SCHARP

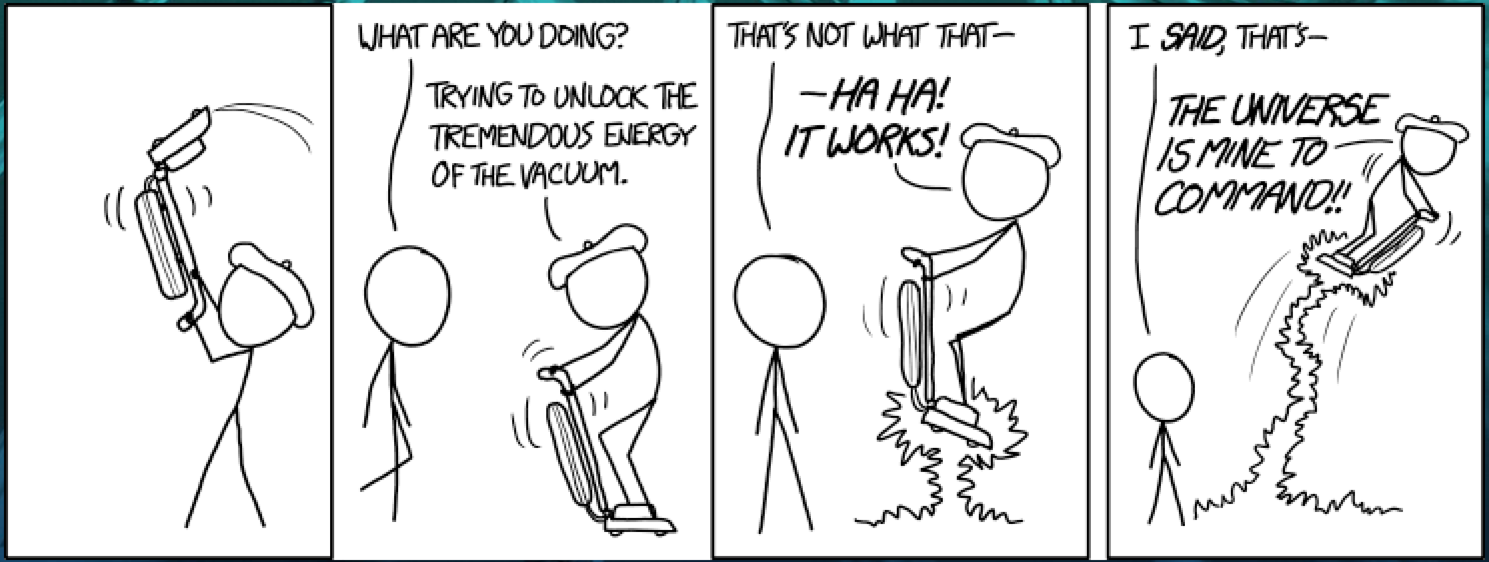
at FRED HUTCH

http://Lloyd.TheAlbins.com/AutoVacuum

AutoVacuum

By Lloyd Albin

Additional notes in the note section of the slide when you see this icon (📝) in the upper right corner of the slide.


WHAT ARE YOU DOING?
TRYING TO UNLOCK THE TREMENDOUS ENERGY OF THE VACUUM.
THAT'S NOT WHAT THAT—
—HA HA! IT WORKS!
I SAID, THAT'S—
THE UNIVERSE IS MINE TO COMMAND!!

SCHARP at FRED HUTCH

AutoVacuum: Is this presentation for me?

- Who should pay attention to this presentation?
  - Anyone who is running a lot of transactions, especially lots of deletes, updates, or rollbacked inserts. All of these commands cause deleted tuples (records) in the tables which needs to be vacuumed.
- Why should I adjust these values?
  - For most people PostgreSQL needs to be re-tuned because PostgreSQL is configured to run optimally on old hardware.
  - PostgreSQL has acknowledged this with Postgres 12 by changing ONE of the seven default AutoVacuum values.
- What happens if I don’t adjust the AutoVacuum?
  - Table bloat can happen, especially on heavily used PostgreSQL Clusters (Servers). Enough bloat can happen on very heavily used systems to run them out of disk space which then crashes the PostgreSQL Cluster.

AutoVacuum: Is this presentation for me?

- Where can I find out how to adjust the AutoVacuum?
  - This presentation gives you all the knowledge you need to be able to re-tune your AutoVacuum along with links to relevant documentation.
- When can I tell if the adjustments work?
  - After you manually vacuum one of the bloated table, you should be able to see that the bloat does not come back after churning the tuples (records) within the table.

Causes and Effects

Heavy churn on small table

- A Small table that gets thousands of rows added and deleted. This table is then joined to a large table to view a set of results.
    - *Effect: Your queries get slower and slower until a query that was 10 ms now takes 5 seconds or more, but if you start and stop your application, then the problem self fixes itself.*
    - *Caused by the AutoAnalyze not updating the table stats. This causes the query planner to run slowly and to pick the wrong query plan which then causes the query to perform in minutes vers milliseconds. A manual Analyze will fix this issue. This issue can be verified by checking the row count vers the estimated table rows in the stats and/or by monitoring the query planning time. When you stop and restart your application, this gives AutoVacuum and AutoAnalyze time to catch up and update the table stats.*
    - *Solution: Make your AutoVacuum/AutoAnalyze more aggressive if you have the spare Disk I/O. If you have high Disk I/O then you need to figure out the right trade off between database responsiveness and acceptable table bloat.*

Heavy churn on large tables

- Large table with tens of millions of rows where you are deleting and adding millions of rows per hour.
  - *Effect: You find that your queries are slowing down and when you look at your AutoVacuum, the AutoVacuum seems to be AutoVacuumig this table 24 hours per day, but if you stop your application for a few days or dump and restore your database it fixes itself.*

  1. *Caused by not enough time for the AutoVacuum to complete before the next churn cycle. This make the AutoVacuum take longer and longer to complete with the table bloat getting worse and worse until the AutoVacuum takes over 24 hours to complete. Stopping the Application lets the AutoVacuum catch up and then the queries run fast again. This can be fixed by either stopping the churn and doing a manual vacuum/analyze or by increasing your AutoVacuum's I/O rate by increasing the AutoVacuum Cost Limit. This can be verified by watching the currently running AutoVacuum's and the estimated dirty rows in the table stats.*
  2. *Caused by a long running transaction on the same server or a long running transaction on the secondary server, via streaming replication, that prevents the autovacuum event horizon from moving forward.*

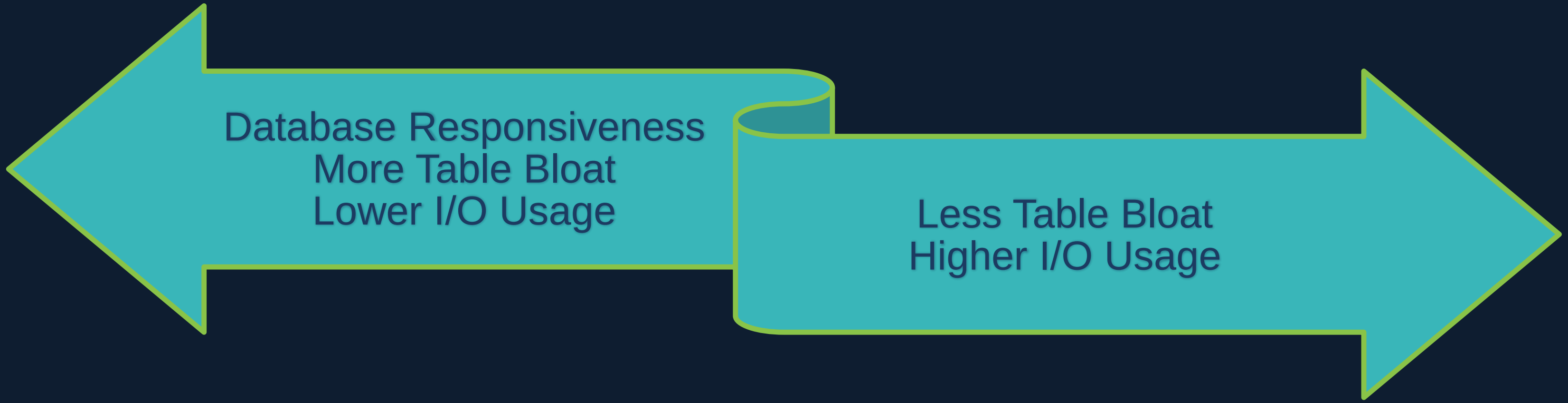
Heavy churn on large table indexs

- Large table with tens of millions of rows where you are deleting and then adding millions of rows per hour with the same index values inside of a transaction.
  - *Effect: Your queries get slower and slower, but if you dump and restore your database, everything is fast again.*
  - *Caused by the deleted records needing to be in the index at the same time as the new records. This means that all the index pages need to be split causing the index to double in size (or more) to allow the new entries to be inserted. The space is not recovered by the AutoVacuum system because it does not do a true vacuum of the index. The AutoVacuum will only vacuum/delete an index page which is 100% empty. The AutoVacuum does not do anything with partially filled pages except to remove deleted index entries. This can be verified by either looking for index bloat or by replacing the index and the query runs fast again.*
  - *Solution: Make your AutoVacuum more aggressive if you have the spare Disk I/O. If you have high Disk I/O then you need to figure out the right trade off between database responsiveness and the AutoVacuum*

Drop database is slow

- When you try dropping a database and it takes 5 minutes.
  - *Effect: When you try dropping a database it takes a long time, such as 5 minutes.*
  - *Caused by the AutoVacuum being very aggressive <u>and</u> the server having high Disk I/O with lots of Pending Writes.*
  - *Solution: This one is not so easy because you have to make tough choice between database responsiveness and table/index bloat. This is a process of trial and error finding the right balance between AutoVacuum's aggressiveness and Disk I/O and Pending Writes.*

High Disk I/O AutoVacuum Tunning Choices



AutoVacuum

What to monitor

What to Monitor

- Currently running AutoVacuum Process’s
- AutoVacuum Thresholds per Table
- Active and Idle in Transaction Connections
- AutoVacuum Threads in Use
- Max AutoVacuum’s
- CPU Usage
- Disc Read Usage
- Disc Write Usage
- Disc Pending Write
- Write Efficiency
- Disc I/O Time
- Last AutoVacuum per Table
- Disc Free Space
- Custom Table Settings Per Table
- Granted Locks
- AutoVacuum Settings

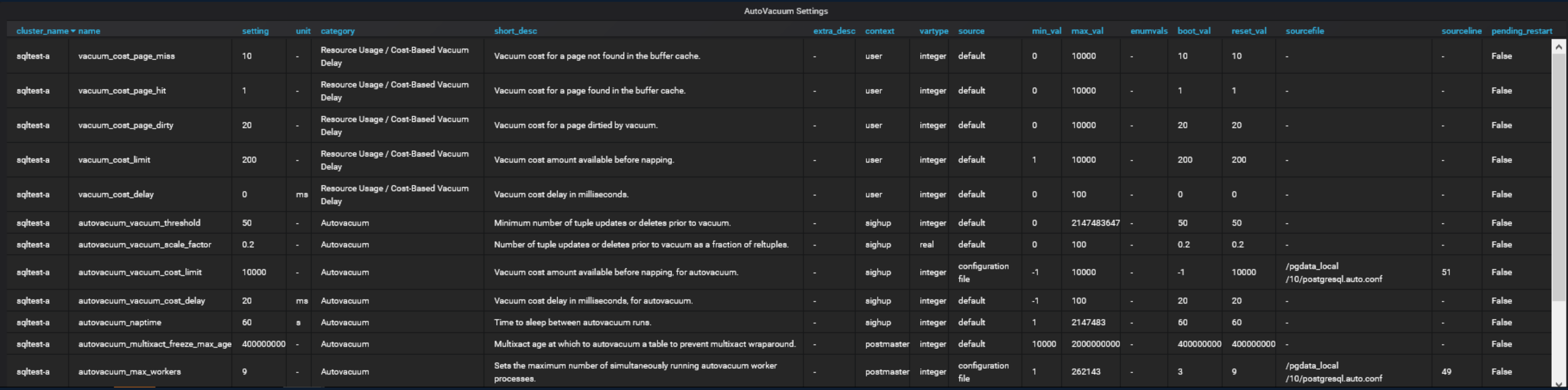
AutoVacuum Settings

Viewing the server settings

SCHARP at FRED HUTCH

AutoVacuum Settings

- This is more informational than anything, but the line will highlight when a restart is required for the change to take effect and Pending Restart will be True.
- Postgres 12+ vacuum_cost_delay's default changed from 20ms to 2ms
    - BUT if you upgrade your database to v12, you will still have your original databases defaults. You need to manually update this value.



AutoVacuum Settings

cluster_name ▾	name	setting	unit	category	short_desc	extra_desc	context	vartype	source	min_val	max_val	enumvals	boot_val	reset_val	sourcefile	sourceline	pending_restart
sqltest-a	vacuum_cost_page_miss	10	-	Resource Usage / Cost-Based Vacuum Delay	Vacuum cost for a page not found in the buffer cache.	-	user	integer	default	0	10000	-	10	10	-	-	False
sqltest-a	vacuum_cost_page_hit	1	-	Resource Usage / Cost-Based Vacuum Delay	Vacuum cost for a page found in the buffer cache.	-	user	integer	default	0	10000	-	1	1	-	-	False
sqltest-a	vacuum_cost_page_dirty	20	-	Resource Usage / Cost-Based Vacuum Delay	Vacuum cost for a page dirtied by vacuum.	-	user	integer	default	0	10000	-	20	20	-	-	False
sqltest-a	vacuum_cost_limit	200	-	Resource Usage / Cost-Based Vacuum Delay	Vacuum cost amount available before napping.	-	user	integer	default	1	10000	-	200	200	-	-	False
sqltest-a	vacuum_cost_delay	0	ms	Resource Usage / Cost-Based Vacuum Delay	Vacuum cost delay in milliseconds.	-	user	integer	default	0	100	-	0	0	-	-	False
sqltest-a	autovacuum_vacuum_threshold	50	-	Autovacuum	Minimum number of tuple updates or deletes prior to vacuum.	-	sighup	integer	default	0	2147483647	-	50	50	-	-	False
sqltest-a	autovacuum_vacuum_scale_factor	0.2	-	Autovacuum	Number of tuple updates or deletes prior to vacuum as a fraction of reltuples.	-	sighup	real	default	0	100	-	0.2	0.2	-	-	False
sqltest-a	autovacuum_vacuum_cost_limit	10000	-	Autovacuum	Vacuum cost amount available before napping, for autovacuum.	-	sighup	integer	configuration file	-1	10000	-	-1	10000	/pgdata_local/10/postgresql.auto.conf	51	False
sqltest-a	autovacuum_vacuum_cost_delay	20	ms	Autovacuum	Vacuum cost delay in milliseconds, for autovacuum.	-	sighup	integer	default	-1	100	-	20	20	-	-	False
sqltest-a	autovacuum_naptime	60	s	Autovacuum	Time to sleep between autovacuum runs.	-	sighup	integer	default	1	2147483	-	60	60	-	-	False
sqltest-a	autovacuum_multixact_freeze_max_age	400000000	-	Autovacuum	Multixact age at which to autovacuum a table to prevent multixact wraparound.	-	postmaster	integer	default	10000	2000000000	-	400000000	400000000	-	-	False
sqltest-a	autovacuum_max_workers	9	-	Autovacuum	Sets the maximum number of simultaneously running autovacuum worker processes.	-	postmaster	integer	configuration file	1	262143	-	3	9	/pgdata_local/10/postgresql.auto.conf	49	False

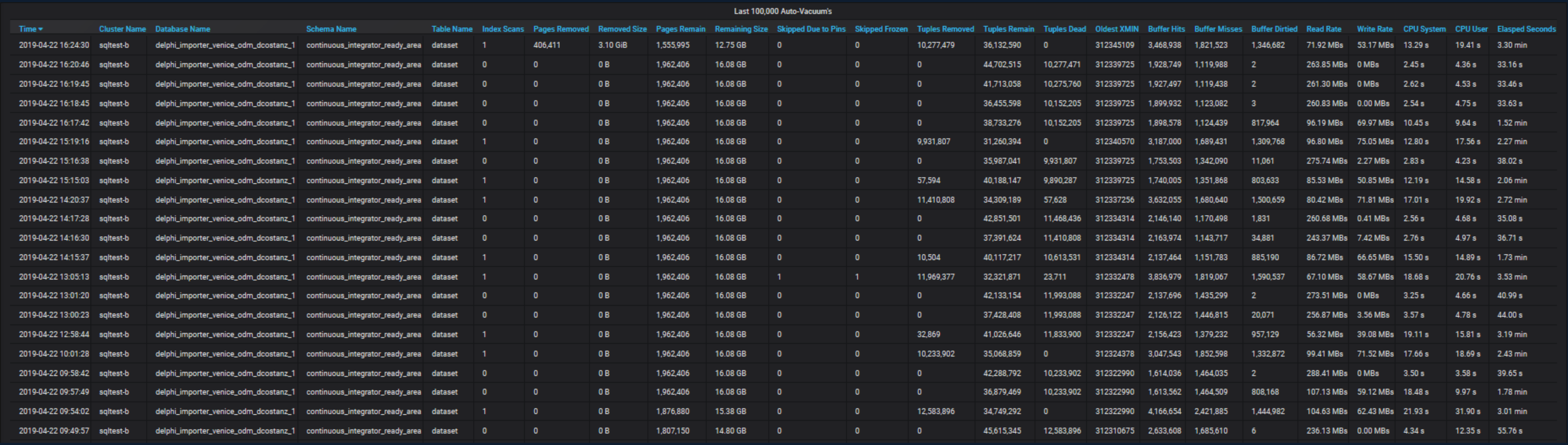
SCHARP at FRED HUTCH

AutoVacuum Logs

Viewing the log files

AutoVacuum Logs

- Pages Removed / Removed Size – The AutoVacuum was able to reduce the table size.
- Tuples Removed – The AutoVaccum was able to remove records.
- Tupled Dead – The AutoVacuum was not able to remove these records due to they were created after the Oldest XMIN aka Oldest Transaction ID.

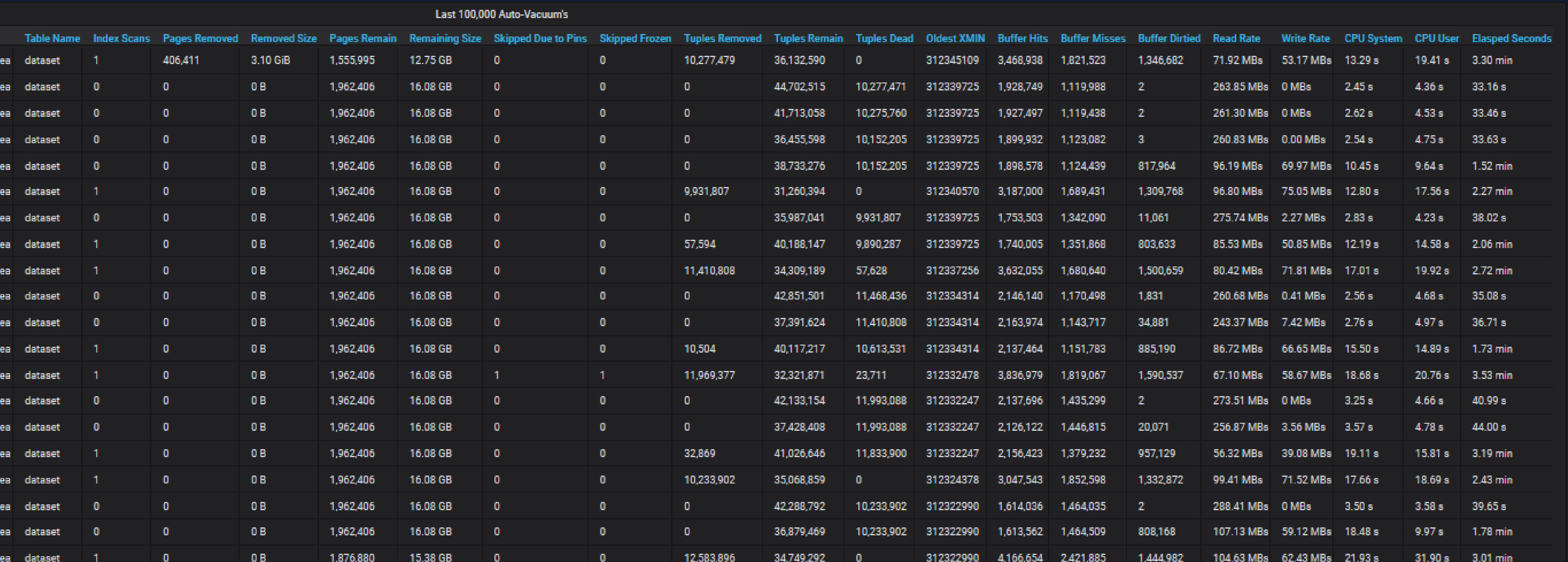


Last 100,000 Auto-Vacuum's

Time ▾	Cluster Name	Database Name	Schema Name	Table Name	Index Scans	Pages Removed	Removed Size	Pages Remain	Remaining Size	Skipped Due to Pins	Skipped Frozen	Tuples Removed	Tuples Remain	Tuples Dead	Oldest XMIN	Buffer Hits	Buffer Misses	Buffer Dirtied	Read Rate	Write Rate	CPU System	CPU User	Elasped Seconds
2019-04-22 16:24:30	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	1	406,411	3.10 GiB	1,555,995	12.75 GB	0	0	10,277,479	36,132,590	0	312345109	3,468,938	1,821,523	1,346,682	71.92 MBs	53.17 MBs	13.29 s	19.41 s	3.30 min
2019-04-22 16:20:46	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	44,702,515	10,277,471	312339725	1,928,749	1,119,988	2	263.85 MBs	0 MBs	2.45 s	4.36 s	33.16 s
2019-04-22 16:19:45	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	41,713,058	10,275,760	312339725	1,927,497	1,119,438	2	261.30 MBs	0 MBs	2.62 s	4.53 s	33.46 s
2019-04-22 16:18:45	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	36,455,598	10,152,205	312339725	1,899,932	1,123,082	3	260.83 MBs	0.00 MBs	2.54 s	4.75 s	33.63 s
2019-04-22 16:17:42	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	38,733,276	10,152,205	312339725	1,898,578	1,124,439	817,964	96.19 MBs	69.97 MBs	10.45 s	9.64 s	1.52 min
2019-04-22 15:19:16	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	1	0	0 B	1,962,406	16.08 GB	0	0	9,931,807	31,260,394	0	312340570	3,187,000	1,689,431	1,309,768	96.80 MBs	75.05 MBs	12.80 s	17.56 s	2.27 min
2019-04-22 15:16:38	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	35,987,041	9,931,807	312339725	1,753,503	1,342,090	11,061	275.74 MBs	2.27 MBs	2.83 s	4.23 s	38.02 s
2019-04-22 15:15:03	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	1	0	0 B	1,962,406	16.08 GB	0	0	57,594	40,188,147	9,890,287	312339725	1,740,005	1,351,868	803,633	85.53 MBs	50.85 MBs	12.19 s	14.58 s	2.06 min
2019-04-22 14:20:37	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	1	0	0 B	1,962,406	16.08 GB	0	0	11,410,808	34,309,189	57,628	312337256	3,632,055	1,680,640	1,500,659	80.42 MBs	71.81 MBs	17.01 s	19.92 s	2.72 min
2019-04-22 14:17:28	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	42,851,501	11,468,436	312334314	2,146,140	1,170,498	1,831	260.68 MBs	0.41 MBs	2.56 s	4.68 s	35.08 s
2019-04-22 14:16:30	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	37,391,624	11,410,808	312334314	2,163,974	1,143,717	34,881	243.37 MBs	7.42 MBs	2.76 s	4.97 s	36.71 s
2019-04-22 14:15:37	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	1	0	0 B	1,962,406	16.08 GB	0	0	10,504	40,117,217	10,613,531	312334314	2,137,464	1,151,783	885,190	86.72 MBs	66.65 MBs	15.50 s	14.89 s	1.73 min
2019-04-22 13:05:13	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	1	0	0 B	1,962,406	16.08 GB	1	1	11,969,377	32,321,871	23,711	312332478	3,836,979	1,819,067	1,590,537	67.10 MBs	58.67 MBs	18.68 s	20.76 s	3.53 min
2019-04-22 13:01:20	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	42,133,154	11,993,088	312332247	2,137,696	1,435,299	2	273.51 MBs	0 MBs	3.25 s	4.66 s	40.99 s
2019-04-22 13:00:23	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	37,428,408	11,993,088	312332247	2,126,122	1,446,815	20,071	256.87 MBs	3.56 MBs	3.57 s	4.78 s	44.00 s
2019-04-22 12:58:44	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	1	0	0 B	1,962,406	16.08 GB	0	0	32,869	41,026,646	11,833,900	312332247	2,156,423	1,379,232	957,129	56.32 MBs	39.08 MBs	19.11 s	15.81 s	3.19 min
2019-04-22 10:01:28	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	1	0	0 B	1,962,406	16.08 GB	0	0	10,233,902	35,068,859	0	312324378	3,047,543	1,852,598	1,332,872	99.41 MBs	71.52 MBs	17.66 s	18.69 s	2.43 min
2019-04-22 09:58:42	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	42,288,792	10,233,902	312322990	1,614,036	1,464,035	2	288.41 MBs	0 MBs	3.50 s	3.58 s	39.65 s
2019-04-22 09:57:49	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	36,879,469	10,233,902	312322990	1,613,562	1,464,509	808,168	107.13 MBs	59.12 MBs	18.48 s	9.97 s	1.78 min
2019-04-22 09:54:02	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	1	0	0 B	1,876,880	15.38 GB	0	0	12,583,896	34,749,292	0	312322990	4,166,654	2,421,885	1,444,982	104.63 MBs	62.43 MBs	21.93 s	31.90 s	3.01 min
2019-04-22 09:49:57	sqltest-b	delphi_importer_venice_odm_dcostanz_1	continuous_integrator_ready_area	dataset	0	0	0 B	1,807,150	14.80 GB	0	0	0	45,615,345	12,583,896	312310675	2,633,608	1,685,610	6	236.13 MBs	0.00 MBs	4.34 s	12.35 s	55.76 s

AutoVacuum Logs

- ((Buffer Hits * vacuum_cost_page_hit) + (Buffer Misses * vacuum_cost_page_miss) + (Buffer Dirtied * vacuum_cost_page_dirty)) = Total Cost
- (((Buffer Hits * vacuum_cost_page_hit) + (Buffer Misses * vacuum_cost_page_miss) + (Buffer Dirtied * vacuum_cost_page_dirty)) / autovacuum/vacuum_cost_limit) = Number of Delay Cycles
- (((Buffer Hits * vacuum_cost_page_hit) + (Buffer Misses * vacuum_cost_page_miss) + (Buffer Dirtied * vacuum_cost_page_dirty)) / autovacuum/vacuum_cost_limit) * autovacuum/vacuum_cost_delay = Total Delay's for Disk IO to Catch up.



Last 100,000 Auto-Vacuum's

	Table Name	Index Scans	Pages Removed	Removed Size	Pages Remain	Remaining Size	Skipped Due to Pins	Skipped Frozen	Tuples Removed	Tuples Remain	Tuples Dead	Oldest XMIN	Buffer Hits	Buffer Misses	Buffer Dirtied	Read Rate	Write Rate	CPU System	CPU User	Elasped Seconds
ea	dataset	1	406,411	3.10 GiB	1,555,995	12.75 GB	0	0	10,277,479	36,132,590	0	312345109	3,468,938	1,821,523	1,346,682	71.92 MBs	53.17 MBs	13.29 s	19.41 s	3.30 min
ea	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	44,702,515	10,277,471	312339725	1,928,749	1,119,988	2	263.85 MBs	0 MBs	2.45 s	4.36 s	33.16 s
ea	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	41,713,058	10,275,760	312339725	1,927,497	1,119,438	2	261.30 MBs	0 MBs	2.62 s	4.53 s	33.46 s
ea	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	36,455,598	10,152,205	312339725	1,899,932	1,123,082	3	260.83 MBs	0.00 MBs	2.54 s	4.75 s	33.63 s
ea	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	38,733,276	10,152,205	312339725	1,898,578	1,124,439	817,964	96.19 MBs	69.97 MBs	10.45 s	9.64 s	1.52 min
ea	dataset	1	0	0 B	1,962,406	16.08 GB	0	0	9,931,807	31,260,394	0	312340570	3,187,000	1,689,431	1,309,768	96.80 MBs	75.05 MBs	12.80 s	17.56 s	2.27 min
ea	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	35,987,041	9,931,807	312339725	1,753,503	1,342,090	11,061	275.74 MBs	2.27 MBs	2.83 s	4.23 s	38.02 s
ea	dataset	1	0	0 B	1,962,406	16.08 GB	0	0	57,594	40,188,147	9,890,287	312339725	1,740,005	1,351,868	803,633	85.53 MBs	50.85 MBs	12.19 s	14.58 s	2.06 min
ea	dataset	1	0	0 B	1,962,406	16.08 GB	0	0	11,410,808	34,309,189	57,628	312337256	3,632,055	1,680,640	1,500,659	80.42 MBs	71.81 MBs	17.01 s	19.92 s	2.72 min
ea	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	42,851,501	11,468,436	312334314	2,146,140	1,170,498	1,831	260.68 MBs	0.41 MBs	2.56 s	4.68 s	35.08 s
ea	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	37,391,624	11,410,808	312334314	2,163,974	1,143,717	34,881	243.37 MBs	7.42 MBs	2.76 s	4.97 s	36.71 s
ea	dataset	1	0	0 B	1,962,406	16.08 GB	0	0	10,504	40,117,217	10,613,531	312334314	2,137,464	1,151,783	885,190	86.72 MBs	66.65 MBs	15.50 s	14.89 s	1.73 min
ea	dataset	1	0	0 B	1,962,406	16.08 GB	1	1	11,969,377	32,321,871	23,711	312332478	3,836,979	1,819,067	1,590,537	67.10 MBs	58.67 MBs	18.68 s	20.76 s	3.53 min
ea	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	42,133,154	11,993,088	312332247	2,137,696	1,435,299	2	273.51 MBs	0 MBs	3.25 s	4.66 s	40.99 s
ea	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	37,428,408	11,993,088	312332247	2,126,122	1,446,815	20,071	256.87 MBs	3.56 MBs	3.57 s	4.78 s	44.00 s
ea	dataset	1	0	0 B	1,962,406	16.08 GB	0	0	32,869	41,026,646	11,833,900	312332247	2,156,423	1,379,232	957,129	56.32 MBs	39.08 MBs	19.11 s	15.81 s	3.19 min
ea	dataset	1	0	0 B	1,962,406	16.08 GB	0	0	10,233,902	35,068,859	0	312324378	3,047,543	1,852,598	1,332,872	99.41 MBs	71.52 MBs	17.66 s	18.69 s	2.43 min
ea	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	42,288,792	10,233,902	312322990	1,614,036	1,464,035	2	288.41 MBs	0 MBs	3.50 s	3.58 s	39.65 s
ea	dataset	0	0	0 B	1,962,406	16.08 GB	0	0	0	36,879,469	10,233,902	312322990	1,613,562	1,464,509	808,168	107.13 MBs	59.12 MBs	18.48 s	9.97 s	1.78 min
ea	dataset	1	0	0 B	1,876,880	15.38 GB	0	0	12,583,896	34,749,292	0	312322990	4,166,654	2,421,885	1,444,982	104.63 MBs	62.43 MBs	21.93 s	31.90 s	3.01 min

AutoVacuum Logs

- Here is how these formulas affect how long the AutoVacuum waits idle to let the Disk IO catch up for other processes.
- Table Size 15.85 GB, 3.10 GB Remove from the end of the table by AutoVacuum with 12.75 GB remaining. Removing 10,277,479 Tuples / Records / Rows.

Buffer Hits	vacuum_cost_page_hit	Buffer Misses	vacuum_cost_page_miss	Buffers Dirtied	vacuum_cost_page_dirty	Total Cost
3,468,938	1	1,821,682	10	1,346,682	20	48,619,398

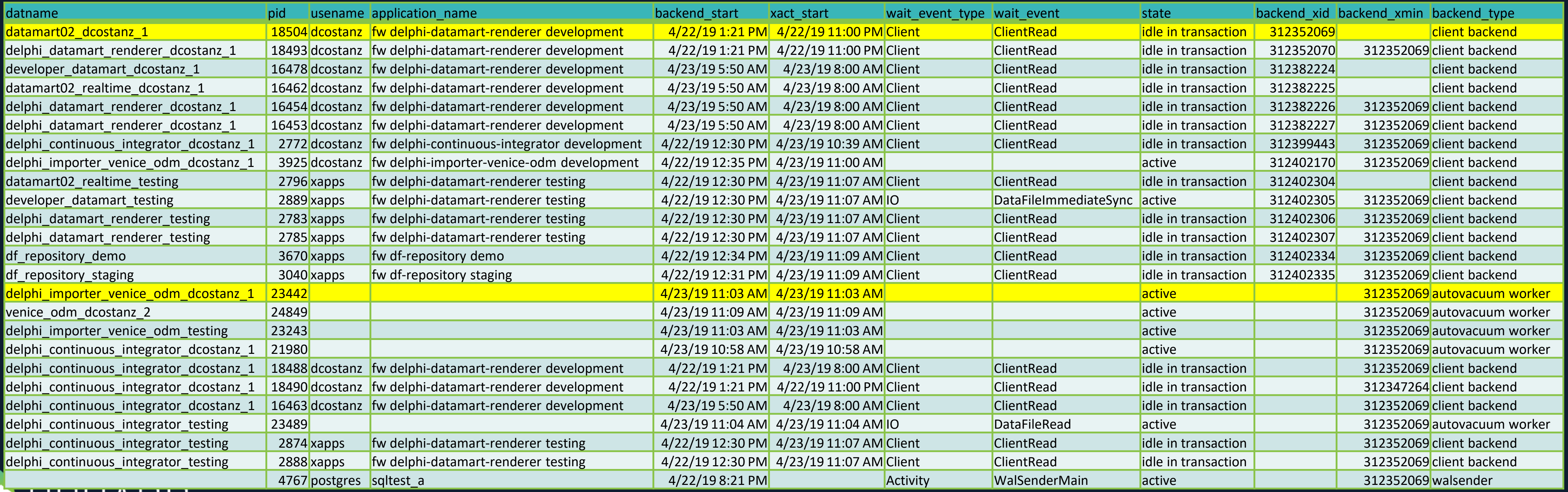
Total Cost	autovacuum_cost_limit	Numer of Delay Cycles	autovacuum_cost_delay	Total Delay's for Disk IO
48,619,398	200	243,096	20 ms	4,861,940 ms or 81 m
48,619,398	200	243,096	2 ms	486,194 ms or 8 m
48,619,398	10,000	4,861	20 ms	97,239 ms or 1 m
48,619,398	8,000	6,077	20 ms	121,548 ms or 2 m
48,619,398	5,000	9,723	20 ms	194,478 ms or 3 m
48,619,398	500	97,238	2 ms	194,478 ms or 3 m

- This AutoVacuum took 3.30 minutes with the autovacuum_cost_limit of 10,000 of which 1.63 minutes was the AutoVacuum sitting idle. This means that the AutoVacuum really only takes 1.67 minutes. This means that the default setting on PostgreSQL 11- would be 82.67 minutes and on PostgreSQL 12+ 9.67 minutes. Using out current tuning, it takes 4.67 minutes.
- If your Tuple is over 2K LZ compressed, the extra amount will be stored in the Toast Table.

SCHARP at FRED HUTCH

pg_stat_activity

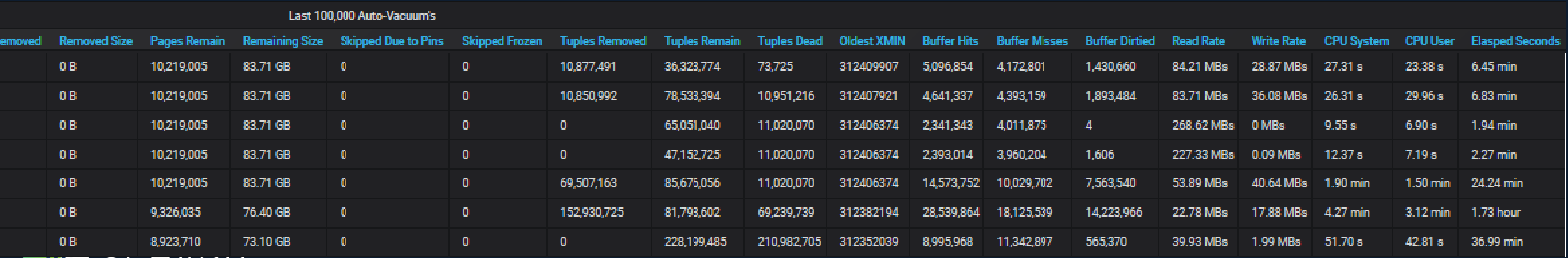
- Looking at pg_stat_activity, we can see that the long running transaction in a different database is creating the xmin event horizon that the autovacuum is using. Only items older than the xmin event horizon will be autovacuumed.
- backend_xid = Top-level transaction identifier of this backend, if any.
- backend_xmin = The current backend's xmin horizon.
- The xmin event horizon can also come from a long running transaction on the secondary server due to hot_standby_feedback turn on for streaming replication.
  - However, the cleanup situation will be no worse than if the standby queries were running directly on the primary server, and you are still getting the benefit of off-loading execution onto the standby.



datname	pid	usename	application_name	backend_start	xact_start	wait_event_type	wait_event	state	backend_xid	backend_xmin	backend_type
datamart02_dcostanz_1	18504	dcostanz	fw delphi-datamart-renderer development	4/22/19 1:21 PM	4/22/19 11:00 PM	Client	ClientRead	idle in transaction	312352069		client backend
delphi_datamart_renderer_dcostanz_1	18493	dcostanz	fw delphi-datamart-renderer development	4/22/19 1:21 PM	4/22/19 11:00 PM	Client	ClientRead	idle in transaction	312352070	312352069	client backend
developer_datamart_dcostanz_1	16478	dcostanz	fw delphi-datamart-renderer development	4/23/19 5:50 AM	4/23/19 8:00 AM	Client	ClientRead	idle in transaction	312382224		client backend
datamart02_realtime_dcostanz_1	16462	dcostanz	fw delphi-datamart-renderer development	4/23/19 5:50 AM	4/23/19 8:00 AM	Client	ClientRead	idle in transaction	312382225		client backend
delphi_datamart_renderer_dcostanz_1	16454	dcostanz	fw delphi-datamart-renderer development	4/23/19 5:50 AM	4/23/19 8:00 AM	Client	ClientRead	idle in transaction	312382226	312352069	client backend
delphi_datamart_renderer_dcostanz_1	16453	dcostanz	fw delphi-datamart-renderer development	4/23/19 5:50 AM	4/23/19 8:00 AM	Client	ClientRead	idle in transaction	312382227	312352069	client backend
delphi_continuous_integrator_dcostanz_1	2772	dcostanz	fw delphi-continuous-integrator development	4/22/19 12:30 PM	4/23/19 10:39 AM	Client	ClientRead	idle in transaction	312399443	312352069	client backend
delphi_importer_venice_odm_dcostanz_1	3925	dcostanz	fw delphi-importer-venice-odm development	4/22/19 12:35 PM	4/23/19 11:00 AM			active	312402170	312352069	client backend
datamart02_realtime_testing	2796	xapps	fw delphi-datamart-renderer testing	4/22/19 12:30 PM	4/23/19 11:07 AM	Client	ClientRead	idle in transaction	312402304		client backend
developer_datamart_testing	2889	xapps	fw delphi-datamart-renderer testing	4/22/19 12:30 PM	4/23/19 11:07 AM	IO	DataFileImmediateSync	active	312402305	312352069	client backend
delphi_datamart_renderer_testing	2783	xapps	fw delphi-datamart-renderer testing	4/22/19 12:30 PM	4/23/19 11:07 AM	Client	ClientRead	idle in transaction	312402306	312352069	client backend
delphi_datamart_renderer_testing	2785	xapps	fw delphi-datamart-renderer testing	4/22/19 12:30 PM	4/23/19 11:07 AM	Client	ClientRead	idle in transaction	312402307	312352069	client backend
df_repository_demo	3670	xapps	fw df-repository demo	4/22/19 12:34 PM	4/23/19 11:09 AM	Client	ClientRead	idle in transaction	312402334	312352069	client backend
df_repository_staging	3040	xapps	fw df-repository staging	4/22/19 12:31 PM	4/23/19 11:09 AM	Client	ClientRead	idle in transaction	312402335	312352069	client backend
delphi_importer_venice_odm_dcostanz_1	23442			4/23/19 11:03 AM	4/23/19 11:03 AM			active		312352069	autovacuum worker
venice_odm_dcostanz_2	24849			4/23/19 11:09 AM	4/23/19 11:09 AM			active		312352069	autovacuum worker
delphi_importer_venice_odm_testing	23243			4/23/19 11:03 AM	4/23/19 11:03 AM			active		312352069	autovacuum worker
delphi_continuous_integrator_dcostanz_1	21980			4/23/19 10:58 AM	4/23/19 10:58 AM			active		312352069	autovacuum worker
delphi_continuous_integrator_dcostanz_1	18488	dcostanz	fw delphi-datamart-renderer development	4/22/19 1:21 PM	4/23/19 8:00 AM	Client	ClientRead	idle in transaction		312352069	client backend
delphi_continuous_integrator_dcostanz_1	18490	dcostanz	fw delphi-datamart-renderer development	4/22/19 1:21 PM	4/22/19 11:00 PM	Client	ClientRead	idle in transaction		312347264	client backend
delphi_continuous_integrator_dcostanz_1	16463	dcostanz	fw delphi-datamart-renderer development	4/23/19 5:50 AM	4/23/19 8:00 AM	Client	ClientRead	idle in transaction		312352069	client backend
delphi_continuous_integrator_testing	23489			4/23/19 11:04 AM	4/23/19 11:04 AM	IO	DataFileRead	active		312352069	autovacuum worker
delphi_continuous_integrator_testing	2874	xapps	fw delphi-datamart-renderer testing	4/22/19 12:30 PM	4/23/19 11:07 AM	Client	ClientRead	idle in transaction		312352069	client backend
delphi_continuous_integrator_testing	2888	xapps	fw delphi-datamart-renderer testing	4/22/19 12:30 PM	4/23/19 11:07 AM	Client	ClientRead	idle in transaction		312352069	client backend
	4767	postgres	sqltest_a	4/22/19 8:21 PM		Activity	WalSenderMain	active		312352069	walsender

AutoVacuum Logs

- You will notice that once the blocking transactions completed that the table went from 228,199,485 records to 36,323,774 records but did not decrease in size.
- AutoVacuum:
  - Removes empty pages at the end of the table.
  - Mark's old records as reusable space.
  - DOES NOT condense pages
  - DOES NOT remove empty pages in the middle of the table
- This means that once your table is bloated like this, there are only several solutions.
  - Vacuum Full
  - Cluster
  - Truncate and re-insert the data – Truncate cleans up the pages immediately.
- This type of bloat will cause sequential table scans to run slowly, because they have to read every page, even the empty ones.



Last 100,000 Auto-Vacuum's

emoved	Removed Size	Pages Remain	Remaining Size	Skipped Due to Pins	Skipped Frozen	Tuples Removed	Tuples Remain	Tuples Dead	Oldest XMIN	Buffer Hits	Buffer Misses	Buffer Dirtied	Read Rate	Write Rate	CPU System	CPU User	Elasped Seconds
	0 B	10,219,005	83.71 GB	0	0	10,877,491	36,323,774	73,725	312409907	5,096,854	4,172,801	1,430,660	84.21 MBs	28.87 MBs	27.31 s	23.38 s	6.45 min
	0 B	10,219,005	83.71 GB	0	0	10,850,992	78,533,394	10,951,216	312407921	4,641,337	4,393,159	1,893,484	83.71 MBs	36.08 MBs	26.31 s	29.96 s	6.83 min
	0 B	10,219,005	83.71 GB	0	0	0	65,051,040	11,020,070	312406374	2,341,343	4,011,875	4	268.62 MBs	0 MBs	9.55 s	6.90 s	1.94 min
	0 B	10,219,005	83.71 GB	0	0	0	47,152,725	11,020,070	312406374	2,393,014	3,960,204	1,606	227.33 MBs	0.09 MBs	12.37 s	7.19 s	2.27 min
	0 B	10,219,005	83.71 GB	0	0	69,507,163	85,675,056	11,020,070	312406374	14,573,752	10,029,702	7,563,540	53.89 MBs	40.64 MBs	1.90 min	1.50 min	24.24 min
	0 B	9,326,035	76.40 GB	0	0	152,930,725	81,793,602	69,239,739	312382194	28,539,864	18,125,539	14,223,966	22.78 MBs	17.88 MBs	4.27 min	3.12 min	1.73 hour
	0 B	8,923,710	73.10 GB	0	0	0	228,199,485	210,982,705	312352039	8,995,968	11,342,897	565,370	39.93 MBs	1.99 MBs	51.70 s	42.81 s	36.99 min

SCHARP at FRED HUTCH

Finding tuple usage on each page within the table

- We can find out the used verse unused tuples on each page by running this query.
- To be able to run this query you need to load my toolset from https://github.com/LloydAlbin/SCHARP-PG-DBA-Debugging-Tools which uses the pageinspect extension.

```
SELECT
  p,
  sum(unused_tuples) AS unused_tuples,
  sum(used_tuples) AS used_tuples,
  sum(deleted_tuples) AS deleted_tuples
FROM (
  SELECT
    p,
    CASE WHEN (t_xmin IS NULL AND t_xmax IS NULL)
      THEN 1
      ELSE 0
      END AS unused_tuples,
    CASE WHEN (t_xmin IS NULL AND t_xmax IS NULL)
      THEN 0
      ELSE 1
      END AS used_tuples
    CASE WHEN heap_xmax_committed
      THEN 1
      ELSE 0
      END AS deleted_tuples
  FROM
tools.heap_page_item_attrs_details('continuous_integrator_re
ady_area.dataset’)
) a
GROUP BY p
ORDER BY p;
```

SCHARP at FRED HUTCH

Finding 100% empty pages

- If create a view from the previous query and then write it’s output to a table, we can run various queries against the results.
- One such query is how many pages are totally blank, this are only recoverable with a VACUUM FULL.
- Default block size is 8K. You can, but should not, override this when compiling PostgreSQL.

```
CREATE TABLE tools.dataset_pages AS
SELECT * FROM previous_pages_view;

SELECT
  count(*) AS empty_pages,
  count(*) * current_setting('block_size')::bigint AS bytes,
  pg_size_pretty(count(*) * current_setting('block_size')::bigint)
    AS empty_page_size
FROM tools.dataset_pages
WHERE used_tuples = 0;
```

empty_pages	bytes	empty_page_size
7,568,398	62,000,316,416	58GB

Seeing the total bloat in tuples

- In the first result, I am using the WHERE clause to ignoring the 100% empty pages.
- In the second result, I am including the 100% empty pages.

```
SELECT
  sum(used_tuples) AS used_tuples,
  sum(unused_tuples) AS unused_tuples,
  sum(deleted_tuples) AS deleted_tuples
FROM tools.dataset_pages
WHERE used_tuples > 0;

SELECT
  sum(used_tuples) AS used_tuples,
  sum(unused_tuples) AS unused_tuples,
  sum(deleted_tuples) AS deleted_tuples
FROM tools.dataset_pages;
```

used_tuples	unused_tuples	deleted_tuples
66,813,680	6,280,558	39,466,746
66,813,680	203,492,055	39,466,746

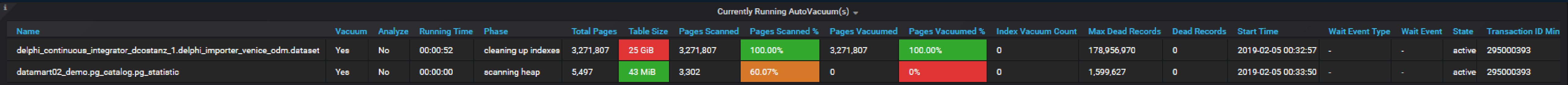
Currently Running AutoVacuum(s)

Knowing what is a happening and the speed at which it is happening.

Requires PostgreSQL 9.6+

Current Running AutoVacuum(s)

- Table Name
  - This will be displayed in one of the following formats: Cluster.Database.Schema.Table or Database.Schema.Table or Schema.Table depending on your template settings.
- Vacuum / Analyze
  - This shows you if vacuum and/or Analyze are going to be happening
- Running Time
  - How long the AutoVacuum has been running on this specific talbe.
- Phase
  - Initializing, scanning heap, vacuuming indexes, vacuuming heap, cleaning up indexes, truncating heap, performing final cleanup
- Total Pages
  - This is the number of pages that needs to be processed. Pages by default are 8K.
- Table Size
  - This is the heap_blks_read * Yours block/page size giving you the size of your table on disc.

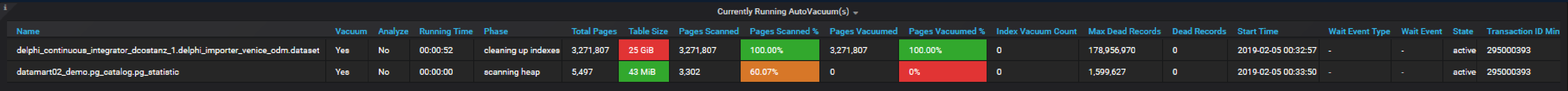


Currently Running AutoVacuum(s)

Name	Vacuum	Analyze	Running Time	Phase	Total Pages	Table Size	Pages Scanned	Pages Scanned %	Pages Vacuumed	Pages Vacuumed %	Index Vacuum Count	Max Dead Records	Dead Records	Start Time	Wait Event Type	Wait Event	State	Transaction ID Min
delphi_continuous_integrator_dcostanz_1.delphi_importer_venice_odm.dataset	Yes	No	00:00:52	cleaning up indexes	3,271,807	25 GiB	3,271,807	100.00%	3,271,807	100.00%	0	178,956,970	0	2019-02-05 00:32:57	-	-	active	295000393
datamart02_demo.pg_catalog.pg_statistic	Yes	No	00:00:00	scanning heap	5,497	43 MiB	3,302	60.07%	0	0%	0	1,599,627	0	2019-02-05 00:33:50	-	-	active	295000393

Current Running AutoVacuum(s)

- Pages Scanned
  - This is the number of blocks/pages that have been scanned. By watching this, you can tell how fast this part of the process it taking.
- Pages Scanned %
  - This is the percent of blocks/pages that have been scanned. By watching this, you can tell how fast this part of the process it taking.
- Pages Vacuumed
  - This is the number of blocks/pages that have been vacuumed. By watching this, you can tell how fast this part of the process it taking.
- Pages Vacuumed %
  - This is the percent of blocks/pages that have been vacuumed. By watching this, you can tell how fast this part of the process it taking.
- Index Vacuum Count
  - After the “vacuuming indexes” stage has been processed, it will show you the number of indexes.
- Max Dead Records
  - This is the max number of records that can be processed before an index vacuum is required.

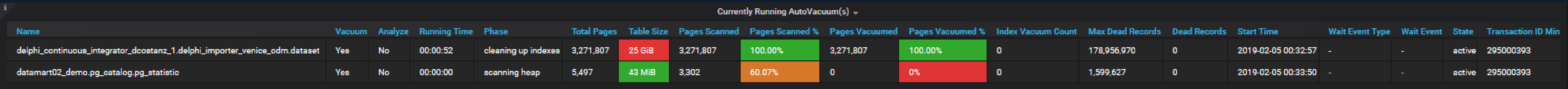


Currently Running AutoVacuum(s)

Name	Vacuum	Analyze	Running Time	Phase	Total Pages	Table Size	Pages Scanned	Pages Scanned %	Pages Vacuumed	Pages Vacuumed %	Index Vacuum Count	Max Dead Records	Dead Records	Start Time	Wait Event Type	Wait Event	State	Transaction ID Min
delphi_continuous_integrator_dcostanz_1.delphi_importer_venice_odm.dataset	Yes	No	00:00:52	cleaning up indexes	3,271,807	25 GiB	3,271,807	100.00%	3,271,807	100.00%	0	178,956,970	0	2019-02-05 00:32:57	-	-	active	295000393
datamart02_demo.pg_catalog.pg_statistic	Yes	No	00:00:00	scanning heap	5,497	43 MiB	3,302	60.07%	0	0%	0	1,599,627	0	2019-02-05 00:33:50	-	-	active	295000393

Current Running AutoVacuum(s)

- Dead Records
  - Estimated number of dead tuples.
- Start Time
  - This is when the AutoVacuum started working on this table.
- Wait Event Type
  - LWLock, Lock, BufferPin, Activity, Extension, Client, IPC, Timeout, IO
- Wait Event
  - To many to cover here, see https://www.postgresql.org/docs/current/monitoring-stats.html#WAIT-EVENT-TABLE
- State
  - This should normally read “Active”.
- Transaction ID Min
  - This is the oldest Transaction ID that could read the table.



Currently Running AutoVacuum(s)

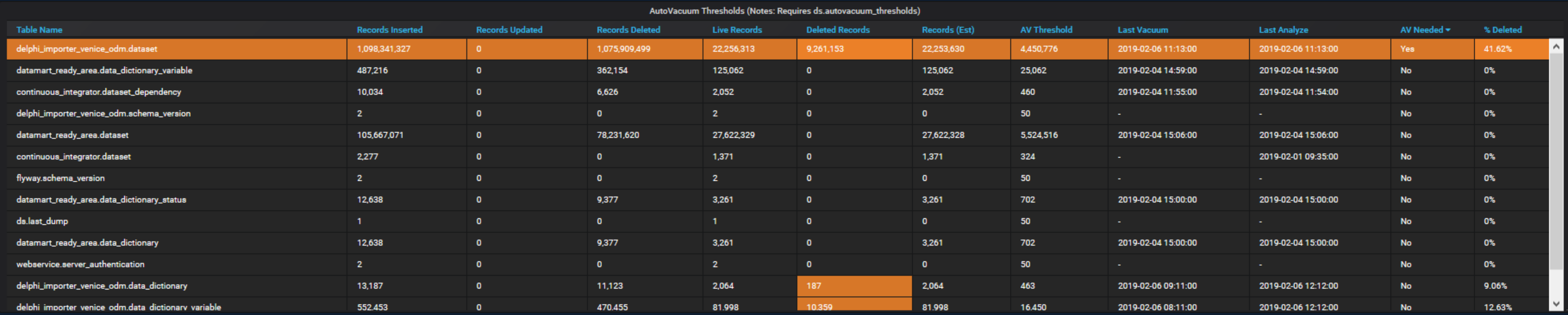
Name	Vacuum	Analyze	Running Time	Phase	Total Pages	Table Size	Pages Scanned	Pages Scanned %	Pages Vacuumed	Pages Vacuumed %	Index Vacuum Count	Max Dead Records	Dead Records	Start Time	Wait Event Type	Wait Event	State	Transaction ID Min
delphi_continuous_integrator_dcostanz_1.delphi_importer_venice_odm.dataset	Yes	No	00:00:52	cleaning up indexes	3,271,807	25 GiB	3,271,807	100.00%	3,271,807	100.00%	0	178,956,970	0	2019-02-05 00:32:57	-	-	active	295000393
datamart02_demo.pg_catalog.pg_statistic	Yes	No	00:00:00	scanning heap	5,497	43 MiB	3,302	60.07%	0	0%	0	1,599,627	0	2019-02-05 00:33:50	-	-	active	295000393

AutoVacuum Thresholds

Knowing when a table will be AutoVacuum’ed

AutoVacuum Thresholds

- Table Name
  - This will be displayed in one of the following formats: Cluster.Database.Schema.Table or Database.Schema.Table or Schema.Table depending on your template settings.
- Records Inserted, Records Updated, Records Deleted
  - This is the total number of records inserted, updated and deleted. These numbers will only ever go up.
- Live Records, Deleted Records, Record (est)
  - This is the estimated number of readable records, deleted records and total records.



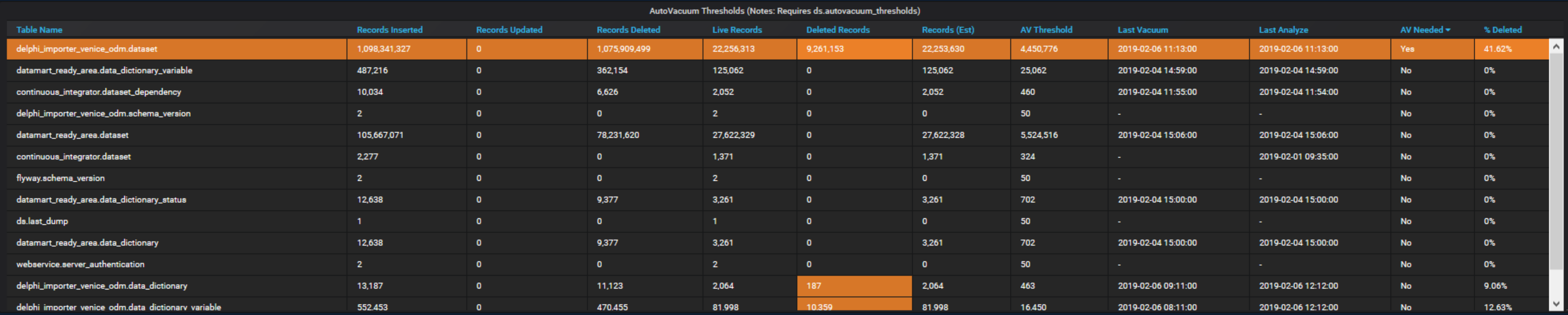
AutoVacuum Thresholds (Notes: Requires ds.autovacuum_thresholds)

Table Name	Records Inserted	Records Updated	Records Deleted	Live Records	Deleted Records	Records (Est)	AV Threshold	Last Vacuum	Last Analyze	AV Needed ▾	% Deleted
delphi_importer_venice_odm.dataset	1,098,341,327	0	1,075,909,499	22,256,313	9,261,153	22,253,630	4,450,776	2019-02-06 11:13:00	2019-02-06 11:13:00	Yes	41.62%
datamart_ready_area.data_dictionary_variable	487,216	0	362,154	125,062	0	125,062	25,062	2019-02-04 14:59:00	2019-02-04 14:59:00	No	0%
continuous_integrator.dataset_dependency	10,034	0	6,626	2,052	0	2,052	460	2019-02-04 11:55:00	2019-02-04 11:54:00	No	0%
delphi_importer_venice_odm.schema_version	2	0	0	2	0	0	50	-	-	No	0%
datamart_ready_area.dataset	105,667,071	0	78,231,620	27,622,329	0	27,622,328	5,524,516	2019-02-04 15:06:00	2019-02-04 15:06:00	No	0%
continuous_integrator.dataset	2,277	0	0	1,371	0	1,371	324	-	2019-02-01 09:35:00	No	0%
flyway.schema_version	2	0	0	2	0	0	50	-	-	No	0%
datamart_ready_area.data_dictionary_status	12,638	0	9,377	3,261	0	3,261	702	2019-02-04 15:00:00	2019-02-04 15:00:00	No	0%
ds.last_dump	1	0	0	1	0	0	50	-	-	No	0%
datamart_ready_area.data_dictionary	12,638	0	9,377	3,261	0	3,261	702	2019-02-04 15:00:00	2019-02-04 15:00:00	No	0%
webservice.server_authentication	2	0	0	2	0	0	50	-	-	No	0%
delphi_importer_venice_odm.data_dictionary	13,187	0	11,123	2,064	187	2,064	463	2019-02-06 09:11:00	2019-02-06 12:12:00	No	9.06%
delphi_importer_venice_odm.data_dictionary_variable	552,453	0	470,455	81,998	10,359	81,998	16,450	2019-02-06 08:11:00	2019-02-06 12:12:00	No	12.63%

SCHARP at FRED HUTCH

AutoVacuum Thresholds

- AV Threshold
  - This is the number of deleted records needed to kick off the AutoVacuum process for this table.
- Last Vacuum, Last Analyze
  - These are the Last Vacuum / AutoVacuum and Last Vacuum Analyze or Last AutoVacuum Analyze
- AV Needed
  - This will show “Yes” if the Deleted Records is more than the AV Threshold
- % Deleted
  - This shows you the percentage of the table that is deleted.



AutoVacuum Thresholds (Notes: Requires ds.autovacuum_thresholds)

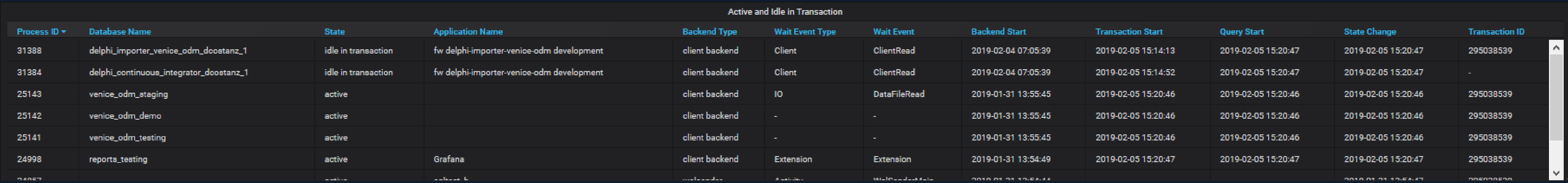
Table Name	Records Inserted	Records Updated	Records Deleted	Live Records	Deleted Records	Records (Est)	AV Threshold	Last Vacuum	Last Analyze	AV Needed ▾	% Deleted
delphi_importer_venice_odm.dataset	1,098,341,327	0	1,075,909,499	22,256,313	9,261,153	22,253,630	4,450,776	2019-02-06 11:13:00	2019-02-06 11:13:00	Yes	41.62%
datamart_ready_area.data_dictionary_variable	487,216	0	362,154	125,062	0	125,062	25,062	2019-02-04 14:59:00	2019-02-04 14:59:00	No	0%
continuous_integrator.dataset_dependency	10,034	0	6,626	2,052	0	2,052	460	2019-02-04 11:55:00	2019-02-04 11:54:00	No	0%
delphi_importer_venice_odm.schema_version	2	0	0	2	0	0	50	-	-	No	0%
datamart_ready_area.dataset	105,667,071	0	78,231,620	27,622,329	0	27,622,328	5,524,516	2019-02-04 15:06:00	2019-02-04 15:06:00	No	0%
continuous_integrator.dataset	2,277	0	0	1,371	0	1,371	324	-	2019-02-01 09:35:00	No	0%
flyway.schema_version	2	0	0	2	0	0	50	-	-	No	0%
datamart_ready_area.data_dictionary_status	12,638	0	9,377	3,261	0	3,261	702	2019-02-04 15:00:00	2019-02-04 15:00:00	No	0%
ds.last_dump	1	0	0	1	0	0	50	-	-	No	0%
datamart_ready_area.data_dictionary	12,638	0	9,377	3,261	0	3,261	702	2019-02-04 15:00:00	2019-02-04 15:00:00	No	0%
webservice.server_authentication	2	0	0	2	0	0	50	-	-	No	0%
delphi_importer_venice_odm.data_dictionary	13,187	0	11,123	2,064	187	2,064	463	2019-02-06 09:11:00	2019-02-06 12:12:00	No	9.06%
delphi_importer_venice_odm.data_dictionary_variable	552,453	0	470,455	81,998	10,359	81,998	16,450	2019-02-06 08:11:00	2019-02-06 12:12:00	No	12.63%

Active and Idle in Transaction

Knowing what might be holding locks to prevent AutoVacuum

Active and Idle in Transaction

- Process ID
    - This is the number of deleted records needed to kick off the AutoVacuum process for this table.
- Database Name
    - These are the Last Vacuum / AutoVacuum and Last Vacuum Analyze or Last AutoVacuum Analyze
- State
    - This will show “Yes” if the Deleted Records is more than the AV Threshold
- Application Name
    - This shows you the percentage of the table that is deleted.
- Backend Type
    - This shows you the percentage of the table that is deleted.

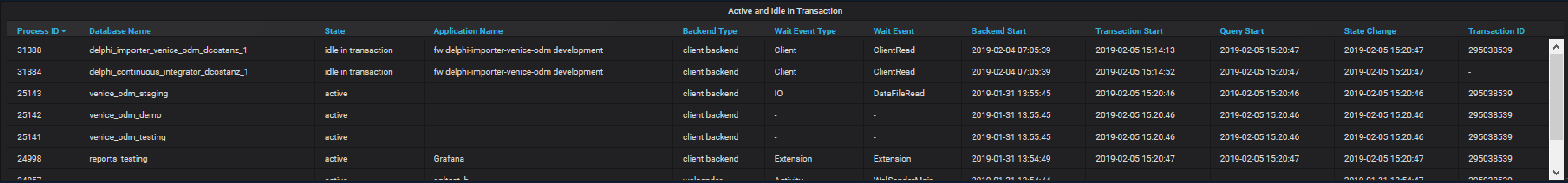


Active and Idle in Transaction

Process ID ▾	Database Name	State	Application Name	Backend Type	Wait Event Type	Wait Event	Backend Start	Transaction Start	Query Start	State Change	Transaction ID
31388	delphi_importer_venice_odm_dcostanz_1	idle in transaction	fw delphi-importer-venice-odm development	client backend	Client	ClientRead	2019-02-04 07:05:39	2019-02-05 15:14:13	2019-02-05 15:20:47	2019-02-05 15:20:47	295038539
31384	delphi_continuous_integrator_dcostanz_1	idle in transaction	fw delphi-importer-venice-odm development	client backend	Client	ClientRead	2019-02-04 07:05:39	2019-02-05 15:14:52	2019-02-05 15:20:47	2019-02-05 15:20:47	-
25143	venice_odm_staging	active		client backend	IO	DataFileRead	2019-01-31 13:55:45	2019-02-05 15:20:46	2019-02-05 15:20:46	2019-02-05 15:20:46	295038539
25142	venice_odm_demo	active		client backend	-	-	2019-01-31 13:55:45	2019-02-05 15:20:46	2019-02-05 15:20:46	2019-02-05 15:20:46	295038539
25141	venice_odm_testing	active		client backend	-	-	2019-01-31 13:55:45	2019-02-05 15:20:46	2019-02-05 15:20:46	2019-02-05 15:20:46	295038539
24998	reports_testing	active	Grafana	client backend	Extension	Extension	2019-01-31 13:54:49	2019-02-05 15:20:47	2019-02-05 15:20:47	2019-02-05 15:20:47	295038539
24857		active	[illegible]	walsender	Activity	WalSenderMain	2019-01-31 13:54:44			2019-01-31 13:54:47	295038539

Active and Idle in Transaction

- Wait Event Type
  - LWLock, Lock, BufferPin, Activity, Extension, Client, IPC, Timeout, IO
- Wait Event
  - To many to cover here, see https://www.postgresql.org/docs/current/monitoring-stats.html#WAIT-EVENT-TABLE
- Backend Start / Transaction Start / Query Start / State Change
  - The Backend Start, is when the connection to the server was established.
  - The Transaction Start is when you did a BEGIN transaction or started a single item transaction.
  - The Query Start is the start of your Query inside of the transaction and will be the same as Transaction Start if running a single item transaction.
  - The State Change is the change in state, such as switching from active to “idle” or “idle in transaction”, allowing you to know how long the transaction or connection has been sitting idle.
- Transaction ID
  - This shows you the percentage of the table that is deleted.



Active and Idle in Transaction

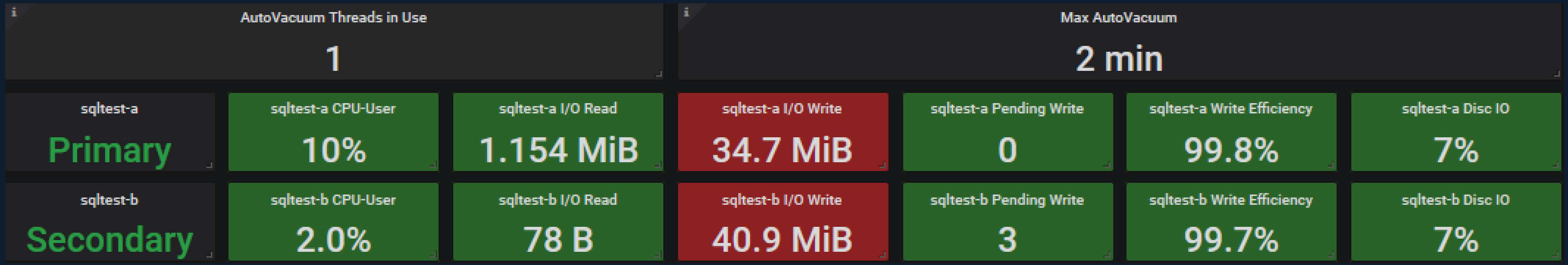
Process ID	Database Name	State	Application Name	Backend Type	Wait Event Type	Wait Event	Backend Start	Transaction Start	Query Start	State Change	Transaction ID
31388	delphi_importer_venice_odm_dcostanz_1	idle in transaction	fw delphi-importer-venice-odm development	client backend	Client	ClientRead	2019-02-04 07:05:39	2019-02-05 15:14:13	2019-02-05 15:20:47	2019-02-05 15:20:47	295038539
31384	delphi_continuous_integrator_dcostanz_1	idle in transaction	fw delphi-importer-venice-odm development	client backend	Client	ClientRead	2019-02-04 07:05:39	2019-02-05 15:14:52	2019-02-05 15:20:47	2019-02-05 15:20:47	-
25143	venice_odm_staging	active		client backend	IO	DataFileRead	2019-01-31 13:55:45	2019-02-05 15:20:46	2019-02-05 15:20:46	2019-02-05 15:20:46	295038539
25142	venice_odm_demo	active		client backend	-	-	2019-01-31 13:55:45	2019-02-05 15:20:46	2019-02-05 15:20:46	2019-02-05 15:20:46	295038539
25141	venice_odm_testing	active		client backend	-	-	2019-01-31 13:55:45	2019-02-05 15:20:46	2019-02-05 15:20:46	2019-02-05 15:20:46	295038539
24998	reports_testing	active	Grafana	client backend	Extension	Extension	2019-01-31 13:54:49	2019-02-05 15:20:47	2019-02-05 15:20:47	2019-02-05 15:20:47	295038539
24857		active	[illegible]	walsender	Activity	WalSenderMain	2019-01-31 13:54:44			2019-01-31 13:54:47	295038539

Current Drive Performance

Monitoring current drive performance

Current Drive Performance

- AutoVacuum Threads in Use
  - This is the number of AutoVacuum’s threads currently running.
- Max AutoVacuum
  - This is the longest running AutoVacuum
- Server-a / Server-b
  - Let’s us know which server is primary vers seconday.
- CPU-User
  - How much cpu the user, such as user postgres, is currently using.
  - servers.virtual.$ServerName-a.aggregation-cpu-average.cpu-user
  - servers.virtual.$ServerName-b.aggregation-cpu-average.cpu-user



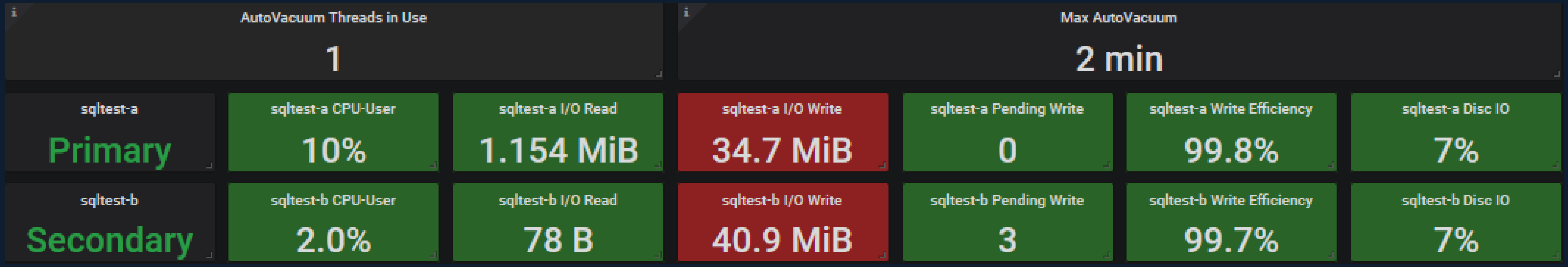
AutoVacuum Threads in Use
1

Max AutoVacuum
2 min

sqltest-a	sqltest-a CPU-User	sqltest-a I/O Read	sqltest-a I/O Write	sqltest-a Pending Write	sqltest-a Write Efficiency	sqltest-a Disc IO
Primary	10%	1.154 MiB	34.7 MiB	0	99.8%	7%
sqltest-b	**sqltest-b CPU-User**	**sqltest-b I/O Read**	**sqltest-b I/O Write**	**sqltest-b Pending Write**	**sqltest-b Write Efficiency**	**sqltest-b Disc IO**
Secondary	2.0%	78 B	40.9 MiB	3	99.7%	7%

Current Drive Performance

- I/O Read / I/O Write
  - Read and Writing in Bytes / KB / MB / GB
  - servers.{virtual,physical}.$ServerName-a.disk-xvdb1.disk_octets.read
  - servers.{virtual,physical}.$ServerName-b.disk-xvdb1.disk_octets.read
  - servers.{virtual,physical}.$ServerName-a.disk-xvdb1.disk_octets.write
  - servers.{virtual,physical}.$ServerName-b.disk-xvdb1.disk_octets.write
- Pending Writes
  - Number of writes that were delayed due to I/O saturation.
  - servers.{virtual,physical}.$ServerName-a.disk-xvdb1.pending_operations
  - servers.{virtual,physical}.$ServerName-b.disk-xvdb1.pending_operations



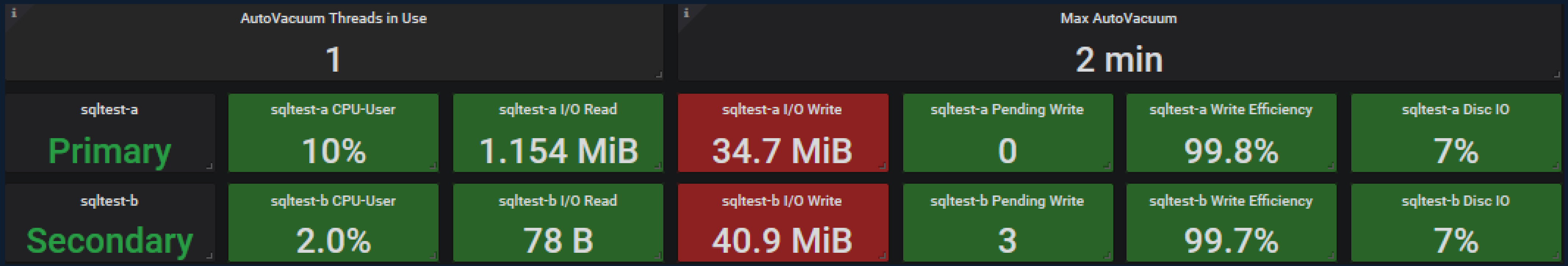
AutoVacuum Threads in Use
1

Max AutoVacuum
2 min

sqltest-a Primary	sqltest-a CPU-User 10%	sqltest-a I/O Read 1.154 MiB	sqltest-a I/O Write 34.7 MiB	sqltest-a Pending Write 0	sqltest-a Write Efficiency 99.8%	sqltest-a Disc IO 7%
sqltest-b Secondary	sqltest-b CPU-User 2.0%	sqltest-b I/O Read 78 B	sqltest-b I/O Write 40.9 MiB	sqltest-b Pending Write 3	sqltest-b Write Efficiency 99.7%	sqltest-b Disc IO 7%

Current Drive Performance

- Write Efficiency
  - Random write is low efficiency, sequential write is high efficiency. We want to see high efficiency.
  - offset(scale(asPercent(servers.{virtual,physical}.$ServerName-a.disk-xvdb1.disk_ops.write,servers.{virtual,physical}.$ServerName-a.disk-xvdb1.disk_octets.write),-1),1)
  - offset(scale(asPercent(servers.{virtual,physical}.$ServerName-b.disk-xvdb1.disk_ops.write,servers.{virtual,physical}.$ServerName-b.disk-xvdb1.disk_octets.write),-1),1)
- Disc I/O
  - Disc I/O usage.
  - scale(servers.{virtual,physical}.$ServerName-a.disk-xvdb1.disk_io_time.io_time,0.1)
  - scale(servers.{virtual,physical}.$ServerName-b.disk-xvdb1.disk_io_time.io_time,0.1)



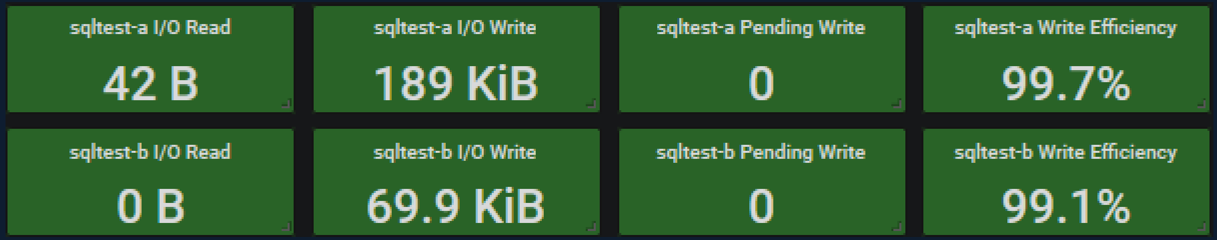
AutoVacuum Threads in Use	Max AutoVacuum
1	2 min

sqltest-a	sqltest-a CPU-User	sqltest-a I/O Read	sqltest-a I/O Write	sqltest-a Pending Write	sqltest-a Write Efficiency	sqltest-a Disc IO
Primary	10%	1.154 MiB	34.7 MiB	0	99.8%	7%

sqltest-b	sqltest-b CPU-User	sqltest-b I/O Read	sqltest-b I/O Write	sqltest-b Pending Write	sqltest-b Write Efficiency	sqltest-b Disc IO
Secondary	2.0%	78 B	40.9 MiB	3	99.7%	7%

Current Drive Performance

- I/O Read / I/O Write Settings
  - RAID Type: RAID 6
  - Drive Capacity: 146GB
  - Single Drive Performance: 6,144 MB/s
  - Single Drive Cost: 40
  - Number of drives per RAID group: 16
  - Number of RAID groups: 1
  - Read operations (%): 0 (100% - Write Efficiency, In this case it would be between 0% and 1%)



sqltest-a I/O Read	sqltest-a I/O Write	sqltest-a Pending Write	sqltest-a Write Efficiency
42 B	189 KiB	0	99.7%
sqltest-b I/O Read	**sqltest-b I/O Write**	**sqltest-b Pending Write**	**sqltest-b Write Efficiency**
0 B	69.9 KiB	0	99.1%

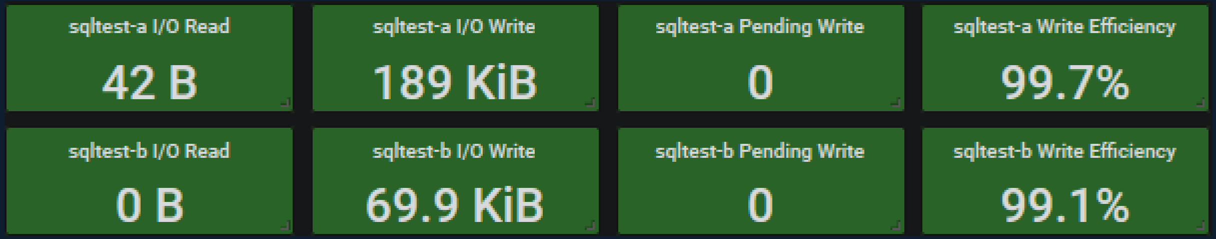
Percent	MB	KB	B
100%	92	93,696	95,944,704
80%	73	74,957	76,755,763
50%	46	46,848	47,972,352

- Online RAID Calculator *(See Page Notes)*

Drive (Type / RPM)	IOPS (4KB block, random)	IOPS (64KB block, random)	MB/s (64KB block, random)	IOPS (512KB block, random)	MB/s (512KB block, random)	MB/s (large block, sequential)
FC / 15K	163-178	151-169	9.7-10.8	97-123	49.7-63.1	73.5-127.5
SAS / 15K	188-203	175-192	11.2-12.3	115-135	58.9-68.9	91.5-126.3
FC / 10K	142-151	130-143	8.3-9.2	80-104	40.9-53.1	58.1-107.2
SAS / 10K	142-151	130-143	8.3-9.2	80-104	40.9-53.1	58.1-107.2
SAS/SATA / 7200	73-79	69-76	4.4-4.9	47-63	24.3-32.1	43.4-97.8
SATA / 5400	57	55	3.5	44	22.6	
SSD	To evaluate SSD RAID performance use the SSD version of the calculator					

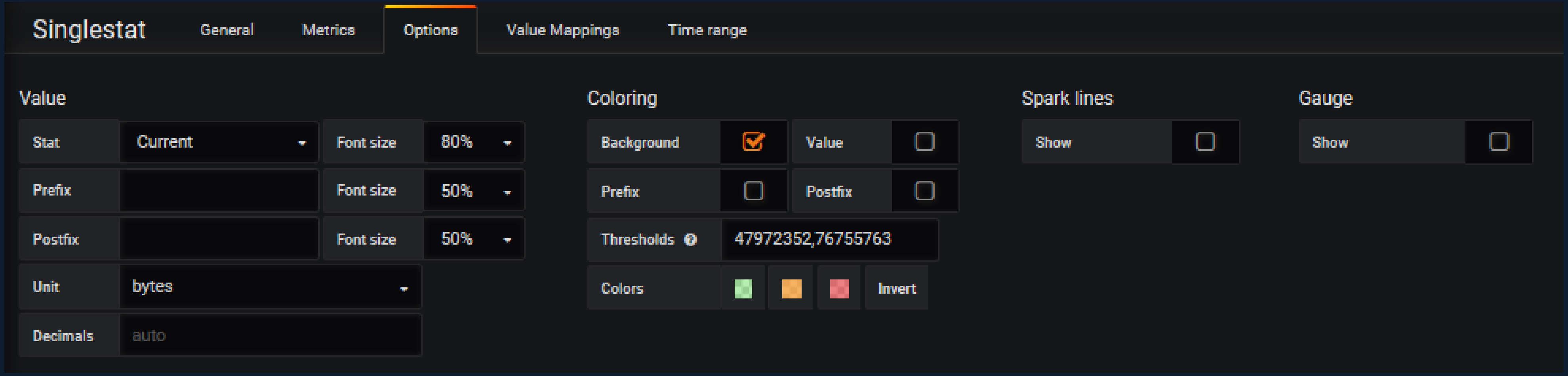
Current Drive Performance

- I/O Read / I/O Write Settings
  - RAID Type: RAID 6
  - Drive Capacity: 146GB
  - Single Drive Performance: 6,144 MB/s
  - Single Drive Cost: 40
  - Number of drives per RAID group: 16
  - Number of RAID groups: 1
  - Read operations (%): 0 (100% - Write Efficiency, In this case it would be between 0% and 1%)



sqltest-a I/O Read 42 B	sqltest-a I/O Write 189 KiB	sqltest-a Pending Write 0	sqltest-a Write Efficiency 99.7%
sqltest-b I/O Read 0 B	sqltest-b I/O Write 69.9 KiB	sqltest-b Pending Write 0	sqltest-b Write Efficiency 99.1%

Percent	MB	KB	B
100%	92	93,696	95,944,704
80%	73	74,957	76,755,763
50%	46	46,848	47,972,352



Value

- Stat: Current — Font size: 80%
- Prefix: — Font size: 50%
- Postfix: — Font size: 50%
- Unit: bytes
- Decimals: auto

Coloring

- Background ☑ — Value □
- Prefix □ — Postfix □
- Thresholds: 47972352,76755763
- Colors 🟩 🟧 🟥 Invert

Spark lines

- Show □

Gauge

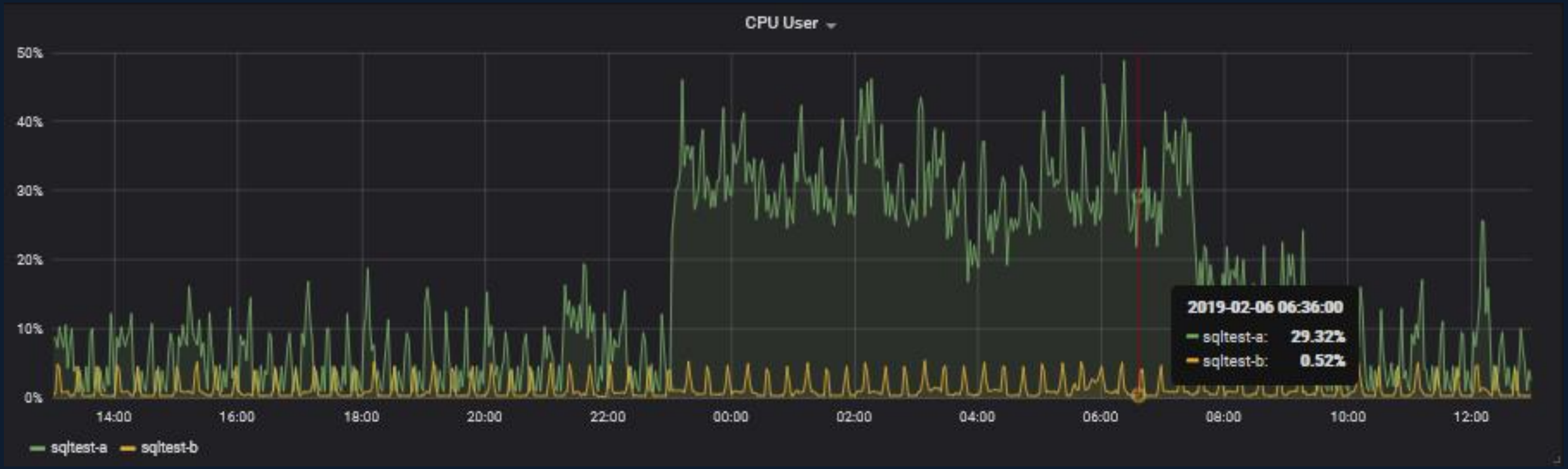
- Show □

Performance History

Monitoring history as thing happen when we are not in front of the monitors.

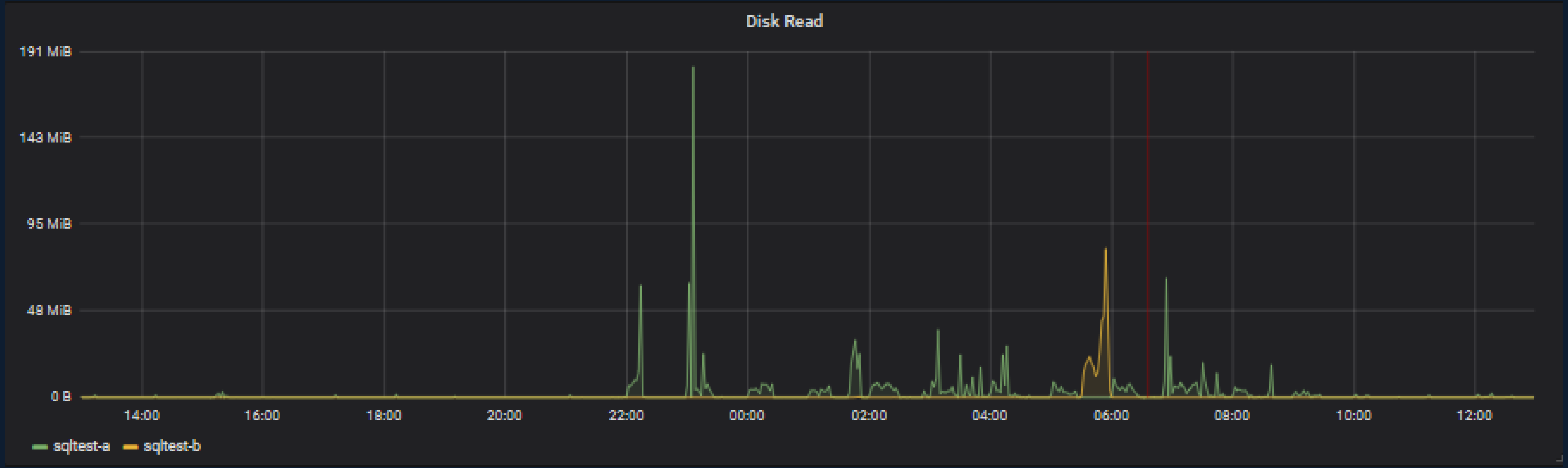
CPU User

- We want to make sure that we are not running at full CPU. I try to keep the server below 50% with a max spike of 80%.
- aliasByNode(servers.{virtual,physical}.$ServerName-a.aggregation-cpu-average.cpu-user,2)
- aliasByNode(servers.{virtual,physical}.$ServerName-b.aggregation-cpu-average.cpu-user,2)



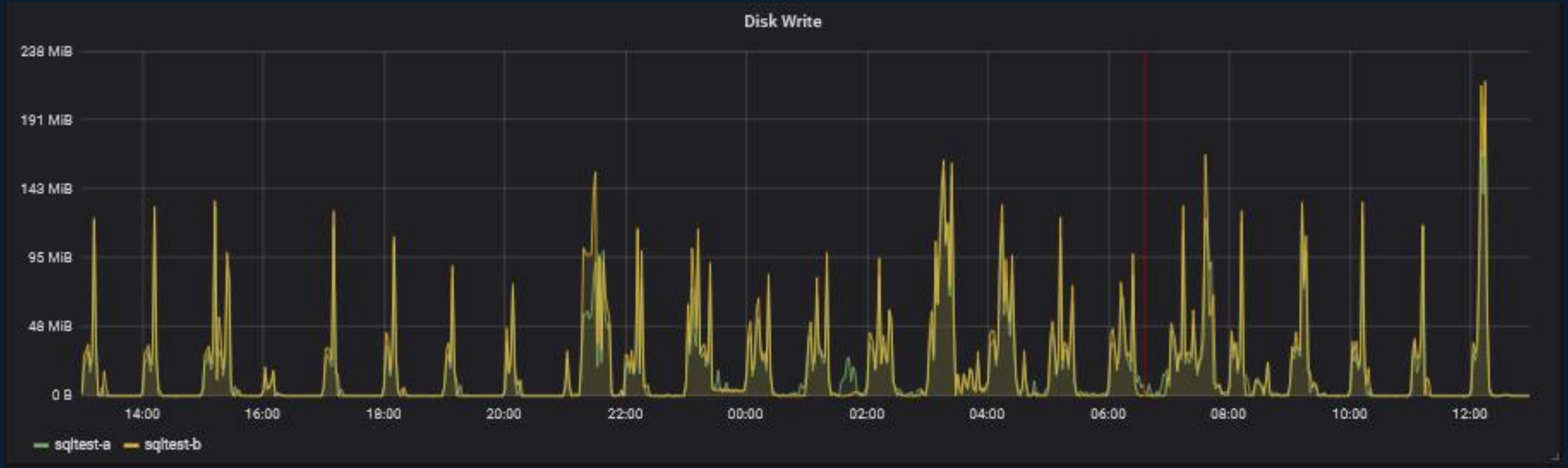
Disk Read

- Short duration spikes are OK, as long as there is no real sustained disk reads.
- aliasByNode(servers.{virtual,physical}.sqltest-a.disk-xvdb1.disk_octets.read,2)
- aliasByNode(servers.{virtual,physical}.sqltest-b.disk-xvdb1.disk_octets.read,2)



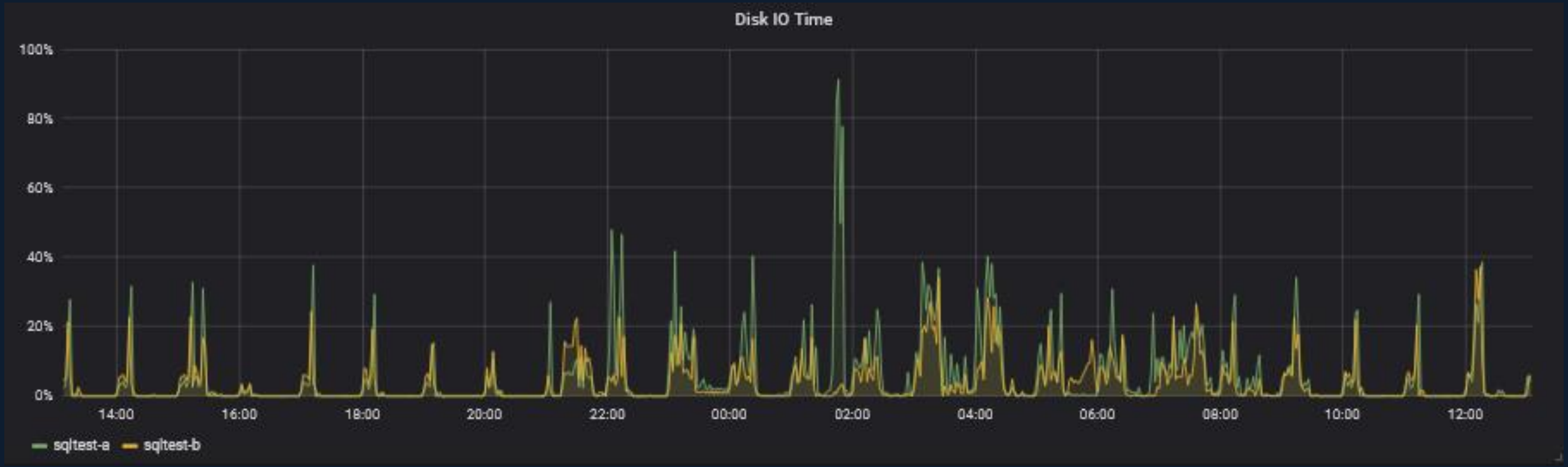
Disk Write

- Short duration spikes are OK, as long as there is no real sustained disk writes causing pending writes to build up.
- aliasByNode(servers.{virtual,physical}.sqltest-a.disk-xvdb1.disk_octets.write,2)
- aliasByNode(servers.{virtual,physical}.sqltest-b.disk-xvdb1.disk_octets.write,2)



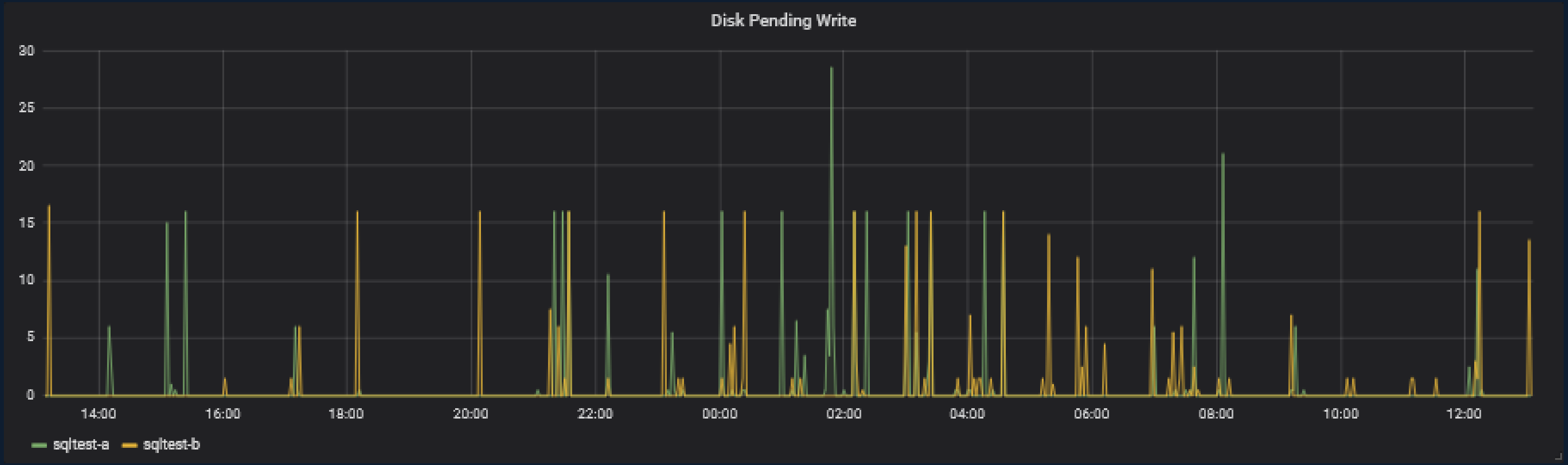
Disk I/O Time

- The vacuum spike to 91% is OK, but I will be reducing it to 80% so that other long queries should not be affected as much by the AutoVacuum.
- aliasByNode(scale(servers.{virtual,physical}.$ServerName-a.disk-xvdb1.disk_io_time.io_time,0.1),2)
- aliasByNode(scale(servers.{virtual,physical}.$ServerName-b.disk-xvdb1.disk_io_time.io_time,0.1),2)



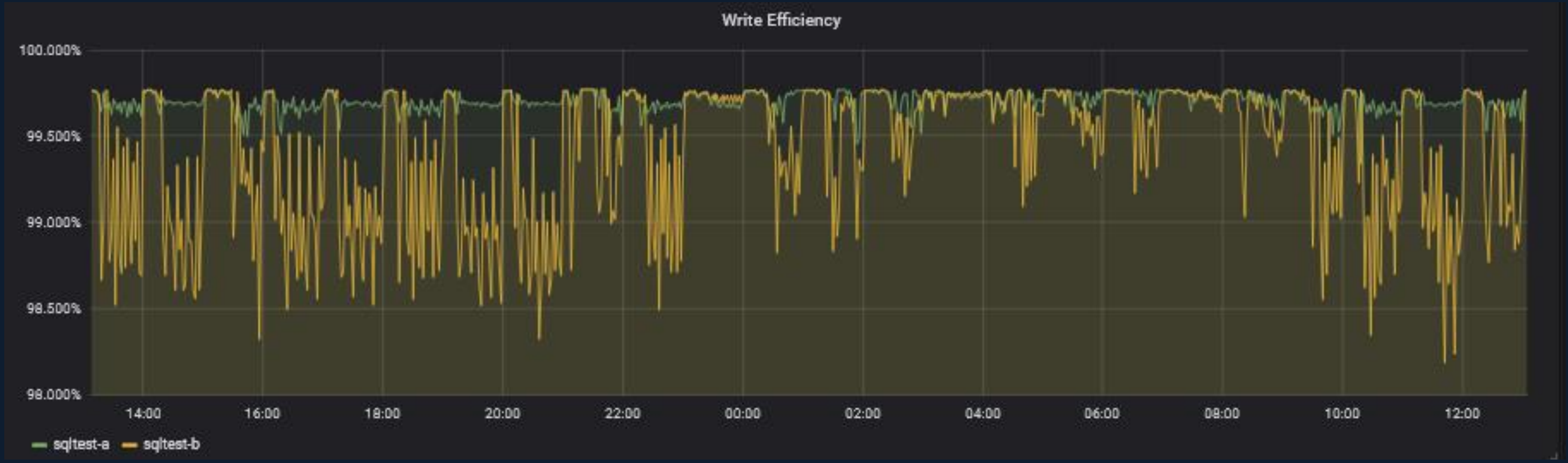
Disk Pending Write

- Short pending writes are OK, we just don't want to see long running pending items.
- aliasByNode(servers.{virtual,physical}.$ServerName-a.disk-xvdb1.pending_operations,2)
- aliasByNode(servers.{virtual,physical}.$ServerName-b.disk-xvdb1.pending_operations,2)



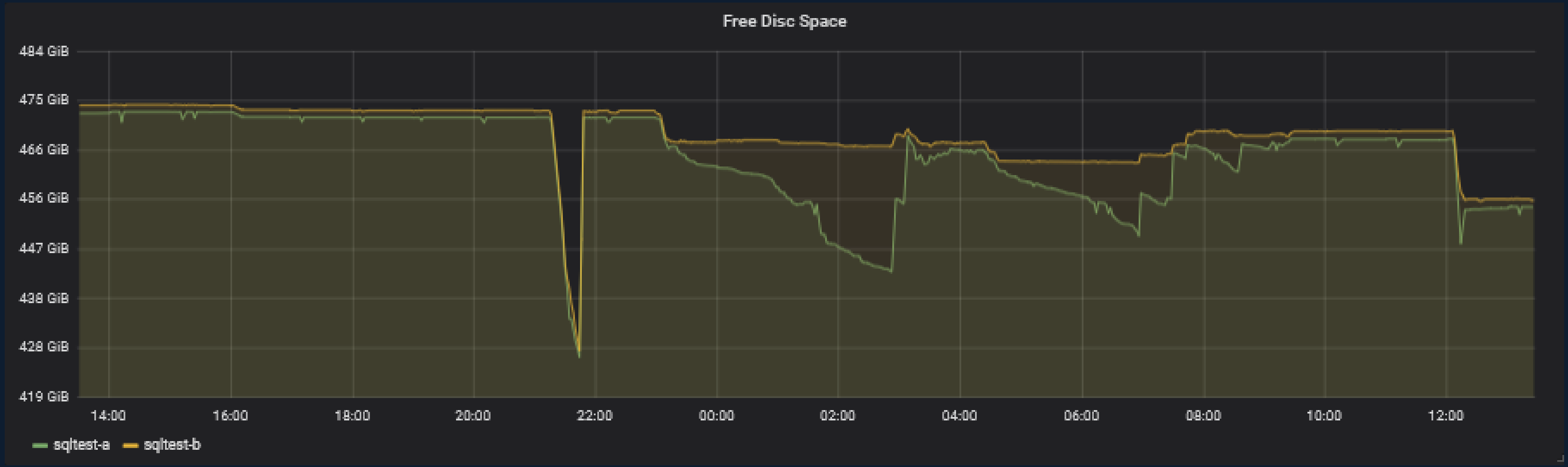
Write Efficiency

- Random write is low efficiency, sequential write is high efficiency. We want to see high efficiency.
- alias(offset(scale(asPercent(servers.{virtual,physical}.$ServerName-a.disk-xvdb1.disk_ops.write,servers.{virtual,physical}.$ServerName-a.disk-xvdb1.disk_octets.write),-1),1),"$ServerName-a")
- alias(offset(scale(asPercent(servers.{virtual,physical}.$ServerName-b.disk-xvdb1.disk_ops.write,servers.{virtual,physical}.$ServerName-b.disk-xvdb1.disk_octets.write),-1),1),"$ServerName-b")



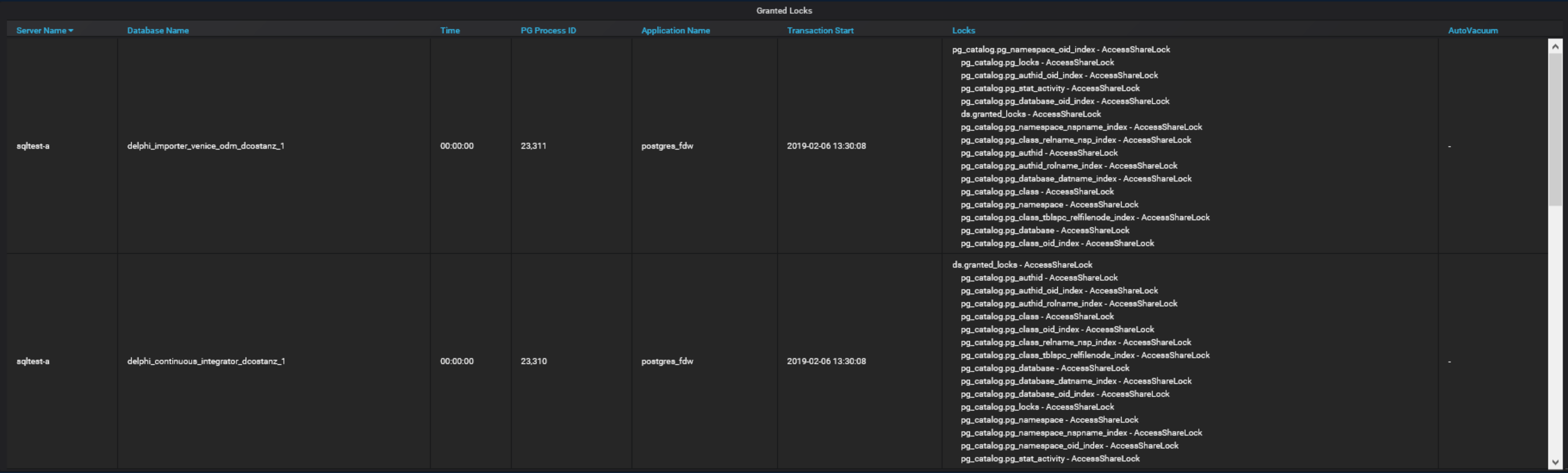
Free Disc Space

- We need to make sure that we don’t run the server out of space due to bloating of the tables.
- aliasByNode(servers.{virtual,physical}.$ServerName-a.df-pgdata_local.df_complex-free,2)
- aliasByNode(servers.{virtual,physical}.$ServerName-b.df-pgdata_local.df_complex-free,2)



Granted Locks

- If a table should be AutoVacuum’ed but is not, it could be because our long running transactions are holding locks on that table. We want to see if there are any Exclusive locks on the tables that should be AutoVacuum’ed.



Granted Locks

Server Name ▾	Database Name	Time	PG Process ID	Application Name	Transaction Start	Locks	AutoVacuum
sqltest-a	delphi_importer_venice_odm_dcostanz_1	00:00:00	23,311	postgres_fdw	2019-02-06 13:30:08	pg_catalog.pg_namespace_oid_index - AccessShareLock pg_catalog.pg_locks - AccessShareLock pg_catalog.pg_authid_oid_index - AccessShareLock pg_catalog.pg_stat_activity - AccessShareLock pg_catalog.pg_database_oid_index - AccessShareLock ds.granted_locks - AccessShareLock pg_catalog.pg_namespace_nspname_index - AccessShareLock pg_catalog.pg_class_relname_nsp_index - AccessShareLock pg_catalog.pg_authid - AccessShareLock pg_catalog.pg_authid_rolname_index - AccessShareLock pg_catalog.pg_database_datname_index - AccessShareLock pg_catalog.pg_class - AccessShareLock pg_catalog.pg_namespace - AccessShareLock pg_catalog.pg_class_tblspc_relfilenode_index - AccessShareLock pg_catalog.pg_database - AccessShareLock pg_catalog.pg_class_oid_index - AccessShareLock	-
sqltest-a	delphi_continuous_integrator_dcostanz_1	00:00:00	23,310	postgres_fdw	2019-02-06 13:30:08	ds.granted_locks - AccessShareLock pg_catalog.pg_authid - AccessShareLock pg_catalog.pg_authid_oid_index - AccessShareLock pg_catalog.pg_authid_rolname_index - AccessShareLock pg_catalog.pg_class - AccessShareLock pg_catalog.pg_class_oid_index - AccessShareLock pg_catalog.pg_class_relname_nsp_index - AccessShareLock pg_catalog.pg_class_tblspc_relfilenode_index - AccessShareLock pg_catalog.pg_database - AccessShareLock pg_catalog.pg_database_datname_index - AccessShareLock pg_catalog.pg_database_oid_index - AccessShareLock pg_catalog.pg_locks - AccessShareLock pg_catalog.pg_namespace - AccessShareLock pg_catalog.pg_namespace_nspname_index - AccessShareLock pg_catalog.pg_namespace_oid_index - AccessShareLock pg_catalog.pg_stat_activity - AccessShareLock	-

SCHARP at FRED HUTCH

Custom Table Settings

Adjusting the AutoVacuum per Table

Custom Table Settings

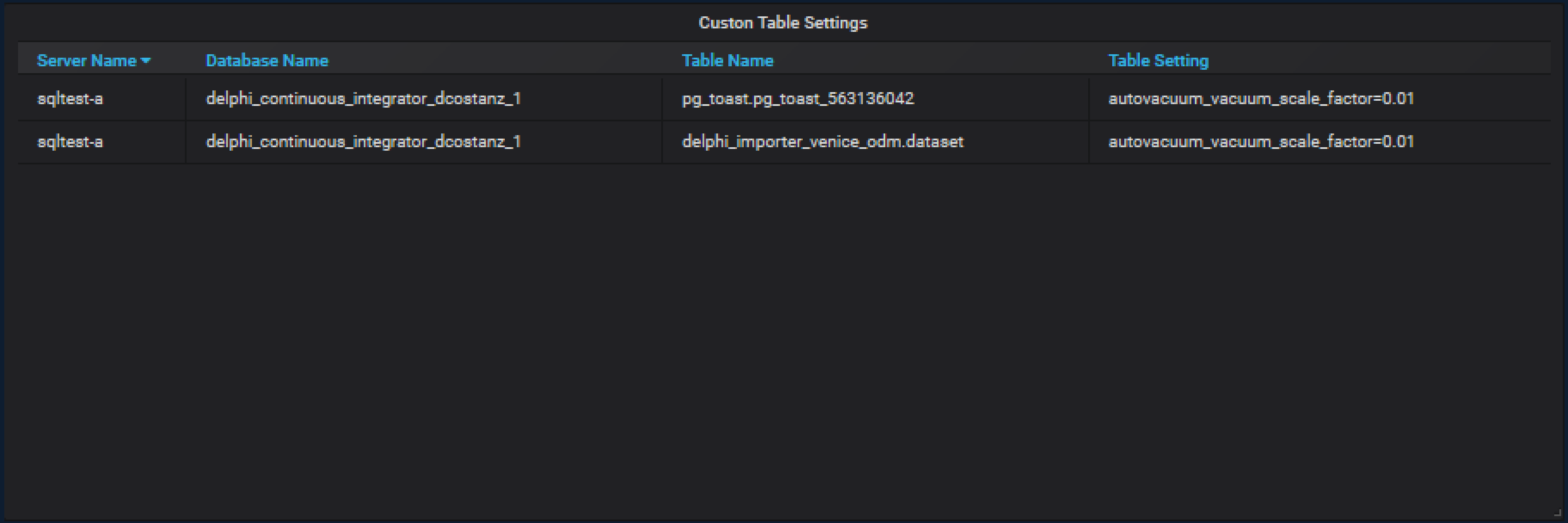
- This query allows us to retrieve all the custom table settings for all tables, excluding the tables located in pg_catalog and the information_schema.

```
SELECT
  current_setting('cluster_name'::text)
    AS cluster_name,
  current_database() AS database_name,
  pn.nspname AS schema_name,
  pc.relname AS table_name,
  quote_ident(pn.nspname::text) ||
    '.'::text ||
    quote_ident(pc.relname::text)
    AS "Table Name",
  unnest(pc.reloptions) AS "Table Setting"
FROM pg_class pc
JOIN pg_namespace pn
  ON pn.oid = pc.relnamespace
WHERE pc.reloptions IS NOT NULL
  AND (pn.nspname <> ALL (ARRAY [
'pg_catalog'::name, 'information_schema':: name ]));
```

Custom Table Settings

- The larger the table, the larger the threshold for AutoVacuum. For very large table we might want to lower this threshold. Instead of doing this for the entire server, we can do it for individual tables.

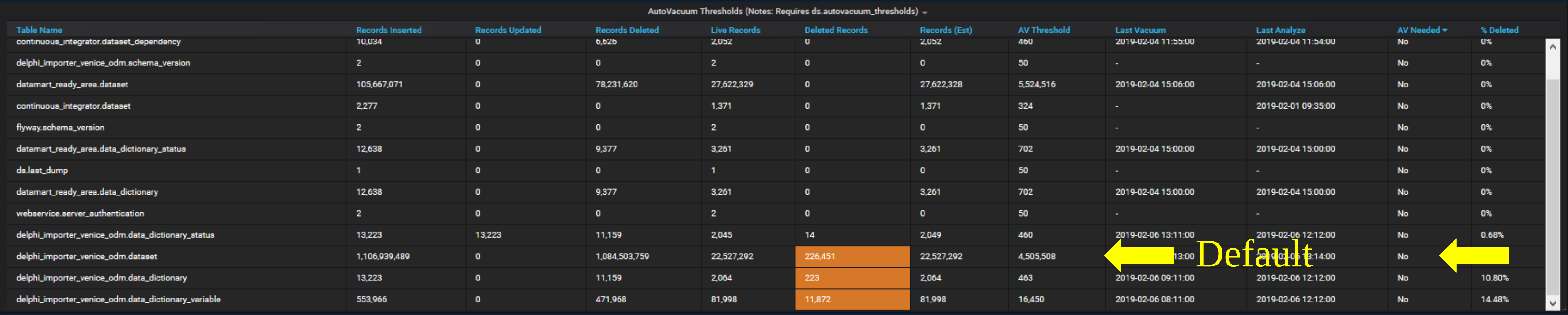
```
ALTER TABLE IF EXISTS ONLY
delphi_importer_venice_odm.dataset
SET
(
    autovacuum_vacuum_scale_factor=0.01,
    toast.autovacuum_vacuum_scale_factor=0.01
);
```



Custon Table Settings

Server Name ▾	Database Name	Table Name	Table Setting
sqltest-a	delphi_continuous_integrator_dcostanz_1	pg_toast.pg_toast_563136042	autovacuum_vacuum_scale_factor=0.01
sqltest-a	delphi_continuous_integrator_dcostanz_1	delphi_importer_venice_odm.dataset	autovacuum_vacuum_scale_factor=0.01

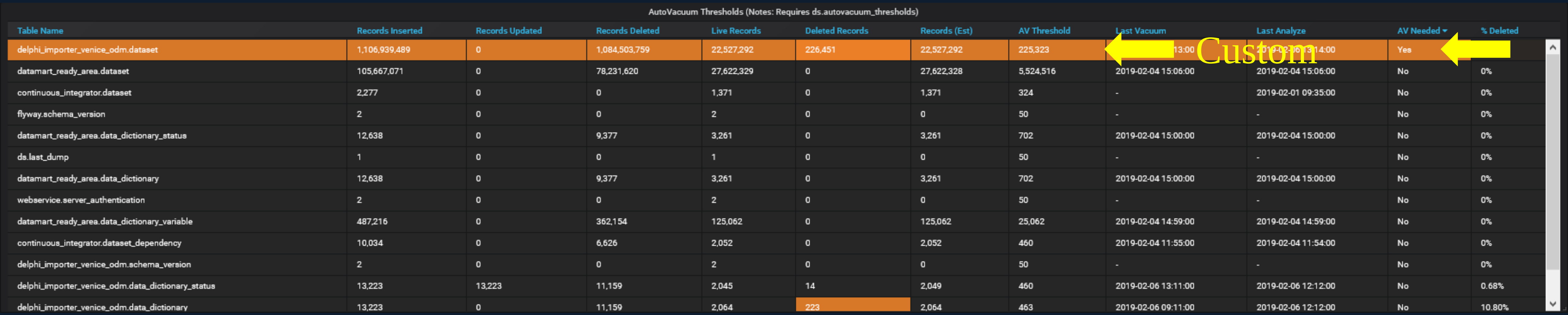
Custom Table Settings



AutoVacuum Thresholds (Notes: Requires ds.autovacuum_thresholds)

Table Name	Records Inserted	Records Updated	Records Deleted	Live Records	Deleted Records	Records (Est)	AV Threshold	Last Vacuum	Last Analyze	AV Needed	% Deleted
continuous_integrator.dataset_dependency	10,034	0	6,626	2,052	0	2,052	460	2019-02-04 11:55:00	2019-02-04 11:54:00	No	0%
delphi_importer_venice_odm.schema_version	2	0	0	2	0	0	50	-	-	No	0%
datamart_ready_area.dataset	105,667,071	0	78,231,620	27,622,329	0	27,622,328	5,524,516	2019-02-04 15:06:00	2019-02-04 15:06:00	No	0%
continuous_integrator.dataset	2,277	0	0	1,371	0	1,371	324	-	2019-02-01 09:35:00	No	0%
flyway.schema_version	2	0	0	2	0	0	50	-	-	No	0%
datamart_ready_area.data_dictionary_status	12,638	0	9,377	3,261	0	3,261	702	2019-02-04 15:00:00	2019-02-04 15:00:00	No	0%
ds.last_dump	1	0	0	1	0	0	50	-	-	No	0%
datamart_ready_area.data_dictionary	12,638	0	9,377	3,261	0	3,261	702	2019-02-04 15:00:00	2019-02-04 15:00:00	No	0%
webservice.server_authentication	2	0	0	2	0	0	50	-	-	No	0%
delphi_importer_venice_odm.data_dictionary_status	13,223	13,223	11,159	2,045	14	2,049	460	2019-02-06 13:11:00	2019-02-06 12:12:00	No	0.68%
delphi_importer_venice_odm.dataset	1,106,939,489	0	1,084,503,759	22,527,292	226,451	22,527,292	4,505,508	[illegible]13:00	2019-02-06 13:14:00	No	[illegible]
delphi_importer_venice_odm.data_dictionary	13,223	0	11,159	2,064	223	2,064	463	2019-02-06 09:11:00	2019-02-06 12:12:00	No	10.80%
delphi_importer_venice_odm.data_dictionary_variable	553,966	0	471,968	81,998	11,872	81,998	16,450	2019-02-06 08:11:00	2019-02-06 12:12:00	No	14.48%

Default



AutoVacuum Thresholds (Notes: Requires ds.autovacuum_thresholds)

Table Name	Records Inserted	Records Updated	Records Deleted	Live Records	Deleted Records	Records (Est)	AV Threshold	Last Vacuum	Last Analyze	AV Needed	% Deleted
delphi_importer_venice_odm.dataset	1,106,939,489	0	1,084,503,759	22,527,292	226,451	22,527,292	225,323	[illegible]13:00	2019-02-06 13:14:00	Yes	[illegible]
datamart_ready_area.dataset	105,667,071	0	78,231,620	27,622,329	0	27,622,328	5,524,516	2019-02-04 15:06:00	2019-02-04 15:06:00	No	0%
continuous_integrator.dataset	2,277	0	0	1,371	0	1,371	324	-	2019-02-01 09:35:00	No	0%
flyway.schema_version	2	0	0	2	0	0	50	-	-	No	0%
datamart_ready_area.data_dictionary_status	12,638	0	9,377	3,261	0	3,261	702	2019-02-04 15:00:00	2019-02-04 15:00:00	No	0%
ds.last_dump	1	0	0	1	0	0	50	-	-	No	0%
datamart_ready_area.data_dictionary	12,638	0	9,377	3,261	0	3,261	702	2019-02-04 15:00:00	2019-02-04 15:00:00	No	0%
webservice.server_authentication	2	0	0	2	0	0	50	-	-	No	0%
datamart_ready_area.data_dictionary_variable	487,216	0	362,154	125,062	0	125,062	25,062	2019-02-04 14:59:00	2019-02-04 14:59:00	No	0%
continuous_integrator.dataset_dependency	10,034	0	6,626	2,052	0	2,052	460	2019-02-04 11:55:00	2019-02-04 11:54:00	No	0%
delphi_importer_venice_odm.schema_version	2	0	0	2	0	0	50	-	-	No	0%
delphi_importer_venice_odm.data_dictionary_status	13,223	13,223	11,159	2,045	14	2,049	460	2019-02-06 13:11:00	2019-02-06 12:12:00	No	0.68%
delphi_importer_venice_odm.data_dictionary	13,223	0	11,159	2,064	223	2,064	463	2019-02-06 09:11:00	2019-02-06 12:12:00	No	10.80%

Custom

SCHARP at FRED HUTCH

THANK YOU

SCHARP
at FRED HUTCH