gStore0.9.1 Version User Guide

1.Update Log

1.1 Version Update Record

gStore 0.9.1 Version

- Update Time：2021-11-23
- Update function:
  - By separating gStore kernel parsing and execution, the query performance can be further improved through join order and other technologies, and the performance can be improved by more than 40% in complex queries;
  - Rewrite the http service component ghttp of gStore, and add user permissions, heartbeat detection, batch import, batch delete and other functions, and write a standard GHTTP API document (see interface list), further enrich the functions of GHTTP, improve the robustness of GHTTP;
  - Add Personalized PageRank (PPR) custom function, which can be used to calculate the relatedness between entities to find the node with the most influence;
  - Added support for arithmetic and logical operations in Filter statements, such as arithmetic operations (e.g.? x + ? Y = 5); Logical operations (e.g.? x + ? y = 5 && ? Y > 0);
  - The gServer component is added to realize two-way Socket API communication. Users can access gStore remotely through GHTTP component and gServer component;
  - The local operation command format is planned, and the --help command is introduced. Users can view the detailed command format of each function. For example, bin/gbuild -h/--help can view the detailed command format of gbuild;
  - Fixed a number of bugs.

gStore 0.9.0 Version

- Update Time：2021-02-10
- Update function：
  - Upgrade the SPARQL parser generator from ANTLR V3 to the latest, well-documented, and well-maintained V4;
  - Support writing numeric literals without data type suffixes in SPARQL queries;
  - Support for arithmetic and logical operators in SELECT clauses;
  - Supports aggregations of SUM, AVG, MIN, and MAX in the SELECT clause;
  - Additional support is built into filters, and function functions including `datatype`, `contains`, `ucase`, `lcase`, `strstarts`, `now`, `year`, `month`, `day`, and `abs`;
  - Supports path-related functions as an extension of SPARQL 1.1, including loop detection, shortest paths, and K-Hop reachability;
  - Supports full and incremental database backup and recovery. Administrators can enable automatic full backup;
  - Supports Log-based rollback operations;

    - Supports transactions with three levels of isolation: committed reads, snapshot isolation, and serializable ；
    - Expand the data structure to accommodate large-scale graphs of up to 5 billion triples.

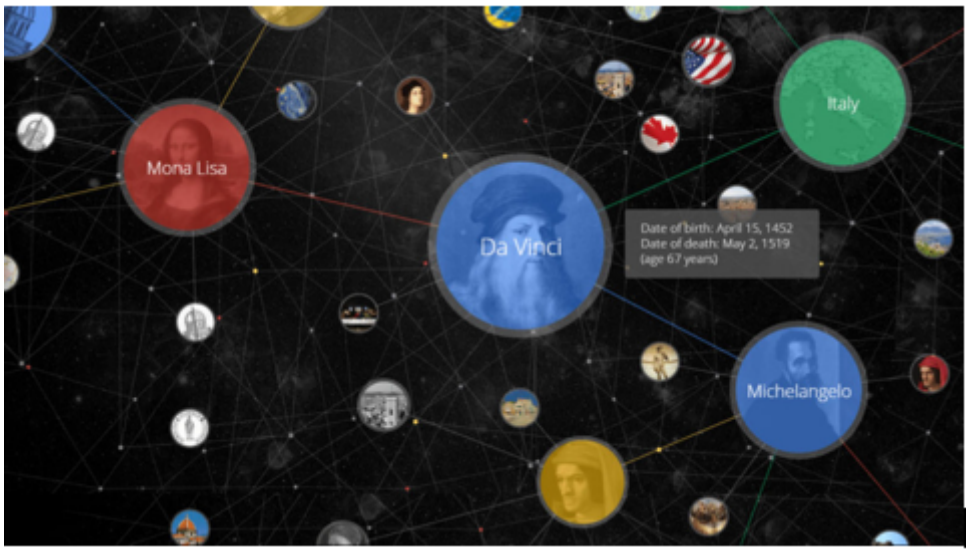
1.2 Version Update Record

- Modified the contents about Quick Start to match gStore 0.9.1 version
- Modify common API content to match gStore 0.9.1 version
- Modified Workbench console content to match gStore 0.9.1 version
- Add update log to record the version of graph database gStore and related document updates
- Added a document download directory for users to download documents

2. Introduction of Knowledge Graph and gStore

2.1 Introduction of Knowledge Graph

In recent years, with the revival of the concept of "artificial intelligence", in addition to the hot term "deep learning", "knowledge graph" is undoubtedly another "silver bullet" in the eyes of researchers, industry and investors. To put it simply, "knowledge Graph" is a data model that displays "entities", "attributes" and "relationships" among entities in the form of Graph. Below is an example from Google's knowledge Graph introduction page. There are four entities in the example, "Da Vinci", "Italian", "Monlarissa" and "Michelangelo". This diagram clearly shows the individual attributes and attribute values of "Da Vinci" (e.g., name, date of birth, time of death, etc.) as well as the relationships between them (e.g., Monlarissa is a painting by Da Vinci, Da Vinci was born in Italy, etc.).



At present, the knowledge graph generally uses RDF(Resource Description Framework) model in semantic Web Framework to represent data. The Semantic Web is a concept proposed by Tim Berners-Lee, the father of the World Wide Web, in 1998. Its core is to build a data-centric network, namely the Web of Data. RDF is the standard for data description in the SEMANTIC Web framework of W3C. It is often called RDF triples (Subject, predicate, object). Where the principal must be a described resource, represented by a URI. A predicate can represent an attribute of a subject or a relationship between a subject and an object. When representing an attribute, the object is the attribute value, usually a literal. Otherwise the object is another resource represented by a URI. The figure below shows a knowledge graph dataset for RDF triples of a people encyclopedia. For example: y Abraham_Lincoln said an entity URI prefix (y = http://en.wikipedia.org/wiki/), it has three properties (hasName, BornOdate DiedOnDate) and a relationship (DiedIn).

Prefix: y= http://en.wikipedia.org/wiki/

主体	属性	客体
y:Abraham_Lincoln	hasName	“Abraham Lincoln”
y:Abraham_Lincoln	BornOnDate	“1809-02-12”
y:Abraham_Lincoln	DiedOnDate	1865-04-15
y:Abraham_Lincoln	DiedIn	y:Washington_D.C
y:Washington_D.C	hasName	“Washington D.C.”
y:Washington_D.C	FoundYear	1790
y:Washington_D.C	rdf:type	y:city
y:United_States	hasName	“United States”
y:United_States	hasCapital	y:Washington_D.C
y:United_States	rdf:type	Country
y:Reese_Witherspoon	rdf:type	y:Actor
y:Reese_Witherspoon	BornOnDate	“1976-03-22”
y:Reese_Witherspoon	BornIn	y:New_Orleans,_Louisiana
y:Reese_Witherspoon	hasName	“ReeseWitherspoon”
y:New_Orleans,_Louisiana	FoundYear	1718
y:New_Orleans,_Louisiana	rdf:type	y:city
y:New_Orleans,_Louisiana	locatedIn	y:United_States

Figure 1-1 RDF data example

For RDF data set, W3C proposes a structured query language SPARQL. It is similar to SQL, the query language for relational databases. Like SQL, SPARQL is also a descriptive structured query language. That is, users only need to describe the information they want to query according to the syntax rules defined by SPARQL, without specifying the steps of the computer to perform the query. SPARQL became an official W3C standard in January 2008. The WHERE clause in SPARQL defines the query criteria, which are also represented by triples. We do not cover much syntax detail, but interested readers may refer to [1]. The following example illustrates the SPARQL language. Suppose we need to query the RDF data above for "the name of a person born on February 12, 1809 and who died on April 15, 1865?" This query can be represented as a SPARQL statement as shown below. The figure below shows a knowledge graph dataset for RDF triples of a people encyclopedia. For example: y Abraham_Lincoln said an entity URI prefix (y = http://en.wikipedia.org/wiki/), it has three properties (hasName, BornOdate DiedOnDate) and a relationship (DiedIn).

```
SELECT ?name                                    //查询返回的变量值
WHERE
{ ?m  <hasName>    ?name.                       //查询条件
 ?m <BornOnDate>  “1809-02-12” .
 ?m <DiedOnDate>  “1865-04-15” .
}
```

Figure 1-2 SPARQL Query example

A core problem of knowledge graph data management is how to efficiently store RDF datasets and quickly answer SPARQL queries. In general, there are two completely different sets of thinking. First, we can use existing mature database management systems (such as relational database systems) to store knowledge graph data. SPARQL queries oriented to RDF knowledge graph can be converted into queries oriented to such mature database management systems, such as SQL queries oriented to relational databases. Use existing relational database products or related technologies to answer queries. The core research problem is how to build relational tables to store RDF knowledge graph data and make the transformed SQL query statement query performance higher. The other is to directly develop a knowledge graph data storage and query system (Native RDF graph database system) for RDF knowledge graph data. Considering the characteristics of RDF knowledge graph management, optimization is carried out from the bottom of the database system.

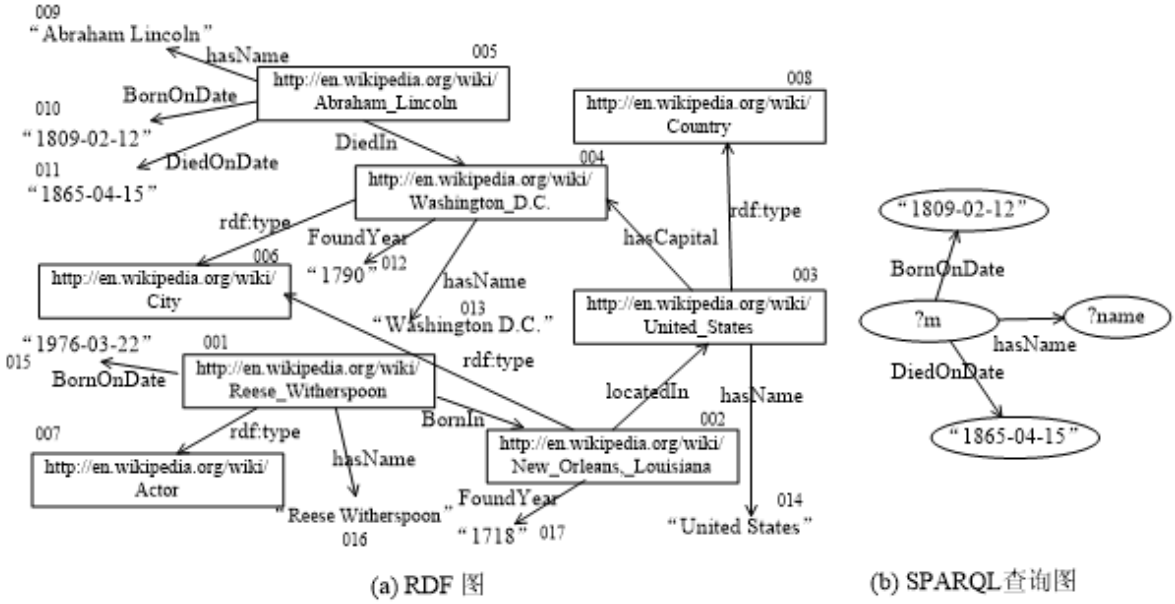


Figure 1-3 RDF graph and SPARQL query graph

2.2 Introduction of gStore

The gStore system developed by us belongs to the latter. gStore is an open source graph database system (commonly known as Triple Store) for RDF data model developed by data Management Laboratory (PKUMOD) of Wangxuan Institute of Computer Technology of Peking University after ten years. Different from traditional relational database based knowledge graph data management method , gStore **Native Graph Model**, maintains the **Graph structure of the original RDF knowledge graph**; Its data model is labeled, directed polygon graph, each vertex corresponds to a subject or object. We convert SPARQL queries for RDF to ** **Subgraph matching queries for RDF graphs**, the graph structure-based index (VS-tree) proposed by us is used to speed up query performance. Figure 1-3 shows the structure of the RDF graph and SPARQL query graph corresponding to the above example. Answering SPARQL queries is essentially finding the matching position of the subgraph of a SPARQL query graph in an RDF graph, which is the theoretical basis of answering SPARQL queries based on graph databases. In the example in Figure 1-3, the subgraph derived from nodes 005,009,010 and 011 is a match of the query graph, from which it is easy to know that the SPARQL query result is "Abraham Lincoln." For the core academic ideas of gStore, please refer to the published papers of Development Resources - Papers and Patents.

gStore begin withdata Management Laboratory (PKUMOD) of Wangxuan Institute of Computer Technology of Peking University (Lei Zou, Jinghui Mo, Lei Chen,M. Tamer Ozsu, Dongyan Zhao, gStore: Answering SPARQL Queries Via Subgraph Matching, Proc. VLDB 4(8): 482-493, 2011), VLDB 2011 paper by Prof. Lei Zou, Prof. Tamer Ozsu, University of Waterloo, and Prof. Lei Chen, Hong Kong University of Science and Technology , proposes a query execution scheme using subgraph matching to answer Basic Graph Pattern (BGP) statements in SPARQL. Since the publication of this paper, PKUMOD Laboratory has been continuously engaged in the open source, maintenance and system optimization of gStore system under the funding of the National Natural Science Foundation of China and the key RESEARCH and development projects of the Ministry of Science and Technology of China. At present, the open source gStore system on Github can support SPARQL 1.1 standard defined by W3C (see SPARQL Query Language for details).

After a series of tests, the results showed that gStore was faster than other database systems at answering complex queries (for example, containing circles). For simple queries, gStore and other database systems work fine. The standalone version of gStore can support more than **5 billion** RDF triples and SPARQL queries. The distributed system gStore (distributed version, not open source at present) has very good scalability. According to the test report given by "China Software Evaluation Center", The distributed gStore system has second query times on ten billion RDF triplet datasets.

Since the gStore system was opened on Github, BSD 3-clause, which is widely used in the open source community, has been adopted to promote the construction of gStore related knowledge graph technology ecology. According to this agreement, we require users to allow users to modify and redistribute codes freely on the premise of fully respecting the copyright of code authors, and also allow users to develop, distribute and sell commercial software freely on the basis of gStore codes. However, the above conditions must meet the relevant legal provisions stipulated in Chapter 10 "Legal Provisions" according to the BSD 3-clause open source agreement. We strictly require users to mark "powered by gStore" and the gStore logo (see gStore Logo for details) on the software they distribute based on the gStore code. We strongly recommend that users refer to the "Open Source and Legal Provisions" before using gStore.

3. Installation Instructions

3.1 System Requirements

Project	Requirement
Operating System	Linux, such as CentOS, Ubuntu etc.
Framework	x86_64
Disk Size	Depends on the size of the data set
Memory Size	Depends on the size of the data set
glibc	Must install version >= 2.14
gcc	Must install version >= 5.0
g++	Must install version >= 5.0
make	Must install
cmake	Must install
pkg-config	Must install
uuid	Must install
boost	Must install version >= 1.56 && <= 1.59
readline	Must install
readline-devel	Must install
libcurl-devel	Must install
openjdk	If using the Java API, yes
openjdk-devel	If using the Java API, yes
requests	If using Python http API, yes
node	If using Nodejs http api must install version >=10.9.0
curl-devel	If using php http API, yes
pthreads	If using php http API, yes
realpath	If using gconsole, yes
ccache	Optional, used to speed up compilation

3.2 Installation Environment

Running the corresponding scripts in scripts/setup/ for your operating system will automatically solve most of your problems for you. For example, if you are an Ubuntu user, you can execute the following command:

```
$ . scripts/setup/setup_ubuntu.sh
```

Before running the script, we recommend you install gcc and g++ 5.0 or later.

Of course, you can also choose to manually step prepare the environment; Detailed installation instructions for each system requirement are provided below.

3.2.1 gcc and g++ installation

Check g++ version:

```
$ g++ --version
```

If the version is earlier than 5.0, reinstall 5.0 or later version. Using 5.4.0 as an example :(for Ubuntu and CentOS)

```
$ wget http://ftp.tsukuba.wide.ad.jp/software/gcc/releases/gcc-5.4.0/gcc-
5.4.0.tar.gz
$ tar xvf gcc-5.4.0.tar.gz
$ cd gcc-5.4.0
$ ./contrib/download_prerequisites
$ cd ..
$ mkdir gcc-build-5.4.0
$ cd gcc-build-5.4.0
$ ../gcc-5.4.0/configure --prefix=/opt/gcc-5.4.0 --enable-checking=release --
enable-languages=c,c++ --disable-multilib
$ sudo make -j4   #Allows four compile commands to be executed simultaneously,
speeding up the compilation process
$ sudo make install
```

Ubuntu can also be installed directly using the following commands:

```
$ apt install -y gcc-5 g++-5
```

After successful installation,

- **You need to change the default versions of GCC and g++** If gcc and g++ 5.0 or later are installed in the '/prefix/bin' directory, run the following command:

  ```
  $ export PATH=/prefix:$PATH
  ```

- **The dynamic link library path needs to be modified:** If gcc and g++ dynamic link libraries above 5.0 are in the '/prefix/lib' path, you need to run the following comman:

  ```
  $ export LD_LIBRARY_PATH=/prefix/lib:$LD_LIBRARY_PATH
  ```

3.2.2 jdk installation

Check whether the JDK is installed

```
$ java -version
```

If not, install it

```
$ sudo yum install java-1.8.0-openjdk-devel.x86_64      #centos system
$ sudo apt install -y openjdk-8-jdk                     #ubuntu system
```

3.2.3 readline installation

Check whether readline is installed

```
$ yum list installed | grep readline    #centos system$ dpkg -s readline
#ubuntu system
```

if not, install it

```
$ sudo yum install readline-devel       #centos system$ sudo apt install -y
libreadline-dev #ubuntu system
```

3.2.4 boost installation (Please use 1.56-1.59)

Check whether boost is installed

```
$ yum list installed | grep boost       #centos system$ dpkg -s boost
       #ubuntu system
```

If not, install it: (use version 1.56.0 as example)

version:1.56.0

address: http://sourceforge.net/projects/boost/files/boost/1.56.0/boost_1_56_0.tar.gz

Installation script: (for CentOS and Ubuntu)

```
$ wget
http://sourceforge.net/projects/boost/files/boost/1.56.0/boost_1_56_0.tar.gz$ tar
 -xzvf boost_1_56_0.tar.gz$ cd boost_1_56_0$ ./bootstrap.sh$ sudo ./b2$ sudo
./b2 install
```

Ubuntu can also be installed directly using the following commands:

```
$ sudo apt install -y libboost-all-dev
```

Note: please install boost after ensuring that the g++ version is above 5.0 Undefined reference to 'boost::...' - undefined reference to 'boost::... '"), most likely because you compiled Boost with GCC versions lower than 5.0. At this point, recompile Boost using the following step:

- Clear old files: `./b2 --clean-all`
- In the user-config.jam file under./tools/build/ SRC (if this file does not exist under this path, Please find a sample user-config.jam file under./tools/build/example or some other directory

and copy it to./tools/build/ SRC) to add ： `using gcc : 5.4.0 : gcc-5.4.0's path ;`
- Run under ./ `./bootstrap.sh --with-toolset=gcc`
- `sudo ./b2 install --with-toolset=gcc`

Then recompile gStore (please start from 'make pre')

After successful installation，

- **Need to modify the dynamic link library path:** Assuming boost's dynamically linked library is in the '/prefix/lib' path, you need to execute the following command：

```
$ export LD_LIBRARY_PATH=/prefix/lib:$LD_LIBRARY_PATH
```

- **The header file path needs to be changed:** Assuming boost's header file is in the '/prefix/include' path, you need to execute the following command:

```
$ export CPATH=/prefix/include:$CPATH
```

3.2.5 curl installation

Check whether curl is installed

```
$ curl --version            #centos system$ curl --version          #ubuntu
system
```

if not, then install：

version: 7.55.1

address: https://curl.haxx.se/download/curl-7.55.1.tar.gz

Installation scripts (for CentOS and Ubuntu)

```
$ wget https://curl.haxx.se/download/curl-7.55.1.tar.gz$ tar -xzvf  curl-
7.55.1.tar.gz$ cd curl-7.55.1$ ./configure$ make$ make install
```

Or use the following command to install

```
$ sudo yum install -y libcurl-devel libcurl-dev                    #centos
system$ sudo apt install -y curl libcurl4 libcurl4-openssl-dev  #ubuntu system
```

3.2.6 cmake installation

Check whether cmake is installed

```
$ cmake --version              #centos system$ cmake --version
#ubuntu system
```

if not, install:

version: 3.6.2

address: https://curl.haxx.se/download/curl-7.55.1.tar.gz

Installation scripts (for CentOS and Ubuntu)

```
$ wget https://cmake.org/files/v3.6/cmake-3.6.2.tar.gz$ tar -xvf cmake-
3.6.2.tar.gz && cd cmake-3.6.2/$ ./bootstrap$ make$ make install
```

Ubuntu can also be installed directly using the following commands:

```
$ sudo apt install -y cmake
```

3.2.7 pkg-config installation

Check whether pkg-config is installed

```
$ pkg-config --version      #centos system$ pkg-config --version        #ubuntu
system
```

if not, install

```
$ sudo yum install pkgconfig.x86_64         #centos system$ sudo apt install -y
pkg-config            #ubuntu system
```

3.2.8 uuid installation

Check whether uuid is installed

```
$ uuid -m      #centos system$ uuid -m      #ubuntu system
```

if not, install

```
$ sudo yum install libuuid-devel        #centos system$ sudo apt install -y
uuid-dev            #ubuntu sysyem
```

3.2.9 zip/unzip installation

Decompress the Store ZIP package

Check whether zip/unzip is installed

```
$ yum list installed | grep unzip       #centos system$ dpkg -s unzip
#ubuntu system
```

if not, install

```
$ sudo yum install -y unzip zip         #centos system$ sudo apt-get install
unzip zip      #ubuntu system
```

3.3 gStore obtain

If you encounter permission issues, prefix the command with `sudo`

3.3.1 method 1：download

open https://github.com/pkumod/gStore , download gStore.zip; decompress zip package。

gStore has been uploaded to gitee (code cloud), which is recommended for faster download for mainland China users. The website is https://gitee.com/PKUMOD/gStore

3.3.2 method 2：clone(recommend)

Run the following code to clone：

```
$ git clone https://github.com/pkumod/gStore.git  #github$ git clone
https://gitee.com/PKUMOD/gStore.git #gitee(code cloud) faster for mainland China
user.
```

Note: Git need to be installed first 。

```
$ sudo yum install git      #centos system$ sudo apt-get install git    #ubuntu
system
```

3.4 gStore compile

Switch to the gStore directory:

```
$ cd gStore
```

Execute the following command:

```
$ make pre$ make            #If the compilation completes successfully, it will
appear at the end: Compilation ends successfully! result$ bin/ginit
#Initialization, if successfully completed, will finally appear: system.db is
built successfully! result
```

If 'make pre' still asks for gcc 5.0 or later after installing gcc 5.0 or later, locate gcc 5.0 or later and run the following command in the gStore directory ：

```
$ export CXX=<5.0 or later gcc's path>
```

Then `make pre` again. If the same error is reported after this step, manually delete the `CMakeCache. TXT` and `CMakeFiles` folders under `tools/ antlr4-cpp-Run-time 4/` and `make pre` again.

3.5 Deploy gStore using Docker

> We provide two ways to deploy gStore from containers:
>
> One is to build it yourself from the Dockerfile file in the project root and then run the container.
>
> Another option is to download the image that has been automatically built and run it directly.

3.5.1 environment preparation

Docker, refer to the address [docker]

3.5.2 Run by pulling the image directly(recommend)

No need to download the project or build your own, just enter `sudo docker pull pkumod/gstore:latest` pull images that have been automatically built on the Docker Hub。After pulling, `sudo docker run -p 9000:80 -it pkumod/gstore:latest` It can be directly started and used in container.

3.5.3 Build the image from Dockerfile

Waiting for adjustment

There are probably a lot of other things that need to be added, so for now I've just added a basic version. The basic environment build is just the first step in containerization

4. Quick Start

4.1 Data Format

`gStore` is a graph database engine based on the RDF model, and its data format follows the RDF model. RDF, the W3C standard for describing real-world resources, is a general way to describe information so that it can be read and understood by computer applications. Any entity in the real world can be represented as a resource in the RDF model, such as a book's title, author, modification date, content, and copyright information. These resources can be used to abstract concepts, entities, and events from the objective world in the knowledge graph. An attribute of each resource and its attribute value, or its relationship to other resources, is called a piece of knowledge. Properties and relationships can be represented as triples.

A triad consists of three elements: Subject, Property 1, and Object. It usually describes the relationship between two resources or some Property of a resource. When a triple describes the attributes of a resource, its three elements are also called body, attribute, and Property Value. For example, the triad "Aristotle, Birthplace, Chalcis" expresses the fact that Aristotle was born in Chalcis.

Using these properties and relationships, a large number of resources can be linked together to form a large RDF knowledge graph dataset. Therefore, a knowledge graph can often be viewed as a collection of triples. These collections of triples, in turn, form an RDF dataset. The triplet set of knowledge graph can be stored in relational database or graph database.

RDF data should be provided in n-triple format (XML is not currently supported), and queries must be provided in SPARQL1.1 syntax. The following is an example of the n-triple format file:

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
_:a  foaf:name   "Johnny Lee Outlaw" .
_:a  foaf:mbox   <mailto:jlow@example.com> .
_:b  foaf:name   "Peter Goodguy" .
_:b  foaf:mbox   <mailto:peter@example.org> .
_:c  foaf:mbox   <mailto:carol@example.org> .
```

Triples are typically stored in the W3C-defined NT file format and represent three RDF data, where the values wrapped in `<` and `>` are urls of an entity, and the values wrapped in '"' are literals representing the value of an attribute of the entity, followed by'`^^` to indicate the type of the value. The following three RDF data points represent two attributes of `John`, `gender` and `age`, with values of `male` and `28` respectively. The last one indicates that `John` and `Li` have a `friend` relationship.

```
<John> <gender> "male"^^<http://www.w3.org/2001/XMLSchema#String>.
<John> <age> "28"^^<http://www.w3.org/2001/XMLSchema#Int>.
<John> <friend> <Li>.
```

More specific information about N-Triple please check N-Triple. Not all syntax in SPARQL1.1 is parsed and answered in gStore; for example, property paths are beyond the capabilities of the gStore system.

4.2 Initialize the system database

As soon as you download and compile the code for the gStore system, a database named System (the real directory name system.db) is automatically created. This is the database that manages system statistics, including all users and all databases. You can query this database using the gquery command, but do not modify it using the editor.

The `system` database is the built-in system database of gStore. This database cannot be deleted. It is used to store the information related to the system, especially the information of the built database

4.2.1 Command line mode(ginit)

ginit is used to initialize the database

instruction:

```
bin/ginit -db [db_name1],[db_name2],[...]
```

command parameter:

```
db_name1: database name
```

If no database names are written, the reinitialized `system` database will have no other database information

example:

```
[root@localhost gStore]$ bin/ginit -db lubm
================================================================================
UPDATE
Insert:
{
    <system>    <built_time>    "2021-02-21 22:50:05".
    <lubm>  <database_status>   "already_built".
    <lubm>  <built_by>  <root>.
    <lubm>  <built_time>    "2021-02-21 22:50:05".
}
================================================================================
parse query  successfully! .
unlock the query_parse_lock .
after Parsing, used 96ms.
write priviledge of update lock acquired
QueryCache cleared
Total time used: 97ms.
update num : 4
system.db is built successfully!
```

4.3 Create database

Creating a database is one of the most important operations in gStore. It is also the first operation that users need to perform after gStore installation. gStore provides multiple methods to create a database.

4.3.1 Command line mode (gbuild)

The gbuild command is used to create a new database from an RDF file:

```
bin/gbuild -db dbname -f filename
```

Parameter definition:

```
dbname: database name
filename: filepath of files with ".nt" or ".n3" as suffix
```

For example, we build a database called "lubm.db" from lubm.nt, which can be found in the data folder.

```
[root@localhost gStore]$ bin/gbuild -db lubm -f ./data/lubm/lubm.nt
gbuild...
argc: 3 DB_store:lubm      RDF_data: ./data/lubm/lubm.nt
begin encode RDF from : ./data/lubm/lubm.nt ...
```

Note:

- You cannot create a database from an empty RDF dataset
- Note that you cannot directly `CD` into the `bin` directory. Instead, you need to perform the gbuild operation in the gStore installation root director

4.3.2 Visual tool (gWorkbench)

gWorkbench is a visual management tool of gStore. Through gWorkbench, you can connect to gStore and create a graph database through database management module. For details, see [Development document] - [Visual Tool Workbench] - [query function] - [database management] function].

4.3.3 HTTP API (ghttp)

gStore provides GHTTP component as HTTP API service component. Users can realize relevant functions by sending HTTP requests to ghttp In ghttp, graph database is constructed by `build` request.

4.3.4 Socket API (gServer)

gStore provides gServer component as Socket API service component. Users can realize related functions by sending Socket request to gServer. gServer builds graph database through `build` request. See [Development document] - [Common API] - [gServer interface Description] for details.

4.4 Database list

The database list function is used to obtain information about all available databases in the following formats

4.4.1 Command line mode (gshow)

gshow is used to get the list of all available databases.

instruction:

```
bin/gshow
```

example:

```
[root@localhost gStore]$ bin/gshow
==========================================
database: system
creator: root
built_time: "2019-07-28 10:26:00"
==========================================
database: lubm
creator: root
built_time: "2019-07-28 10:27:24"
```

4.4.2 Visual tool (gWorkbench)

gWorkbench is a visual management tool of gStore. Through gWorkbench, you can connect to gStore and create a graph database through database management module. For details, see [Development document] - [Visual Tool Workbench] - [query function] - [database management] function].

4.4.3 HTTP API (ghttp)

gStore provides ghttp component as http API service component, users can send http request to ghttp to achieve relevant functions, ghttp through the `show` command to achieve relevant functions, details see [development document] - [common API] - [ghttp interface description].

4.4.4 Socket API (gServer)

gStore provides gServer component as Socket API service component. Users can realize related functions by sending Socket request to gServer. gServer displays database list through `show` request. See [Development document] - [Common API] - [gServer interface Description] for details.

4.5 Database status Query

The database status query function is used to obtain statistics about a specified database in the following ways.

4.5.1 Command line mode (gmonitor)

gmonitor is used to get statistics about a specified database.

instruction:

```
bin/gmonitor -db db_name
```

parameter definition:

```
db_name: database name
```

example:

```
[root@localhost gStore]$ bin/gmonitor -db lubm
database: lubm
creator: root
built_time: "2019-07-28 10:27:24"
triple num: 99550
entity num: 28413
literal num: 0
subject num: 14569
predicate num: 17
```

4.5.2 Visual tool（gWorkbench)

gWorkbench is a visual management tool of gStore. Through gWorkbench, you can connect to gStore and create a graph database through database management module. For details, see [Development document] - [Visual Tool Workbench] - [query function] - [database management] function].

4.5.3 HTTP API（ghttp)

gStore provides ghttp component as http API service component. Users can realize relevant functions by sending http requests to ghttp. In ghttp, database statistics can be obtained through `monitor`.

4.6 Database query

Database query is one of the most important functions of gStore. gStore supports SPARQL 1.1 query language defined by W3C. Users can use the gStore database query function in the following ways.

4.6.1 Command line mdoe（gquery)

gquery is used to query existing databases using files containing SPARQL queries. (Each file contains an exact SPARQL statement. SPARQL statements can be used not only for queries, but also for additions and deletions. For details on SPARQL statements, see Chapter 8.)

1.To query the database named db name, enter the following command:

```
bin/gquery -db db_name -q query_file
```

parameter definition：

```
db_name: database name
query_file: The path to the SPARQL statement ending with ".sql "(other name
extensions are acceptable)
```

For example, we execute the SPARQL statement in./data/lubm/ lubM_q0.sql to query the LUBM database.

The query result is：

```
[root@localhost gStore]$ bin/gquery -db lubm -f ./data/lubm/lubm_q0.sql
There has answer: 15
final result is :
?x
<http://www.Department0.University0.edu/FullProfessor0>
<http://www.Department1.University0.edu/FullProfessor0>
<http://www.Department2.University0.edu/FullProfessor0>
<http://www.Department3.University0.edu/FullProfessor0>
<http://www.Department4.University0.edu/FullProfessor0>
<http://www.Department5.University0.edu/FullProfessor0>
<http://www.Department6.University0.edu/FullProfessor0>
<http://www.Department7.University0.edu/FullProfessor0>
<http://www.Department8.University0.edu/FullProfessor0>
<http://www.Department9.University0.edu/FullProfessor0>
<http://www.Department10.University0.edu/FullProfessor0>
<http://www.Department11.University0.edu/FullProfessor0>
<http://www.Department12.University0.edu/FullProfessor0>
<http://www.Department13.University0.edu/FullProfessor0>
<http://www.Department14.University0.edu/FullProfessor0>
```

2.To learn more about how to use gQuery, enter the following command:

```
bin/gquery --help
```

3. Command to enter gquery console:

```
bin/gquery -db dbname
```

he program displays a command prompt (" gsql > ") where you can enter command:

> Use `help` to see the basic information for all commands

> Enter `quit` to exit the gquery console

> For the `sparql` command, use `sparQL query_file` to execute the SPARQL query statement, and query_file indicates the path of the file storing the SPARQL statement. When the program finishes answering the query, it displays the command prompt again
>
> We use lubm.nt as example。

```
(base) [root@iz8vb0u9hafhzz1mn5xcklz gStore]# bin/gquery -db lubm

gsql>sparql ./data/lubm/lubm_q0.sql
... ...
Total time used: 4ms.
final result is :
<http://www.Department0.University0.edu/FullProfessor0>
<http://www.Department1.University0.edu/FullProfessor0>
<http://www.Department2.University0.edu/FullProfessor0>
<http://www.Department3.University0.edu/FullProfessor0>
<http://www.Department4.University0.edu/FullProfessor0>
<http://www.Department5.University0.edu/FullProfessor0>
<http://www.Department6.University0.edu/FullProfessor0>
<http://www.Department7.University0.edu/FullProfessor0>
<http://www.Department8.University0.edu/FullProfessor0>
<http://www.Department9.University0.edu/FullProfessor0>
<http://www.Department10.University0.edu/FullProfessor0>
<http://www.Department11.University0.edu/FullProfessor0>
<http://www.Department12.University0.edu/FullProfessor0>
<http://www.Department13.University0.edu/FullProfessor0>
<http://www.Department14.University0.edu/FullProfessor0>

gsql>help
help - print commands message
quit - quit the console normally
sparql - load query from the second argument

gsql>quit
```

- Note:

  - If there is no answer, "[Empty Result]" is printed with a blank line following all results.

  - Use readline lib, so you can use the arrow keys in the keyboard to view the command history, and use the and arrow keys to move and modify the entire command.

  - Utility supports path completion. (Not built-in command completion)

  - Note that you cannot directly 'CD' into the 'bin' directory, but rather perform the 'gquery' operation in the gStore installation root directory

4.6.2 Visual tool (gWorkbench)

gWorkbench is a visual management tool of gStore. Through gWorkbench, you can connect to gStore and create a graph database through database management module. For details, see [Development document] - [Visual Tool Workbench] - [query function] - [database management] function].

4.6.3 HTTP API（ghttp)

gStore provides ghttp component as http API service component, users can realize related functions by sending http request to ghttp, ghttp through `query` request to query graph database, including query, delete, insert, See [Development document] - [common API] - [ghttp interface Description] for details.

4.6.4 Socket API（gServer)

gStore provides gServer component as Socket API service component. Users can realize related functions by sending Socket request to gServer. gServer can query graph database through `query` request, including query, delete and insert. See [Development document] - [Common API] - [gServer interface Description] for details.

4.7 Database export

The export database function enables you to export a database as an. Nt file. There are three forms:

4.7.1 Command line mode (gexport)

gexport is used to export database.

instruction:

```
bin/gexport -db db_name -f path
```

parameter definition:

```
db_name: database name
path: Export to specified folder (if empty, export to gStore root by default)
```

example:

```
[root@localhost gStore]# bin/gexport -db lubm
after Handle, used 0 ms.
QueryCache didn't cache
after tryCache, used 0 ms.
in getFinal Result the first half use 0  ms
after getFinalResult, used 0ms.
Total time used: 1ms.
finish exporting the database.
```

4.7.2 Visual tool (gWorkbench)

gWorkbench is a visual management tool of gStore. Through gWorkbench, you can connect to gStore and create a graph database through database management module. For details, see [Development document] - [Visual Tool Workbench] - [query function] - [database management] function].

4.7.3 HTTP API (ghttp)

gStore provides ghttp component as httpAPI service component. Users can realize relevant functions by sending http requests to ghttp. The function of `export` is adopted in ghttp.

4.8 Database deletion

Deleting a database You can delete a database in the following three ways

4.8.1 Command line mode (gdrop)

gdrop is use to delete database

instruction:

```
bin/gdrop -db db_name
```

parameter definition:

```
db_name: database name
```

example:

```
[root@localhost gStore]$ bin/drop -db lubm2
after tryCache, used 0 ms.
QueryCache cleared
Total time used: 97ms.
update num : 3
lubm2.db is dropped successfully!
```

To delete the database, you should not just type `rm -r db_name. Db` because this will not update the built-in database named `system`. Instead, you should type `bin/ gdrop-db db_name`.

4.8.2 Visual tool（gWorkbench)

gWorkbench is a visual management tool of gStore. Through gWorkbench, you can connect to gStore and create a graph database through database management module. For details, see [Development document] - [Visual Tool Workbench] - [query function] - [database management] function].

4.8.3 HTTP API（ghttp)

gStore provides ghttp component as http API service component, users can send http request to ghttp to achieve relevant functions, ghttp through the `drop` command to achieve relevant functions, details see [development document] - [common API] - [ghttp interface description]

4.8.4 Socket API（gServer)

gStore provides gServer component as Socket API service component. Users can realize related functions by sending Socket request to gServer. gServer can delete graph database by `drop` request. See [Development document] - [Common API] - [gServer interface Description] for details.

4.9 Additional data

Inserting RDF data is a routine gStore operation, and you can do it in one of the following ways.

4.9.1 Command line mode (gadd)--file

gadd is use to insert triples from a file into an existing database.

instruction:

```
bin/gadd -db db_name -f rdf_triple_file_path
```

parameter definition:

```
db_name: database name
rdf_triple_file_path: The file path with the suffix ".nt" or ".n3"
```

example:

```
[bookug@localhost gStore]$ bin/gadd -db lubm -f ./data/lubm/lubm.nt
...
argc: 3 DB_store:lubm   insert file:./data/lubm/lubm.nt
get important pre ID
...
insert rdf triples done.
inserted triples num: 99550
```

** Note: **

1. Gadd is primarily used for data insertion into RDF files

2. Do not go directly to the 'bin' directory. Instead, perform the 'gadd' operation in the gStore installation root directory

4.9.2 Command line mode (gquery)---SPARQL statement

SPARQL can be defined by `insert data` instruction to insert data. Based on this principle, users can also write SPARQL insert statements and then use gStore's `gQuery` tool to insert data. Examples of SPARQL insert statements are as follows:

```
insert data {
 <张三> <性别> "男"^^<http://www.w3.org/2001/XMLSchema#String>.
 <张三> <年龄> "28"^^<http://www.w3.org/2001/XMLSchema#Int>.
 <张三> <好友> <李四>.
}
```

Multiple RDF data can be contained with `{}`, taking care that each one ends with a `.`

Since the database query function can be used to insert data, the following functions can also be used to insert data.

4.9.3 Visual tool （gWorkbench)

gWorkbench is a visual management tool of gStore. Through gWorkbench, you can connect to gStore and create a graph database through database management module. For details, see [Development document] - [Visual Tool Workbench] - [query function] - [database management] function].

4.9.4 HTTP API（ghttp)

gStore provides ghttp component as AN http API service component. Users can realize relevant functions by sending http requests to ghttp. ghttp inserts data through `query` request and batch inserts data through `batchInsert`. See [Development document] - [common API] - [ghttp interface Description] for details.

4.9.5 Socket API（gServer)

gStore provides gServer component as Socket API service component. Users can implement related functions by sending Socket requests to gServer. gServer inserts data through 'query' requests. See [Development document] - [Common API] - [gServer interface Description] for details.

4.10 Data deletion

Deleting RDF data is a routine gStore operation, and you can perform this operation in the following ways.

4.10.1 Command line mode（gsub）--file deletion

gsub is used to remove triples from files in an existing database.

instruction：

```
bin/gsub db_name rdf_triple_file_path
```

parameter definition：

```
rdf_triple_file_path: The path to the data file to be deleted with the suffix
".nt" or ".n3"
```

example：

```
[root@localhost gStore]$ bin/gsub lubm ./data/lubm/lubm.nt
...
argc: 3 DB_store:lubm  remove file: ./data/lubm/lubm.nt
...
remove rdf triples done.
removed triples num: 99550
```

4.10.2 Command line mode（gquery)---SPARQL statement

SPARQL can be defined by 'delete data' instruction to insert data. Based on this principle, users can also write SPARQL insert statements and then use gStore's `gQuery` tool to insert data. Examples of SPARQL insert statements are as follows:

```
delete data {
 <张三> <性别> "男"^^<http://www.w3.org/2001/XMLSchema#String>.
<张三> <年龄> "28"^^<http://www.w3.org/2001/XMLSchema#Int>.
<张三> <好友> <李四>.
}
```

Multiple RDF data can be contained with '{}', taking care that each one ends with a `.`

SPARQL also allows you to delete data based on a subquery structure using the 'DELETE WHERE' statement, as shown below

```
delete where
{
   <张三> ?x ?y.
}
```

This statement deletes all information (including attributes and relationships) about an entity

Since the database query function can be used to insert data, the following functions can also be used to insert data

4.10.3 Visual tool（gWorkbench)

gWorkbench is a visual management tool for gStore. Through gWorkbench, you can connect to gStore and delete data by writing SPARQL statements through database query module. See [Development document] - [Visualization Tool Workbench] - [query function] - [Graph database query] for details.

4.10.4 HTTP API（ghttp)

gStore provides ghttp component as http API service component, users can send http request toghttp to achieve relevant functions, ghttp through `query` request to delete data and `batchRemove` to delete data in batches. See [Development document] - [common API] - [ghttp interface Description] for details

4.10.5 Socket API（gServer)

gStore provides gServer component as Socket API service component. Users can realize related functions by sending Socket requests to gServer. gServer can delete data through `query` requests. See [Development document] - [Common API] - [gServer interface Description] for details.

4.11 HTTP API service

ghttp is an external access interface provided by gStore. It is an http API service. Users can remotely connect to and operate gStore by sending http requests to ghttp

4.11.1 Start ghttp service

After gStore is compiled, a ghttp service is displayed in the bin directory of gStore. However, the ghttp service is not started by default. You need to manually start the ghttp service:

```
bin/ghttp -db db_name -p serverPort
```

parameter definition:

```
db_name: The name of the database to start GHTTP (optional, if the default value
is not set, the default value is system database. The main function of this
parameter is that when starting GHTTP, the related information of the database
will be loaded into the memory. If other databases are queried later, the server
will automatically load the required database.

serverPort: GHTTP listening port. You need to manually specify this port and
ensure that it is not prohibited by the server firewall (optional, if this port
is not specified, the default port number is 9000).
```

`ghttp` supports GET and POST request types.

`ghttp` supports concurrent read-only queries, but when a query containing updates arrives, the entire database is locked. On a computer with dozens of kernel threads, the recommended number of queries to run concurrently is less than 300, but we were able to run 13,000 queries simultaneously in our experiment. To use the concurrency feature, it is best to change the system Settings for Open File and Maximum Process to 65535 or greater

You'd better send the 'checkpoint' command 'GHTTP' to the console often if you send a query `ghttp` containing updates. Otherwise, updates may not sync with disk, and will be lost if the 'ghttp' server stops abnormally (for example, type 'Ctrl + C')

4.11.2 Shutdown ghttp service

Shut down ghttp service please use the following command to shut down ghttp service. It is better not to just enter command `Ctrl + C` or `kill` to stop ghttp service, because it is not safe.

```
bin/shutdown -p serverPort
```

parameter definition:

```
serverPort: Port set when GHTTP starts (default 9000 if no port is set when GHTTP
starts)
```

4.11.3 HTTP API（ghttp)

ghttp provides a rich API interface so that users can remotely operate most of the functions of gStore. For details, see [Development document] - [common API] - [ghttp interface Description]

4.12 Socket API service

gServer is an external access interface provided by gStore and a Socket API service. Users can connect and operate gStore remotely through two-way socket communication.

4.12.1 Start gServer service

After gStore is compiled, a gServer service is displayed in the bin directory of gStore. However, the gServer service is not started by default. You need to manually start the gServer service:

```
bin/gserver -s
```

Other optional parameters:

```
-t,--stop: shutdown gserver service;
-r,--restart: restart gserver service;
-p,--port: Modify the socket connection port configuration. The default port is
9000. After the modification, restart the GServer service
-P,--printport: pritn current socket connection port configuration
-d,--debug: start debug mode（keep gserver service running in the foreground）
-k,--kill: Forcibly stop the service. You are advised to stop the service only
when the service cannot be stopped
```

4.12.2 Shutdown gServer service

To stop the gServer service, use the following command to stop it. It is better not to simply enter the command 'Ctrl + C' or 'kill' to stop the gServer, because it is not safe.

```
bin/gserver -t
```

4.12.3 gServer related API

gServer provides rich API interfaces so that users can remotely operate most functions of gStore. See [Development document] - [common API] - [gServer interface Description] for specific interfaces.

5. API Usage

5.1 API instruction

gStore provides API services to users through http and Socket services, and its components are ghttp and gServer.

5.1.1 HTTP API instruction

We now provide c++, Java, python, PHP, and nodejs apis for GHTTP. Please refer to the examples in `API/HTTP/CPP`, `API/HTTP/Java`, `API/HTTP /python`, `API/HTTP/PHP` and `API/HTTP /nodejs`. To use these examples, make sure you have generated an executable file. ** Next, use the `bin/ghttp` command to start the ghttp service. ** If you know of a running available GHTTP server and try to connect to it, that's ok too. Then, for c++ and Java code, you need to compile the sample code in the directory `API/HTTP/CPP /example` and `API/HTTP/Java /example`.

For details on how to start and shut down ghttp, see [development documentation] - [Quick Start] - [http API service]

After the API is started, the ghttp access address is as follow:

```
http://serverip:port/
```

`ServerIP` is the IP address of the gStore server, and `port` is the port on which ghttp is started.

5.1.2 Socker API instruction

We now provide c++, Java (python, PHP, nodejs) apis for gserver. See API /socket/ CPP, API /socket/ Java for example code. To use these examples, make sure you have generated an executable file. **Next, use the `bin/gserver -s` command to start the gserver service.**If you know of a running available gServer and try to connect to it, this is also ok. Then, for c++ and Java code, you need to compile the sample code in the directory `API/HTTP/CPP /example` and `API/HTTP/Java /example`.

Start and close gServer

After the Socket API is started, you can connect through the Socket. The default port of gServer is 9000

5.2 HTTP API framework

The gStore HTTP API is placed in the API/HTTP directory of the gStore root directory and contains the following contents:

- gStore/api/http/
  - cpp/ (the C++ API)
    - client.cpp (C++ API's source code)
    - client.h
    - example/ (Example program using C++ API)
      - GET-example.cpp
      - Benchmark.cpp
      - POST-example.cpp
      - Makefile
    - Makefile (Compile and build the lib)
  - java/ (the Java API)
    - client.java
    - lib/
    - src/
      - Makefile
      - jgsc/
        - GstoreConnector.java (Java API's source code)
    - example/ (A sample program using the Java API)
      - Benckmark.java

        ```
        - GETexample.java
        ```

      - POSTexample.java
      - Makefile
    - python/ (the Python API)
      - example/ (A sample program using the Python API)
        - Benchmark.py
        - GET-example.py
        - POST-example.py
      - src/
        - GstoreConnector.py (A sample program using the Python API)
    - nodejs/ (the Nodejs API)
      - GstoreConnector.js (Nodejs API's source code)
      - LICENSE
      - package.json
      - README.md
      - example/ (A sample program using the Nodejs API)
        - POST-example.js

        - GET-example.js
    - php/ (the Php API)
        - example/ (A sample program using the php API)
            - Benchmark.php
            - POST-example.php
            - GET-example.php
        - src/
            - GstoreConnector.php (php API's source code)
- -

5.3 ghttp API instruction

5.3.1API Interconnection Mode

> The ghttp interface adopts the `HTTP` protocol and supports multiple ways to access the interface. If the ghttp is started on the port `9000`, the interface interconnection content is as follows
>
> API address:
>
> ```
> http://ip:9000/
> ```
>
> The interface supports both `GET` and `POST` requests, where `GET` requests place parameters in the URL and `POST` requests place parameters in the `body` request.

> **Note: `GET` request parameters contain special characters, such as? , @,& and other characters, you need to use urlencode encoding, especially the `SPARQL` parameter must be encoded**

5.3.2 API list

API name	Definition	Note
build	build graph database	The database file must be locally stored on the server
load	load graph database	Load the database into memory
unload	unload graph database	Unload the database from memory
monitor	monitor graph database	Count information about the specified database (such as the number of triples, etc.)
drop	drop graph database	Logical deletion and physical deletion can be performed
show	display graph database	Display a list of all databases
usermanage（added）	user management	Add, delete, or modify user information
showuser	display all user list	Display a list of all users
userprivilegemanage(added)	user privilege management	Add, delete, or modify user's privilege information
backup	backup database	backup database information
restore	restore database	restore database information
query	query database	Including query, delete, and insert
export	export database	Export database as NT file
login	login to database	It is used to authenticate user names and passwords
check（rewrite）	Detect ghttp heartbeat signal	
init（abandon）	Initialize system	This operation should not be initialized by ghttp
refresh（abandon）	reload database	This can be done by unload+ Load

API name	Definition	Note
parameter（abandon）	Set parameters for backup	All parameters are currently boiled down to the config.ini file and cannot be changed remotely
begin	Start transaction	Transaction starts and needs to be used in conjunction with TQuery
tquery	Querying the database (with transactions)	Data queries with transaction mode (insert and DELETE only)
commit	commit transactions	Commit the transaction after it completes
rollback	rollback transaction	Roll back the transaction to begin state
txnlog	Obtain transaction log information	Return transcation log information as json
checkpoint	write data to a disk	After an INSERT or delete operation is performed on the database, manually checkpoint is required
testConnect	testing connectivity	Used to check whether GHTTP is connected
getCoreVersion	get gStore version	Get the gStore version number
batchInsert	batch insert data	Batch insert NT data
batchRemove	batch delete data	Batch delete NT data
querylog（added）	Get query log information	Query log information is returned as json
ipmanage（added）	black/white list management	Maintains a blacklist and whitelist of IP addresses that access gStore

5.3.3 API specific intruction

> The input and output parameters of each interface are specified in this section. Assume that the IP address of the GHTTP server is 127.0.0.1 and the port is 9000

5.3.3.1 build -build datbase

Brief description

- Create a database based on existing NT files
- Files must exist on the gStore server

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is **build**
db_name	yes	string	Database name (.db is not required)
db_path	yes	string	Database file path (can be an absolute path or a relative path. The relative path uses the gStore installation root directory as a reference directory)
username	yes	string	User name
password	yes	string	Password (plain text)

Return value

Parameter name	Tpye	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{
    "StatusCode": 0,
    "StatusMsg": "Import RDF file to database done."
}
```

5.3.3.2 check

Brief description

- Check whether the GHTTP service is online

Request URL

- `http://127.0.0.1:9000/`

Request mode

- **GET/POST**

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	type	Note
operation	yes	string	Operation name, fixed value is **check**

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{
    "StatusCode": 0,
    "StatusMsg": "the ghttp server is running..."
}
```

5.3.3.3 load

Brief description

- Loading a database into memory is a prerequisite for many operations, such as Query, monitor, and so on.

Request URL

- `http://127.0.0.1:9000/`

Request mode

- **GET/POST**

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**load**
db_name	yes	string	Database name (.db is not required)
username	yes	string	user name
password	yes	string	Password (plain text)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{
    "StatusCode": 0,
    "StatusMsg": "Database loaded successfully."
}
```

5.3.3.4 monitor

Brief description

- et database statistics (load database first)

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**monitor**
db_name	yes	string	Database name (.db is not required)
username	yes	string	user name
password	yes	string	Password (plain text)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
database	string	database name
creator	string	creator
built_time	string	create time
triple num	string	number of triples
entity num	int	number of entities
literal num	int	number of characters (attribute value)
subject num	int	number of subjects
predicate num	int	number of objects
connection num	int	number of connections

Return sample


```
{
    "StatusCode": 0,
    "StatusMsg": "success",
    "database": "test_lubm",
    "creator": "root",
    "built_time": "2021-08-27 21:29:46",
    "triple num": "99550",
    "entity num": 28413,
    "literal num": 0,
    "subject num": 14569,
    "predicate num": 17,
    "connection num": 0
}
```

5.3.3.5 unload

Brief description

- Unload the database from memory (all changes are flushed back to hard disk)

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**unload**
db_name	yes	string	Database name (.db is not required)
username	yes	string	user name
password	yes	string	Password (plain text)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{
    "StatusCode": 0,
    "StatusMsg": "Database unloaded."
}
```

5.3.3.6 drop

Brief description

- Delete the database (either logically or physically)

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**drop**
db_name	yes	string	Database name (.db is not required)
username	yes	string	user name
password	yes	string	Password (plain text)
is_backup	no	string	True: Logical deletion, false: represents physical deletion (default true)， if it's logical deletion, change the file folder to .bak file foder, user can change the folder name to. Db and add the folder to the system database by calling bin/ ginit-db database name

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{
    "StatusCode": 0,
    "StatusMsg": "Database test_lubm dropped."
}
```

5.3.3.7 show

Brief description

- Display all database list

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**show**
username	yes	string	user name
password	yes	string	Password (plain text)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
ResponseBody	JSONArray	JSON arrays (each of which is a database information)
-------- database	string	database name
---------creator	string	creator
--------- built_time	string	create time
---------status	string	database status

Return sample

```
{
    "StatusCode": 0,
    "StatusMsg": "Get the database list successfully!",
    "ResponseBody": [
        {
```

```
            "database": "lubm",
            "creator": "root",
            "built_time": "2021-08-22 11:08:57",
            "status": "loaded"
        },
        {
            "database": "movie",
            "creator": "root",
            "built_time": "2021-08-27 20:56:56",
            "status": "unloaded"
        }
    ]
}
```

5.3.3.8 usermanage（added)

Brief description

- Manage users (including adding, deleting, and changing users)

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**usermanage**
type	yes	string	Operation Type（1：adduser，2：deleteUser 3：alterUserPassword)
username	yes	string	User name
password	yes	string	Password (plain text)
op_username	yes	string	User name for the operation
op_password	yes	string	Password of the operation (if the password is to be changed, the password is the password to be changed) (If the operation contains special characters and the get request is adopted, the value must be urlencode-encoded)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{    "StatusCode": 1004,    "StatusMsg": "username already existed, add user failed."}
```

5.3.3.9 showuser

Brief description

- Display all user information

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**showuser**
username	yes	string	user name
password	yes	string	Password (plain text)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
ResponseBody	JsonArray	JSON object array
------username	string	user name
------password	string	password
----- query_privilege	string	Query permissions (database names separated by commas)
---- update_privilege	string	Update permissions (database names separated by commas)
----load_privilege	string	Load permissions (database names separated by commas)
--- unload_privilege	string	Unload permissions (database names separated by commas)
---- backup_privilege	string	Back up permissions (database names separated by commas)
---- restore_privilege	string	Restore permissions (database names separated by commas)
--- export_privilege	string	Export permissions (database names separated by commas)

Return sample


```
{    "StatusCode": 0,    "StatusMsg": "success",    "ResponseBody": [        {
        "username": "liwenjie",            "password": "shuaige1982",
 "query privilege": "",            "update privilege": "",            "load
privilege": "",            "unload privilege": "",            "backup
privilege": "",            "restore privilege": "",            "export
privilege": ""        },        {            "username": "liwenjie2",
 "password": "shuaige19888@&",            "query privilege": "lubm,movie,",
      "update privilege": "lubm,movie,",            "load privilege":
"lubm,movie,",            "unload privilege": "lubm,movie,",            "backup
privilege": "lubm,movie,",            "restore privilege": "",
 "export privilege": ""        },        {            "username": "root",
    "password": "123456",            "query privilege": "all",
 "update privilege": "all",            "load privilege": "all",
 "unload privilege": "all",            "backup privilege": "all",
 "restore privilege": "all",            "export privilege": "all"        }    ]}
```

5.3.3.10 userprivilegemanage（added)

Brief description

- Manage user permissions (including adding, deleting, and changing users)

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**userprivilegemanage**
type	yes	string	Operation type（1：add privilege，2：delete privilege 3：clear Privilege)
username	yes	string	user name
password	yes	string	Password (plain text)
op_username	yes	string	User name for the operation
privileges	no	string	Permissions to operate on (multiple permissions separated by commas) (can be null for clear Privilege)1:query,2:load,3:unload,4:update,5:backup,6:restore,7:export, you can set multi privileges by using , to split.
db_name	no	string	The database to operate on (this can be empty if it is clearPrivilege

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{    "StatusCode": 0,    "StatusMsg": "add privilege query successfully. \r\nadd privilege load successfully. \r\nadd privilege unload successfully. \r\nadd privilege update successfully. \r\nadd privilege backup successfully. \r\n"}
```

5.3.3.11 backup

Brief description

- Back up database

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**backup**
username	yes	string	user name
password	yes	string	Password (plain text)
db_name	yes	string	database that need operaions
backup_path	no	string	The backup file path can be relative or absolute. The relative path uses the gStore root directory as reference. The default path is the backup directory in the gStore root directory

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
backupfilepath	string	Backup file path (this value can be used as the input parameter value for restore)

Return sample

```
{    "StatusCode": 0,    "StatusMsg": "Database backup successfully.",
 "backupfilepath": "testbackup/lubm.db_210828211529"}
```

5.3.3.12 restore

Brief description

- Restore database

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**restore**
username	yes	string	user name
password	yes	string	Password (plain text)
db_name	yes	string	database that need operations
backup_path	no	string	The full time-stamped path of the backup file (can be a relative path or an absolute path. The relative path is based on the gStore root directory). The default path is the backup directory in the gStore root directory

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{    "StatusCode": 0,    "StatusMsg": "Database restore successfully."}
```

5.3.3.13 query

Brief description

- query the database

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**query**
username	yes	string	user name
password	yes	string	Password (plain text)
db_name	yes	string	database that need operations
format	no	string	The result set returns in json, HTML, and file format. The default is JSOn
sparql	yes	string	Sparql statement to execute (SPARQL requires URL encoding if it is a GET request)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
head	JSON	head information
results	JSON	Result information (see Return Sample for details)

Return sample

```
{    "head": {        "link": [],        "vars": [            "x"        ]    },
   "results": {        "bindings": [            {                "x": {
           "type": "uri",                    "value": "十面埋伏"                }
           },            {                "x": {                    "type":
"uri",                    "value": "投名状"                }            },
   {                "x": {                    "type": "uri",
 "value": "如花"                }            }        ]    },    "StatusCode": 0,
   "StatusMsg": "success"}
```

5.3.3.14 export

Brief description

- export database

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**restore**
username	yes	string	user name
password	yes	string	Password (plain text)
db_name	yes	string	database that need operations
db_path	no	string	Export path (gstore root by default)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
filepath	string	Path for exporting files

Return sample

```
{    "StatusCode": 0,    "StatusMsg": "Export the database successfully.",
 "filepath": "export/lubm_210828214603.nt"}
```

5.3.3.15 login

Brief description

- Login user (verify username and password)

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**login**
username	yes	string	user name
password	yes	string	Password (plain text)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{    "StatusCode": 1001,    "StatusMsg": "wrong password."}
```

5.3.3.16 begin

Brief description

- start transaction

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**begin**
username	yes	string	User name
password	yes	string	Password (plain text)
db_name	yes	string	Database name
isolevel	yes	string	Transaction isolation level 1:RC(read committed) 2:SI(snapshot isolation) 3:SR(seriablizable)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
TID	string	Transaction ID(this ID is important enough to take as a parameter

Return sample

```
{    "StatusCode": 1001,    "StatusMsg": "wrong password."}
```

5.3.3.17 tquery

Brief description

- query the transaction type

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**tquery**
username	yes	string	User name
password	yes	string	Password (plain text)
db_name	yes	string	Database name
tid	yes	string	Transaction ID
sparql	yes	string	sparql statement

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{    "result": "",    "StatusCode": 0,    "StatusMsg": "success"}
```

5.3.3.18 commit

Brief description

- submit transaction

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**commit**
username	yes	string	User name
password	yes	string	Password (plain text)
db_name	yes	string	Database name
tid	yes	string	Transaction ID

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{    "StatusCode": 0,    "StatusMsg": "transaction commit success. TID: 1"}
```

5.3.3.19 rollback

Brief description

- Rollback trasnsaction

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**rollback**
username	yes	string	User name
password	yes	string	Password (plain text)
db_name	yes	string	Database name
tid	yes	string	Transaction ID

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{    "StatusCode": 0,    "StatusMsg": "transaction rollback success. TID: 2"}
```

5.3.3.20 txnlog

Brief description

- Get transaction logs (this function only works for root user)

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**txnlog**
username	yes	string	user name
password	yes	string	Password (plain text)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
list	JSONArray	Log JSON array

Return sample

```
{    "StatusCode": 0,    "StatusMsg": "Get Transaction log success",    "list":
[        {            "db_name": "lubm2",            "TID": "1",
 "user": "root",            "begin_time": "1630376221590",            "state":
"COMMITED",            "end_time": "1630376277349"        },        {
 "db_name": "lubm2",            "TID": "2",            "user": "root",
 "begin_time": "1630376355226",            "state": "ROLLBACK",
 "end_time": "1630376379508"        }    ]}
```

5.3.3.21 checkpoint

Brief description

- Received Flush data back to hard disk (to make data final)

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**checkpoint**
username	yes	string	User name
password	yes	string	Password (plain text)
db_name	yes	string	Database name

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{    "StatusCode": 0,    "StatusMsg": "Database saved successfully."}
```

5.3.3.22 testConnect

Brief description

- Test whether the server can connect (for workbench)

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**testConnect**
username	yes	string	user name
password	yes	string	Password (plain text)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
CoreVersion	string	Core version number
licensetype	string	License type (Open Source or Enterprise)

Return sample

```
{    "StatusCode": 0,    "StatusMsg": "success",    "CoreVersion": "0.9.1",
 "licensetype": "\"opensource\""}
```

5.3.3.23 getCoreVersion

Brief description

- Get the server version number (for Workbench)

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**getCoreVersion**
username	yes	string	user name
password	yes	string	Password (plain text)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
CoreVersion	string	Core version number

Return sample

```
{    "StatusCode": 0,    "StatusMsg": "success",    "CoreVersion": "0.9.1"}
```

5.3.3.24 batchInsert

Brief description

- batch insert data

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**batchInsert**
username	yes	string	User name
password	yes	string	Password (plain text)
db_name	yes	string	Database name
file	yes	string	The data NT file to insert (can be a relative or absolute path)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
success_num	string	Number of successful executions

Return sample

```
{    "StatusCode": 0,    "StatusMsg": "Batch Insert Data  Successfully.",
 "success_num": "25"}
```

5.3.3.25 batchRemove

Brief description

- batch remove data

Request URL

- `http://127.0.0.1:9000/`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**batchRemove**
username	yes	string	User name
password	yes	string	Password (plain text)
db_name	yes	string	Database name
file	yes	string	Data NT files to be deleted (can be relative or absolute paths)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
success_num	int	Number of successful executions

Return sample

```
{    "StatusCode": 0,    "StatusMsg": "Batch Remove Data  Successfully.",
 "success_num": "25"}
```

5.3.3.26 shutdown

Brief description

- close ghttp

Request URL

- `http://127.0.0.1:9000/shutdown` [Note, address change]

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
username	yes	string	user name（default user name is system)
password	yes	string	Password（This password need to be viewed in the server's system.db/password[port].txt file，for example, if the port is 9000，then check password in password9000.txt file)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
If no value is returned, no information is received by default on success, and an error JSON message is returned on failure
```

5.3.3.27 querylog

Brief description

- Obtaining query Logs

Request URL

`http://127.0.0.1:9000`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**querylog**
username	yes	string	User name
password	yes	string	Password (plain text)
date	yes	string	Date format is yyyyMMdd
pageNo	yes	int	Page number. The value ranges from 1 to N. The default value is 1
pageSize	yes	int	Number of pages. The value ranges from 1 to N. The default value is 10

Return value

Parameter	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
totalSize	int	Total number
totalPage	int	Total page number
pageNo	int	Current page number
pageSize	int	Each page
list	Array	Log array
QueryDateTime	string	Query date/time
Sparql	string	SPARQL statement
Format	string	Query return format
RemoteIP	string	Request IP
FileName	string	Query result set files
QueryTime	int	Time (ms)
AnsNum	int	Result number

Return sample


```
{   "StatusCode":0,    "StatusMsg":"Get query log success", "totalSize":64,
"totalPage":13, "pageNo":2, "pageSize":5,   "list":[        {
"QueryDateTime":"2021-11-16 14:55:52:90ms:467microseconds",
"Sparql":"select ?name where { ?name <不喜欢> <Eve>. }",
"Format":"json",            "RemoteIP":"183.67.4.126",
"FileName":"140163774674688_20211116145552_847890509.txt",
"QueryTime":0,          "AnsNum":2      }        ......    ]}
```

5.3.3.28 ipmanage

Brief description

- Blacklist and whitelist management

Request URL

`http://127.0.0.1:9000`

Request mode

- GET/POST

Parameter transfer mode

- GET request, the parameters are passed directly as the URL
- POST request, `raw` in `body` in `Httprequest`, passed as `JSON` structure

Parameter

查询黑白名单

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**ipmanage**
username	yes	string	User name
password	yes	string	Password (plain text)
type	yes	string	Operation type, fixed value is**1**

保存黑白名单

Parameter name	Mandatory	Type	Note
operation	yes	string	Operation name, fixed value is**ipmanage**
username	yes	string	user name
password	yes	string	Password (plain text)
type	yes	string	Operation type, fixed value is**2**
whiteIPs	yes	string	Whitelist (multiple with **,** split, support range configuration, use **-** connection such as: IP1-1P2)
blackIPs	yes	string	Blacklist (multiple with **,** split, support range configuration, use **-** connection such as: IP1-1P2)

```
//保存POST示例{    "operation": "ipmanage",    "username": "root",    "password": "123456",    "type": "2",    "whiteIPs":"127.0.0.1,183.67.4.126-183.67.4.128",    "blackIPs": "192.168.1.141"}
```

Return value

Parameter	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
ResponseBody	Object	Return data (only for queries)
whiteIPs	array	Whitelist
blackIPs	array	Blacklist

Return sample

```
// 查询黑白名单返回{    "StatusCode": 0,    "StatusMsg": "success",
"ResponseBody": {        "whiteIPs": [            "127.0.0.1",
"183.67.4.126-183.67.4.128"        ],        "blackIPs": [
"192.168.1.141"        ]    }}// 保存黑白名单返回{    "StatusCode": 0,
"StatusMsg": "success"}
```

Attached table 1 return value code table

Code value	Definition
0	Success
1000	The method type is not support
1001	Authentication Failed
1002	Check Privilege Failed
1003	Param is illegal
1004	The operation conditions are not satisfied
1005	Operation failed
1006	Add privilege Failed
1007	Loss of lock
1008	Transcation manage Failed
1100	The operation is not defined
1101	IP Blocked

5.4 C++ HTTP API

To use the C++ API, put the phrase '#include "client.h" in your CPP code, as shown below:

Construct the initialization function

```
GstoreConnector(std::string serverIP, int serverPort, std::string username,
std::string password);
Function: Initialize
Parameter Definition: [Server IP], [GHTTP port on the server], [Username],
[password]
Example: GstoreConnector gc("127.0.0.1", 9000, "root", "123456");
```

Build database: build

```
std::string build(std::string db_name, std::string rdf_file_path, std::string
request_type);
Function: Create a new database from an RDF file
Parameter Definition: [database name], [.nt file path], [request type "GET" and
"POST ", if request type "GET" can be omitted]
Example: gc.build("lubm", "data/lubm/lubm.nt");
```

Load database: load

```
std::string load(std::string db_name, std::string request_type);
Function: Load the database you created
Parameter Definition: [database name], [request type "GET" and "POST ", if
request type "GET" can be omitted]
Example: gc.load("lubm");
```

Stopping database loading: unload

```
std::string unload(std::string db_name, std::string request_type);
Function: Stopping database loading
Parameter Definition: [database name], [request type "GET" and "POST ", if
request type "GET" can be omitted]
Example: gc.unload("lubm");
```

User management: user

```
std::string user(std::string type, std::string username2, std::string addition,
std::string request_type);
Function: The root user can add, delete, or modify the user's permission only
1.Add or delete users:
Parameter Definition: ["add_user" adds user, "delete_user" deletes user],[user
name],[password],[request type "GET" and "POST ", if request type "GET" can be
omitted]
Example: gc.user("add_user", "user1", "111111");
2.Modify user's privilege:
Parameter Definition: ["Add_query" adds query permission, "delete_query" deletes
query permission, "add_load" adds load permission, "delete_load" deletes load
permission, "add_unload" adds no load permission, "delete_unload" deletes no load
permission,  "Add_update" adds update permission, "delete_update" deletes update
permission, "add_backup" adds backup permission, "delete_bakup" deletes backup
permission, "add_restore" adds restore permission,  "Delete_restore" deletes
restore permission,"add_export" adds export permission," delete_export" deletes
export permission],[user name],[database name],[Request type "GET" and "POST ",
if the request type is "GET", it can be omitted.]
Example: gc.user("add_query", "user1", "lubm");
```

Display user: showUser

```
std::string showUser(std::string request_type);
Function: Display all Users
Parameter Definition: [Request types "GET" and "POST" can be omitted if the
request type is "GET"]
Example: gc.showUser();
```

Database query: query

```
std::string query(std::string db_name, std::string format, std::string sparql,
std::string request_type);
Function: Query the database
Parameter Definition: [database name], [query result type JSON, HTML or text],
[SPARQL statement], [Request type "GET" and "POST ", if the request type is
"GET", it can be omitted]
Example:
std::string res = gc.query("lubm", "json", sparql);
std::cout << res << std::endl;  //output result
```

Deleting a Database: drop

```
std::string drop(std::string db_name, bool is_backup, std::string request_type);
Function: Delete the database directly or delete the database while leaving a
backup
Parameter Definition: [database name], [false not backup, true backup], [request
type "GET" and "POST ", if request type "GET" can be omitted]
Example: gc.drop("lubm", false);  //Delete the database without leaving a backup
```

Monitoring database: monitor

```
std::string monitor(std::string db_name, std::string request_type);Function：
Displays information for a specific database.
Parameter Definition：[database name], [request type "GET" and "POST ", if
request type "GET" can be omitted]Example：gc.monitor("lubm");
```

Save the database: checkpoint

```
std::string checkpoint(std::string db_name, std::string request_type);Function：
If you change the database, save the databas Parameter Definition：[database
name], [request type "GET" and "POST ", if request type "GET" can be
omitted]Example：gc.checkpoint("lubm");
```

Show the database: show

```
std::string show(std::string request_type);Function：Displays all created
databases Parameter Definition：[Request types "GET" and "POST" can be omitted if
the request type is "GET"] Example：gc.show();
```

The kernel version information is displayed: getCoreVersion

```
std::string getCoreVersion(std::string request_type);Function：Get kernel version
information Parameter Definition：[Request types "GET" and "POST" can be omitted
if the request type is "GET"]
Example：gc.getCoreVersion();
```

The API version information is displaye: getAPIVersion

```
std::string getAPIVersion(std::string request_type);
Function：Get the API version information
Parameter Definition：[Request types "GET" and "POST" can be omitted if the
request type is "GET"]
Example：gc.getAPIVersion();
```

Query the database and save the file: fquery

```
void fquery(std::string db_name, std::string format, std::string sparql,
std::string filename, std::string request_type);
Function：Query the database and save the results to a file
Parameter Definition：[database name], [query result type JSON, HTML or text],
[SPARQL statement], [file name], [request type "GET" and "POST ", if the request
type is "GET", it can be omitted]
Example：gc.fquery("lubm", "json", sparql, "ans.txt");
```

Exporting the Database

```
std::string exportDB(std::string db_name, std::string dir_path, std::string
request_type);
Function：Export the database to a folder
Parameter Definition：[database name], [path to database export folder], [request
type "GET" and "POST ", if the request type is "GET", can be omitted]
Example：gc.exportDB("lubm", "/root/gStore/");
```

5.5 Java HTTP API

To use the Java API, please refer to the gStore/API/HTTP/Java/SRC/JGSC/GstoreConnector. Java. Specific use is as follows:

Construct the initialization function

```
public class GstoreConnector(String serverIP, int serverPort, String username,
String password);
Function: Initialize
Parameter Definition: [Server IP], [GHTTP port on the server], [Username],
[password]
Example: GstoreConnector gc = new GstoreConnector("127.0.0.1", 9000, "root",
"123456");
```

Building a database: build

```
public String build(String db_name, String rdf_file_path, String request_type);
Function: Create a new database from an RDF file
Parameter Definition: [database name], [.nt file path], [request type "GET" and
"POST ", if request type "GET" can be omitted]
Example: gc.build("lubm", "data/lubm/lubm.nt");
```

Loading a database: load

```
public String load(String db_name, String request_type);
Function: Load the database you created
Parameter Definition: [database name], [request type "GET" and "POST ", if
request type "GET" can be omitted]
Example: gc.load("lubm");
```

Stopping database loading: unload

```
public String unload(String db_name, String request_type);
Function: Stopping database loading
Parameter Definition: [database name], [request type "GET" and "POST ", if
request type "GET" can be omitted]
Example: gc.unload("lubm");
```

User management: user

```
public String user(String type, String username2, String addition, String
request_type);
Function: The root user can add, delete, or modify the user's permission only.
1.Add or delete users:
Parameter Definition: ["add_user" adds user, "delete_user" deletes user],[user
name],[password],[request type "GET" and "POST ", if request type "GET" can be
omitted]
Example: gc.user("add_user", "user1", "111111");
2.Modify user's privilege:
Parameter Definition: ["Add_query" adds query permission, "delete_query" deletes
query permission, "add_load" adds load permission, "delete_load" deletes load
permission, "add_unload" adds no load permission, "delete_unload" deletes no load
permission,  "Add_update" adds update permission, "delete_update" deletes update
permission, "add_backup" adds backup permission, "delete_bakup" deletes backup
permission, "add_restore" adds restore permission,  "Delete_restore" deletes
restore permission,"add_export" adds export permission," delete_export" deletes
export permission],[user name],[database name],[Request type "GET" and "POST ",
if the request type is "GET", it can be omitted.]
Example: gc.user("add_query", "user1", "lubm");
```

Display user：showUser

```
public String showUser(String request_type);
Function: Display all users
Parameter Definition: [Request types "GET" and "POST" can be omitted if the
request type is "GET"]
Example: gc.showUser();
```

Database query: query

```
public String query(String db_name, String format, String sparql, String
request_type);
Function: query databse
Parameter Definition: [database name], [query result type JSON, HTML or text],
[SPARQL statement], [Request type "GET" and "POST ", if the request type is
"GET", it can be omitted]
Example:
String res = gc.query("lubm", "json", sparql);
System.out.println(res); //output result
```

Database deletion： drop

```
public String drop(String db_name, boolean is_backup, String request_type);
Function: Delete the database directly or delete the database while leaving a
backup.
Parameter Definition: [database name], [false not backup, true backup], [request
type "GET" and "POST ", if request type "GET" can be omitted]
Example: gc.drop("lubm", false);  //Delete the database without leaving a backup
```

Monitoring database: monitor

```
public String monitor(String db_name, String request_type);
Function: Displays information for a specific database
Parameter Definition: [database name], [request type "GET" and "POST ", if
request type "GET" can be omitted]Example: gc.monitor("lubm");
```

Save the database: checkpoint

```
public String checkpoint(String db_name, String request_type);
Function: If you change the database, save the database
Parameter Definition: [database name], [request type "GET" and "POST ", if
request type "GET" can be omitted]Example: gc.checkpoint("lubm");
```

Show database：show

```
public String show(String request_type);Function: Displays all created
databasesParameter Definition: [Request types "GET" and "POST" can be omitted if
the request type is "GET"]Example: gc.show();
```

The kernel version information is displayed：getCoreVersion

```
public String getCoreVersion(String request_type);Function: Get kernel version
information Parameter Definition: [Request types "GET" and "POST" can be omitted
if the request type is "GET"]Example: gc.getCoreVersion();
```

Display API version：getAPIVersion

```
public String getAPIVersion(String request_type);Function: Get API version
Parameter: [Request types "GET" and "POST" can be omitted if the request type is
"GET"]Example: gc.getAPIVersion();
```

Query the database and save the file: fquery

```
public void fquery(String db_name, String format, String sparql, String filename,
String request_type);Function: Query the database and save the result to a file
Parameter definition: [database name], [query result type JSON, HTML or text],
[SPARQL statement], [file name], [Request types "GET" and "POST" can be omitted
if the request type is "GET"]Example: gc.fquery("lubm", "json", sparql,
"ans.txt");
```

Export database

```
public String exportDB(String db_name, String dir_path, String
request_type);Function: Export database to parameter definition under file
folder: [database name], [database export folder path], [Request types "GET" and
"POST" can be omitted if the request type is "GET"】Example: gc.exportDB("lubm",
"/root/gStore/");
```

5.6 Python HTTP API

To use the Python API, please refer to the gStore/API/HTTP/Python/SRC/GstoreConnector. Py. Specific use is as follows:

Construct the initialization function

```
public class GstoreConnector(self, serverIP, serverPort, username, password):
Function：Initialization
Parameter definition：[Server IP], [GHTTP port on the server], [Username],
[password].
Example：gc =  GstoreConnector.GstoreConnector("127.0.0.1", 9000, "root",
"123456")
```

Build database: build

```
def build(self, db_name, rdf_file_path, request_type):
Function：Create a new database from an RDF file
Parameter definition：[Database name]，[.nt文件路径]，[Request types "GET" and
"POST" can be omitted if the request type is "GET"】
Example：res = gc.build("lubm", "data/lubm/lubm.nt")
```

Load database: load

```
def load(self, db_name, request_type):
Function：load the database you have created
Parameter definition：[database name], [request type "GET" and "POST ", if
request type "GET" can be omitted]
Example：res = gc.load("lubm")
```

Unload database: unload

```
def unload(self, db_name, request_type):
Function：Unload database
Parameter definition：[database name], [request type "GET" and "POST ", if
request type "GET" can be omitted]
Example：res = gc.unload("lubm")
```

User management: user

```
def user(self, type, username2, addition, request_type):
Function: The root user can add, delete, or modify the user's permission only.
1.Add or delete users:
Parameter definition: ["add_user" adds a user, "delete_user" deletes a user],
[username],[password],[Request types "GET" and "POST" can be omitted if the
Request type is "GET"]
Example: res = gc.user("add_user", "user1", "111111")
2.Modify user's privilege:
Parameter definition: ["Add_query" adds query permission, "delete_query" deletes
query permission, "add_load" adds load permission, "delete_load" deletes load
permission, "add_unload" adds no load permission, "delete_unload" deletes no load
permission,  "Add_update" adds update permission, "delete_update" deletes update
permission, "add_backup" adds backup permission, "delete_bakup" deletes backup
permission, "add_restore" adds restore permission,  "Delete_restore" deletes
restore permission,"add_export" adds export permission," delete_export" deletes
export permission],  [user name],[database name],[Request types "GET" and "POST"
can be omitted if the Request type is "GET"]
Example: res = gc.user("add_query", "user1", "lubm")
```

Display users: showUser

```
def showUser(self, request_type):
Function: Display all users
Parameter definition: [Request types "GET" and "POST" can be omitted if the
request type is "GET"]
Example: res = gc.showUser()
```

Query Database: query

```
def query(self, db_name, format, sparql, request_type):
Function: Query the database
Parameter definition: [Database name], [Query result type JSON, HTML or text],
[SPARQL statement], [Request types "GET" and "POST" can be omitted if the Request
type is "GET"]
Example:
res = gc.query("lubm", "json", sparql)
print(res) //output result
```

Database deletion: drop

```
def drop(self, db_name, is_backup, request_type):
Function: Delete the database directly or delete the database while leaving a
backup
Parameter definition: [database name], [false not backup, true backup],[Request
types "GET" and "POST" can be omitted if the request type is "GET"]
Example: res = gc.drop("lubm", false)  //Delete the database without leaving a
backup
```

Database Monitor： monitor

```
def monitor(self, db_name, request_type):    Function: Displays information for a
specific database Parameter definition: [Database name], [Request types "GET" and
"POST" can be omitted if the request type is "GET"]
Example: res = gc.monitor("lubm")
```

Save database: checkpoint

```
def checkpoint(self, db_name, request_type):Function: If the database is changed, save the meaning of the database parameters: [database name], [request type "GET" and "POST ", if request type "GET" can be omitted] Example: res = gc.checkpoint("lubm")
```

Display database: show

```
def show(self, request_type):Function: Display the meanings of all created databases: [Request types "GET" and "POST" can be omitted if the request type is "GET"]
Example: res = gc.show()
```

The kernel version information is displayed: getCoreVersion

```
def getCoreVersion(self, request_type):Function: Get the definition of kernel version parameter : [Request types "GET" and "POST" can be omitted if the request type is "GET"]
Example: res = gc.getCoreVersion()
```

Display API version: getAPIVersion

```
def getAPIVersion(self, request_type):Function: Get the API version information parameter definition: [Request types "GET" and "POST" can be omitted if the request type is "GET"]
Example: res = gc.getAPIVersion()
```

Query the database and save the file: fquery

```
def fquery(self, db_name, format, sparql, filename, request_type):Function: Query the database and save the result to a file Parameter definition: [database name], [query result type JSON, HTML or text], [SPARQL statement], [file name], [Request types "GET" and "POST" can be omitted if the request type is "GET"]
Example: gc.fquery("lubm", "json", sparql, "ans.txt")
```

Export databse

```
def exportDB(self, db_name, dir_path, request_type): Function: parameter definition of exprotin database to folders: [database name], [database export folder path], [Request types "GET" and "POST" can be omitted if the request type is "GET"]
Example: res = gc.exportDB("lubm", "/root/gStore/")
```

5.7 Nodejs HTTP API

Before using the Nodejs API, type `NPM install Request` and `NPM Install request-promise` to add the required modules under the Nodejs folder.

To use Nodejs API, please refer to the `gStore/API/http/Nodejs/GstoreConnector.js`. Specific use is as follows:

Construct the initialization function

```
class GstoreConnector(ip = '', port, username = '', password = '');
Function: Initialization
Parameter definition: [Server IP], [GHTTP port on the server], [Username],
[password]
Example: gc =  new GstoreConnector("127.0.0.1", 9000, "root", "123456");
```

Build database: build

```
async build(db_name = '', rdf_file_path = '', request_type);
Function: Create a new database from an RDF file
The defintion of parameters are as follows: [database name], [.nt file path],
[Request types "GET" and "POST" can be omitted if the request type is "GET"]
Example: res = gc.build("lubm", "data/lubm/lubm.nt");
```

Load database：load

```
async load(db_name = '', request_type);
Function: Load the database you have created
Parameter definition: [database name], [request type "GET" and "POST ", if
request type "GET" can be omitted]
Example: res = gc.load("lubm");
```

Unload databse：unload

```
async unload(db_name = '', request_type);
Function: Unload databse
Parameter definition: [database name], [request type "GET" and "POST ", if
request type "GET" can be omitted]
Example: res = gc.unload("lubm");
```

User management：user

```
async user(type = '', username2 = '' , addition = '' , request_type);
Function: The root user can add, delete, or modify the user's permission only
1.Add or delete users:
Parameter definition: ["add_user" adds a user, "delete_user" deletes a user],
[user name],[password],[Request types "GET" and "POST" can be omitted if the
Request type is "GET"]
Example: res = gc.user("add_user", "user1", "111111");
2.Privilege to modify user:
Parameter definition: ["Add_query" adds query permission, "delete_query" deletes
query permission, "add_load" adds load permission, "delete_load" deletes load
permission, "add_unload" adds no load permission, "delete_unload" deletes no load
permission,  "Add_update" adds update permission, "delete_update" deletes update
permission, "add_backup" adds backup permission, "delete_bakup" deletes backup
permission, "add_restore" adds restore permission,  "Delete_restore" deletes
restore permission,"add_export" adds export permission," delete_export" deletes
export permission],  [user name],[database name],[Request types "GET" and "POST"
can be omitted if the Request type is "GET"]]
Example: res = gc.user("add_query", "user1", "lubm");
```

Display user：showUser

```
async showUser(request_type);
Function: Display all users
Parameter definition: [Request types "GET" and "POST" can be omitted if the
request type is "GET"]
Example: res = gc.showUser();
```

Query database：query

```
async query(db_name = '', format = '' , sparql = '' , request_type);
Function: Query database
Parameter definition: [Database name], [query result type JSON, HTML or text],
[SPARQL statement], [Request types "GET" and "POST" can be omitted if the Request
type is "GET"]
Example:
res = gc.query("lubm", "json", sparql);
console.log(JSON.stringify(res,",")); //output result
```

Database deletion：drop

```
async drop(db_name = '', is_backup , request_type);
Function: Delete the database directly or delete the database while leaving a
backup
Parameter definition: [database name], [false no backup, true backup], [Request
types "GET" and "POST" can be omitted if the request type is "GET"]
Example: res = gc.drop("lubm", false);  //Delete the database without leaving a
backup
```

Database monitor: monitor

```
async monitor(db_name = '', request_type);     Function: Parameter definition for
displaying information about a specific database: [database name], [request type
"GET" and "POST ", if request type "GET" can be omitted]Example: res =
gc.monitor("lubm");
```

Save database: checkpoint

```
async checkpoint(db_name = '', request_type);Function: If the database is changed, the parameter definition of saving database: [database name], [request type "GET" and "POST ", if request type "GET" can be omitted]Example: res = gc.checkpoint("lubm");
```

Display database: show

```
async show(request_type);Function: Displays all created databases Parameter definition: [Request types "GET" and "POST" can be omitted if the request type is "GET"]Example: res = gc.show();
```

Display kernel version information: getCoreVersion

```
async getCoreVersion(request_type);Function: Get kernel version information Parameter definition: [Request types "GET" and "POST" can be omitted if the request type is "GET"]Example: res = gc.getCoreVersion();
```

Display API version: getAPIVersion

```
async getAPIVersion(request_type);
Function: Get the API version information
Parameter definition: [Request types "GET" and "POST" can be omitted if the request type is "GET"]
Example: res = gc.getAPIVersion();
```

Query the database and save the file: fquery

```
async fquery(db_name = '', format = '' , sparql = '' , filename = '' , request_type);
Function: Query the database and save the results to a file
Parameter definition: [database name], [query result type JSON, HTML or text], [SPARQL statement], [file name],  [Request types "GET" and "POST" can be omitted if the Request type is "GET"]
Example: gc.fquery("lubm", "json", sparql, "ans.txt");
```

Export database

```
async exportDB(db_name = '' , dir_path = '' , request_type);
Function: export database to folders
parameter definition: [database name], [directory where the database is exported], [Request types "GET" and "POST" can be omitted if the Request type is "GET"]
Example: res = gc.exportDB("lubm", "/root/gStore/");
```

5.8 PHP HTTP API

To use the Php API, please refer to the gStore/API/HTTP/Php/SRC/GstoreConnector. Php. Specific use is as follows:

Construct the initialization function

```
class GstoreConnector($ip, $port, $username, $password)
Function: Initialization
Parameter definition: [Server IP], [GHTTP port on the server], [Username],
[password]
Example: $gc = new GstoreConnector("127.0.0.1", 9000, "root", "123456");
```

Build databse：build

```
function build($db_name, $rdf_file_path, $request_type)
Function: Create a new database from an RDF file
Parameter definition: [database name], [.nt file path],[Request types "GET" and
"POST" can be omitted if the request type is "GET"]
Example:
$res = $gc->build("lubm", "data/lubm/lubm.nt");
echo $res . PHP_EOL;
```

Load database：load

```
function load($db_name, $request_type)
Function: Load the database you have created
Parameter definition: [database name], [request type "GET" and "POST ", if
request type "GET" can be omitted]
Example:
$ret = $gc->load("test");
echo $ret . PHP_EOL;
```

Unload databse：unload

```
function unload($db_name, $request_type)
Function: Unload databse
Parameter definition: [database name], [Request types "GET" and "POST" can be
omitted if the request type is "GET"]
Example:
$ret = $gc->unload("test");
echo $ret . PHP_EOL;
```

User management：user

```
function user($type, $username2, $addition, $request_type)
Function: Only the root user can add, delete, or modify the user's permission
1.Add or delete users:
Parameter definition: ["add_user" adds a user, "delete_user" deletes a user],
[user name],[password],[Request types "GET" and "POST" can be omitted if the
Request type is "GET"]
Example:
$res = $gc->user("add_user", "user1", "111111");
echo $res . PHP_EOL;
2.Privilege to modify user:
参数含义: ["Add_query" adds query permission, "delete_query" deletes query
permission, "add_load" adds load permission, "delete_load" deletes load
permission, "add_unload" adds no load permission, "delete_unload" deletes no load
permission,  "Add_update" adds update permission, "delete_update" deletes update
permission, "add_backup" adds backup permission, "delete_bakup" deletes backup
permission, "add_restore" adds restore permission,  "Delete_restore" deletes
restore permission,"add_export" adds export permission," delete_export" deletes
export permission],  [user name],[database name],[Request types "GET" and "POST"
can be omitted if the Request type is "GET"]]
Example:
$res = $gc->user("add_user", "user1", "lubm");
echo $res . PHP_EOL;
```

Display user：showUser

```
function showUser($request_type)
Function: Display all users
Parameter definition: [Request types "GET" and "POST" can be omitted if the
request type is "GET"]
Example:
$res = $gc->showUser();
echo $res. PHP_EOL;
```

Query database：query

```
function query($db_name, $format, $sparql, $request_type)
Parameter definition: [database name], [query result type JSON, HTML or text],
[SPARQL statement],[Request types "GET" and "POST" can be omitted if the request
type is "GET"]
Example:
$res = $gc->query("lubm", "json", $sparql);
echo $res. PHP_EOL; //output result
```

Database deletion：drop

```
function drop($db_name, $is_backup, $request_type)
Function: Delete the database directly or delete the database while leaving a
backup
Parameter definition: [database name], [false not backup, true backup],[Request
types "GET" and "POST" can be omitted if the request type is "GET"]
Example:
$res = $gc->drop("lubm", false); //Delete the database without leaving a backup
echo $res. PHP_EOL;
```

Database monitor: monitor

```
function monitor($db_name, $request_type)Function：Displays information for a
specific database
Parameter definition：[database name], [request type "GET" and "POST ", if
request type "GET" can be omitted]
Example：$res = $gc->monitor("lubm");echo $res. PHP_EOL;
```

Save database: checkpoint

```
function checkpoint($db_name, $request_type)Function：the Parameter definition of
save database if it has been changed：[database name], [request type "GET" and
"POST ", if request type "GET" can be omitted]Example：$res = $gc-
>checkpoint("lubm");echo $res. PHP_EOL;
```

Display database: show

```
function show($request_type)Function：Displays all created databases
Parameter definition：[Request types "GET" and "POST" can be omitted if the
request type is "GET"]
Example：$res = $gc->show();echo $res. PHP_EOL;
```

Display kernel version information: getCoreVersion

```
function getCoreVersion($request_type)Function：get kernel version information
Parameter definition: [Request types "GET" and "POST" can be omitted if the
request type is "GET"]
Example：$res = $gc->getCoreVersion();echo $res. PHP_EOL;
```

Display API version: getAPIVersion

```
function getAPIVersion($request_type)
Function：Get API version
Parameter definition: [Request types "GET" and "POST" can be omitted if the
request type is "GET"]
Example：
$res = $gc->getAPIVersion();
echo $res. PHP_EOL;
```

Query the database and save the file：fquery

```
function fquery($db_name, $format, $sparql, $filename, $request_type)
Function：Query the database and save the results to a file
Parameter definition：[database name], [query result type JSON, HTML or text],
[SPARQL statement], [file name],[Request types "GET" and "POST" can be omitted if
the request type is "GET"]
Example：$gc->fquery("lubm", "json", $sparql, "ans.txt");
```

Export database

```
function exportDB($db_name, $dir_path, $request_type)
Function：Export the database to a folder
Parameter definition：[database name], [database export folder path]，[Request
types "GET" and "POST" can be omitted if the request type is "GET"]
Example：$res = $gc->exportDB("lubm", "/root/gStore/");
```

5.9 gServer API instruction

5.9.1 API Interconnection Mode

> The gServer interface uses the `socket` protocol and supports multiple ways to access the interface. If the port `9000` is started from the gServer in the Main directory, the contents of the interface interconnection are as follows:
>
> API address:
>
> ```
> http://ip:9000/
> ```
>
> The API supports the input of a parameter list in JSON format, as shown below:
>
> ```
> {"op": "[op_type]", "[paramname1]": "[paramvalue1]", "[paramname2]": "[paramvalue2]"……}
> ```

5.9.2 API List

API name	Definition	Note
build	Build graph database	The database file must be locally stored on the server
load	Load graph database	Load the database into memory
unload	Unload graph database	Unload the database from memory
drop	Delete graph database	Logical deletion and physical deletion can be performed
show	Display graph database	Display list of all databases
query	Query graph database	Including query, delete, and insert
stop	Close server	Only root user root can perform this operation
close	Close client server connection	Process client connection closure requests
login	login to database	authenticate user names and password

5.9.3 API specific instruction

> This section describes the input and output parameters of each interface. Assume that the IP address of the gserver is 127.0.0.1 and the port is 9000

5.9.3.1 build- build database

Brief description

- Create a database based on existing NT file
- Files must exist on the gStore server

Request ip

- `127.0.0.1`

Request port number

- `9000`

Parameter transfer mode

- Pass it as a `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
op	yes	string	Operation name, fixed value is**build**
db_name	yes	string	Database name (.db is not required)
db_path	yes	string	Database file path (can be an absolute path or a relative path. The relative path uses the gStore installation root directory as a reference directory)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{
    "StatusCode": 0,
    "StatusMsg": "Import RDF file to database done."
}
```

5.9.3.2 load

Brief description

- To load a database into memory, a load operation is a prerequisite for many operations, such as Query

Request ip

- `127.0.0.1`

Request port number

- `9000`

Parameter transfer mode

- Pass it as a `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
op	yes	string	Operation name, fixed value is**load**
db_name	yes	string	Database name (.db is not required)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{
    "StatusCode": 0,
    "StatusMsg": "Load database successfully."
}
```

5.9.3.3 unload

Brief description

- Unmount the database from memory (all changes are flushed back to hard disk)

Request ip

- `127.0.0.1`

Request port number

- `9000`

Parameter transfer mode

- Pass it as a `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
op	yes	string	Operation name, fixed value is**unload**
db_name	yes	string	Database name (.db is not required)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{
    "StatusCode": 0,
    "StatusMsg": "Unload database done."
}
```

5.9.3.4 drop

Brief description

- Delete the database

Request ip

- `127.0.0.1`

Request port number

- `9000`

Parameter transfer mode

- Pass it as a `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
op	yes	string	Operation name, fixed value is**drop**
db_name	yes	string	Database name (.db is not required)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{
    "StatusCode": 0,
    "StatusMsg": "Drop database done."
}
```

5.9.3.5 show

Brief description

- Display all database list

Request ip

- `127.0.0.1`

Request port number

- `9000`

Parameter transfer mode

- Pass it as a `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
op	yes	string	Operation name, fixed value is**show**

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
ResponseBody	JSONArray	JSON arrays (each of which is a database information0
-------- database	string	database name
---------status	string	database status

Return sample

```
{
    "StatusCode": 0,
    "StatusMsg": "success",
    "ResponseBody": [
            "lubm": "loaded",
            "lubm10K": "unloaded"
    ]
}
```

5.9.3.6 query

Brief description

- query database

Request ip

- `127.0.0.1`

Request port number

- `9000`

Parameter transfer mode

- Pass it as a `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
op	yes	string	Operation name, fixed value is**query**
db_name	yes	string	database that need operations
format	no	string	The result set return format is json by default
sparql	yes	string	The SPARQL statement to execute

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information
head	JSON	Head information
results	JSON	Result information (see Return Sample for details)

Return sample

```
{
    "head": {
        "link": [],
        "vars": [
            "x"
        ]
    },
    "results": {
        "bindings": [
            {
```

```
                "x": {
                    "type": "uri",
                    "value": "十面埋伏"
                }
            },
            {
                "x": {
                    "type": "uri",
                    "value": "投名状"
                }
            },
            {
                "x": {
                    "type": "uri",
                    "value": "如花"
                }
            }
        ]
    },
    "StatusCode": 0,
    "StatusMsg": "success"
}
```

5.9.3.7 login

Brief description

- Login user (verify username and password)

Request ip

- `127.0.0.1`

Request port number

- `9000`

Parameter transfer mode

- Pass it as a `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
op	yes	string	Operation name, fixed value is**login**
username	yes	string	user name
password	yes	string	Password (plain text)

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{
    "StatusCode": 1001,
    "StatusMsg": "wrong password."
}
```

5.9.3.8 stop

Brief description

- Close server

Request ip

- `127.0.0.1`

Request port number

- `9000`

Parameter transfer mode

- Pass it as a `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
op	yes	string	Operation name, fixed value is**stop**

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{
    "StatusCode": 0,
    "StatusMsg": "Server stopped."
}
```

5.9.3.9 close

Brief description

- Close the connection to the client

Request ip

- `127.0.0.1`

Request port number

- `9000`

Parameter transfer mode

- Pass it as a `JSON` structure

Parameter

Parameter name	Mandatory	Type	Note
op	yes	string	Operation name, fixed value is**close**

Return value

Parameter name	Type	Note
StatusCode	int	Return value code value (refer to attached table: Return value code table for details)
StatusMsg	string	Return specific information

Return sample

```
{
    "StatusCode": 0,
    "StatusMsg": "Connection disconnected."
}
```

Attached table 1 return value code table

Code value	Definition
0	Success
1000	The method type is not support
1001	Authentication Failed
1002	Check Privilege Failed
1003	Param is illegal
1004	The operation conditions are not satisfied
1005	Operation failed
1006	Add privilege Failed
1007	Loss of lock
1008	Transcation manage Failed
1100	The operation is not defined
1101	IP Blocked

6. SPARQL query syntax

6.1 Graph Patterns

This document mainly refer to the SPARQL 1.1 标准文档, but also increased the gStore own customized content, if you want to learn more about gStore SPARQL statement of support, Please read our documentation carefully!

Unless otherwise specified, this document will continue to use the following RDF data instances as objects for queries:

```
<刘亦菲> <姓名> "刘亦菲" .
<刘亦菲> <姓名> "Crystal Liu" .
<刘亦菲> <性别> "女" .
<刘亦菲> <星座> "处女座" .
<刘亦菲> <职业> "演员" .

<林志颖> <姓名> "林志颖" .
<林志颖> <性别> "男" .
<林志颖> <职业> "演员" .
<林志颖> <职业> "导演" .

<胡军> <姓名> "胡军" .
<胡军> <性别> "男" .
<胡军> <星座> "双鱼座" .
<胡军> <职业> "演员" .
<胡军> <职业> "配音" .
<胡军> <职业> "制片" .
<胡军> <职业> "导演" .

<天龙八部> <主演> <林志颖> .
<天龙八部> <主演> <刘亦菲> .
<天龙八部> <主演> <胡军> .
<天龙八部> <类型> <武侠片> .
<天龙八部> <类型> <古装片> .
<天龙八部> <类型> <爱情片> .
<天龙八部> <豆瓣评分> "8.3"^^<http://www.w3.org/2001/XMLSchema#float> .
<天龙八部> <上映时间> "2003-12-
11T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime> .

<恋爱大赢家> <主演> <林志颖> .
<恋爱大赢家> <主演> <刘亦菲> .
<恋爱大赢家> <类型> <爱情片> .
<恋爱大赢家> <类型> <剧情片> .
<恋爱大赢家> <豆瓣评分> "6.1"^^<http://www.w3.org/2001/XMLSchema#float> .
<恋爱大赢家> <上映时间> "2004-11-
30T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
```

Since there is no official Chinese translation of the SPARQL 1.1 standard document, the English version of the term will be indicated when it first appears in the following paragraphs.

By standard, **keywords in SPARQL queries are case insensitive**

6.1.1 The simplest graph mode

Let's start with the simplest query:

```
SELECT ?movie
WHERE
{
    ?movie <主演> <刘亦菲> .
}
```

查询由两部分组成：**SELECT 语句**指定需要输出查询结果的变量，**WHERE 语句**提供用来与数据图匹配的图模式。上面的查询中，图模式由单条**三元组** `?movie <主演> <刘亦菲>` 构成，其中作为主语的 `?movie` 是**变量**，作为谓词的 `<主演>` 和作为宾语的 `<刘亦菲>` 是 **IRI** (International Resource Identifier, 国际资源标识符)。这个查询将返回由刘亦菲主演的所有影视作品，在示例数据上运行结果如下：

The query consists of two parts: the **SELECT statement** specifies the variables that need to output the query results, and the **WHERE statement** provides the graph pattern used to match the data graph. In the above query, the graph pattern consists of a single **triplet** `?movie <主演> <刘亦菲>` `?movie` is **variable** , and `< 主演 >` as predicate and `<刘亦菲 >` as object are **IRI** (International Resource Identifier). This query will return all movies and TV works starring 刘亦菲i. The results run on the sample data are as follows:

?movie
<天龙八部>
<恋爱大赢家>

The subject, predicate and object of a triple can all be IRI. Objects can also be **RDF literals** . The following query will give all the people in the sample data whose profession is director:

```
SELECT ?person
WHERE
{
    ?person <职业> "导演" .
}
```

Where 'director' is an RDF literal

Result are as follows:

?person
<胡军>
<林志颖>

Under the current version of gStore, RDF literals with data types are queried with suffixes corresponding to those in the data file. For example, the following query will give a douban rating of 8.3:

```
SELECT ?movie
WHERE
{
    ?movie <豆瓣评分> "8.3"^^<http://www.w3.org/2001/XMLSchema#float> .
}
```

Results are as follows:

?movie
<天龙八部>

Other common data types include `<http://www.w3.org/2001/XMLSchema#integer>` (integer), `<http://www.w3.org/2001/XMLSchema#decimal>` (point type), `xsd:double` (A double-precision floating point type), `<http://www.w3.org/2001/XMLSchema#string>` (String type), `<http://www.w3.org/2001/XMLSchema#boolean>` (Boolean), `<http://www.w3.org/2001/XMLSchema#dateTime>` (Date/time).Other data types may also appear in data files, simply using the form `^^< data type suffix >` in the query.

6.1.2 Basic Graph Pattern

Base graph pattern is a collection of triples; The two queries in the previous section both have only the outermost braces and therefore belong to the **basic graph mode**; Enclosing the outermost braces is a single base graph pattern **Group Graph Pattern。**

The basic graph pattern in the two queries in the previous section consists of a single triple. The following query uses a basic graph pattern consisting of multiple triples to give all male leads of Tianlongba in the sample data:

```
SELECT ?person
WHERE
{
    <天龙八部> <主演> ?person .
    ?person <性别> "男" .
}
```

Result are as follows:

?person
<胡军>
<林志颖>

6.1.3 Group Graph Pattern

Group graph patterns are separated by paired braces. A group graph pattern can be composed of a single base graph pattern, as described in the previous section, or multiple subgroups of graph patterns nested with the following operations: **OPTIONAL**, **UNION**, and **MINUS**. **FILTER** filters the results within the range of a group graph pattern

OPTIONAL

The keyword OPTIONAL uses the following syntax:

```
pattern1 OPTIONAL { pattern2 }
```

The result of the query must match `pattern1` and selectively match `pattern2`. `Pattern2` is known as the OPTIONAL graph pattern. If there is a match for `pattern2`, add it to the match for `Pattern1`; Otherwise, the match for `pattern1` is still printed. For this reason, OPTIONAL is often used when some data is missing.

The following query gives the gender and constellation information of the person in the sample data. Among them, as long as there is gender information of the character will be returned, regardless of whether there is the constellation information of the character; If both exist, additional returns are returned

```
SELECT ?person ?gender ?horoscope
WHERE
{
    ?person <性别> ?gender .
    OPTIONAL
    {
        ?person <星座> ?horoscope .
    }
}
```

Results are as follows:

?person	?gender	?horoscope
<刘亦菲>	"女"	"处女座"
<林志颖>	"男"	
<胡军>	"男"	"双鱼座"

UNION

The keyword UNION is syntactically similar to OPTIONAL. In a graph pattern joined by UNION, as long as there is one that matches a piece of data, that data matches the whole joined by UNION. Therefore, UNION can be understood as finding the set of matching results of each graph pattern it joins (actually using multiple set semantics because it allows repeating results).

The following query gives the sample data for films and television works whose category is costume film or drama film:

```
SELECT ?movie
WHERE
{
    {?movie <类型> <古装片> .}
    UNION
    {?movie <类型> <剧情片> .}
}
```

Results are as follows:

?movie
<天龙八部>
<恋爱大赢家>

MINUS

The usage syntax of the keywords MINUS is similar to OPTIONAL and UNION. MINUS The matching of the left and right graph patterns will be calculated, and the part that can match the right graph pattern will be removed from the matching result of the left graph pattern as the final result. Therefore, MINUS can be understood as the difference of the matching result set of the two graph patterns connected to it (the left is the subtracted set, multiple set semantics).

The following query will give the sample data of the characters who starred in The Dragon Eight but did not star in the love winner:

```
SELECT ?person
WHERE
{
    <天龙八部> <主演> ?person .
    MINUS
    {<恋爱大赢家> <主演> ?person .}
}
```

Results are as follows:

?person
<胡军>

FILTER

The keyword FILTER is followed by a constraint condition, and the results that do not meet this condition in the current pattern group will be filtered out and not returned. FILTER conditions can use equations, inequalities, and various built-in functions.

The following query will give the film and television works with douban score higher than 8 points in the sample data:

```
SELECT ?movie
WHERE
{
    ?movie <豆瓣评分> ?score .
    FILTER (?score > "8"^^<http://www.w3.org/2001/XMLSchema#float>)
}
```

Results are as follows：

?movie
<天龙八部>

No matter where a FILTER is placed in a group graph pattern, as long as it remains in the same nesting layer, its semantics remain unchanged and the scope of the constraint remains the current group graph pattern. For example, the following query is equivalent to the previous one：

```
SELECT ?movie
WHERE
{
    FILTER (?score > "8"^^<http://www.w3.org/2001/XMLSchema#float>)
    ?movie <豆瓣评分> ?score .
}
```

One of the built-in functions commonly used for FILTER conditions is the regular expression **REGEX**. The following query gives the liu surname in the sample data：

```
SELECT ?person
WHERE
{
    ?person <姓名> ?name .
    FILTER REGEX(?name, "刘.*")
}
```

Results are as follows：

?person
<刘亦菲>

6.2 Aggregates

Aggregate functions are used in SELECT statements with the following syntax:

```
SELECT (AGGREGATE_NAME(?x) AS ?y)
WHERE
{
    ...
}
```

Where `AGGREGATE_NAME` is the name of the aggregation function, variable `?X` is the aggregate function on the object, variable `?Y` is the column name of the aggregate function value in the final result.

The aggregate function acts on each group of outcomes. All results are returned as a set by default. The aggregation functions supported by gStore are as follows

COUNT

Aggregate function for counting.

The following query gives the number of actors in the sample data:

```
SELECT (COUNT(?person) AS ?count_person)
WHERE
{
    ?person <职业> "演员" .
}
```

Results are as follows:

?count_person
"3"^^<http://www.w3.org/2001/XMLSchema#integer>

SUM

Aggregate function for summation.

The following query will give the sum of douban ratings for all movies in the sample data:

```
SELECT (SUM(?score) AS ?sum_score)
WHERE
{
    ?movie <豆瓣评分> ?score .
}
```

Results are as follows:

?sum_score
"14.400000"^^http://www.w3.org/2001/XMLSchema#float

AVG

An aggregate function for averaging.

The following query gives the average Douban score for all movies in the sample data:

```
SELECT (AVG(?score) AS ?avg_score)
WHERE
{
    ?movie <豆瓣评分> ?score .
}
```

Results are as follows:

?avg_score
"7.200000"^^http://www.w3.org/2001/XMLSchema#float

MIN

An aggregate function for minimizing.

The following query will give the lowest Douban score for all movies in the sample data:

```
SELECT (MIN(?score) AS ?min_score)
WHERE
{
    ?movie <豆瓣评分> ?score .
}
```

Results are as follows:

?min_score
"6.1"^^http://www.w3.org/2001/XMLSchema#float

MAX

Aggregate function used to find the maximum value.

The following query will give the highest Douban ratings for all movies in the sample data:

```
SELECT (MAX(?score) AS ?max_score)
WHERE
{
    ?movie <豆瓣评分> ?score .
}
```

Results are as follows:

?max_score
"8.3"^^http://www.w3.org/2001/XMLSchema#float

GROUP BY

If you want to GROUP the results BY the value of a variable, you can use the keyword GROUP BY. For example, the following query gives all occupations and their corresponding numbers in the sample data:

```
SELECT ?occupation (COUNT(?person) AS ?count_person)
WHERE
{
    ?person <职业> ?occupation .
}
GROUP BY ?occupation
```

Results are as follows:

?occupation	?count_person
"演员"	"3"^^<http://www.w3.org/2001/XMLSchema#integer>
"导演"	"2"^^<http://www.w3.org/2001/XMLSchema#integer>
"配音"	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
"制片"	"1"^^<http://www.w3.org/2001/XMLSchema#integer>

6.3 Solution Sequences and Modifiers

The following keywords are result sequence modifiers that post process the query results to form the final returned results.

DISTINCT: Removes duplicate results

A query with a SELECT statement without the keyword DISTINCT will retain duplicate results in the final result. For example, the following query gives all the occupations in the sample data

```
SELECT ?occupation
WHERE
{
    ?person <职业> ?occupation .
}
```

Results are as follows:

?occupation
"演员"
"演员"
"演员"
"导演"
"导演"
"制片"
"配音"

If you want to see DISTINCT job categories, you can add the keyword DISTINCT in the SELECT statement:

```
SELECT DISTINCT ?occupation
WHERE
{
    ?person <职业> ?occupation .
}
```

Results are as follows:

?occupation
"演员"
"导演"
"制片"
"配音"

DISTINCT can also be used in the aggregate function COUNT. The following query gives the number of occupations in the sample data:

```
SELECT (COUNT(DISTINCT ?occupation) AS ?count_occupation)
WHERE
{
    ?person <职业> ?occupation .
}
```

Results are as follows:

?count_occupation
"4"^^<http://www.w3.org/2001/XMLSchema#integer>

ORDER BY: 排序

Query results are unordered by default. If you want to sort the results based on the values of some variables, you can add an ORDER BY statement after the WHERE statement. For example, the following query will sort the film and television works in the sample data according to douban score. If the order is not specified, it will be in ascending order by default

```
SELECT ?movie ?score
WHERE
{
    ?movie <豆瓣评分> ?score
}
ORDER BY ?score
```

Results are as follows:

?movie	?score
<恋爱大赢家>	"6.1"^^<http://www.w3.org/2001/XMLSchema#float>
<天龙八部>	"8.3"^^<http://www.w3.org/2001/XMLSchema#float>

If you want to sort in descending order, you need to modify the variable name with the keyword DESC:

```
SELECT ?movie ?score
WHERE
{
    ?movie <豆瓣评分> ?score
}
ORDER BY DESC(?score)
```

Results are as follows:

?movie	?score
<天龙八部>	"8.3"^^<http://www.w3.org/2001/XMLSchema#float>
<恋爱大赢家>	"6.1"^^<http://www.w3.org/2001/XMLSchema#float>

The ORDER BY statement can contain multiple space-separated variables, each of which can be decorated with DESC. gStore does not currently support the use of four-operation expressions and built-in functions in ORDER BY statements.

OFFSET: skips a certain number of results

The OFFSET statement follows the WHERE statement and has the following syntax:

```
OFFSET nonnegative_integer
```

`nonnegative_INTEGER` must be a non-negative integer, indicating the number of results to be skipped. `OFFSET 0` is syntactic but has no effect on the result. Because the query results are unordered by default, SPARQL semantics do not guarantee that the skipped results meet any deterministic conditions. Therefore, the OFFSET statement is typically used in conjunction with the

ORDER BY statement。

The following query sorts the film and television works in the sample data by douban score from lowest to highest, and skips the film and television works with the lowest score:

```
SELECT ?movie ?score
WHERE
{
    ?movie <豆瓣评分> ?score .
}
ORDER BY ?score
OFFSET 1
```

Results are as follows:

?movie	?score
<天龙八部>	"8.3"^^<http://www.w3.org/2001/XMLSchema#float>

LIMIT: Limit the number of results

The syntax of the LIMIT statement is similar to that of the OFFSET statement:

```
LIMIT nonnegative_integer
```

`nonnegative_INTEGER` must be a non-negative integer, indicating the maximum number of results allowed. Similar to OFFSET, the LIMIT statement is typically used in conjunction with the ORDER BY statement because the query result defaults to unordered.

The following query gives the film and television works with the highest douban score in the sample data:

```
SELECT ?movie ?scoreWHERE{  ?movie <豆瓣评分> ?score .}ORDER BY DESC(?score)LIMIT
1
```

Results are as follows:

?movie	?score
<天龙八部>	"8.3"^^<http://www.w3.org/2001/XMLSchema#float>

6.4 Update graph

By **INSERT DATA**, **DELETE DATA**, and **DELETE WHERE** queries, we can INSERT or DELETE triples from the database.

INSERT DATA

INSERT DATA is used to INSERT triples into a database. The syntax is similar to that of a SELECT query, except that there are no variables in the triples that make up the group graph pattern.

The following query inserts the relevant information of the film and TELEVISION works Chinese Paladin into the sample data:

```
INSERT DATA
{
    <仙剑奇侠传> <主演> <胡歌> .
    <仙剑奇侠传> <主演> <刘亦菲> .
    <仙剑奇侠传> <类型> <武侠片> .
    <仙剑奇侠传> <类型> <古装片> .
    <仙剑奇侠传> <类型> <爱情片> .
    <仙剑奇侠传> <豆瓣评分> "8.9"^^<http://www.w3.org/2001/XMLSchema#float> .
}
```

The query that appears in the "Graph pattern - The simplest Graph pattern" section

```
SELECT ?movie
WHERE
{
    ?movie <主演> <刘亦菲> .
}
```

After inserting the above data, the result becomes:

?movie
<天龙八部>
<恋爱大赢家>
<仙剑奇侠传>

DELETE DATA

DELETE DATA is used to DELETE triples from a database. The usage is exactly similar to INSERT DATA.

DELETE WHERE

DELETE DATA is used to DELETE eligible triples from the database; In contrast to DELETE DATA, its WHERE statement is exactly the same as the WHERE statement of a SELECT query, meaning that variables are allowed in triples. For example, the following query removes all information about swordsman films from the sample data:

```
DELETE WHERE{   ?movie <类型> <武侠片> . ?movie ?y ?z .}
```

Run the query that appeared in the "Graph patterns - The simplest Graph patterns" section again:

```
SELECT ?movie
WHERE
{
    ?movie <主演> <刘亦菲> .
}
```

Result change to：

?movie
<恋爱大赢家>

6.5 Advanced functions

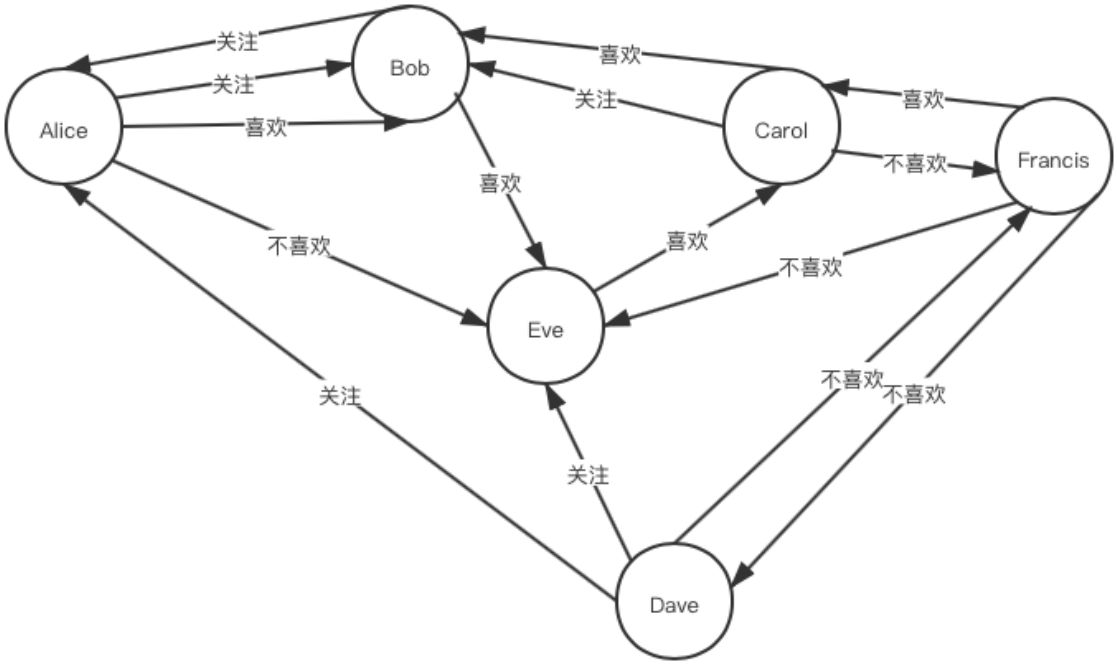
In **kernel version V0.9.1**, gStore has added a number of queries related to the path and centrality of nodes in the data graph, including loop query, shortest path query, K-hop reachable query and Personalized PageRank query.

6.5.1 Sample data

To better demonstrate the advanced functionality, use the following social relationship data as sample data：

```
<Alice> <关注> <Bob> .
<Alice> <喜欢> <Bob> .
<Alice> <不喜欢> <Eve> .
<Bob> <关注> <Alice> .
<Bob> <喜欢> <Eve> .
<Carol> <关注> <Bob> .
<Carol> <喜欢> <Bob> .
<Carol> <不喜欢> <Francis> .
<Dave> <关注> <Alice> .
<Dave> <关注> <Eve> .
<Dave> <不喜欢> <Francis> .
<Eve> <喜欢> <Carol> .
<Francis> <喜欢> <Carol> .
<Francis> <不喜欢> <Dave> .
<Francis> <不喜欢> <Eve> .
```

The above data are illustrated below:



Unless otherwise specified, functions that return paths represent a path/a ring/a subgraph in JSON format as follows:

```
{
    "src": "<src_IRI>", "dst": "<dst_IRI>",
    "edges": [
        { "fromNode": 0, "toNode": 1, "predIRI": "<pred>" }
    ],
    "nodes": [
        { "nodeIndex": 0, "nodeIRI": "<src_IRI>" },
        { "nodeIndex": 1, "nodeIRI": "<dst_IRI>" }
    ]
}
```

The final return value represents a set of paths/rings/subgraphs as follows :(where the 'paths' element has the format above)

```
{ "paths": [{...}, {...}, ...] }
```

6.5.2Path-related query

6.5.1.1The loop query

Queries for the existence of a ring containing nodes `u` and `v`

```
cyclePath(u, v, directed, pred_set)
cycleBoolean(u, v, directed, pred_set)
```

Used in SELECT statements, using the same syntax as aggregate functions

Parameter

`u`, `v` : Variable or node IRI

`directed` ： A Boolean value, true for directed and false for undirected (all edges in the graph are considered bidirectional

`pred_set` ： The set of predicates that make up the edges of a ring. If set to null '{}', all predicates in the data are allowed

Return value

- `cyclePath` ： Returns a ring containing the nodes `u` and `v` (if any) in JSON form. If `u` or `v` is a variable, a loop is returned for each set of valid values of the variable
- `cycleBoolean` ： Return true if there is a ring containing nodes `u` and `v`; Otherwise, return false

The following query asks if there is a directed ring that contains Carol, a person Francis dislikes (Dave or Eve in the sample data), and whose edges can only be marked by a "like" relationship：

```
select (cycleBoolean(?x, <Carol>, true, {<喜欢>}) as ?y)where{    <Francis> <不喜欢> ?x .}
```

Results are as follows：

?y
"true"^^<http://www.w3.org/2001/XMLSchema#>

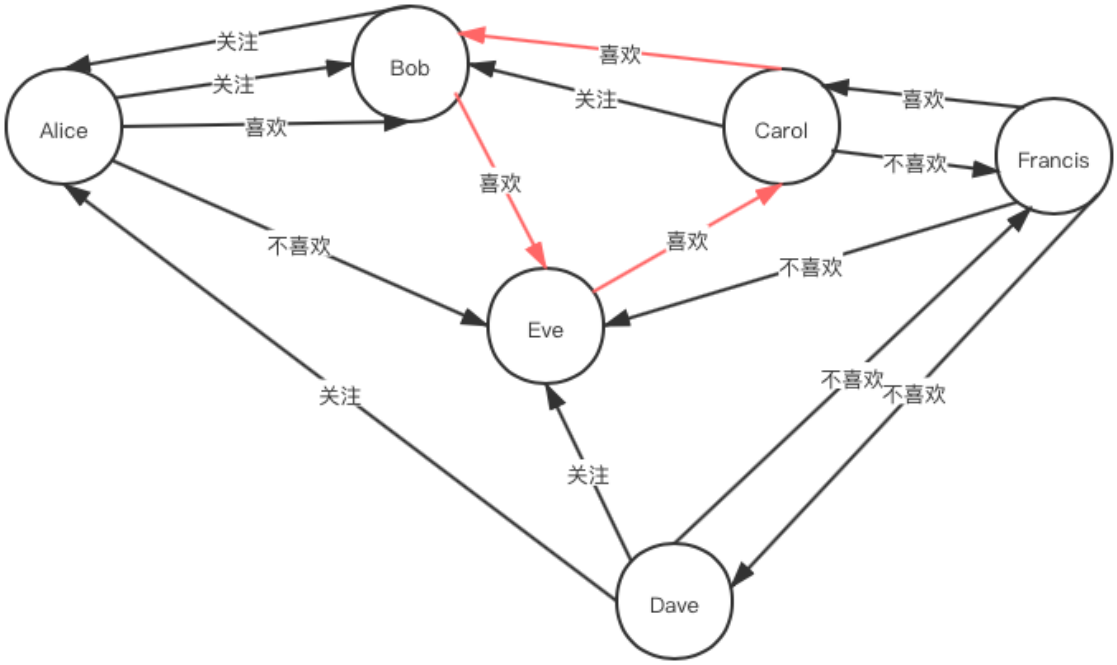
If you want to output a loop that meets the above criteria, use the following query:

```
SELECT (cyclePath(?x, <Carol>, true, {<喜欢>}) as ?y)
WHERE
{
    <Francis> <不喜欢> ?x .
}
```

Results are as follows，It can be seen that one of the rings satisfying the condition is formed by Eve likes carol-carol likes bob-bob likes Eve in sequence :(the escape of the outermost double quotation marks and the inner double quotation marks are omitted for ease of reading)

```
{
    "paths":[{
    "src":"<Eve>",
    "dst":"<Carol>",
    "edges":
    [{"fromNode":2,"toNode":3,"predIRI":"<喜欢>"},
{"fromNode":3,"toNode":1,"predIRI":"<喜欢>"},{"fromNode":1,"toNode":2,"predIRI":"<喜欢>"}],
    "nodes":
    [{"nodeIndex":1,"nodeIRI":"<Bob>"},{"nodeIndex":3,"nodeIRI":"<Carol>"},
{"nodeIndex":2,"nodeIRI":"<Eve>"}]
    }]
}
```

The red part below is the ring:



6.5.1.2 Shortest path Query

Query the shortest path from node `u` to node `v`

```
shortestPath(u, v, directed, pred_set)
shortestPathLen(u, v, directed, pred_set)
```

Used in SELECT statements, using the same syntax as aggregate function.

Parameter

`u` , `v` : Variable or node IRI

`directed` : Boolean value, true for directed, false for undirected (all edges in the graph are considered bidirectional)

`pred_set` : The set of predicates that are allowed to occur on the side that makes up the shortest path. If set to null '{}', all predicates in the data are allowed

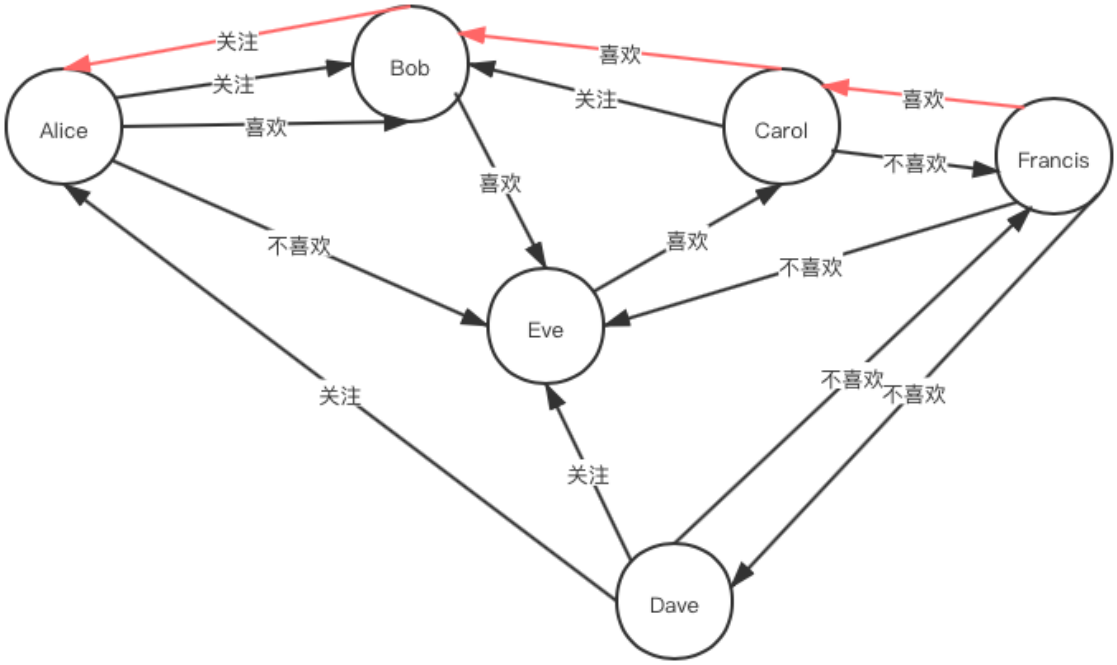
Return value

- `shortestPath` : Returns a shortest path (if reachable) from node `u` to `v` in JSON form. If `u` or `v` is a variable, a shortest path is returned for each set of valid values of the variable.
- `shortestPathLen` : Returns the shortest path length (if reachable) from node `u` to `v`. If `u` or `v` is a variable, return a shortest path length value for each set of valid values of the variable。

The following query returns the shortest path from Francis to a person (Alice in the example data) that Bob likes, cares about, or dislikes, and is not disliked by Francis, with a relationship that can be like or care about.

```
SELECT (shortestPath(<Francis>, ?x, true, {<喜欢>, <关注>}) AS ?y)WHERE{    <Bob>
?pred ?x .    MINUS { <Francis> <不喜欢> ?x . }}
```

The red part below is the shortest path:



Results are as follows：(For easy reading, the outermost double quotation mark and the escape of the inner double quotation mark are omitted.)

```
{   "paths":[{      "src":"<Francis>",      "dst":"<Alice>",        "edges":
[{"fromNode":4,"toNode":3,"predIRI":"<喜欢>"},
{"fromNode":3,"toNode":1,"predIRI":"<喜欢>"},{"fromNode":1,"toNode":0,"predIRI":"
<关注>"}],      "nodes":        [{"nodeIndex":0,"nodeIRI":"<Alice>"},
{"nodeIndex":1,"nodeIRI":"<Bob>"},{"nodeIndex":3,"nodeIRI":"<Carol>"},
{"nodeIndex":4,"nodeIRI":"<Francis>"}]       }]}
```

If you want to output only the shortest path length, use the following query:

```
SELECT (shortestPathLen(<Francis>, ?x, true, {<喜欢>, <关注>}) AS ?y)WHERE{ <Bob>
?pred ?x .    MINUS { <Francis> <不喜欢> ?x . }}
```

Results are as follows：(For easy reading, the outermost double quotation mark and the escape of the inner double quotation mark are omitted)

```
{"paths":[{"src":"<Francis>","dst":"<Alice>","length":3}]}
```

6.5.1.3 Reachability/K hop reachability query

Query reachable from node `u` to node `v`/reachable from node `u` to node `v`/reachable from node `v` to node `K`。

```
kHopReachable(u, v, directed, k, pred_set)kHopReachablePath(u, v, directed, k,
pred_set)
```

Parameter

`u`, `v` : Variable or node IRI

`k` : If it is set to a non-negative integer, it is the upper limit of the path length (query k-hop reachability). If set to negative, query reachability

`directed` : A Boolean value, true for directed and false for undirected (all edges in the graph are considered bidirectional

`pred_set` : The set of predicates that are allowed to occur on the side that constitutes a path. If set to null `{}`, all predicates in the data are allowed

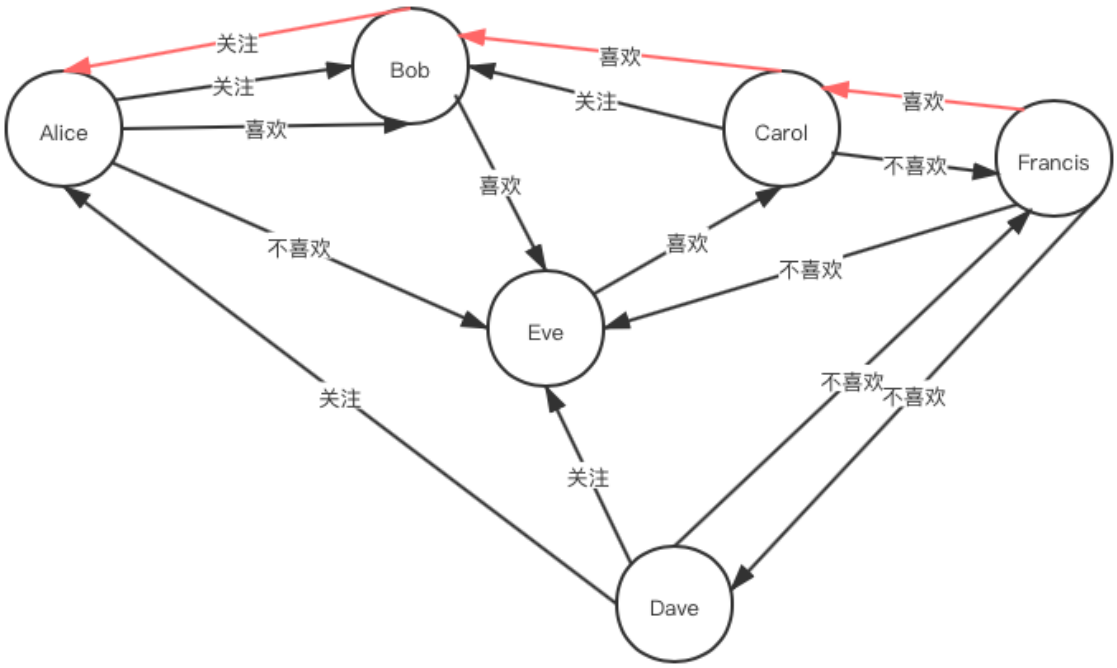
Return value

- `kHopReachable`: Return true if node `u` is reachable to node `v` (or K hop reachable, depending on the value of parameter `K`); Otherwise, return false. If `u` or `v` is a variable, return a true/false value for each set of valid values of the variable
- `kHopReachablePath`: Returns any path from node `u` to node `v` (if reachable) or a k-hop path, that is, a path with length less than or equal to `K` (if reachable, depending on the value of `K`). If `u` or `v` is a variable, return a path (if reachable) or a k-hop path (if reachable) for each set of valid values of the variable

The following query follows the example query from the previous section, "Shortest path Query" : It starts with Francis and ends with a person that Bob likes, cares about, or dislikes, and is not disliked by Francis (which is Alice in the example data). Ask if the relationship between the two people is 2 hops or within reach through liking or following.

```
SELECT (kHopReachable(<Francis>, ?x, true, 2, {<喜欢>, <关注>}) AS ?y)WHERE{
<Bob> ?pred ?x .    MINUS { <Francis> <不喜欢> ?x . }}
```

Since the shortest path length satisfying the condition is known to be 3:



Therefore, the above query results are false:

```
{"paths":[{"src":"<Francis>","dst":"<Alice>","value":"false"}]}
```

Francis and Alice, on the other hand, are reachable, but the shortest path length exceeds the above limit. So a query for reachability (with `k` set to negative) returns true:

```
SELECT (kHopReachable(<Francis>, ?x, true, -1, {<喜欢>, <关注>}) AS ?y)WHERE{
<Bob> ?pred ?x .    MINUS { <Francis> <不喜欢> ?x . }}
```

Results are as follows:

```
{"paths":[{"src":"<Francis>","dst":"<Alice>","value":"true"}]}
```

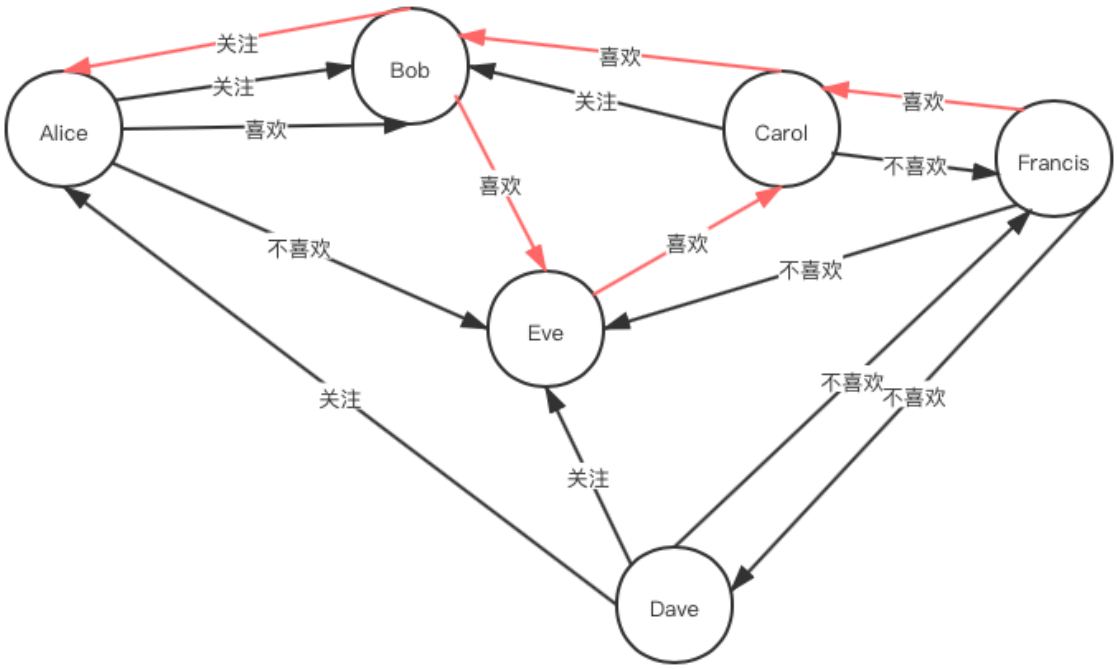
If you want to return a path that satisfies the condition between two people, you can call the `kHopReachablePath` function:

```
SELECT (kHopReachablePath(<Francis>, ?x, true, -1, {<喜欢>, <关注>}) AS ?y)WHERE{
<Bob> ?pred ?x .    MINUS { <Francis> <不喜欢> ?x . }}
```

The result may be the shortest path described above:

```
{   "paths":[{      "src":"<Francis>",      "dst":"<Alice>",        "edges":
[{"fromNode":4,"toNode":3,"predIRI":"<喜欢>"},
{"fromNode":3,"toNode":1,"predIRI":"<喜欢>"},{"fromNode":1,"toNode":0,"predIRI":"
<关注>"}],      "nodes":        [{"nodeIndex":0,"nodeIRI":"<Alice>"},
{"nodeIndex":1,"nodeIRI":"<Bob>"},{"nodeIndex":3,"nodeIRI":"<Carol>"},
{"nodeIndex":4,"nodeIRI":"<Francis>"}]       }]}
```

It could also be a non-shortest path with a ring in it, as shown in the figure below:



6.5.2 Personalized PageRank

Compute the value of PPR relative to `u`。

```
PPR(u, {v1, v2, ...}, pred_set)
PPR(u, k, pred_set)
```

Parameter

`u`： Variable or IRI, representing the source node

`{v1, v2, ...}`： Variable or IRI list, representing the target node. If the list is empty, all nodes in the graph are target nodes by default.

`k`： Integer, indicating that IRI of node with TOP-K PPR value and its corresponding PPR are returned (if the sum of points is less than K, IRI of all node and its corresponding PPR are returned)

`pred_set` ： The set of predicates to be considered (given empty, the default is to consider all predicates

Return value

The return value is in the following form, where SRC is the result of IRI or variable query corresponding to u. Which destination nodes DST contains depends on the second argument to the function; The corresponding PPR value is a double precision floating point number.

```
{"paths":
    [
        {"src":"<Francis>",
            [{"dst":"<Alice>", "PPR":0.1}, {"dst":"<Bob>", "PPR": 0.01}, ...]
        }
    ],
    ...
}
```

Example

```
PREFIX ns:   <http://example.org/ns#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>

SELECT (PPR(ns:a, 10, {ns:pred2}) AS ?z)
WHERE
{
    ?y ns:pred1 ns:b .
}
```

7. gStore Visual Tool Workbench

7.1 Installation and deployment

gStore Workbench is a web tool developed by gStore team for online management of gStore graph database and query visualization of gStore. Currently, gStore official website provides Workbench download, the download link is http://www.gstore.cn. Select [Product] - [gStore Workbench] and after filling in the relevant information, you will get a Workbench package, but you will need to install and deploy. The steps to install and deploy are described in detail below

7.2 Download tomcat

Workbench is a Web site that requires a Web server as a Web container to run, and we recommend tomcat8 as the Web server to download from https://tomcat.apache.org/download-80.cgi After downloading the zip package, unzip it

- Put the Workbench package in tomcat's webapps directory and unzip it
- To the bin directory of tomcat
- To the bin directory of tomcat

  [root@node1 bin]# ./startup.sh
- Stop Tomcat:

```
[root@node1 bin]# ./shutdown.sh
```

7.3 Lgoin

7.3.1 Browser Access to the System

Login address is:

```
http://workbench`Self-deployed servers`ip:8080/gworkbench/views/user/login.html
```

7.3.2 Connect to gStore instance

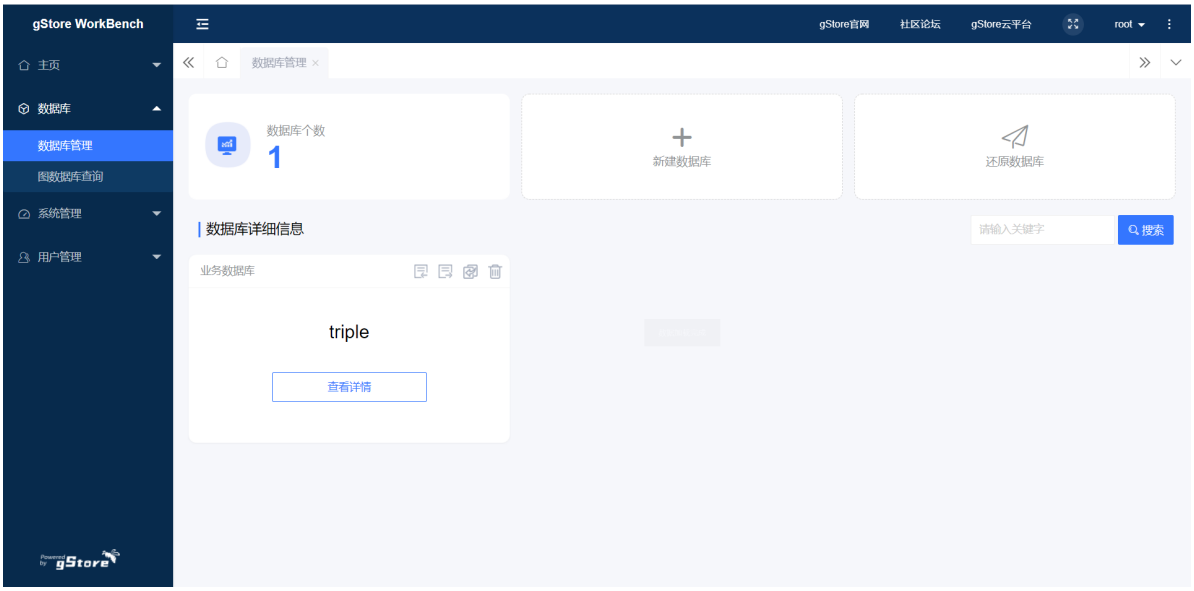
Set the IP address and port of the remote server and save the IP address and port number of the remote server to the gStore diagram database management system. Note that the remote server must install gStore and start the GHTTP service

Enter the user name, password, and verification code to log in to the gStore graph database management system on the saved server (gStore default user name root, password 12345)

7.4 Query function

7.4.1 Database management

- **View information about the loaded database**



Click on the database to see the details of the database

- Create a database

1.Enter the name of the new database, for example, lubm

2.Upload files in two ways:

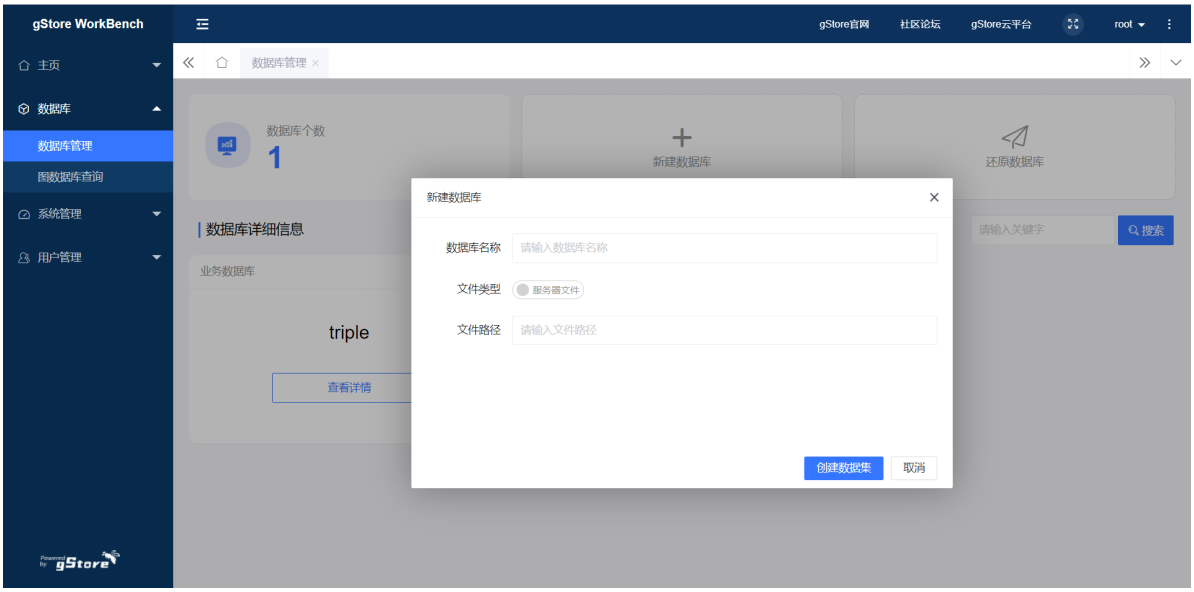
One is to upload from the server. Enter the correct path of the NT file or N3 file. You can enter the absolute path or the relative path.

For example, path selection

```
/root/gStore/data/lubm.nt    绝对路径
./data/lubm.nt               相对路径
```

The other option is to upload from local. Note that using this approach you must ensure that the **Workbench server is the same server on which gStore is installed**. First select the NT or N3 file locally, then click Upload File。

3.Click Create Dataset

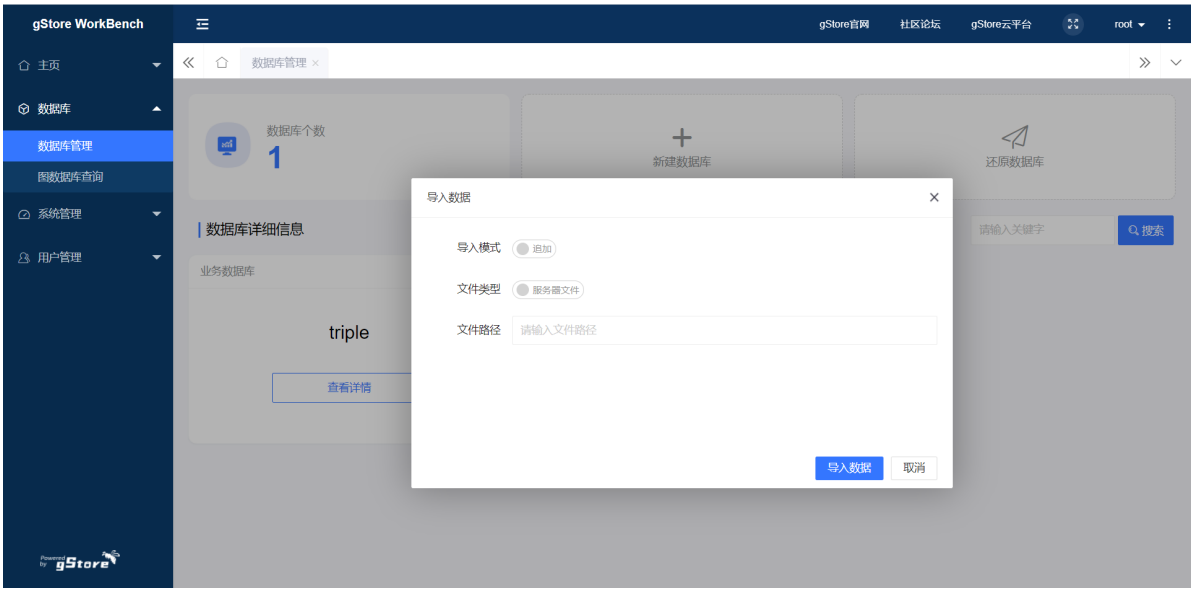


- **Database deletion**

Click the delete button in the upper right corner of the database and select Delete or delete completely to delete the database. **system The database cannot be delete**

- **Import data**

Click "Database Management", select the database to import, click the import icon in the upper right corner, import mode can be "Add" and "overwrite"; The file type can be server file or local file. To import local files, select nt or N3 format files, click "Upload File", and then click "Import Data".



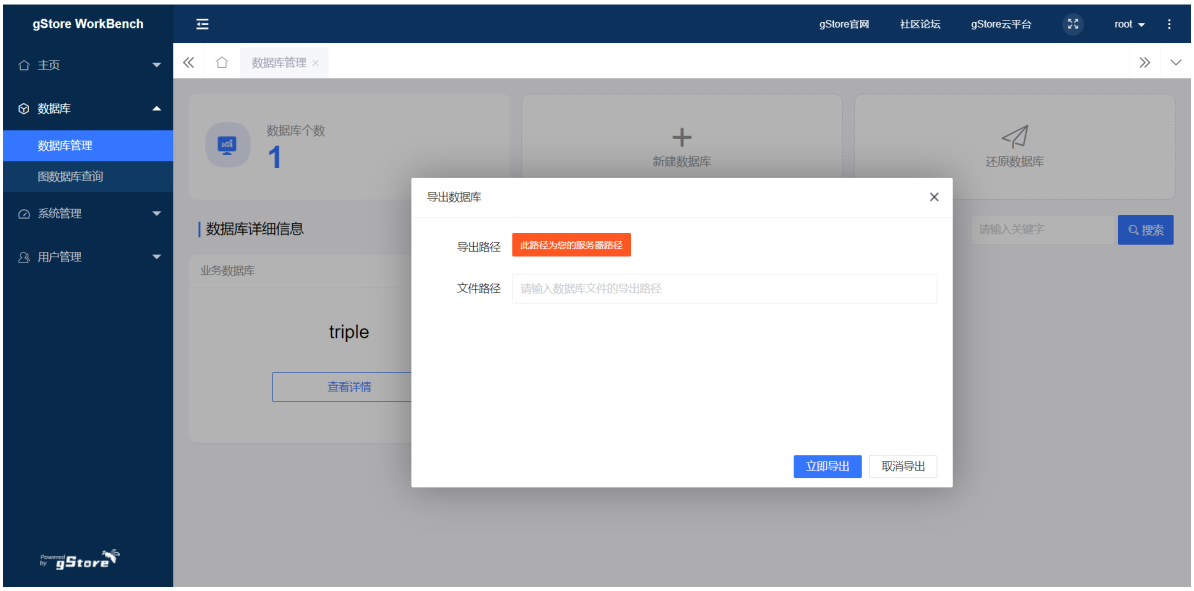
- **Export data**

To export the database as an NT file, click the export button in the upper right corner of the database and select the directory where the exported NT file resides. You can enter an absolute path or a relative path. If you want to enter a relative path, note that the current path is the root directory of gStore installation.

For example, path selection

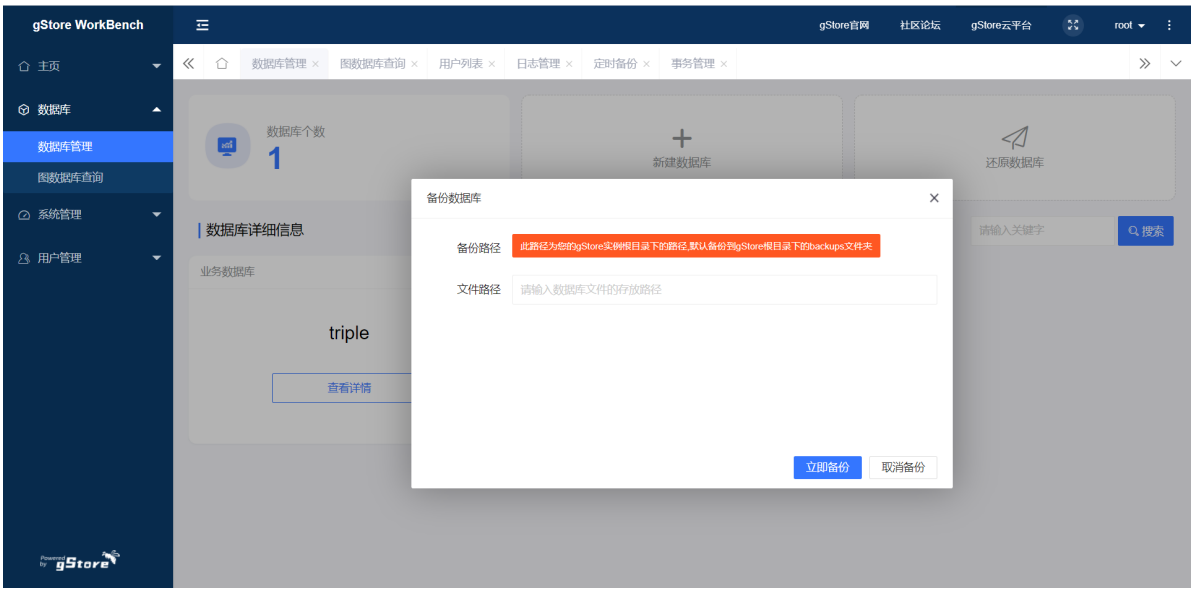
```
/root/gStore/data    绝对路径
./data               相对路径
```

Enter the correct path and click Export Now. The **system database cannot be exported**

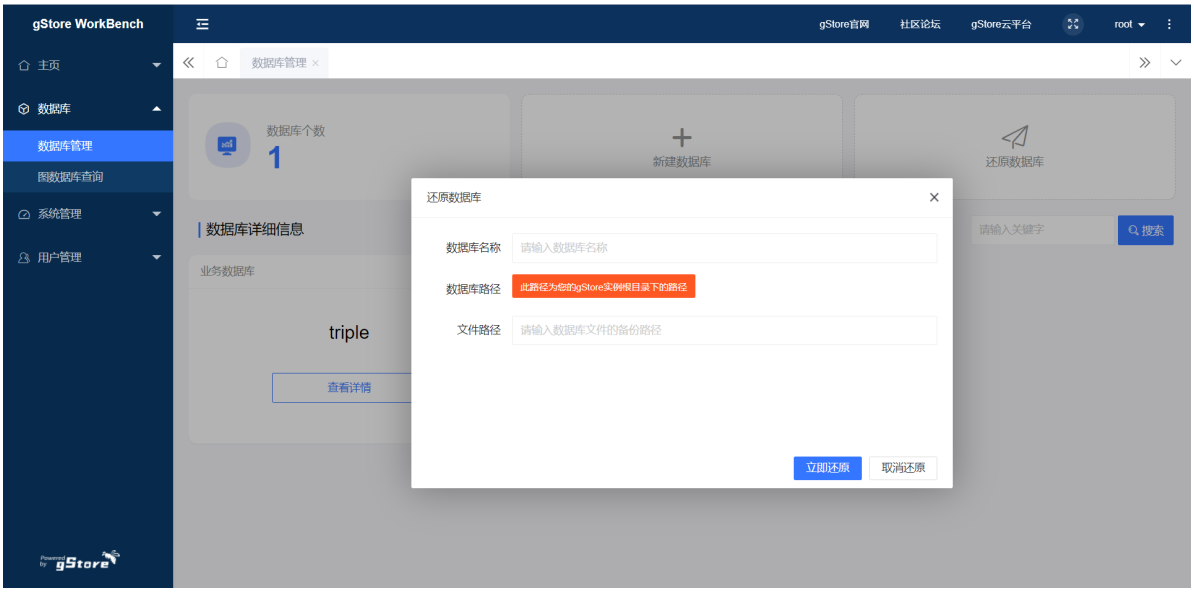


- **Backup Database**

Click the "Back up" button on the database you want to back up, and the following dialog box pops up.



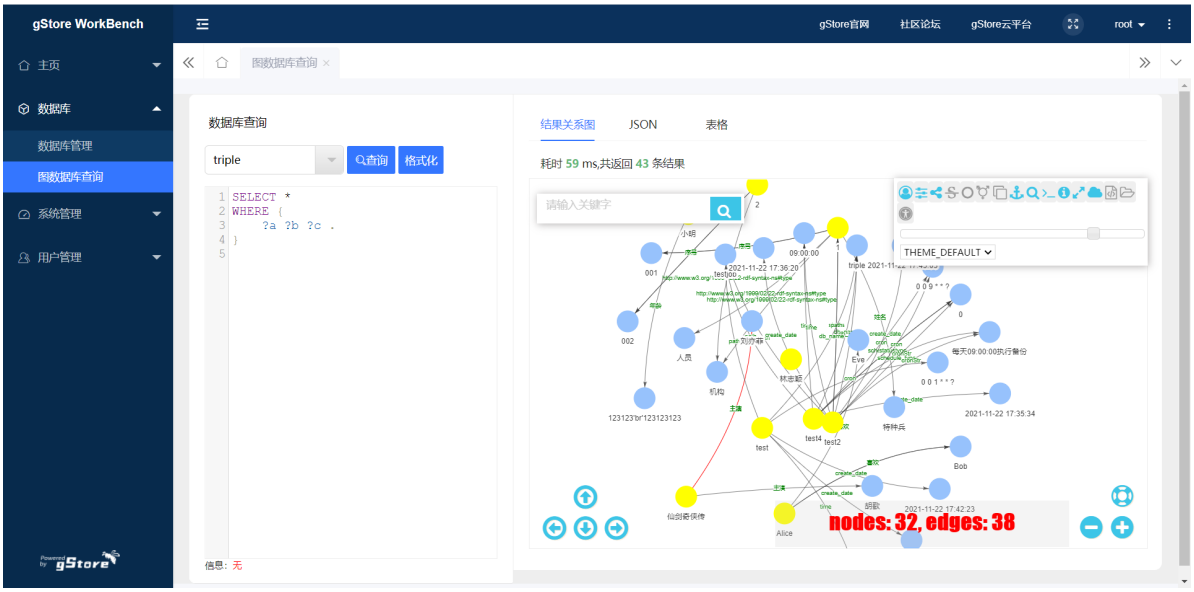
- Restore database



7.4.2 Graph database query

- Select the database to query
- Enter the query statement according to the SPARQL document and return the result graph, JSON, and table.

  Graphical display



Json display

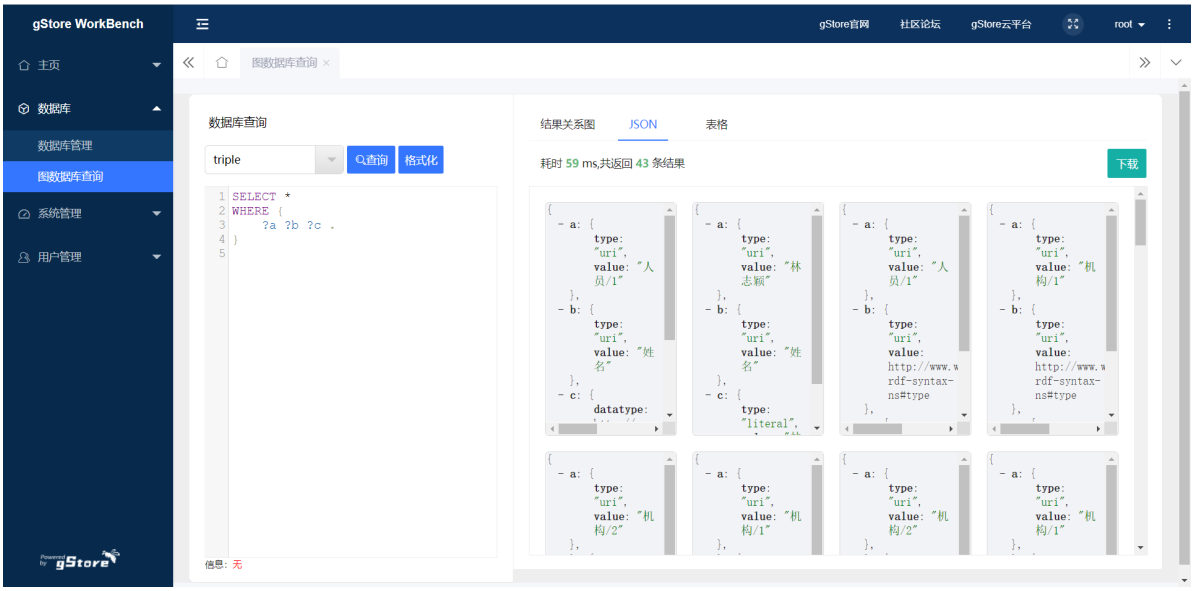
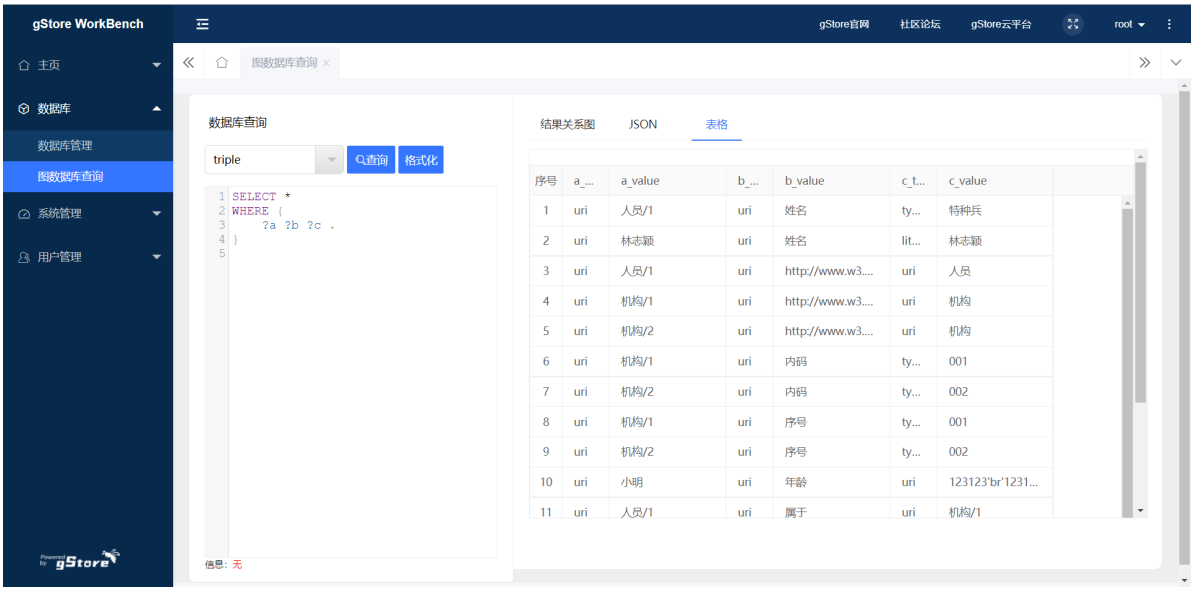


Table display

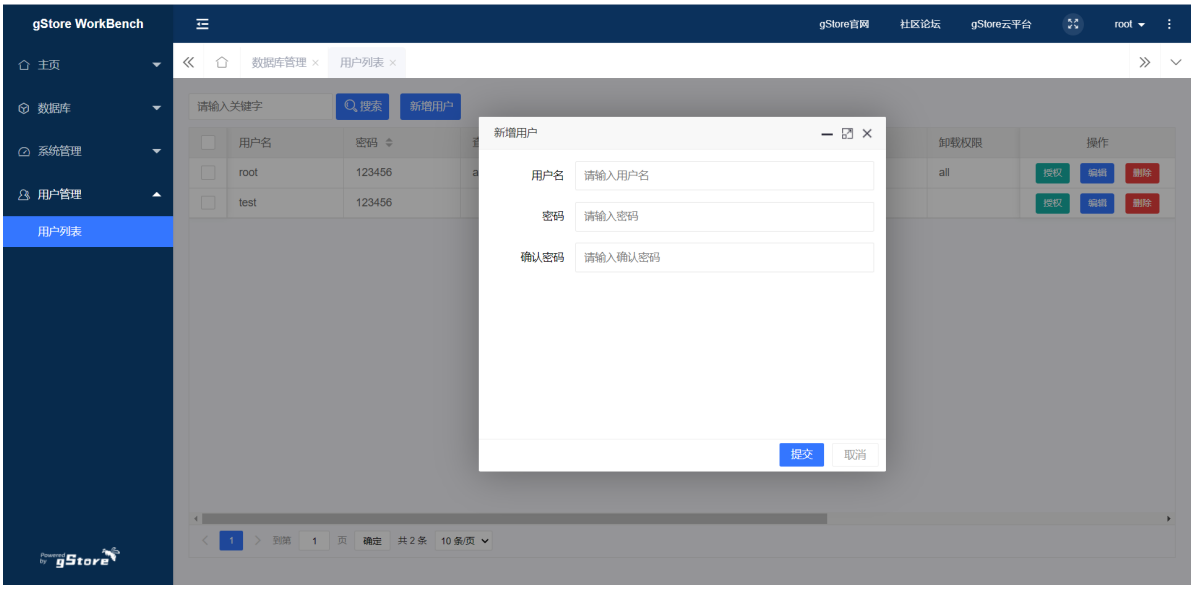


7.5 User management（Only for root user)

7.5.1 Add user

- add new users

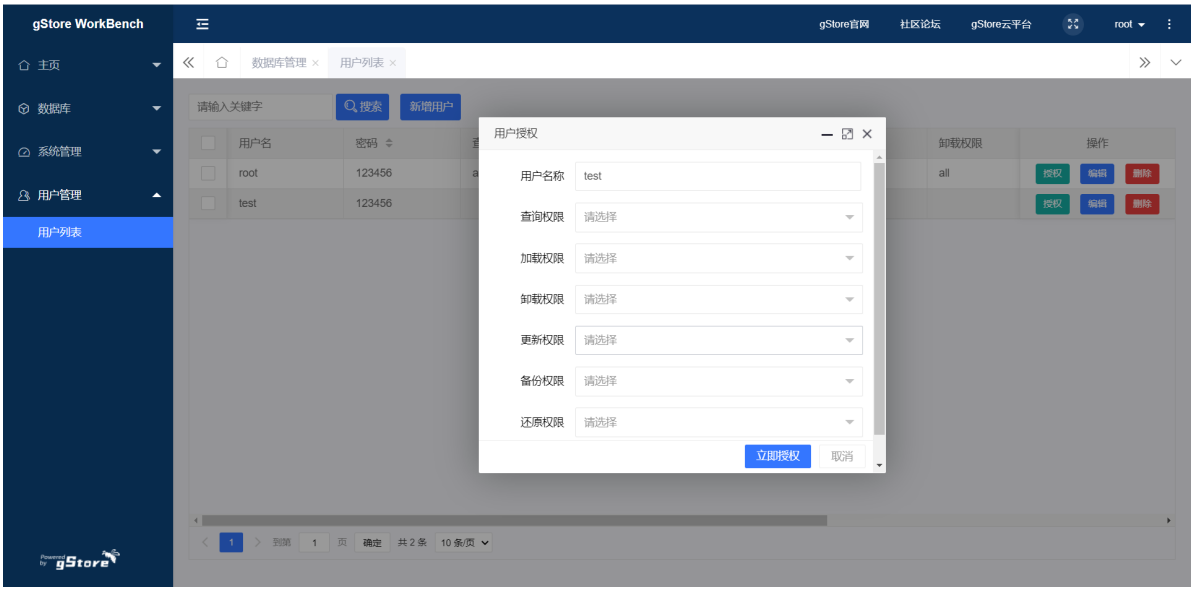
Enter the user name and password to add a user



7.5.2 User authorization

- Authorize functions for users

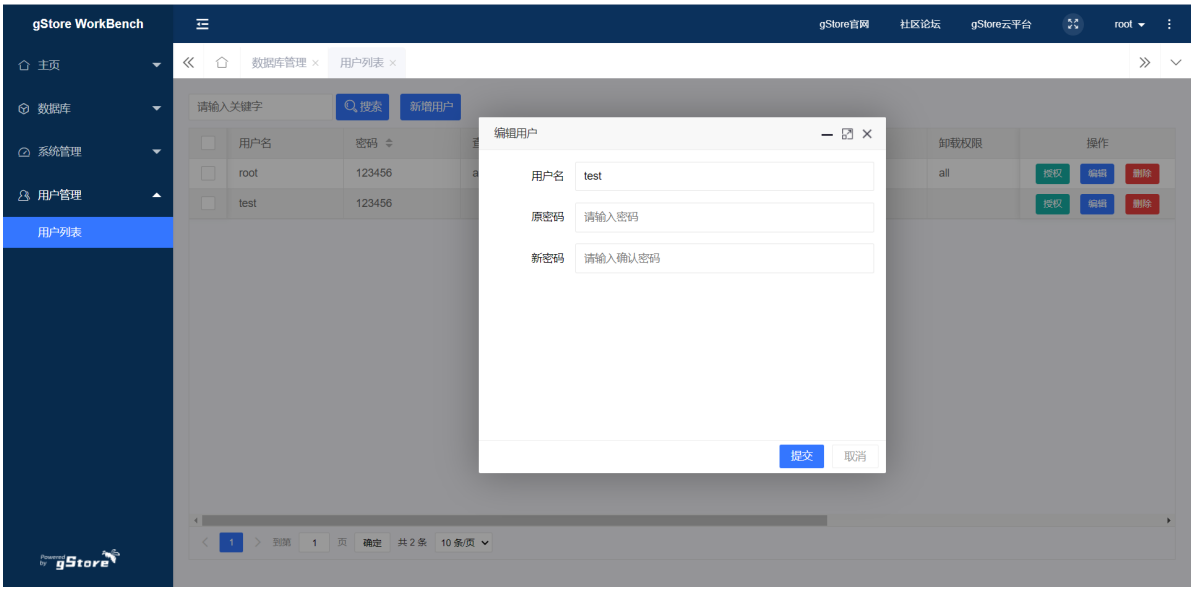
Select the users and databases that you want to authorize and add or remove query, load, unload, update, backup, restore, and export permissions.



7.5.3 Edit account

- Edit user account details

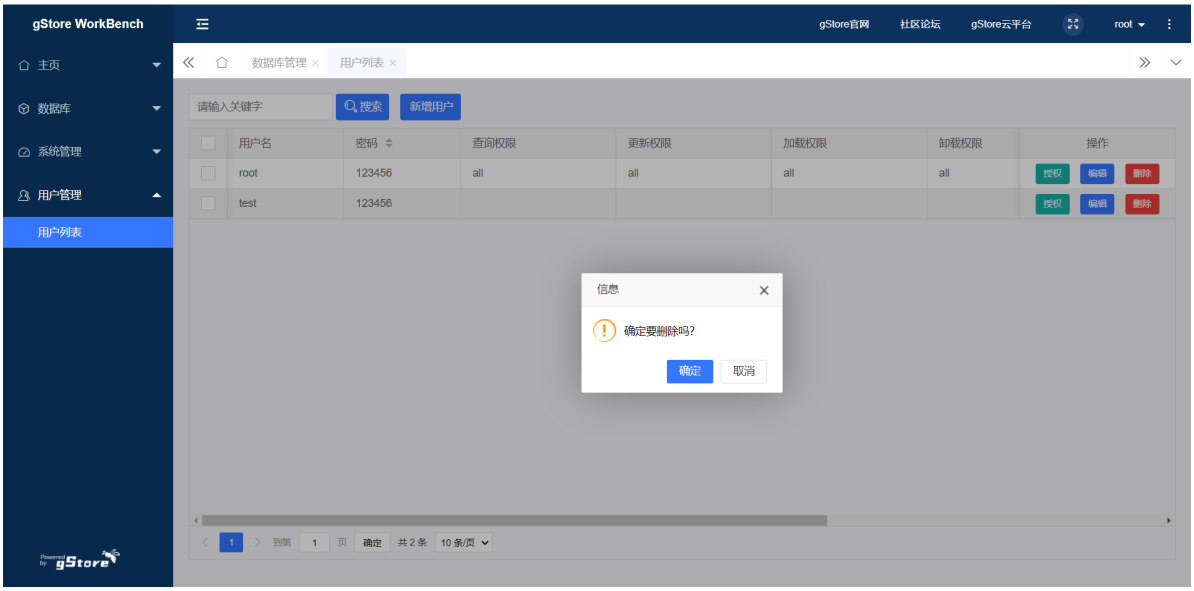
Click [User Management], select a User account, click [Edit] under the operation bar, input relevant information and click [Submit] to modify the User password



7.5.4 Account deleted

- Delete user accounts

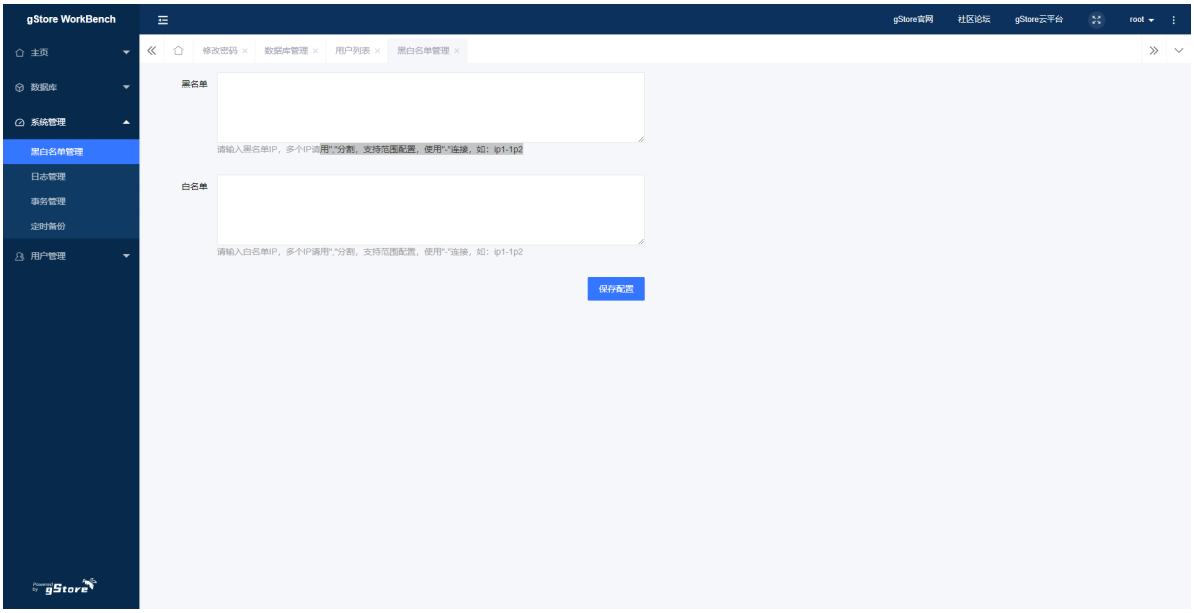
Click [User Management], select a User account, and click [Delete] under the operation bar to delete the User.



7.6 System management （Only for root user)

7.6.1 Blacklist and whitelist management

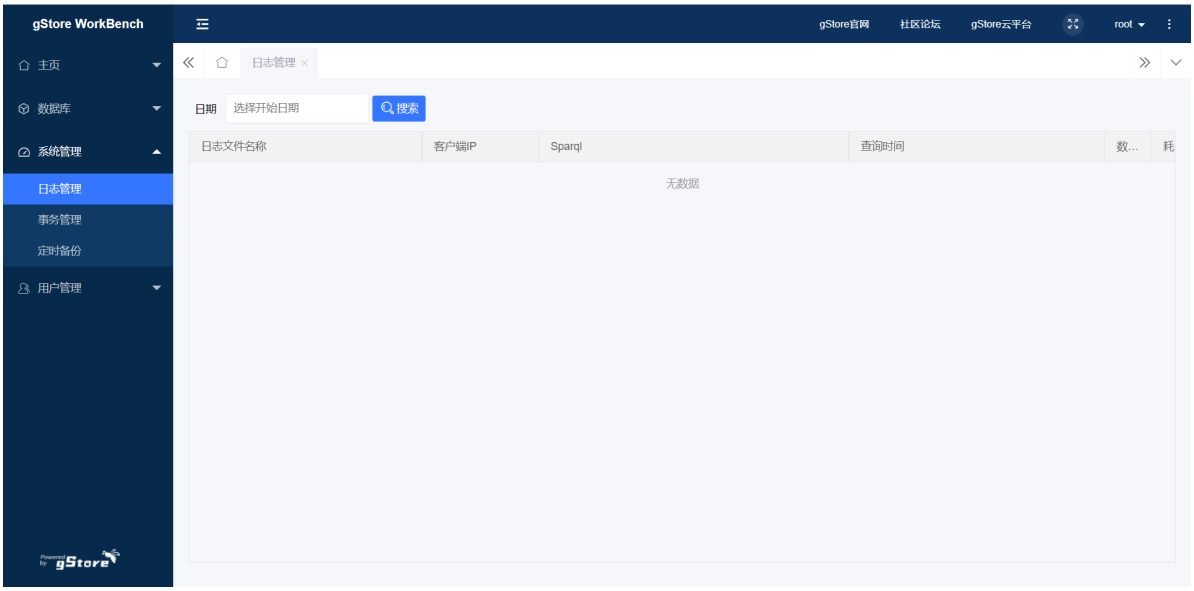
Enter the blacklist and whitelist IP address, separated by commas (,), support range configuration, use "-" connection, such as: IP1-1P2.



7.6.2 Log Management

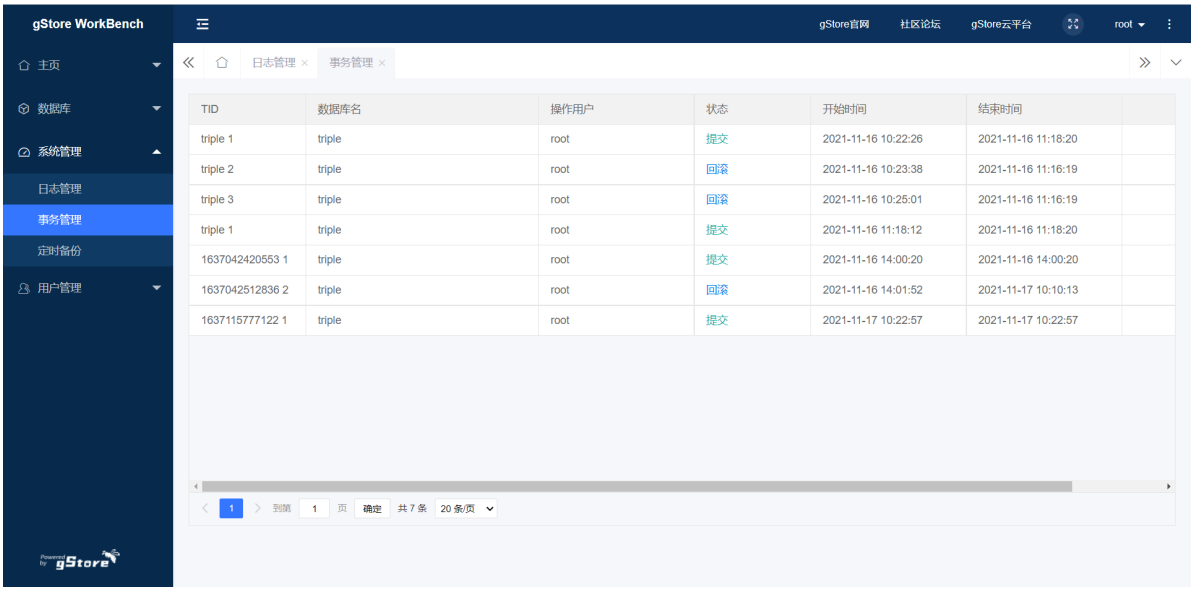
- User can view system logs on the web UI

Click on the "system management", "log management", you can see the detailed log information, including name of the log file, the client IP, SPARQL, query time, data format, the time-consuming (ms) and the number of results choose need to authorized users and database, add or delete query, loading, unloading, update, backup, restore and export rights.



7.6.3 Transaction management

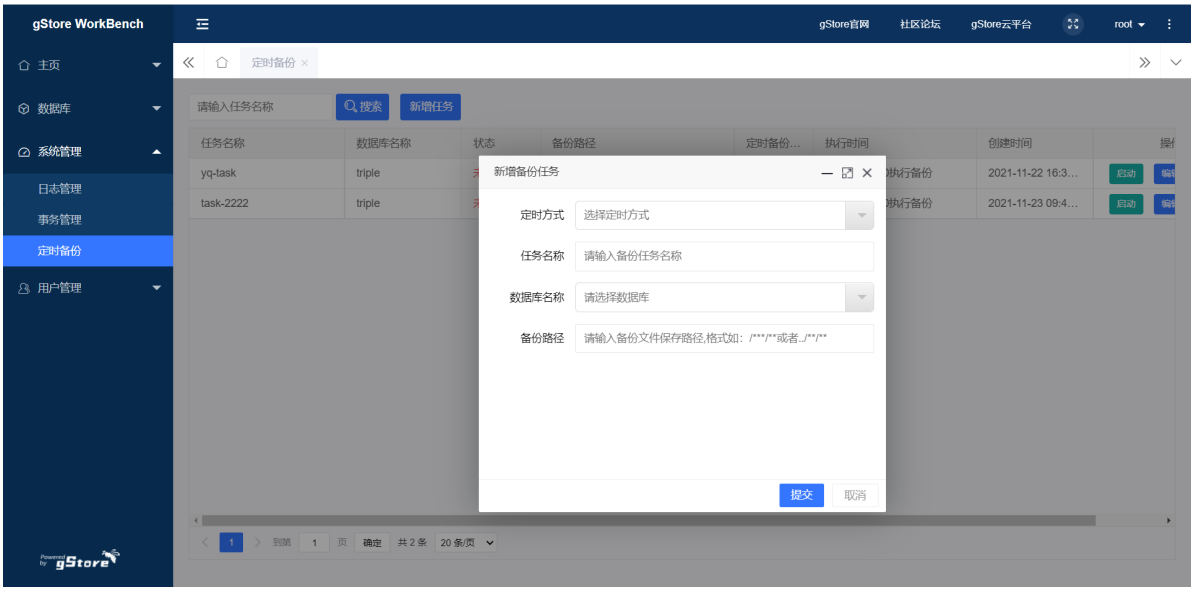
Click "System Management" - "Transaction Management" to view specific transaction information, including TID, database name, operation user, status, start time, end time, etc.



7.6.4 Scheduled backup

- backup user account

Click "Scheduled Backup" and then "New Task" to add a new backup task by filling in the time mode, task name, database name and backup path.


gStore WorkBench
gStore官网
社区论坛
gStore云平台
root
主页
数据库
系统管理
日志管理
事务管理
定时备份
用户管理
定时备份
请输入任务名称
搜索
新增任务
任务名称
数据库名称
状态
备份路径
定时备份...
执行时间
创建时间
yq-task
triple
执行备份
2021-11-22 16:3...
启动
task-2222
triple
执行备份
2021-11-23 09:4...
启动
新增备份任务
定时方式
选择定时方式
任务名称
请输入备份任务名称
数据库名称
请选择数据库
备份路径
请输入备份文件保存路径,格式如：/***/**或者../**/**
提交
取消
到第
页
确定
共 2 条
20 条/页

8. gStore Cloud platform user manual

8.1 Brief introduction

8.1.1 What is gStore?

gStore is a graph-based RDF triplet storage data management system developed by the Data Management Laboratory of Wangxuan Institute of Computer Technology of Peking University . It can be used to manage huge interconnected data. It has four advantages: original innovation, standard system, superior performance, and independent control

8.1.2 What is gStore cloud platform?

The gStore cloud platform is the cloud service version of the gStore system. It can be used after being registered online and approved. No download or installation is required.

8.1.3 What is the use of gStore

gStore can be used for large-scale data processing, which gives it a wide range of uses, including but not limited to government big data, fintech, smart healthcare, artificial intelligence, etc.

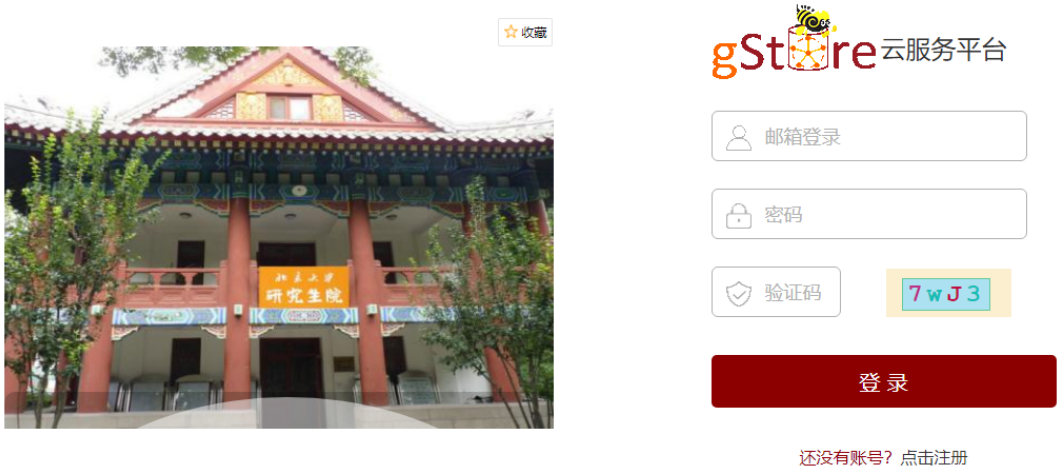
8.1.4 How does gStore play a role in the above transactions?

Taking fintech as an example, the system can query multi-level equity through graph database. In this case, up to five layers of equity relationship data can be found.

8.2 How to use

8.2.1 Registration and Login

Cloud Platform website：http://cloud.gstore.cn



If you are using the gStore cloud platform for the first time, you need to register. The registration page is as follow:

After registration and approval, you can log in to the cloud system.

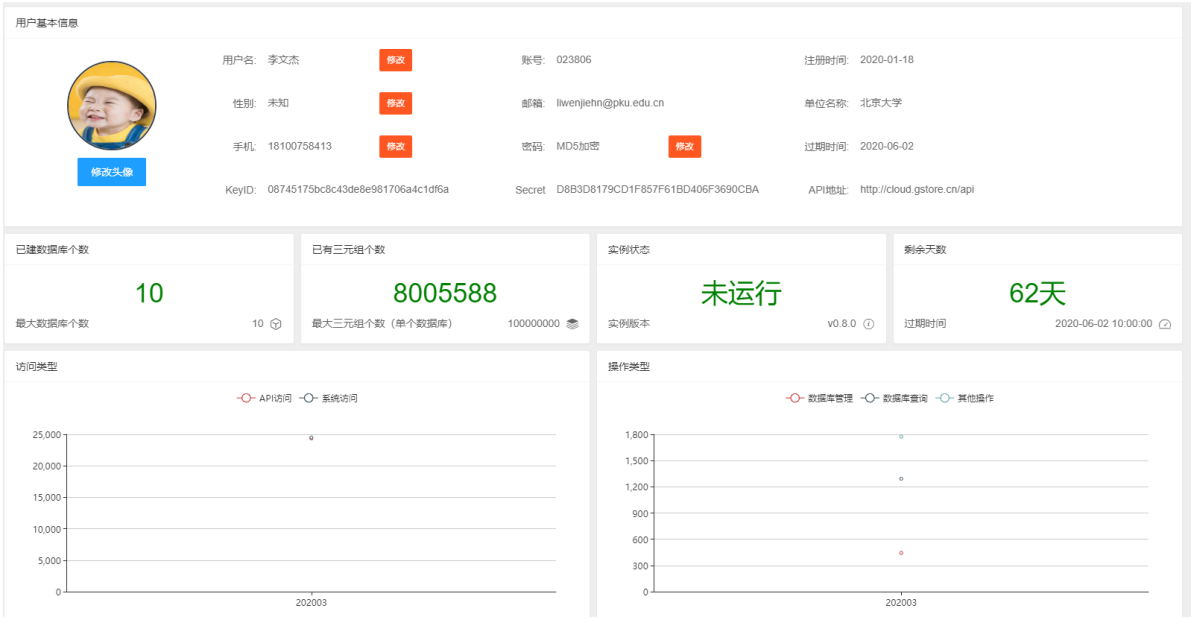
8.2.2 Platform home page

The homepage of the platform is shown in the figure below, which will display the number of currently built databases, total number of triples and expiration time, as well as information related to the platform (including news and version information, etc.). Click the relevant information to view the details, as well as some commonly used links such as gStore official website and team information.

8.2.3 Personal center

The personal center is in the upper right corner of the gStore page.

After logging in to the personal center, you can view basic user information and operation logs of this week. The basic user information includes KeyID and Secret, which are used as keys when other programs interconnect with gStore cloud platform.



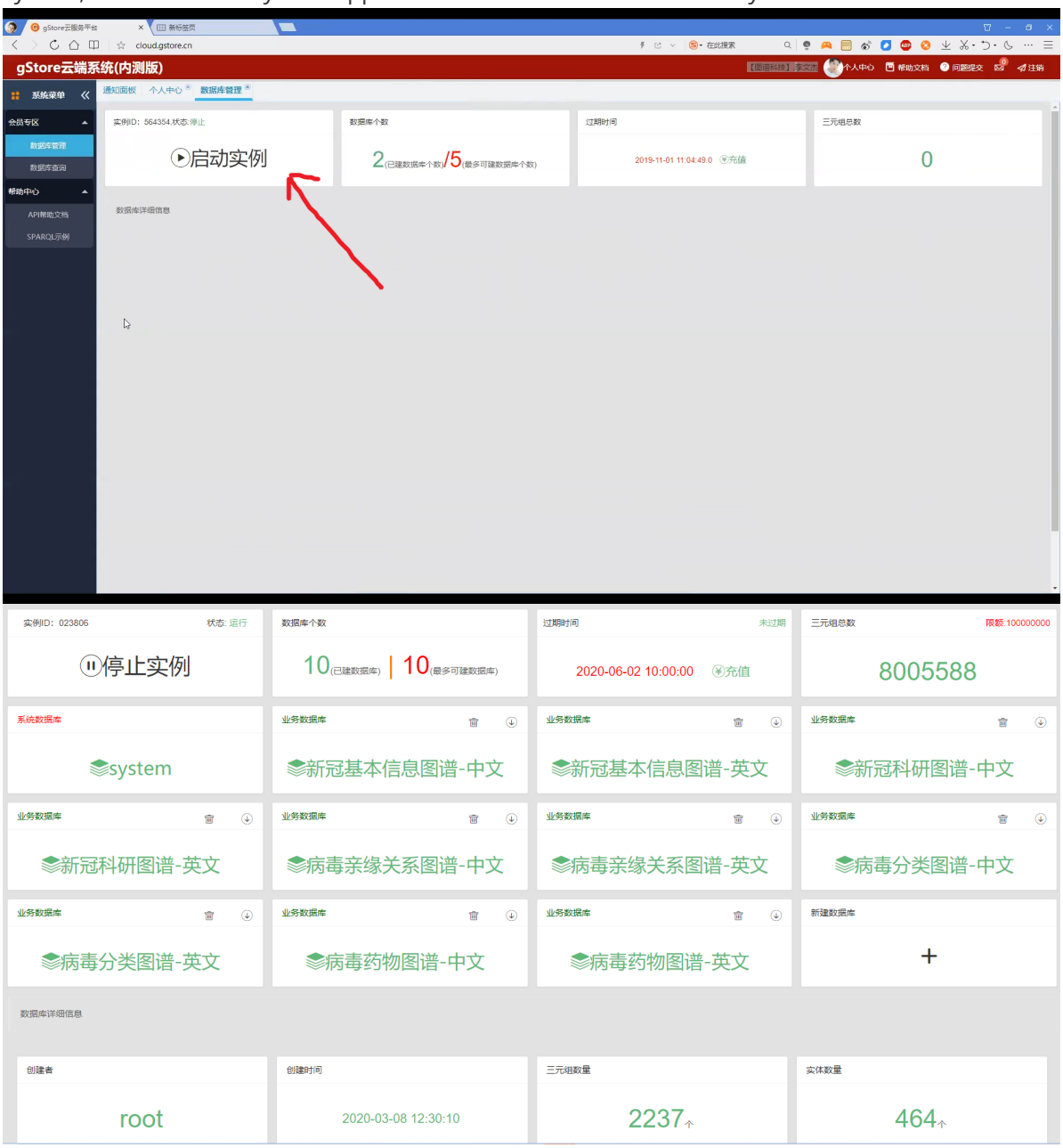
8.2.4 Database management

The left area is the system menu, including members area and help center two areas.

The member area is divided into two functional modules, database management and database inquiry.

8.2.4.1 Starting a database instance

There is one important feature in database management: instances. When first entering the system, the instance may be stopped and need to be started manually.



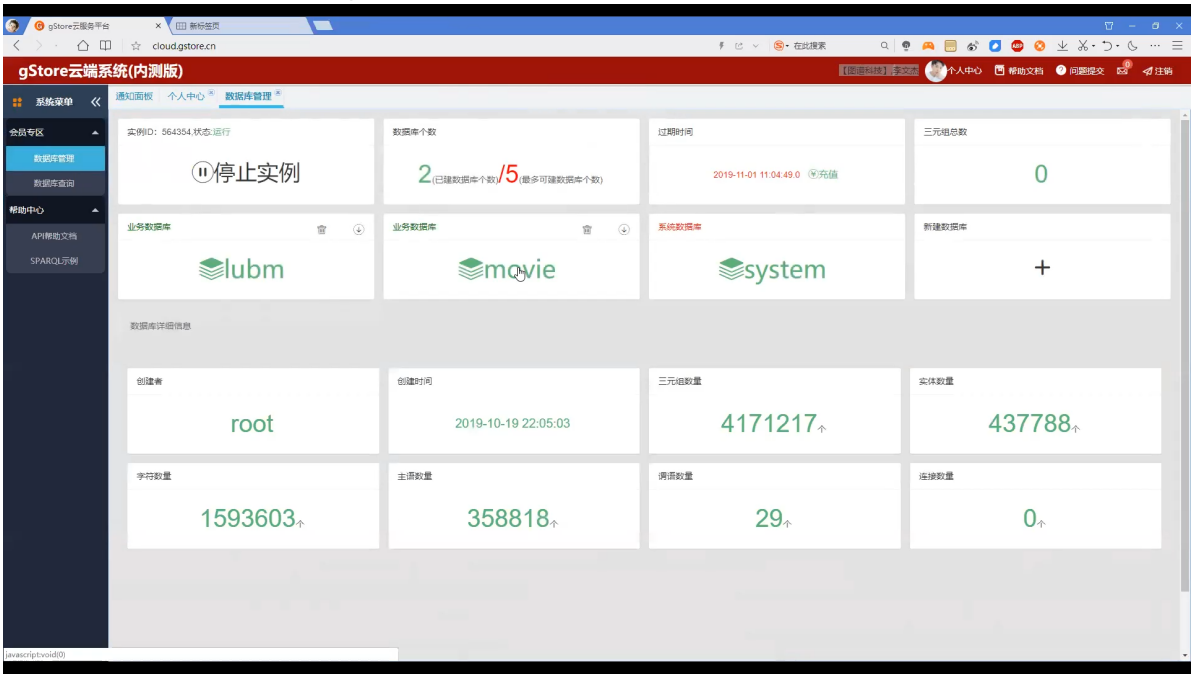
8.2.4.2Viewing database Information

After the instance is started, you can see the state of the instance, the number of databases created and the maximum number of databases that can be created, the gStore expiration time, the total number of triples, and so on in the above row.

Below, you can see the databases that have been created, including a system database (created by the system, not operable, not included in the maximum number of databases that can be created) and several business databases (created by yourself, operable).

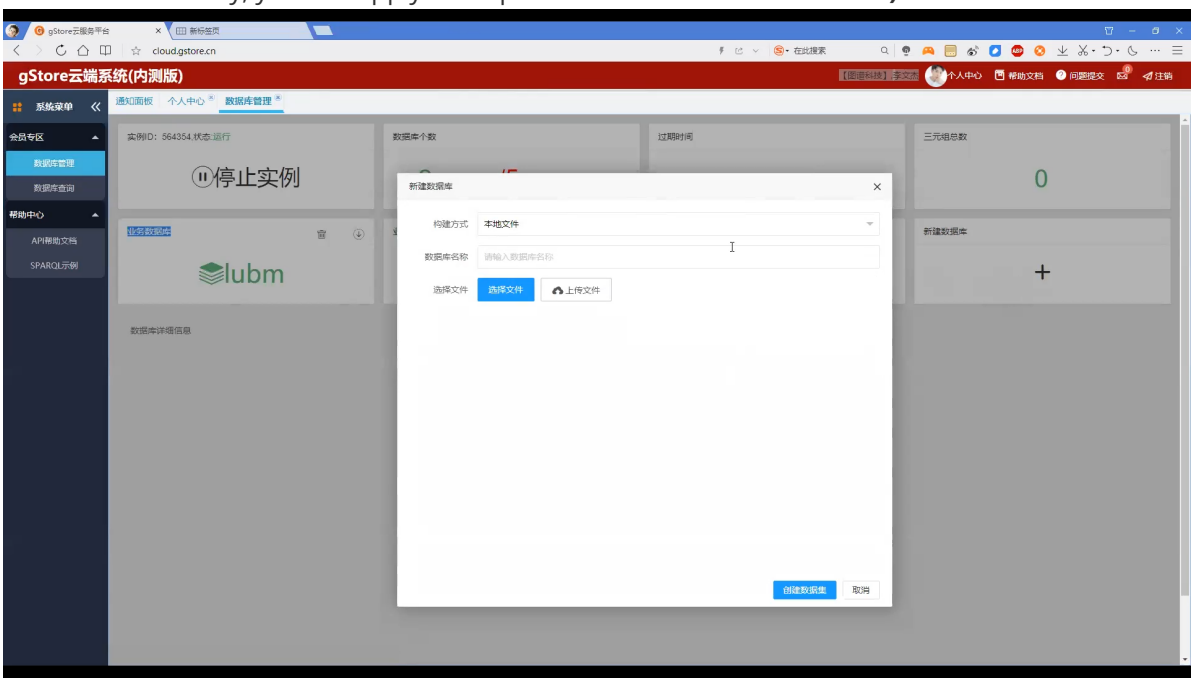
You can create, delete, export, and obtain information about a service database.

Click on a database to get information about it, including creator, creation time, number of triples, number of entities, number of characters, number of subjects, number of predicates, number of connections, and so on. The picture below shows information about the Movie database:

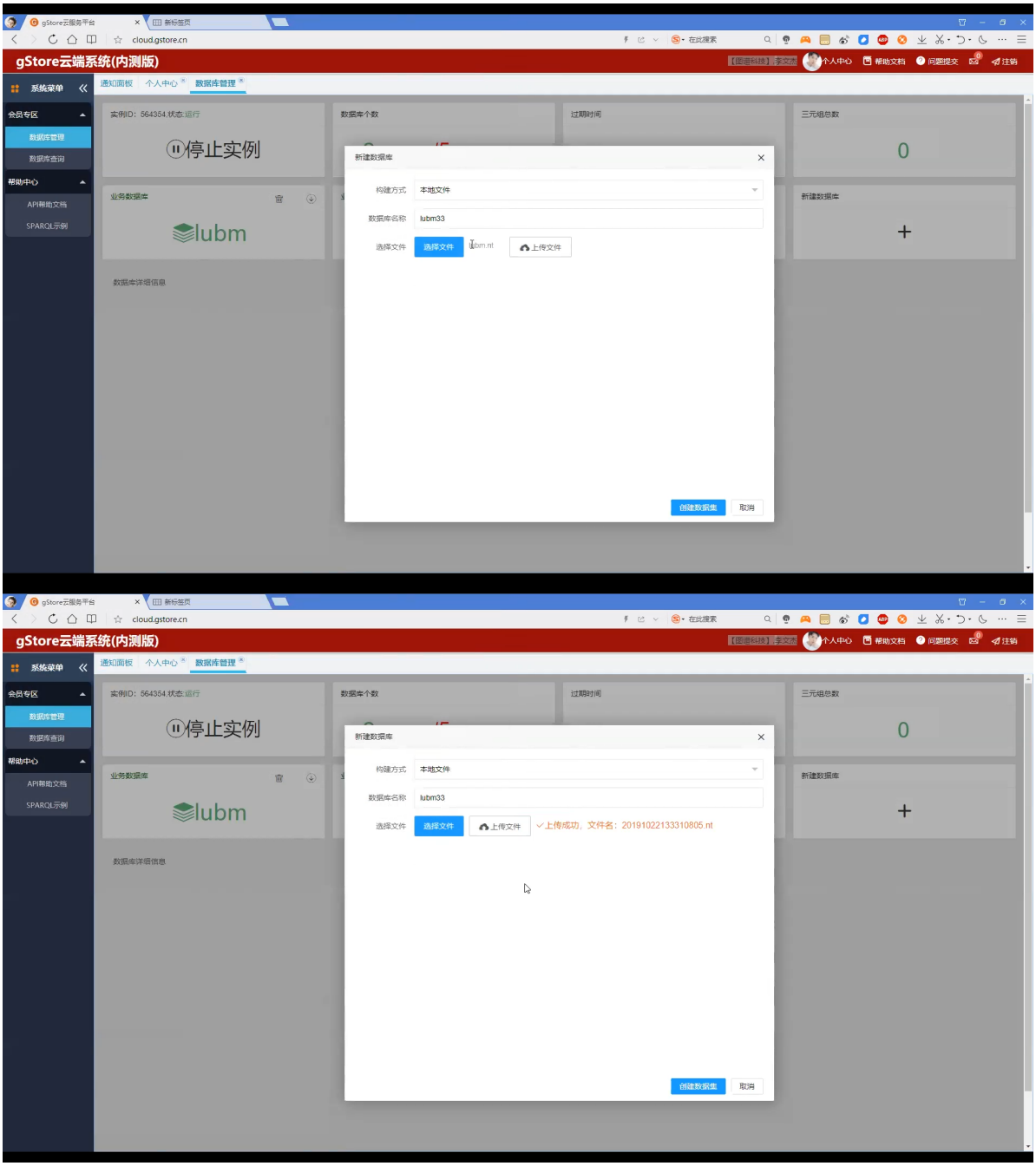


8.2.4.3 Creating a database

Click the plus sign to create a database: (** Due to limited resources, the number of databases created by each user is limited to 5, and the number of triples of each database is limited to 1 million. If necessary, you can apply for expansion from the administrator **)

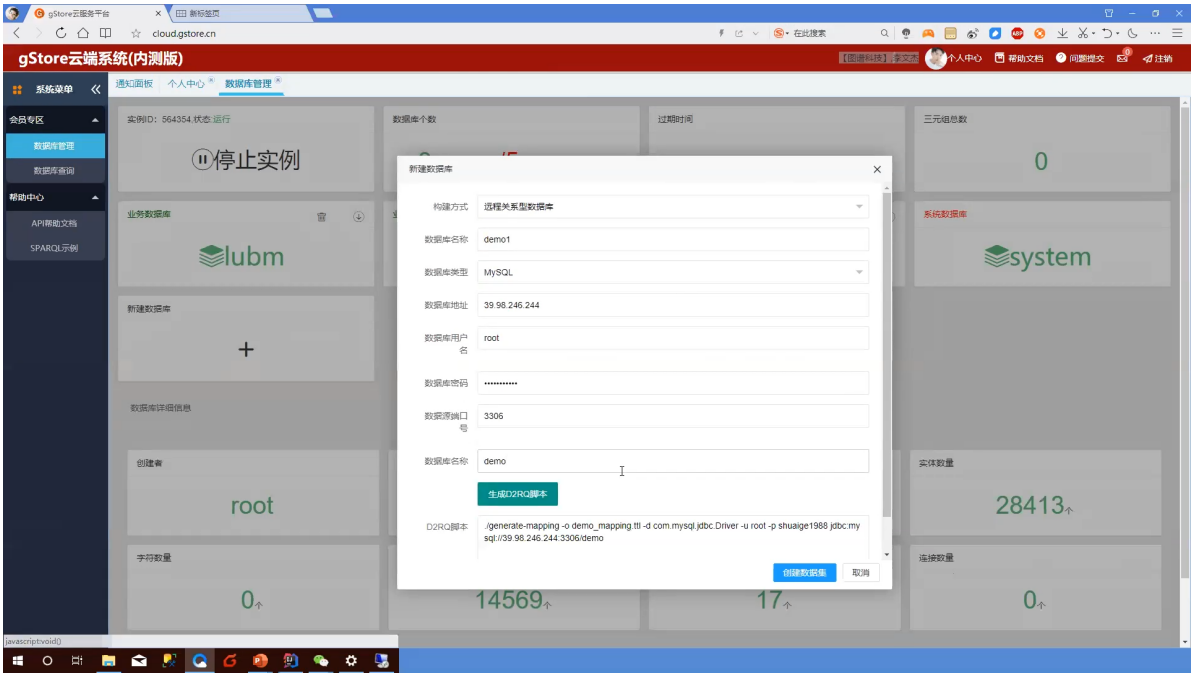


There are three ways to create a database. The first is a local file, that is, upload a file from the local PC to the server. Currently, the system only supports NT files, but n3 files may be supported in the future



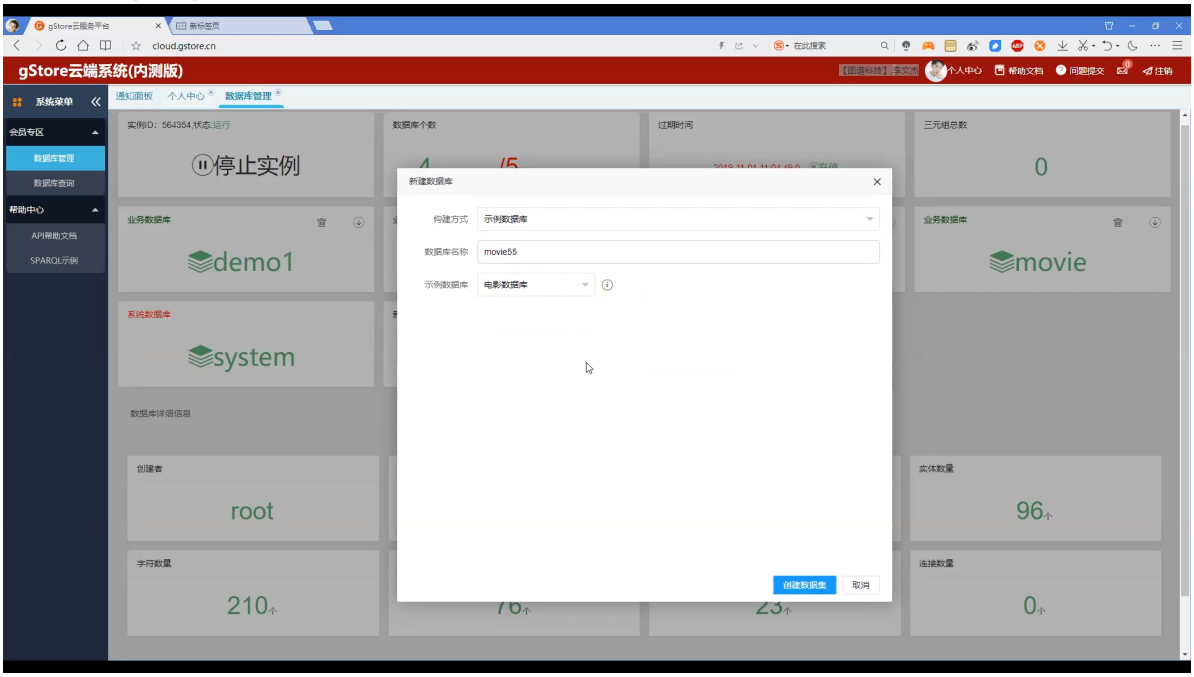
Note that the file size cannot exceed 2GB and the number of lines cannot exceed 1 million

The second way to create a database is a remote relational database, which remotely accesses a database on the network and imports it into the cloud platform. The cloud platform supports four relational databases: MySQL, Oracle, SQLServer, and Postgre. When creating a database, you need to input its related information and generate a D2RQ script to generate a database.



Note that you need to enter two names, the first name is the name of the database you created and the second name is the name of the database you connected to.

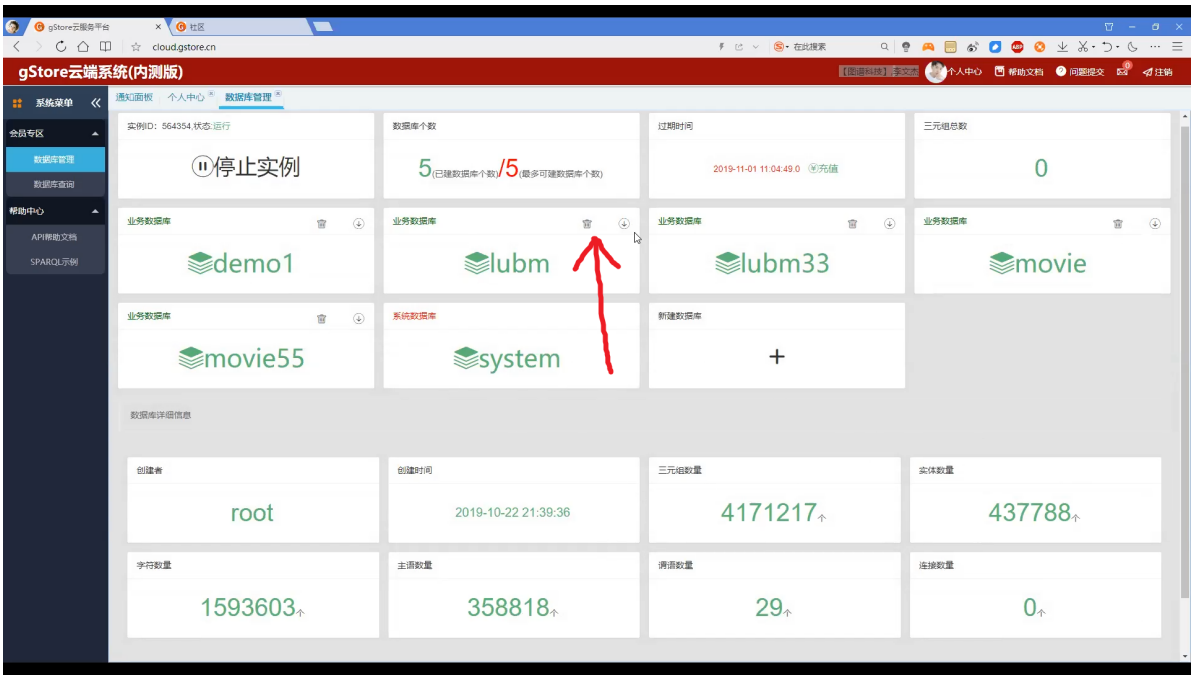
The third way to create a database is the Example database. Currently, the movie database is the only sample database in the cloud platform, but it will be added over time. The movie database, for example, contains more than 4 million triples containing information about movies, directors, actors, release dates, movie ratings, and more. (**Sample database triples are not limited to 1 million triples per database**)



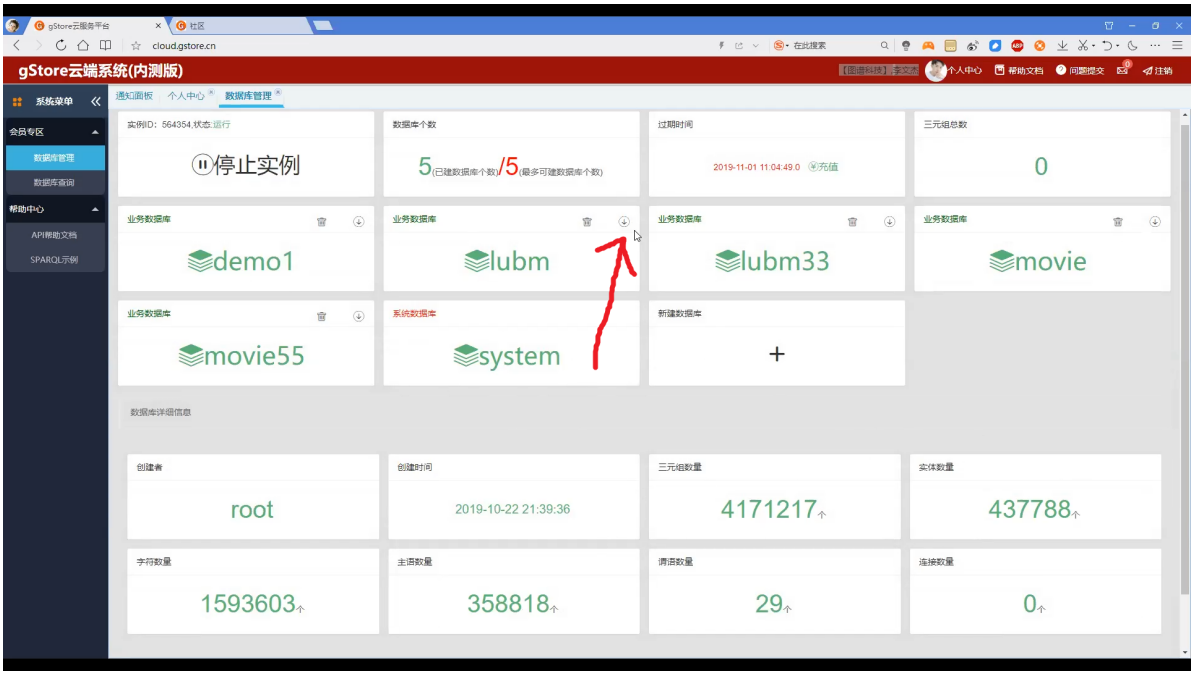
8.2.4.4 Deleting a Database

You can delete a database by clicking on the trash can icon in the upper right left corner of the database.

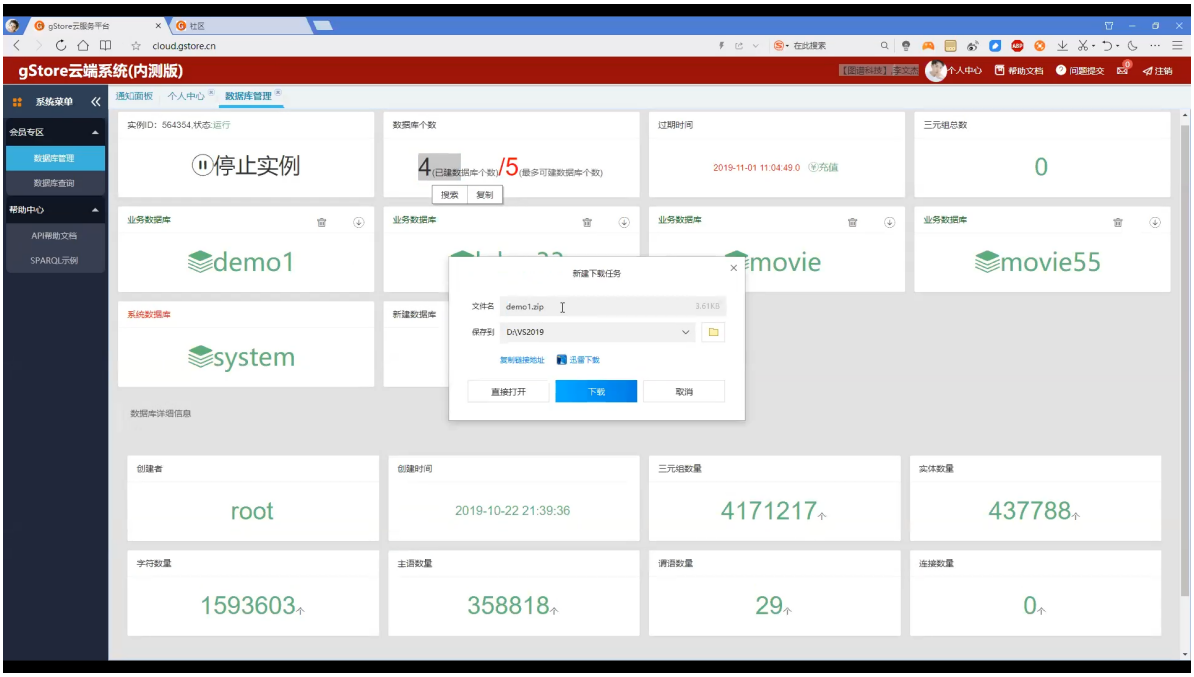
The system provides a 15-day recovery period for all deleted databases to prevent accidental deletion.



You can export a database by clicking the down arrow icon in the upper right corner of the database.



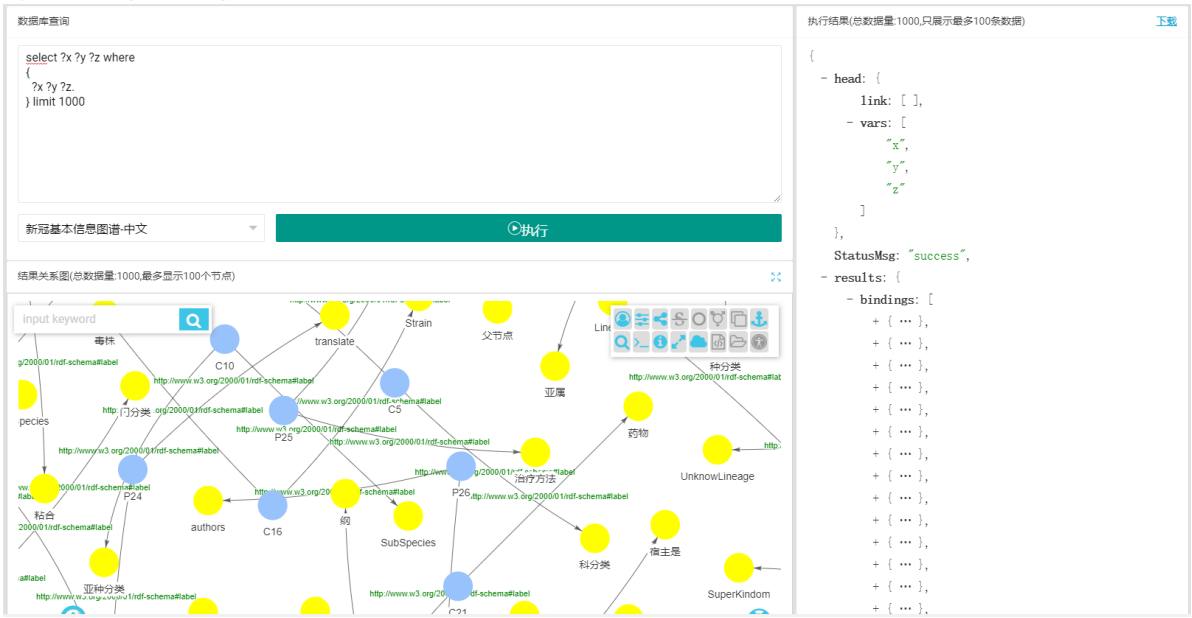
After clicking Export, a zip file will be created, downloaded and decompressed to get the NT file corresponding to the database.



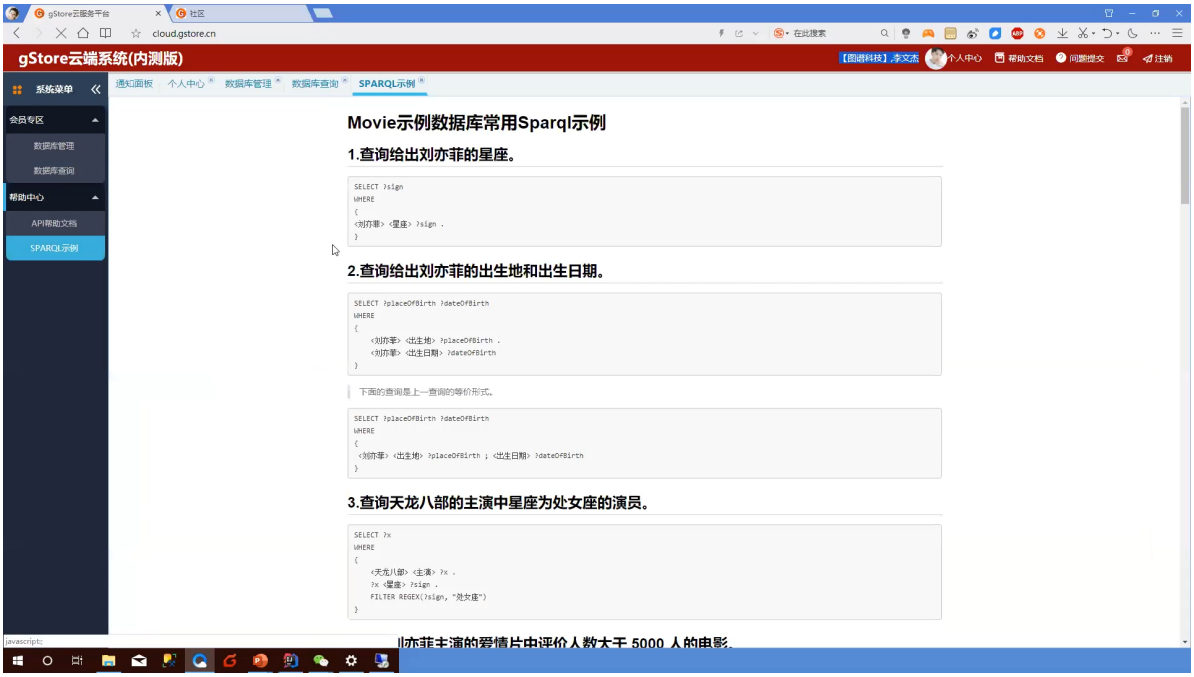
After that, more database functions such as database rename and database backup will also come online.

8.2.5 Database query

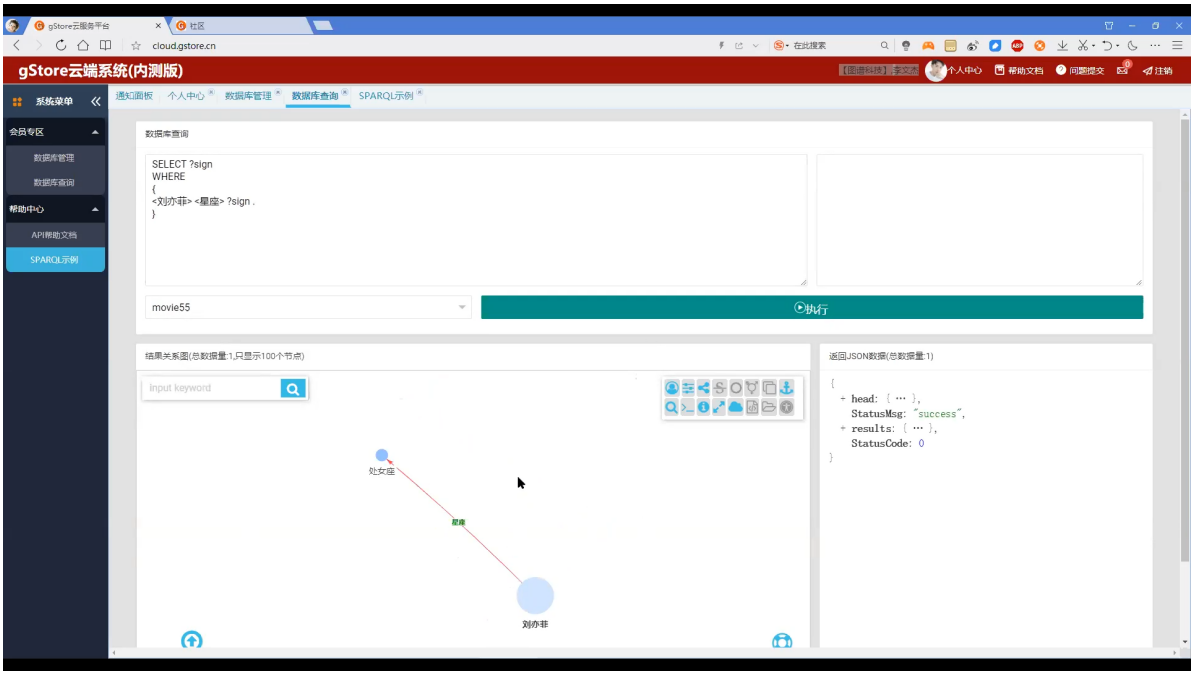
In terms of database query, gStore cloud system provides a visual query interface. Enter Sparql statements in the text box below to get results. (**Note: Considering system performance, the diagram and JSON data only show 100 data, you can click the "Download" button next to the JSON to get all the returned data, and the resulting diagram can be shown in full screen by clicking the right full-screen button**)



Users who are not familiar with Sparql statements can learn about Sparql statements through the Sparql sample function module in the help center area.



Take the first question in the example as an example, query Liu Yifei's constellation, Results are as follows:



You can see the visual graphical result on the left and the literal result of the JSON data on the right.

Sparql statements can also be used to insert or delete data from the database.

8.2.6 Help center

In order to better provide services for users, the platform provides users with a variety of help document information, and will continue to improve and enrich the document information in the future. Currently, the following documents are provided.

8.2.6.1 Platform User Manual

The platform user manual mainly explains the use of gStore cloud platform, so that users can understand the problems related to the use of platform

8.2.6.2 API help document

In addition to directly using gStore cloud platform to manage and query graph data, users can also directly access data by using API. The API help document introduces interface parameters and returned value information in detail.

8.2.6.3 SPARQL sample

Some users may not be familiar with SPARQL, so the platform provides a SAMPLE SPARQL document. The document takes the example database Movie as an example to introduce the main SPARQL statements supported by the platform in detail. Users can directly test the complex SPARQL statements in the help document and in the database query functiom.

8.2.7 API

According to the requirements of some users, we encapsulate the database query operation into AN API interface, through which users can realize remote database access and facilitate users to embed in other systems. KeyID and Secret need to be used in connection. Now the platform has three data interfaces, which are to obtain the current data list, obtain database details, query database. For details, see the API help documentation module in the help center.

8.3 End

This is the end of the help manual for gStore Cloud platform. If you have any questions about the use of the cloud platform, you can click the "Submit questions" in the upper right corner of the cloud platform to give your opinion in the community.

9. gStore Chronology

Year 2021

- In November，gStore0.9.1version released
- In October，gBuilder 2.0 version released
- In February，gStore products have completed the "Graph Database Basic Ability Test" project of China Academy of Information and Communication Technology；
- In February，gStore's new official website launched；

Year 2020

- - In December, gStore added advanced query functions such as shortest/longest path, K-hop reachable query and loop detection to further enrich the gStore algorithm library;

  - In December, gStore Beta (V0.9) and gStore Stable (V0.8) were officially released on Github and Gitee;

  - In November, gBuilder V0.1 version of knowledge graph automation construction platform was launched;

  - In October, gStore distributed version gMaster was demonstrated in related projects of institute of Computing Technology, Chinese Academy of Sciences;

  - In July, gStore was successfully adapted with TONGxin UOS operating system and domestic CPUS of Kunpeng/Haiguang/Zhaoxin/Feiteng

Year 2019

- - In December, PKU graphics database system gStore launched China Science and Technology Cloud 2.0;

  - In November, China Software Evaluation Center conducted a performance test on gStore distributed system, and the test results showed that the average query response time of gStore distributed system was 1.79 seconds under the condition of 10.6 billion data storage scale;

  - In October, pKU Graphics database system gStore cloud platform was deployed and launched;

  - In September, THE GRAPH database system was successfully adapted with the domestic "PK "system (Feiteng CPU+ Kirin operating system);

Year 2018

- Multi-query Optimization in Federated RDF Systems 23rd International Conferenceon Database Systems for Advanced Applications (DASFAA) BEST PAPER AWARD

  - Related theoretical research work of gStore system "Large-scale Graph structure Data Management", won the second prize of Natural Science of Ministry of Education of China (Zou Lei ranked first)

  GAnswer system is officially open source on Github with version NUMBER V0.1

  - gAnswer system took part in the Knowledge base Natural language question contest QALD-9 held by eu and won the first prize

Year 2017

- The PKUMOD research team has released gStore milestone V0.5 on Github.

Year 2016

- PKUMOD research team was funded by the Key research and development project "Key Technology and System of Graph Data Management" of Ministry of Science and Technology of China.

Year 2015

- gStore code is officially open source on Github, version 0.1

Year 2014

- - PKUMOD Graph data management related theoretical research "Massive graph structure data storage and query optimization theory research", won the second prize of Natural Science of China Computer Society (Zou Lei ranked first)

  - The first academic paper related to natural language question answering based on knowledge graph was published

  Lei Zou, Ruizhe Huang, Haixun Wang, Jeffery Xu Yu, Wenqiang He, Dongyan Zhao, Natural Language Question Answering over RDF ---- A Graph Data Driven Approach, SIGMOD 2014

  - PKUMOD research team was supported by the National Natural Science Foundation of China (NSFC) project "Key Technology research on Graph Based Matching Query for Large-scale Heterogeneous Information Network"

Year 2011

- - The first academic paper of gStore was published

  Lei Zou., et al., gStore: Answering SPARQL Queries via Subgraph Matching. PVLDB 4(8): 482-493 (2011)

  - PKUMOD research team was supported by the "Research on Massive RDF Data Storage and query Method based on Graph Database Theory" project of natural Science Foundation of China (NSFC)

10. Open source and legal provision

10.1 Open Source and Community

gStore system from January 2015 open source in the making, comply with the BSD, 3 - Clause open source licenses https://github.com/pkumod/gStore is open source address; We advocate users to use and modify gStore freely on the premise of respecting the copyright of code authors, develop various industry applications of knowledge graph based on gStore, and promote the healthy and sustainable development of knowledge graph industry software. We encourage users to actively use the gStore system, report problems, make suggestions, and contribute code to the gStore open Source project to join us and make the gStore system better.

If you have any questions in the process of using gStore, if you would like to tell us your name, organization, purpose of using gStore and email address, we will timely reply to you by sending an email to service@gstore.cn. We guarantee that the privacy of you and your company will not be disclosed, only used to enhance the gStore system itself

10.2 Legal Issues

The gStore system always adopts the BSD 3-clause which is widely used in the open source community. Under this Agreement, User is free to modify and redistribute the Code, subject to the following terms, and user is free to develop, distribute and sell commercial software based on the gStore code. The specific terms are as follows:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Peking University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

中文条款声明如下（具有同等法律效应）：

11. gStore Logo

11.1 gStore's Logo is as follow:

11.2 Powered by gStore recommend logo is as follow