

Принцип работы базы SkyDNS Octo

В этом документе описан формат базы категорированных интернет-ресурсов СкайдНС Octo и способы доступа к ней.

- База представляет из себя файл базы данных в формате SQLite с несколькими таблицами, подробнее в [Описание базы данных и таблиц](#).
- Для запроса в базу данных исходный URL требуется представить в каноничной форме по специальному алгоритму.
- Алгоритм описан в [Преобразование URL в каноничную форму](#).
- Для прямого поиска по базе URL требуется преобразование с помощью хеширования. Способ хеширования и прочие преобразования описаны в [Хеширование URL](#).
- В [Запросы и интерпретация результатов поиска](#) приведены инструкции по интерпретации результатов поиска.

Обратите внимание, что для упрощения и ускорения интеграции базы в ваш продукт мы настоятельно рекомендуем использовать доступ через библиотеку, а не прямой доступ к базе.

Формат и режим работы базы

Файл базы данных записан в формате SQLite. SQLite это компактная встраиваемая реляционная база данных. Слово «встраиваемый» (embedded) означает, что SQLite не использует парадигму клиент-сервер, движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит базу данных в единственном файле базы данных.

База SQLite может работать в двух режимах: rollback и wal. Разница в том, как записываются изменения в базу. В первом случае файл нельзя изменить, когда его кто-то читает или изменяет в данный момент. Второй режим позволяет одновременно читать и изменять файл базы, но при этом создает рядом с базой служебные файлы.

При использовании Модуль `skydns_url2cat` на языке программирования Python база работает в режиме wal.

Для работы нужны права на запись даже для тех пользователей, которые будут читать базу. При копировании базы на устройство только для чтения необходимо перевести базу в режим rollback. Как это сделать написано в документации по [SQLite3](#).

Описание базы данных и таблиц

База данных содержит две таблицы:

- Таблица `result`
- Таблица `cat`

Подробнее рассмотрим каждую таблицу.

Таблица `result`

Таблица «`result`» содержит хешированные записи URL и их категории.

Схема базы данных

Название	Данные
<code>domain_hash</code>	хеш от домена
<code>path_hash</code>	хеш от пути
<code>cat_id</code>	список категорий

С первичным ключом по полям «`domain_hash`», «`path_hash`». Это «таблица измерений» в терминах реляционных баз данных. Способ формирования запроса в эту таблицу описан в Запросы и интерпретация результатов поиска.

Данные в поле «`cat_id`» хранятся в виде blob массива, для стандартизации перечисляемого типа, где каждая категория хранится в виде unsigned short.

Таблица `cat`

Таблица «`cat`» содержит список записей с названиями и идентификаторами категорий. В зависимости от требований клиента, база СкайДНС Octo может поставляться с различным числом категорий с более детальной категоризацией или уникальными именами категорий.

Схема базы данных

Название	Данные
----------	--------

locale	идентификатор локализации
cat_id	идентификатор категории
name	название категории

С первичным составным ключом («locale», «cat_id»). Это таблица «справочник» в терминах реляционных баз данных. Идентификаторы из поля «cat_id» таблицы «result» являются внешним ключом на эту таблицу. После locale содержит идентификатор локализации языка, на котором записано название категории в поле «name». Поле «cat_id» содержит числовой идентификатор категории, данные cat_id не являются последовательными. Поле «name» содержит локализованные названия категорий.

Присутствие в таблице записей на английском языке в любой локализации не считается ошибкой, а указывает на переведенные категории.

Преобразование URL в каноничную форму

Единый указатель ресурса URL состоит из нескольких частей, но нас интересует только два из них:

1. Доменное имя (domain)
2. Путь (path)

Рассмотрим на примере случайного URL:

`directory.google.com/example/test.php?key=value&one=1`

где домен это:

`directory.google.com`

а это путь URL:

`/example/test.php?key=value&one=1`

Поскольку URL указывает на ресурс, он может быть записан в различных формах и различными способами. Для прямого поиска в базе надо привести все разнообразные формы URL, указывающие на один ресурс, к одному каноничному виду. Это важно для получения правильной категоризации запрашиваемого URL.

Мы используем стандартный вариант каноникализации URL из проекта Google Safe Browsing <https://developers.google.com/safe-browsing/> с версией API больше 2. В этом проекте описаны техники для канонизации идентификатора ресурса, с примерами и алгоритмами которого можно ознакомиться на странице <https://developers.google.com/safe-browsing/v4/urls-hashing#canonicalization> (версия от 27.09.2016).

Примеры каноникализации:

Исходный урл	Каноничный
<i>http://evil.com/foo-bar-baz</i>	<i>http://evil.com/foo</i>
<i>http://host/%25%32%35</i>	<i>http://host/%25</i>
<i>http://evil.com/foo-bar-baz</i>	<i>http://evil.com/foo</i>
<i>http://test.com/kek../</i>	<i>http://test.com/</i>

Таким образом мы получим каноничную форму URL. Но из-за природы URL нельзя положиться на один URL, для поиска наилучшего соответствия ему в базе требуется сформировать производные URL, путем поочередного отбрасывания частей первичного URL с левого и правого края.

Рассмотрим на примере случайного URL:

directory.google.com/example/test.php?key=value&one=1

производные URL будут следующими:

1. directory.google.com/example/test.php
2. directory.google.com/example/
3. directory.google.com/
4. google.com/example/test.php?key=value&one=1
5. google.com/example/test.php
6. google.com/example/
7. google.com/

Получившиеся URL требуется проверить по базе.

Хеширование URL

Для прямого поиска URL в базе нужно по отдельности произвести хеширование доменного имени и пути. Хеширование производится функцией MD5 **RFC 1321**, но от результата функции берутся только первые 8 байт. У корня пути (/) особое значение - пустой blob массив.

Примеры на некоторых языках:

Python

```
import hashlib

_root_path_hash = buffer('')

def skydns_hash( value):
```

```
if value == '/':
    return _root_path_hash
return buffer(hashlib.md5(value).digest()[:8])
```

Запросы и интерпретация результатов поиска

Прямые запросы в базу производятся с помощью языка SQL, в таблицу «result».

Пример запроса:

```
SELECT cat_id FROM result
WHERE domain_hash = ? AND path_hash = ?;
```

где вместо вопросительных знаков требуется подставить результаты хеширования доменного имени и пути, преобразовав их blob. Результат запроса различается в зависимости от присутствия этого URL в базе. Если запрос вернул 0 строк, этого URL нет в базе и про него ничего не известно. Если URL найден в базе, результатом на выше приведенный запрос будет одна строка с одной колонкой «cat_id» с типом blob. Это список категорий, для интерпретации надо разобрать это как массив unsigned short. Пример с преобразованием:

```
import struct

def parse_blob_cats(data):
    return struct.unpack('>%dH' % (len(data) // 2), data)
```

Числа в массиве означают идентификатор категории, значения и имена которых описаны в Таблица cat.

Revision #7

Created 11 December 2023 11:12:33 by Виктор

Updated 11 June 2026 05:52:31 by Ольга