Article
Dmitry Maslennikov · Oct 3, 2016 6m read

Internal Structure of Caché Database Blocks, Part 1

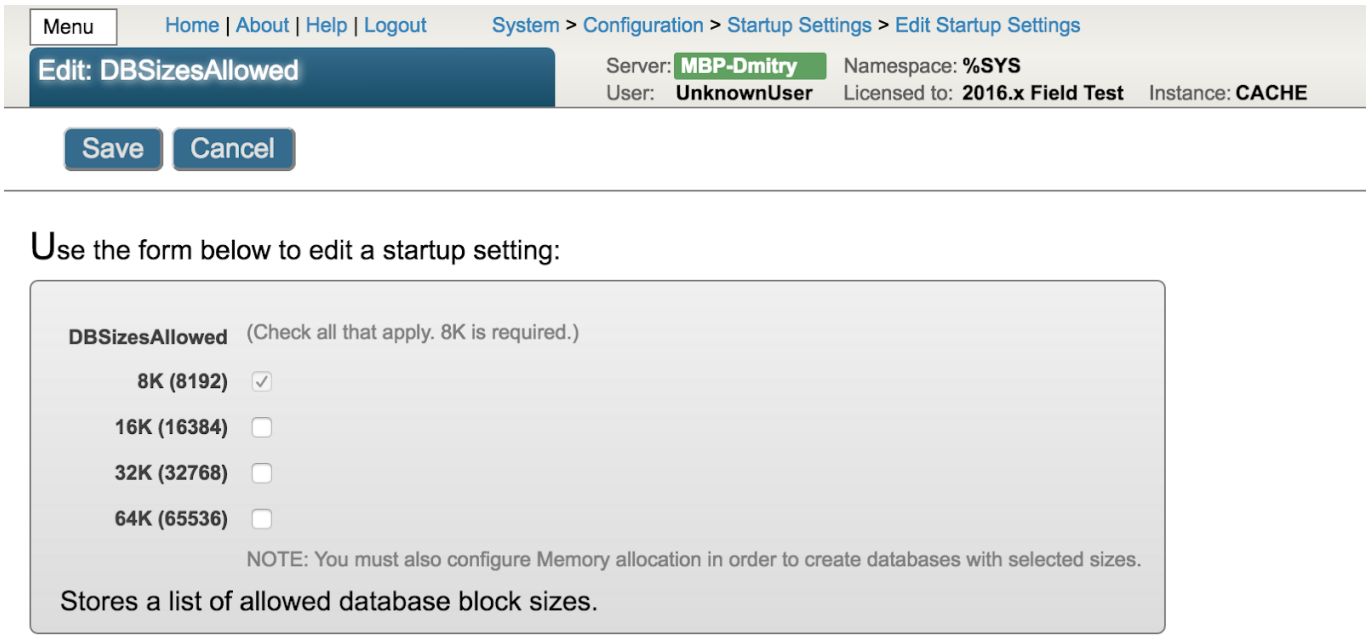
InterSystems Caché globals provide very convenient features for developers. But why are globals so fast and efficient?

Theory

Basically, the Caché database is a catalog having the same name as the database and containing the CACHE.DAT file. On Unix systems, the database can also be an ordinary disk partition.

All data in Caché is stored in blocks which, in turn, are organized as a balanced B* tree. Taking into account that all globals are basically stored in a tree, global's subscripts will be represented as branches, while values of global's subscripts will be stored as leaves. The difference between a balanced B* tree and an ordinary B tree is that its branches also have right links that can help to iterate through subscripts (i.e. globals in our case) rapidly using the $Order and $Query functions without getting back to the trunk of the tree.

By default, each block in the database file has the fixed size of 8,192 bytes. You can not modify size of block for already existed database. While you creating new database you can choose 16kB, 32 kB or even 64 kB blocks – depending on the data type you are going to store. However, always keep in mind that all data is read block-by-block – in other words, even if you request a single value of 1 byte in size, the system will read several blocks among which the requested data block will be the last one. You should also remember that Caché uses global buffers to store database blocks in memory for second use, and as well as size of blocks buffers has the same sizes. You cannot mount an existing database or create a new one when a global buffer with the corresponding block size is missing in the system. You should define size of memory which you want to be allocated for concrete size of blocks. It is possible to use buffers blocks bigger than database blocks, but in this case each buffer block will store only one database block, even smaller.



In this picture, a memory for the global buffer of 8 kB in size is allocated for the use with databases consisting of 8 kB blocks. Blocks which are not empty in this database defined in maps, so that one of the maps defines 62,464 blocks (for 8kb blocks).

Block types

The system supports multiple block types. At each level, right links of a block must point to a block of the same type or to a null block which defines the end of data.

- **Type 9**: Block of a global catalog. These blocks usually describe all existing globals along with their parameters, including collation of global's subscripts which is one of the most important parameters that cannot be modified once the global is created.
- **Type 66**: Block of high-level pointers. Only a block of a global catalog can be on top of these blocks.
- **Type 6**: Block of low-level pointers. Only blocks of high-level pointers can be on top of these blocks and only data blocks can be placed lower.
- **Type 70**: Block of both high-level and low-level pointers. These blocks are used when the corresponding global stores a small number of values and therefore multiple levels of blocks are not required. These blocks usually point to data blocks, just as blocks of a global catalog do.
- **Type 2**: Block of pointers for storing a relatively large global. In order to allocate values among data blocks evenly, you may want to create additional levels of blocks of pointers. These blocks are usually placed between the blocks of pointers.
- **Type 8**: Data block. These blocks usually store values of several global nodes rather than of only one node.
- **Type 24**: Block for large strings. When a value of a single global is bigger than a block, then such value will be recorded into a special block for large strings, while the data block node will be storing links to the list of blocks for large strings along with the total length of this value.
- **Type 16**: Map block. These blocks are designed for storing information about unallocated blocks.

So, the first block of a typical Caché database contains service information about the database file itself, while the second blocks provides a map of blocks. The first catalog block goes on the third place (block #3), and a single database can have several catalog blocks. The next ones are blocks of pointers (branches), data blocks (leafs), and blocks of large strings. As I have mentioned above, blocks of global catalogs store information about all existing globals in the database or global settings (if not data is available in such a global). In this case, a node that describes such a global will have a null lower pointer. You can view the list of existing globals from the global catalog on the management portal. This portal also allows you to save a global in the catalog after deletion (e.g. save its collating sequence) as well as create a new global with either default or custom collate.

Menu Home | About | Help | Logout System > Configuration > Local Databases > Globals

Globals

Server: MBP-Dmitry Namespace: %SY
User: UnknownUser Licensed to: 201

Create New Global

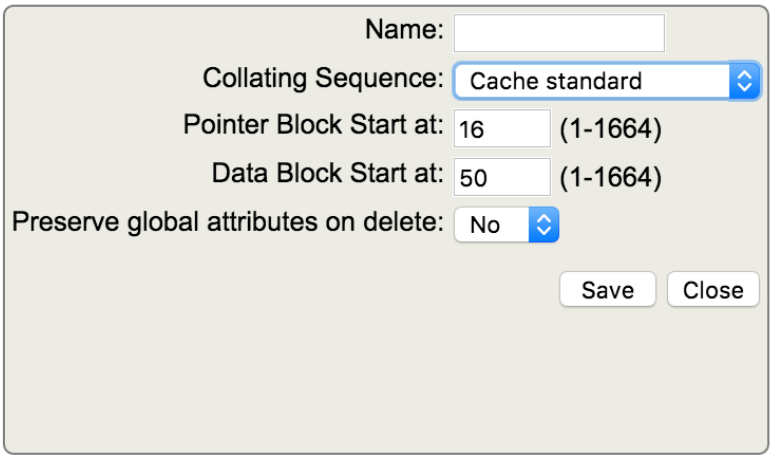
Globals in database /opt/cache/mgr/samples/:

Globals: * Go System Filter: Page size: 20 Items

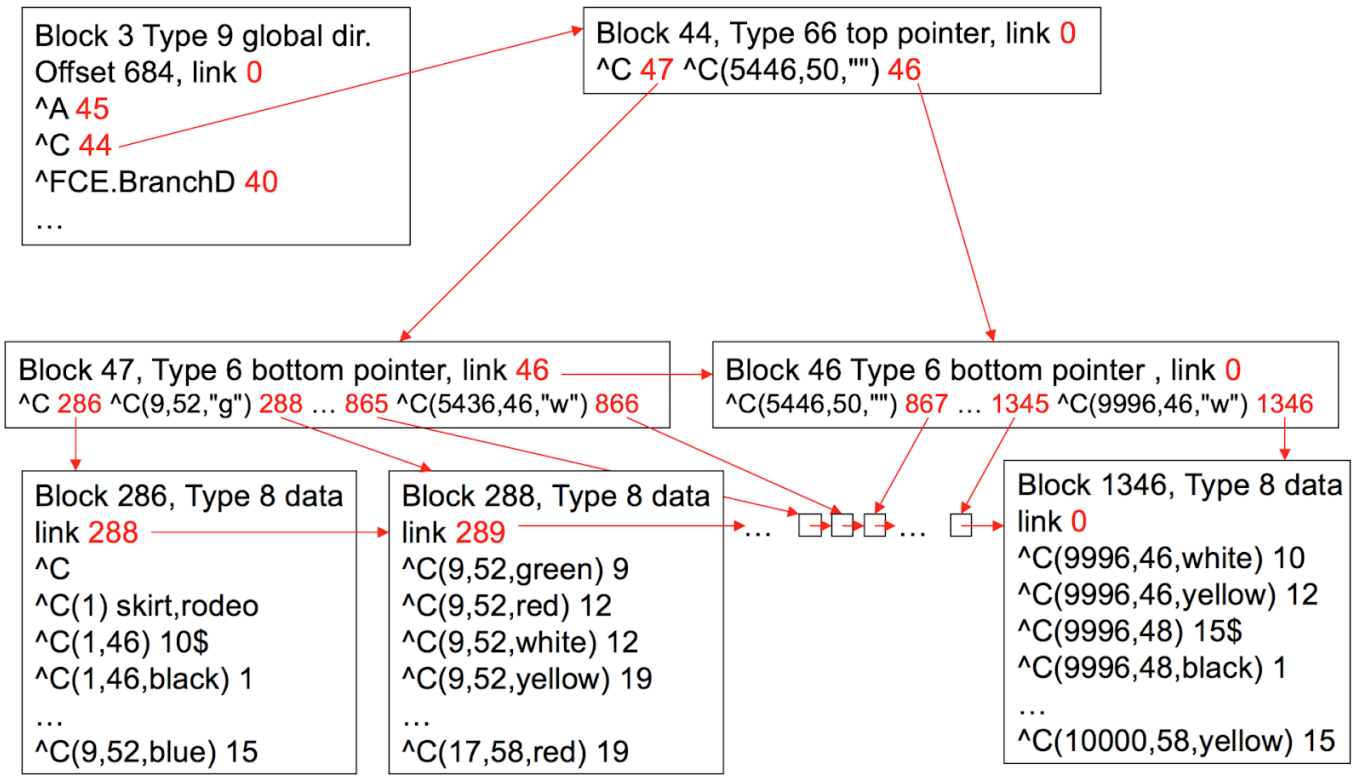
Name	Permission	Empty	Keep			
Aviation.AircraftD	RW	No	No	View	-	Properties
Aviation.AircraftI	RW	No	No	View	-	Properties
Aviation.Countries	RW	No	No	View	-	Properties
Aviation.CrewI	RW	No	No	View	-	Properties
Aviation.EventD	RW	No	No	View	-	Properties
Aviation.EventI	RW	No	No	View	-	Properties
Aviation.States	RW	No	No	View	-	Properties
CacheMsg	RW	No	No	View	-	Properties
Cinema.ReviewD	RW	No	No	View	-	Properties
CinemaooFilmCategoryD	RW	No	No	View	-	Properties
CinemaooFilmCategoryI	RW	No	No	View	-	Properties
CinemaooFilmD	RW	No	No	View	-	Properties
CinemaooFilmI	RW	No	No	View	-	Properties
CinemaooShowD	RW	No	No	View	-	Properties
CinemaooTheaterD	RW	No	No	View	-	Properties
DeepSee.ActiveTasks	RW	Yes	No	View	Delete	Properties
DeepSee.AgentLog	RW	No	No	View	-	Properties
DeepSee.Agents	RW	No	No	View	-	Properties
DeepSee.BucketList	RW	No	No	View	-	Properties
DeepSee.Cache.Results	RW	No	No	View	-	Properties

[Next page]

Create a new global in /opt/cache/mgr/samples/:

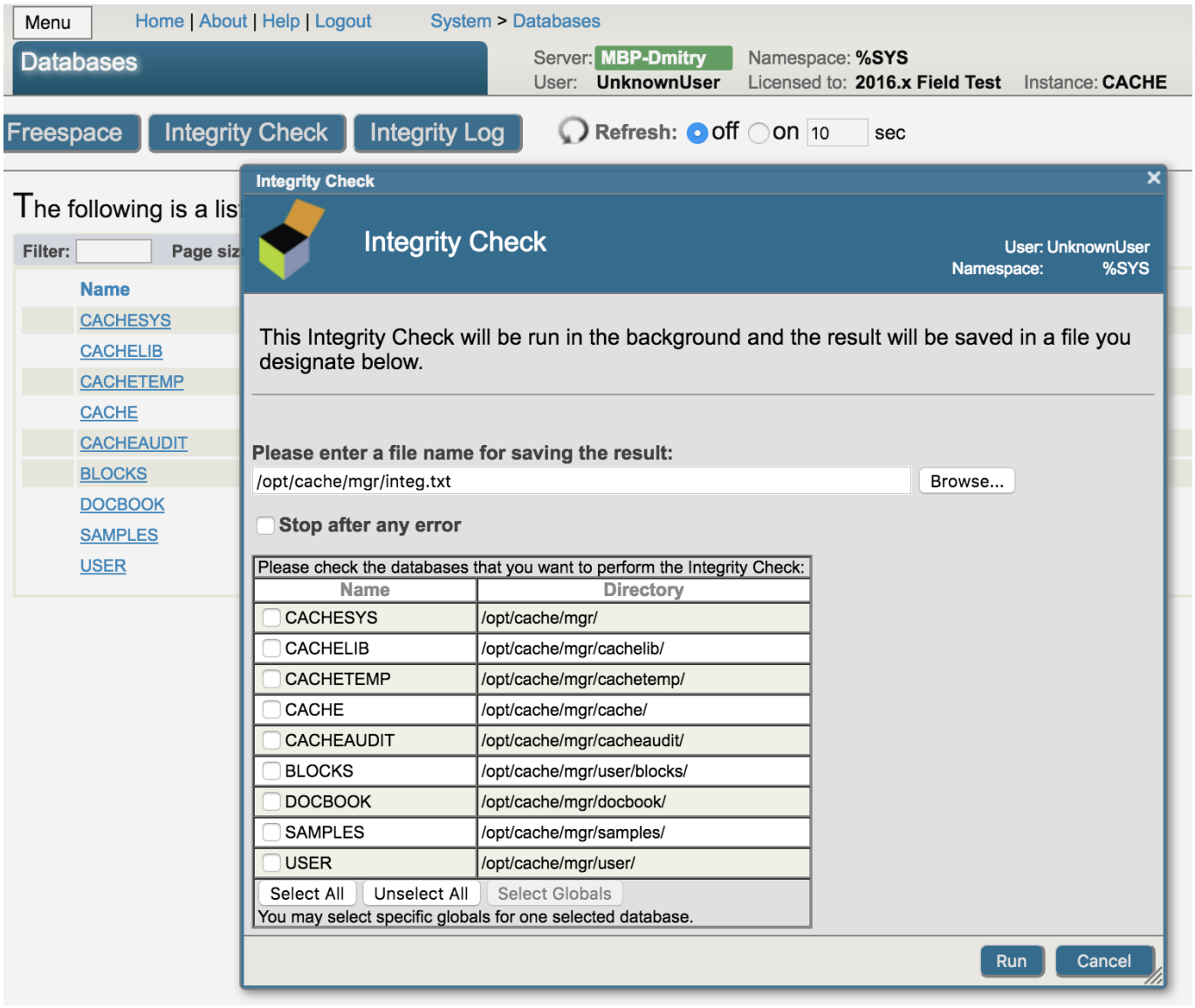


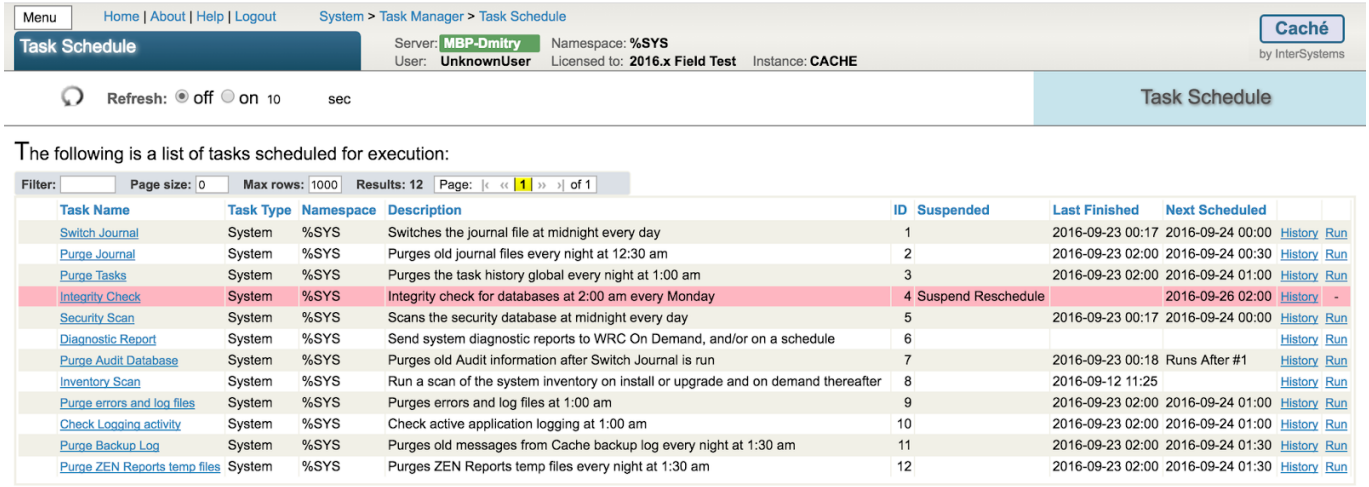
In general, the tree of blocks can be represented as in the picture below. Note that links to blocks are shown in red color.



Database integrity

In the current version of Caché, we have resolved the most important issues and problems with databases, so that the risks of database degradation are extremely low. However, we still recommend that you run automatic integrity checks regularly using our ^Integrity tool – you can launch it in the terminal from the %SYS namespace, via our management portal, on the Database page or via the task manager. By default, the automatic integrity check is already set up and predefined, so that the only thing you need to do is to activate it:





The integrity check includes verification of links at the lower levels, validation of block types, analysis of the right links and matching of global nodes with the applied collating sequence. If any errors are found during the integrity check, you can run our ^REPAIR tool from the %SYS namespace. Using this tool, you can view any block and modify it as required, i.e. repair your database.

Practice

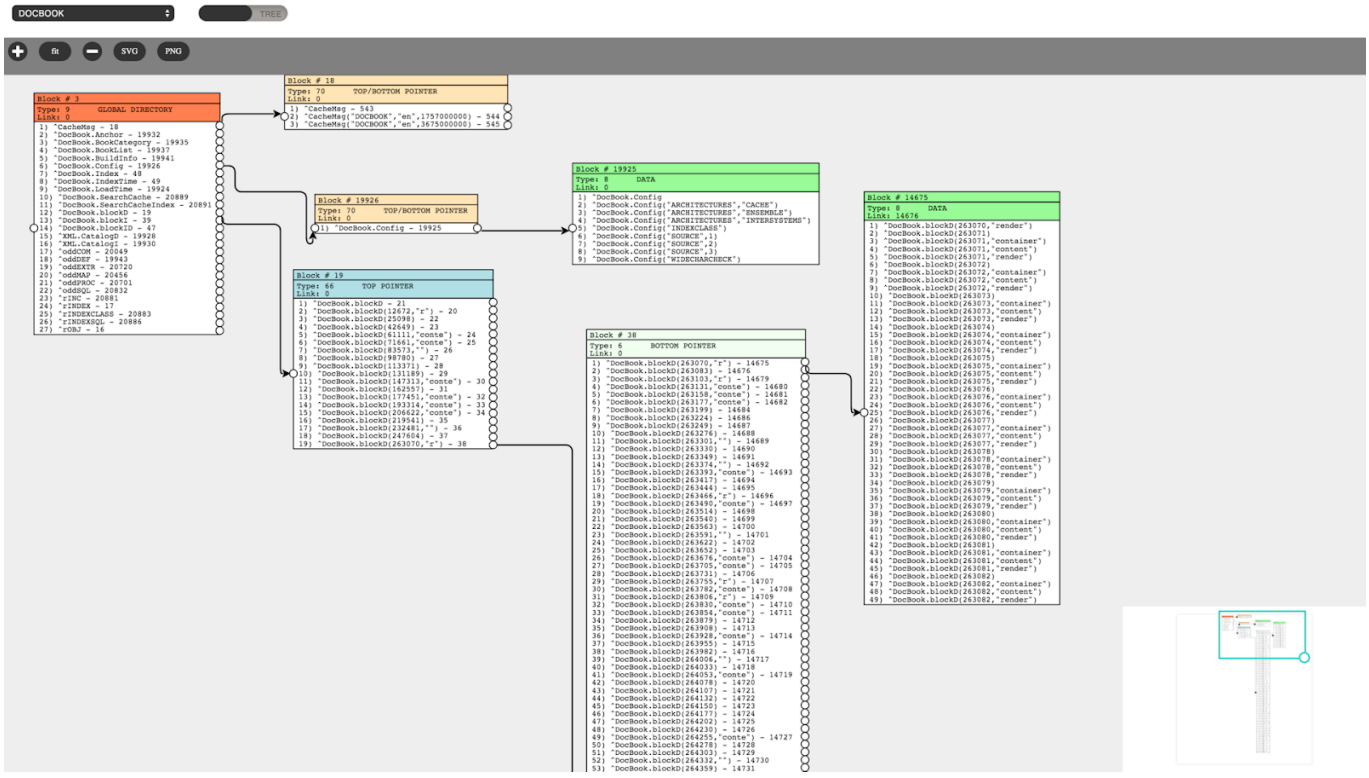
However, this was only theory. It is still difficult to figure out what do a global and its blocks actually look like. Presently, the only way to view blocks is to use our ^REPAIR tool mentioned above. The typical output of this program is shown below:

```
Map Blocks: 2
Entering Block Repair Menu
Block #: 3
Block # 3                    Type: 9 GLOBAL DIR
Link Block: 0                Offset: 2516
Count of Nodes: 111          Big String Nodes: 0
Pointer Length:0             Next Pointer Length:0   Diff Byte:Hex 0
Pointer Reference:          ^Aviation.AircraftD
Next Pointer Reference:
Next pointer stored? No


--more--

#     Node                    PTR,NEW GROWTH,COLL,TYPE,PROT
1     ^Aviation.AircraftD     8804     50          5     0     195
2     ^Aviation.AircraftI     8869     50          5     0     195
3     ^Aviation.Countries     8874     50          5     0     195
4     ^Aviation.CrewI         8876     50          5     0     195
5     ^Aviation.EventD        8882     50          5     0     195
6     ^Aviation.EventI        10036     50          5     0     195
7     ^Aviation.States        10040     50          5     0     195
8     ^CacheMsg               49     50          5     0     195
9     ^Cinema.ReviewD         10136     50          5     0     195
```

A year ago, I have started a new project to develop a tool that iterates through a tree of blocks without any risks of database damage, visualizes these blocks in a web UI and provides options for saving their visualization in SVG or PNG. The project is called CacheBlocksExplorer, and you can download its source code on Github.



The implemented features include:

- View any configured or simply mounted database in the system;

- View information by block, block type, right pointer, list of nodes with links;
- View detailed information about any node pointing to a lower block;
- Hide blocks by removing links to them (without any damage to the data stored in these blocks).

The to-do list:

- Displaying right links: in the current version, right links are shown in the information about blocks, but it would be better to display them as arrows;

- Display blocks of large strings: they are simply not shown in the current version;

- Display all blocks of global catalog rather than only the third one.

I would also want to display the entire tree, but still cannot find any library able to rapidly render hundreds of thousands blocks along with their links – the current library renders them in web browsers much slower than Caché reads the whole structure.

In the next article, I will try to describe in more detail how it works, provide some usage examples and demonstrate how to retrieve lots of actionable data about globals and blocks using my Cache Block Explorer.

#Databases #System Administration #Caché