select.pg* GRASS–RDBMS interface tool. The location to query is selected by the user with the mouse. The radius around this point to search is specified by the distance argument. Sites are selected by comparing the values in the database columns containing X and Y coordinate data against the coordinate data at the current mouse location. The columns in the table containing the coordinate values are returned in the report. If information from more than one table is required use the **-s** flag and the alternate command line format (see below). The user can also provide the center of the search point and the distance around it to find sites within the circle around the search center non–interactively.

COMMAND LINE OPTIONS

Parameters:

tab=*database tablename*

Table containing X, Y coordinate values

xcol=*database column name*

Column containing the X coordinate (E/W) value.

ycol=*database column name*

Column containing the Y coordinate (N/S) value.

distance=*value*

Radius distance from the current mouse location to conduct database search.

xpos=value

X coord. (E/W) of search.

ypos=value

Y coord. (N/S) of search.

hv=name

(h)Horizontal/(v)vertical output (default is v)

ALTERNATE COMMAND LINE USAGE

The alternate command line structure is selected using the the `[-s]` flag on the command line. When using this flag the user must include the name of a text file on the command line as well. This file must include a complete, well formed SQL SELECT statement specifying the query criteria.

The user must also specify the name of the GRASS data layer containing these category values on the command line. The following example illustrates the syntax which must be used in constructing a SELECT statement for use with the `-s` flag.

EXAMPLE 1. (single table)

```
d.what.s.pg -s sql=izb.sql distance=100
```

here *izb.sql*:

```
SELECT * from izb where point(xcoor,ycoor)
```

EXAMPLE 2. (two tables)

```
d.what.s.pg -s sql=izb2.sql distance=100
```

here *izb2.sql*:

```
SELECT *, huts_info.description from izb where huts_info.num=num and point(xcoor,ycoor)
```

Flag:

`-s`

SQL select statements are input from a prepared file.

Parameters:***sql=filename***

SQL statements specifying well formed selection criteria. These criteria must include the database table name, the Xcol and Ycol names as well as the search distance. Additional criteria may be placed in the "where" clause to further subset information returned from the database.

distance=value

Radius distance from the current mouse location to conduct database search.

hv=name

(h)Horizontal/(v)vertical output (default is v)

BUGS

None known.

NOTE

This program requires the Postgres database software.

SEE ALSO

[g.column.pg](#), [g.select.pg](#), [g.stats.pg](#), [g.table.pg](#), [d.rast.pg](#), [d.site.pg](#), [d.vect.pg](#), [d.what.r.pg](#), [d.what.v.pg](#), [r.reclass.pg](#), [r.rescale.pg](#), [v.reclass.pg](#)

AUTHOR

Original Informix SQL-tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)



NAME

d.what.v.pg – Report database attributes associated with vector features.

SYNOPSIS:

d.what.v.pg [-f] **map**=name **tab**=name **col**=name [**color**=name]
 [**fillcolor**=name] [**hv**=name]

d.what.v.pg [-sf] **sql**=file **map**=name [**color**=name] [**fillcolor**=name]
 [**hv**=name]

DESCRIPTION

Flags:

- s Use [-s] for SQL command file input.
- f Fill polygons?

Parameters:

sql

Name of the SQL command file

map

Vector map to run query on

tab

Postgres table with categories

col

Column with categories from this table

color

Selected lines color
 default: yellow

fillcolor

Selected areas color (for fill)
 options: red,orange,yellow,green,blue,indigo,white,black,brown, magenta,aqua,gray,grey

default: gray

hv

Type of database output – [h/v]

default: v

tab is Postgres table containing column linked to vector attribute values, **col** – column associated with vector attribute values, **map** is name of an existing vector map with attribute values linked to the currently selected database.

Reports database attributes from the currently selected database which are associated with specific vector features identified using the mouse. The currently selected database is identified by the GRASS environment variable \$PG_DBASE which is set using the *g.select.pg* GRASS–RDBMS interface tool. If this environment variable is not set the program terminates with a message to the user. If the alternate form of this command is selected with the *–s* flag the user has greater control over the manner in which the attribute information is displayed.

d.what.v.pg is used to query lines/areas with mouse. Like *d.what.vect*, it lists attributes and optionally fills queried areas (like *v.area*). It may be used for the UPDATE command in Postgres through the input SQL file (see Example 2). The idea is to prepare an arbitrary algorithm, code it in SQL, and then update DB with clicking on chosen polys/lines. You may consider this as piping through various filters anything you select on screen.

The database information can be optionally output as comma-separated list (horizontal) which is convenient for feeding it to other programs like table editors.

EXAMPLE:

1. *d.what.v.pg –s sql=census.sql map=census*

and census.sql is:

```
select slope, type_id from info_kuruma where rec_id = ?
```

Result: only two fields information is displayed on screen when the mouse is clicked upon the vector object.

2. *d.what.v.pg –s sql=process.sql map=census*

process.sql is:

```
update census_info set type_good = '1' where rec_id = ? and info_kuruma.type_id = 35 and info_kuruma.rec_id = rec_id;
```

Result: this would only change field of the *census_info* table for the clicked polygon or line, if the corresponding type taken from *info_kuruma* table is equal to 35 (the last limitation is a filter for any polygon/line picked from the map with mouse) .

Flags: SQL select statements are input from a prepared file.

Parameters: SQL statements specifying well formed selection criteria. Name of an existing vector map.

BUGS

1. Tcl-Tk modules require input "where" clause restricted to one rule typed without "whitespaces", (unlike terminal input where number of subclause is not limited).

This program requires the Postgres database software.

SEE ALSO

[g.column.pg](#), [g.select.pg](#), [g.stats.pg](#), [g.table.pg](#), [d.rast.pg](#), [d.site.pg](#), [d.vect.pg](#), [d.what.r.pg](#), [d.what.s.pg](#), [r.reclass.pg](#), [r.rescale.pg](#), [v.reclass.pg](#)

AUTHOR

Original Informix SQL-tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)



NAME

g.column.pg – Generate a list of database columns for a specified table in the currently selected database. (GRASS–RDBMS General Interface Program)

SYNOPSIS

g.column.pg
g.column.pg help
g.column.pg [-v] table=name

DESCRIPTION

g.column.pg is used to generate a list of the columns in the specified table [table] of the currently selected database. If the [-v] option is used the data type and length of the column are reported as well. The currently selected database is identified by the GRASS environment variable \$PG_DBASE which is set using the *g.select.pg* GRASS–RDBMS interface tool. If this environment variable is not set the program terminates with a message to the user. To generate a list of the available tables in the currently selected database use the command *g.table.pg*. The information provided by *g.column.pg* will be used in subsequent GRASS–RDBMS applications to formulate and construct well formed database queries.

COMMAND LINE OPTIONS

Flag:

-v
 Use v for a verbose listing of columns

Parameter:

table=databasetablename
 Name of table in the currently selected database.

EXAMPLE OUTPUT (terse)

g.column.pg table=nri

The following columns are defined in table: utm

fips	flood	hydunit
landc182	landc187	mlra
nriptr	other	point

GRASS Drivers

psu	refname	slope
soil5	textmod	texture
tfact	ukfact	utme
utm	xfact	

EXAMPLE OUTPUT (verbose)

g.column.pg table=nri -v

The following columns are defined in table: utm

flood	char16
hydunit	char16
landcl82	char
landcl87	char
mlra	char16
nriptr	int4
other	text

BUGS

None known.

NOTE

This program requires the Postgres database software.

SEE ALSO

[*g.select.pg*](#), [*g.stats.pg*](#), [*g.table.pg*](#), [*d.rast.pg*](#), [*d.site.pg*](#), [*d.vect.pg*](#), [*d.what.r.pg*](#), [*d.what.s.pg*](#), [*d.what.v.pg*](#), [*r.reclass.pg*](#), [*r.rescale.pg*](#), [*v.reclass.pg*](#)

AUTHOR

Original Informix SQL-tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)



NAME

g.select.pg – Select a Postgres database to be used in subsequent GRASS–RDBMS applications. (GRASS–RDBMS General Interface Program)

SYNOPSIS

g.select.pg
g.select.pg help
g.select.pg database=name

DESCRIPTION

g.select.pg is used to identify a Postgres database for subsequent GRASS–RDBMS applications. If the database specified on the command line can be found *g.select.pg* sets the GRASS environment variable PG_DBASE to this name. If the database is not found a list of the database directories is provided to the user. This program assumes that database queries have been granted to the user.

COMMAND LINE OPTIONS

Parameter:

database=databasename
Name of existing Postgres database

BUGS

None known.

NOTE

This program requires the Postgres database software.

SEE ALSO

[*g.column.pg*](#), [*g.stats.pg*](#), [*g.table.pg*](#), [*d.rast.pg*](#), [*d.site.pg*](#), [*d.vect.pg*](#), [*d.what.r.pg*](#), [*d.what.s.pg*](#), [*d.what.v.pg*](#), [*r.reclass.pg*](#), [*r.rescale.pg*](#), [*v.reclass.pg*](#)

AUTHOR

Original Informix SQL-tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)



NAME

g.stats.pg – Generate a range of simple statistics for the values in a database column. (GRASS–RDBMS General Interface Program)

SYNOPSIS

```
g.stats.pg
g.stats.pg help
g.stats.pg [-vf] table=name column=name [where=clause]
```

DESCRIPTION

g.stats.pg generates a series of simple statistics for a numeric column in the specified table of the currently selected database. The currently selected database is identified by the GRASS environment variable `$PG_DATABASE` which is set using the command *g.select.pg*. If this environment variable is not set the program terminates with a message to the user. *g.stats.pg* generates statistics for any numeric column in the table specified by the user. To identify the data types for individual columns in a table in the currently selected database use the command *g.column.pg* with the `-v` flag. Statistics generated for the column include count, sum, average, minimum value and maximum value.

COMMAND LINE OPTIONS

Parameters:

```
table=databasetablename
    Name of table in currently selected database.
column=databasecolumnname
    Column in [table] which is numeric in type.

where=clause
    Clause to select only some records from table
```

Flags:

```
-v
    Verbose output

-f
    Use frequencies instead of min, max, mean
```

EXAMPLE

11/06/2003

g.stats.pg table=utm column=ukfact

Statistics for column: ukfact

```
count
171
sum
47.67
avg
0.28
max
0.49
min
0.00
```

BUGS

None known.

NOTE

This program requires the Postgres database software.

SEE ALSO

[*g.column.pg*](#), [*g.select.pg*](#), [*g.table.pg*](#), [*d.rast.pg*](#), [*d.site.pg*](#), [*d.vect.pg*](#), [*d.what.r.pg*](#), [*d.what.s.pg*](#), [*d.what.v.pg*](#),
[*r.reclass.pg*](#), [*r.rescale.pg*](#), [*v.reclass.pg*](#)

AUTHOR

Original Informix SQL–tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)



NAME

g.table.pg – Generate a list of database tables in the currently selected database. (GRASS–RDBMS General Interface Program)

SYNOPSIS

g.table.pg
g.table.pg help

DESCRIPTION

g.table.pg is used to generate a list of the tables in the currently selected SQL database. The currently selected database is identified by the GRASS environment variable `$PG_DBASE` which is set using the *g.select.pg* GRASS–RDBMS interface tool. If this environment variable is not set the program terminates with a message to the user. Otherwise, the names of the tables in the currently selected database are displayed on the screen in three columns. The names of the tables in the current database are needed to construct query criteria for use in many of the other tools. This command can be run at any time and is provided for reference purposes, to be used while performing GRASS–RDBMS applications.

COMMAND LINE OPTIONS

None

EXAMPLE

g.table.pg

The following tables are available in database: nri

county	cover	geoaggs
hydros	landclass	locks
mlra	nrifldvals	point
practices	priors	psu
psutrends	recorders	state
stdrpts	streams	treatments
trends82	trends87	udrpt
udsubj	utm	valuelist
water	windbreaks	

BUGS

None known.

NOTE

This program requires the Postgres database software.

SEE ALSO

[*g.column.pg*](#), [*g.select.pg*](#), [*g.stats.pg*](#), [*d.rast.pg*](#), [*d.site.pg*](#), [*d.vect.pg*](#), [*d.what.r.pg*](#), [*d.what.s.pg*](#), [*d.what.v.pg*](#), [*r.reclass.pg*](#), [*r.rescale.pg*](#), [*v.reclass.pg*](#)

AUTHOR

Original Informix SQL-tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)



NAME

r.reclass.pg – Generate a new raster reclass map based on the results of multiple queries to the currently selected database. (GRASS–RDBMS Raster Interface Program)

SYNOPSIS

r.reclass.pg

r.reclass.pg help

r.reclass.pg *sql=file input=name key=name output=name [label=name]*

DESCRIPTION

r.reclass.pg generates reclass rules for a new raster map layer based on the results of multiple queries to the currently selected database. The user constructs a series of mutually exclusive SQL retrieve statements designed to return groups of records from the database. Each group of records should be internally consistent in terms of attribute characteristics specified by the user in the retrieve clause. These groups should also be mutually exclusive, thereby insuring that a row returned by one retrieve clause is not also returned by a subsequent retrieve clause. Each group of records therefore forms the basis for a single category in the resulting GRASS raster reclass map. *r.reclass.pg* processes each retrieve statement in order generating the GRASS reclass rules needed to create a new raster map. As each retrieve statement is processed the group of records returned receives a common category value. The category value is incremented by one for each subsequent retrieve statement which is processed. The resulting reclass map will have one category for each of the original retrieve statements and an additional category for no data areas. No data in this case includes actual no data areas and areas for which no rows were returned by the database queries.

COMMAND LINE OPTIONS

Parameters:

sql=filename

Name of file containing SQL query statements.

input=map

Name of an existing raster map layer.

key=databasecolumnname

Name of the database column linked to GRASS via the categories in the input map layer.

output=map

Name of new raster (reclass), file.

label=name

Label for new categories.

EXAMPLE:

r.reclass.pg sql=dbrr.sql key=grasscat input=seco.soils output=tfact.recl label="Tfact="

dbrr.sql:

retrieve (secosoilcats.grasscat)

where layer.tfact >0 and layer.tfact <=2 and layer.muid=secosoilcats.muid

retrieve (secosoilcats.grasscat)

where layer.tfact >0 and layer.tfact <=2 and layer.muid=secosoilcats.muid

retrieve (secosoilcats.grasscat)

where layer.tfact = 5 and layer.muid=secosoilcats.muid

BUGS

None known.

NOTE

This program requires the Postgres database software.

SEE ALSO

[*g.column.pg*](#), [*g.select.pg*](#), [*g.stats.pg*](#), [*g.table.pg*](#), [*d.rast.pg*](#), [*d.site.pg*](#), [*d.vect.pg*](#), [*d.what.r.pg*](#), [*d.what.s.pg*](#), [*d.what.v.pg*](#), [*r.rescale.pg*](#), [*v.reclass.pg*](#)

AUTHOR

Original Informix SQL–tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)



NAME

r.rescale.pg – Generate a raster map layer in which the categories represent values in a database column which have been divided into equal interval units. (GRASS–RDBMS Raster Interface Program)

SYNOPSIS

r.rescale.pg

r.rescale.pg help

r.rescale.pg *tab=name* **key=name** **col=name** **cats=name** **input=name** **output=name** [**join=tab,tabkey,pkey**]

DESCRIPTION

r.rescale.pg creates a reclassified raster map layer by dividing the values in a numeric column in the currently selected database into equal interval units. The number of resulting categories is determined by the user via the command line parameter [*cats=*]. *r.rescale.pg* evaluates the range of values for the database column and subsets these values into equal interval groups of records returned by the query. For example, if the database column contains values which range from 1–1000 and the [*cats*] value is equal to 10 the resulting raster map layer will contain the 10 categories: 1=1–100, 2=101–200 etc. In other words, each category in the new raster map layer will represent a range of 100 values from the database column used in the rescale operation. The database column being evaluated must be numeric in type. To identify the data types of columns in a database table use the *g.column.pg* command with the [*-v*] flag. *r.rescale.pg* does not take outlying data values into account. Therefore, if the range of values for a database column contains a limited number of extreme values the resulting rescale operation will be skewed in the direction of these values.

COMMAND LINE OPTIONS

Parameters:

tab=tablename

Table containing a column linked to category values in an existing raster map.

key=databasecolumnname

Column corresponding to category values in an existing raster map.

col=databasecolumnname

Column to base rescale operation on which is numeric in type.

cats=value

Number of categories to define in the resulting reclass map.

input=map

Name of an existing raster file with category values linked to a column in the currently selected database.

output=map

Name of new raster map

join=tab,tabkey,pkey

Tab is the table used to develop the current postQUEL query. Tabkey is the database column used to relate information in this table with data in the table linked to the GRASS category file. Pkey is the associated column in the table linked to the GRASS category file which is related to tabkey in the current table.

For instance, assume that stf1main is a table containing column values associated with category values in a the GRASS raster file blkgrp.ids. In addition, assume that stf1main is a table containing attribute data on age in the column pop100. In this example stf1main is the table associated with the GRASS raster map and tractblck is the column linking stf1main to the GRASS category file. The column pop100 in stf1main will be the basis for the rescale effort. To specify the rescale:

```
r.rescale.pg tab=stf1main key=tractblck col=pop100 cats=5 input=blkgrp.ids output=pop100.rescale
```

Specifying these conditions would insure that all rows from table stf1main which satisfy the query criteria would be related to the spatial features in the GRASS data layer via the GRASS category values.

BUGS

None known.

NOTE

This program requires the Postgres database software.

SEE ALSO

[*g.column.pg*](#), [*g.select.pg*](#), [*g.stats.pg*](#), [*g.table.pg*](#), [*d.rast.pg*](#), [*d.site.pg*](#), [*d.vect.pg*](#), [*d.what.r.pg*](#), [*d.what.s.pg*](#), [*d.what.v.pg*](#), [*r.reclass.pg*](#), [*v.reclass.pg*](#)

AUTHOR

Original Informix SQL–tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)



NAME

r.to.pg – exports GRASS raster map to PostgreSQL database into a three column table.
(*GRASS Raster Script*)

SYNOPSIS

r.to.pg
r.to.pg help
r.to.pg rastermap [table_name]

DESCRIPTION

The *r.to.pg* program exports GRASS raster map to PostgreSQL database into a three column table. A connection to PostgreSQL database is needed. The new table will be created in the currently connected database, if the table name does not exist – please check first.

Parameters:

rastermap
 Name of an existing raster map to be exported to PostgreSQL.

table_name
 Name of new table to create in currently connected PostgreSQL database (*Default is rastermap name*)

EXAMPLE

A PostgreSQL table with three columns will be created:

ID	column_name	label
1	45.2525	text
2	45.5	label text
3	44.124	text
4	46	text
...		

with:

id:	Cell ID	(int8)
column_name:	Value of rastermap cells	(float)
label:	Category label	(text)

KNOWN BUGS

The number check of imported values sometimes does not work properly because of some shell problems with 'asterix'(*).

SEE ALSO

[*db.connect*](#)

AUTHOR

Markus Neteler
category and ID support by Otto Dassau

Last changed: \$Date: 2002/07/27



NAME

v.reclass.pg – Generate new vector map layer derived from attribute data in the currently selected database. (GRASS–RDBMS Vector Interface Program)

SYNOPSIS

v.reclass.pg

v.reclass.pg help

v.reclass.pg [-d] **input**=name **key**=name **tab**=name **col**=name [**where**=name] [**output**=name] **type**=name

v.reclass.pg [-sd] **sql**=file **input**=name [**output**=name] **type**=name

DESCRIPTION

v.reclass.pg reclasses vector maps according to the SELECT query. The input map may be of vector area, line or vector site type, resulting in the reclass map of the same type.

The sql file if chosen to input contains a single line like: 'select key_col, reclass_col from info_tab [where clause]'.

The users needs to either input two names of the columns interactively or feed them from the input sql file.

The first name is the old map key column, the second one is the key for the new map. Additionally, a clause to choose only some vectors to be reclassified from the input map can be imposed by user, either interactively or from the input file. Both columns used for creating the reclass rules must be of numeric (integer) type.

If the user omitted the output map name, the reclass map would only be displayed on monitor and not created.

COMMAND LINE OPTIONS

Flag:

-d

Dissolve common boundaries between reclassified areas

Parameters:

sql=filename

Name of file containing SQL query statements

key=databasecolumnname

Key column in db

tab=name

Table containing [col]

col=name

Column to base reclass on

where=name

Where clause for query (ie. where col='paved')

type=area/line/site

Select area, line or site as type of the input/output vector map

input=map

Name of existing vector file to be reclassified using query output.

output=map

Name of new raster (reclass) file

EXAMPLE:

1. Reclass to vector map of quartiles from forest stands map (*kuruma_id*).

```
v.reclass.pg -s -d sql=reclass.sql input=kuruma_id output=kuruma_quart type=area
```

and reclass.sql is:

```
select rec_id, quartnum from info_kuruma
```

2. Reclass to vector map of forest types (*kuruma_oak*) from map of forest plots (*kuruma_id*) taking only oak (types 32–37).

```
v.reclass.pg -d kuruma_id key=rec_id col=type_id tab=info_kuruma where='type_id > 31 and type_id < 38'  
output=kuruma_oak type=area
```

BUGS

None known.

NOTE

This program requires the [PostgreSQL](#) database software. It uses other GRASS modules *v.reclass* and *d.vect* launched from inside.

SEE ALSO

[g.column.pg](#), [g.select.pg](#), [g.stats.pg](#), [g.table.pg](#), [d.rast.pg](#), [d.site.pg](#), [d.vect.pg](#), [d.what.r.pg](#), [d.what.s.pg](#), [d.what.v.pg](#), [r.reclass.pg](#), [r.rescale.pg](#)

AUTHOR

Original Informix SQL–tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)



NAME

v.to.pg – Export areas and lines from an existing vector map to Postgres/PostGIS table. (GRASS–RDBMS Interface Display Program)

SYNOPSIS

v.to.pg

v.to.pg help

v.to.pg [*-fvtp*] **key**=*name* **tab**=*name* **type**=*name* [**where**=*name*] **map**=*name* [**color**=*name*]

ALTERNATE

v.to.pg -s help

v.to.pg [*-sfvp*] **sql**=*filename* **map**=*name* **type**=*name* [**color**=*name*]

DESCRIPTION

v.to.pg exports vectors from an existing vector map based on the unique values in a database column (or optionally all vectors without referencing to any existing table, see *-t* option). Each row returned by a user constructed database query will be associated with a vector feature which may be drawn on the graphics display if X–windows are there. The user can control the color of the vector draw by specifying a color on the command line.

As result, a new Postgres table *table_bnd* or *table_arc* (or *table_mpoly* and *table_mstring* for PostGIS) is created to hold areas as internal type "polygon" and lines as "open path", where *table* is the **tab** parameter. Besides these elements in fields called *boundary* or *segment*, the table would also have the category field (named after **key** parameter), case number this category occurred in map (as field called 'num'), and an extra boolean field 'ex' for 'true' if the polygon is external and 'f' if it is a hole for *table_bnd* only.

For PostGIS, the difference is that types of the imported entities are POLYGON (i.e., with "holes" all in one POLYGON) and LINESTRING, respectively. The fields are called *grass_poly* and *grass_line*. There is no need to define whether a polygon is internal ("hole") or external, in this case, therefore there is no field 'ex'.

COMMAND LINE OPTIONS

Flag:

-f

Fill polygons selected on the query criteria.

-t

- Export all map vectors without reference to existing table ('where' clause ignored).
- v**
Verbose mode with statistics on the completion of the insertions.
- p**
Create and populate PostGIS GEOMETRY format table instead of normal Postgres geometry types.

Parameters:

key=databasecolumnname

Column in table "tab" of the currently selected database containing values corresponding to the vector maps category values. Table is designated on the command line by `tab=tablename` and vector is given on the command line by `map=mapname`.

tab=databasetablename

Table in the currently selected database containing a column which has values corresponding to vector category values in the map designated by `map=map`.

where=SQLwhereclause

SQL "where" clause which specifies the query criteria to be used in subsetting the database. The field names specified in the where option must indicate the column(s) to be used, the operators to be used in the evaluation and the values which the data in the column will be evaluated against.

If the database column used as the selection criteria is a character field then the associated value must be placed in quotes. To determine the data types associated with columns in the currently selected database use the `g.column.pg` command with the `-v` flag.

Queries which are more complex are best implemented using the `-s` flag and a prepared SQL file.

map=map

Name of an existing vector map with category values which correspond to values in a specified column in the currently selected database.

color=name

Color to draw vectors in

Colors: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, black

type=area,line

Type of the coverage to export.

ALTERNATE COMMAND LINE USAGE

The alternate command line usage is provided to simplify the process of retrieving information from more than one table in the query criteria. The alternate command line structure is selected using the the `[-s]` flag on the command line. When using this flag the user must include the name of a text file on the command line as well. This file must include a complete, well formed SQL SELECT statement specifying the query criteria to retrieve values which correspond to category values in a GRASS data layer.

Flag:

- s**
SQL select statements are input from a prepared file
- f**
Fill polygons selected on the query criteria
- v**

Verbose mode with statistics on the completion of the insertions.

-p

Create and populate PostGIS GEOMETRY format table instead of normal Postgres geometry types.

Parameters:

sql=filename

SQL statements specifying well formed selection criteria.

map=name

Name of an existing vector map

color=name

Color to draw vectors in

Colors: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, black

type=area,line

Type of the coverage to export.

EXAMPLE:

1. *v.to.pg -f key=rec_id map=kuruma_id tab=info_kuruma type=area where='type_id >32 and type_id < 38' color=red*

Result: only polygons with type in 33–37 range would be inserted in table *info_kuruma_bnd* and displayed in red color.

2. *v.to.pg -v -s -f -p sql=oak.sql map=kuruma_id type=area*

oak.sql is:

```
select rec_id from info_kuruma where type_id > 32 and type_id <38;
```

Result: this command would do the same as in the first example, some information printed on screen. However, the result table would be in PostGIS format.

BUGS

none

NOTE

This program requires the Postgres database software.

The 'total' mode of import (i.e., without referencing to existing table *and* to categories count in map) leads to that field 'num' in result tables would be incoherent (simply counts vectors from beginning to end).

SEE ALSO

[*g.column.pg*](#), [*g.select.pg*](#), [*g.stats.pg*](#), [*g.table.pg*](#), [*d.rast.pg*](#), [*d.site.pg*](#), [*d.what.r.pg*](#), [*d.what.s.pg*](#), [*d.what.v.pg*](#),
[*v.reclass.pg*](#)

AUTHOR

Alex Shevlakov (sixote@yahoo.com)