**Создание БД для хранения данных с датчиков «умного дома»**

**Вводные**

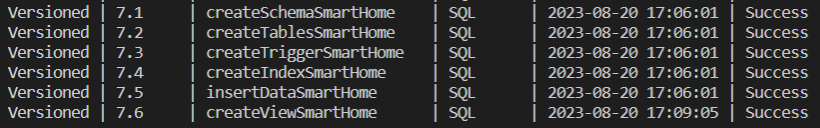
Мы оборудуем дом шестью видами датчиков. Они шлют информацию о своем статусе в БД каждые 10 секунд.

Задача: организовать оптимальное хранение данных и быстрый доступ к ним, продемонстрировав на тестовой выборке в 10 000 строк эффективность применённых настроек.

1. Создаём схему и таблицы для БД в инструменте версионирования flyway
2. FILLFACTOR 100 – апдейты таблицы не предусмотрены
3. TOAST-таблицы не наполняются, т.к. нет больших текстовых полей
4. Таблицы будут журналируемыми, чтобы иметь возможность отслеживать проблемы с датчиками
5. Каждый день по триггеру будет создаваться новая партиция таблицы с данными, которые отправляют счетчики
6. Для ускорения работы с данными используется индекс по дате и типу датчика
7. Тестовые данные для таблицы-лога создаются python-скриптом

**Создание схемы и таблиц**

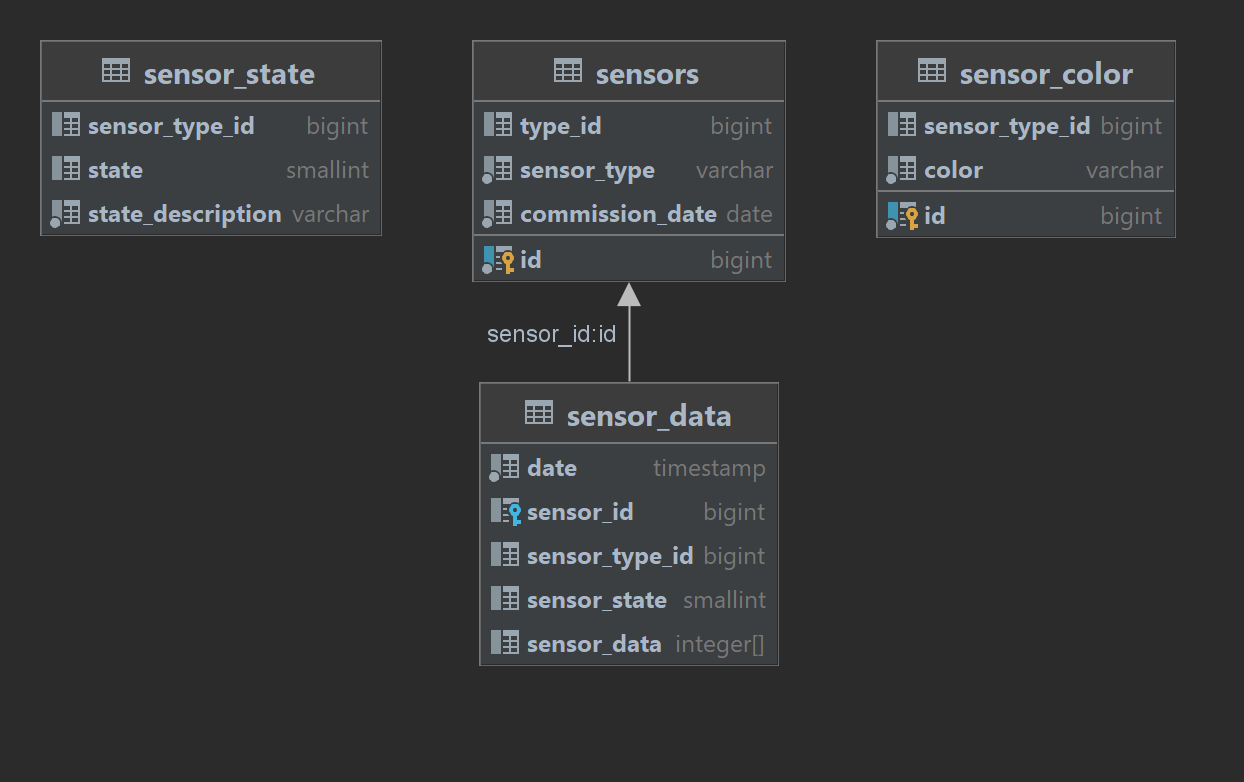
Журнал применения миграций flyway:



Скрипт для создания таблиц в БД (к таблицам применяются ограничения по значениям параметров):

CREATE SCHEMA smart\_home  
create table smart\_home.sensors  
(  
 id bigint primary key,  
 type\_id bigint,  
 sensor\_type varchar not null,  
 commission\_date date not null  
);  
  
create table smart\_home.sensor\_color  
(  
 id bigint primary key,  
 sensor\_type\_id bigint,  
 color varchar not null  
);  
  
alter table smart\_home.sensor\_color add constraint ch\_color check ( color in ('выключено', 'красный', 'зеленый', 'синий') );  
  
create table smart\_home.sensor\_state  
( sensor\_type\_id bigint,  
 state smallint,  
 state\_description varchar not null  
  
);  
  
alter table smart\_home.sensor\_state add constraint ch\_state check ( state in (0, 1) );  
  
create table smart\_home.sensor\_data  
( date timestamp not null,  
 sensor\_id bigint,  
 sensor\_type\_id bigint,  
 sensor\_state smallint,  
 sensor\_data integer[],  
 constraint fk\_sensor\_data\_sensor\_type foreign key (sensor\_id) references smart\_home.sensors(id));  
  
  
alter table smart\_home.sensor\_data add constraint ch\_intensity check ( sensor\_data[2] BETWEEN 0 AND 100 );

Схема таблиц:



**Наполнение БД данными**

Наполнение таблицы с названиями датчиков (в квартире есть несколько датчиков одного типа, все они введены в эксплуатацию одновременно):

insert into smart\_home.sensors values (1000, 1, 'Датчик движения', '2023-01-01');  
insert into smart\_home.sensors values (1001, 1, 'Датчик движения', '2023-01-01');  
insert into smart\_home.sensors values (1002, 1, 'Датчик движения', '2023-01-01');  
insert into smart\_home.sensors values (2000, 2, 'Датчик протечки воды', '2023-01-01');  
insert into smart\_home.sensors values (2001, 2, 'Датчик протечки воды', '2023-01-01');  
insert into smart\_home.sensors values (2002, 2, 'Датчик протечки воды', '2023-01-01');  
insert into smart\_home.sensors values (3000, 3, 'Датчик открытия двери/окна', '2023-01-01');  
insert into smart\_home.sensors values (3001, 3, 'Датчик открытия двери/окна', '2023-01-01');  
insert into smart\_home.sensors values (3002, 3, 'Датчик открытия двери/окна', '2023-01-01');  
insert into smart\_home.sensors values (4000, 4, 'Датчик для управления жалюзи/шторами', '2023-01-01');  
insert into smart\_home.sensors values (4001, 4, 'Датчик для управления жалюзи/шторами', '2023-01-01');  
insert into smart\_home.sensors values (4002, 4, 'Датчик для управления жалюзи/шторами', '2023-01-01');  
insert into smart\_home.sensors values (5000, 5, 'Датчик дыма', '2023-01-01');  
insert into smart\_home.sensors values (5001, 5, 'Датчик дыма', '2023-01-01');  
insert into smart\_home.sensors values (5002, 5, 'Датчик дыма', '2023-01-01');  
insert into smart\_home.sensors values (6000, 6, 'Датчик управления цветом и интенсивностью домашнего освещения', '2023-01-01');  
insert into smart\_home.sensors values (6001, 6, 'Датчик управления цветом и интенсивностью домашнего освещения', '2023-01-01');  
insert into smart\_home.sensors values (6002, 6, 'Датчик управления цветом и интенсивностью домашнего освещения', '2023-01-01');

Наполнение справочника цветов для 6 датчика:

insert into smart\_home.sensor\_color values (0, '6', 'выключено');

insert into smart\_home.sensor\_color values (1, '6', 'красный');  
insert into smart\_home.sensor\_color values (2, '6', 'зеленый');  
insert into smart\_home.sensor\_color values (3, '6', 'синий');

Создание триггера на партицирование данных по дате (оно будет равномерным, т.к. датчики каждый день отправляют одинаковое количество сигналов):

CREATE OR REPLACE FUNCTION *create\_partitions*()  
 RETURNS TRIGGER AS  
$$  
BEGIN  
 EXECUTE *format*('CREATE TABLE IF NOT EXISTS smart\_home.sensor\_data\_%s (LIKE smart\_home.sensor\_data INCLUDING ALL) INHERITS (smart\_home.sensor\_data)', *to\_char*(NEW.date, 'DD\_MM\_YY')) USING NEW;  
 EXECUTE *format*('INSERT INTO smart\_home.sensor\_data\_%s (  
 date,  
 sensor\_type\_id,  
 sensor\_id,  
  
 sensor\_state,  
 sensor\_data  
 ) VALUES (  
 $1.date,  
 $1.sensor\_type\_id,  
 $1.sensor\_id,  
 $1.sensor\_state,  
 $1.sensor\_data  
 )', *to\_char*(NEW.date, 'DD\_MM\_YY')) USING NEW;  
  
 RETURN NULL;  
END;  
$$language plpgsql;  
  
CREATE TRIGGER partitioning\_by\_date BEFORE INSERT ON smart\_home.sensor\_data FOR EACH ROW  
EXECUTE function *create\_partitions*();

Наполнение таблицы с данными:

Скрипт, формирующий файл smart\_home\_data.tsv: https://github.com/MourineRyabova/PostgresCourse

COPY smart\_home.sensor\_data FROM '/var/lib/postgresql/smart\_home\_data.tsv' DELIMITER E'\t';

**Создание VIEW**

Создание человекочитаемой вью-таблицы:

CREATE VIEW smart\_home.sensor\_data\_view AS  
(  
SELECT date AS "Время отправки показаний",  
 sensor\_type AS "Тип датчика",  
 state\_description AS "Состояние датчика",  
 case sd.sensor\_id  
 when 6 then *json\_agg*((SELECT x FROM (SELECT sc.color AS "Цвет", sd.sensor\_data[2] AS "Интенсивность") AS x))  
 else null end AS "Дополнительные характеристики"  
from smart\_home.sensor\_data sd  
 left join smart\_home.sensors ON sd.sensor\_id = sensors.id  
 left join smart\_home.sensor\_state ss on (sd.sensor\_id = ss.sensor\_id AND sd.sensor\_state = ss.state)  
 left join smart\_home.sensor\_color sc on sd.sensor\_data[1] = sc.id  
group by date, sensor\_type, sd.sensor\_id, state\_description  
 );

**Создание индекса для повышения скорости доступа к данным**

Запрос для проверки быстродействия:

SELECT date, sensor\_state, sensor\_data FROM smart\_home.sensor\_data  
WHERE sensor\_id = 1001 AND date BETWEEN '2023-08-01 23:30:00.000000' AND '2023-08-01 23:50:00.000000'

Создание уникального индекса:

CREATE UNIQUE INDEX idx ON smart\_home.sensor\_data(date, sensor\_id);  
ANALYZE smart\_home.sensor\_data;

Тестирование pgbench (время теста 20 секунд)

Содержание теста:

\set sensor\_id random(1000,6003)

BEGIN;

SELECT date, sensor\_state, sensor\_data FROM smart\_home.sensor\_data

WHERE sensor\_id=:sensor\_id AND date BETWEEN '2023-08-01 23:30:00.000000' AND '2023-08-01 23:50:00.000000';

END;

Команда запуска теста:

pgbench -T 20 -f hw\_final.sql db\_course -U user\_course

Результаты теста:

|  |  |  |  |
| --- | --- | --- | --- |
| Попытка | number of transactions | Latency | TPS |
| Запрос без оптимизации | 18485 | 1.082 ms | 924.412928 |
| Запрос с индексом по полям date, sensor\_id  Но без опции наследования INCLUDING ALL | 19306 | 1.036 ms | 965.541385 |
| Запрос с индексом по полям date, sensor\_id  C опцией наследования INCLUDING ALL | 26324 | 0.760 ms | 1316.484690 |

План запроса:

EXPLAIN ANALYZE SELECT date, sensor\_state, sensor\_data FROM smart\_home.sensor\_data  
WHERE sensor\_id = 1001 AND date BETWEEN '2023-08-01 23:30:00.000000' AND '2023-08-01 23:50:00.000000'

