Installation and Administration Guide

rasdaman version 6.0

rasdaman Version 6.0 Installation and Administration Guide

Copyright © 1999-2005 rasdaman GmbH. All rights reserved. No part of this publication is allowed to be reproduced, stored in a retrieval system, or transmitted in any form or by any means, be it electronically, mechanically, or in photocopy, without prior permission of rasdaman GmbH.

Printed in Germany.

This software/documentation contains proprietary information; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright patent and other intellectual property law. Reverse engineering of the software is prohibited.

Information in this document is subject to change without notice and should not be construed as a commitment. Rasdaman GmbH does not warrant that this document is error-free or complete. If you find any problems in the documentation, please report them to us in writing to the following address:

rasdaman GmbH
Adam-Berg-Str. 172a
D-81735 Munich
GERMANY

Tel. +49-(89)-67 000 146, fax –67 000 147

Email: baumann@rasdaman.com

Rasdaman is a registered trademark of rasdaman GmbH.

SQL is a registered trademark of International Business Machines Corporation.

Unix is a registered trademark of Unix System Laboratories, Inc.

Sun, SunOS, and Solaris are registered trademarks of Sun Microsystems, Inc.

HP-UX is a registered trademark of Hewlett-Packard Company.

Windows is a registered trademark of Microsoft Corporation.

IBM, DB2, and Informix are registered trademarks of IBM Corporation.

Oracle is a registered trademark of Oracle Corporation.

All other trade names referenced are service mark, trademark, or registered trademark of the respective manufacturer.

Preface

Overview

This guide provides information about how to use the rasdaman multidimensional database system, in particular: installation and system administration.

rasdaman interoperates with conventional database systems, be they relational or object-oriented, in an integrated manner. There is a clear, natural distribution of work between the two different database systems according to the data types: multidimensional data are managed by rasdaman, whereas the alphanumeric data remain in the conventional system[1]. At the bottom line, however, all data - multidimensional or

[1] In some disciplines, multidimensional data are referred to as *raw data* or *processed data*, depending on their status, whereas the accompanying alphanumeric data are called *meta data*.

alphanumeric - end up in the same physical database, thereby considerably easing database maintenance wrt. consistency, backup, etc. To this end, rasdaman makes use of the storage management facilities of the database system it is coupled to.

For the purpose of this documentation, we will call the conventional database system to which rasdaman is interfaced the *base DBMS*, understanding that this base DBMS is in charge of all alphanumeric data maintained as relational tables or object-oriented semantic nets.

rasdaman is available on different base DBMSs. This manual covers the base DBMS independent issues. Please consult additionally the *External Products Integration Guide* available for the different base DBMSs, as well as the other rasdaman guides for features of the rasdaman system which are common to all platforms.

Audience

The information in this manual is intended primarily for database and system administrators.

Rasdaman Documentation Set

This manual should be read in conjunction with the complete rasdaman documentation set which this guide is part of. The documentation set in its completeness covers all important information needed to work with the rasdaman system, such as programming and query access to databases, usage of utilities such as the graphical-interactive query tool *rView*, platform specific details, and release notes.

In particular, current restrictions, known bugs, and workarounds are listed in the Release Notes. All documents, therefore, always have to be considered in conjunction with the Release Notes.

The rasdaman Documentation Set consists of the following documents:

- C++ Developer's Guide
- Java Developer's Guide
- Query Language Guide
- Installation and Administration Guide
- External Products Integration Guides
- Error Messages
- rView Guide
- Release Notes

Additionally, the following documents describe the geographic map service applications rasgeo and rasogc:

- Rasgeo Guide
- Rasogc Guide

Table of Contents

1 Getting Started

This section outlines the hardware and software requirements, explains the licence key mechanism, and describes the rasdaman distribution directory. Make sure you are familiar with this before proceeding to the next Section which describes the installation procedure.

1.1 System requirements

For successful operation of rasdaman, the hardware and software environment needs to comply with the following server and client side requirements.

Hardware

It is recommended to have at least 2 GHz processor frequency and 1 GB main memory. Disk space depends on the size of the databases, as well as the requirements of the base DBMS of rasdaman chosen.

Software

Serverside software requirements are:

- One of the operating systems supported.
- One of the relational database systems supported by rasdaman, configured for operation with rasdaman (cf. *rasdaman Installation and Administration Guide*).
- Some Web browser capable of HTML 4.0
- Adobe Acrobat Reader for reading the manuals (available from http://www.adobe.com/products/acrobat/readstep2.html).

rasgeo requirements

To operate the map application rasgeo (see *rasgeo Guide*) you need:

- For the rasgeo and rasogc servlets: Have started at least one rasdaman server of communication type `HTTP`
- For the import/export tools: Have started at least one rasdaman server of communication type `RNP` or `RPC`
- Korn shell (available from http://www.kornshell.com)
- Java Runtime Environment (JRE) 1.4 or higher (available from http://www.sun.com)
- Jakarta Tomcat 1.4 or higher (available from http://www.sun.com)
- Tifflib 3.5 or higher (available from http://www.libtiff.org)
- ImageMagick (available from http://www.imagemagick.org/)

For Java and ImageMagick, you may want to use the standard Unix rpm tools. Assuming, for example, a Linux/Intel platform you can install the following rpm packages from your standard distribution:

- j2sdk-1_4_1_01-fcs-linux-i586.rpm
- jakarta-3_2_3_43.rpm
- tiff-3_5_5.rpm
- ImageMagick-5_3_8.rpm

These packages, among others, are provided in the `~rasdaman/tools` directory. Note the disclaimer file `DISCLAIMER.txt` there.

Clientside rasgeo requirements

- A Web browser supporting HTML 4.0.

  Rasgeo has been tested successfully with the current versions of Netscape Navigator, Microsoft Internet Explorer, Konqueror, and Opera. No restrictions are known for other browsers, however there is no guarantee that rasgeo will work properly in other environments.

1.2 Licence Key

In order to run a rasdaman server you have to obtain a license from rasdaman GmbH. For generating a license key, the ID of the host on which the system is supposed to run is needed. The license key string must be stored on the machine where the rasdaman server runs.

How Can I Obtain the Host ID?

The host id, which consists of a hexadecimal number, usually is obtained with the Unix command

```
hostid
```

Unfortunately, sometimes Unix is not set up properly and delivers a wrong host id. If in doubt, use the tool `rasgethostid` in `~rasdaman/bin/` to get an authoritative answer.

How Do I Install the License Key?

Most of the server functionality requires a license key. Should anything go wrong with the license check, the server will terminate with a diagnostic console output describing the problem.

After forwarding the host id to rasdaman GmbH, you will promptly receive a license key for your server. This key has to be stored in the file `.rmankey` in the rasdaman home directory.

Obviously, this file has to be readable by the rasdaman server during its runtime.

The key you obtain consists of a string of printable ASCII characters. Insert this string into the license file as one single line. The file must not contain any further characters, except possibly a line break at the end of the key string. In particular, there must not occur any whitespace characters inbetween the character sequence.

Note

Automatic mail formatting may cause splitting of the key string into two lines. For example, an email program may add a line break after, say, 80 characters so that you receive a license key which seemingly consists of two or more lines. In this case, delete the line break(s) manually so that the key string in the license file consists of only one single, consecutive line of characters.

Licensing on PC Hardware

The license key is validated against the server host id. While on other machines the host ID is coded into the hardware, on PC hardware the host id is derived from the networks settings. Plainly said, changing the server host's IP address will change its host id, and the rasdaman license key will no longer be valid for this machine.

Hence, it is highly recommended to first install the future rasdaman server host in the final LAN environment, and then obtain the license key.

1.3 Software Installation Directory Location

The recommended installation place for rasdaman is `/opt`.

Other locations are well possible (such as `/usr/local` or – not recommended - `/home`), and rasdaman works well provided the `$RMANHOME` variable is set properly.

The file system in which this directory resides should have at least 100 MB free space available.

1.4 Software Distribution Directory Structure

Having unpacked the rasdaman distribution set, the `$RMANHOME` release directory will have the following structure:

```
$RMANHOME
  admin/
  bin/
  doc/
    html/
    pdf/
  examples/
    c++/
    java/
    rasdl/
    rasql/
  images/
  include/
  jlib/
  lib/
  log/
  rasgeo/
  tools/
  versions/
```

The contents of the directories is as listed below:

`admin/`

This directory contains scripts to, among others, set up the base DBMS properly; see the *External Products Integration Guide* for your base DBMS to find out what you find there and what it does. The intention is that only the administrator activates these scripts.

bin/

contains the rasdaman executables and adjuncts, in particular

- the rasdaman server executables
- `rview`, the graphical-interactive database inspection tool
- `insertdemo.sh`, a script for setting up a demo database
- and some other rasdaman utilities.

doc/

contains the rasdaman documentation. File index.html is the Browser start for all documentation files, but you may as well navigate manually into the pdf directory. The doc/ directory provides valuable information to the rasdaman application developers, so it should be accessible to them.

- `doc/html/`: HTML documentation for the raslib C++ and rasj java client API.
- `doc/pdf/`: software documentation in pdf format. This contains all manuals for the rasdaman system.

examples/

contains example programs which demonstrate the usage of the rasdaman API. For C++, Java, and the rasdaman query language RasQL there is a separate subdirectory each. The following example material is provided (see Section 6.2 and the corresponding API manuals for details):

- `examples/c++/`: sample C++ sources using the raslib interface, together with a Makefile
- `examples/java/`: sample Java sources using the rasj interface, together with a Makefile
- `examples/rasdl/`: rasdl schema definition for some basic types
- `examples/rasql/`: sample RasQL queries

  For more information on the examples, please consult the *C++ Developers Guide*.

images/

Images for the demo database referred to in the *Query Language Guide.* These images can be inserted into the database using the `bin/insertdemo.sh` in conjunction with the `insertppm` utility (source in `examples/c++`, executable in `bin`).

include/

Header files for the raslib C++ API components. Central is the include file `rasdaman.hh` which in turn includes files from the subdirectories `raslib/` and `rasodmg/`.

jlib/

Java packages of the rasj API.

lib/

library files to be linked when using the rasdaman C++ API.

log/

server log files will be written into this directory during operation.

util/

third-party tools (such as free utilities) are located here.

rasgeo/

This directory contains the rasdaman geographic mapping service application, rasgeo, together with the OGC WMS service interface, rasogc. See the Rasgeo and Rasogc Guides for details.

These components have been distributed separately in the past; since this version, they are delivered in one package. The installation procedure, location, and other features, however, remain unchanged. In particular, a rasdaman version 5.1 installation is completely compatible with the one described here.

tools/

This directory contains third-party tools and utilities which may be useful. They are provided as is, without any warranty and support, and without being priced. The contents of this directory may change without prior notice.

versions/

This directory contains patches and update packages, as well as files saved during an update installation. It is generated only upon such an patch/update procedure.

1.5 Server Types

Rasdaman internally uses three different client/server communication protocols:

- SUN ONC RPC. This mechanism has been used by all C++ based programs and, consequently, the rasdaman tools which are binary executables (such as rasql), but is deprecated now.
- HTTP. This has been used by rasdaman applications programmed in Java, such as rasogc, but is deprecated now.
- RNP. This protocol substitutes both the other protocols.

To accommodate the different clients, an appropriate server has to be started as the counterpart. For example, a Java application needs an RNP or HTTP type server and is not able to communicate with an RPC type server.

The communication type of a rasdaman server is set during server definition using rascontrol (see Section 4.6).

2 Getting It Up: System Installation and Database Creation

This section describes how to install rasdaman, software package dependencies, and how to initialise rasdaman and the base DBMS so that the rasdaman system finally is up and running.

Please read this in conjunction with the rasdaman *External Product Integration Guide* for your particular base DBMS brand and the relevant manuals of the base DBMS vendor.

This section outlines the procedure for installing rasdaman from scratch. For incremental installation, such as from patch files and updates or upgrades follow the particular instructions there (e.g., in the release notes).

2.1 Hardware Requirements

Make sure that the target machine complies with the hardware requirements listed in Section 1.1.

2.2 Required Packages

Make sure that all packages required by rasdaman are properly installed and have the version required (see Section 1.1).

2.3 Operating System Account

Create an operating system user named `rasdaman`. It is recommended (though not necessary) to also create a group `rasdaman` which contains user `rasdaman`.

The recommended home directory for rasdaman is `/opt`. Other locations are well possible though (such as `/usr/local` or – not recommended - `/home`), and rasdaman works well provided the `$RMANHOME` variable is set properly.

2.4 Install rasdaman File Set

In the sequel, we assume that *`versionname`* is the name of the rasdaman version to be installed. This is a string like, for example, `5.1revD`.

Extract files

Log in as user `rasdaman` and expand the archive which constitutes the rasdaman distribution set into this directory. For a compressed tar archive *`rasdamantar.tgz`*, this is done by:

```
cd
tar xvfz rasdamantar.tgz
```

The resulting directory structure is as described in Section 1.4, however sitting in directory `~rasdaman/versions/`*`versionname`*.

Install files

To install the file set at the proper place and take into account an existing installation if any, invoke the installation script as user `rasdaman`; before doing so, manually set variable `RMANHOME`:

```
cd
export RMANHOME=~rasdaman
versions/install_versionname.sh
```

Notes

The above procedure uniformly applies to both new installations, complete version updates, and patches.

Preexisting files are, instead of being overwritten, shifted into directory `versions/`*`versionname`*`/Save` where they can be recovered easily if needed.

It is perfectly safe to invoke the installation script several times - pre-existing files are only saved if there isn't already another saved file sitting there.

If you accidentally installed a version and need to revert, copy back the contents of directory `versions/versionname/` into their previous places.

2.5 Licence Key Installation

Put the licence key you received (see Section 1.2) into file `~rasdaman/.rmankey`. Make sure the file is readable for user `rasdaman`.

2.6 Environment Setup

Set up the local environment. To this end, a resource file `.rmanrc` is shipped which contains all settings except for the ones determined by the individual target environment.

Follow these steps:

- Customize file `.rmanrc` to accommodate your local situation. In particular, choose the base DBMS brand and adapt the pertaining paths.
- Add the following line to your standard login file (`~rasdaman/profile`, `~rasdaman/.bashrc`, or similar):

  ```
  . .rmanrc
  ```

- Exit and log in again
- Test your rasdaman settings by invoking `initgeo.sh` without parameters. If it delivers usage information then the rasdaman path is ok.
- Optionally you can configure your system to automatically launch and shut down rasdaman upon system boot and shutdown. To this end, look into the script `~rasdaman/rasdaman` and follow the advice given there. Note that this requires some knowledge about Unix system administration, otherwise your system may not work properly! Settings should be verified through a reboot.

2.7 Base DBMS Set-Up

The base DBMS has to be set up according to the instruction in the *rasdaman External Products Integration Guide*. In particular it is strongly recommended to study Section 3, *Database Creation Procedure*, in said

document because it may be necessary to execute base DBMS specific steps before actually creating a database.

Roughly speaking, base DBMS configuration is ok if an SQL interpreter can successfully connect to the base DBMS when logged in as user `rasdaman`. If not you may have to reconsider settings in `.rmanrc` (see above).

2.8 Database Initialization

Create rasdaman database

Databases are created using the `rasdl` processor. During this process the tables needed by rasdaman are created, as well as initial administrative information.

The following creates a database with name `RASBASE`:

```
rasdl --database RASBASE -c
```

You may have to indicate a particular connect string that shows rasdl the DBMS server to address and to identify the particular database. While in Oracle usually the default connect string "`/`" does it, under Informix you may have to state, for server *`srv`* and database *`RASBASE`*,

```
rasdl --database RASBASE –c --connect RASBASE@srv
```

Type import

Next, a set of types must be imported which is very common and used, e.g., by the demo database and the geo service, rasgeo. Among the types defined here are binary, greyscale and color images. A type definition script is delivered which serves this purpose. Import it through

```
rasdl -r examples/rasdl/basictypes.dl –i
```

The `rasdl` processor reads the specified rasdl source file (`-r`) and imports the schema information into the database (`-i`).

Further `rasdl` options include writing of a corresponding C++ header file (`-hh`) and generating a list of available types in the database (`-p`).

A detailed description of the `rasdl` command together with a description of the structure of type definition files can be found in the *rasdaman C++ Programming Guide* and the *rasdaman Query Language Guide.*

Warning

Invocation of `rasdl` on a database is successful only as the `rasdaman` user (i.e., the one which has created the database).

Notes

For creation of new databases, rasdaman servers have to be restarted, otherwise the databases may not be recognized. Updating a rasdaman database schema, however, does not need a server restart.

Some base DBMS support the notion of multiple databases. In these cases, rasdaman also supports multiple databases.

2.9 Server Configuration

Rasdaman is a multi-server multi-user system. The server processes available must be configured initially. The number of processes that can be started depend on the licence acquired.

For details on configuration see Section 4; here we focus on the basics. Follow these steps:

- Edit file `~rasdaman/rasmgr.conf` and adapt the connect string following parameter "`-connect`" to the string that allowed you to successfully log in with rasdl; if you didn't need one, use "`-connect /`". Further, adapt the host definition in the `define` command to the name (or IP address) of the machine where rasdaman runs; frequently this will be "`-host localhost`". Define as many servers as you want, irrespective of your licence model.
- Edit file `~rasdaman/.rmanrc` and adapt the `RASSERVERS` variable to contain the choice of servers which will be launched during rasdaman startup. If the list contains more servers than your licence model allows then servers will be started in the sequence indicated until the number of licences is exhausted.
- Start rasdaman by calling `start_rasdaman.sh`.

Notes

- The messages printed by `start_rasdaman.sh` will do not always show the detailed system state. If, for example, the rasdaman servers fail to contact the base DBMS then nevertheless a message "Server started" will appear.

  Workaround: Consult `~rasdaman/log/rasstatus.html` (linked from the documentation main page `~rasdaman/doc/index.html`) or use this command to get the actual server state:

  ```
  rascontrol –x –e "list srv"
  ```

2.10 Demo Database

The rasdaman distribution contains a demo database which serves as a first test of successful installation.

Inserting demo data into the fresh database is done through

```
insertdemo.sh localhost 7001 images rasadmin rasadmin
```

Note that repeated invocations are not harmful – each of the sample collection will simple receive additional objects made of the same images.

After successful completion, you can check whether the three rasdaman collections containing the example images have been created through:

```
rasql –q "select r from RAS_COLLECTIONNAMES as r" \
      --out string
```

This command shows a list of all collections existing in the database. There should be `mr`, `mr2`, and `rgb`.

Further, you can inspect these data using, for example, rview (see the *rasdaman rView Guide*).

Congratulations! At this point, if everything completed successfully, rasdaman is up and running and prepared for data definition, data import and retrieval, and any other suitable task.

3 rasdaman Server Architecture

The parallel server architecture of rasdaman offers a scalable, distributed environment to efficiently process even very large numbers of concurrent client requests. Yet, server administration is easy to accomplish, with only few things to do to have a smoothly running, highly performant installation. Moreover, the system is implemented in a special high availability technique where most server management operations can be done with the server up and running, limiting the need for a server shutdown to the absolute minimum.

In this Section the general rasdaman server architecture is outlined. It is recommended to study this section so as to understand server administration terminology used in the next Section.

3.1 Server Executables Overview

The following server-side executables are provided in the `$RMANHOME/bin` directory:

- `rasmgr` is the central rasdaman request dispatcher;

- `rasserver` is the rasdaman server engine, it should not (and actually cannot) be invoked in a standalone manner;
- `rascontrol` allows to interactively control the rasdaman server by communicating with `rasmgr`;
- `raspasswd` allows to administrate rasdaman database user logins;
- `rasdl` is the schema maintenance tool.

The `rasdl` utility is explained in detail in the *rasdaman Query Language Guide*.

3.2 Server Manager and Server

Overview and Terminology

The rasdaman server configuration consists of one dispatcher process per computer, `rasmgr` (we will refer to it as *manager* in the sequel), and server processes, `rasserver` (referred to as *servers*), of which at a given time none, one, or several ones can be running. All server processes are under control of the manager. Server manager and rasdaman server(s) all run on the same physical hardware, the *rasdaman host*.

The servers resolve requests, thereby generating calls to the relational database system which in turn accesses its database files. For the purpose of this manual, the relational server together with the database it maintains are collectively called the *database*. The machine the relational database server runs on is referred to as *database host* (Fig. 1).

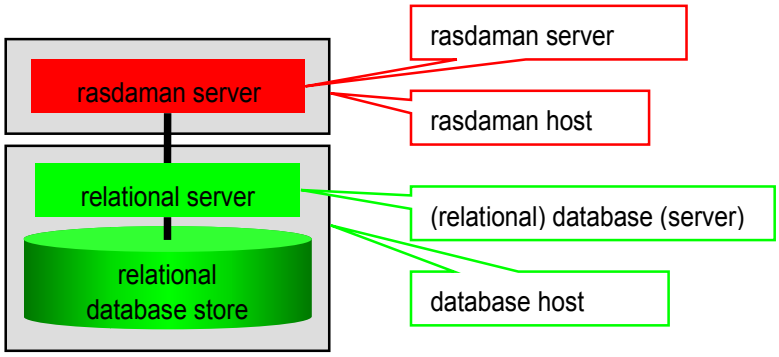

Figure 1: Overall server hierarchy, introducing the terminology for rasdaman hardware and software environment

Server Structure in General

The manager accepts client requests and assigns server instances to them, taking them from the pool of server processes it maintains. In distributed installations, it keeps contact to the managers on other machines to further dispatch client requests across all the rasdaman servers available. Whenever needed, the administrator can launch further

server instances (according to product license terms), or shut them down again.

Upon system configuration definition (see Section 4), a unique name is assigned to each server identifying it to the manager.

Each rasdaman server is assigned to a relational database server, laid down in the manager configuration file. Databases can be registered and associated to particular rasdaman servers at any time.

rasdaman hosts and database hosts are identified by their resp. host name in common domain address form, e.g., `martini.rasdaman. com` or `199.198.197.0.`

`Rascontrol` is the interactive front-end to `rasmgr` and, as such, the main utility for user and system management. It provides the necessary functions to manage the whole system configuration, to add and remove user, to change their rights, and to obtain information about system activity.

The rasdaman server, i.e., `rasserver`, is controlled by the manager which starts and stops server instances. Hence, the `rasserver` executable should not (and actually cannot) be invoked directly.

Dynamic Server Assignment

The process of client/server communication and server scheduling is done as follows (see numbers in Figure 2).

1. The client starts every `OPENDB` and `BEGIN TRANSACTION` request with an HTTP call to the manager, providing the required service type (RPC, HTTP, etc.) and the database name, together with user name and password.
2. The manager's answer is the server ID of a free server, or an error message in case no server is available or access is denied for the given login.
3. Client-Server communication to perform the database requests.
4. Upon `CLOSEDB` and `ABORT/COMMIT TRANSACTION` the server informs the manager that it is available again. This is also done upon a client timeout.

These negotiation steps are performed between client library and server, hence transparent to the application.

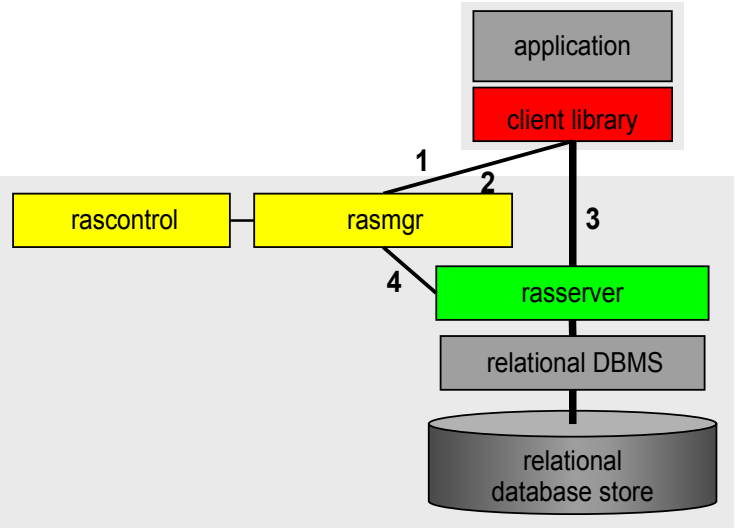

Figure 2: Internal server management

System Start-up

The rasdaman server system is started by invoking the server manager `rasmgr` (see Section 4.2). If it finds a configuration file, them autopmatically all servers indicated will be started; alternatively, server configuration can be done directly through `rascontrol` (see Section 4.3).

Invocation of the `rasmgr` executable must be done under the operating system login under which the rasdaman installation has been done, usually (and recommended) `rasdaman`.

Cascaded Servers

rasdaman servers running on different computers can be coupled so as to form one single server network. To this end, one of the rasdaman managers is assigned to act as master, doing the overall management, and all the others are slave rasdaman managers, doing local server management as before.

Clients address only the master manager, hence from a client point of view nothing changes, there still is only one point of service.

On every machine hosting rasdaman servers a separate manager has to run using its individual licence key. This key states, for its machine, parameters such as the maximum number of rasdaman servers and the manager is in master or slave mode. Additionally the manager maintains a configuration and an authorization file. None of them should be changed by the administrator, as they are generated, maintained, and overwritten by the manager.

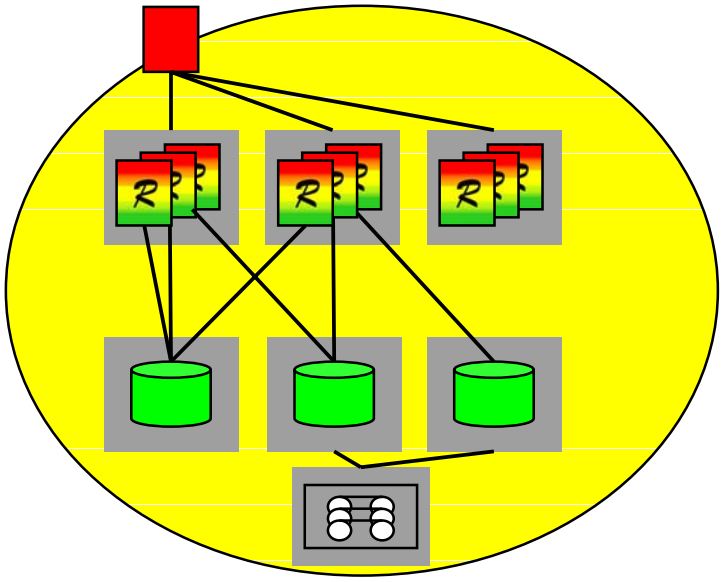

Figure 3: Cascaded rasdaman server, forming a rasdaman server farm with the Master Manager as single client entry point

rasdaman Manager Defaults

The master manager's default name is the hostname (the one reported by the UNIX command `hostname`), but it can be changed (see the `change` command). By default, it listens to port 7001 for incoming requests and uses port 7001 for outgoing requests to slave managers:

- Manager host name — output of the Unix command `hostname`
- Master manager listening port number default — 7001
- Manager port number for addressing slave managers — 7001
- Master manager to be addressed by slave — no default

Port Number Recommendations

To keep overview of the ports used, it is recommended to use the following schema (there is, however, no restriction preventing from choosing another schema – just keep an overview…):

- use port number 7001 for the server manager;
- use port numbers 7002 to 7999 for rasdaman servers.

4 Server Administration

In this Section, it is explained how the manager is used to start up and shut down servers, as well as how to monitor and influence server state.

It is recommended to first study the previous section so as to understand server administration terminology used here.

4.1 General Procedure

`rasmgr` vs. `rascontrol`

It is important to distinguish between the manager, `rasmgr`, and its control front-end, `rascontrol`. The manager runs as a background process, supervising activity of local (and possibly remote) rasdaman servers. Interaction between user (i.e., administrator) and the manager takes place through the interactive control front end.

In the sequel, it is first described how to launch the manager `rasmgr`, then `rascontrol` commands are detailed.

Important Security Note

To remain compatible with older rasdaman versions, clients use login "rasguest" / password "rasguest" by default (i.e., when no user and password are explicitly set by the application). In the distribution configuration, this user is defined to have read-only access to the databases – in plain words,

- According to the default configuration,
- users can access,
- but not manipulate databases
- without authentication.

Therefore, the administrator is strongly urged to adapt authentication settings to the local security policy before switching databases online.

See Section 4.11 to learn more about user management mechanisms.

4.2 Running the Manager

Manager Startup

Starting up the rasdaman system is done by invoking the rasdaman manager, `rasmgr`, from a shell under the `rasdaman` operating system login. Usually the manager will be sent to the background:

```
rasmgr &
```

Starting `rasmgr` is the only direct action to be done on it. Any further administration is performed using `rascontrol`.

Note that, unless a server configuration has been defined already, no rasdaman server is available just by starting the manager.

Master vs. Slave

Invocation of a master and slave manager is different. A master manager is the central server access turnpike accepting requests from clients; hence, the only additional information it needs is the port to listen to.

A slave manager, on the contrary, is a relay station which maintains local rasdaman servers and keeps contact to the master manager. Hence, a slave manager additionally must be provided with domain name and port number of the master manager to which the slave is subordinate. Below is the syntax for the two variants.

Invocation Synopsis

Master manager invocation synopsis:

`rasmgr [--help] [--hostname` *h*`] [--port` *p*`]`

`--help` print this help

`--hostname` *h* host on which the manager process is running is accessible under name / IP address *h*
(default: output of Unix command `hostname`)

`--port` *p* manager will listen to port number *p*
(default: `7001`)

Slave manager invocation synopsis:

`rasmgr [--help] [--name` *h*`] [--port` *p*`] --master` *m* `[--mport` *x*`]`

`--help` print this help

`--name` *h* master manager will know this slave started as running on host with name *h*
(default: host name of machine)

`--port` *p* manager will listen to port number *p*
(default: `7001`)

`--master` *m* assume *m* as domain name of the master manager (no default, mandatory parameter)

`--mport` *p* assume *p* as port number of the master manager (default: `7001`)

Syntactically, the distinction between launching a master and a slave manager is in the `--master` *m* option: if this option is missing, then a master server manager is started; if it is present, a slave is started which will try to connect to master *m*.

The normal launch procedure is to start the master first, then every slave, providing the master's host name and port number. The slaves try to contact the master and to register their status (i.e., being up and allowing the licensed number of concurrent rasdaman servers). If the master is not up, or cannot be contacted, the slave also waits to be contacted. The only reasons why a slave stops working at start-up time is (i) that no valid license key is available or (ii) when it gets a refuse by the master; the latter happens when the master doesn't know the slave.

Examples

To start a master manager which will listen at port 7001:

```
rasmgr --port 7001
```

To start a slave manager on server `martini` (also listening on port 7001) which will be subordinate to the master manager on host `campari` (both machines are assumed to run in the same local networks domain):

```
rasmgr –-name martini --port 7001 \
       –-master campari –-mport 7001
```

4.3 `rascontrol` Invocation

The manager front end, `rascontrol`, is a command-line interface used for rasdaman administration. It allows to define the whole rasdaman system configuration, including start up and shut down of server instances and user logins and rights.

To secure access to the server administration facilities, `rascontrol` performs a login process requesting login name and password similar to the Unix `rlogin` command. User name must be one of the users defined in the rasdaman authentication list (see Section 4.11).

`rascontrol` Synopsis

```
rascontrol [-h|--help] [--host h] [--port n] [--prompt n]
           [--quiet]
           [--login|--interactive|--execute cmd|--testlogin]
```

`--host` *h* — name of the host where the master manager runs (default: `localhost`)

`-h`
`--help` — this help

`--port` *n* — port number at which the master manager listens to requests (default: `7001`)

`--prompt` *n* — change rascontrol prompt as following:
`0` - prompt `'>'`
`1` - prompt `'rasc>'`
`2` - prompt `'user:host>'`
(default: 2)

`--quiet` — quiet, don't print header (default for `--login` and `--testlogin`)

`--login` — print login and password, obtained from interactive input, to `stdout`, then exit (see *Script Use* below)

`--interactive` — read login and password from environment variable `RASLOGIN` instead of requesting it interactively

`--execute` *cmd* — execute single *cmd* and exit (batch mode); all text following `–x` until end of line is passed as `command`; this option implicitly assumes `-e`

`--testlogin`	just do a login and nothing else to check whether the login/password combination provided in the `RASLOGIN` variable is valid

Interactive Use

In interactive use, `rascontrol` will be invoked with the host parameter only. Following successful authentication, `rascontrol` accepts command line input from `stdin`.

Here is an example session (*`mypasswd`* will not be echoed on screen):

```
tcsh> rascontrol
Login name: mylogin
Password: mypasswd
mylogin:localhost> define dbh h1 -connect /
mylogin:localhost> define db  d1 –dbh h1
mylogin:localhost> define srv s1 -host localhost
                          -type h -dbh h1
mylogin:localhost> up srv s1
mylogin:localhost> save
mylogin:localhost> exit
tcsh>
```

Script Use

Alternatively to interactive login, user and password information can be taken from the environment variable `RASLOGIN`. This variant is suitable for batch scripting in conjunction with the `–x` option.

The following example shows how first the `RASLOGIN` is set appropriately:

```
export RASLOGIN=`rascontrol --login`
```

…and then a sample Unix shell script which starts all rasdaman servers defined in the system configuration, performing implicit login from the environment variable contents which has been obtained from the previous command and pasted into the shell script:

```
#!/bin/ksh
export RASLOGIN=rasadmin:mytotallyencryptedpassword
rascontrol -x up srv -all
```

Comments in Scripts

To enhance legibility of scripts, `rascontrol` accepts comments in the usual shell syntax: Lines beginning with a hash sign '#' will be ignored, whatever they may contain. An example is usage in shell *here documents* (type `man sh` in your favourite shell for further information on this feature):

```
rascontrol <<EOF
# this is the command submitted to rascontrol:
list srv -all
# now terminate rascontrol:
exit
```

```
# the following line terminates rascontrol input:
EOF
```

Prefabricated Script

To ease system installation, a shell script, `rasconfig.sh`, is delivered in the `bin` directory which contains initial server definitions to set up a rasdaman Classic Edition configuration (i.e., one server process).

Don't simply run this script as is; prior to running it you may want to first understand it and then adapt it to your local situation.

4.4 `rascontrol` Command List

Command Synopsis

`help`	display information (general or about specific command)
`exit`	exit `rascontrol`
`list`	list info about the current status of the system
`up`	start server(s)
`down`	stop server(s) and rasdaman server manager(s)
`define`	define a new object
`remove`	remove an object
`change`	change parameters of objects
`save`	make configuration changes permanent
`check`	check current status of a slave rasdaman server manager

In the remainder of this section, commands are explained in detail.

4.5 Server Hosts

Define Server Hosts

`define host` *h* `-net` *n* `-port` *p*

h	symbolic host name
`-net` *n*	set network host name to *n*
`-port` *p*	TCP/IP port on which the rasdaman manager will listen

Change Server Host Settings

```
change host h [-name n] [-net x] [-port p]
       [-uselocalhost [on|off] ]
```

h	host name whose entry is to be updated
`-name` *n*	change host name to *n*
`-net` *x*	change network name to *x*
`-port` *p*	change port number to *p*
`-uselocalhost [on|off]`	use domain name `localhost` (IP address 127.0.0.1) instead of regular network host name; usually this speeds up communication a little (master server manager only; default: `on`)

Note that it is not possible to change network name or port for a host while this server is running.

Remove Server Host Definitions

```
remove host h
```

h	host name whose entry is to be deleted

Remove host *h* from the definition table.

It is not possible to remove a host definition while the corresponding host has active servers.

Status Information

```
list host
```

List all hosts currently defined.

4.6 rasdaman Servers

Define rasdaman Servers

```
define srv s -host h -type t -port p -dbh d
       [-autorestart [on|off] [-countdown c]
       [-xp options]
```

s	a unique, not yet used name for the server
`-host` *h*	name of the host where the server will run
`-type` *t*	communication type: *t* is `r` for RPC, `h` for http
`-port` *p*	the RPC *program number* for RPC servers (recommended:0x2999001 - 0x2999999), TCP/IP port for http servers (recommended 7002 - 7999)

`-dbh d` database host where the relational database server to which the rasdaman server connects will run

`-autorestart a`
for `a=on`: automatically restart rasdaman server after unanticipated termination
for `a=off`: don't restart
(default: `a=on`)

`-countdown c`
for `c>0`: restart rasdaman server after c requests
for `c=0`: run rasdaman server indefinitely
(default: `c`=1000)

`-xp options` pass option string `options` to server upon start
(default: no options, i.e., empty string)

Option `-xp` must be the last option. Everything following "`-xp`" until end of line is considered to be "`options`" and will be passed, at startup time, to the server; see *Server Control Options* below for the list of options available.

Change Server Settings

```
change srv s [-name n] -type t [-port p] [-dbh d]
       [-autorestart [on|off] [-countdown c]
       [-xp options]
```

`s` change settings for server `s`

`-name n` change server name to `n`

`-port p` change port number to `p`

`-dbh d` new database host where the relational database server runs to which the rasdaman server connects

`-autorestart a`
for `a=on`: automatically restart rasdaman server after unanticipated termination
for `a=off`: don't restart

`-countdown c`
for `c>0`: restart rasdaman server after c requests
for `c=0`: run rasdaman server indefinitely

`-xp options` pass option string `options` to server upon start

Option `-xp` must be the last option. Everything following "`-xp`" until end of line is considered to be "`options`" and will be passed, at startup time, to the server; see *Server Control Options* below for the list of options available.

Restrictions:

- The server host cannot be changed.
- The server name cannot be changed while the server is up.

- The new settings will be used only next time the server starts.

Remove rasdaman Server Definitions

`remove srv` *s*

h	server name whose entry is to be deleted

Remove server *s* from the definition table.

It is not possible to remove a server definition while the corresponding server is up and running

Status Information

`list srv [` *s* `| -host` *h* `| -all ] [-p]`

s	give information about server *s*
`-host` *h*	give information about all servers running on host *h* information is requested
`-all`	list information about all servers on all hosts (default)
`-p`	additionally list configuration information

The first is variant prints status information of the currently defined server(s); if *s* is provided, then only server `s` is listed.

4.7 Database Hosts

Define Database Hosts

`define dbh` *h* `[-connect` *c*`]`

h	a unique symbolic database host name, usually the host machine name
`-connect` *c*	the connection string used to connect the `rasserver` to the database server, i.e., the `-connect` parameter for `rasserver`

Change Database Host Settings

`change dbh` *h* `[-name` *n*`] [-connect c]`

h	database host whose entry is to be changed
`-name` *n*	change symbolic database host name to *n*
`-connect` *c*	change connect string to *c*

The connection string can be changed at any time, however the servers will get the information only when they are restarted.

Remove Database Host Definitions

`remove dbh` *h*

h	database host name whose entry is to be deleted

Remove database host *h* from the definition table.

It is not possible to remove a database host definition while this database host has active servers connected to it.

Status Information

`list dbh`

List all relational database hosts currently defined.

4.8 Databases

Databases represent the physical database itself, together with the relational database server accessing them. It is possible to have multiple database definitions in the rasdaman server environment which are distinguished by the database host; the interpretation, then, is that the same contents (be it the same physical database or a mirrored copy) is available through relational servers running on the different hosts mentioned. In other words, when a client opens a database, the server manager can freely choose any of the database hosts on which the database indicated is defined.

The pair (database,database host) must be unique.

Define Databases

`define db` *d* `–dbh` *db*

d	define database with name *d*
`-dbh` *db*	set database host name to *db*

Change Database Settings

`change db` *d* `-name` *n*

d	database whose name is to be changed
`-name` *n*	change to new database name *n*

Remove Database Definitions

`remove db` *d* `–dbh` *db*

d	name of database to be removed
`-dbh` *db*	host name of database to be removed

Remove definition of database *d* from the definition table. The database itself remains unchanged, it is not physically deleted.

It is not possible to remove a database definition while the corresponding database has open transactions.

Status Information

`list db [ d | -dbh h | -all ]`

d	give information about servers connected to database *d*
`-dbh` *h*	give information about all servers connected to database *d* via database host *h*
`-all`	list information about all servers connected to any known database (default)

List relational database(s) defined.

4.9 Server Start-up and Shutdown

Server Start

`up srv [ s | -host h | -all ]`

s	start only server *s*
`–host` *s*	start all servers on host *h*; this requires that a manager has been started on this host previously.
`–all`	start all servers defined; note that only those servers can be started on whose host a manager is currently running.

Look up the named server(s) in the definition list, and start the specified one(s) using the previously defined individual startup parameters.

At least one of the options *s*, `-host` *s*, and `–all` must be present.

Server Shutdown

`down srv [ s | -host h | -all ] [-force] [-kill]`

s	name of the server to be stopped
`–host` *s*	terminate all servers on host *h*
`–all`	terminate all servers
`-force`	`send SIGTERM` immediately, don't wait for transaction end
`-kill`	`send SIGKILL` immediately, don't wait for transaction end

This command shuts down the indicated server(s). At least one of the options *s*, `-host` *s*, and `–all` must be present.

Without `-force` and `-kill`, the server is marked for shut down and will actually be terminated by sending `SIGTERM` after completing the current transaction. With `–force` and `-kill`, the server is terminated instantaneously; this should be handled with extreme caution, as experience shows that relational database systems react differently on such a situation: usually a running transaction is aborted (which is the desired behavior), but sometimes the running transaction is committed (most likely leaving the database in an inconsistent state). See a Unix manual for the difference between `SIGTERM` and `SIGKILL` signals.

The manager on host *h* is not terminated.

4.10 Inter-Manager Status Request

`check host` *s*

s check if slave manager on *s* is up, retrieve status information

The master manager can contact a slave manager with this command. Normally this is not necessary as the slave during startup synchronizes with its master. However, if the slave has been launched before the master then the master server this way can manually synchronize with this slave. Similarly, it helps to overcome race conditions during automatic startup in the boot process of a database server network.

4.11 Users and Their Rights

Similarly to operating systems, rasdaman knows named users with access rights associated to them. Each rasdaman client must log in to the system under a specific login name using its specific password; this holds for database clients as well as for database administration. With each login name, a set of rights is associated which determines the set of actions admitted to the user under this login.

To this end, the rasdaman administrator manages user login names (user names) equipped with a password and rights to access the databases.

Attention: There is no way to retrieve a lost password!

The set of known logins as well as the associated rights all are under administrator control; the `define` and `remove` commands serve to add or delete user logins, the `change user` command allows to individually assign rights to a login.

In the rasdaman system's initial state after installation, user `rasadmin` is defined owning all possible rights (see below). Depending on the

rasdaman license acquired, a further user `rasguest` is defined which owns read-only access ("`R`") rights.

For both users, the password initially is identical with the user name. It is highly recommended to change this immediately using the `raspasswd` utility (See Section 5).

Define New User

`define user` *u* `[-passwd` *p*`] [-rights` *r*`]`

u	login name, must be unique (i.e., not yet existing)
-passwd p	set login password to pass (default: user name)
`-rights` *r*	rights associated with this login (default: `R`, i.e., read-only)

The user's password can be changed at any time using the `raspasswd` utility (see Section 5).

Remove User

`remove user` *u*

u	login name to be removed

The user is removed from the login list and henceforth cannot login to the rasdaman system any more.

User Rights

User rights are indicated by upper case letters. They are divided into two categories: *system rights* and *database rights*. System rights apply to the whole system configuration of a server machine, whereas database rights can be specified individually for a database.

The following system rights are defined:

`C`	user may change the system configuration
`A`	access control: the user may perform user management
`S`	start/stop right: the user may start and stop the system, in particular: rasdaman servers
`I`	info retrieval: the user may retrieve server status information

The following database rights are defined:

`R`	user is allowed read data (`select`…`from`…`where`) from rasdaman databases
`W`	user is granted write access (`update`, `insert`, `delete`) to rasdaman databases

Notation of Rights

In the `change user` command used for user rights administration, a user's rights set is described by a *rights string*. It is built from letters denoting the rights to be granted.

To revoke a right, leave out the corresponding character. To grant no rights at all, use – (minus sign).

No blanks or other characters are allowed in a rights string.

Examples of valid rights strings are:

grant all rights: `CASIRW`

grant read access only: `R`

grant no rights at all: `-`

These are examples for *invalid* rights strings:

Blanks between rights: `CA SIR  W`

Invalid characters I: `AXYZS`

Invalid characters II: `"A_+S`

Change User Attributes

`change user` *u* `[-name` *n* `| -passwd` *p* `| -rights` *r*`]`

`Options:`

u	user login to be updated
`-name` *n*	change user name to *n*
`-passwd` *p*	change password to *p*
`-rights` *r*	change rights of user *u* according to rights string *r*

Change name of user, login password, or user rights.

To change the password it is recommended to use the `raspasswd` utility instead of the `change user` command, as `raspasswd` does not echo the plain password on screen while `rascontrol` does.

Status Information

`list user [–rights]`

`-rights`	additionally list rights assigned to each user

List all user names currently defined, optionally with their rights.

4.12 Server Control Options

The following options can be passed to the server when it is started by the server manager using the `up srv` command. Option settings are defined

for a particular server using the `rascontrol` command `change srv -xp` which passes the rest of the line after `-xp` on to the server upon starting it (see Section 4.6).

`--log` *`logfile`*
: print log to *`logfile`*.
If *`logfile`* is `stdout`, then log output will be printed to standard output.
(default: `$RMANHOME/log/`*`serverid`*`.log`)

`--mgmntint [`*`m`*`]`
: server management interval in seconds for garbage collection (default: 120 sec)

`--notimeout`
: server does not check for client timeouts (default)

`--timeout [`*`t`*`]`
: client time out in seconds for sign-of-life signal.
If no `t` indicated: 300 sec; if set to 0, no sign-of-life check is done.
Activated only if `--mgmntint` is also set.

`--opt` *`n`*
: optimization level
one of 0, 1, 2, 3, 4 whereby
0 = no / 4 = maximum optimisation
(default: 4)

`--transbuffer` *`b`*
: set maximum size of transfer buffer to *`b`* bytes
(default: 4 MB = 4,194,304 bytes)

4.13 Miscellaneous

Help

`help`

Display top level help page

`help [`*`command`*`]`
`command` `help`

Display information specific to *`command`* (both syntax variants are equivalent).

Version and License Information

`list [ license | version ]`

`license`
: display licence information about the rasdaman installation.

`version`
: display rasdaman server version.

Save Changes to Disk

```
save
```

The `save` operation writes the current configuration and authorization values to disk. All changes done during the session thus become permanent.

`rascontrol` Termination

```
exit
```

terminates `rascontrol`.

5 Database User Password Administration

The `raspasswd` utility allows users to interactively change their password. After requesting the user name and the current password, the user is prompted for the new password is twice to exclude typing errors.

If the system configuration consists of several server managers (distributed installation), then the master server manager must be addressed to change the password.

Invocation Synopsis

`raspasswd [--help] [--host` *`masterhost`*`] [--port` *`p`*`]`

`--help` this help text

`--host` *`masterhost`*
master server manager host name
default: `localhost`

`--port` *`p`* port number of master server manager

Example

```
raspasswd localhost
Login name: mylogin
```

```
Password: mypasswd
New password: mynewpasswd
Retype new password: mynewpasswd
```

Note that none of the passwords actually is echoed.

6 Example Database and Programs

6.1 Example Database

A demonstration database is provided as part of the delivery package which contains the collections and images described in the *Query Language Guide*. To populate this database, first install the system as described here in the *Installation Guide* and the base DBMS specific *External Products Integration Guide*, and then invoke `insertdemo.sh` in the `bin` directory. This script makes use of the example images sitting in the `examples` directory, as well as the `insertppm` executable described below.

It is recommended to populate this demo database – it occupies only marginal disk space – first: Successful generation of this database shows overall successful rasdaman installation.

Before the test programs can be used, the demo database has to be created and schema information has to be imported. The following command line creates the database *RASBASE* (see also Section 2.7):

```
rasdl --basename RASBASE --createdatabase
```

The following imports schema information:

```
rasdl --basename RASBASE
      --read examples/rasdl/basictypes.dl –-insert
```

Finally, the following line establishes the demo database (using a script from the `bin` directory which itself relies on the `insertppm` executable):

```
insertdemo.sh base
```

It is not important whether the rasdaman server is running during `rasdl` execution, however, the server is required for the `insertdemo.sh` script, as this is a client application.

For further information on `rasdl` see the *C++ Programming Guide*.

6.2 Example Programs

Several example programs are provided in the `c++` and `java` subdirectories of `$RMANHOME/examples`. Each directory contains a Makefile plus several `.cc` and `.java` sources, resp.

Makefile

The `Makefile` serves to compile and link the sample C++ sources files delivered. It is a good source for hints on the how-tos of compiler and linker flags etc.

`insertppm.cc`

The `insertppm` program inserts a PPM / PGM / PBM image into a rasdaman database. This program serves a dual purpose: On the one hand, it shows how a custom application can be created which inserts single images into a database from a file in some data exchange format. On the other hand, it is the utility to establish the sample database used in the *Query Language Guide.*

A compiled version of this program can also be found in the `bin` directory to avoid that this important utility is overwritten by exploring the sample code.

`lookup.cc` and `lookup.java`

The `lookup` program reads a specified collection and prints the MDDs and their content. Any collection from the demo database can be inspected, but be warned of the data volume generated by ASCII printouts.

`avg-cell.cc` and `avg-cell.java`

This program computes the average cell value from all images of a given collection.

Note that it requires grayscale images! A good candidate collection is `mr` from the demo database.

`avg-cell-red.cc` and `avg-cell-red.java`

Same as `avg-cell`, but takes the red component of an RGB image for averaging.

A good candidate collection is `rgb` from the demo database.

Note

All programs, once compiled and linked, print a usage synopsis when invoked without parameter.

7 Troubleshooting

7.1 Cannot start rasdaman server

Question:

Upon startup of the server, I keep on getting a message

```
Checking if I am the only rasmgr on this
machine....Error: rasmgr instance already active.
```

…however, no rasmgr process is active.

Answer:

A previous forced termination of rasmgr may have left open the portmapper connection. Delete it by issuing, as superuser:

```
rpcinfo -d 697892864 1
```

8 PDF Documentation

The whole documentation set, including the manual you are reading right now, is delivered electronically as PDF. After unpacking the distribution, the PDF documentation can be found in the directory `$RMANHOME/doc/pdf`[2]. It can be read with Acrobat Reader Version 2.1 or later. The following files are included in this directory:

- `error-messages.pdf`: Error Messages, a list of all error codes and the corresponding error explanation texts.
- `inst-guide.pdf`: This document.
- `ql-guide.pdf`: Query Language Guide, a description of the declarative query language RasQL.
- `dev-guide-c++.pdf`: C++ Programming Guide, explaining the use of the raslib C++ API to write user programs using the rasdaman server. Also contains information on how to link programs and how to use the rasdl preprocessor to manage schema information.

[2] Sometimes the PDF and HTML documentation may be delivered separately.

- `dev-guide-java.pdf`: Java Programming Guide, explaining the use of the rasj Java API to write user programs using the rasdaman server.
- `release-notes.pdf`: Release Notes, a summary of features and specialities of the current release.
- `rview-guide.pdf`: rView Guide, a description of the graphical user interface for interactively working with the rasdaman server.
- `external-guide-*.pdf`: External Products Integration Guide for the specific base DBMS. Contains specific information for the base DBMS and information on how to access data in the base DBMS in a rasdaman client program and link it with data stored in rasdaman.