SQLite Analyzer Documentation

Release 0.1

Santiago Gil

Jan 25, 2018

Table of Contents

1 Introduction **1**

2 API Reference **3**
- 2.1 StorageAnalyzer . . . 3
- 2.2 `SQLiteHeader` class . . . 8

3 Examples **11**
- 3.1 Human-readable header . . . 11
- 3.2 Making a pie chart of space usage with **pyplot** . . . 12

4 License **15**

CHAPTER 1

Introduction

This Python 3 package currently provides two classes:

StorageAnalyzer Allows to analyze the storage usage statistics of an SQLite 3 database

and,

SQLiteHeader Allows easy access to the values stored in the first 100 bytes of an SQLite file

CHAPTER 2

API Reference

2.1 StorageAnalyzer

2.1.1 `StorageAnalyzer` class

class sqliteanalyzer.**StorageAnalyzer**(*db_path: str*)

Extracts storage-space usage statistics from an SQLite3 database.

It uses as a starting point the metrics provided by the `DBSTAT` virtual table.

Parameters **db_path** – path to an SQLite3 database file

Note: SQLite3 must have been compiled with the `-DSQLITE_ENABLE_DBSTAT_VTAB` flag enabled.

References

https://www.sqlite.org/dbstat.html

item_count() → int

Number of rows defined in table `SQLITE_MASTER`.

Returns `SELECT COUNT(*) from SQLITE_MASTER`

file_size() → int

Physical size of the database in bytes, as reported by `os.stat()`.

Returns Size of the database [bytes]

logical_file_size() → int

Number of bytes that the database should take given the size of a page and the number of pages it contains.

If there is no compression, then this value is equal to the physical file size (*`file_size()`*).

Returns Logical size of the database [bytes]

page_size() → int
Size in bytes of the database pages.

Returns `PRAGMA page_size` [bytes]

page_count() → int
Number of reported pages in the database.

Returns `PRAGMA page_count`

calculated_free_pages() → int
Number of free pages.

Returns *page_count()* - *in_use_pages()* - *autovacuum_page_count()*

calculated_page_count() → int
Number of calculated pages in the database.

Returns The sum of pages in use, pages in the freelist and pages in the autovacuum pointer map.

page_count() + *in_use_pages()* + *autovacuum_page_count()*

freelist_count() → int
Number of pages in the freelist.

Those are unused pages in the database.

Returns `PRAGMA freelist_count`

pages() → [<class 'sqliteanalyzer.types.Page'>]
Returns the definition for all pages in the database.

It is a dump of the `DBSTAT` virtual table.

Reference: https://www.sqlite.org/dbstat.html

Returns a list of *Page* objects

in_use_pages() → int
Number of pages currently in use.

Returns `leaf_pages` + `internal_pages` + `overflow_pages`

in_use_percent() → float
Percentage of pages from the total that are currently in use.

Returns % of pages of the DB that are currently in use

tables() → [<class 'str'>]
Names of the tables defined in the database.

Returns tables in the database

indices() → [<class 'sqliteanalyzer.types.Index'>]
Returns the indices defined in the database.

Returns a list of *Index*

index_list(*table: str*) → [<class 'sqliteanalyzer.types.IndexListEntry'>]
Given a table, returns its entries in `PRAGMA index_list`.

Returns A list of *IndexListEntry* namedtuples.

References

https://sqlite.org/pragma.html#pragma_index_list

ntable() → int
Number of tables in the database.

nindex() → int
Number of indices in the database.

nautoindex() → int
Number of automatically-created indices in the database.

nmanindex() → int
Number of manually-created indices in the database.

payload_size() → int
Space in bytes used by the user's payload.

It does not include the space used by the `sqlite_master` table nor any indices.

is_compressed() → bool
Returns whether the database file is compressed.

autovacuum_page_count() → int
The number of pages used by the *auto-vacuum* pointer map.

table_space_usage() → {}
Space used by each table in the database.

Returns A dictionary from table names to page counts.

table_page_count(*name: str*, *exclude_indices=False*) → int
Number of pages that the table is currently using.

If `exclude_indices == True`, then it does not count those pages taken by indices that might point to that table.

Parameters

- **name** – name of the table
- **exclude_indices** – whether to avoid counting pages used
- **indices on the table.** (*by*) –

index_page_count(*name: str*) → int
Number of pages that the index is currently using.

Parameters **name** – name of the index

Returns number of pages

index_stats(*name: str*) → sqliteanalyzer.storageanalyzer.StorageMetrics
Returns statistics for the index.

Parameters **name** – name of the index

Returns a *StorageMetrics* object

table_stats(*name: str*, *exclude_indices=False*) → sqliteanalyzer.storageanalyzer.StorageMetrics
Returns statistics for a table.

The value of the optional parameter `exclude_indices`, determines whether indices are considered part of the actual table or not.

Parameters **name** – name of the table

Returns a *StorageMetrics* object

global_stats(*exclude_indices=False*) → sqliteanalyzer.storageanalyzer.StorageMetrics
Storage metrics for all tables and/or indices in the database

The value of the optional parameter `exclude_indices` determines whether indices are considered.

Parameters **exclude_indices** – bool: if False, space used by indices is not considered.

Returns a StorageMetrics object

indices_stats() → sqliteanalyzer.storageanalyzer.StorageMetrics
Return metadata about the indices in the database.

Raises `ValueError` – If no indices exist

is_without_rowid(*table: str*) → bool
Returns whether the given table is a `WITHOUT ROWID` table.

Parameters **table** – name of the table

References

https://sqlite.org/withoutrowid.html

stat_db_dump() → [<class 'str'>]
Returns a dump of the DB containing the stats.

Returns list of lines containing an SQL dump of the stat database.

2.1.2 Return types

class sqliteanalyzer.**Index**(*name*, *table*)

name
str – Name of the index

table
str – Table to which the index points

class sqliteanalyzer.**IndexListEntry**(*seq*, *name*, *unique*, *origin*, *partial*)

name
str – name of the index

origin
str – How was the index created. `c` if it was created by a `CREATE_INDEX` statement, `u` if created by a `UNIQUE` constraint, or `pk` if created by a `PRIMARY_KEY` constraint

partial
bool – whether the index covers only a subset of rows of a table

seq
int – internal sequence number of the index

unique
bool – whether index is `UNIQUE`

class sqliteanalyzer.**Page**(*name*, *path*, *pageno*, *pagetype*, *ncell*, *payload*, *unused*, *mx_payload*, *pgoffset*, *pgsize*)

mx_payload
Largest payload size of all cells on the page

name
Name of table or index

ncell
Cells on page (0 for overflow)

pageno
Page number

pagetype
`'internal'`, `'leaf'` or `'overflow'`

path
Path from the root to the page

payload
Bytes of payload on the page

pgoffset
Offset of the page in the file

pgsize
Size of the page

unused
Bytes of unused space on the page

class sqliteanalyzer.**StorageMetrics**(**args*, ***kwargs*)
Storage metrics for a given database object.

It contains the following keys:

- `'nentry'`
- `'payload'`
- `'ovfl_payload'`
- `'mx_payload'`
- `'ovfl_cnt'`
- `'leaf_pages'`
- `'int_pages'`
- `'ovfl_pages'`
- `'leaf_unused'`
- `'int_unused'`
- `'ovfl_unused'`
- `'gap_cnt'`
- `'compressed_size'`
- `'depth'`
- `'cnt'`

- 'total_pages'
- 'total_pages_percent'
- 'storage'
- 'is_compressed'
- 'compressed_overhead'
- 'payload_percent'
- 'total_unused'
- 'total_metadata'
- 'metadata_percent'
- 'average_payload'
- 'average_unused'
- 'average_metadata'
- 'ovfl_percent'
- 'fragmentation'
- 'int_unused_percent'
- 'ovfl_unused_percent'
- 'leaf_unused_percent'
- 'total_unused_percent

2.2 SQLiteHeader class

class sqliteanalyzer.**SQLiteHeader**(*db_path: str*)

Read an SQLite database file and extract the information contained in its header (first HEADER_SIZE_BYTES = 100 bytes).

Reference: https://sqlite.org/fileformat.html#the_database_header

header_string = None
str – header string ('SQLite format 3\x00')

page_size = None
int – page size in bytes

format_read_version = None
int – SQLite version used in the last read

format_write_version = None
int – SQLite version used in the last write

reserved_space = None
int – Unused bytes reserved at the end of a page

max_embedded_payload = None
int – Maximum embedded payload fraction (must be 64)

min_embedded_payload = None
int – Minimum embedded payload fraction (must be 32)

leaf_payload = None
int – Leaf payload fraction. Must be 32.

change_counter = None
int – File change counter

page_count = None
int – Size of the DB in pages

freelist_start = None
int – Page# of the first freelist trunk page

freelist_count = None
int – Total number of freelist pages

schema_cookie = None
int – Schema cookie

schema_format = None
int – Schema format number

page_cache_size = None
int – Default page cache size

largest_root_page = None
int – Page# of the largest root b-tree page when in autovacuum or incremental vacuum modes

encoding_value = None
str – Text encoding `'UTF-8'`, `'UTF-16le'`, `'UTF-16be'` or `None` if not valid

user_version = None
int – User version as set by the `user_version` pragma

incremental_vacuum_mode = None
bool – Incremental vacuum mode enabled

application_id = None
int – Application ID as set by the `application_id` pragma

reserved = None
int – Bytes reserved by SQLite for expansion. (Must be 0.)

version_valid_for = None
int – `version-valid-for` number

sqlite_version_number = None
int – SQLite version number

header_seems_valid() → bool
Returns whether the values in fields that have constraints do respect them (i.e. the header appears to be valid).

CHAPTER 3

Examples

3.1 Human-readable header

```
>>> from sqliteanalyzer import SQLiteHeader
>>> h = SQLiteHeader('test.db')
>>> print(h)
Header seems valid? True
Header string: SQLite format 3
Page size: 4096
Format read version: 1
Format write version: 1
Reserved space: 0
Max. embeded payload: 64
Min. embeded payload: 32
Leaf payload: 32
Change counter: 4
Page count: 146754
Freelist start page: 0
Freelist size: 0
Schema cookie: 2
Schema format: 4
Page cache size: 0
Largest b-tree-root page #: 0
Text encoding: UTF-8
User version: 0
Incremental vacuum mode: False
Application id.: 0
Version valid for: 4
SQLite version number: 3021000
```

3.2 Making a pie chart of space usage with pyplot

```
import matplotlib.pyplot as plt
from sqliteanalyzer import StorageAnalyzer

path_to_db = '/home/yourusername/.mozilla/firefox/.../places.sqlite'
a = StorageAnalyzer(path_to_db)

tables = a.tables()
page_usages = [a.table_page_count(table) for table in tables]

plt.pie(page_usages, labels=tables, autopct='%1.1f%%')
plt.show()
```

[1] https://matplotlib.org/api/_as_gen/matplotlib.pyplot.pie.html#matplotlib.pyplot.pie
[2] https://developer.mozilla.org/en-US/docs/Mozilla/Tech/Places/Database

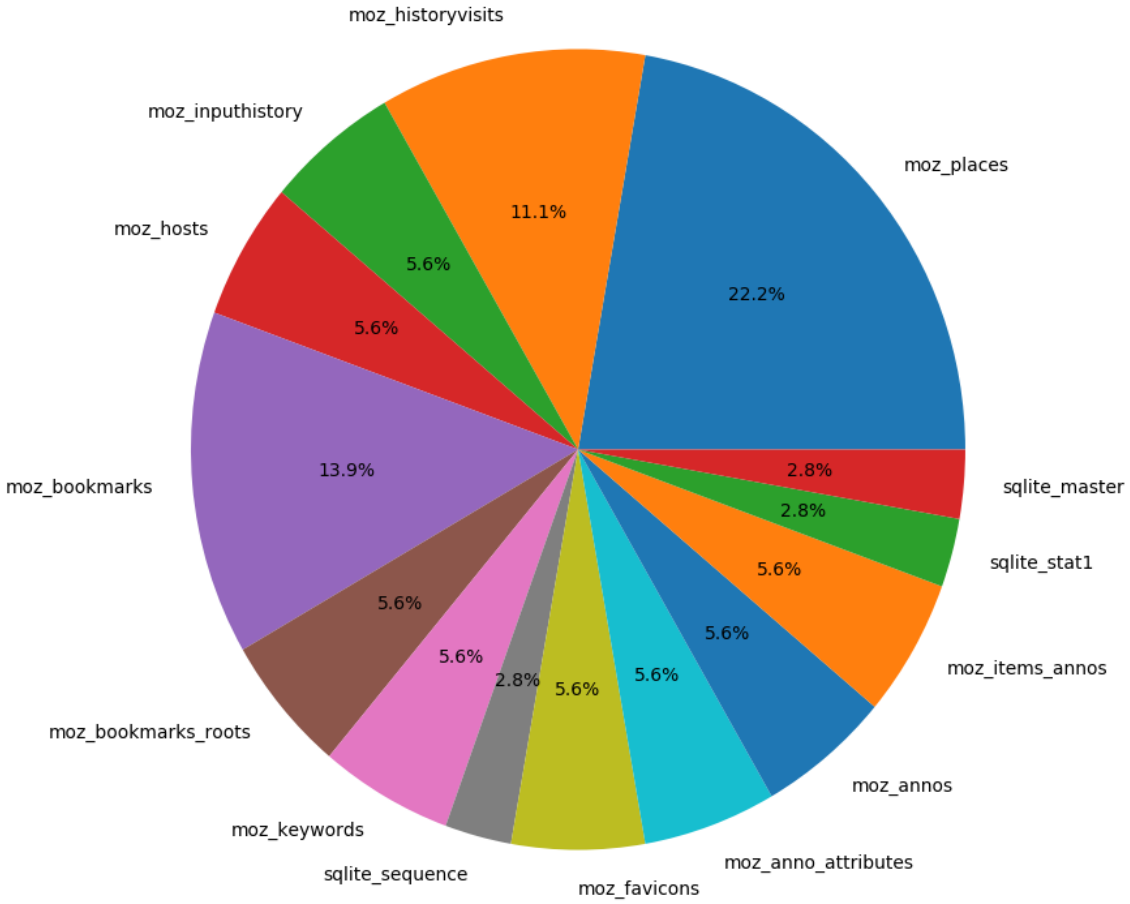

Fig. 3.1: Pie chart[1] showing the portion of storage used by each table in a Firefox's Places database, `places.sqlite`[2].

CHAPTER 4

License

Index

A

application_id (sqliteanalyzer.SQLiteHeader attribute), 9
autovacuum_page_count() (sqliteanalyzer.StorageAnalyzer method), 5

C

calculated_free_pages() (sqliteanalyzer.StorageAnalyzer method), 4
calculated_page_count() (sqliteanalyzer.StorageAnalyzer method), 4
change_counter (sqliteanalyzer.SQLiteHeader attribute), 9

E

encoding_value (sqliteanalyzer.SQLiteHeader attribute), 9

F

file_size() (sqliteanalyzer.StorageAnalyzer method), 3
format_read_version (sqliteanalyzer.SQLiteHeader attribute), 8
format_write_version (sqliteanalyzer.SQLiteHeader attribute), 8
freelist_count (sqliteanalyzer.SQLiteHeader attribute), 9
freelist_count() (sqliteanalyzer.StorageAnalyzer method), 4
freelist_start (sqliteanalyzer.SQLiteHeader attribute), 9

G

global_stats() (sqliteanalyzer.StorageAnalyzer method), 6

H

header_seems_valid() (sqliteanalyzer.SQLiteHeader method), 9
header_string (sqliteanalyzer.SQLiteHeader attribute), 8

I

in_use_pages() (sqliteanalyzer.StorageAnalyzer method), 4
in_use_percent() (sqliteanalyzer.StorageAnalyzer method), 4
incremental_vacuum_mode (sqliteanalyzer.SQLiteHeader attribute), 9
Index (class in sqliteanalyzer), 6
index_list() (sqliteanalyzer.StorageAnalyzer method), 4
index_page_count() (sqliteanalyzer.StorageAnalyzer method), 5
index_stats() (sqliteanalyzer.StorageAnalyzer method), 5
IndexListEntry (class in sqliteanalyzer), 6
indices() (sqliteanalyzer.StorageAnalyzer method), 4
indices_stats() (sqliteanalyzer.StorageAnalyzer method), 6
is_compressed() (sqliteanalyzer.StorageAnalyzer method), 5
is_without_rowid() (sqliteanalyzer.StorageAnalyzer method), 6
item_count() (sqliteanalyzer.StorageAnalyzer method), 3

L

largest_root_page (sqliteanalyzer.SQLiteHeader attribute), 9
leaf_payload (sqliteanalyzer.SQLiteHeader attribute), 8
logical_file_size() (sqliteanalyzer.StorageAnalyzer method), 3

M

max_embedded_payload (sqliteanalyzer.SQLiteHeader attribute), 8
min_embedded_payload (sqliteanalyzer.SQLiteHeader attribute), 8
mx_payload (sqliteanalyzer.Page attribute), 7

N

name (sqliteanalyzer.Index attribute), 6
name (sqliteanalyzer.IndexListEntry attribute), 6
name (sqliteanalyzer.Page attribute), 7
nautoindex() (sqliteanalyzer.StorageAnalyzer method), 5
ncell (sqliteanalyzer.Page attribute), 7

nindex() (sqliteanalyzer.StorageAnalyzer method), 5
nmanindex() (sqliteanalyzer.StorageAnalyzer method), 5
ntable() (sqliteanalyzer.StorageAnalyzer method), 5

O

origin (sqliteanalyzer.IndexListEntry attribute), 6

P

Page (class in sqliteanalyzer), 6
page_cache_size (sqliteanalyzer.SQLiteHeader attribute), 9
page_count (sqliteanalyzer.SQLiteHeader attribute), 9
page_count() (sqliteanalyzer.StorageAnalyzer method), 4
page_size (sqliteanalyzer.SQLiteHeader attribute), 8
page_size() (sqliteanalyzer.StorageAnalyzer method), 3
pageno (sqliteanalyzer.Page attribute), 7
pages() (sqliteanalyzer.StorageAnalyzer method), 4
pagetype (sqliteanalyzer.Page attribute), 7
partial (sqliteanalyzer.IndexListEntry attribute), 6
path (sqliteanalyzer.Page attribute), 7
payload (sqliteanalyzer.Page attribute), 7
payload_size() (sqliteanalyzer.StorageAnalyzer method), 5
pgoffset (sqliteanalyzer.Page attribute), 7
pgsize (sqliteanalyzer.Page attribute), 7

R

reserved (sqliteanalyzer.SQLiteHeader attribute), 9
reserved_space (sqliteanalyzer.SQLiteHeader attribute), 8

S

schema_cookie (sqliteanalyzer.SQLiteHeader attribute), 9
schema_format (sqliteanalyzer.SQLiteHeader attribute), 9
seq (sqliteanalyzer.IndexListEntry attribute), 6
sqlite_version_number (sqliteanalyzer.SQLiteHeader attribute), 9
SQLiteHeader (class in sqliteanalyzer), 8
stat_db_dump() (sqliteanalyzer.StorageAnalyzer method), 6
StorageAnalyzer (class in sqliteanalyzer), 3
StorageMetrics (class in sqliteanalyzer), 7

T

table (sqliteanalyzer.Index attribute), 6
table_page_count() (sqliteanalyzer.StorageAnalyzer method), 5
table_space_usage() (sqliteanalyzer.StorageAnalyzer method), 5
table_stats() (sqliteanalyzer.StorageAnalyzer method), 5
tables() (sqliteanalyzer.StorageAnalyzer method), 4

U

unique (sqliteanalyzer.IndexListEntry attribute), 6
unused (sqliteanalyzer.Page attribute), 7
user_version (sqliteanalyzer.SQLiteHeader attribute), 9

V

version_valid_for (sqliteanalyzer.SQLiteHeader attribute), 9