# Документ плана оценки производительности запросов SQLite.

**Какие типы данных поддерживает SQLite**

Каждое значение, хранящееся в базе данных SQLite, имеет один из следующих классов хранения.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| **NULL** | Значение - значение NULL. |
| **INTEGER** | Значение представляет собой целое число со знаком, сохраненное в 1, 2, 3, 4, 6 или 8 байтах в зависимости от величины значения. |
| **REAL** | Значение представляет собой значение с плавающей запятой, которое хранится как 8-байтовое число с плавающей точкой IEEE. |
| **TEXT** | Значение представляет собой текстовую строку, хранящуюся с использованием кодировки базы данных (UTF-8, UTF-16BE или UTF-16LE) |
| **BLOB** | Значение представляет собой блок данных, который хранится точно так же, как он был введен. |

**Тип слияния SQLite**

SQLite поддерживает концепцию **affinity** (близость)типа к столбцам. Любой столбец может хранить данные любого типа, но предпочтительный класс хранения для столбца называется **affinity**. Каждому столбцу таблицы в базе данных SQLite3 присваивается одно из следующих аффинностей типа:

|  |  |
| --- | --- |
| **Название** | **Описание** |
| **TEXT** | В этом столбце хранятся все данные с использованием классов хранения NULL, TEXT или BLOB. |
| **NUMERIC** | Этот столбец может содержать значения, используя все пять классов хранения. |
| **INTEGER** | Работает так же, как столбец с NUMERIC сродством, с исключением в выражении CAST. |
| **REAL** | Ведет себя как столбец с NUMERIC сродством, за исключением того, что он приводит целые значения в представление с плавающей запятой. |
| **NONE** | Столбец с аффинностью NONE не предпочитает один класс хранения над другим, и не предпринимаются попытки принудить данные из одного класса хранения к другому. |

**Идентификация и имена типов SQLite**

В следующих списках таблиц перечислены имена типов данных, которые можно использовать при создании таблиц SQLite3 с соответствующим применимым сродством.

|  |  |
| --- | --- |
| **Тип** | **Близость** |
| * INT * INTEGER * TINYINT * SMALLINT * MEDIUMINT * BIGINT * UNSIGNED BIG INT * INT2 * INT8 | INTEGER |
| * CHARACTER(20) * VARCHAR(255) * VARYING CHARACTER(255) * NCHAR(55) * NATIVE CHARACTER(70) * NVARCHAR(100) * TEXT * CLOB | TEXT |
| * BLOB * no datatype specified | NONE |
| * REAL * DOUBLE * DOUBLE PRECISION * FLOAT | REAL |
| * NUMERIC * DECIMAL(10,5) * BOOLEAN * DATE * DATETIME | NUMERIC |

SQLite не имеет отдельного булевского класса хранения. Вместо этого булевые значения сохраняются как целые числа 0 (ложь) и 1 (истина).

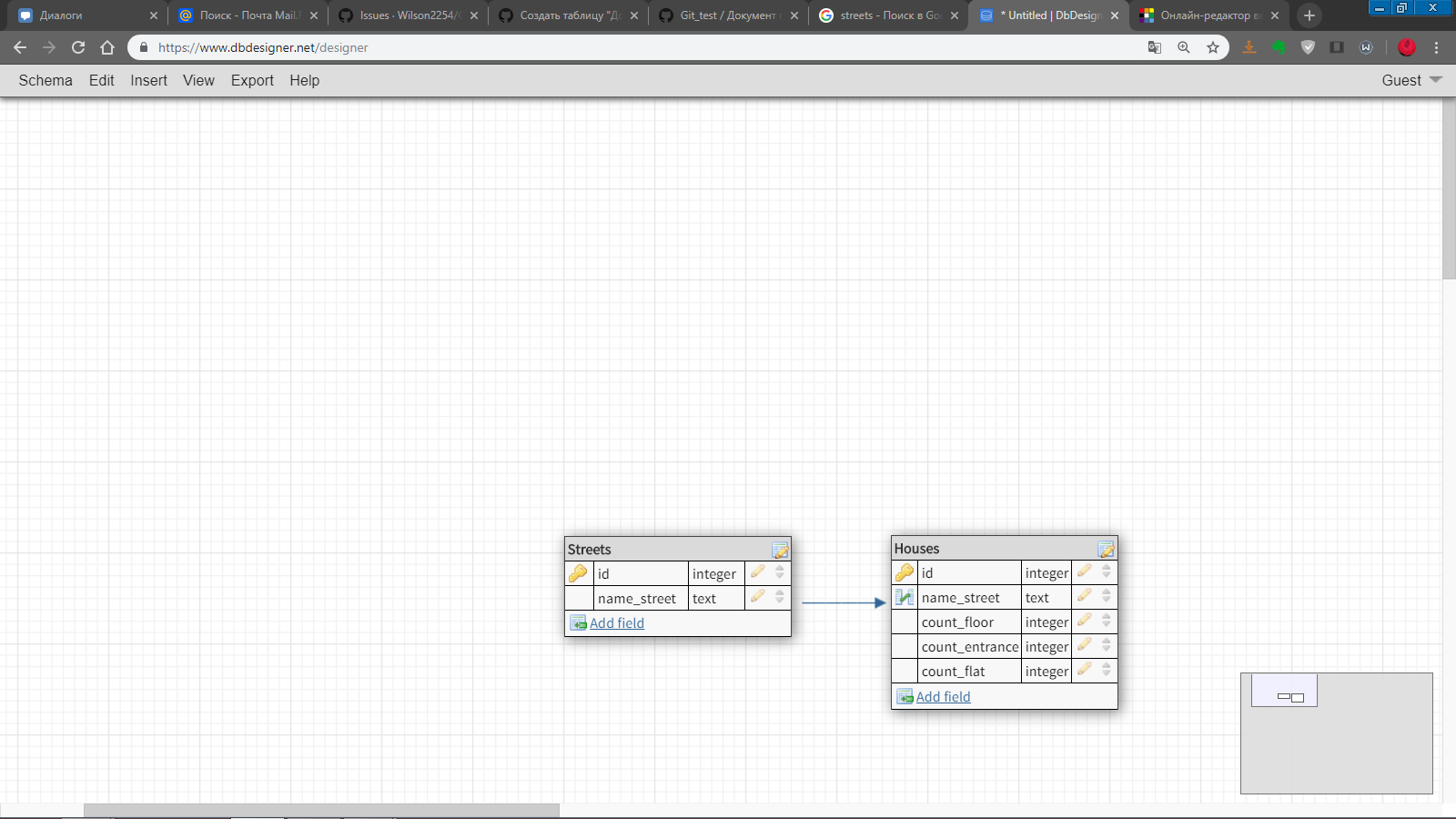
SQLite не имеет отдельного класса хранения для хранения дат и времени, но SQLite способен хранить даты и время как значения TEXT, REAL или INTEGER.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| **TEXT** | Дата в формате "YYYY-MM-DD HH:MM:SS.SSS" |
| **REAL** | Число дней с полудня в Гринвиче 24 ноября 4714 г. до н.э. |
| **INTEGER** | Количество секунд с 1970-01-01 00:00:00 UTC |

* SQLite не поддерживает хранимые процедуры
* SQLIte поддерживает индексирование
* SQLIte поддерживает представления

Для измерения времени запроса нужно создать приложение, которое будет показывать время выполнения заданного запроса.

В базе данных нужно создать 2 таблицы: «улицы» и «дома». Таблица «улицы» будет содержать номер улицы и имя улицы. В таблице «дома» у нас будут храниться номер дома, название улицы, на которой стоит дом, количество этажей, подъездов и квартир в доме. Модель БД приведена ниже.



Скорость запроса должна проверяться на разном количестве записей.