

# **GET DATABASE MEASURES**

By Roland Lannuzel, 4D Program Team Member, 4D S.A.S.

Technical Note 15-02

## Table of Contents

---

Table of Contents .....	2
Abstract .....	3
Introduction .....	3
About GetMeasures Component .....	3
Global view .....	4
Display: Bytes / Count .....	5
History Length: 15/30/60/120/180 .....	5
Proportional .....	5
Pause .....	5
Log every / Start Logging / Show Log Folder .....	5
Global view - Disk operations .....	6
Tables .....	9
Indexes .....	11
Queries and Sorts .....	13
Test .....	14
"Log" Content .....	14
About MeasuresReport Application .....	15
Projects .....	15
Disk Access .....	16
Tables .....	17
Indexes .....	19
An interesting case .....	20
Queries & Sorts (Counts) .....	21
Counter-Example! .....	23
Queries & Sorts (Statements) .....	24
Bottom page warning .....	25
Disk Timeline .....	26
Peak value .....	27
Display Delta option .....	27
Display Segment info .....	27
Proportional .....	27
Table Timeline & Index Timeline .....	28
Conclusion .....	29
Bonus .....	30

## Abstract

---

4Dv14 R3 introduces a new command "Get database measures" that allows to know almost everything about what's happening inside the 4D engine, the memory, the disks exchanges, the queries and the sorts. This command is quite simple to call but the returned results might be a bit complex to understand. This Technical Note will explain these results into details and will also provide you with graphic tools to make these results even more explicit.

## Introduction

---

This Technical Note will describe two features. The first one is a 4D component called "GetMeasures" based on the command "Get Database Measures". It will graphically display all the information that is available "in live" during the exploitation of the database. Beside that, this component allows to generate a log with minimum performance impact in a production system, to be analyzed by the second feature later/offline.

The second feature is a database called "MeasuresReport". This database will import the logs generated by the component and give more specific information about the activity of the main database. All information will be either displayed graphically (like the component does but more precisely) or inside a list box, with numbers in order to have more precise information.

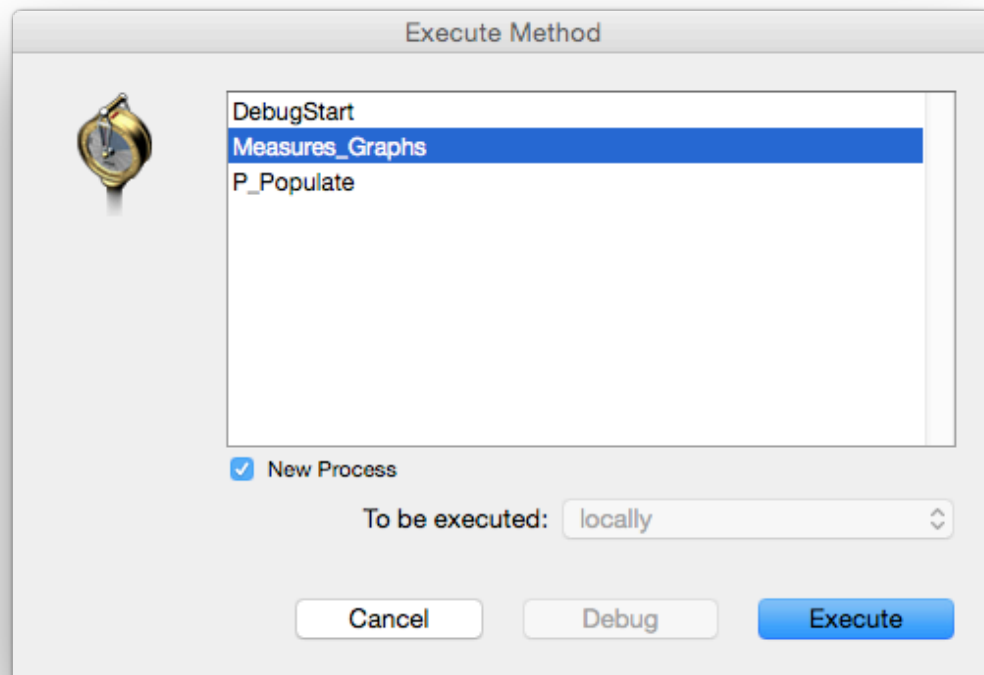
This tool will also help to point out some potential problems or possible optimizations.

## About GetMeasures Component

---

The component has been written and compiled with a 4D v14 R5. It takes advantage of all the information that is returned by the command. This component exposes only one method called "Measures\_Graphs" that will be called by the host database.

The method will create a separated process, so it can be called directly from a method of the host database or launched directly, as in the picture below (the checkbox "New process" does NOT need to be checked).



## Global view

Once the Measures\_Graphs method has been launched, a window is opened and displays a tab-control, a couple of buttons and some graphic zones.

As long as most buttons and controls are common to almost every page, let's explain their function right now.

### **Display: Bytes / Count**

This option allows you to choose to display either "bytes" or "counts" for each graphic zone (read or write, data or indexes, etc). The unit for bytes is automatically adjusted (B, KB, MB, GB). The "counts" value represents the number of hits for each zone.

### **History Length: 15/30/60/120/180**

This is the number of seconds that will be displayed inside the graphs. The maximum displayed history is 3 minutes (180 seconds).

### **Proportional**

This checkbox determines how each graph should be displayed, either as a single graph (the max value in the graph belongs to the graph itself) or based on max values of all the displayed graphs.

### **Pause**

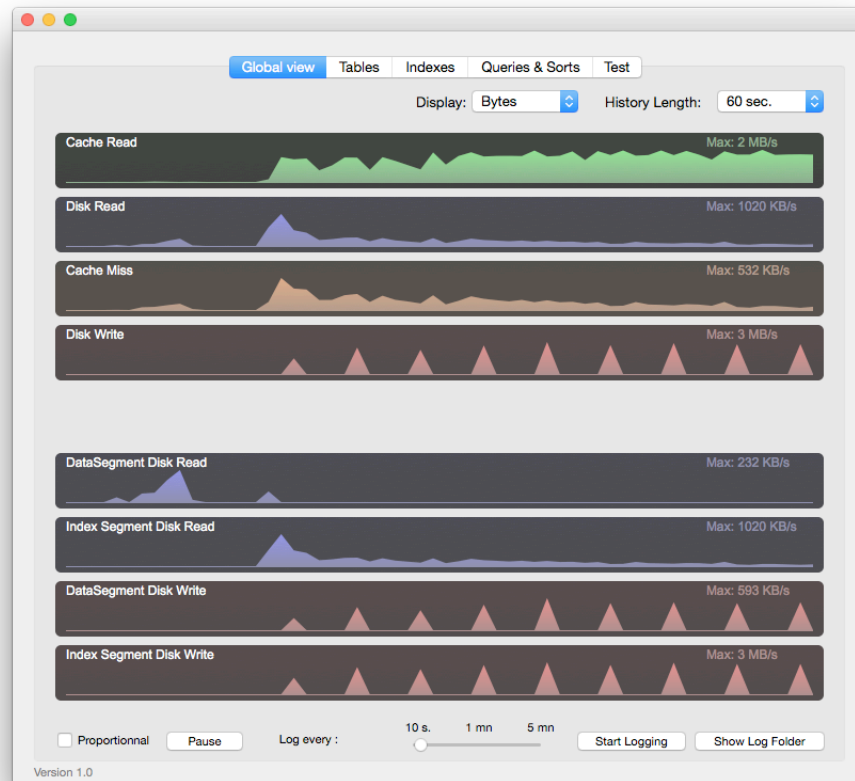
This button allows to pause/resume the graphic updates. When running, graphics are updated every second. This button will NOT pause the "log" (see below).

### **Log every / Start Logging / Show Log Folder**

These three settings set up the "log" behavior. The content of the log file is very simple, it's basically a timestamp followed by the content of a "stringified" version of a full measure (but without history) of the database.

The logs will be written inside a folder named "MeasuresLogs" next to the host structure (the folder will be created if it does not exist).

## Global view - Disk operations

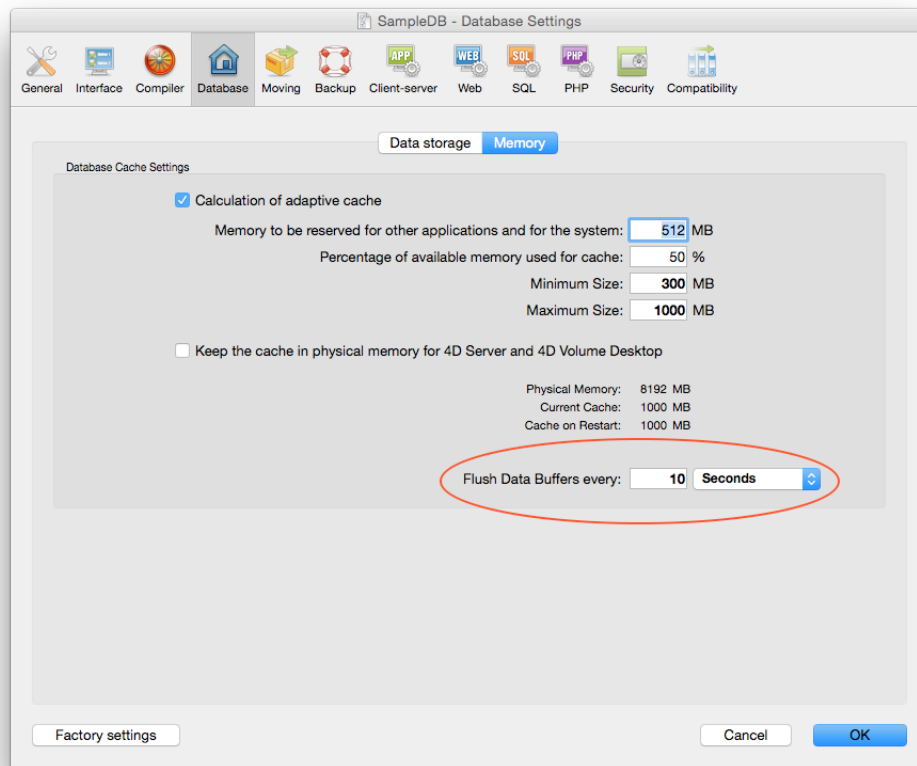


According to the display choice (Bytes or Count) the 4 first graphs (on the top) will display the result of the following paths :

DB.cacheReadBytes DB.diskReadBytes DB.cacheMissBytes DB.diskWriteBytes	DB.diskReadCount DB.cacheReadCount DB.cacheMissCount DB.diskWriteCount
---	---

- "cacheMiss" are the bytes that could not be read in the cache. This happens when the database has just been launched for example, and the cache has not been filled yet or the cache is not large enough to store all data, the requested data was not contained in the cache.
- "diskRead" usually happens just after a "cacheMiss". As the required data was not in the cache, 4D will read it from disk, so we have a diskRead.
- "cacheRead" represents the amount of data read from the cache (without the need to access the hard disk)
- "diskWrites" represents the bytes written on the disk.

*Note: Don't be surprised if "disk write" action does not happen after each "save record" for instance. The disk write actions are performed every "x" seconds according to the database settings.*



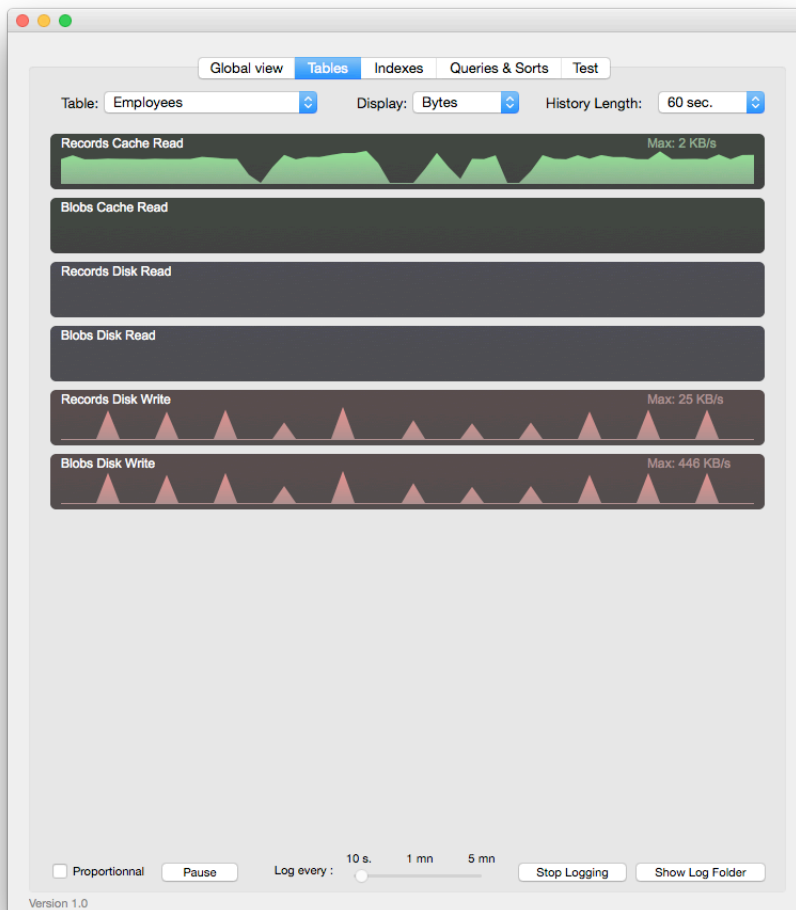
The four next graphics (on the bottom) represent the disk access, in read/write mode for each segment ("data" segment AKA as ".4dd" and "index" segment AKA as "4dindx"). Once again either in "Bytes" or in "Counts".

DB.dataSegment1.diskReadBytes DB.indexSegment.diskReadBytes DB.dataSegment1.diskWriteBytes DB.indexSegment.diskWriteBytes	DB.dataSegment1.diskReadCount DB.indexSegment.diskReadCount DB.dataSegment1.diskWriteCount DB.indexSegment.diskWriteCount
--	--

## Tables

The "Tables" panel gives you the same kind of information as the first ones of the "Global View" panel but with more details!

1. You can choose a specific table within a popup menu.
2. Each action (cacheRead, diskRead, diskWrite) are split into two parts: "Records" and "Blobs". The "records" (bytes or counts) concern *regular* fields like integers, longintegers, reals, dates, hours, alpha... The "Blobs" concern large type of fields (Text, Images... and Blobs) except if they have been specifically defined to be stored inside the records which is not the default option for those type of fields (see the structure definition).



The graphics matches the following paths, where *tableName* is the name of the selected table in the dropdown list:

DB.tables. <i>tableName</i> .records.cacheReadBytes	DB.tables. <i>tableName</i> .records.cacheReadCount
DB.tables. <i>tableName</i> .blobs.cacheReadBytes	DB.tables. <i>tableName</i> .blobs.cacheReadCount
DB.tables. <i>tableName</i> .records.diskReadBytes	DB.tables. <i>tableName</i> .records.diskReadCount
DB.tables. <i>tableName</i> .blobs.diskReadBytes	DB.tables. <i>tableName</i> .blobs.diskReadCount
DB.tables. <i>tableName</i> .records.diskWriteBytes	DB.tables. <i>tableName</i> .records.diskWriteCount
DB.tables. <i>tableName</i> .blobs.diskWriteBytes	DB.tables. <i>tableName</i> .blobs.diskWriteCount

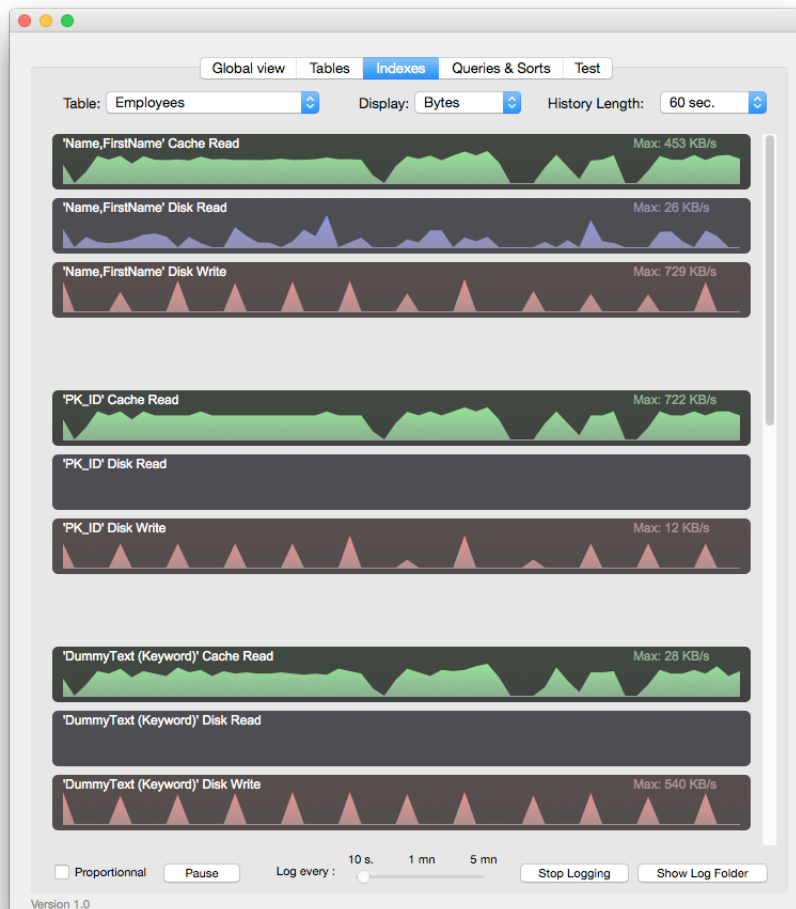
## Indexes

The "Indexes" panel gives you information about... indexes! To be more specific, it gives you information about indexes that have been used since the moment the database was launched. Again, a table must be selected using the dropdown list as well as "bytes" or "counts".

Each index has three graphics that match the following paths:

DB.tables. <i>tableName</i> . <i>FieldName</i> .cacheReadBytes
DB.tables. <i>tableName</i> . <i>FieldName</i> .cacheReadBytes
DB.tables. <i>tableName</i> . <i>FieldName</i> .diskReadBytes

DB.tables. <i>tableName</i> . <i>FieldName</i> .cacheReadCount
DB.tables. <i>tableName</i> . <i>FieldName</i> .cacheReadCount
DB.indexes. <i>tableName</i> . <i>FieldName</i> .diskReadCount



*Notes: When an index is based on many fields (composite index) then the field names are represented separated with a comma. Ex: "Name,FirstName" but this a single index!*

*When an index is based on keywords then the name of the field is suffixed with "(Keywords)". Just remind that a text field can have two indexes at once: a standard index (btree or cluster) AND keywords.*

*The "cacheMissBytes" and "cacheMissCount" are not represented on theses graphics.*

## Queries and Sorts

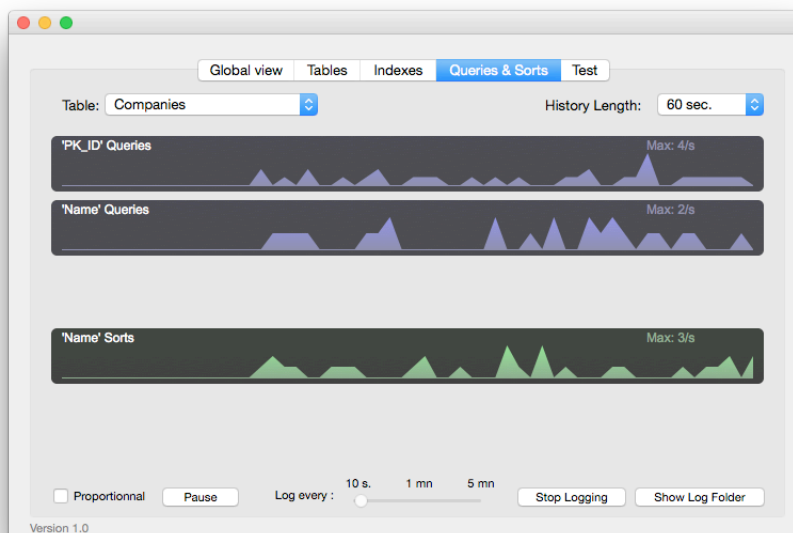
This requires the usage of 4D v14 R5. The 4D v14 R3 and R4 versions do not contain information about searches or sorts.

This panels displays, also in real time, all the queries and sorts that are performed on the selected table. These graphics are based on the paths described below.

DB.tables.*tableName*.fields.*FieldName*.sortCount  
DB.tables.*tableName*.fields.*FieldName*.queryCount

The *tableName* is selected in the dropdown list and all the fields.*FieldName* are taken into account.

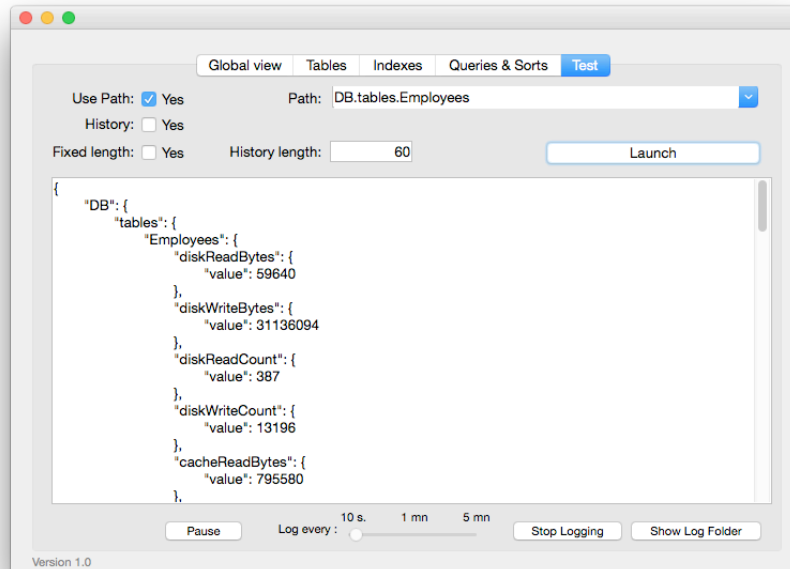
*Important note: ALL the queries and sorts are displayed, not only the ones based on the indexes.*



## Test

This panel has been used mainly for the development of this component. As long as you may find it useful, we've decided to let it there :o)

It will also help you to understand the way the command "Get Database Measures" works, how you can setup the parameters and how does the returned object look like.



## "Log" Content

As we said at the beginning, this component logs the result of the command "Get Database Measures".

The content of the log is as simple as possible. It contains a time stamp (####20141217\_14:57:31####) followed by the content, stringified, of the object returned by the command ({"DB":{" etc.)

```
####20141217_14:57:31####
{"DB":{"diskReadBytes":{"value":631273628},"diskWriteBytes":{"value":552312577},
"diskReadCount":{"value":95317},"diskWriteCount":{"value":884
(...)
}}}}}}
####20141217_14:57:41####
{"DB":{"diskReadBytes":
(...)
}
```

These logs will be used by the application "MeasuresReport" that we will describe right now!

## About MeasuresReport Application

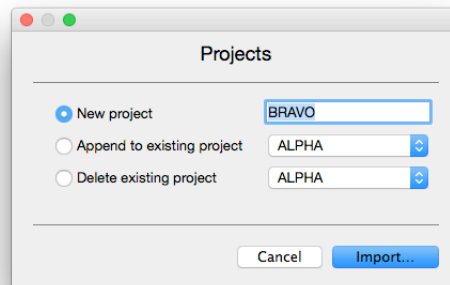
---

This application has been designed to analyze the log content produced by the "GetMeasures" component.

### Projects

In order to import a log, you have to create a new project. You may also select an existing project if you want to append the log to another one, already imported.

This is a very simple operation. Just give a name to your project then click "Import" and select the log file to be imported.



Once imported, the content of any project can be displayed graphically. The first four tabs are based on the last item of the log and the three last ones are based on all the measures of the period.

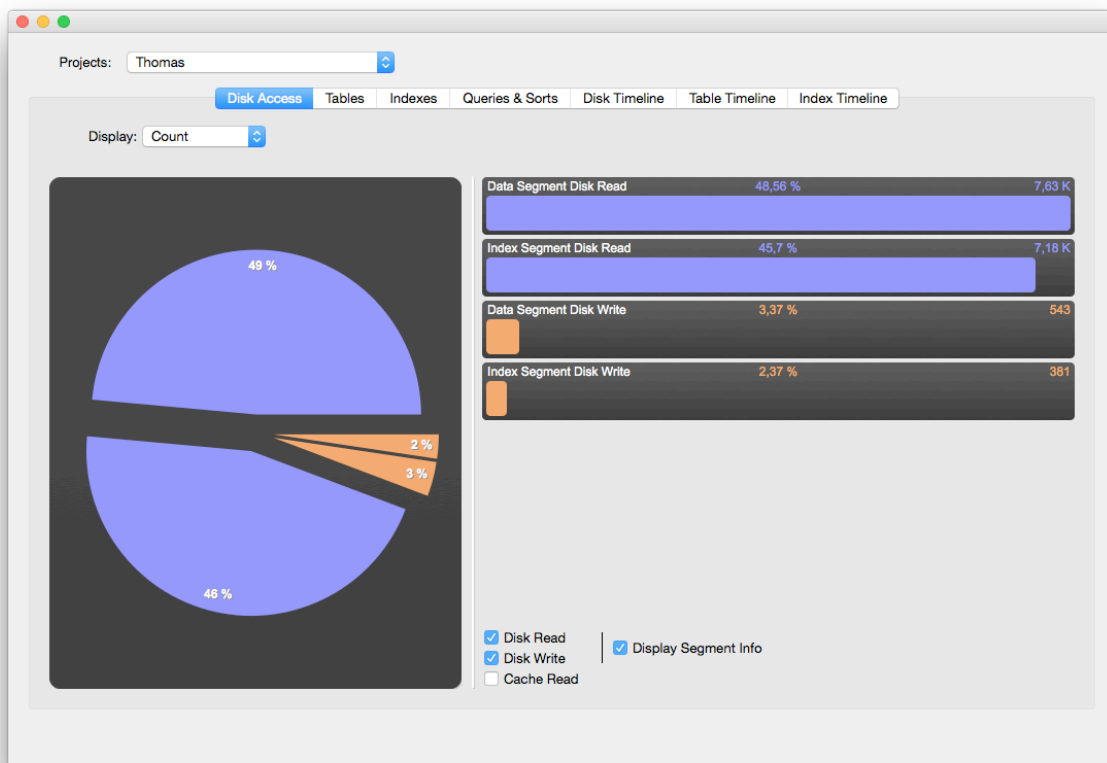
*The last item of the log contains the most recent measure written during the log period. What does that mean exactly? The command **Get Database Measures** returns values cumulated since the launch of the database. So this last item is sufficient to analyze and display most graphics ("Disk Access", "Tables", "Indexes", "Queries and Sorts"). Logging for a long period is useful only if you need to know more about the "life" of each table or index during the period; Otherwise, a single "snapshot" at the end of the day should be enough for most needed information.*

## Disk Access

This very first tab displays the disk access in bytes or in hits (counts).

The access values are in read mode and write mode and can be split in two parts: Data segment and Index Segment.

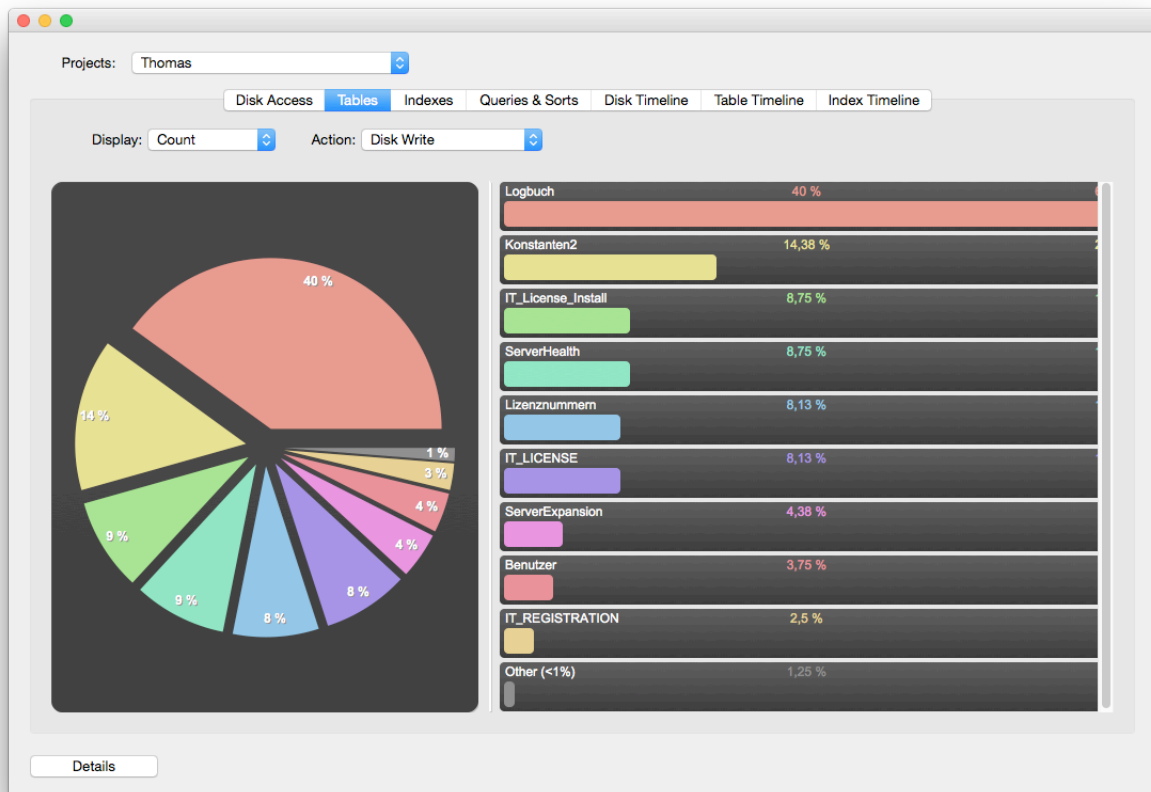
The "cache read" can also be displayed but this usually makes the other pies very small and quite unreadable (which makes sense and is a good thing: It proves that the cache is used as much as possible!)



## Tables

The "tables" panel displays as many pies as tables whose activity is greater or equal to 1% of the global selected activity.

Example: If the *activity* is "Disk Write Count" then all the tables with more than 1% of the global "Disk Write Count" will be displayed. All the other tables will be grouped inside a grey pie named "others" (whose size might be greater than 1% of course!)



If more details are needed then the button "Details" should be used. Once clicked, all the collected numerical values are displayed at once.

Table Names	Disk Read Bytes	Disk Write Bytes	Cache Read Bytes	Cache Miss Bytes	Disk Read Count	Disk Write Count	Cache Read Count	Cache Miss Count
Logbuch	0	67 236	0	0	0	64	0	0
Konstanten2	3 270	3 350	562 166	3 270	23	23	4 364	23
IT_License_Install	6 497	2 030	17 485	6 497	45	14	121	45
ServerHealth	1 258	3 278	226	1 258	6	14	1	6
Lizenznummern	77 730	7 216	81 254	77 730	133	13	141	133
IT_LICENSE	40 939	11 441	112 614	40 939	51	13	140	51
ServerExpansion	61 710	2 050	14 436	61 650	152	7	43	152
Benutzer	132 386	2 288	1 731 381	132 350	57	6	390	57
IT_REGISTRATION	3 147	492	6 824	3 147	25	4	54	25
IT_SEED_MACHINE	2 372 448	188	18 380	2 372 448	12 600	1	96	12 600
IT_PERSO...ACCOUNT	7 600	550	42 550	7 600	14	1	76	14
Auftrage	18 911 768	0	2 842 153	18 911 768	27 616	0	4 021	27 616
Ansprechpartner	254 602	0	208 050	254 602	845	0	674	845
Kunden	1 315 026	0	5 860 912	1 315 026	785	0	3 321	785
Telefon	97 657	0	112 720	97 657	447	0	496	447
Follow_Chat	722 076	0	167 670 444	721 808	134	0	43 954	134
	23 MB	98 KB	191 MB	23 MB	42 K	160	58 K	42 K

Graphs

*Note: It's quite important to understand that when you click on a header, it's not a single column sort that is performed... It's a complete "refill" based on the selected value.*

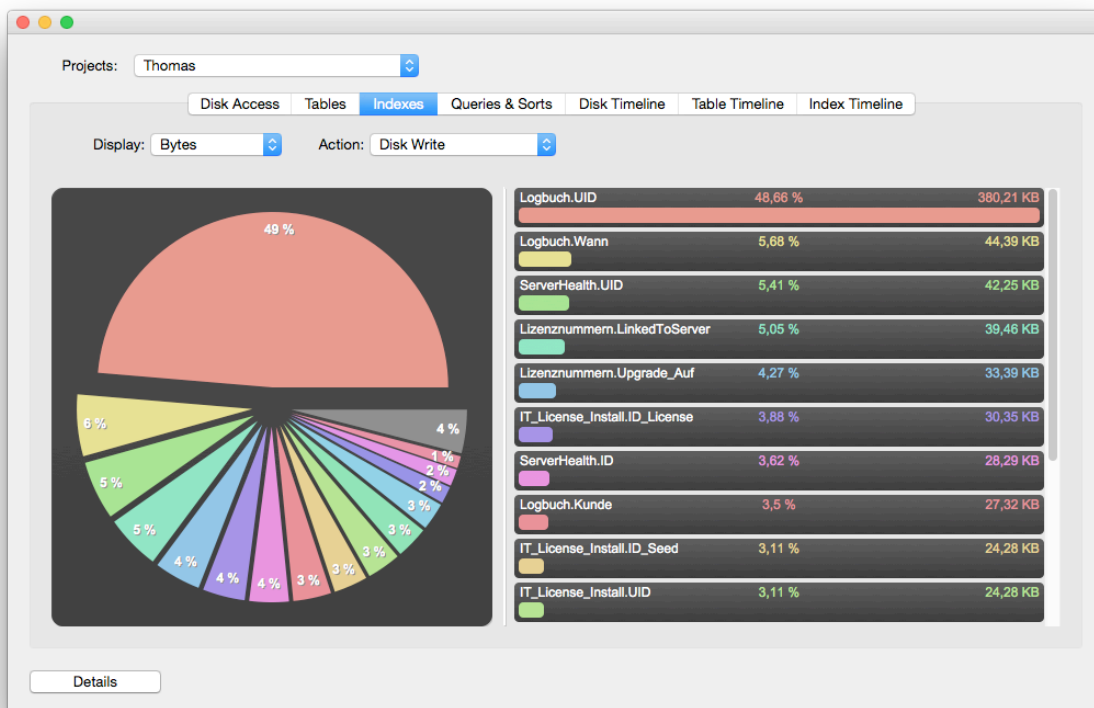
Example: clicking on the "Cache Read Count" header will have the same result as selecting "Display: Count" and "Action: Cache Read" on the previous page. The graphs will be recalculated and the arrays filled with new values.

## Indexes

The "Indexes" panel displays the same kind of information as "Tables" panel. There are as many pies as indexes whose activity is greater or equal to 1% of the global selected activity.

Example: If the *activity* is "Disk Write Bytes" then all the indexes with more than 1% of the global "Disk Write Bytes" will be displayed.

Other indexes whose activity is lower than 1% (if any) are grouped inside the grey pie. Note that the sum of values lower than one percent can be larger than one percent (In the picture below, it represents 4%.)



As with tables, you can also display numerical details. The same principle worth for indexes: when a header is clicked, a new search is performed and the columns are recalculated.

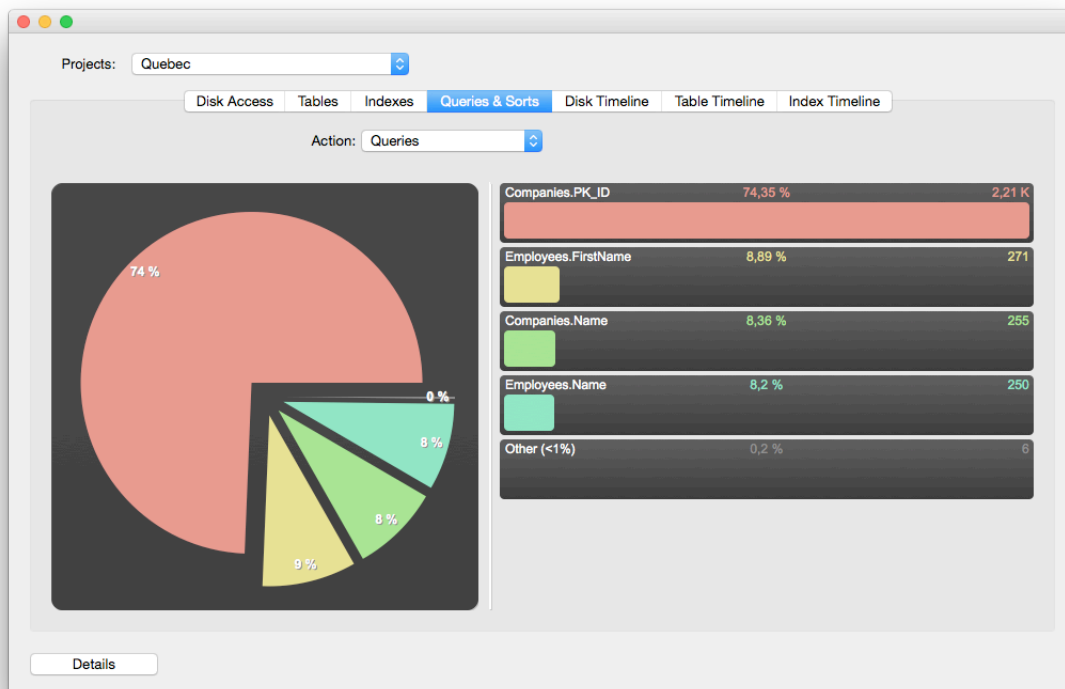


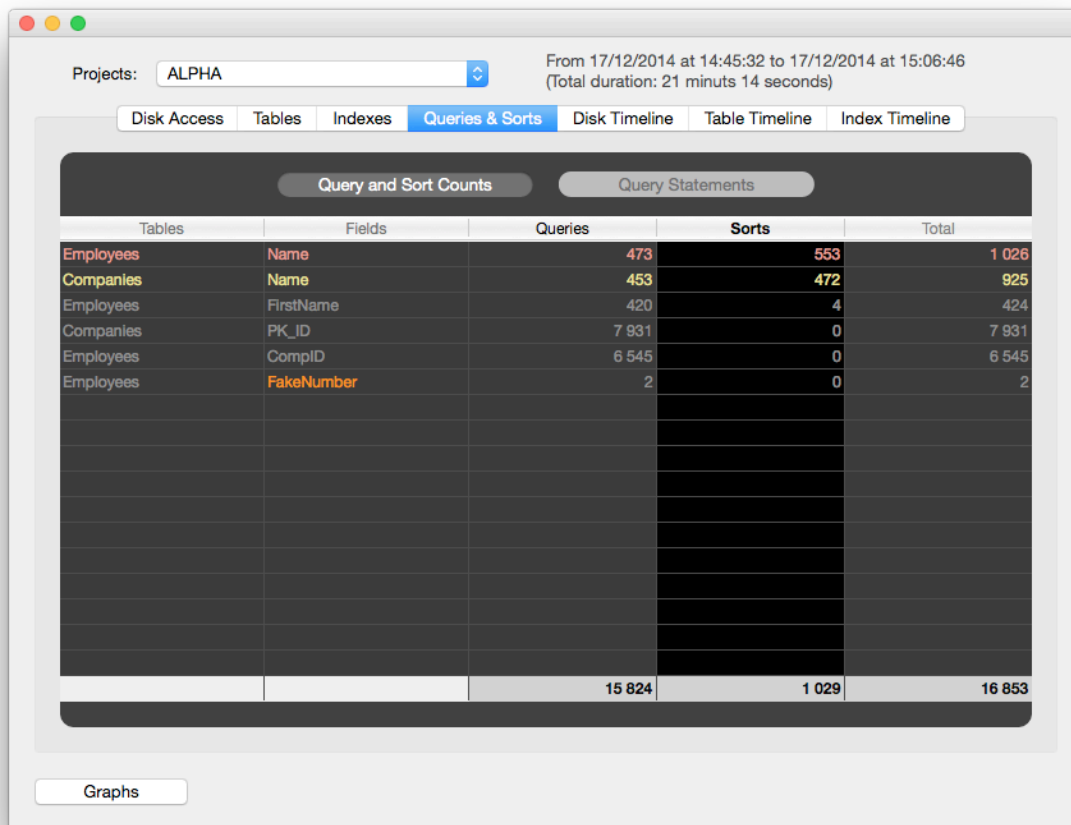
## Queries & Sorts (Counts)

This feature requires the usage of 4D v14 R5 during the log generation. In fact, the **Get database measures** command has been enhanced with v14 R5.

The logs generated with 4D v14 R3 and R4 versions do not contain information about queries or sorts.

The number of queries and sorts can also be displayed either graphically or in details. These queries and sorts numbers do not depend on indexes. Any query or any sort performed on any field will be taken into account.





In the detailed view mode, the fields that are either sorted or queried but are not indexed will be displayed in orange. According to the number of queries or sorts (and the number or records in the table) this *might* be an indication of a missing index. *Please note that this is only an indication! (See below).*

## Counter-Example!

In the case described below (see picture), you might think that employees have been sorted once by "name" and once by "first name" and, in this case, an index on the field "first name" *might* be useful. Right?

Employees	Name	0	1	1
Employees	First Name	0	1	1

This is not true. It only means that these fields have both been involved in a sort operation.

What happened in reality? Only one sort has been performed on both fields at once, "Name + First Name". This can be verified with the index activity counters.

As you can see on the first line, the field Employees.Name was never sorted alone. In the other hand, the index "Employees.Name,First Name" has been used for one sort.

In this case, everything is fine: There was a composite index and it has been used for this operation.

Index Names	Disk Read Bytes	Disk Write Bytes	Cache Read Bytes	Cache Miss Bytes	Disk Read Count	Disk Write Count	Cache Read Count	Cache Miss Count	Query Count	Sort Count
Employees.Name	0	17 050 620	0	0	0	3 254	0	0	0	0
Companies.PK_ID	68 376	0	4 225 760	67 760	22	0	1 372	22	464	0
Employees.Name,First Name	21 141 206	0	0	20 849 892	5 536	0	0	5 536	0	1

## Queries & Sorts (Statements)

This feature requires the usage of 4D v14 R5 during the log generation. In fact, the **Get database measures** command has been enhanced with v14 R5.

The logs generated with 4D v14 R3 and R4 versions do not contain information about queries or sorts.

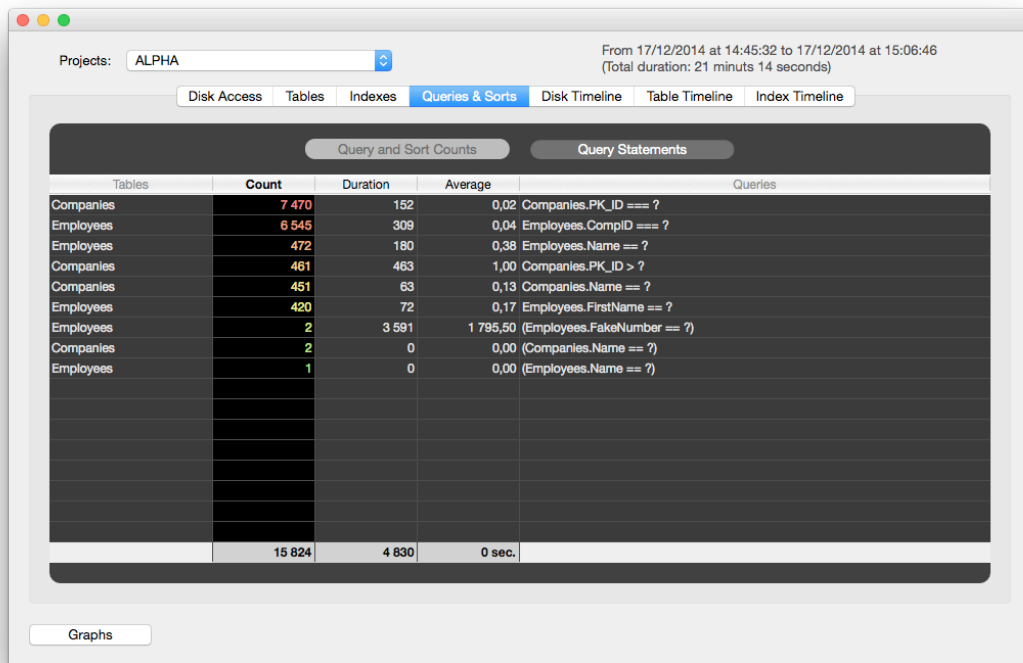
The query statements are also returned by the command "Get Database measures", so they are present in the log file generated by the component and can be displayed and analyzed.

These statements may come either from direct queries (Query dialog), programmed queries (Methods) or internal queries (Automatic links, etc.).

Only query counts, durations and statements are memorized; arguments are ignored.

*Examples:*

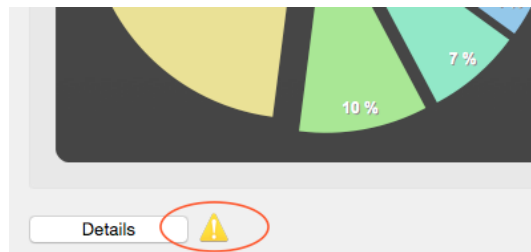
- The query (name = "a") and the query "name = "b"" are the same queries. The arguments "a" and "b" are ignored but the statement (name=) is counted twice.
- The query (name = "c") and query (name > "d") are distinct queries. They are each counted once for each statement.
- These arrays of result will show you which query is performed the greatest number or time, or take the longest time (in absolute or in average).



## Bottom page warning

A warning icon will be displayed next to the "Details" button at the bottom of page in some cases:

- When an index key has been added or removed but no query or sort has been performed on the indexed field
- When a non-indexed field has been used in a query or a sort
- When a query average is greater than 1000ms.



## Disk Timeline

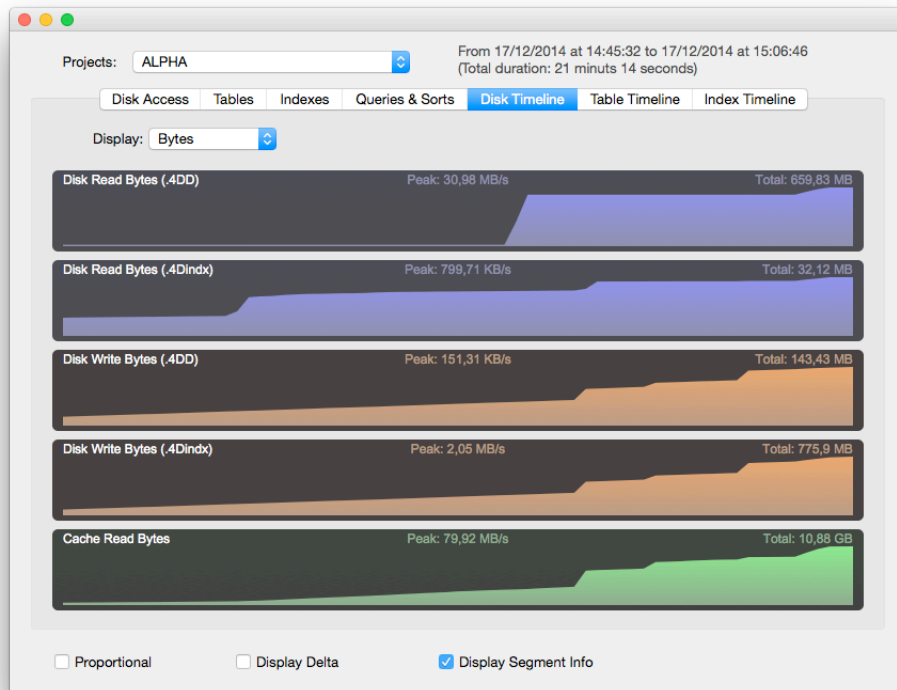
Starting from this tab, the graphics are based on the complete period of logging, not only the last sample.

The disk timeline represents the disk activity during the log period.

As long as the graphics are based on the result of the command "Get Database measures" that returns cumulated values since the launch of the database, the curves will never decrease.

Read and write operations are separated (blue / orange) as well as data and indexes (.4DD on top / .4Dindx on bottom).

It's interesting to have a look on totals... In the example below, less than 1GB have been read on the disk, but -good news- over 10GB have been read in the cache!



## **Peak value**

The "Peak" values need a little explanation: It is based on the difference between two logged measures samples. The value would be exact if a measure was done and logged in every second. However, it is not the case here.

In the example below, logs have been generated every 10 seconds. The biggest "Disk Read Bytes" difference between a sample and another was 309.8 MB. So the peak value should have been displayed "309.8 MB / 10 sec."

To enhance peak value readability, the peak value is divided by 10 (number of seconds between two log measures). So, the displayed value is more an "average of peak values" than a real "peak".

## **Display Delta option**

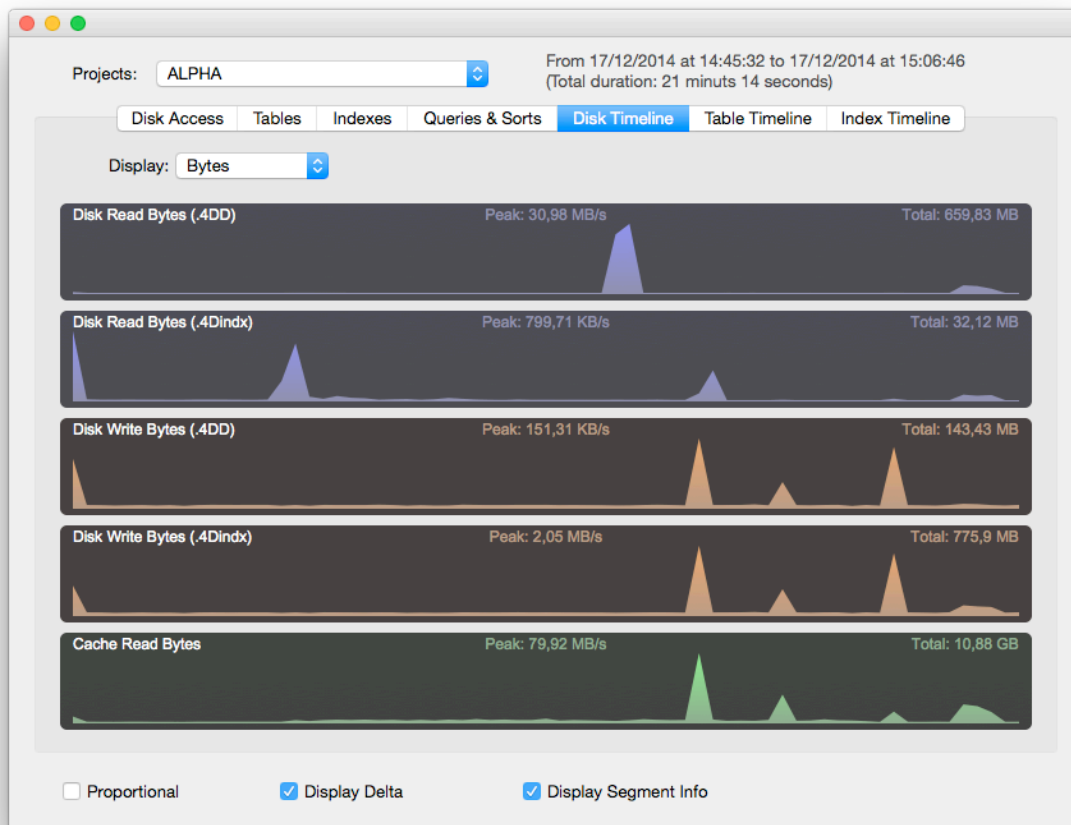
This option displays the differences between two measures rather than the raw content of each measure. This can make the graphics more easily comprehensive.

## **Display Segment info**

This option allows to group or to separate values for data and indexes during the read/write operation.

## **Proportional**

Finally, this option makes graphics proportional to two highest values of all the graphics. When it is not checked, the highest value of each graphic is used. Usually when this option is activated, all the graphics are flattened except the "Cache Read Bytes" (which, again, is normal and is a good thing!)



## Table Timeline & Index Timeline

These graphics will display the same kind of information as the previous one, but this time for a specific table (Table Timeline) or a specific index (Index Timeline).

## Conclusion

---

These two tools will help you to clearly understand what is happening inside your database as well as in the 4D Engine.

Number of disk access per seconds, number of GigaBytes read or written per hours, number of queries and sorts on fields, etc. All these information are now available thanks to the command "Get Database measures".

Remind that those graphics are based on the results of the command "Get Database measures" and that theses results are based on the database activity. If some tables or indexes do not appear in the graphics it only means that they don't have been used during the log period!

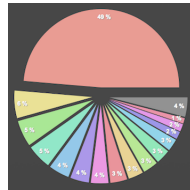
## Bonus

---

**Both component and app are open source!** Feel free to browse and learn how all the SVG charts are done or copy and paste code into your application.

Example:

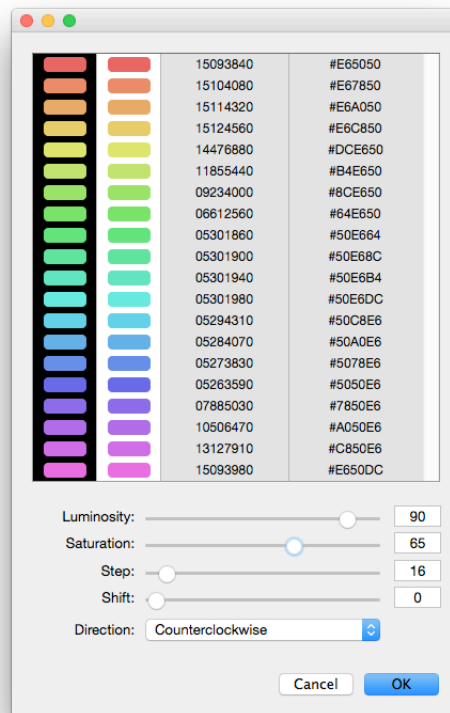
Did you ask yourself where those fancy colors came from? ;o)



There is a color generator integrated in the database. It can be called two ways: with or without a dialog. In both case it will fill arrays according to sent parameters. At least one array is needed, either a longint array or a text array (or both).

They will be filled with RGB values but formatted differently. 0xRRGGBB for the Longint array and #RRGGBB (hex mode) for the Text array.

You can call the dialog first ([ColorsPaletteDial](#)) to get used with the parameters (Luminosity, Saturation, Step, Shift and direction... then when you're satisfied, you can call the other command ([ColorsPaletteBuild](#)) directly!



```

ARRAY LONGINT ($_kkk_i;20)
ARRAY TEXT ($_kkk_s;20)

C_OBJECT ($param)

OB SET ($param;"palette_i";->$_kkk_i)
OB SET ($param;"palette_s";->$_kkk_s)

OB SET ($Param;"luminosity";80)
OB SET ($Param;"saturation";60)

ColorsPaletteDial ($Param)

```

```

OB SET ($param;"palette_i";-><>_Palette_i)
OB SET ($param;"palette_s";-><>_Palette_s)

OB SET ($Param;"luminosity";90)
OB SET ($Param;"saturation";45)
OB SET ($Param;"step";50) // (0+(2*360))/ $n //50
OB SET ($Param;"shift";5)

ColorsPaletteBuild ($param)

```