QaoS User Manual

Version 1.0, February 2016

by Sebastian Freundt
and QaoS developers

This file documents the GAP Interface to QaoS databases

Version 1.0
February 2016.

Short Contents

QaoS 1
1 Concept 3
2 Installation 5
3 User Functions 7
4 Index 13

Table of Contents

QaoS 1

1 Concept 3

2 Installation 5

2.1 Installation of the GAP package 5
2.2 Installation of cURL 5

3 User Functions 7

3.1 Retrieve objects matching a query 7
3.2 Examine the contents of a database collection 9

4 Index 13

QaoS

This manual documents the GAP Interface to QaoS databases. These are databases of algebraic objects at the KANT Group Berlin.

This manual should be considered as introduction for users. For more information about QaoS, we refer to the manual of QaoS.

1 Concept

QaoS serves as gateway within GAP to the QaoS databases of Algebraic Objects in Berlin. The database frontend is able to output internal data in a GAP-conform format. This GAP package both triggers the database and retrieves the data. Beyond that, it is also able to retransform received information into GAP objects which then could be used in the 'normal way'.

QaoS itself is a backend/frontend combination of an SQL database which is designed to grant public anonymous access via HTTP for reading operations, and restricted access via HTTP and TCP/IP for writing operations.

The package functionality is partly driven by external utilities. At the moment we support cURL only. cURL is used for network communication, HTTP GET and POST requests and reading results from the network socket, while the GAP part is used for content dependent operations.

Data for Transitive Groups has been taken from the current GAP distribution. We like to thank Alexander Hulpke for his permission to offer transitive group data in our databases.

- Alexander Hulpke, Constructing Transitive Permutation Groups; J. Symb. Comp. 39 (2005), 1-30.

2 Installation

Installation of the package is fairly easy. Fetch the latest qaos package at

`http://www.math.tu-berlin.de/~kant/download/gap/qaos.tar.bz2`

or via FTP at

`ftp://ftp.math.tu-berlin.de/pub/algebra/Kant/contrib/gap/`

2.1 Installation of the GAP package

If you have permission to add files to the installation of GAP 4 on your system you may install the qaos package into the `pkg/` subdirectory of the GAP installation tree.

```
shell> cd /path/to/GAP4/installation/tree/
shell> cd pkg/
shell> tar xjf /path/to/qaos.tar.bz2
```

This yields another subdirectory called `qaos/` with all the necessary files.

If you do not have the permission to install the package globally just install it to some private area, for example your home directory.

```
shell> cd ~
shell> mkdir mygap
shell> mkdir mygap/pkg
shell> cd mygap/pkg/
shell> tar xjf /path/to/qaos.tar.bz2
```

Now whenever you start GAP, be sure to pass the `mygap/` directory to the package search path of GAP.

```
shell> gap -l ";$HOME/mygap"
```

2.2 Installation of cURL

Go to `http://curl.haxx.se` and fetch the latest release of cURL for your system. Install it. Refer to cURL installation instructions if necessary.

If you have downloaded precompiled binary packages for your system and none of them seem to work, you may also try installing cURL via sources. Just fetch the source archive, unpack it somewhere and say

```
shell> ./configure && make && make install
```

Finally, you can test for a successful curl installation by

```
shell> curl http://curl.haxx.se
```

If this command spits out lots of HTML into your terminal everything is installed properly. If not, adjust your *$PATH* variable such that

```
shell> which curl
```

finds a valid path to the curl binary.

3 User Functions

QaoS provides some user functions to obtain information from the databases. Currently, we support queries for transitive groups and algebraic number fields. The corresponding commands are *QaosTransitiveGroup* and *QaosNumberField*.

3.1 Retrieve objects matching a query

`QaosTransitiveGroup` *query* [*optarg*] [Function]

Return transitive groups matching *query*. The amount of results is limited by the global variable *QaosDefaultLimit*.

Optional argument *optarg* is a record with some of the following components:

- *Action* (string)
  - determine an action to perform on *query*
  - default value: `"query"`
  - possible values: `"query"`, `"count"`
- *ColGroups* (list of strings)
  - determine which result groups should be returned; results of QaoS are usually grouped by a so-called 'column group'. In the web interface these column groups are choosable by checking the corresponding name under the text field for the query.
  - default value: `["size","props","perm-struct"]`
  - possible values: combinations of `"size"`, `"props"`, `"perm-struct"`, `"fp-struct"`
- *Limit* (integer/string)
  - determine how many results are retrieved (maximally)
  - default value: *QaosDefaultLimit*
  - possible values: any positive integer or `"all"`
- *Offset* (integer/string)
  - determine an offset on the set of results, this may be used to sequentially retrieve blocks of results.
  - default value: 0
  - possible values: any non-negative integer

`QaosNumberField` *query* [*optarg*] [Function]

Return number fields matching *query*. The amount of results is limited by the global variable *QaosDefaultLimit*.

Optional argument *optarg* is a record with some of the following components:

- *Action* (string)
  - determine an action to perform on *query*
  - default value: `"query"`
  - possible values: `"query"`, `"count"`
- *ColGroups* (list of strings)

- determine which result groups should be returned; results of QaoS are usually grouped by a so-called 'column group'. In the web interface these column groups are choosable by checking the corresponding name under the text field for the query.
- default value: ["size","struct","clsgroup","galgroup"]
- possible values: combinations of "size", "struct", "clsgroup", "galgroup", "galprops"

- *Limit* (integer/string)
  - determine how many results are retrieved (maximally)
  - default value: *QaosDefaultLimit*
  - possible values: any positive integer or "all"
- *Offset* (integer/string)
  - determine an offset on the set of results, this may be used to sequentially retrieve blocks of results.
  - default value: 0
  - possible values: any non-negative integer

`QaosDefaultLimit` *Integer/String* [Variable]

Determine the (maximal) amount of results returned in a query. Usually, this can be overridden with the *Limit*-component in optargs.

The default value is 25.

The value can be any positive integer or the string "all", in which case no limit is set and all results are returned.

CAUTION: Setting this variable to "all" may result in exhaustive use of memory, network bandwidth, and time. Therefore, always consider to perform the count action on a query before loading all of the results. The database of number fields contains more than 1.35 million number field objects. Downloading them all means retrieving a string of approx. 1.8 GB length!!!

The following examples show their usage:

Query for transitive groups of degree 4.

```
gap> QaosTransitiveGroup("d4");
#I Retrieved 5 Transitive Groups.
#I (C) 2004-2005 QaoS developers <kantdb@math.tu-berlin.de>,
#I The Kant Project <kant@math.tu-berlin.de>
#I qaos--dev--1.0--patch-32
#I 2005-08-29 07:54:18 UTC
<collection from database: 5 transitive groups; "d4">
```

Count transitive groups of degree 8.

```
gap> QaosTransitiveGroup("d8",rec(Action:="count"));
#I (C) 2004-2005 QaoS developers <kantdb@math.tu-berlin.de>,
#I The Kant Project <kant@math.tu-berlin.de>
#I qaos--dev--1.0--patch-33
#I 2005-08-29 10:04:49 UTC
```

```
50 transitive group satisfy "d8"
```

Retrieve the first 25 transitive groups of degree 8.

```
gap> q1:=QaosTransitiveGroup("d8");
#I Retrieved 25 Transitive Groups.
#I (C) 2004-2005 QaoS developers <kantdb@math.tu-berlin.de>,
#I The Kant Project <kant@math.tu-berlin.de>
#I qaos--dev--1.0--patch-33
#I 2005-08-29 10:08:50 UTC
<collection from database: 25 transitive groups; "d8">
```

Now retrieve the next 25 results.

```
gap> q2:=QaosTransitiveGroup("d8",rec(Offset:=25));
#I Retrieved 25 Transitive Groups.
#I (C) 2004-2005 QaoS developers <kantdb@math.tu-berlin.de>,
#I The Kant Project <kant@math.tu-berlin.de>
#I qaos--dev--1.0--patch-33
#I 2005-08-29 10:09:41 UTC
<collection from database: 25 transitive groups; "d8">
```

Retrieve number fields of degree 4.

```
gap> QaosNumberField("d4");
#I Retrieved 25 Algebraic Number Fields.
#I (C) 2004-2005 QaoS developers <kantdb@math.tu-berlin.de>,
#I The Kant Project <kant@math.tu-berlin.de>
#I qaos--dev--1.0--patch-32
#I 2005-08-29 07:55:11 UTC
<collection from database: 25 number fields; "d4">
```

When either of these functions is called with no arguments a short general help is displayed:

```
gap> QaosTransitiveGroup();
QaosTransitiveGroup(<query> [, <optarg>]) -> <result>

Searches the Algebraic Objects Database in Berlin.
The query string equals the keyword search method in the web surface.
See
‘http://qaos.math.tu-berlin.de/qaos/query.scm?type=trnsg&action=Help’
for more information about the syntax and keywords.

Note: You must have ‘curl’ (see http://curl.haxx.se) installed and properly
configured in order to use QaoS from within GAP.

true
```

In this manual we also refer to the help screen of the web interface to obtain more detailed information about the query string.

3.2 Examine the contents of a database collection

In order to actually see the information there is a function called *QaosResult* which displays a list of retrieved objects.

`QaosResult` *database-collection* [Function]
Return the list of retrieved objects in *database-collection*.
This function actually reveals the result of a query.

`QaosResult` *database-count* [Function]
Return the (non-negative) integer of a count query (see *Action*-component in optarg record).

Let us examine the groups behind a query for degree 8 abelian transitive groups.

```
gap> Q:=QaosTransitiveGroup("d8 abelian");
#I Retrieved 3 Transitive Groups.
#I (C) 2004-2005 QaoS developers <kantdb@math.tu-berlin.de>,
#I The Kant Project <kant@math.tu-berlin.de>
#I qaos--dev--1.0--patch-32
#I 2005-08-29 08:01:30 UTC
<collection from database: 3 transitive groups; "d8 abelian">
gap>
gap> QaosResult(Q);
[ <transitive group from database: C(8) = 8; id 39>,
  <transitive group from database: 4[x]2; id 40>,
  <transitive group from database: E(8) = 2[x]2[x]2; id 41> ]
```

To use the retrieved group objects in GAP you may use the *AsGroup* operation on a result.

`AsGroup` *database-group-object* [Function]
Return the GAP group object representing *database-group-object*.
If *database-group-object* cannot be turned into a group, return *fail*.

From the above example, let us GAP-ify the third group object.

```
gap> AsGroup(QaosResult(Q)[3]);
Group([ (1,8)(2,3)(4,5)(6,7), (1,3)(2,8)(4,6)(5,7), (1,5)(2,6)(3,7)(4,8) ])
```

For number field objects the corresponding operation is *AsField*.

`AsField` *database-field-object* [Function]
Return the GAP field object representing *database-field-object*.
If *database-field-object* cannot be turned into a field, return *fail*.

Consider following example for number fields (number fields of degree 5 with absolute discrimnant between 100000 and 120000 and non-trivial class group):

```
gap> R:=QaosNumberField("d5 |disc| >=100000 |disc| <=120000 cn>1");
#I Retrieved 18 Algebraic Number Fields.
#I (C) 2004-2005 QaoS developers <kantdb@math.tu-berlin.de>,
#I The Kant Project <kant@math.tu-berlin.de>
#I qaos--dev--1.0--patch-32
#I 2005-08-29 08:32:35 UTC
<collection from database: 18 number fields; "d5 |disc| >=100000 |disc| <=120000 cn
gap> QaosResult(R);
[ <number field from database: x_1^5+x_1^4-2*x_1^2+7*x_1+1; id 843712>,
```

```
  <number field from database: x_1^5+x_1^4+2*x_1^3-4*x_1^2+2*x_1-1; id 843791>,
  <number field from database: x_1^5+2*x_1^4-2*x_1^3+2*x_1^2+x_1+4; id 843841>,
  <number field from database: x_1^5+x_1^4+4*x_1^3+4*x_1-1; id 843892>,
  <number field from database: x_1^5+x_1^4+x_1^3+4*x_1^2+3*x_1-1; id 843894>,
  <number field from database: x_1^5+x_1^4-3*x_1^3-2*x_1^2+6*x_1-4; id 843897>,
  <number field from database: x_1^5+2*x_1^3+2*x_1^2+4*x_1+1; id 843930>,
  <number field from database: x_1^5+2*x_1^4+6*x_1^3+4*x_1^2+4*x_1-1; id 843955>,
  <number field from database: x_1^5+x_1^4-4*x_1^2+2*x_1+1; id 844004>,
  <number field from database: x_1^5+4*x_1^2+1; id 844011>,
  <number field from database: x_1^5+2*x_1^4+3*x_1^3+2*x_1^2-3*x_1+1; id 844036>,
  <number field from database: x_1^5+x_1^4+3*x_1^3+2*x_1^2+8*x_1-1; id 844043>,
  <number field from database: x_1^5+x_1^4+5*x_1^3+5*x_1^2+2*x_1+1; id 844062>,
  <number field from database: x_1^5+2*x_1^4-3*x_1^3+6*x_1^2-3*x_1+1; id 844066>,
  <number field from database: x_1^5+x_1^4-x_1^3-4*x_1^2+5*x_1-1; id 844082>,
  <number field from database: x_1^5+2*x_1^4-x_1^3-3*x_1^2+4*x_1+1; id 844107>,
  <number field from database: x_1^5+x_1^4+2*x_1^3+3*x_1^2-2*x_1+1; id 844110>,
  <number field from database: x_1^5+2*x_1^4-6*x_1^3+5*x_1^2-2*x_1+1; id 844145> ]
gap> AsField(QaosResult(R)[4]);
<field in characteristic 0>
```

4 Index

A

`AsField` 10
`AsGroup` 10

Q

`QaosDefaultLimit` 8
`QaosNumberField` 7
`QaosResult` 10
`QaosTransitiveGroup` 7