Министерство образования и науки Российской Федерации

(МИНОБРНАУКИ РОССИИ)

ЯРОСЛАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

им. П. Г. Демидова

Факультет информатики и вычислительной техники

Кафедра дискретного анализа

Отчёт

по дисциплине «Объектные базы данных»

на тему «Автотранспортное предприятие»

Выполнил

Студент группы ИТ-2МО

Куликова Э. В

Проверил

Профессор д.ф.-м.н.

Башкин В. А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ярославль 2022

**Содержание**

[Предметная область 3](#_Toc122897247)

[Создание БД и таблиц 6](#_Toc122897248)

[Заполнение таблиц данными 11](#_Toc122897249)

[Работа с представлениями (VIEW) 21](#_Toc122897250)

[Работа с данными 23](#_Toc122897251)

[Создание маршрута в формате KML 24](#_Toc122897252)

[Роли доступа 26](#_Toc122897253)

# **Предметная область**

База данных для автотранспортного предприятия предоставляет данные о рейсах, маршрутах, водителях и транспортных средствах.

В настоящее время применяют проектирование с использованием метода "Сущность-связь" (ER–method), который является комбинацией предметного и прикладного методов и обладает достоинствами обоих. Этап инфологического проектирования начинается с моделирования предметной области.

ER-диаграммы хорошо вписываются в методологию структурного анализа и проектирования информационных систем. Такие методологии обеспечивают строгое и наглядное описание проектируемой системы, которое начинается с ее общего обзора и затем уточняется, давая возможность получить различную степень детализации объекта с различным числом уровней.

Проанализировав предметную область, выделим следующие сущности:

1. Маршруты;
2. Рейсы;
3. Транспортные средства;
4. Водители;
5. Остановки;
6. Остановки на маршрутах.

Под понятием «сущность» в модели понимается некоторая абстракция реально существующего объекта процесса или явления.

Эти сущности представляют собой ядро проектируемой базы данных. Сущность «маршруты» будет содержать информацию о маршрутах, по которым будет следовать транспортное средство. Сведения о транспорте будет содержать сущность «транспортные средства». Сущность «водители» будет содержать информацию о водителях управляющих транспортом. Сущность «остановки» содержит информацию об остановках, включённых в маршрут.

Для уменьшения избыточности хранимой информации выделим таблицу «рейсы» между таблицами «маршруты», «транспортные средства» и «водители». В ней будет содержаться информация обо всех перевозках транспортного предприятия. А также выделим таблицу «остановки на маршрутах» между таблицами «остановки» и «маршрут», чтобы знать на каких остановках будет останавливаться автобус.

В результате анализа мы получили шесть отдельных таблиц, каждая из которых представляет определенный класс сущности.

Определим классы для наших сущностей. Стержневыми сущностями будут являться следующие: маршруты, транспортные средства, остановки и водители. Эти сущности имеют независимое существование. Ассоциациями в нашей базе данных будут сущности: рейсы и остановки на маршрутах. Эти сущности имеют связь вида «один ко многим» и «многим к одной».

Для каждой сущности выбираются свойства (атрибуты) и определяются типы отношений между сущностями или атрибутами. Определим атрибуты для нашей базы данных.

Таблица 1 - Спецификация атрибутов сущности «Транспортные средства»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Описание атрибута | Тип |
| Код транспортного средства | Идентификационный номер транспортного средства | integer |
| Технический паспорт | Технический паспорт транспортного средства, содержит обычно: номер паспорта, номер автомобиля, номер двигателя, номер кузова (VIN), цвет ТС, марка ТС, год выпуска стоит ли ТС на учёте ГИБДД и т. д. | jsonb |

Ключевым атрибутом данной сущности является «Код транспортного средства».

Таблица 2 - Спецификация атрибутов сущности "Остановки"

|  |  |  |
| --- | --- | --- |
| Название атрибута | Описание атрибута | Тип |
| Название остановки | Уникальное название остановки | character varying(40) |
| Координаты остановки | Координаты остановки с использованием POSTGIS индексов | geometry |

В качестве ключа использовали «Название остановки».

Таблица 3 - Спецификация атрибутов сущности "Водители"

|  |  |  |
| --- | --- | --- |
| Название атрибута | Описание атрибута | Тип |
| Код водителя | Идентификационный номер | integer |
| ФИО | Фамилия Имя Отчество водителя | character varying(60) |
| Дата рождения | Дата рождения водителя | date |
| Категория водительского удостоверения | Категория водительского удостоверения позволяет управлять тем или иным транспортным средством | character varying(3)[] |

Ключевым атрибутом данной сущности является «Код водителя».

Таблица 4 - Спецификация атрибутов сущности "Маршруты"

|  |  |  |
| --- | --- | --- |
| Название атрибута | Описание атрибута | Тип |
| Код маршрута | Идентификационный номер | integer |
| Номер маршрута | Номер маршрута присвоенный ПАТП | character varying(5) |
| Количество остановок | Количество остановок на данном маршруте | integer |
| Длина маршрута | Длина маршрута в (км) | real |

В качестве ключа использовали «Код маршрута».

Таблица 5 - Спецификация атрибутов сущности "Рейсы"

|  |  |  |
| --- | --- | --- |
| Название атрибута | Описание атрибута | Тип |
| Дата рейса | Дата и время конкретного рейса | timesamp withpou time zone |
| Код транспортного средства | Номер транспортного средства | integer |
| Код водителя | Номер водителя | integer |
| Код маршрута | Номер маршрута | integer |
| Выручка | Количество заработанных денег за рейс | numeric |

В сущности «Рейсы» был выбран составной первичный ключ – «Код транспортного средства», «Код водителя», «Код маршрута». Эти атрибуты также будут являться внешними ключами. «Код ТС» связывает нас с таблицей «транспортные средства» через идентификатор ТС. «Код водителя» - с таблицей «водители» через идентификатор водителя. «Код маршрута» - с таблицей «маршруты» через идентификатор маршрута.

Таблица 6 - Спецификация атрибутов сущности "Остановки на маршрутах"

|  |  |  |
| --- | --- | --- |
| Название атрибута | Описание атрибута | Тип |
| Код маршрута | Номер маршрута | integer |
| Название остановки | Название остановки на маршруте | character varying(40) |

Ключевым атрибутом данной сущности является «Код маршрута» и «Название остановки». Они также будут и внешними ключами, которые связывают нас с таблицами «маршруты» и «остановки» по названиям атрибутов соответственно.

На основании выше изложенного построим полную инфологическую модель базы данных «автотранспортного предприятия» и изобразим ее в виде ER – диаграммы.

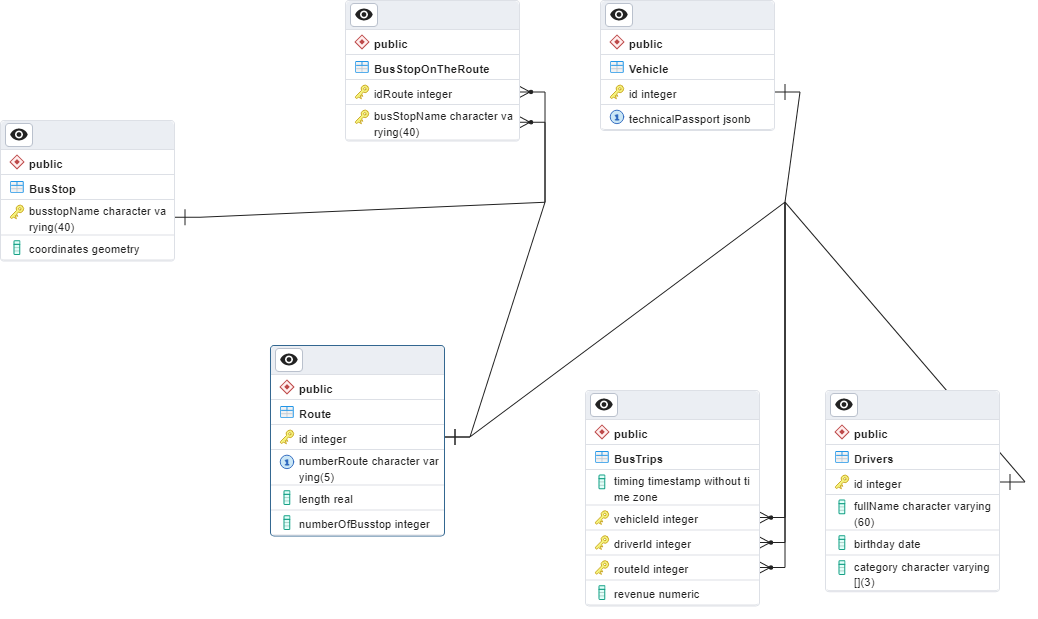


Рисунок 1 - ER - диаграмма "Автотранспортное предприятие"

# **Создание БД и таблиц**

Используемая СУБД – PostgresSQL.

Основные команды для создания, удаления и изменения базы данных «автотранспортное предприятие»:

-- DROP DATABASE IF EXISTS "TransportCompany";

**CREATE** **DATABASE** "TransportCompany"

**WITH**

**OWNER** = postgres

**ENCODING** = 'UTF8'

LC\_COLLATE = 'Russian\_Russia.1252'

LC\_CTYPE = 'Russian\_Russia.1252'

TABLESPACE = pg\_default

**CONNECTION** **LIMIT** = -**1**

IS\_TEMPLATE = **False**;

**ALTER** **DATABASE** "TransportCompany"

**SET** search\_path **TO** "$user", **public**, topology, tiger;

Основные команды для создания и удаления схемы:

-- DROP SCHEMA IF EXISTS public ;

**CREATE** **SCHEMA** IF **NOT** **EXISTS** **public**

**AUTHORIZATION** pg\_database\_owner;

**COMMENT** **ON** **SCHEMA** **public**

**IS** 'standard public schema';

**GRANT** **USAGE** **ON** **SCHEMA** **public** **TO** **PUBLIC**;

**GRANT** **ALL** **ON** **SCHEMA** **public** **TO** pg\_database\_owner;

Создадим последовательности для автоматического заполнения идентификаторов водителя, маршрута и транспортного средства:

-- создание, удаление и изменение последовательности для автоматического заполнения идентификатора водителя

-- DROP SEQUENCE IF EXISTS public."Drivers\_id\_seq";

**CREATE** SEQUENCE IF **NOT** **EXISTS** **public**."Drivers\_id\_seq"

**INCREMENT** **1**

**START** **1**

**MINVALUE** **1**

**MAXVALUE** **2147483647**

**CACHE** **1**

OWNED **BY** "Drivers".id;

**ALTER** SEQUENCE **public**."Drivers\_id\_seq"

**OWNER** **TO** postgres;

-- создание, удаление и изменение последовательности для автоматического заполнения идентификатора маршрута

-- DROP SEQUENCE IF EXISTS public."Route\_id\_seq";

**CREATE** SEQUENCE IF **NOT** **EXISTS** **public**."Route\_id\_seq"

**INCREMENT** **1**

**START** **1**

**MINVALUE** **1**

**MAXVALUE** **2147483647**

**CACHE** **1**

OWNED **BY** "Route".id;

**ALTER** SEQUENCE **public**."Route\_id\_seq"

**OWNER** **TO** postgres;

-- создание, удаление и изменение последовательности для автоматического заполнения идентификатора транспортного средства

-- DROP SEQUENCE IF EXISTS public."Vehicle\_id\_seq";

**CREATE** SEQUENCE IF **NOT** **EXISTS** **public**."Vehicle\_id\_seq"

**INCREMENT** **1**

**START** **1**

**MINVALUE** **1**

**MAXVALUE** **2147483647**

**CACHE** **1**

OWNED **BY** "Vehicle".id;

**ALTER** SEQUENCE **public**."Vehicle\_id\_seq"

**OWNER** **TO** postgres;

Установим расширение POSTGIS для работы с координатами остановок:

-- Включить PostGIS (начиная с 3.0 содержит только геометрию/географию)

**CREATE** EXTENSION postgis;

-- включить поддержку растра (для 3+)

**CREATE** EXTENSION postgis\_raster;

-- Включить топологию

**CREATE** EXTENSION postgis\_topology;

-- Включить PostGIS Advanced 3D

-- и другие алгоритмы геообработки

-- sfcgal доступен не во всех дистрибутивах

**CREATE** EXTENSION postgis\_sfcgal;

-- нечеткое сопоставление, необходимое для Tiger

**CREATE** EXTENSION fuzzystrmatch;

-- стандартизатор на основе правил

**CREATE** EXTENSION address\_standardizer;

-- пример набора данных правила

**CREATE** EXTENSION address\_standardizer\_data\_us;

-- Включить US Tiger Geocoder

**CREATE** EXTENSION postgis\_tiger\_geocoder;

Теперь перейдём к созданию таблиц по сущностям, описанным в предметной области:

-- создание, удаление и изменение таблицы "Транспортное средство"

-- DROP TABLE IF EXISTS public."Vehicle";

**CREATE** **TABLE** IF **NOT** **EXISTS** **public**."Vehicle"

(

id integer **NOT** **NULL** **DEFAULT** nextval('"Vehicle\_id\_seq"'::regclass),

"technicalPassport" jsonb **NOT** **NULL**,

**CONSTRAINT** "Vehicle\_pkey" **PRIMARY** **KEY** (id),

**CONSTRAINT** "Vehicle\_technicalPassport\_key" **UNIQUE** ("technicalPassport")

)

TABLESPACE pg\_default;

**ALTER** **TABLE** IF **EXISTS** **public**."Vehicle"

**OWNER** **to** postgres;

-- создание, удаление и изменение таблицы "Остановки"

-- DROP TABLE IF EXISTS public."BusStop";

**CREATE** **TABLE** IF **NOT** **EXISTS** **public**."BusStop"

(

"busstopName" character varying(**40**) **COLLATE** pg\_catalog."default" **NOT** **NULL**,

coordinates geometry(Point,**26910**) **NOT** **NULL**,

**CONSTRAINT** "BusStop\_pkey" **PRIMARY** **KEY** ("busstopName")

)

TABLESPACE pg\_default;

**ALTER** **TABLE** IF **EXISTS** **public**."BusStop"

**OWNER** **to** postgres;

-- создание и удаление индекса для координат остановок

-- DROP INDEX IF EXISTS public.busstop\_gix;

**CREATE** **INDEX** IF **NOT** **EXISTS** busstop\_gix

**ON** **public**."BusStop" **USING** gist

(coordinates)

TABLESPACE pg\_default;

-- создание, удаление и изменение таблицы "Маршруты"

-- DROP TABLE IF EXISTS public."Route";

**CREATE** **TABLE** IF **NOT** **EXISTS** **public**."Route"

(

id integer **NOT** **NULL** **DEFAULT** nextval('"Route\_id\_seq"'::regclass),

"numberRoute" character varying(**5**) **COLLATE** pg\_catalog."default" **NOT** **NULL**,

**length** real **NOT** **NULL**,

"numberOfBusstop" integer **NOT** **NULL**,

**CONSTRAINT** "Route\_pkey" **PRIMARY** **KEY** (id),

**CONSTRAINT** "Route\_numberRoute\_key" **UNIQUE** ("numberRoute"),

**CONSTRAINT** "positiveLength" **CHECK** (**length** > **0**::double **precision**),

**CONSTRAINT** "minBusstop" **CHECK** ("numberOfBusstop" >= **2**)

)

TABLESPACE pg\_default;

**ALTER** **TABLE** IF **EXISTS** **public**."Route"

**OWNER** **to** postgres;

-- создание, удаление и изменение таблицы "Остановки на маршрутах"

-- DROP TABLE IF EXISTS public."BusStopOnTheRoute";

**CREATE** **TABLE** IF **NOT** **EXISTS** **public**."BusStopOnTheRoute"

(

"idRoute" integer **NOT** **NULL**,

"busStopName" character varying(**40**) **COLLATE** pg\_catalog."default" **NOT** **NULL**,

**CONSTRAINT** "busStopOnTheRoute\_pkey" **PRIMARY** **KEY** ("idRoute", "busStopName"),

**CONSTRAINT** "busStopOnTheRoute\_busStopName\_fkey" **FOREIGN** **KEY** ("busStopName")

**REFERENCES** **public**."BusStop" ("busstopName") **MATCH** **SIMPLE**

**ON** **UPDATE** **CASCADE**

**ON** **DELETE** **CASCADE**,

**CONSTRAINT** "busStopOnTheRoute\_idRoute\_fkey" **FOREIGN** **KEY** ("idRoute")

**REFERENCES** **public**."Route" (id) **MATCH** **SIMPLE**

**ON** **UPDATE** **CASCADE**

**ON** **DELETE** **CASCADE**

)

TABLESPACE pg\_default;

**ALTER** **TABLE** IF **EXISTS** **public**."BusStopOnTheRoute"

**OWNER** **to** postgres;

-- создание, удаление и изменение таблицы "Водители"

-- DROP TABLE IF EXISTS public."Drivers";

**CREATE** **TABLE** IF **NOT** **EXISTS** **public**."Drivers"

(

id integer **NOT** **NULL** **DEFAULT** nextval('"Drivers\_id\_seq"'::regclass),

"fullName" character varying(**60**) **COLLATE** pg\_catalog."default" **NOT** **NULL**,

birthday date,

category character varying(**3**)[] **COLLATE** pg\_catalog."default" **NOT** **NULL**,

**CONSTRAINT** "Drivers\_pkey" **PRIMARY** **KEY** (id)

)

TABLESPACE pg\_default;

**ALTER** **TABLE** IF **EXISTS** **public**."Drivers"

**OWNER** **to** postgres;

-- создание, удаление и изменение таблицы "Рейсы"

-- DROP TABLE IF EXISTS public."BusTrips";

**CREATE** **TABLE** IF **NOT** **EXISTS** **public**."BusTrips"

(

timing **timestamp** **without** time **zone** **NOT** **NULL**,

"vehicleId" integer **NOT** **NULL**,

"driverId" integer **NOT** **NULL**,

"routeId" integer **NOT** **NULL**,

revenue numeric,

**CONSTRAINT** "BusTrips\_pkey" **PRIMARY** **KEY** ("vehicleId", "driverId", "routeId"),

**CONSTRAINT** "BusTrips\_driverId\_fkey" **FOREIGN** **KEY** ("driverId")

**REFERENCES** **public**."Drivers" (id) **MATCH** **SIMPLE**

**ON** **UPDATE** **CASCADE**

**ON** **DELETE** **CASCADE**,

**CONSTRAINT** "BusTrips\_routeId\_fkey" **FOREIGN** **KEY** ("routeId")

**REFERENCES** **public**."Route" (id) **MATCH** **SIMPLE**

**ON** **UPDATE** **CASCADE**

**ON** **DELETE** **CASCADE**,

**CONSTRAINT** "BusTrips\_vehicleId\_fkey" **FOREIGN** **KEY** ("vehicleId")

**REFERENCES** **public**."Vehicle" (id) **MATCH** **SIMPLE**

**ON** **UPDATE** **CASCADE**

**ON** **DELETE** **CASCADE**,

**CONSTRAINT** "positiveRevenue" **CHECK** (revenue >= **0**::numeric)

)

TABLESPACE pg\_default;

**ALTER** **TABLE** IF **EXISTS** **public**."BusTrips"

**OWNER** **to** postgres;

**Заполнение таблиц данными**

Далее необходимо заполнить таблицы данными. Для этого создадим несколько процедур. Одни будут добавлять по одной строке данных с атрибутами в каждой определённой таблице, а другие полностью заполнять таблицы конкретными данными. Заметим, что в таблице «остановки», мы используем специальные методы POSTGIS для создания геометрических координат. К примеру, метод - ST\_GeomFromText - создает объект PostGIS ST\_Geometry из общеизвестного текстового представления OGC. То есть точки будут представлены типом geometry для дальнейшей работы с геометрией и топологией.

-- создание, удаление и изменение процедуры добавления одной остановки в таблицу "Остановки"

-- DROP PROCEDURE IF EXISTS public.addbusstop(character varying, geometry);

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.addbusstop(

**IN** "\_busstopName" character varying,

**IN** \_coordinates geometry)

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**BEGIN**

**insert** **into** **public**."BusStop"("busstopName", coordinates)

**values** ("\_busstopName", \_coordinates);

**END**;

$BODY$;

**ALTER** **PROCEDURE** **public**.addbusstop(character varying, geometry)

**OWNER** **TO** postgres;

-- создание, удаление, изменение процедуры заполнения таблицы "Остановки" данными

-- DROP PROCEDURE IF EXISTS public.filltablebusstop();

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.filltablebusstop(

)

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**BEGIN**

**CALL** AddBusStop('ЯМЗ', ST\_GeomFromText('POINT(57.646997 39.850673)', **26910**));

**CALL** AddBusStop('Областная онкологическая больница', ST\_GeomFromText('POINT(57.640081 39.856833)', **26910**));

**CALL** AddBusStop('Площадь Карла Маркса', ST\_GeomFromText('POINT(57.636343 39.866502)', **26910**));

**CALL** AddBusStop('Советская улица', ST\_GeomFromText('POINT(57.639914 39.875262)', **26910**));

**CALL** AddBusStop('Октябрьская площадь', ST\_GeomFromText('POINT(57.638169 39.884107)', **26910**));

**CALL** AddBusStop('Дачная улица', ST\_GeomFromText('POINT(57.644224 39.910436)', **26910**));

**CALL** AddBusStop('Школа № 50', ST\_GeomFromText('POINT(57.644233 39.910403)', **26910**));

**CALL** AddBusStop('Школа № 46', ST\_GeomFromText('POINT(57.638172 39.925519)', **26910**));

**CALL** AddBusStop('8-й переулок Маяковского', ST\_GeomFromText('POINT(57.638208 39.925519)', **26910**));

**CALL** AddBusStop('Медсанчасть ЯЗДА', ST\_GeomFromText('POINT(57.629571 39.936309)', **26910**));

**CALL** AddBusStop('Университетский городок', ST\_GeomFromText('POINT(57.626590 39.940127)', **26910**));

**CALL** AddBusStop('Шоссейная улица', ST\_GeomFromText('POINT(57.613302 39.942262)', **26910**));

**CALL** AddBusStop('Средний посёлок', ST\_GeomFromText('POINT(57.607061 39.950427)', **26910**));

**CALL** AddBusStop('Школа № 47', ST\_GeomFromText('POINT(57.607560 39.959325)', **26910**));

**CALL** AddBusStop('1-я Больничная улица', ST\_GeomFromText('POINT(57.608251 39.972216)', **26910**));

**CALL** AddBusStop('Школа № 51', ST\_GeomFromText('POINT(57.608627 39.979204)', **26910**));

**CALL** AddBusStop('Нижний посёлок', ST\_GeomFromText('POINT(57.608877 39.983189)', **26910**));

**CALL** AddBusStop('Студенческий городок', ST\_GeomFromText('POINT(57.621780 39.927338)', **26910**));

**CALL** AddBusStop('Красная площадь', ST\_GeomFromText('POINT(57.632453 39.887929)', **26910**));

**CALL** AddBusStop('Линейная улица', ST\_GeomFromText('POINT(57.648001 39.926475)', **26910**));

**CALL** AddBusStop('Красноборская улица', ST\_GeomFromText('POINT(57.652541 39.936034)', **26910**));

**CALL** AddBusStop('Проспект Машиностроителей', ST\_GeomFromText('POINT(57.654151 39.941497)', **26910**));

**CALL** AddBusStop('Проезд Доброхотова', ST\_GeomFromText('POINT(57.653346 39.944228)', **26910**));

**CALL** AddBusStop('Кинотеатр Аврора', ST\_GeomFromText('POINT(57.651003 39.947779)', **26910**));

**CALL** AddBusStop('Улица Саукова', ST\_GeomFromText('POINT(57.647635 39.951876)', **26910**));

**CALL** AddBusStop('Улица Папанина', ST\_GeomFromText('POINT(57.645219 39.955017)', **26910**));

**CALL** AddBusStop('Улица Сахарова', ST\_GeomFromText('POINT(57.639141 39.962527)', **26910**));

**CALL** AddBusStop('Школа № 48', ST\_GeomFromText('POINT(57.642436 39.958704)', **26910**));

**CALL** AddBusStop('Хутор', ST\_GeomFromText('POINT(57.636285 39.966078)', **26910**));

**CALL** AddBusStop('ЯЗДА', ST\_GeomFromText('POINT(57.621193 39.967717)', **26910**));

**CALL** AddBusStop('Машприбор', ST\_GeomFromText('POINT(57.615624 39.963620)', **26910**));

**CALL** AddBusStop('Залесская улица', ST\_GeomFromText('POINT(57.614525 39.953241)', **26910**));

**CALL** AddBusStop('Торговый переулок', ST\_GeomFromText('POINT(57.625101 39.885419)', **26910**));

**CALL** AddBusStop('Республиканская улица', ST\_GeomFromText('POINT(57.633595 39.878915)', **26910**));

**CALL** AddBusStop('Улица Победы', ST\_GeomFromText('POINT(57.634505 39.873826)', **26910**));

**CALL** AddBusStop('Ярославль-Главный', ST\_GeomFromText('POINT(57.625935 39.835948)', **26910**));

**CALL** AddBusStop('Полиграфкомбинат', ST\_GeomFromText('POINT(57.627215 39.843961)', **26910**));

**CALL** AddBusStop('Дом обуви', ST\_GeomFromText('POINT(57.629410 39.849416)', **26910**));

**CALL** AddBusStop('Улица Кудрявцева', ST\_GeomFromText('POINT(57.630964 39.853166)', **26910**));

**CALL** AddBusStop('Юбилейная площадь', ST\_GeomFromText('POINT(57.633433 39.860326)', **26910**));

**CALL** AddBusStop('Сабанеевская улица', ST\_GeomFromText('POINT(57.629371 39.968558)', **26910**));

**END**;

$BODY$;

**ALTER** **PROCEDURE** **public**.filltablebusstop()

**OWNER** **TO** postgres;

-- заполнить данными таблицу "Остановки"

**CALL** filltablebusstop();

-- создание, удаление и изменение процедуры добавления одного маршрута в таблицу "Маршруты"

-- DROP PROCEDURE IF EXISTS public.addroute(character varying, integer, real);

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.addroute(

**IN** "\_numberRoute" character varying,

**IN** "\_numberOfBusstop" integer,

**IN** \_length real)

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**BEGIN**

**insert** **into** **public**."Route"("numberRoute", "numberOfBusstop", **length**)

**values** ("\_numberRoute", "\_numberOfBusstop", \_length)

**ON** CONFLICT **DO** **NOTHING**;

**END**;

$BODY$;

**ALTER** **PROCEDURE** **public**.addroute(character varying, integer, real)

**OWNER** **TO** postgres;

-- создание, удаление, изменение процедуры заполнения таблицы "Маршруты" данными

-- DROP PROCEDURE IF EXISTS public.filltableroute();

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.filltableroute(

)

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**BEGIN**

**CALL** AddRoute('22', **17**, **12**.**80**);

**CALL** AddRoute('22c', **9**, **7**.**10**);

**CALL** AddRoute('12', **18**, **12**.**20**);

**CALL** AddRoute('39', **25**, **14**.**90**);

**CALL** AddRoute('30', **23**, **12**.**80**);

**END**;

$BODY$;

**ALTER** **PROCEDURE** **public**.filltableroute()

**OWNER** **TO** postgres;

-- заполнить данными таблицу "Маршруты"

**CALL** filltableroute();

-- создание, удаление и изменение процедуры добавления одной остановке на маршруте в таблицу "Остановки на маршрутах"

-- DROP PROCEDURE IF EXISTS public.addbusstopontheroute(integer, character varying);

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.addbusstopontheroute(

**IN** "\_idRoute" integer,

**IN** "\_busStopName" character varying)

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**BEGIN**

**insert** **into** **public**."BusStopOnTheRoute"("idRoute", "busStopName")

**values** ("\_idRoute", "\_busStopName")

**ON** CONFLICT **DO** **NOTHING**;

**END**;

$BODY$;

**ALTER** **PROCEDURE** **public**.addbusstopontheroute(integer, character varying)

**OWNER** **TO** postgres;

-- создание, удаление, изменение процедуры заполнения таблицы "Остановки на маршрутах" данными

-- DROP PROCEDURE IF EXISTS public.filltablebusstopontheroute();

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.filltablebusstopontheroute(

)

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**BEGIN**

**CALL** AddBusStopOnTheRoute(**31**, 'ЯМЗ');

**CALL** AddBusStopOnTheRoute(**31**, 'Областная онкологическая больница');

**CALL** AddBusStopOnTheRoute(**31**, 'Площадь Карла Маркса');

**CALL** AddBusStopOnTheRoute(**31**, 'Советская улица');

**CALL** AddBusStopOnTheRoute(**31**, 'Октябрьская площадь');

**CALL** AddBusStopOnTheRoute(**31**, 'Дачная улица');

**CALL** AddBusStopOnTheRoute(**31**, 'Школа № 50');

**CALL** AddBusStopOnTheRoute(**31**, 'Школа № 46');

**CALL** AddBusStopOnTheRoute(**31**, '8-й переулок Маяковского');

**CALL** AddBusStopOnTheRoute(**31**, 'Медсанчасть ЯЗДА');

**CALL** AddBusStopOnTheRoute(**31**, 'Университетский городок');

**CALL** AddBusStopOnTheRoute(**31**, 'Шоссейная улица');

**CALL** AddBusStopOnTheRoute(**31**, 'Средний посёлок');

**CALL** AddBusStopOnTheRoute(**31**, 'Школа № 47');

**CALL** AddBusStopOnTheRoute(**31**, '1-я Больничная улица');

**CALL** AddBusStopOnTheRoute(**31**, 'Школа № 51');

**CALL** AddBusStopOnTheRoute(**31**, 'Нижний посёлок');

**CALL** AddBusStopOnTheRoute(**32**, 'Красная площадь');

**CALL** AddBusStopOnTheRoute(**32**, 'Октябрьская площадь');

**CALL** AddBusStopOnTheRoute(**32**, 'Дачная улица');

**CALL** AddBusStopOnTheRoute(**32**, 'Школа № 50');

**CALL** AddBusStopOnTheRoute(**32**, 'Школа № 46');

**CALL** AddBusStopOnTheRoute(**32**, '8-й переулок Маяковского');

**CALL** AddBusStopOnTheRoute(**32**, 'Медсанчасть ЯЗДА');

**CALL** AddBusStopOnTheRoute(**32**, 'Университетский городок');

**CALL** AddBusStopOnTheRoute(**32**, 'Студенческий городок');

**CALL** AddBusStopOnTheRoute(**33**, 'Торговый переулок');

**CALL** AddBusStopOnTheRoute(**33**, 'Красная площадь');

**CALL** AddBusStopOnTheRoute(**33**, 'Республиканская улица');

**CALL** AddBusStopOnTheRoute(**33**, 'Улица Победы');

**CALL** AddBusStopOnTheRoute(**33**, 'Октябрьская площадь');

**CALL** AddBusStopOnTheRoute(**33**, 'Дачная улица');

**CALL** AddBusStopOnTheRoute(**33**, 'Школа № 50');

**CALL** AddBusStopOnTheRoute(**33**, 'Школа № 46');

**CALL** AddBusStopOnTheRoute(**33**, '8-й переулок Маяковского');

**CALL** AddBusStopOnTheRoute(**33**, 'Медсанчасть ЯЗДА');

**CALL** AddBusStopOnTheRoute(**33**, 'Университетский городок');

**CALL** AddBusStopOnTheRoute(**33**, 'Шоссейная улица');

**CALL** AddBusStopOnTheRoute(**33**, 'Средний посёлок');

**CALL** AddBusStopOnTheRoute(**33**, 'Школа № 47');

**CALL** AddBusStopOnTheRoute(**33**, '1-я Больничная улица');

**CALL** AddBusStopOnTheRoute(**33**, 'Школа № 51');

**CALL** AddBusStopOnTheRoute(**33**, 'Нижний посёлок');

**CALL** AddBusStopOnTheRoute(**34**, 'Школа № 50');

**CALL** AddBusStopOnTheRoute(**34**, 'Школа № 46');

**CALL** AddBusStopOnTheRoute(**34**, '8-й переулок Маяковского');

**CALL** AddBusStopOnTheRoute(**34**, 'Медсанчасть ЯЗДА');

**CALL** AddBusStopOnTheRoute(**34**, 'Университетский городок');

**CALL** AddBusStopOnTheRoute(**34**, 'Студенческий городок');

**CALL** AddBusStopOnTheRoute(**34**, 'Линейная улица');

**CALL** AddBusStopOnTheRoute(**34**, 'Красноборская улица');

**CALL** AddBusStopOnTheRoute(**34**, 'Проспект Машиностроителей');

**CALL** AddBusStopOnTheRoute(**34**, 'Проезд Доброхотова');

**CALL** AddBusStopOnTheRoute(**34**, 'Кинотеатр Аврора');

**CALL** AddBusStopOnTheRoute(**34**, 'Улица Саукова');

**CALL** AddBusStopOnTheRoute(**34**, 'Улица Папанина');

**CALL** AddBusStopOnTheRoute(**34**, 'Улица Сахарова');

**CALL** AddBusStopOnTheRoute(**34**, 'Школа № 48');

**CALL** AddBusStopOnTheRoute(**34**, 'Хутор');

**CALL** AddBusStopOnTheRoute(**34**, 'Сабанеевская улица');

**CALL** AddBusStopOnTheRoute(**34**, 'ЯЗДА');

**CALL** AddBusStopOnTheRoute(**34**, 'Машприбор');

**CALL** AddBusStopOnTheRoute(**34**, 'Залесская улица');

**CALL** AddBusStopOnTheRoute(**34**, 'Средний посёлок');

**CALL** AddBusStopOnTheRoute(**34**, 'Школа № 47');

**CALL** AddBusStopOnTheRoute(**34**, '1-я Больничная улица');

**CALL** AddBusStopOnTheRoute(**34**, 'Школа № 51');

**CALL** AddBusStopOnTheRoute(**34**, 'Нижний посёлок');

**CALL** AddBusStopOnTheRoute(**35**, 'Ярославль-Главный');

**CALL** AddBusStopOnTheRoute(**35**, 'Полиграфкомбинат');

**CALL** AddBusStopOnTheRoute(**35**, 'Дом обуви');

**CALL** AddBusStopOnTheRoute(**35**, 'Улица Кудрявцева');

**CALL** AddBusStopOnTheRoute(**35**, 'Юбилейная площадь');

**CALL** AddBusStopOnTheRoute(**35**, 'Площадь Карла Маркса');

**CALL** AddBusStopOnTheRoute(**35**, 'Советская улица');

**CALL** AddBusStopOnTheRoute(**35**, 'Октябрьская площадь');

**CALL** AddBusStopOnTheRoute(**35**, 'Дачная улица');

**CALL** AddBusStopOnTheRoute(**35**, 'Школа № 50');

**CALL** AddBusStopOnTheRoute(**35**, 'Линейная улица');

**CALL** AddBusStopOnTheRoute(**35**, 'Красноборская улица');

**CALL** AddBusStopOnTheRoute(**35**, 'Проспект Машиностроителей');

**CALL** AddBusStopOnTheRoute(**35**, 'Проезд Доброхотова');

**CALL** AddBusStopOnTheRoute(**35**, 'Кинотеатр Аврора');

**CALL** AddBusStopOnTheRoute(**35**, 'Улица Саукова');

**CALL** AddBusStopOnTheRoute(**35**, 'Улица Папанина');

**CALL** AddBusStopOnTheRoute(**35**, 'Улица Сахарова');

**CALL** AddBusStopOnTheRoute(**35**, 'Школа № 48');

**CALL** AddBusStopOnTheRoute(**35**, 'Хутор');

**CALL** AddBusStopOnTheRoute(**35**, 'Сабанеевская улица');

**CALL** AddBusStopOnTheRoute(**35**, 'ЯЗДА');

**CALL** AddBusStopOnTheRoute(**35**, 'Машприбор');

**END**;

$BODY$;

**ALTER** **PROCEDURE** **public**.filltablebusstopontheroute()

**OWNER** **TO** postgres;

-- заполнить данными таблицу "Остановки на маршрутах"

**CALL** filltablebusstopontheroute();

-- создание, удаление и изменение процедуры добавления одного водителя в таблицу "Водители"

-- DROP PROCEDURE IF EXISTS public.adddriver(character varying, character varying[], date);

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.adddriver(

**IN** "\_fullName" character varying,

**IN** \_category character varying[],

**IN** \_birthday date **DEFAULT** **NULL**::date)

**LANGUAGE** 'sql'

**AS** $BODY$

**insert** **into** **public**."Drivers"("fullName", category, birthday)

**values** ("\_fullName", \_category, \_birthday)

$BODY$;

**ALTER** **PROCEDURE** **public**.adddriver(character varying, character varying[], date)

**OWNER** **TO** postgres;

-- создание, удаление, изменение процедуры заполнения таблицы "Водители" данными

-- DROP PROCEDURE IF EXISTS public.filltabledrivers();

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.filltabledrivers(

)

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**BEGIN**

**CALL** AddDriver('Иванов Олег Васильевич', '{"B", "D"}', 'January 8, 1989');

**CALL** AddDriver('Петрова Елизавета Игоревна', '{"D", "Tm"}', 'August 23, 1994');

**CALL** AddDriver('Каталов Василий Петрович', '{"D"}');

**CALL** AddDriver('Носков Илья Владимирович', '{"D", "Tb"}', 'September 12, 1974');

**CALL** AddDriver('Кириллова Эмма Игнатьевна', '{"B", "D"}', 'May 10, 1990');

**CALL** AddDriver('Шубин Вячеслав Евгеньевич', '{"D"}');

**END**;

$BODY$;

**ALTER** **PROCEDURE** **public**.filltabledrivers()

**OWNER** **TO** postgres;

-- заполнить данными таблицу "Водители"

**CALL** filltabledrivers();

-- создание, удаление и изменение процедуры добавления одного транспортного средства в таблицу "Транспортное средство"

-- DROP PROCEDURE IF EXISTS public.addvehicle(jsonb);

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.addvehicle(

**IN** "\_technicalPassport" jsonb)

**LANGUAGE** 'sql'

**AS** $BODY$

**insert** **into** **public**."Vehicle"("technicalPassport")

**values** ("\_technicalPassport")

$BODY$;

**ALTER** **PROCEDURE** **public**.addvehicle(jsonb)

**OWNER** **TO** postgres;

-- создание, удаление, изменение процедуры заполнения таблицы "Транспортное средство" данными

-- DROP PROCEDURE IF EXISTS public.filltablevehicle();

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.filltablevehicle(

)

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**BEGIN**

**CALL** AddVehicle('{"passportNumber": "76 YX 933716", "registerSine": "X342XM76RUS", "release": 1990, "VIN": "XTA210530L1152417", "brend": "ВАЗ 21053", "engineNumber": "5020 YT4201", "color": "white", "isOnGIBDD": true}'::jsonb);

**CALL** AddVehicle('{"passportNumber": "61 КА 417463", "registerSine": "C104VU76RUS", "release": 1995, "VIN": "WMAH06ZZX6M451416", "brend": "ЛИАЗ 677", "engineNumber": "4664 BT1588", "color": "blue", "isOnGIBDD": true}'::jsonb);

**CALL** AddVehicle('{"passportNumber": "37 46 815308", "registerSine": "H238UV76RUS", "release": 2022, "VIN": "VVCSORXRCS4X00526", "brend": "ЛИАЗ 5292", "engineNumber": "4664 BT1588", "color": "white", "isOnGIBDD": true}'::jsonb);

**CALL** AddVehicle('{"passportNumber": "82 26 066388", "registerSine": "A383BO76RUS", "release": 2008, "VIN": "XTC5320000H026879", "brend": "ЛИАЗ 6213", "engineNumber": "4664 BT1588", "color": "white", "isOnGIBDD": true}'::jsonb);

**CALL** AddVehicle('{"passportNumber": "42 KУ 258404", "registerSine": "H208OY76RUS", "release": 2002, "VIN": "XWB3K32CDAA056459", "brend": "ЛИАЗ 6212", "engineNumber": "4664 BT1588", "color": "gray", "isOnGIBDD": true}'::jsonb);

**CALL** AddVehicle('{"passportNumber": "42 ТХ 126975", "registerSine": "K512RR76RUS", "release": 2015, "VIN": "JMBSREA3A1Z000286", "brend": "ЛИАЗ 4292", "engineNumber": "4664 BT1588", "color": "greeen", "isOnGIBDD": true}'::jsonb);

**END**;

$BODY$;

**ALTER** **PROCEDURE** **public**.filltablevehicle()

**OWNER** **TO** postgres;

-- заполнить данными таблицу "Транспортное средство"

**CALL** filltablevehicle();

-- создание, удаление и изменение процедуры добавления одного рейса в таблицу "Рейсы"

-- DROP PROCEDURE IF EXISTS public.addbustrips(timestamp without time zone, integer, integer, integer, numeric);

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.addbustrips(

**IN** \_timing **timestamp** **without** time **zone**,

**IN** "\_vehicleId" integer,

**IN** "\_driverId" integer,

**IN** "\_routeId" integer,

**IN** \_revenue numeric)

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**BEGIN**

**insert** **into** **public**."BusTrips"(timing, "vehicleId", "driverId", "routeId", revenue)

**values** (\_timing, "\_vehicleId", "\_driverId", "\_routeId", \_revenue)

**ON** CONFLICT **DO** **NOTHING**;

**END**;

$BODY$;

**ALTER** **PROCEDURE** **public**.addbustrips(**timestamp** **without** time **zone**, integer, integer, integer, numeric)

**OWNER** **TO** postgres;

-- создание, удаление, изменение процедуры заполнения таблицы "Рейсы" данными

-- DROP PROCEDURE IF EXISTS public.filltablebustrips();

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.filltablebustrips(

)

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**BEGIN**

**CALL** AddBusTrips('2022-12-08 06:32:00', **2**, **1**, **31**, **0**.**0**);

**CALL** AddBusTrips('2022-12-08 07:50:00', **3**, **4**, **31**, **0**.**0**);

**CALL** AddBusTrips('2022-12-08 09:04:00', **4**, **2**, **31**, **0**.**0**);

**CALL** AddBusTrips('2022-12-08 12:15:00', **2**, **2**, **32**, **0**.**0**);

**CALL** AddBusTrips('2022-12-08 14:05:00', **5**, **5**, **32**, **0**.**0**);

**CALL** AddBusTrips('2022-12-08 15:10:00', **3**, **4**, **32**, **0**.**0**);

**CALL** AddBusTrips('2022-12-08 17:40:00', **6**, **3**, **33**, **0**.**0**);

**CALL** AddBusTrips('2022-12-08 17:28:00', **1**, **1**, **33**, **0**.**0**);

**CALL** AddBusTrips('2022-12-08 19:28:00', **6**, **2**, **33**, **0**.**0**);

**CALL** AddBusTrips('2022-12-08 21:00:00', **2**, **5**, **34**, **0**.**0**);

**CALL** AddBusTrips('2022-12-08 22:05:00', **4**, **4**, **34**, **0**.**0**);

**CALL** AddBusTrips('2022-12-08 22:35:00', **2**, **3**, **34**, **0**.**0**);

**END**

$BODY$;

**ALTER** **PROCEDURE** **public**.filltablebustrips()

**OWNER** **TO** postgres;

-- заполнить данными таблицу "Рейсы"

**CALL** filltablebustrips();

А также создадим процедуру для обновления значения атрибута «выручки» в таблице «рейсы», чтобы диспетчер рейсов по окончанию рабочего дня мог вносить выручку с каждого рейса.

-- создание, удаление и изменение процедуры для записи дохода за рейс

-- DROP PROCEDURE IF EXISTS public.updaterevenue(timestamp without time zone, numeric);

**CREATE** **OR** **REPLACE** **PROCEDURE** **public**.updaterevenue(

**IN** \_timing **timestamp** **without** time **zone**,

**IN** \_revenue numeric)

**LANGUAGE** 'plpgsql'

**AS** $BODY$

**BEGIN**

**UPDATE** **public**."BusTrips" **SET** "revenue" = \_revenue **WHERE** "timing" = \_timing;

**END**;

$BODY$;

**ALTER** **PROCEDURE** **public**.updaterevenue(**timestamp** **without** time **zone**, numeric)

**OWNER** **TO** postgres;

-- обновить колонку дохода за определённый рейс

**CALL** updaterevenue('2022-12-08 17:28:00', **3028**)

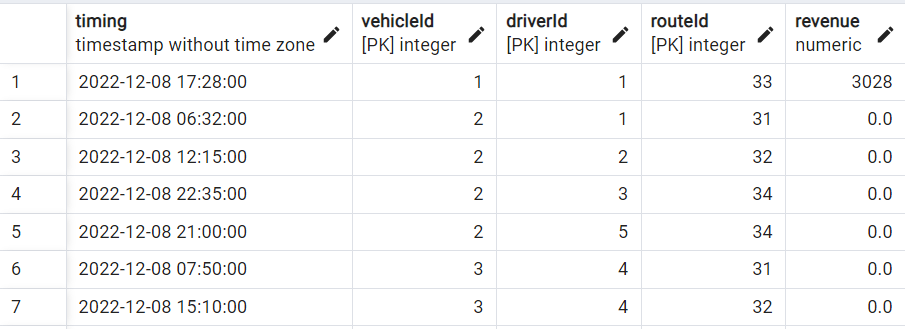


Рисунок 2 - Пример таблицы "Рейсы"

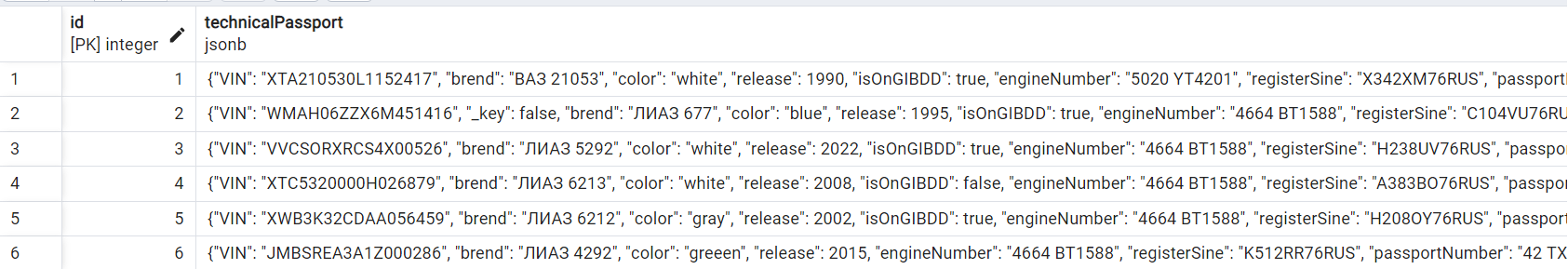


Рисунок 3 - Пример таблицы "Транспортное средство"

**Работа с представлениями (VIEW)**

Создадим два представления, в которых будет видна вся основная общая информация по рейсам, а именно дата/время, номер маршрута, водитель, количество остановок, остановки, номер машины и т. д.

-- создание, удаление и изменение представления (VIEW) для просмотра остановок на маршрутах

-- DROP VIEW public.busstopsroute;

**CREATE** **OR** **REPLACE** **VIEW** **public**.busstopsroute

**AS**

**SELECT** "Route"."numberRoute" **AS** "Номер маршрута",

**count**("Route"."numberOfBusstop") **AS** "Количество остановок",

**avg**("Route".**length**) **AS** "Длина маршрута",

array\_agg("BusStopOnTheRoute"."busStopName") **AS** "Остановки",

array\_agg("BusStop".coordinates) **AS** "Координаты остановок"

**FROM** "Route"

**LEFT** **JOIN** "BusStopOnTheRoute" **ON** "Route".id = "BusStopOnTheRoute"."idRoute"

**LEFT** **JOIN** "BusStop" **ON** "BusStopOnTheRoute"."busStopName"::text = "BusStop"."busstopName"::text

**GROUP** **BY** "Route"."numberRoute";

**ALTER** **TABLE** **public**.busstopsroute

**OWNER** **TO** postgres;

-- создание, удаление и изменение представления (VIEW) для просмотра рейсов

-- DROP VIEW public.bustrips;

**CREATE** **OR** **REPLACE** **VIEW** **public**.bustrips

**AS**

**SELECT** "BusTrips".timing **AS** "Дата рейса",

busstopsroute."Номер маршрута",

busstopsroute."Количество остановок",

busstopsroute."Остановки",

"Drivers"."fullName" **AS** "Водитель",

busstopsroute."Длина маршрута",

"Vehicle"."technicalPassport" ->> 'registerSine'::text **AS** "Номер машины"

**FROM** "BusTrips"

**LEFT** **JOIN** "Vehicle" **ON** "BusTrips"."vehicleId" = "Vehicle".id

**LEFT** **JOIN** "Drivers" **ON** "BusTrips"."driverId" = "Drivers".id

**LEFT** **JOIN** "Route" **ON** "BusTrips"."routeId" = "Route".id

**LEFT** **JOIN** busstopsroute **ON** "Route"."numberRoute"::text = busstopsroute."Номер маршрута"::text;

**ALTER** **TABLE** **public**.bustrips

**OWNER** **TO** postgres;



Рисунок 4 - Пример представления

# **Работа с данными**

Технический паспорт транспортного средства представляет собой документ в формате jsonb. PostgreSQL предлагает два типа для хранения данных JSON: json и jsonb. Типы данных json и jsonb принимают на вход практически идентичные наборы значений. Основное практическое отличие заключается в эффективности. Тип данных json хранит точную копию входного текста, которую функции обработки должны повторно анализировать при каждом выполнении; в то время как данные jsonb хранятся в декомпозированном двоичном формате, что немного замедляет ввод из-за дополнительных затрат на преобразование, но значительно ускоряет обработку, поскольку повторный анализ не требуется. jsonb также поддерживает индексацию, что может быть значительным преимуществом.

Далее показаны запросы, с помощью которых можно работать с форматом jsonb и поиск по json, изменять значения ключей или удалять ключи.

-- пример поиска по json

**SELECT** "technicalPassport"->'color' **as** color **FROM** **public**."Vehicle" **WHERE** "technicalPassport" ? 'color' = **true** **AND** "id" = **4**;

**SELECT** "technicalPassport"->'VIN' **as** VIN **FROM** **public**."Vehicle" **WHERE** "technicalPassport" ? 'VIN' = **true**;

**SELECT** "technicalPassport"->'brend' **as** brend, "technicalPassport"->'release' **as** "release" **FROM** **public**."Vehicle" **WHERE** "technicalPassport" ? 'brend' = **true** **AND** "technicalPassport"->>'brend'::text **LIKE** '%ЛИАЗ%';

-- пример обновления значение ключа json

**UPDATE** **public**."Vehicle" **SET** "technicalPassport" = jsonb\_set("technicalPassport", '{isOnGIBDD}', 'false') **WHERE** "technicalPassport" ? 'isOnGIBDD' = **true** **AND** "id" = **4**;

-- пример удаление ключа из json

**UPDATE** **public**."Vehicle" **SET** "technicalPassport" = "technicalPassport" - 'isOnGIBDD' **WHERE** "technicalPassport" ? 'isOnGIBDD' = **true** **AND** "id" = **6**;

-- пример удаления строки из таблицы "Транспортное средство"

-- DELETE FROM public."Vehicle" WHERE "technicalPassport" ->> 'color' = 'brown';

Ниже приведены примеры запросов по представлению рейсов:

-- примеры запросов по VIEW просмотра рейсов

-- все рейсы маршрута номер 12

**SELECT** \* **FROM** **public**."bustrips" **WHERE** "Номер маршрута" = '12';

-- посмотреть все даты рейсов водителя 'Иванов Олег Васильевич'

**SELECT** "Водитель", array\_agg("Дата рейса") **FROM** **public**."bustrips" **WHERE** "Водитель" = 'Иванов Олег Васильевич' **GROUP** **BY** "Водитель";

-- посмотреть сколько водителей будут в конкретную дату на определённой машине

**SELECT** "Номер машины", **count**("Водитель") **from** **public**."bustrips" **WHERE** "Номер машины"='C104VU76RUS' **AND** **CAST**("Дата рейса" **AS** DATE) = '2022-12-08' **GROUP** **BY** "Номер машины";

-- количество остановок, которые проедет водитель Петрова Елизавета Игоревна

**SELECT** **sum**("Количество остановок") **FROM** **public**."bustrips" **WHERE** "Водитель" = 'Петрова Елизавета Игоревна' **GROUP** **BY** "Водитель";

Здесь представлены примеры работы с методами POSTGIS.

* ST\_Distance – для типов геометрии возвращает минимальное двумерное декартово (плоское) расстояние между двумя геометриями в единицах проекции (единицы пространственной ссылки).
* ST\_Intersection – сравнивает две геометрии и возвращает true, если они пересекаются или точки пересечения.
* ST\_AsText – возвращает представление OGC Well-Known Text (WKT) геометрии/географии, то есть координаты точек как «POINT (0 0)»

-- POSTGIS

-- ближайшие пять остановок к остановке 'Студенческий городок'

**SELECT** t2."busstopName", ST\_Distance(t1.coordinates, t2.coordinates) **AS** nearest **FROM** "BusStop" t1, "BusStop" t2 **WHERE** t1."busstopName" = 'Студенческий городок' **and** t2."busstopName" <>

t1."busstopName" **ORDER** **BY** ST\_Distance(t1.coordinates, t2.coordinates) **LIMIT** **5**;

-- пересечение точек двух маршрутов

**SELECT** ST\_AsText(ST\_Intersection(**unnest**(t1."Координаты остановок"), **unnest**(t2."Координаты остановок"))) **AS** "intersection" **FROM** **public**."busstopsroute" t1, **public**."busstopsroute" t2 **WHERE** t1."Номер маршрута" = '39' **and** t2."Номер маршрута" =

'22c';

# **Создание маршрута в формате KML**

KML - язык разметки на основе XML для представления трёхмерных геопространственных данных. То есть мы можем построить маршрут (линию) по координатам остановок и создать текст в формате .kml, который мы можем импортировать в какой-нибудь сервис карт и нам отобразиться наш маршрут, если мы добавим названия остановок и стиль отображения точек на карте.

Поэтому создадим функцию, которая будет импортировать наш маршрут, состоящий из координат остановок в .kml формат.

* ST\_MakeLine – создаёт LineString, содержащую точки геометрии Point, MultiPoint или LineString.
* ST\_AsKML – возвращает геометрию как элемент языка разметки KML. Максимальное количество знаков после запятой по умолчанию — 15, пространство имен по умолчанию — без префикса.

-- DROP FUNCTION IF EXISTS public.getkmlroute(character varying);

**CREATE** **OR** **REPLACE** **FUNCTION** **public**.getkmlroute(

\_routenumber character varying)

**RETURNS** text

**LANGUAGE** 'sql'

COST **100**

**VOLATILE** PARALLEL UNSAFE

**AS** $BODY$

**SELECT** ST\_AsKML(ST\_MakeLine("Координаты остановок"), **4326**) **as** kml **FROM** **public**."busstopsroute" **WHERE** "Номер маршрута" = \_routeNumber;

$BODY$;

**ALTER** **FUNCTION** **public**.getkmlroute(character varying)

**OWNER** **TO** postgres;

-- формат KML для каждого маршрута

**COPY**(**SELECT** \* **FROM** getkmlroute('22')) **to** 'D:/owl/demid/semestr3/bd/22.kml';

**COPY**(**SELECT** \* **FROM** getkmlroute('22c')) **to** 'D:/owl/demid/semestr3/bd/22c.kml';

**COPY**(**SELECT** \* **FROM** getkmlroute('12')) **to** 'D:/owl/demid/semestr3/bd/12.kml';

**COPY**(**SELECT** \* **FROM** getkmlroute('30')) **to** 'D:/owl/demid/semestr3/bd/30.kml';

**COPY**(**SELECT** \* **FROM** getkmlroute('39')) **to** 'D:/owl/demid/semestr3/bd/39.kml';

Пример полученного KML маршрута номер «22»:

<LineString><coordinates>-127.48822742503624,0.00035942845210712476 -127.48822748699673,0.00035948401147097015 -127.48822752048552,0.00035957121986118356 -127.48822748849297,0.00035965022966542165 -127.48822750412646,0.0003597300060935259 -127.48822744987982,0.00035996747743529734 -127.48822744979918,0.0003599671797957237 -127.48822750409977,0.00036010351675061237 -127.48822750377724,0.0003601035167507725 -127.48822758115621,0.0003602008358466335 -127.488227607863,0.00036023527183472946 -127.48822772691025,0.0003602545281566424 -127.48822778282343,0.00036032817138477143 -127.48822777835292,0.00036040842585008083 -127.48822777216232,0.00036052469470949336 -127.48822776879375,0.0003605877221541963 -127.48822776655403,0.00036062366439349965</coordinates></LineString>

# **Роли доступа**

Нам необходимо разграничить доступ к нашим данным для безопасности. Создадим три новые роли: администратор, диспетчер рейсов и гость. У каждого будет разный уровень доступа. У гостя – минимальный, только на просмотр таблиц и представлений. У администратора полный доступ. А у диспетчера рейсов средний – просмотр, изменения таблиц и представлений и работа с процедурами и функциями.

-- создание и удаление роли "Администратор"

-- DROP ROLE IF EXISTS administrator;

**CREATE** **ROLE** administrator **WITH**

LOGIN

SUPERUSER

INHERIT

**CREATEDB**

CREATEROLE

REPLICATION

**ENCRYPTED** PASSWORD 'SCRAM-SHA-256$4096:u/zhpvwjKDS8DDmDplP2Pg==$SDO2uNG3IKuH1fm9SPtrcB6a/cvrxRBhzFTmHU3a9kc=:CS5z8eCeFip9R75uMmDGrxGBm7HZg9E0hfJI9eMSjZE='; (**admin**)

-- создание и удаление роли "Диспетчер автобуса"

-- DROP ROLE IF EXISTS bus\_dispatcher;

**CREATE** **ROLE** bus\_dispatcher **WITH**

LOGIN

NOSUPERUSER

NOINHERIT

**NOCREATEDB**

NOCREATEROLE

NOREPLICATION

**ENCRYPTED** PASSWORD 'SCRAM-SHA-256$4096:/rim6RzEn3a1YkyBAzTGkQ==$iYMXTShBGi5YBbKk9IdEwWO8TFxqyO2gYJqj75YF0ng=:8UbM+fjp3vxdHlSCnvmRPpLYwwNASAXnSYiU7tYPkaM=';

-- создание и удаление роли "Гость"

-- DROP ROLE IF EXISTS guest;

**CREATE** **ROLE** guest **WITH**

NOLOGIN

NOSUPERUSER

NOINHERIT

**NOCREATEDB**

NOCREATEROLE

NOREPLICATION;

-- предоставить доступ на чтение таблиц и последовательностей гостю

**GRANT** **SELECT** **ON** SEQUENCE **public**."Drivers\_id\_seq", **public**."Route\_id\_seq", **public**."Vehicle\_id\_seq" **TO** guest;

**GRANT** **SELECT** **ON** **TABLE** **public**."BusStop", **public**."BusStopOnTheRoute", **public**."BusTrips", **public**."Drivers", **public**."Route", **public**."Vehicle", **public**.busstopsroute, **public**.bustrips **TO** guest;

-- предоставить доступ на запуск функций и процедур, на чтение, удаление, обновление и заполнение таблиц диспетчеру автобусов

**GRANT** **EXECUTE** **ON** **FUNCTION** **public**.getkmlroute(\_routenumber character varying) **TO** bus\_dispatcher;

**GRANT** **EXECUTE** **ON** **PROCEDURE** **public**.addbusstop(**IN** "\_busstopName" character varying, **IN** \_coordinates geometry),

**public**.addbusstopontheroute(**IN** "\_idRoute" integer, **IN** "\_busStopName" character varying),

**public**.addbustrips(**IN** \_timing **timestamp** **without** time **zone**, **IN** "\_vehicleId" integer, **IN** "\_driverId" integer, **IN** "\_routeId" integer, **IN** \_revenue numeric),

**public**.adddriver(**IN** "\_fullName" character varying, **IN** \_category character varying[], **IN** \_birthday date),

**public**.addroute(**IN** "\_numberRoute" character varying, **IN** "\_numberOfBusstop" integer, **IN** \_length real),

**public**.addvehicle(**IN** "\_technicalPassport" jsonb), **public**.updaterevenue(**IN** \_timing **timestamp** **without** time **zone**, **IN** \_revenue numeric) **TO** bus\_dispatcher;

**GRANT** **SELECT**, **UPDATE** **ON** SEQUENCE **public**."Drivers\_id\_seq", **public**."Route\_id\_seq", **public**."Vehicle\_id\_seq" **TO** bus\_dispatcher;

**GRANT** **SELECT**, **UPDATE**, **INSERT**, **DELETE** **ON** **TABLE** **public**."BusStop", **public**."BusStopOnTheRoute", **public**."BusTrips",

**public**."Drivers", **public**."Route", **public**."Vehicle", **public**.busstopsroute, **public**.bustrips **TO** bus\_dispatcher;