



**User Manual**  
**working with raster data**

**Version 1.0**

**20.09.2013**

**Author: Dirk Zacher**



Faculty for Mathematics and Computer Science

Database Systems for New Applications

58084 Hagen, Germany

## Table of Contents

1	Introduction .....	3
2	Quick Start – learning by doing.....	4
3	Reference Manual.....	30
3.1	Tile algebra data types .....	30
3.1.1	Grid data types .....	30
3.1.2	Spatial data types.....	31
3.1.3	Moving spatial data types .....	32
3.1.4	Instant spatial data types.....	33
3.2	Tile algebra operators .....	34
3.2.1	atinstant.....	34
3.2.2	atlocation.....	35
3.2.3	atperiods.....	36
3.2.4	atrange .....	37
3.2.5	bbox .....	38
3.2.6	CELL1 .....	39
3.2.7	CELL2 .....	40
3.2.8	CELLS .....	42
3.2.9	compose .....	43
3.2.10	deftime .....	44
3.2.11	fromline .....	45
3.2.12	fromregion .....	46
3.2.13	getgrid.....	47
3.2.14	inst .....	48
3.2.15	map .....	49
3.2.16	map2 .....	50
3.2.17	matchgrid .....	56
3.2.18	maximum .....	57
3.2.19	minimum.....	58
3.2.20	t2mt.....	59
3.2.21	tiles .....	60
3.2.22	toraster2 .....	61
3.2.23	toregion .....	62
3.2.24	val .....	62
A	References .....	63

## 1 Introduction

SECONDO is an extensible database management system (DBMS). It contains two algebras supporting work with raster data, Raster2 algebra and Tile algebra. Both algebras provide similar data types and operators for raster data. The main difference between Raster2 algebra and Tile Algebra lies in the implementation and usage of data types of these algebras.

In Raster2 algebra an object contains many raster data, for example height data at coordinates North 50 East 7 through North 52 East 10 in Germany. In contrast, a Tile algebra object normally contains less raster data than a Raster2 algebra object, because Tile algebra data types were implemented as attribute data types. Thus Tile algebra data types can be used in relations in contrast to Raster2 algebra data types.

Display of Raster2 algebra data types and Tile algebra data types in SECONDO GUI was implemented identical to ensure comparable visualisation.

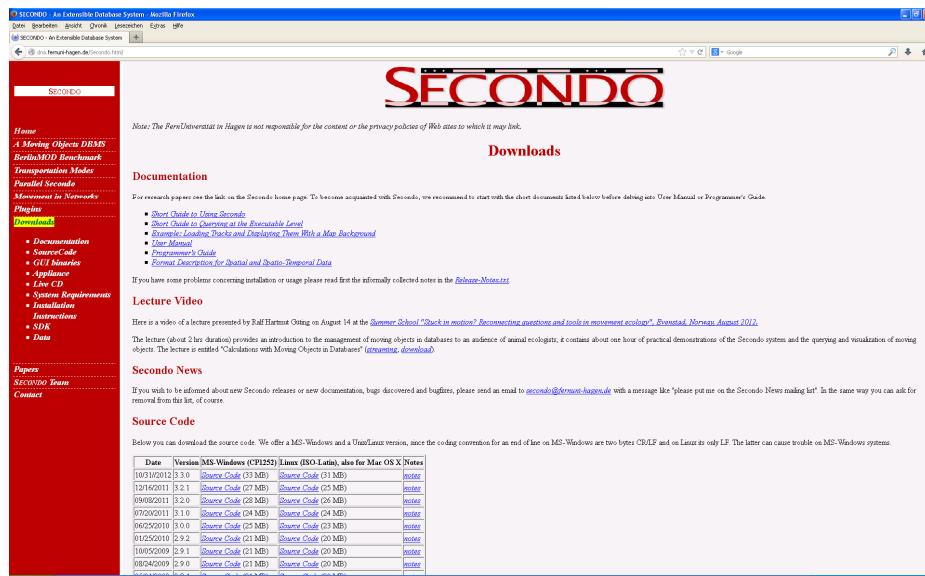
It is also possible to work with objects of Raster2 algebra and objects of Tile algebra at the same time. For this purpose Tile algebra provides operators tiles and toraster2. These operators allow the conversion of a Raster2 algebra object to a corresponding Tile algebra object and the reverse conversion of a Tile algebra object to a corresponding Raster2 algebra object.

## 2 Quick Start – learning by doing

The following chapter describes how to work with the SECONDO system and particularly how to work with raster data in SECONDO.

Work through the following steps:

1. If you do not have a SECONDO installation, go to the SECONDO download page <http://dna.fernuni-hagen.de/Secondo.html> and select an installation suitable for your platform, for example Linux, Mac OS X or Microsoft Windows.



2. If necessary, get the latest version of SECONDO from the web site and extract it, preferably into the directory \$HOME/secondo.

3. Open a shell window and change into the SECONDO directory by command **cd secondo**.



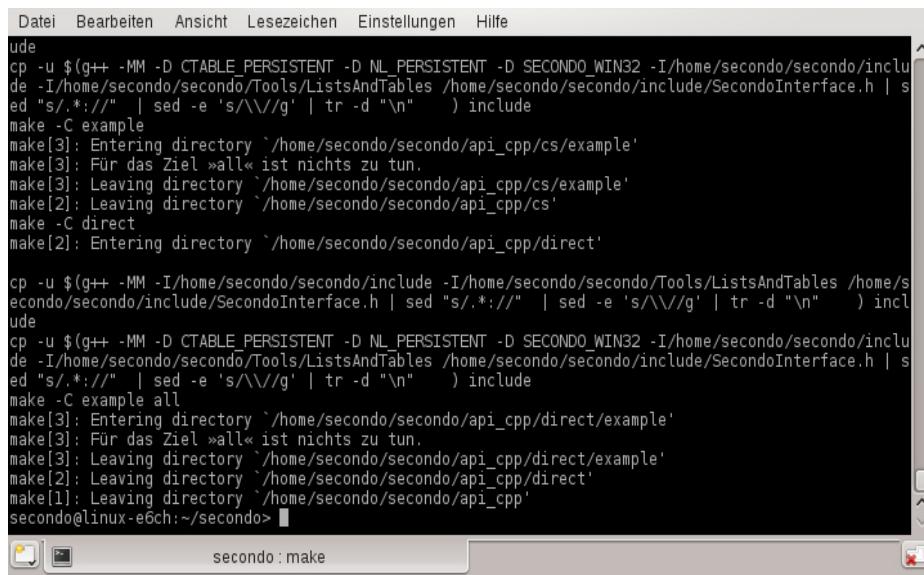
The screenshot shows a terminal window with a dark background and light-colored text. At the top, there is a menu bar with German labels: Datei, Bearbeiten, Ansicht, Lesezeichen, Einstellungen, Hilfe. Below the menu, the terminal prompt is 'seconde@linux-e6ch:~>'. The user then types 'cd secondo' and presses Enter. The response 'seconde@linux-e6ch:~/seconde>' is displayed, indicating the successful change of directory. The window has a standard title bar with icons for minimize, maximize, and close, and is titled 'seconde:bash'.

4. If the location of SECONDO system is not the directory \$HOME/seconde, you need to set an environment variable to this directory.  
On Windows installations this can be done by command **setvar \$PWD**,  
on Linux use command **secroot**.  
In any case what happens is **export SECONDO\_BUILD\_DIR=\$PWD**.

5. Compile your SECONDO system by command make.



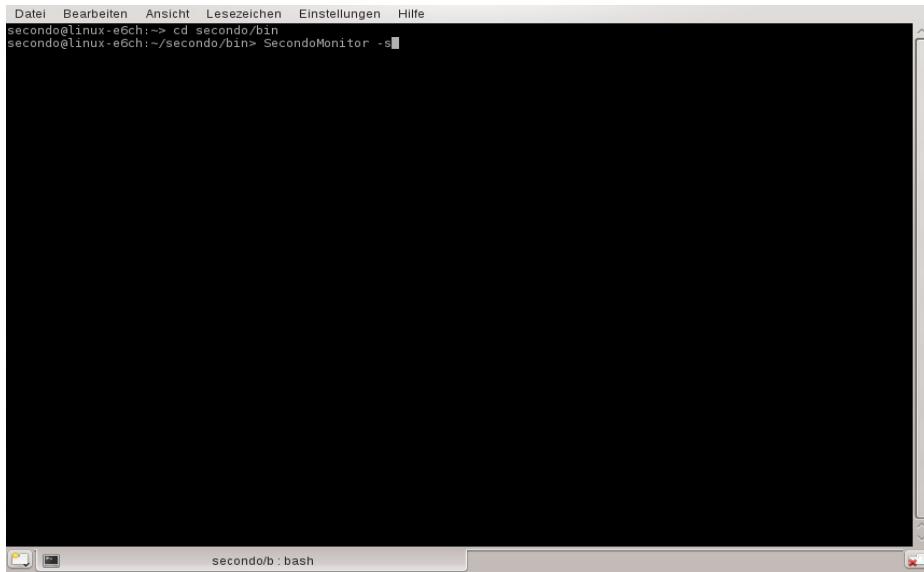
```
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
seondo@linux-e6ch:~/seondo> make
```



```
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
ude
cp -u $(g++ -MM -I/home/seondo/include -I/home/seondo/secondo/Tools/ListsAndTables /home/seondo/include/SecondoInterface.h | sed "s/.*/"/" | sed -e 's/\\//g' | tr -d "\n" ) include
make -C example
make[3]: Entering directory `/home/seondo/seondo/api_cpp/cs/example'
make[3]: Für das Ziel »all« ist nichts zu tun.
make[3]: Leaving directory `/home/seondo/seondo/api_cpp/cs/example'
make[2]: Leaving directory `/home/seondo/seondo/api_cpp/cs'
make -C direct
make[2]: Entering directory `/home/seondo/seondo/api_cpp/direct'
cp -u $(g++ -MM -D CTABLE_PERSISTENT -D NL_PERSISTENT -D SECONDO_WIN32 -I/home/seondo/seondo/include -I/home/seondo/secondo/Tools/ListsAndTables /home/seondo/include/SecondoInterface.h | sed "s/.*/"/" | sed -e 's/\\//g' | tr -d "\n" ) include
make[3]: Entering directory `/home/seondo/seondo/api_cpp/direct/example'
make[3]: Für das Ziel »all« ist nichts zu tun.
make[3]: Leaving directory `/home/seondo/seondo/api_cpp/direct/example'
make[2]: Leaving directory `/home/seondo/seondo/api_cpp/direct'
make[1]: Leaving directory `/home/seondo/seondo/api_cpp'
seondo@linux-e6ch:~/seondo>
```

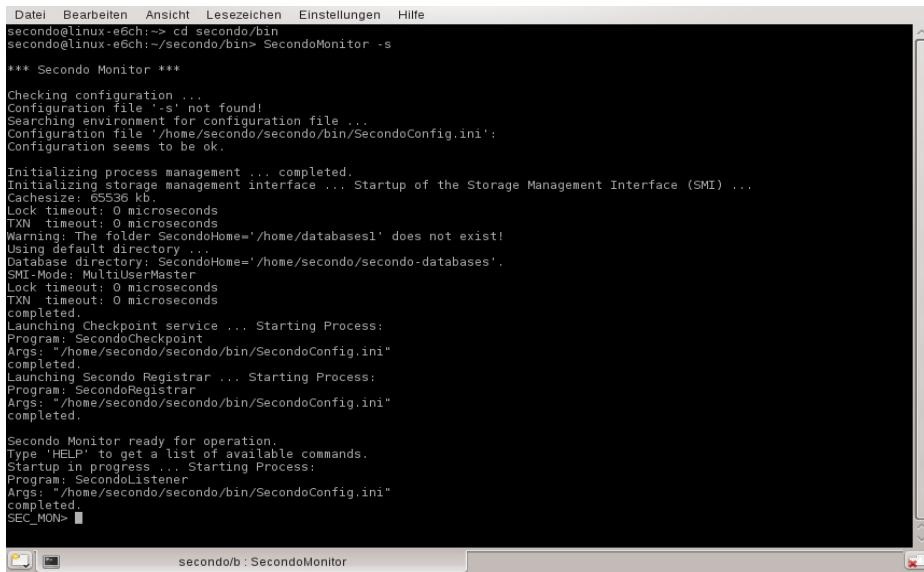
After compile a runnable SECONDO system is available.

6. Open a new shell window, change into directory secondo/bin by command cd secondo/bin and execute command **SecondoMonitor -s**.



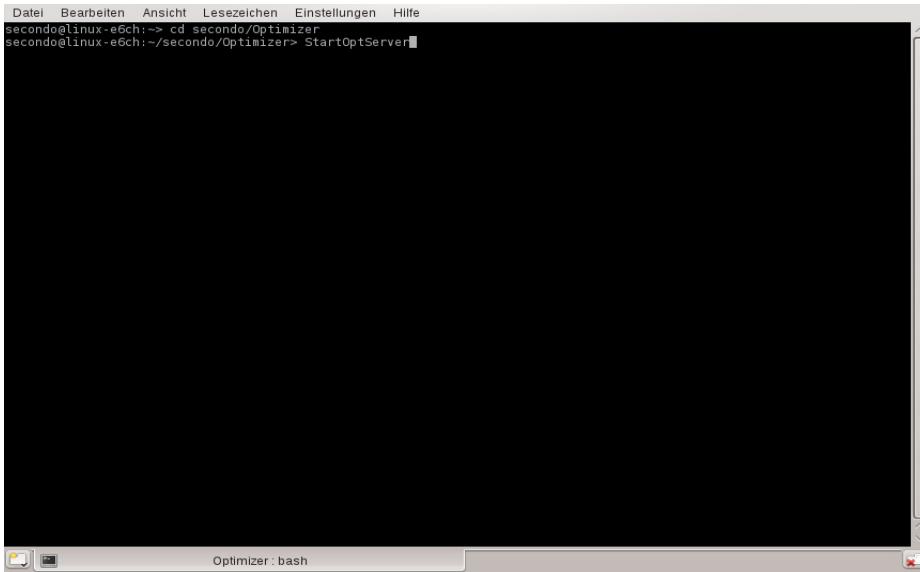
```
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe  
seconde@linux-e6ch:~> cd secondo/bin  
seconde@linux-e6ch:~/seconde/bin> SecondoMonitor -s
```

Command **SecondoMonitor -s** starts a process listening for user interfaces to register with the kernel.



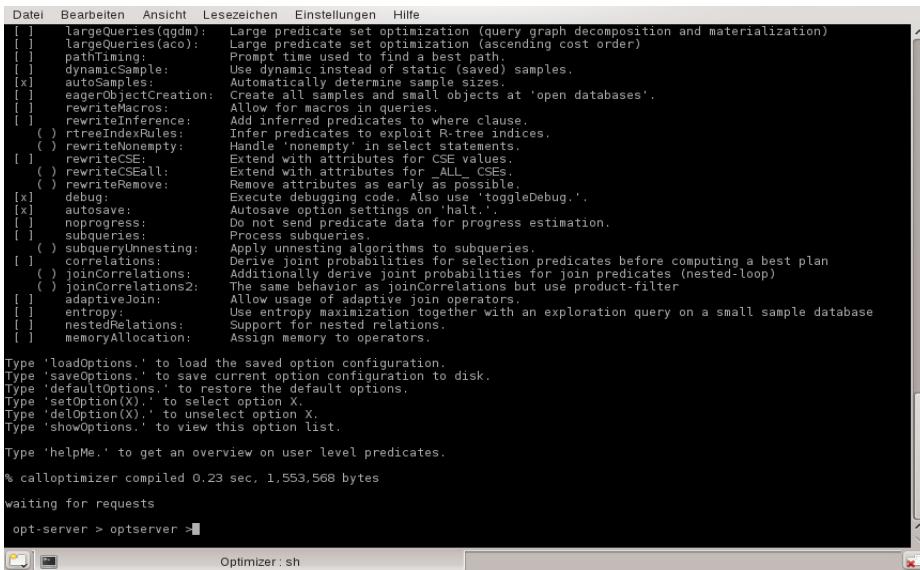
```
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe  
seconde@linux-e6ch:~> cd secondo/bin  
seconde@linux-e6ch:~/seconde/bin> SecondoMonitor -s  
*** Secondo Monitor ***  
  
Checking configuration...  
Configuration file 's' not found!  
Searching environment for configuration file...  
Configuration file '/home/seconde/seconde/bin/SecondoConfig.ini':  
Configuration seems to be ok.  
  
Initializing process management ... completed.  
Initializing storage management interface ... Startup of the Storage Management Interface (SMI) ...  
Cachesize: 65536 kb.  
Lock timeout: 0 microseconds  
TXN timeout: 0 microseconds  
Warning: folder SecondoHome='/home/databases1' does not exist!  
Using default directory.  
Database directory: SecondoHome='/home/seconde/seconde-databases'.  
SMI-Mode: MultiUserMaster  
Lock timeout: 0 microseconds  
TXN timeout: 0 microseconds  
completed.  
Launching Checkpoint service ... Starting Process:  
Program: SecondoCheckpoint  
Args: "/home/seconde/seconde/bin/SecondoConfig.ini"  
completed.  
Launching Secondo Registrar ... Starting Process:  
Program: SecondoRegistrar  
Args: "/home/seconde/seconde/bin/SecondoConfig.ini"  
completed.  
  
Secondo Monitor ready for operation.  
Type 'HELP' to get a list of available commands.  
Startup in progress ... Starting Process:  
Program: SecondoListener  
Args: "/home/seconde/seconde/bin/SecondoConfig.ini"  
completed.  
SEC_MON>
```

7. Open a new shell window, change into directory secondo/Optimizer by command cd secondo/Optimizer and execute command StartOptServer.



```
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
secondeo@linux-e6ch:~> cd secondo/Optimizer
secondeo@linux-e6ch:~/secondeo/Optimizer> StartOptServer
```

Command StartOptServer starts the optimizer process.

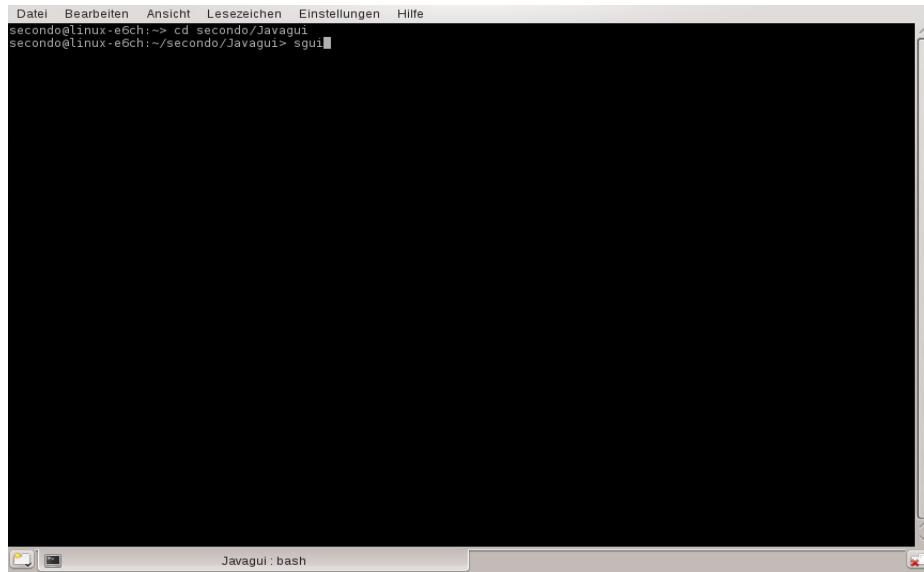


```
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
[ ] largeQueries(qdm): Large predicate set optimization (query graph decomposition and materialization)
[ ] largeQueries(aco): Large predicate set optimization (ascending cost order)
[ ] pathTiming: Prompt time used to find a best path.
[ ] dynamicSample: Use dynamic instead of static (saved) samples.
[x] autoSamples: Automatically determine sample sizes.
[ ] eagerObjectCreation: Create all samples and small objects at 'open databases'.
[ ] rewriteMacros: Allow for macros in queries.
[ ] rewriteInference: Add inferred predicates to where clause.
( ) rtreeIndexRules: Infer predicates to exploit RTree indices.
( ) rewriteEmpty: Handle empty relations in select statements.
[ ] rewriteCSE: Extend with attributes for CSE values.
( ) rewriteCSEAll: Extend with attributes for ALL_CSEs.
( ) rewriteRemove: Remove attributes as early as possible.
[x] debug: Execute debugging code. Also use 'toggleDebug'.
[x] autosave: Autosave option settings on 'halt'.
[ ] noprogress: Do not send predicate data for progress estimation.
[ ] subqueries: Process subqueries.
( ) subqueryUnnesting: Apply unnesting algorithms to subqueries.
[ ] correlations: Derive joint probabilities for selection predicates before computing a best plan
( ) nestedCorrelations: Add nested relations to joint probabilities for join predicates (nested-loop)
( ) joinCorrelations2: The same behavior as joinCorrelations but use product-filter
[ ] adaptiveJoin: Allow usage of adaptive join operators.
[ ] entropy: Use entropy maximization together with an exploration query on a small sample database
[ ] nestedRelations: Support for nested relations.
[ ] memoryAllocation: Assign memory to operators.

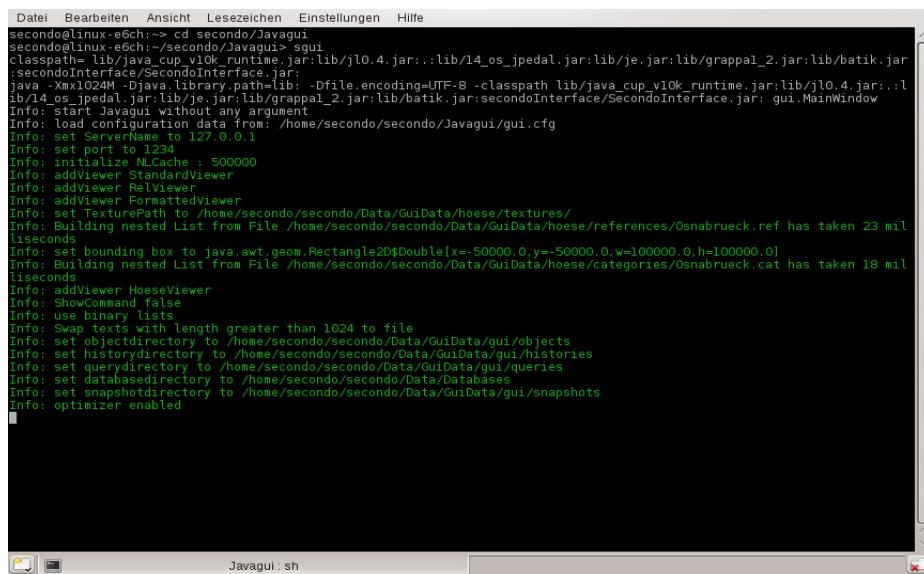
Type 'loadOptions.' to load the saved option configuration.
Type 'saveOptions.' to save current option configuration to disk.
Type 'defaultOptions.' to restore the default options.
Type 'setOption(X)...' to select option X.
Type 'delOption(X)...' to unselect option X.
Type 'showOptions.' to view this option list.

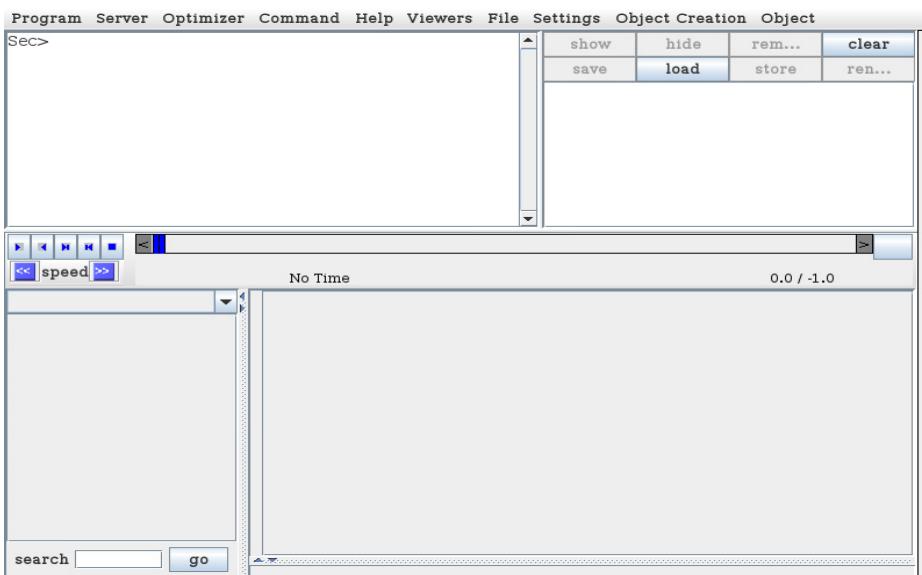
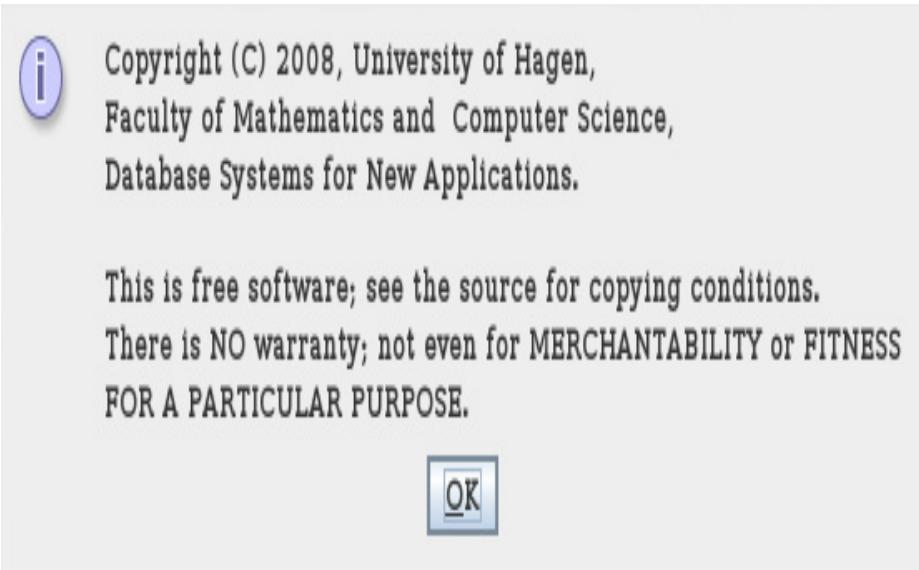
Type 'helpMe.' to get an overview on user level predicates.
% calloptimizer compiled 0.23 sec, 1,553,568 bytes
waiting for requests
opt-server > optserver >
```

8. Open a new shell window, change into directory secondo/Javagui by command **cd secondo/Javagui** and execute command **sgui**.

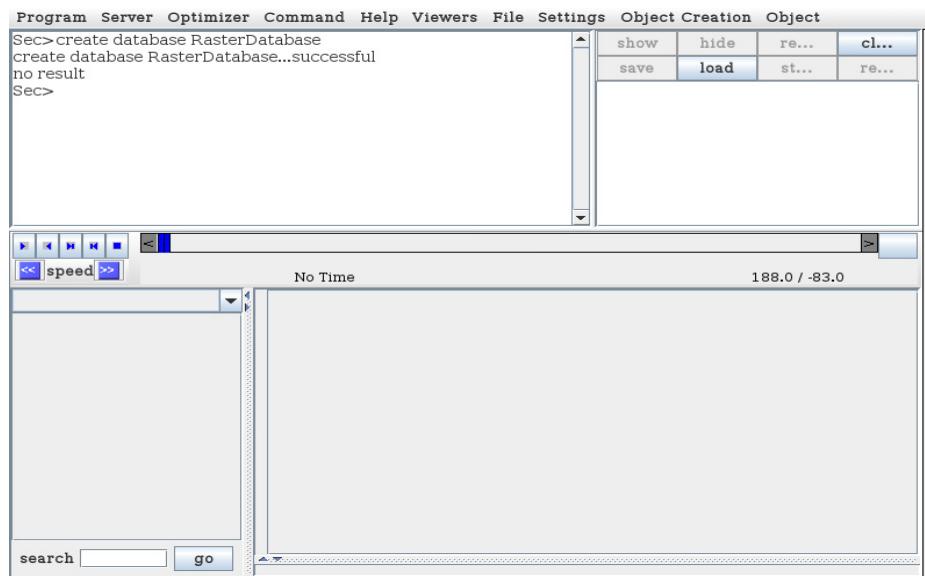


Command **sgui** starts the graphical user interface of SECONDO.

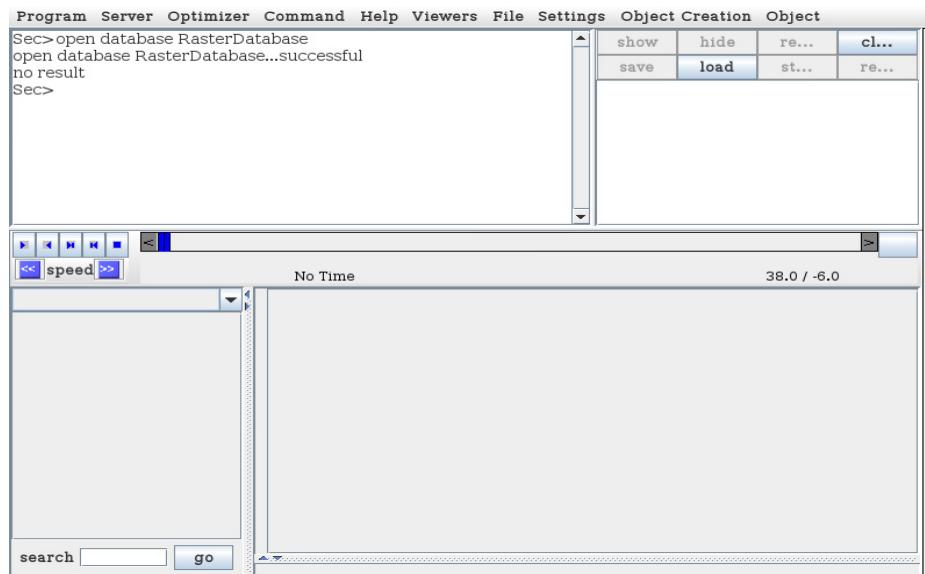




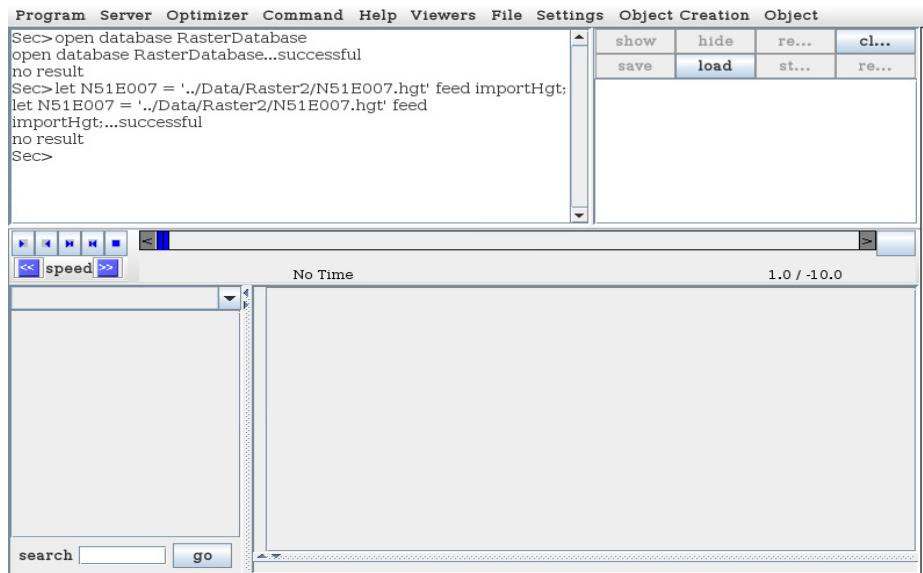
9. At the command window in the upper left area of the SECONDO GUI execute command **create database RasterDatabase**.



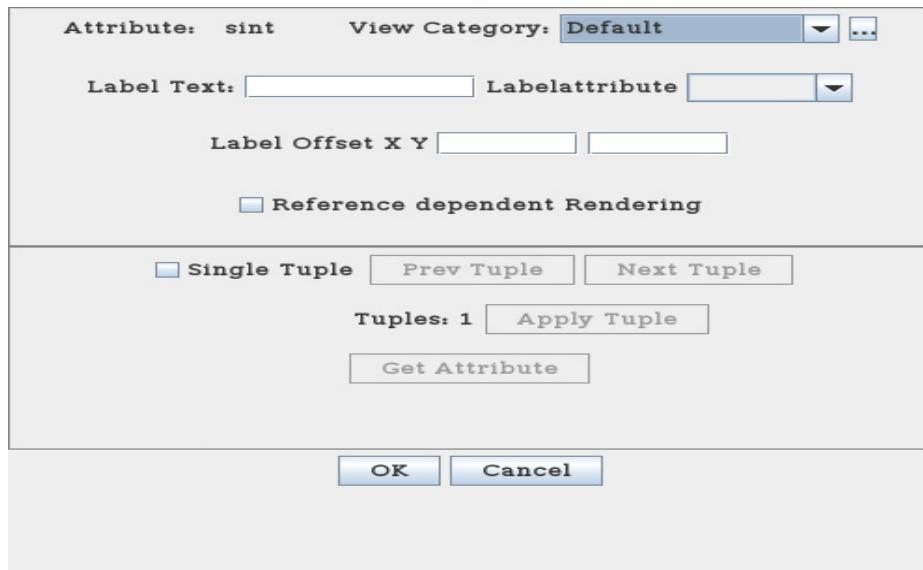
10. Open database RasterDatabase by command **open database RasterDatabase**.

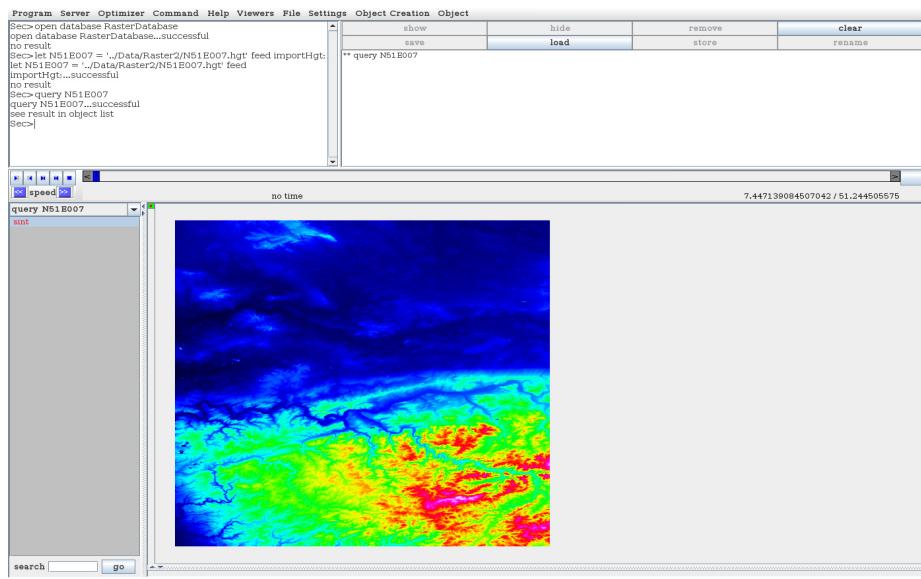


11. Import raster data of hgt format from file secondo/Data/Raster2/N51E007.hgt into a Raster2 algebra object of type sint by command  
let N51E007 = './Data/Raster2/N51E007.hgt' feed importHgt.

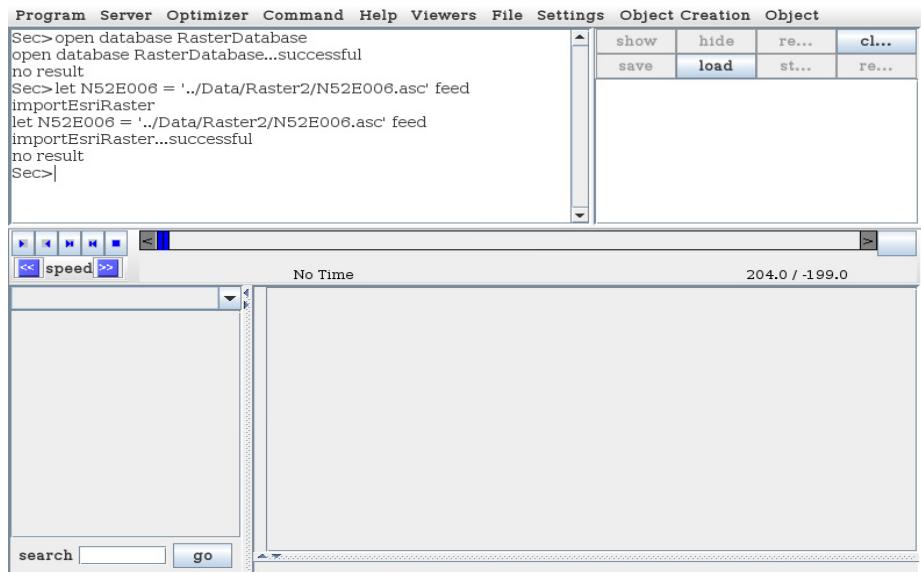


12. Display Raster2 algebra N51E007 object of type sint in Hoese Viewer by command query N51E007.

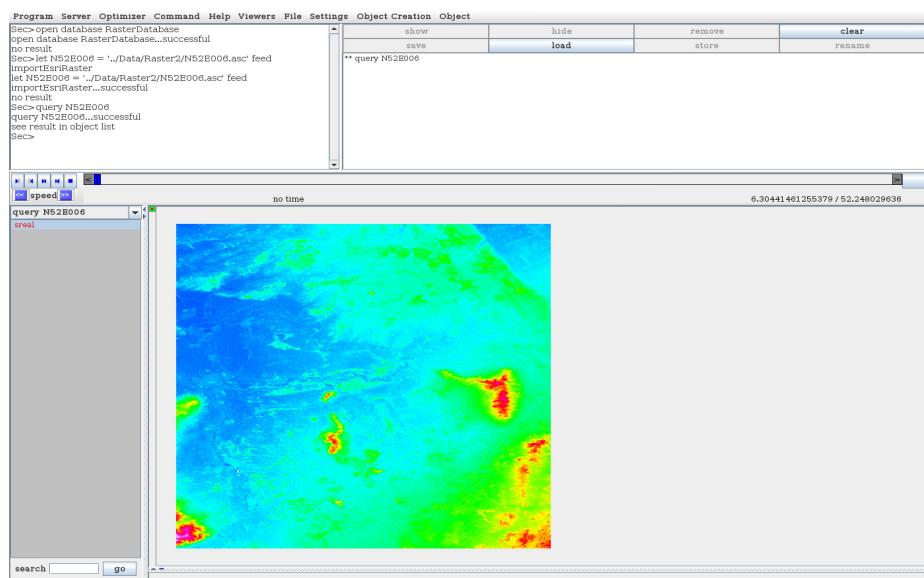
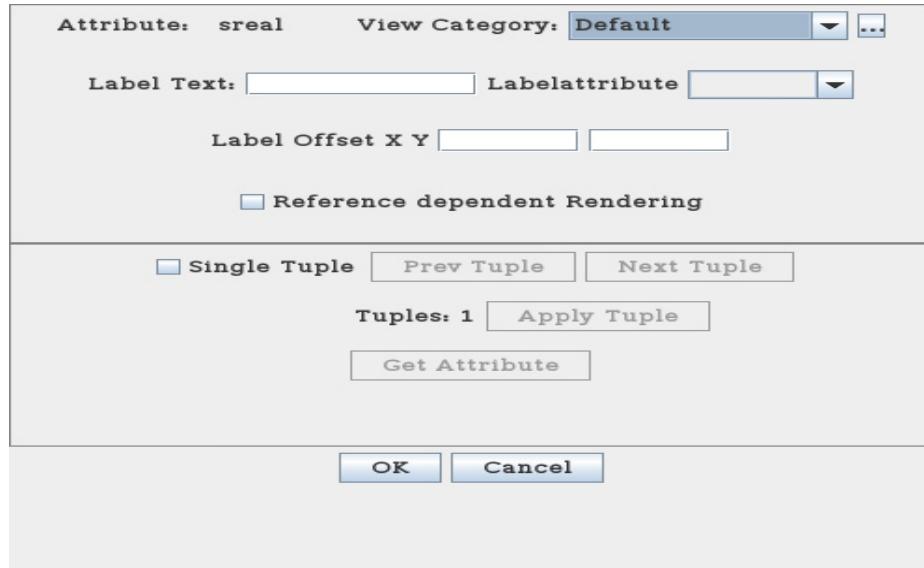




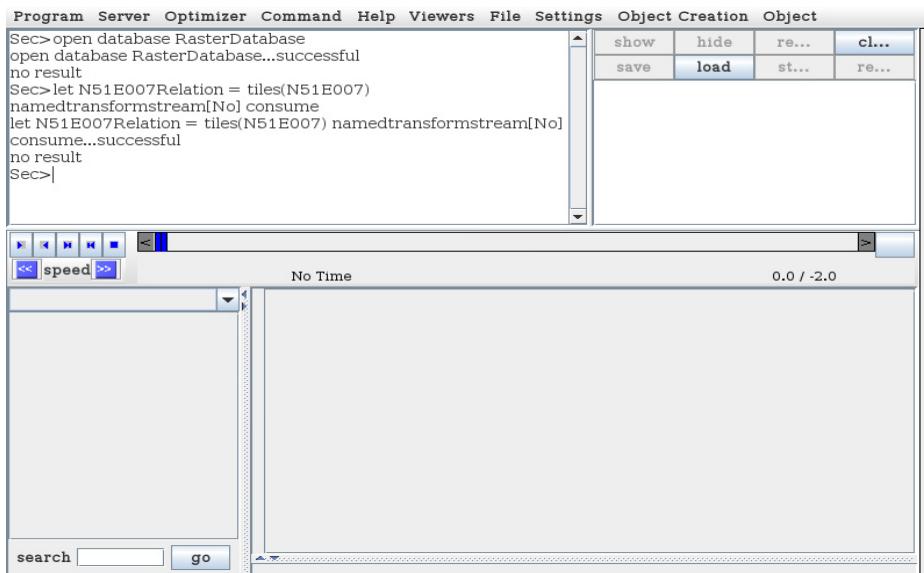
13. Import raster data of esri raster format from file secondo/Data/Raster2/N52E006.asc into a Raster2 algebra object of type sreal by command  
let N52E006 = './Data/Raster2/N52E005.asc' feed importEsriRaster.



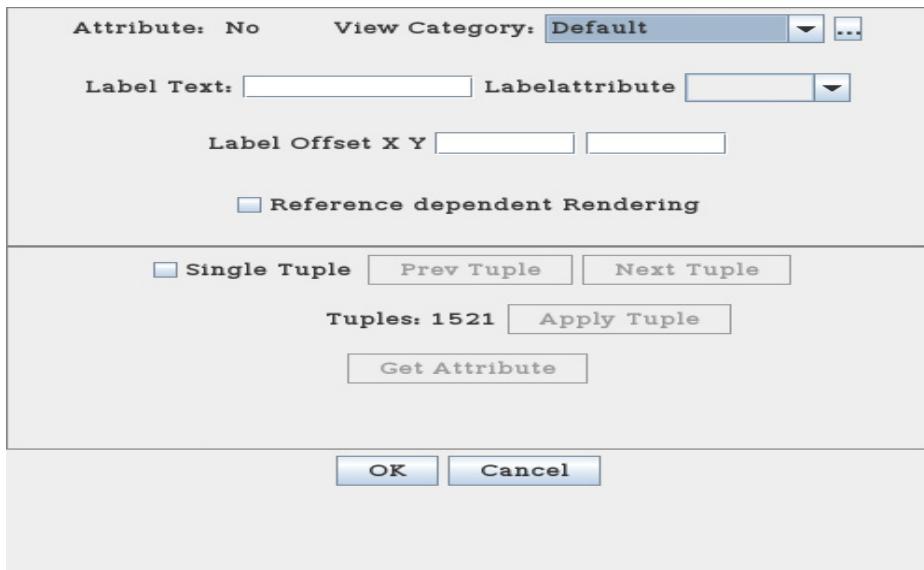
14. Display Raster2 algebra N52E006 object of type sreal in Hoese Viewer by command query N52E006.

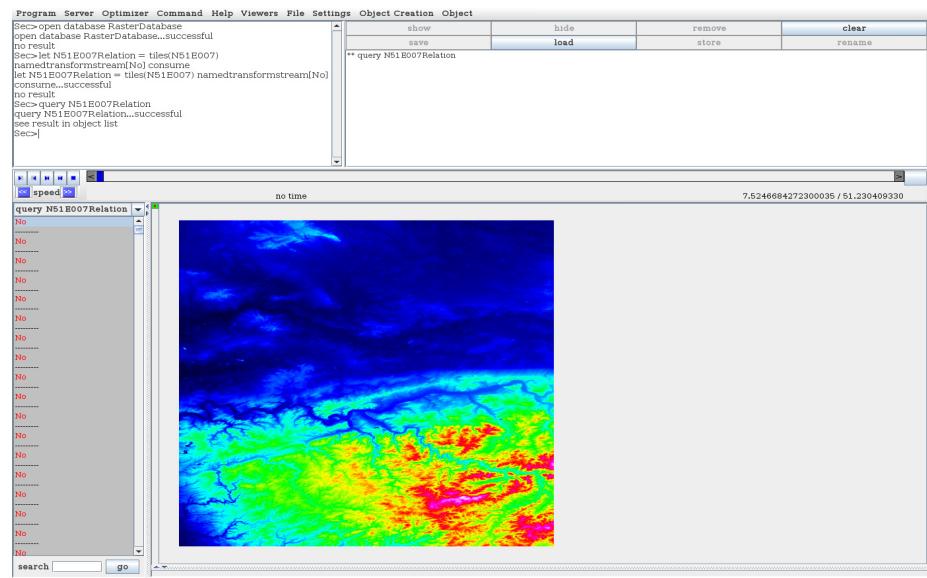


15. Convert Raster2 algebra N51E007 object of type sint into a relation of Tile algebra tint objects and save result in N51E007Relation object by command  
let N51E007Relation = tiles(N51E007) namedtransformstream[No] consume.

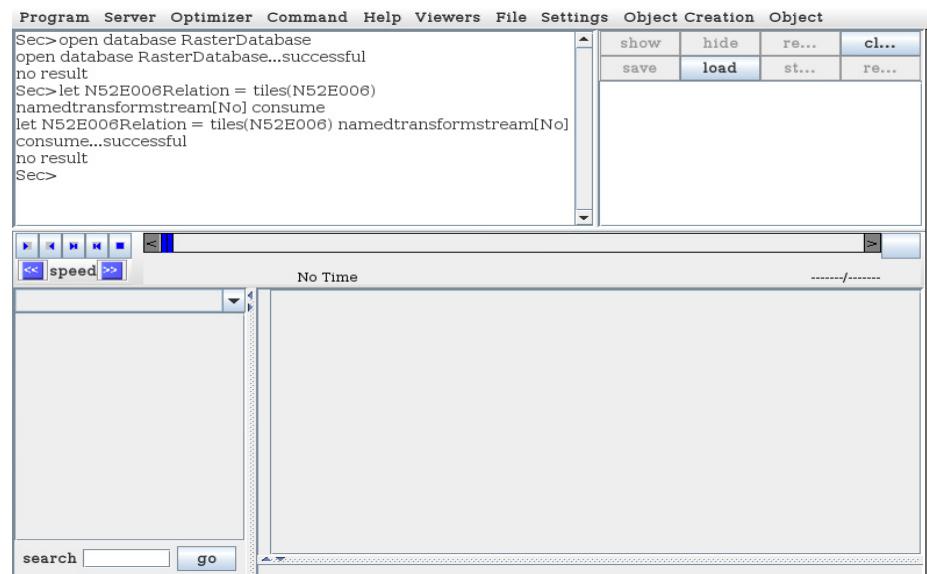


16. Display relation N51E007Relation of Tile algebra tint objects in Hoes Viewr by command query N51E007Relation.

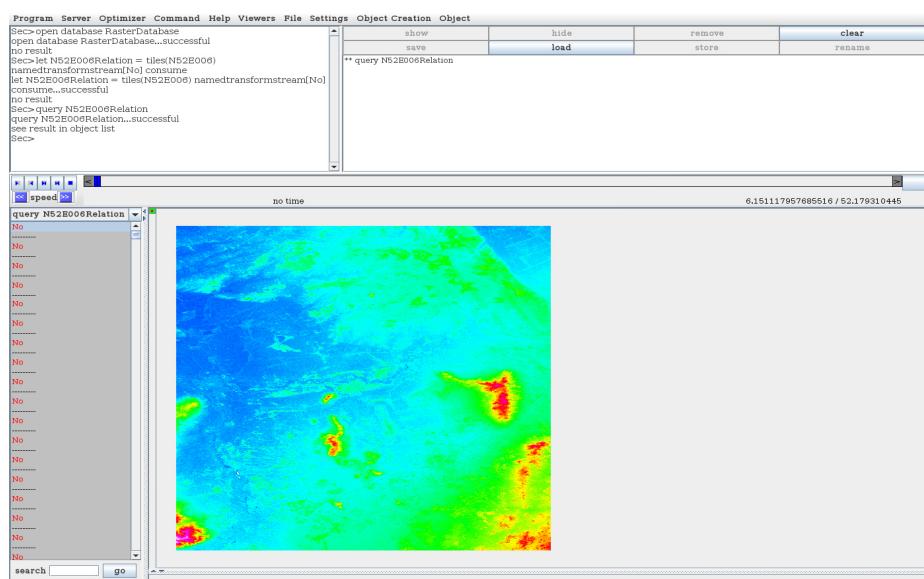
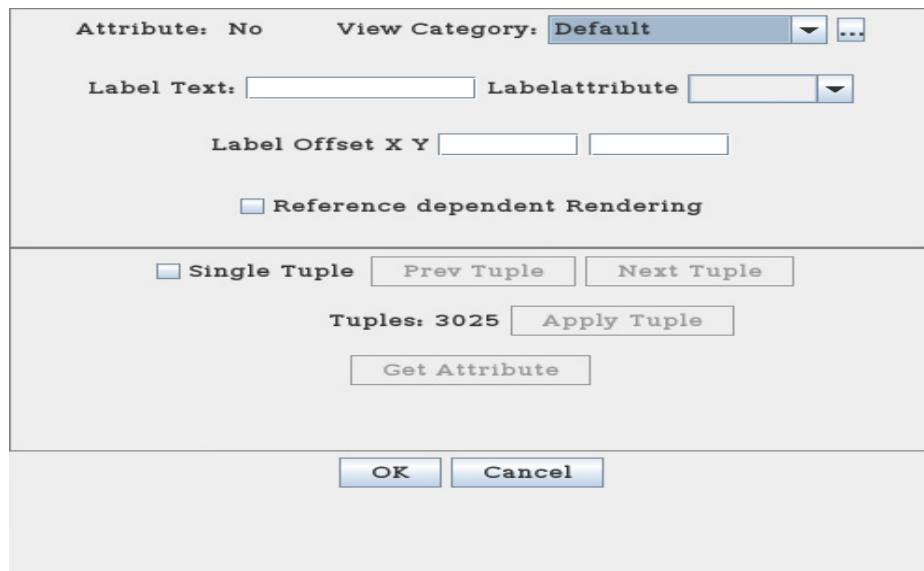




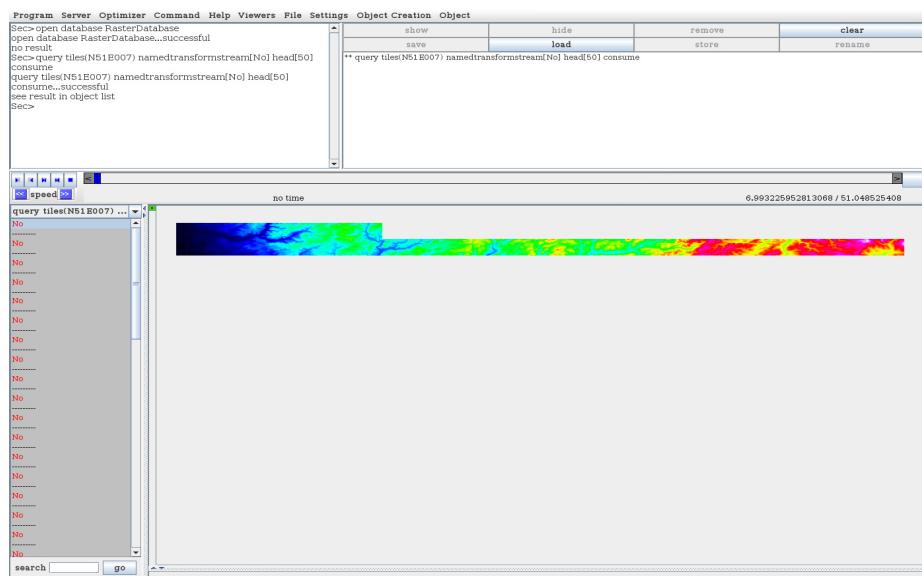
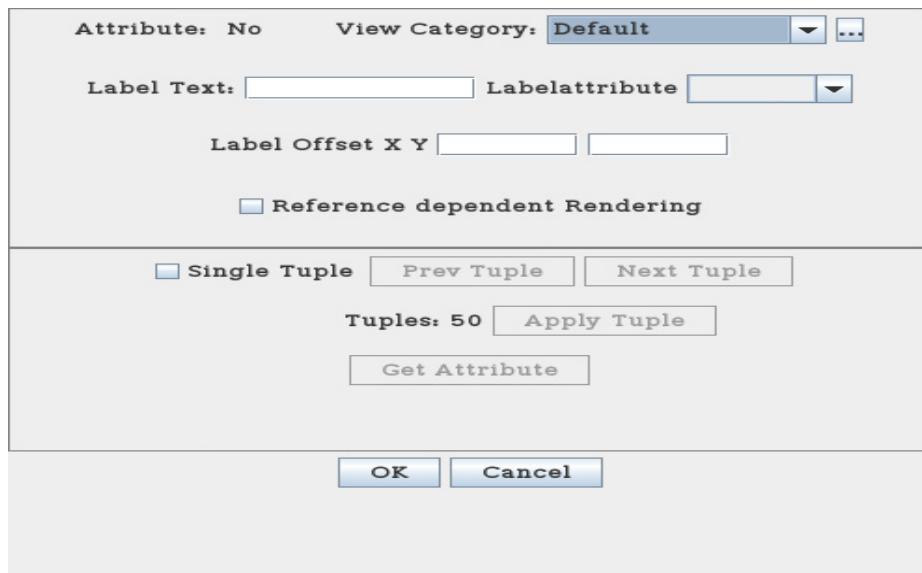
17. Convert Raster2 algebra N52E006 object of type sreal into a relation of Tile algebra treal objects and save result in N52E006Relation object by command let N52E006Relation = tiles(N52E006) namedtransformstream[No] consume.



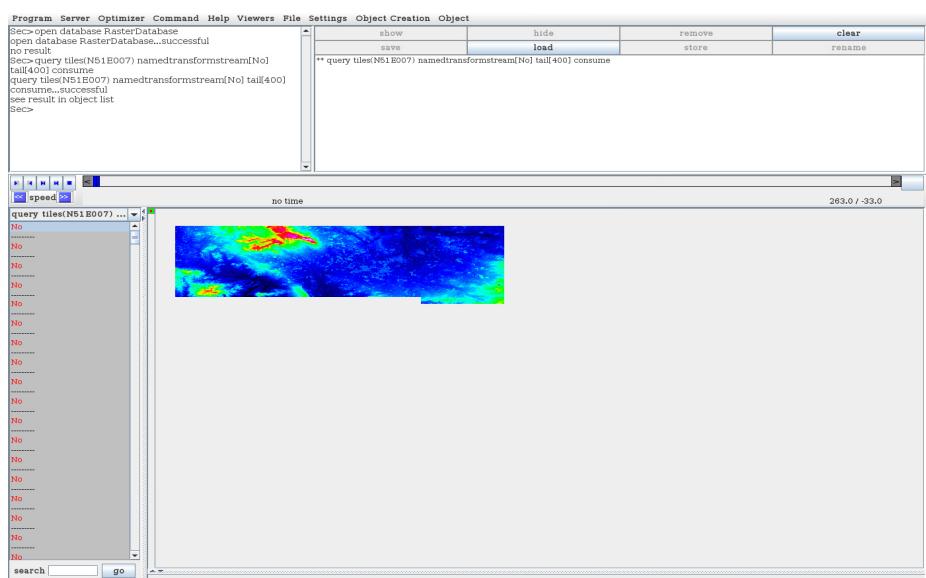
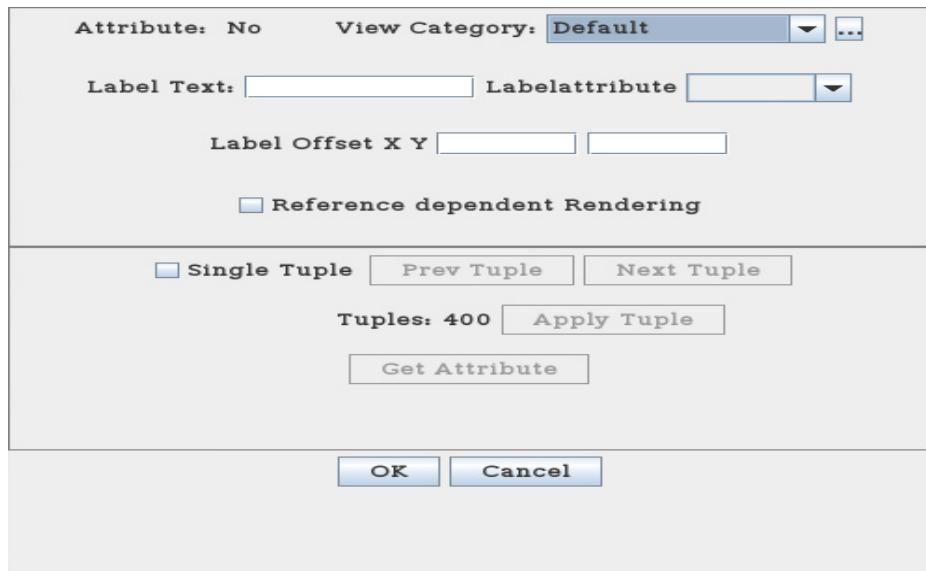
18. Display relation N52E006Relation of Tile algebra treal objects in Hoesel Viewer by command query N52E007Relation.



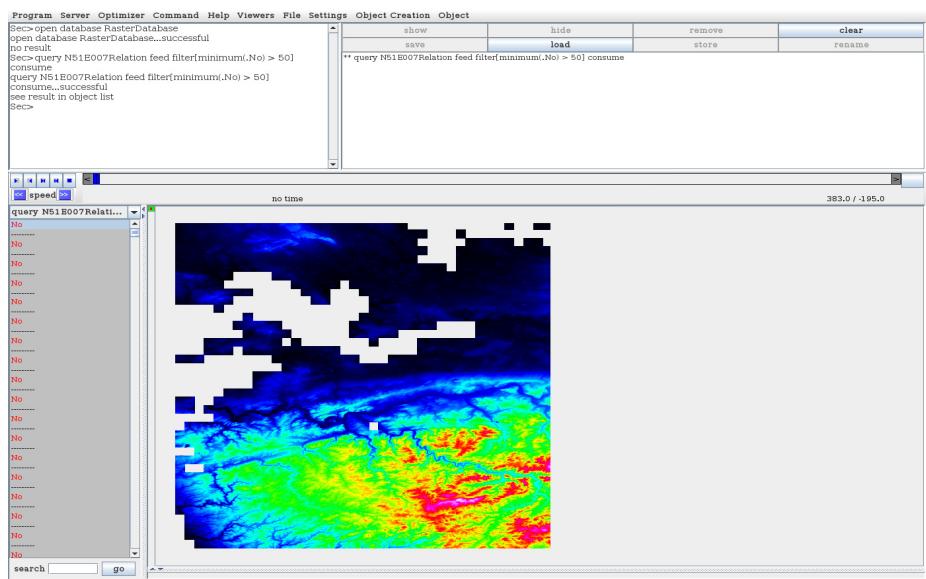
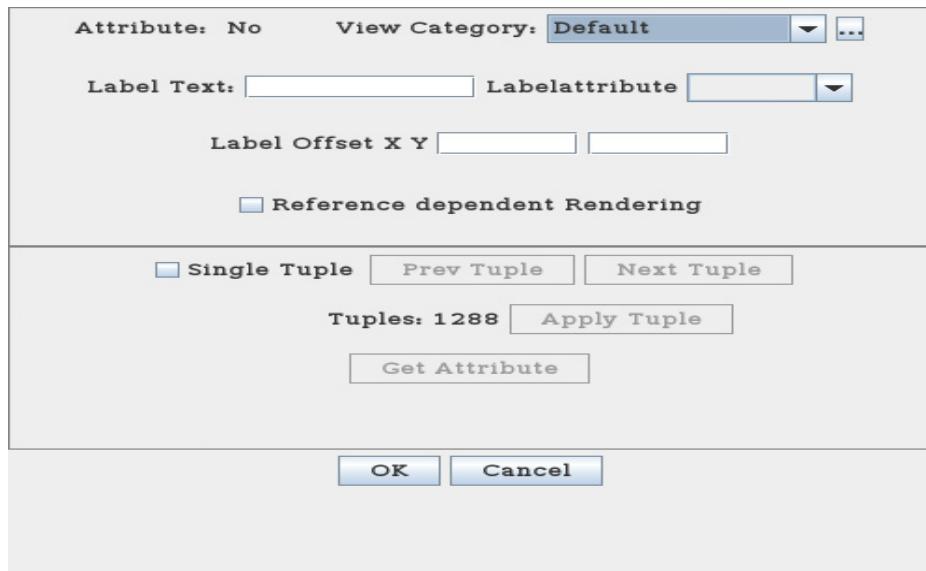
19. Convert Raster2 algebra N51E007 object of type sint into a relation of Tile algebra tint objects and display the first 50 tiles by command query tiles(N51E007) namedtransformstream[No] head[50] consume.



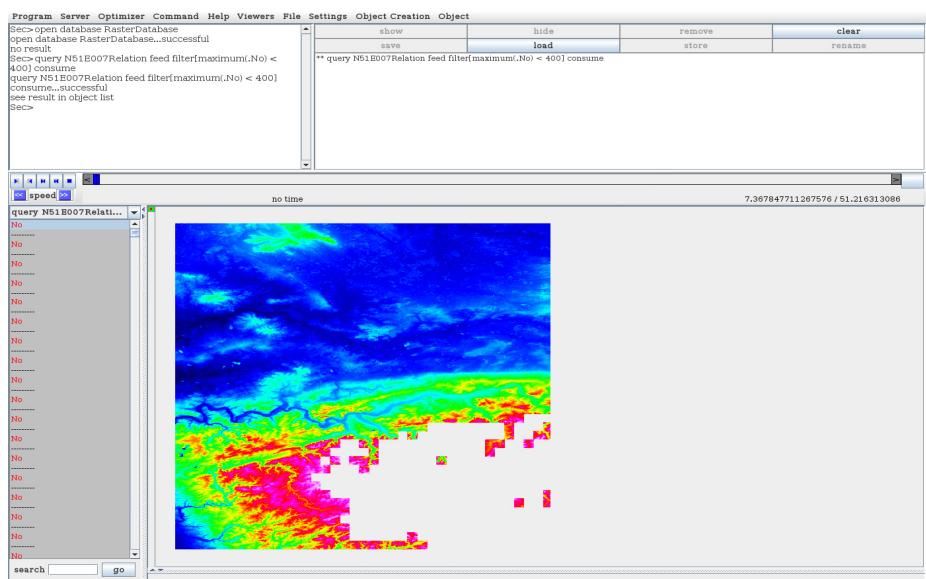
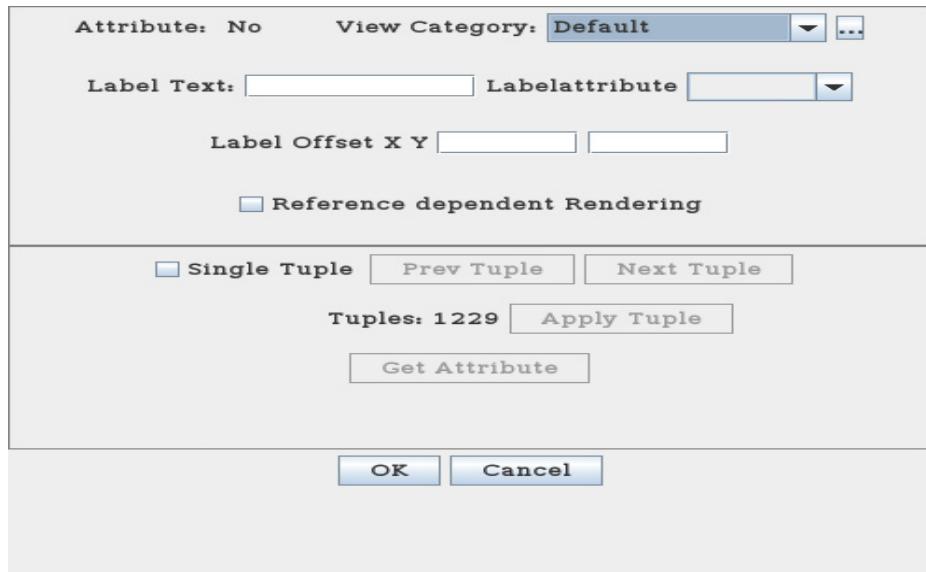
20. Convert Raster2 algebra N51E007 object of type sint into a relation of Tile algebra tint objects and display the last 400 tiles by command query tiles(N51E007) namedtransformstream[No] tail[400] consume.



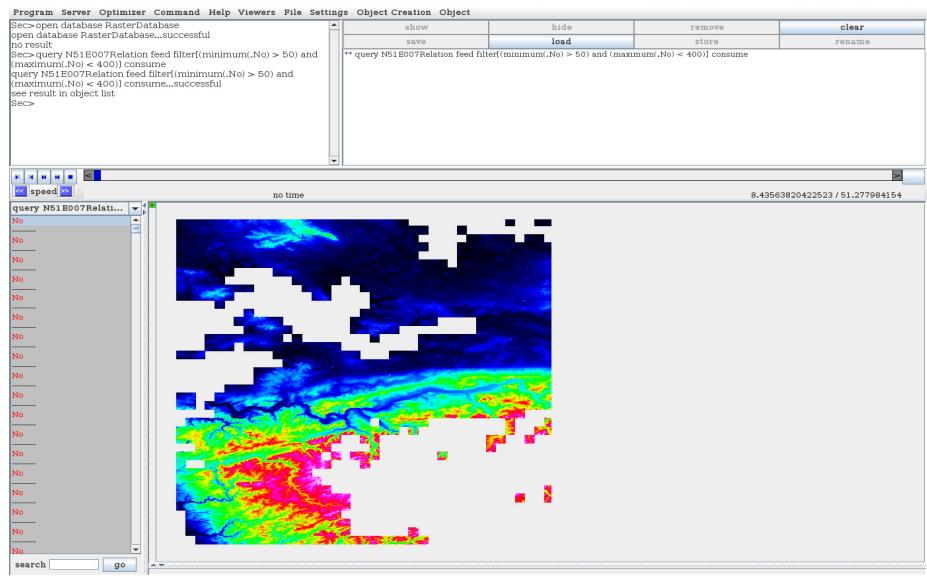
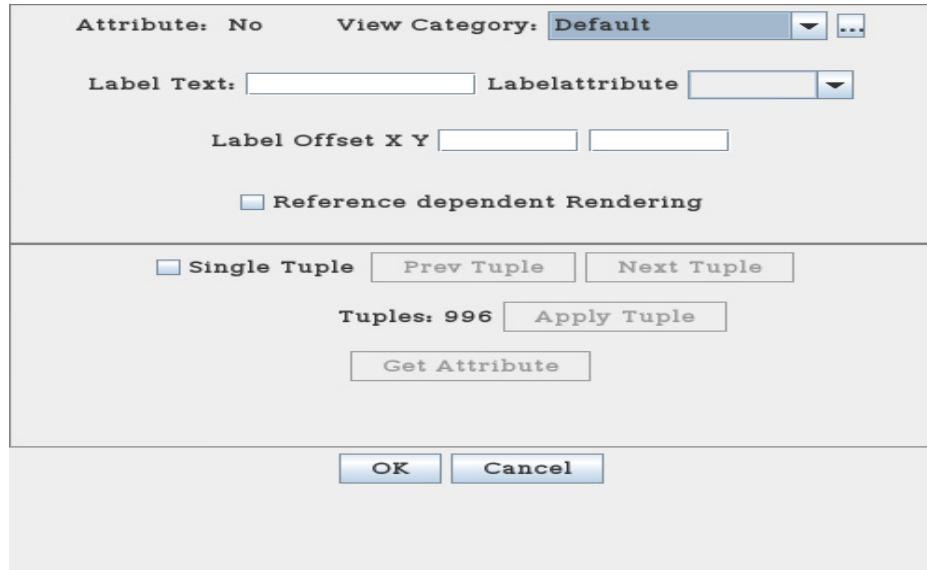
21. Display all tiles of N51E007Relation of Tile algebra tint objects whose minimum is greater than 50 by command  
query N51E007Relation feed filter[minimum(.No) > 50] consume.



22. Display all tiles of N51E007Relation of Tile algebra tint objects whose maximum is less than 400 by command  
query N51E007Relation feed filter[maximum(.No) < 400] consume.



23. Display all tiles of N51E007Relation of Tile algebra tint objects whose minimum is greater than 50 and maximum is less than 400 by command  
query N51E007Rleation feed filter[(minimum(.No) > 50) and  
(maximum(.No) < 400)] consume.



24. Display minimum value of all tiles of N51E007Relation by command  
query N51E007Relation feed extend[M : minimum(.No)] min[M].

The screenshot shows a software interface with a menu bar (Program, Server, Optimizer, Command, Help, Viewers, File, Settings, Object Creation, Object) and a toolbar with buttons for show, hide, save, load, store, and clear. The main window has two panes. The left pane contains a command-line history:

```
Sec>open database RasterDatabase
open database RasterDatabase...successful
no result
Sec>query N51E007Relation feed extend[M : minimum(.No)]
min[M]
query N51E007Relation feed extend[M : minimum(.No)]
min[M]...successful
see result in object list
Sec>|
```

The right pane displays the results of the query:

```
** query N51E007Relation feed extend[M : minimum(.No)]
int : 7
```

Below the panes are buttons for search and go, and a status bar showing "no time" and "233.0 / -0.0".

25. Display maximum value of all tiles of N51E007Relation by command  
query N51E007Relation feed extend[M : maximum(.No)] max[M].

The screenshot shows a software interface with a menu bar (Program, Server, Optimizer, Command, Help, Viewers, File, Settings, Object Creation, Object) and a toolbar with buttons for show, hide, save, load, store, and clear. The main window has two panes. The left pane contains a command-line history:

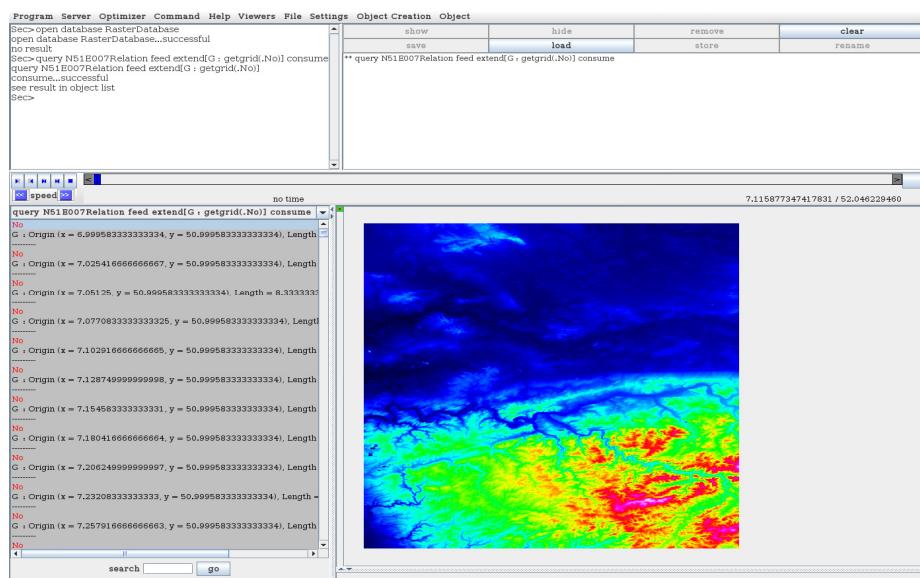
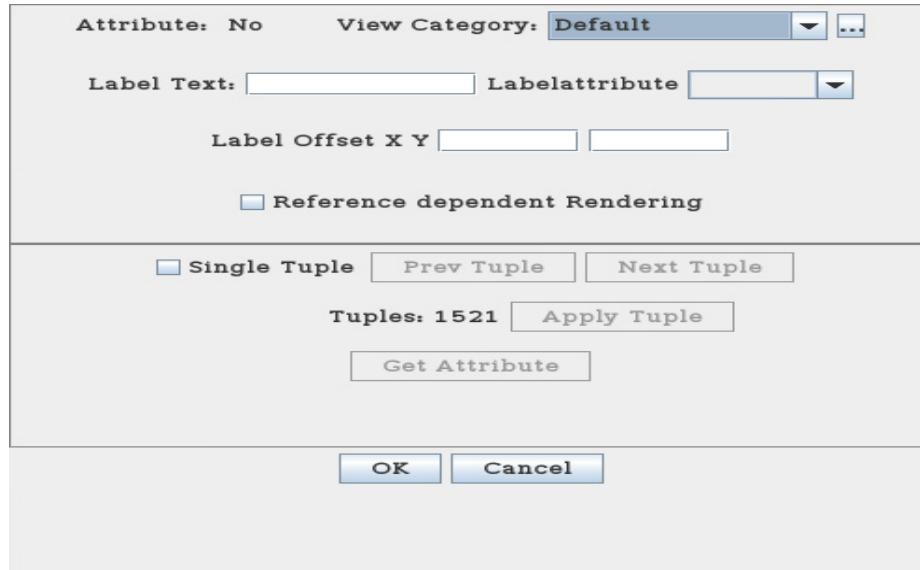
```
Sec>open database RasterDatabase
open database RasterDatabase...successful
no result
Sec>query N51E007Relation feed extend[M : maximum(.No)]
max[M]
query N51E007Relation feed extend[M : maximum(.No)]
max[M]...successful
see result in object list
Sec>|
```

The right pane displays the results of the query:

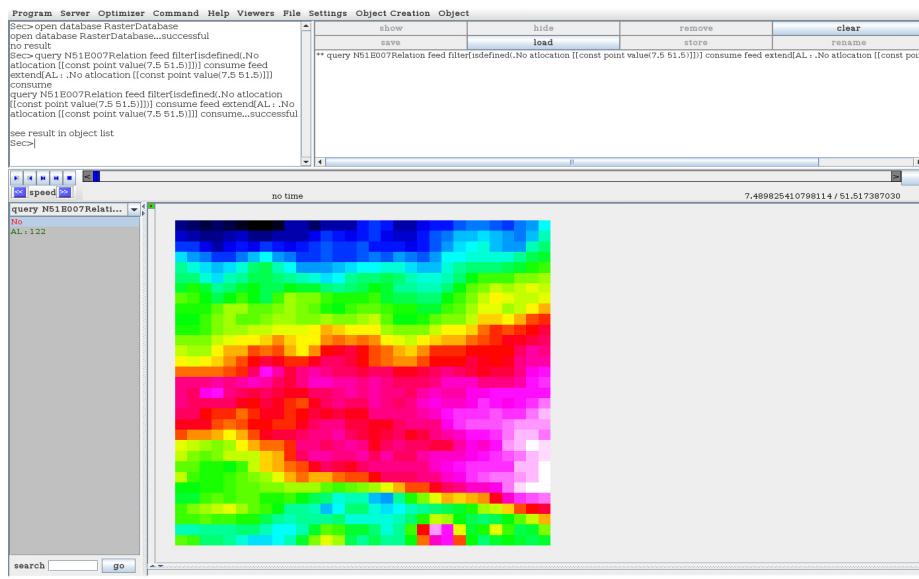
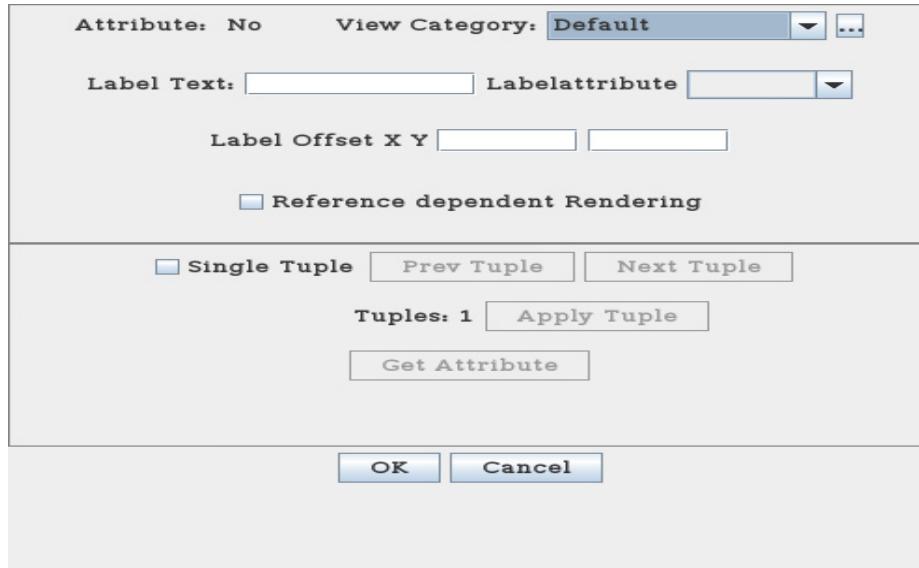
```
** query N51E007Relation feed extend[M : maximum(.No)]
int : 673
```

Below the panes are buttons for search and go, and a status bar showing "no time" and "214.0 / -5.0".

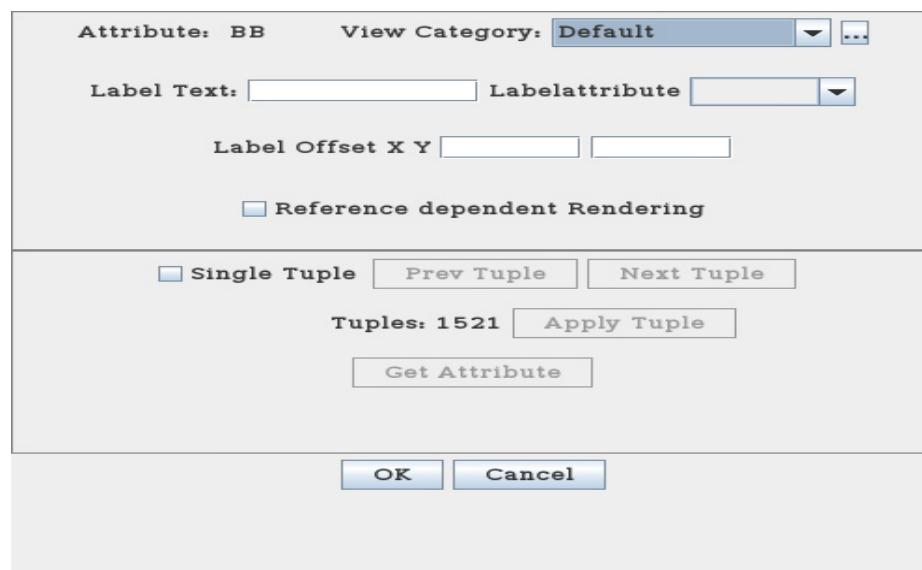
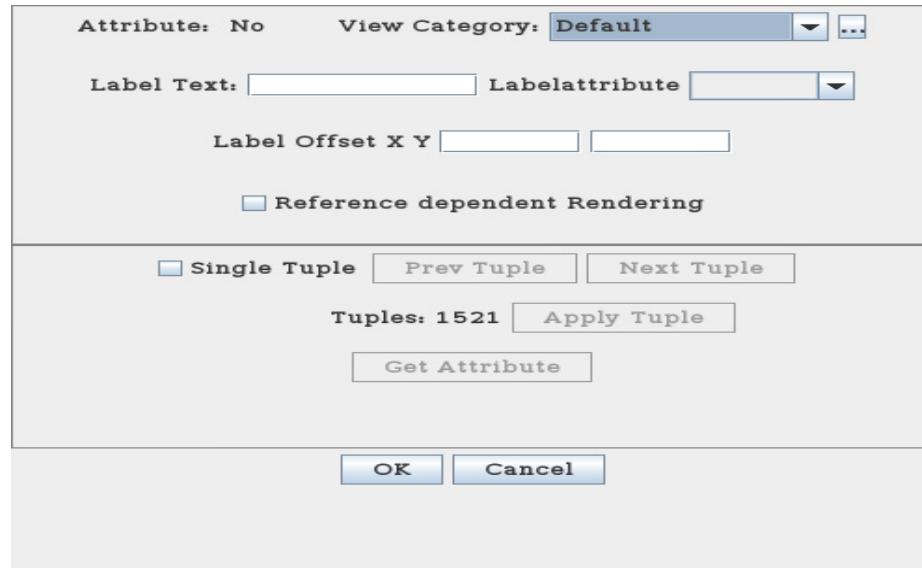
26. Display grid values of all tiles of N51E007Relation by command  
query N51E007Relation feed extend[G : getgrid(.No)] consume.

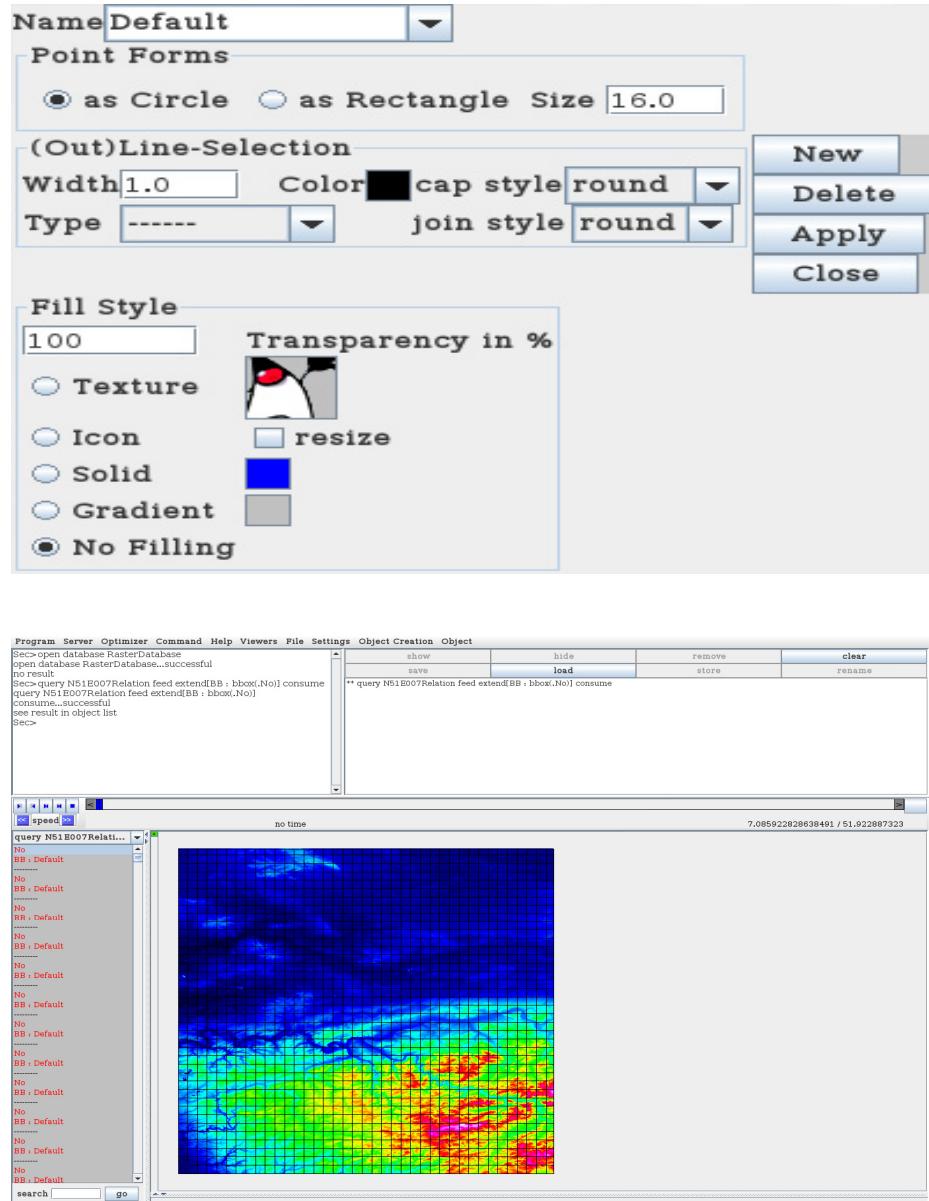


27. Display tile of N51E007Relation that contains a defined value at location x = 7.5 and y = 51.5 and display this value by command query N51E007Relation feed filter[isdefined(.No atlocation [[const point value(7.5 51.5)]])] consume feed extend[AL : .No atlocation [[const point value(7.5 51.5)]]] consume.



28. Display all tiles of N51E007Relation and corresponding bounding boxes by command query N51E007Relation feed extend[BB : bbox(.No)] consume.





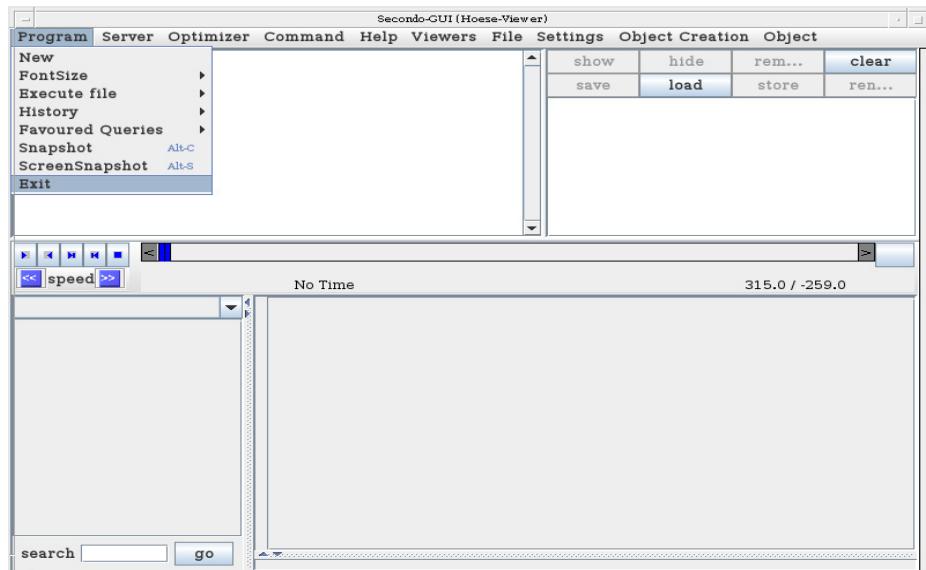
29. To get more information about the SECONDO system and available algebras, types and operations, execute the following commands at the command interface in the SECONDO GUI:

```
list algebras
list algebra Raster2Algebra
list algebra TileAlgebra
list type constructors
list operators
```

It is also possible to find a particular operator, for example, operator atlocation by command

```
query SEC2OPERATORINFO feed filter[.Name contains "atlocation"]
consume
```

30. Terminate SECONDO GUI by main menu command Program -> Exit,  
Secondo optimizer by command quit in the optimizer shell window and  
Secondo Monitor by command quit followed by yes in the SECONDO Monitor  
shell window.



The screenshot shows a terminal window titled "Optimizer: bash". It displays a Java stack trace:

```
Datei Bearbeiten Ansicht Leerzeichen Einstellungen Hilfe
WARNING: Maximum stack size for global stack is 128 MB
execute query: seconde("close database")
Command failed with error code : 6
and error message :
Secondo: No database open.

Secondo: No database open.
no solution found for' seconde('close database')/1

opt-server > connection ended normally

opt-server >
bye client
number of clients is :0

opt-server > quit
execute query: halt( )
#
# A fatal error has been detected by the Java Runtime Environment:
#
# SIGSEGV (0xb) at pc=0xb7790869, pid=6747, tid=306201Z784
#
# JRE version: 6.0_27-b27
# Java VM: OpenJDK Server VM (20.0-b12 mixed mode linux-x86 )
# Derivative: IcedTea6 1.12.4
# Distribution: Dummy Product (i586), package suse-32.1-i386
# Problematic frame:
# C [libpthread.so.0+0xb869] pthread_kill+0x9
#
# An error report file with more information is saved as:
# /home/seconde/Secondo/Optimizer/hs_err_pid6747.log
#
# If you would like to submit a bug report, please include
# instructions how to reproduce the bug and visit:
# http://icedtea.classpath.org/bugzilla
# The crash happened outside the Java Virtual Machine in native code.
# See problematic frame for where to report the bug.
#
#/StartOptServer: Zeile 113: 6747 Abgebrochen      $cmd
seconde@linux-e6ch:~/Secondo/Optimizer>
```

```
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
Launching Checkpoint service ... Starting Process:
Program: SecondoCheckpoint
Args: "/home/secondo/secondo/bin/SecondoConfig.ini"
completed.
Launching Secondo Registrar ... Starting Process:
Program: SecondoRegistrar
Args: "/home/secondo/secondo/bin/SecondoConfig.ini"
completed.

Secondo Monitor ready for operation.
Type 'HELP' to get a list of available commands.
Startup in progress ... Starting Process:
Program: SecondoListener
Args: "/home/secondo/secondo/bin/SecondoConfig.ini"
completed.
SEC_MON> Starting Process:
Program: SecondoEDB
Args: "-srw" "/home/secondo/secondo/bin/SecondoConfig.ini" --socket=5
Redirecting server output to file SecondoServer.msg
Starting Process:
Program: SecondoRDR
Args: "-srw" "/home/secondo/secondo/bin/SecondoConfig.ini" --socket=5
Redirecting server output to file SecondoServer.msg
quit
Really shutdown the system and quit (confirm with 'y' or 'yes')?
SEC_MON> yes
*** Signal SIGTERM (15) caught! Calling default signal handler ...
Shutdown in progress ... completed.
Secondo Listener terminated with return code 0.

Terminating Secondo Monitor ...
*** Signal SIGTERM (15) caught! Calling default signal handler ...
Terminating Secondo Registrar ... completed.
Secondo Registrar terminated with return code 0.
Terminating Checkpoint Service ... completed.
Checkpoint service terminated with return code 256.
SecondoMonitor terminated.
secondo@linux:~/Secondo/bin>
```

## 3 Reference Manual

### 3.1 Tile algebra data types

The following chapter describes all Tile algebra data types.  
These data types can be classified into four data type categories:

grid data types  
spatial data types  
moving spatial data types  
instant spatial data types

#### 3.1.1 Grid data types

Tile algebra contains the following grid data types:

tgrid  
mtgrid

Data type tgrid represents a 2-dimensional grid definition with x origin, y origin and the length of a grid cell.

Data type mtgrid represents a 3-dimensional grid definition with x origin, y origin, length of a grid cell and a duration value for the time dimension.

Operator toraster2 is executable on grid data types.

### 3.1.2 Spatial data types

Spatial data types contain values characterized by a x-coordinate and a y-coordinate. Square size of x-dimension and y-dimension of a spatial data types is restricted by the size of a memory page.

Tile algebra contains the following spatial data types:

tint  
treal  
tbool  
tstring

Data type tint represents a tile of integer values of size 31 x 31 which are stored in a Flob and contains a tgrid object as 2-dimensional grid definition, a minimum value and a maximum value.

Data type treal represents a tile of double values of size 22 x 22 which are stored in a Flob and contains a tgrid object as 2-dimensional grid definition, a minimum value and a maximum value.

Data type tbool represents a tile of char values of size 63 x 63 which are stored in a Flob and contains a tgrid object as 2-dimensional grid definition, a minimum value and a maximum value.

Data type tstring represents a tile of string values of size 31 x 31 which are stored optimized as unique strings in a Flob and contains a tgrid object as 2-dimensional grid definition, a minimum value and a maximum value.

Operators atlocation, atrange, bbox, CELL1, CELL2, CELLS, getgrid, matchgrid, minimum, maximum, map, map2 and t2mt are executable on spatial data types.

### 3.1.3 Moving spatial data types

Moving spatial data types contain values characterized by a x-coordinate, a y-coordinate and a time-coordinate. Square size of x-dimension and y-dimension of a moving spatial data types is restricted by the size of a memory page. Size of time-dimension of a moving spatial data type has the constant value of 10.

Tile algebra contains the following moving spatial data types:

mtint  
mtreal  
mtbool  
mtstring

Data type mtint represents a tile of integer values of size 31 x 31 x 10 which are stored in a Flob and contains a mtgrid object as 3-dimensional grid definition, a minimum value and a maximum value.

Data type mtreal represents a tile of double values of size 22 x 22 x 10 which are stored in a Flob and contains a mtgrid object as 3-dimensional grid definition, a minimum value and a maximum value.

Data type mtbool represents a tile of char values of size 63 x 63 x 10 which are stored in a Flob and contains a mtgrid object as 3-dimensional grid definition, a minimum value and a maximum value.

Data type mtstring represents a tile of string values of size 31 x 31 x 10 which are stored optimized as unique strings in a Flob and contains a mtgrid object as 3-dimensional grid definition, a minimum value and a maximum value.

Operators atinstant, atlocation, atperiods, atrange, bbox, CELL1, CELL2, CELLS, deftime, getgrid, matchgrid, minimum, maximum, map and map2 are executable on moving spatial data types.

### 3.1.4 Instant spatial data types

Instant spatial data types contain values characterized by a x-coordinate and a y-coordinate at corresponding instant value. Square size of x-dimension and y-dimension of an instant spatial data types is restricted by the size of a memory page.

Tile algebra contains the following instant spatial data types:

itint  
itreal  
itbool  
itstring

Data type itint represents a tile of integer values of size 31 x 31 which are stored in a Flob and contains a tgrid object as 2-dimensional grid definition, a minimum value, a maximum value and a corresponding instant value.

Data type itreal represents a tile of double values of size 22 x 22 which are stored in a Flob and contains a tgrid object as 2-dimensional grid definition, a minimum value, a maximum value and a corresponding instant value.

Data type itbool represents a tile of char values of size 63 x 63 which are stored in a Flob and contains a tgrid object as 2-dimensional grid definition, a minimum value, a maximum value and a corresponding instant value.

Data type itstring represents a tile of string values of size 31 x 31 which are stored optimized as unique strings in a Flob and contains a tgrid object as 2-dimensional grid definition, a minimum value, a maximum value and a corresponding instant value.

Operators inst and val are executable on instant spatial data types.

## 3.2 Tile algebra operators

The following chapter describes all Tile algebra operators.

### 3.2.1 atinstant

Operator atinstant returns the values of a moving spatial data type object for an instant time point in the form of an instant spatial data type object.

syntax:

\_ atinstant \_

signatures:

mtint × instant → itint  
mtreal × instant → itreal  
mtbool × instant → itbool  
mtstring × instant → itstring

example:

```
query [const mtint value ((0.0 0.0 1.0 2.0) (2 2 2) (0 0 0 (0 1 2 3 4 5 6 7)))] atinstant
create_instant(2.0);
```

### 3.2.2 atlocation

Operator atlocation returns the value at a location point of a spatial data type object or the value(s) at a location point of a moving spatial data type object.

syntax:

\_ atlocation [\_, \_]

signatures:

tint × point → int  
treal × point → real  
tbool × point → bool  
tstring × point → string  
mtint × point → mint  
mtreal × point → mreal  
mtbool × point → mbool  
mtstring × point → mstring  
mtint × point × instant → int  
mtreal × point × instant → real  
mtbool × point × instant → bool  
mtstring × point × instant → string

examples:

```
query [const tint value ((0.0 0.0 1.0) (2 2) (0 0 (0 1 2 3)) (29 29 (4 5 6 7)))] atlocation [[const point value (1.5 1.5)]];
```

```
query [const mtint value ((0.0 0.0 1.0 1.0) (2 2 2) (0 0 0 (0 1 2 3 4 5 6 7)))] atlocation [[const point value (1.5 1.5)]];
```

```
query [const mtint value ((0.0 0.0 1.0 1.0) (2 2 1) (0 0 0 (0 1 2 3)) (29 29 9 (4 5 6 7)))]  
atlocation [[const point value (1.5 1.5)] , [const instant value "2000-01-03-00:00:00.001"]];
```

### 3.2.3 atperiods

Operator atperiods restricts the values of a moving spatial data type object to given periods.

syntax:

atperiods(\_)

signatures:

mtint × periods → mtint  
mtreal × periods → mtreal  
mtbool × periods → mtbool  
mtstring × periods → mtstring

example:

```
query [const mtint value ((2.0 2.0 1.0 2.0) (2 2 1) (0 0 0 (0 1 2 3)) (29 29 9 (4 5 6 7)))]  
atperiods [const periods value (("2000-01-03" "2000-01-05" TRUE FALSE))];
```

### 3.2.4 atrange

Operator atrange returns the values at range of a rectangle of a spatial data type object, the values at range of a rectangle of a moving spatial data type object or the values at range of a rectangle, a begin instant value and an end instant value of a moving spatial data type object.

syntax:

\_ atrange [\_,\_,\_]

signatures:

tint × rect → tint  
treal × rect → treal  
tbool × rect → tbool  
tstring × rect → tstring  
mtint × rect → mtint  
mtreal × rect → mtreal  
mtbool × rect → mtbool  
mtstring × rect → mtstring  
mtint × rect × instant × instant → mtint  
mtreal × rect × instant × instant → mtreal  
mtbool × rect × instant × instant → mtbool  
mtstring × rect × instant × instant → mtstring

examples:

```
query [const tint value ((2.0 2.0 1.0) (2 2) (0 0 (0 1 2 3)) (29 29 (4 5 6 7)))] atrange [[const rect value (3.0 31.0 3.0 31.0)]];
```

```
query [const mtint value ((2.0 2.0 1.0 1.0) (2 2 1) (0 0 0 (0 1 2 3)) (29 29 9 (4 5 6 7)))] atrange [[const rect value (3.0 31.0 3.0 31.0)]];
```

```
query [const mtint value ((2.0 2.0 1.0 1.0) (2 2 1) (0 0 0 (0 1 2 3)) (29 29 9 (4 5 6 7)))] atrange [[const rect value (2.5 31.5 2.5 31.5)], [const instant value "2000-01-03"], [const instant value "2000-01-04"]];
```

### 3.2.5 bbox

Operator bbox returns the bounding box rectangle of a spatial data type object or the bounding box rectangle of a moving spatial data type object.

syntax:

`bbox(_)`

signatures:

`tint → rect`  
`treal → rect`  
`tbool → rect`  
`tstring → rect`  
`mtint → rect3`  
`mtreal → rect3`  
`mtbool → rect3`  
`mtstring → rect3`

examples:

`query bbox([const tint value ((1.1 2.2 1.0) (2 2) (0 0 (0 1 2 3)) (28 28 (4 5 6 7)))]);`

`query bbox([const mtint value ((1.0 1.0 1.0 1.0) (2 2 1) (0 0 0 (0 1 2 3)) (8 8 9 (4 5 6 7)))]);`

### 3.2.6 CELL1

Operator CELL1 is a type mapping operator.

signatures:

tint × ... → int  
treal × ... → real  
tbool × ... → bool  
tstring × ... → string  
mtint × ... → int  
mtreal × ... → real  
mtbool × ... → bool  
mtstring × ... → string

example:

```
query [const tint value ((0.0 0.0 1.0) (1 1) (0 0 (1)))] map [fun(n: CELL1) n * 2];
```

### 3.2.7 CELL2

Operator CELL2 is a type mapping operator.

signatures:

tint × tint × ... → int  
tint × treal × ... → real  
tint × tbool × ... → bool  
tint × tstring × ... → string  
tint × mtint × ... → int  
tint × mtreal × ... → real  
tint × mtbool × ... → bool  
tint × mtstring × ... → string  
treal × tint × ... → int  
treal × treal × ... → real  
treal × tbool × ... → bool  
treal × tstring × ... → string  
treal × mtint × ... → int  
treal × mtreal × ... → real  
treal × mtbool × ... → bool  
treal × mtstring × ... → string  
tbool × tint × ... → int  
tbool × treal × ... → real  
tbool × tbool × ... → bool  
tbool × tstring × ... → string  
tbool × mtint × ... → int  
tbool × mtreal × ... → real  
tbool × mtbool × ... → bool  
tbool × mtstring × ... → string  
tstring × tint × ... → int  
tstring × treal × ... → real  
tstring × tbool × ... → bool  
tstring × tstring × ... → string  
tstring × mtint × ... → int  
tstring × mtreal × ... → real  
tstring × mtbool × ... → bool  
tstring × mtstring × ... → string

mtint × tint × ... → int  
mtint × treal × ... → real  
mtint × tbool × ... → bool  
mtint × tstring × ... → string  
mtint × mtint × ... → int  
mtint × mtreal × ... → real  
mtint × mtbool × ... → bool  
mtint × mtstring × ... → string  
mtreal × tint × ... → int  
mtreal × treal × ... → real  
mtreal × tbool × ... → bool  
mtreal × tstring × ... → string  
mtreal × mtint × ... → int  
mtreal × mtreal × ... → real  
mtreal × mtbool × ... → bool  
mtreal × mtstring × ... → string  
mtbool × tint × ... → int  
mtbool × treal × ... → real  
mtbool × tbool × ... → bool  
mtbool × tstring × ... → string  
mtbool × mtint × ... → int  
mtbool × mtreal × ... → real  
mtbool × mtbool × ... → bool  
mtbool × mtstring × ... → string  
mtstring × tint × ... → int  
mtstring × treal × ... → real  
mtstring × tbool × ... → bool  
mtstring × tstring × ... → string  
mtstring × mtint × ... → int  
mtstring × mtreal × ... → real  
mtstring × mtbool × ... → bool  
mtstring × mtstring × ... → string

example:

```
query [const tint value ((0.0 0.0 1.0) (1 1) (0 0 (1)))] [const tint value ((0.0 0.0 1.0) (1 1) (0 0 (1)))] map2 [fun(n: int, m: CELL2) n + m];
```

### 3.2.8 CELLS

Operator CELLS is a type mapping operator.

signatures:

```
tint × ... → rel(tuple([Elem : int]))  
treal × ... → rel(tuple([Elem : real]))  
tbool × ... → rel(tuple([Elem : bool]))  
tstring × ... → rel(tuple([Elem : string]))  
mtint × ... → rel(tuple([Elem : int]))  
mtreal × ... → rel(tuple([Elem : real]))  
mtbool × ... → rel(tuple([Elem : bool]))  
mtstring × ... → rel(tuple([Elem : string]))
```

example:

```
query [const tint value ((0.0 0.0 1.0) (1 1) (0 0 (1)))] matchgrid[[const tgrid value (0.0 0.0  
0.5)], fun(c: CELLS) c feed avg[Elem], FALSE];
```

### 3.2.9 compose

Operator compose merges a mpoint object and a spatial data type object into a moving data type object.

syntax:

\_ compose \_

signatures:

mpoint × tint → mint  
mpoint × treal → mreal  
mpoint × tbool → mbool  
mpoint × tstring → mstring  
mpoint × mtint → mint  
mpoint × mtreal → mreal  
mpoint × mtbool → mbool  
mpoint × mtstring → mstring

example:

```
query [const mpoint value (((2000-01-03-00:00:00.000 "2000-01-05-00:00:00.000" TRUE  
FALSE) (0.5 0.5 1.5 1.5)) ((2000-01-05-00:00:00.000 "2000-01-07-00:00:00.000" TRUE  
FALSE) (10.5 10.5 10.0 10.0)) ((2000-01-07-00:00:00.000 "2000-01-09-00:00:00.000"  
TRUE FALSE) (1.0 1.0 0.5 0.5))] compose [const tint value ((0.0 0.0 1.0) (2 2) (0 0 (0 1 2  
3)) (2 2 (4 5 6 7)))];
```

### 3.2.10 deftime

Operator deftime returns the defined periods of a moving spatial data type object.

syntax:

deftime(\_)

signatures:

mtint → periods

mtreal → periods

mtbool → periods

mtstring → periods

example:

```
query deftime([const mtint value ((0.0 0.0 1.0 1.0) (1 1 1) (0 0 0 (1)) (1 1 1 (1)))]);
```

### **3.2.11 fromline**

Operator fromline creates a stream of tbool objects from a line object and a tgrid object.

syntax:

fromline( \_, \_ )

signature:

line × tgrid → stream(tbool)

example:

```
query fromline([const line value ((64.0 14.2 284 44.2))], [const tgrid value (10.0 10.0 2.0)])
count;
```

### **3.2.12 fromregion**

Operator fromregion creates a stream of tbool objects from a region object and a tgrid object.

syntax:

fromregion( \_, \_ )

signature:

region × tgrid → stream(tbool)

example:

```
query fromregion([const region value (((0.0 1.0) (1.0 0.0) (2.0 0.0) (3.0 1.0) (3.0 2.0) (2.0  
3.0) (1.0 3.0) (0.0 2.0) (0.0 1.0)))), [const tgrid value (0.0 0.0 1.0)]]) count
```

### 3.2.13 getgrid

Operator getgrid returns the tgrid object of a spatial data type object or the mtgrid object of a moving spatial data type object.

syntax:

```
getgrid(_)
```

signatures:

tint → tgrid
treal → tgrid
tbool → tgrid
tstring → tgrid
mtint → mtgrid
mtreal → mtgrid
mtbool → mtgrid
mtstring → mtgrid

examples:

```
query getgrid([const tint value ((0.0 1.0 2.0) (2 2) (0 0 (0 1 2 3)) (29 29 (4 5 6 7)))]);
```

```
query getgrid([const mtint value ((0.0 1.0 2.0 3.0) (2 2 1) (0 0 0 (0 1 2 3)) (29 29 9 (4 5 6 7)))]);
```

### 3.2.14 inst

Operator inst returns the instant value of an instant spatial data type object.

syntax:

inst(\_)

signatures:

itint → instant

itreal → instant

itbool → instant

itstring → instant

example:

```
query inst([const itint value ((instant "2013-05-13") ((0.0 0.0 1.0) (2 2) (0 0 (0 1 2 3)) (29 29  
(4 5 6 7))))]);
```

### 3.2.15 map

Operator map maps a spatial data type object to another spatial data type object or a moving spatial data type object to another moving spatial data type object.

syntax:

\_ map[\_]

signatures:

tint × (int → int) → tint  
tint × (int → real) → treal  
tint × (int → bool) → tbool  
tint × (int → string) → tstring  
treal × (real → int) → tint  
treal × (real → real) → treal  
treal × (real → bool) → tbool  
treal × (real → string) → tstring  
tbool × (bool → int) → tint  
tbool × (bool → real) → treal  
tbool × (bool → bool) → tbool  
tbool × (bool → string) → tstring  
tstring × (string → int) → tint  
tstring × (string → real) → treal  
tstring × (string → bool) → tbool  
tstring × (string → string) → tstring  
mtint × (int → int) → mtint  
mtint × (int → real) → mtreal  
mtint × (int → bool) → mtbool  
mtint × (int → string) → mtstring  
mtreal × (real → int) → mtint  
mtreal × (real → real) → mtreal  
mtreal × (real → bool) → mtbool  
mtreal × (real → string) → mtstring  
mtbool × (bool → int) → mtint  
mtbool × (bool → real) → mtreal  
mtbool × (bool → bool) → mtbool  
mtbool × (bool → string) → mtstring  
mtstring × (string → int) → mtint  
mtstring × (string → real) → mtreal  
mtstring × (string → bool) → mtbool  
mtstring × (string → string) → mtstring

example:

```
query [const tint value ((0.0 0.0 1.0) (1 1) (0 0 (1)))] map [. * 2];
```

### 3.2.16 map2

Operator map2 combines a spatial data type object and a spatial data type object to a new spatial data type object, a spatial data type object and a moving spatial data type object to a new moving spatial data type object, a moving spatial data type object and a spatial data type object to a new moving spatial data type object or a moving spatial data type object and a moving spatial data type object to a new moving spatial data type object.

syntax:

\_ \_ map2[\_]

signatures:

tint × tint × (int × int → int) → tint  
tint × tint × (int × int → real) → treal  
tint × tint × (int × int → bool) → tbool  
tint × tint × (int × int → string) → tstring  
tint × treal × (int × real → int) → tint  
tint × treal × (int × real → real) → treal  
tint × treal × (int × real → bool) → tbool  
tint × treal × (int × real → string) → tstring  
tint × tbool × (int × bool → int) → tint  
tint × tbool × (int × bool → real) → treal  
tint × tbool × (int × bool → bool) → tbool  
tint × tbool × (int × bool → string) → tstring  
tint × tstring × (int × string → int) → tint  
tint × tstring × (int × string → real) → treal  
tint × tstring × (int × string → bool) → tbool  
tint × tstring × (int × string → string) → tstring  
treal × tint × (real × int → int) → tint  
treal × tint × (real × int → real) → treal  
treal × tint × (real × int → bool) → tbool  
treal × tint × (real × int → string) → tstring  
treal × treal × (real × real → int) → tint  
treal × treal × (real × real → real) → treal  
treal × treal × (real × real → bool) → tbool  
treal × treal × (real × real → string) → tstring  
treal × tbool × (real × bool → int) → tint  
treal × tbool × (real × bool → real) → treal  
treal × tbool × (real × bool → bool) → tbool  
treal × tbool × (real × bool → string) → tstring  
treal × tstring × (real × string → int) → tint  
treal × tstring × (real × string → real) → treal  
treal × tstring × (real × string → bool) → tbool  
treal × tstring × (real × string → string) → tstring



treal × mtint × (real × int → int) → mtint  
treal × mtint × (real × int → real) → mtreal  
treal × mtint × (real × int → bool) → mtbool  
treal × mtint × (real × int → string) → mtstring  
treal × mtreal × (real × real → int) → mtint  
treal × mtreal × (real × real → real) → mtreal  
treal × mtreal × (real × real → bool) → mtbool  
treal × mtreal × (real × real → string) → mtstring  
treal × mtbool × (real × bool → int) → mtint  
treal × mtbool × (real × bool → real) → mtreal  
treal × mtbool × (real × bool → bool) → mtbool  
treal × mtbool × (real × bool → string) → mtstring  
treal × mtstring × (real × string → int) → mtint  
treal × mtstring × (real × string → real) → mtreal  
treal × mtstring × (real × string → bool) → mtbool  
treal × mtstring × (real × string → string) → mtstring  
tbool × mtint × (bool × int → int) → mtint  
tbool × mtint × (bool × int → real) → mtreal  
tbool × mtint × (bool × int → bool) → mtbool  
tbool × mtint × (bool × int → string) → mtstring  
tbool × mtreal × (bool × real → int) → mtint  
tbool × mtreal × (bool × real → real) → mtreal  
tbool × mtreal × (bool × real → bool) → mtbool  
tbool × mtreal × (bool × real → string) → mtstring  
tbool × mtbool × (bool × bool → int) → mtint  
tbool × mtbool × (bool × bool → real) → mtreal  
tbool × mtbool × (bool × bool → bool) → mtbool  
tbool × mtbool × (bool × bool → string) → mtstring  
tbool × mtstring × (bool × string → int) → mtint  
tbool × mtstring × (bool × string → real) → mtreal  
tbool × mtstring × (bool × string → bool) → mtbool  
tbool × mtstring × (bool × string → string) → mtstring  
tstring × mtint × (string × int → int) → mtint  
tstring × mtint × (string × int → real) → mtreal  
tstring × mtint × (string × int → bool) → mtbool  
tstring × mtint × (string × int → string) → mtstring  
tstring × mtreal × (string × real → int) → mtint  
tstring × mtreal × (string × real → real) → mtreal  
tstring × mtreal × (string × real → bool) → mtbool  
tstring × mtreal × (string × real → string) → mtstring  
tstring × mtbool × (string × bool → int) → mtint  
tstring × mtbool × (string × bool → real) → mtreal  
tstring × mtbool × (string × bool → bool) → mtbool  
tstring × mtbool × (string × bool → string) → mtstring  
tstring × mtstring × (string × string → int) → mtint  
tstring × mtstring × (string × string → real) → mtreal  
tstring × mtstring × (string × string → bool) → mtbool  
tstring × mtstring × (string × string → string) → mtstring

mtint × tint × (int × int → int) → mtint  
mtint × tint × (int × int → real) → mtreal  
mtint × tint × (int × int → bool) → mtbool  
mtint × tint × (int × int → string) → mtstring  
mtint × treal × (int × real → int) → mtint  
mtint × treal × (int × real → real) → mtreal  
mtint × treal × (int × real → bool) → mtbool  
mtint × treal × (int × real → string) → mtstring  
mtint × tbool × (int × bool → int) → mtint  
mtint × tbool × (int × bool → real) → mtreal  
mtint × tbool × (int × bool → bool) → mtbool  
mtint × tbool × (int × bool → string) → mtstring  
mtint × tstring × (int × string → int) → mtint  
mtint × tstring × (int × string → real) → mtreal  
mtint × tstring × (int × string → bool) → mtbool  
mtint × tstring × (int × string → string) → mtstring  
mtreal × tint × (real × int → int) → mtint  
mtreal × tint × (real × int → real) → mtreal  
mtreal × tint × (real × int → bool) → mtbool  
mtreal × tint × (real × int → string) → mtstring  
mtreal × treal × (real × real → int) → mtint  
mtreal × treal × (real × real → real) → mtreal  
mtreal × treal × (real × real → bool) → mtbool  
mtreal × treal × (real × real → string) → mtstring  
mtreal × tbool × (real × bool → int) → mtint  
mtreal × tbool × (real × bool → real) → mtreal  
mtreal × tbool × (real × bool → bool) → mtbool  
mtreal × tbool × (real × bool → string) → mtstring  
mtreal × tstring × (real × string → int) → mtint  
mtreal × tstring × (real × string → real) → mtreal  
mtreal × tstring × (real × string → bool) → mtbool  
mtreal × tstring × (real × string → string) → mtstring  
mtbool × tint × (bool × int → int) → mtint  
mtbool × tint × (bool × int → real) → mtreal  
mtbool × tint × (bool × int → bool) → mtbool  
mtbool × tint × (bool × int → string) → mtstring  
mtbool × treal × (bool × real → int) → mtint  
mtbool × treal × (bool × real → real) → mtreal  
mtbool × treal × (bool × real → bool) → mtbool  
mtbool × treal × (bool × real → string) → mtstring  
mtbool × tbool × (bool × bool → int) → mtint  
mtbool × tbool × (bool × bool → real) → mtreal  
mtbool × tbool × (bool × bool → bool) → mtbool  
mtbool × tbool × (bool × bool → string) → mtstring  
mtbool × tstring × (bool × string → int) → mtint  
mtbool × tstring × (bool × string → real) → mtreal  
mtbool × tstring × (bool × string → bool) → mtbool  
mtbool × tstring × (bool × string → string) → mtstring

mtstring × tint × (string × int → int) → mtint  
mtstring × tint × (string × int → real) → mtreal  
mtstring × tint × (string × int → bool) → mtbool  
mtstring × tint × (string × int → string) → mtstring  
mtstring × treal × (string × real → int) → mtint  
mtstring × treal × (string × real → real) → mtreal  
mtstring × treal × (string × real → bool) → mtbool  
mtstring × treal × (string × real → string) → mtstring  
mtstring × tbool × (string × bool → int) → mtint  
mtstring × tbool × (string × bool → real) → mtreal  
mtstring × tbool × (string × bool → bool) → mtbool  
mtstring × tbool × (string × bool → string) → mtstring  
mtstring × tstring × (string × string → int) → mtint  
mtstring × tstring × (string × string → real) → mtreal  
mtstring × tstring × (string × string → bool) → mtbool  
mtstring × tstring × (string × string → string) → mtstring  
mtint × mtint × (int × int → int) → mtint  
mtint × mtint × (int × int → real) → mtreal  
mtint × mtint × (int × int → bool) → mtbool  
mtint × mtint × (int × int → string) → mtstring  
mtint × mtreal × (int × real → int) → mtint  
mtint × mtreal × (int × real → real) → mtreal  
mtint × mtreal × (int × real → bool) → mtbool  
mtint × mtreal × (int × real → string) → mtstring  
mtint × mtbool × (int × bool → int) → mtint  
mtint × mtbool × (int × bool → real) → mtreal  
mtint × mtbool × (int × bool → bool) → mtbool  
mtint × mtbool × (int × bool → string) → mtstring  
mtint × mtstring × (int × string → int) → mtint  
mtint × mtstring × (int × string → real) → mtreal  
mtint × mtstring × (int × string → bool) → mtbool  
mtint × mtstring × (int × string → string) → mtstring  
mtreal × mtint × (real × int → int) → mtint  
mtreal × mtint × (real × int → real) → mtreal  
mtreal × mtint × (real × int → bool) → mtbool  
mtreal × mtint × (real × int → string) → mtstring  
mtreal × mtreal × (real × real → int) → mtint  
mtreal × mtreal × (real × real → real) → mtreal  
mtreal × mtreal × (real × real → bool) → mtbool  
mtreal × mtreal × (real × real → string) → mtstring  
mtreal × mtbool × (real × bool → int) → mtint  
mtreal × mtbool × (real × bool → real) → mtreal  
mtreal × mtbool × (real × bool → bool) → mtbool  
mtreal × mtbool × (real × bool → string) → mtstring  
mtreal × mtstring × (real × string → int) → mtint  
mtreal × mtstring × (real × string → real) → mtreal  
mtreal × mtstring × (real × string → bool) → mtbool  
mtreal × mtstring × (real × string → string) → mtstring

mtbool × mtint × (bool × int → int) → mtint  
mtbool × mtint × (bool × int → real) → mtreal  
mtbool × mtint × (bool × int → bool) → mtbool  
mtbool × mtint × (bool × int → string) → mtstring  
mtbool × mtreal × (bool × real → int) → mtint  
mtbool × mtreal × (bool × real → real) → mtreal  
mtbool × mtreal × (bool × real → bool) → mtbool  
mtbool × mtreal × (bool × real → string) → mtstring  
mtbool × mtbool × (bool × bool → int) → mtint  
mtbool × mtbool × (bool × bool → real) → mtreal  
mtbool × mtbool × (bool × bool → bool) → mtbool  
mtbool × mtbool × (bool × bool → string) → mtstring  
mtbool × mtstring × (bool × string → int) → mtint  
mtbool × mtstring × (bool × string → real) → mtreal  
mtbool × mtstring × (bool × string → bool) → mtbool  
mtbool × mtstring × (bool × string → string) → mtstring  
mtstring × mtint × (string × int → int) → mtint  
mtstring × mtint × (string × int → real) → mtreal  
mtstring × mtint × (string × int → bool) → mtbool  
mtstring × mtint × (string × int → string) → mtstring  
mtstring × mtreal × (string × real → int) → mtint  
mtstring × mtreal × (string × real → real) → mtreal  
mtstring × mtreal × (string × real → bool) → mtbool  
mtstring × mtreal × (string × real → string) → mtstring  
mtstring × mtbool × (string × bool → int) → mtint  
mtstring × mtbool × (string × bool → real) → mtreal  
mtstring × mtbool × (string × bool → bool) → mtbool  
mtstring × mtbool × (string × bool → string) → mtstring  
mtstring × mtstring × (string × string → int) → mtint  
mtstring × mtstring × (string × string → real) → mtreal  
mtstring × mtstring × (string × string → bool) → mtbool  
mtstring × mtstring × (string × string → string) → mtstring

examples:

query [const tint value ((0.0 0.0 1.0) (1 1) (0 0 (1)))] [const tint value ((0.0 0.0 1.0) (1 1) (0 0 (1)))] map2 [fun(n: int, m: int) n + m];

query [const tint value ((0.0 0.0 1.0) (1 1) (0 0 (1)))] [const mtint value ((0.0 0.0 1.0 1.0) (1 1) (0 0 0 (1)) (0 0 9 (1)))] map2 [fun(n: int, m: int) n + m];

query [const mtint value ((0.0 0.0 1.0 1.0) (1 1 1) (0 0 0 (1)) (0 0 9 (1)))] [const tint value ((0.0 0.0 1.0) (1 1) (0 0 (1)))] map2 [fun(n: int, m: int) n + m];

query [const mtint value ((0.0 0.0 1.0 1.0) (1 1 1) (0 0 0 (1)) (0 0 9 (1)))] [const mtint value ((0.0 0.0 1.0 1.0) (1 1 1) (0 0 0 (1)) (0 0 9 (1)))] map2 [fun(n: int, m: int) n + m];

### 3.2.17 matchgrid

Operator matchgrid applies a user function to a spatial data type object or a moving spatial data type object.

syntax:

\_ matchgrid [\_, \_, \_]

signatures:

tint × tgrid × (rel(tuple([Elem : int])) → int) × bool → tint  
tint × tgrid × (rel(tuple([Elem : int])) → real) × bool → treal  
tint × tgrid × (rel(tuple([Elem : int])) → bool) × bool → tbool  
tint × tgrid × (rel(tuple([Elem : int])) → string) × bool → tstring  
treal × tgrid × (rel(tuple([Elem : real])) → int) × bool → tint  
treal × tgrid × (rel(tuple([Elem : real])) → real) × bool → treal  
treal × tgrid × (rel(tuple([Elem : real])) → bool) × bool → tbool  
treal × tgrid × (rel(tuple([Elem : real])) → string) × bool → tstring  
tbool × tgrid × (rel(tuple([Elem : bool])) → int) × bool → tint  
tbool × tgrid × (rel(tuple([Elem : bool])) → real) × bool → treal  
tbool × tgrid × (rel(tuple([Elem : bool])) → bool) × bool → tbool  
tbool × tgrid × (rel(tuple([Elem : bool])) → string) × bool → tstring  
tstring × tgrid × (rel(tuple([Elem : string])) → int) × bool → tint  
tstring × tgrid × (rel(tuple([Elem : string])) → real) × bool → treal  
tstring × tgrid × (rel(tuple([Elem : string])) → bool) × bool → tbool  
tstring × tgrid × (rel(tuple([Elem : string])) → string) × bool → tstring  
mtint × mtgrid × (rel(tuple([Elem : int])) → int) × bool → mtint  
mtint × mtgrid × (rel(tuple([Elem : int])) → real) × bool → mtreal  
mtint × mtgrid × (rel(tuple([Elem : int])) → bool) × bool → mtbool  
mtint × mtgrid × (rel(tuple([Elem : int])) → string) × bool → mtstring  
mtreal × mtgrid × (rel(tuple([Elem : real])) → int) × bool → mtint  
mtreal × mtgrid × (rel(tuple([Elem : real])) → real) × bool → mtreal  
mtreal × mtgrid × (rel(tuple([Elem : real])) → bool) × bool → mtbool  
mtreal × mtgrid × (rel(tuple([Elem : real])) → string) × bool → mtstring  
mtbool × mtgrid × (rel(tuple([Elem : bool])) → int) × bool → mtint  
mtbool × mtgrid × (rel(tuple([Elem : bool])) → real) × bool → mtreal  
mtbool × mtgrid × (rel(tuple([Elem : bool])) → bool) × bool → mtbool  
mtbool × mtgrid × (rel(tuple([Elem : bool])) → string) × bool → mtstring  
mtstring × mtgrid × (rel(tuple([Elem : string])) → int) × bool → mtint  
mtstring × mtgrid × (rel(tuple([Elem : string])) → real) × bool → mtreal  
mtstring × mtgrid × (rel(tuple([Elem : string])) → bool) × bool → mtbool  
mtstring × mtgrid × (rel(tuple([Elem : string])) → string) × bool → mtstring

examples:

```
query [const tint value ((0.0 0.0 1.0) (1 1) (0 0 (1)))] matchgrid[[const tgrid value (0.0 0.0 0.5)], fun(cell: rel(tuple([Elem : int]))) cell feed avg[Elem], FALSE];  
query [const mtint value ((0.0 0.0 1.0 2.0) (1 1 1) (0 0 0 (1)))] matchgrid[[const mtgrid value (0.0 0.0 0.5 (duration (1 0)))], fun(cell: rel(tuple([Elem : int]))) cell feed avg[Elem], FALSE];
```

### 3.2.18 maximum

Operator maximum returns the maximum value of a spatial data type object or a moving spatial data type object.

syntax:

maximum(\_)

signatures:

tint → int  
treal → real  
tbool → bool  
tstring → string  
mtint → int  
mtreal → real  
mtbool → bool  
mtstring → string

examples:

```
query maximum([const tint value ((0.0 0.0 1.0) (2 2) (0 0 (0 1 2 3)) (29 29 (4 5 6 7)))];  
query maximum([const mtint value ((0.0 0.0 1.0 1.0) (2 2 1) (0 0 0 (0 1 2 3)) (29 29 9 (4 5 6 7)))]);
```

### 3.2.19 minimum

Operator minimum returns the minimum value of a spatial data type object or a moving spatial data type object.

syntax:

```
minimum(_)
```

signatures:

```
tint → int  
treal → real  
tbool → bool  
tstring → string  
mtint → int  
mtreal → real  
mtbool → bool  
mtstring → string
```

examples:

```
query minimum([const tint value ((0.0 0.0 1.0) (2 2) (0 0 (0 1 2 3)) (29 29 (4 5 6 7)))]);
```

```
query minimum([const mtint value ((0.0 0.0 1.0 1.0) (2 2 1) (0 0 0 (0 1 2 3)) (29 29 9 (4 5 6 7)))]);
```

### 3.2.20 t2mt

Operator t2mt adds a time dimension to a spatial data type object and returns resulting moving spatial data type object.

syntax:

t2mt( \_, \_, \_, \_ )

signatures:

tint × duration × instant × instant → mtint  
treal × duration × instant × instant → mtreal  
tbool × duration × instant × instant → mtbool  
tstring × duration × instant × instant → mtstring

example:

```
query t2mt([const tint value ((0.0 0.0 1.0) (1 1) (0 0 (1)))], [const duration value (0 43200000)], [const instant value "2000-01-03"], [const instant value "2000-01-04"]);
```

### 3.2.21 tiles

Operator tiles converts a Raster2 algebra grid2 object to a Tile algebra tgrid object, a Raster2 algebra grid3 object to a Tile algebra mtgrid object, a Raster2 algebra spatial data type object to a stream of Tile algebra spatial data type objects, a Raster2 algebra moving spatial data type object to a stream of Tile algebra moving spatial data type objects or a Raster2 algebra instant spatial data type object to a stream of Tile algebra instant spatial data type objects.

syntax:

`tiles(_)`

signatures:

`grid2 → tgrid`  
`grid3 → mtgrid`  
`sint → stream(tint)`  
`sreal → stream(treal)`  
`sbool → stream(tbool)`  
`sstring → stream(tstring)`  
`msint → stream(mtint)`  
`msreal → stream(mtreal)`  
`msbool → stream(mtbool)`  
`msstring → stream(mtstring)`  
`isint → stream(itint)`  
`isreal → stream(itreal)`  
`isbool → stream(itbool)`  
`isstring → stream(itstring)`

examples:

```
query tiles([const grid2 value (1.2 3.4 5.6)]);  
query tiles([const grid3 value (1.2 3.4 5.6 (duration (7 0)))]);  
query tiles([const sint value ((0.0 0.0 1.0) (2 2) (0 0 (0 1 2 3)) (60 60 (4 5 6 7)))] count;  
query tiles([const msint value ((0.0 0.0 1.0 1.0) (2 2 1) (0 0 0 (0 1 2 3)) (60 60 9 (4 5 6 7)))] count;  
query tiles([const isint value ((instant "2013-05-13") ((0.0 0.0 1.0) (2 2) (0 0 (0 1 2 3)) (60 60 (4 5 6 7))))] count;
```

### 3.2.22 toraster2

Operator toraster2 converts a Tile algebra tgrid object to a Raster2 algebra grid2 object, a Tile algebra mtgrid object to a Raster2 algebra grid3 object, a stream of Tile algebra spatial data type objects to a Raster2 algebra spatial data type object, a stream of Tile algebra moving spatial data type objects to a Raster2 algebra moving spatial data type object or a stream of Tile algebra instant spatial data type objects to a Raster2 algebra instant spatial data type object.

syntax:

toraster2(\_)

signatures:

tgrid → grid2  
mtgrid → grid3  
stream(tint) → sint  
stream(treal) → sreal  
stream(tbool) → sbool  
stream(tstring) → sstring  
stream(mtint) → msint  
stream(mtreal) → msreal  
stream(mtbool) → msbool  
stream(mtstring) → msstring  
stream(itint) → isint  
stream(itreal) → isreal  
stream(itbool) → isbool  
stream(itstring) → isstring

examples:

query toraster2([const tgrid value (1.2 3.4 5.6)]);

query toraster2([const mtgrid value (1.2 3.4 5.6 (duration (7 0)))]);

query toraster2(tiles([const sint value ((-10.0 -10.0 1.0) (1 1) (0 0 (0)) (31 0 (1)) (0 31 (2)) (61 61 (3))))];

query toraster2(tiles([const msint value ((-10.0 -10.0 1.0 1.0) (1 1 1) (0 0 0 (0)) (10 0 0 (1)) (0 10 0 (2)) (19 19 9 (3))))];

query toraster2(tiles([const isint value ((instant "2013-05-13") ((-10.0 -10.0 1.0) (1 1) (0 0 (0)) (31 0 (1)) (0 31 (2)) (61 61 (3))))]);

### **3.2.23 toregion**

Operator toregion maps a tbool object to a region object.

syntax:

\_ toregion

signature:

tbool → region

### **3.2.24 val**

Operator val returns the spatial data type object of an instant spatial data type object.

syntax:

val(\_)

signatures:

itint → tint

itreal → treal

itbool → tbool

itstring → tstring

example:

```
query val([const itint value ((instant "2013-05-13") ((0.0 0.0 1.0) (2 2) (0 0 (0 1 2 3)) (29 29  
(4 5 6 7))))]);
```

## References

- [BT12] Betreuerteam  
Rasterdaten in Seundo, Aufgabenbeschreibung Phase 2  
FernUniversität Hagen, November 2012
- [G11] Ralf Hartmut Güting  
A Short Guide to Writing Executable Queries in SECONDO  
FernUniversität Hagen, April 2011
- [GAB+12] Ralf Hartmut Güting, Dirk Ansorge, Thomas Behr, Christian Düntgen,  
Simone Jandt, Markus Spiekermann  
SECONDO User Manual, Version 3.1  
FernUniversität Hagen, März 2012
- [GBA+04] Ralf Hartmut Güting, Thomas Behr, Victor Almeida, Zhiming Ding,  
Frank Hoffmann, Markus Spiekermann  
SECONDO: An Extensible DBMS Architecture and Prototype  
FernUniversität Hagen, 2004
- [GBJ+12] Ralf Hartmut Güting, Thomas Behr, Simone Jandt, Fabio Valdés  
Fachpraktikum Erweiterbare Datenbanksysteme Wintersemester 2012/13  
Aufgabe der Phase 2 “Rasterdaten”  
FernUniversität Hagen, November 2012
- [ST11] SECONDO Team  
A Database System for Moving Objects  
FernUniversität Hagen, August 2011