**Отчет о выполненной лабораторной работе**

**Тимошенко Никита Михайлович 5 группа 2 курс**

**Условие лабораторной работы:**

1. Разработать физическую модель данных, т.е. указать типы и размеры полей, а также необходимые ограничения целостности; создать таблицы в СУБД MS SQL Server. Определить обязательные и ключевые поля. Задать значения по умолчанию. Определить условия на значение, задать связи между таблицами (требования ссылочной целостности). Задать каскадное удаление и обновление, где необходимо. Требования:
   * определить условие на значение для всех числовых полей (кроме первичных и внешних ключей);
   * определить условие на значение для текстового поля, используя функцию Like, например, потребовать ввод символа @ в поле email;
   * для каждой таблицы определить по крайней мере одно обязательное поле.
2. Разработать индексы:
   * для каждой из таблиц, кроме вспомогательных (справочников) и хранящих небольшой объем данных (до 100 записей), определить минимум один вторичный индекс;
   * определить минимум один составной индекс.
3. Ввести данные.
4. Научиться применять поиск, сортировку и фильтрацию данных средствами MS SQL Management Studio.

**Цель БД:** Организация хранения и систематизации логистической информации о грузоперевозках с участием таких сущностей, как Водители, Транспортные средства, Маршруты, Рейсы и Техническое обслуживание.

**Задачи БД:**

1. Регистрация и учет водительского состава предприятия (ФИО, паспортные данные).
2. Ведение справочника типов транспорта с указанием их технических характеристик, таких как грузоподъемность.
3. Учет транспортных средств логистической сети, включая контроль их пробега и закрепление за определенными типами транспорта.
4. Хранение информации о маршрутах, включая пункты отправления, назначения и точное расстояние.
5. Планирование и протоколирование рейсов, объединяющих конкретный автомобиль, водителя и маршрут на определенную дату.
6. Ведение журнала технического обслуживания для контроля состояния машин и учета финансовых затрат на ремонт.

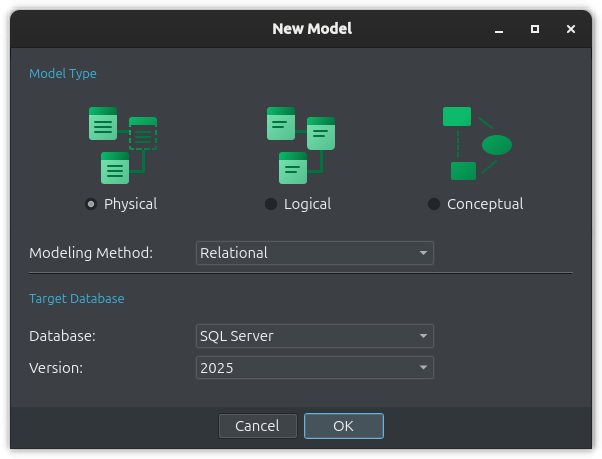
**Для решения поставленных задач были выделены следующие сущности:**

* Drivers — (driver\_id, name, surname, patronymic, passport\_id)
* Vehicle\_types — (type\_id, type\_name, capacity)
* Vehicles — (vehicle\_id, plate\_number, mileage, type\_id)
* Routes — (route\_id, start, end, distance)
* Trips — (trip\_id, vehicle\_id, driver\_id, route\_id, departure\_date)
* Maintenance — (maintenance\_id, vehicle\_id, maintenance\_date, description, cost)

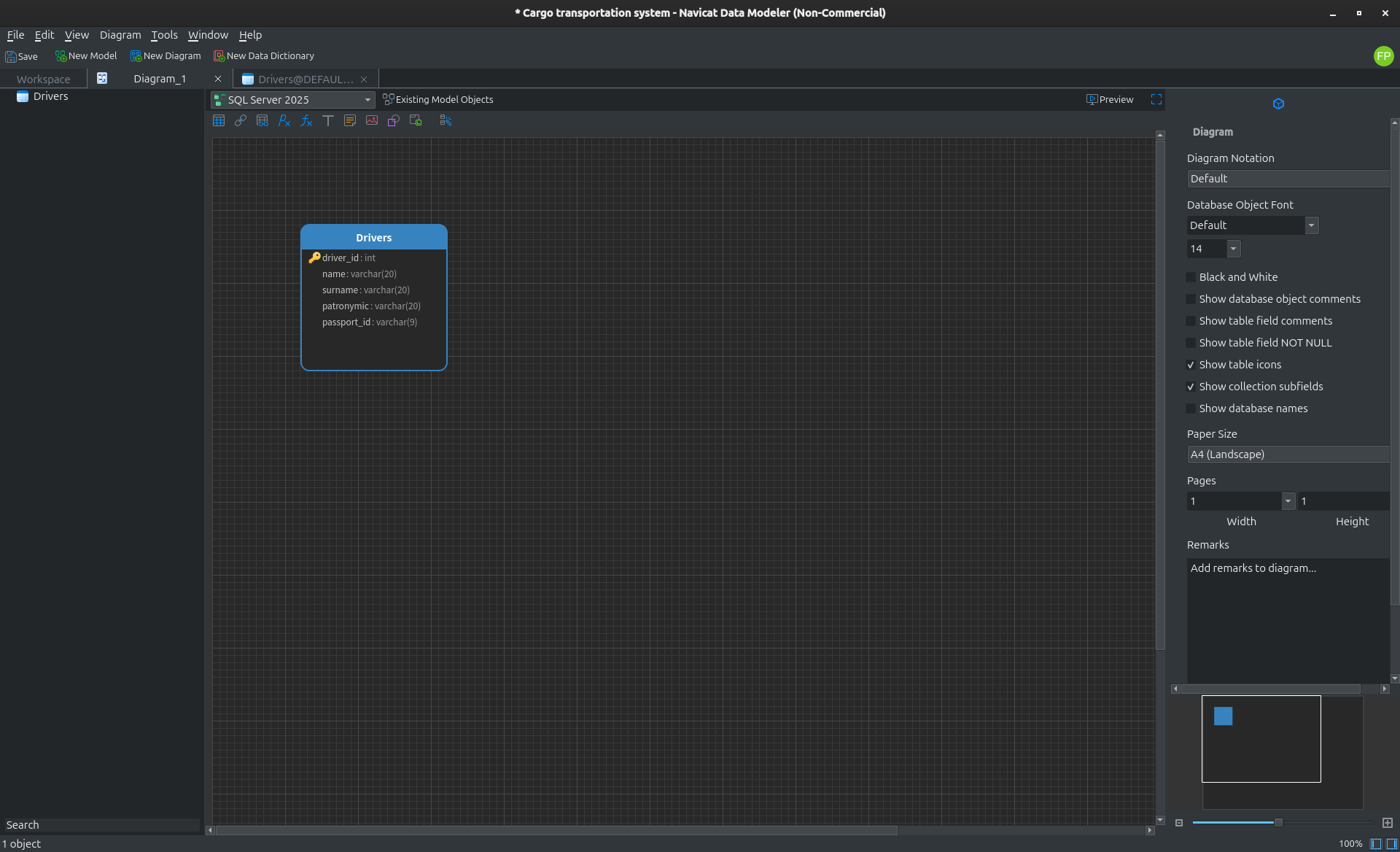
**Выполнение:**

Работа была выполнена с использованием Navicat Data Modeler 4 и DataGrip.

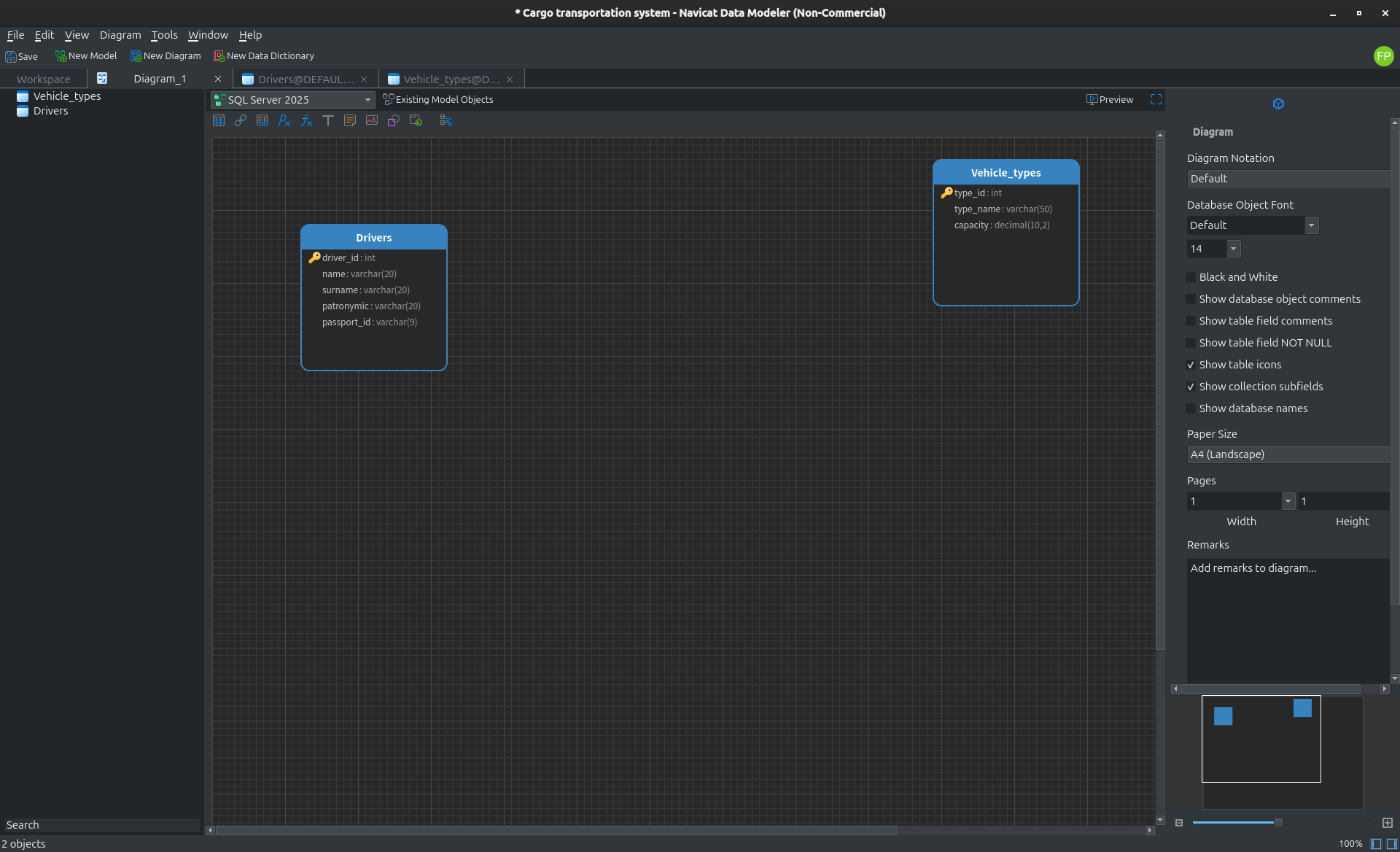
Создаем физическую схему:



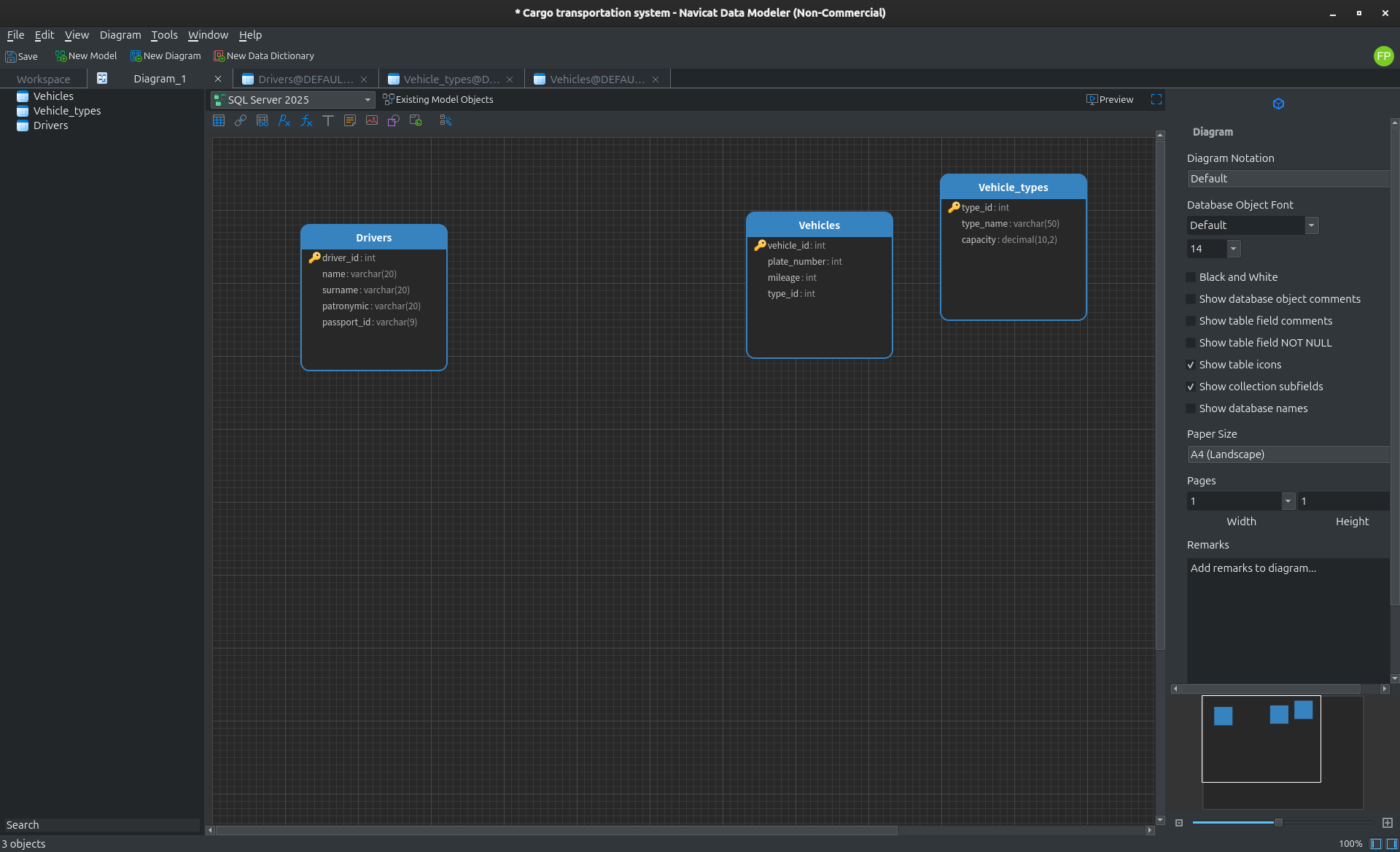
Создаю первую сущность – Drivers. Один Primary Key, остальные атрибуты пока обычные. Выбираю подходящие типы данных для атрибутов. Все поля кроме отчества – patronymic – делаю NOT NULL.



Добавляю следующую сущность – Vehicle\_types. Она будет хранить типы транспортных средств и на нее будет ссылаться следующая сущность.



Следующая сущность – Vehicles. В неё добавлю атрибут type\_id, который и будет ссылаться на Vehicle\_types.

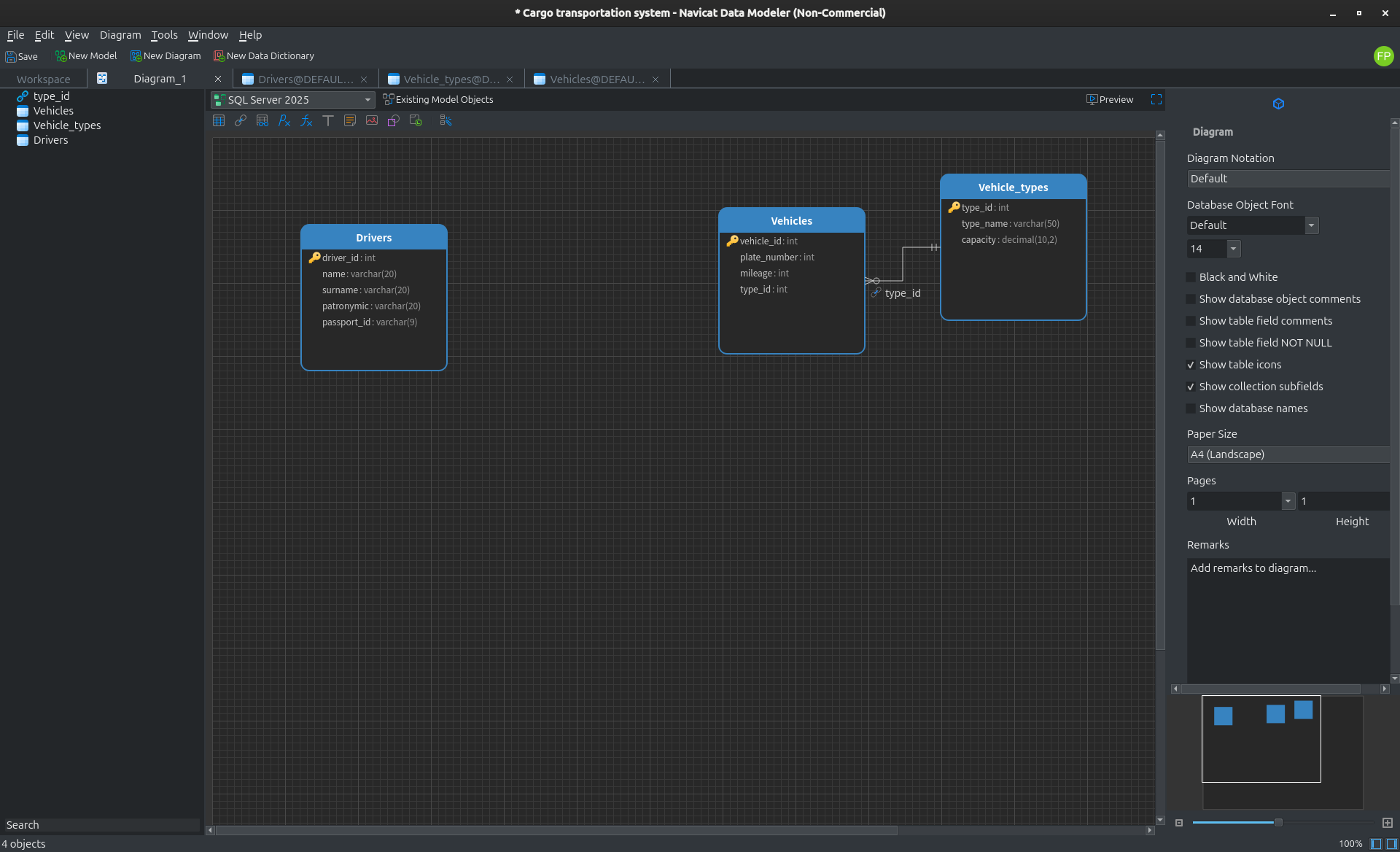


Далее добавляю внешний ключ. Я выбрал NO ACTION on Delete, чтобы обезопасить данные таблиц (при попытке удалить запись из базы пользователь получит ошибку, пока не удалит то, на что ссылается этот FK). В on Update был выбран вариант CASCADE, что позволит без проблем модифицировать записи в таблице, сохраняя связи. Заранее могу отметить, что из данных соображений для всех последующих таблиц будут выбраны те же самые настройки on Update и on Delete.

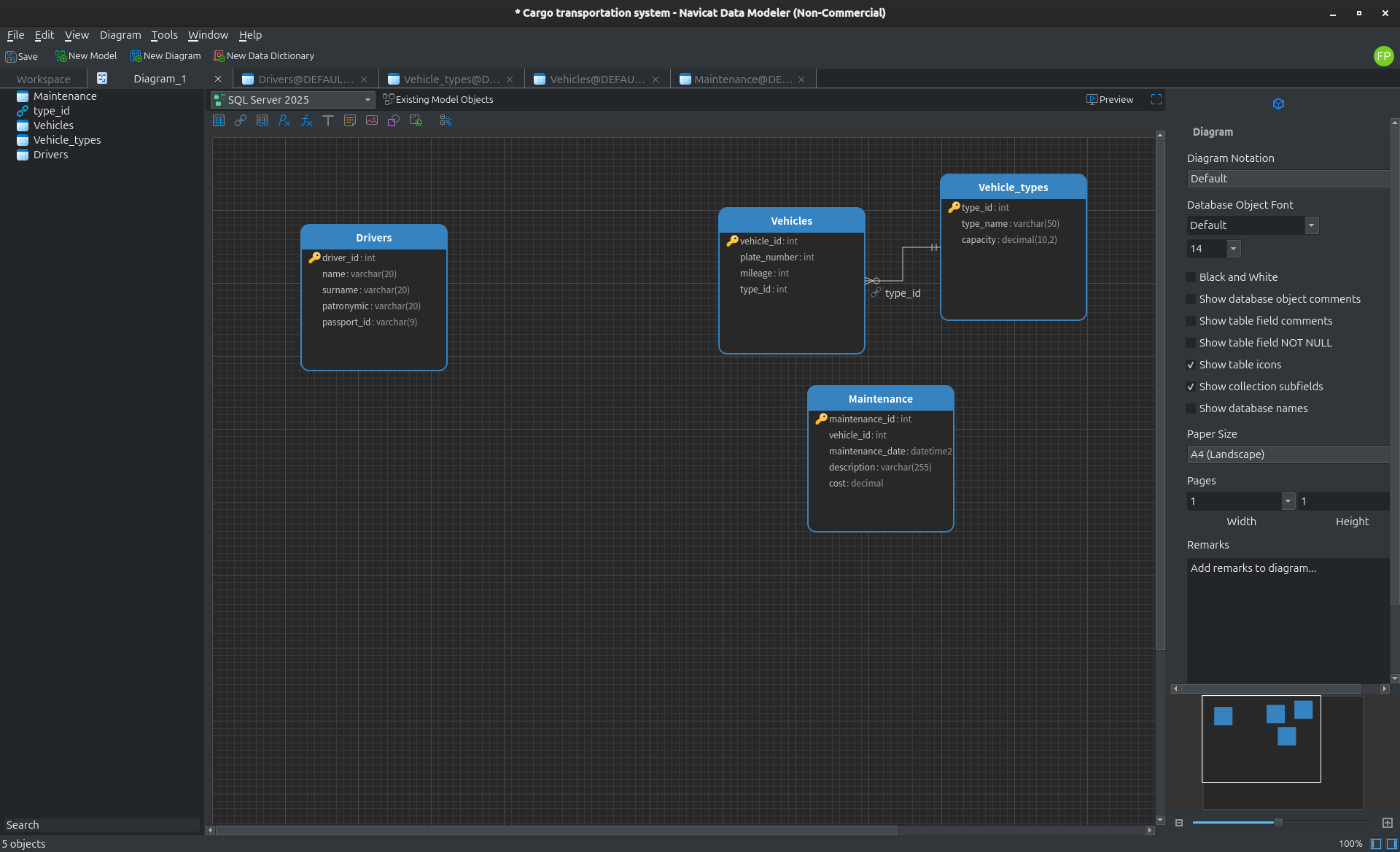
Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

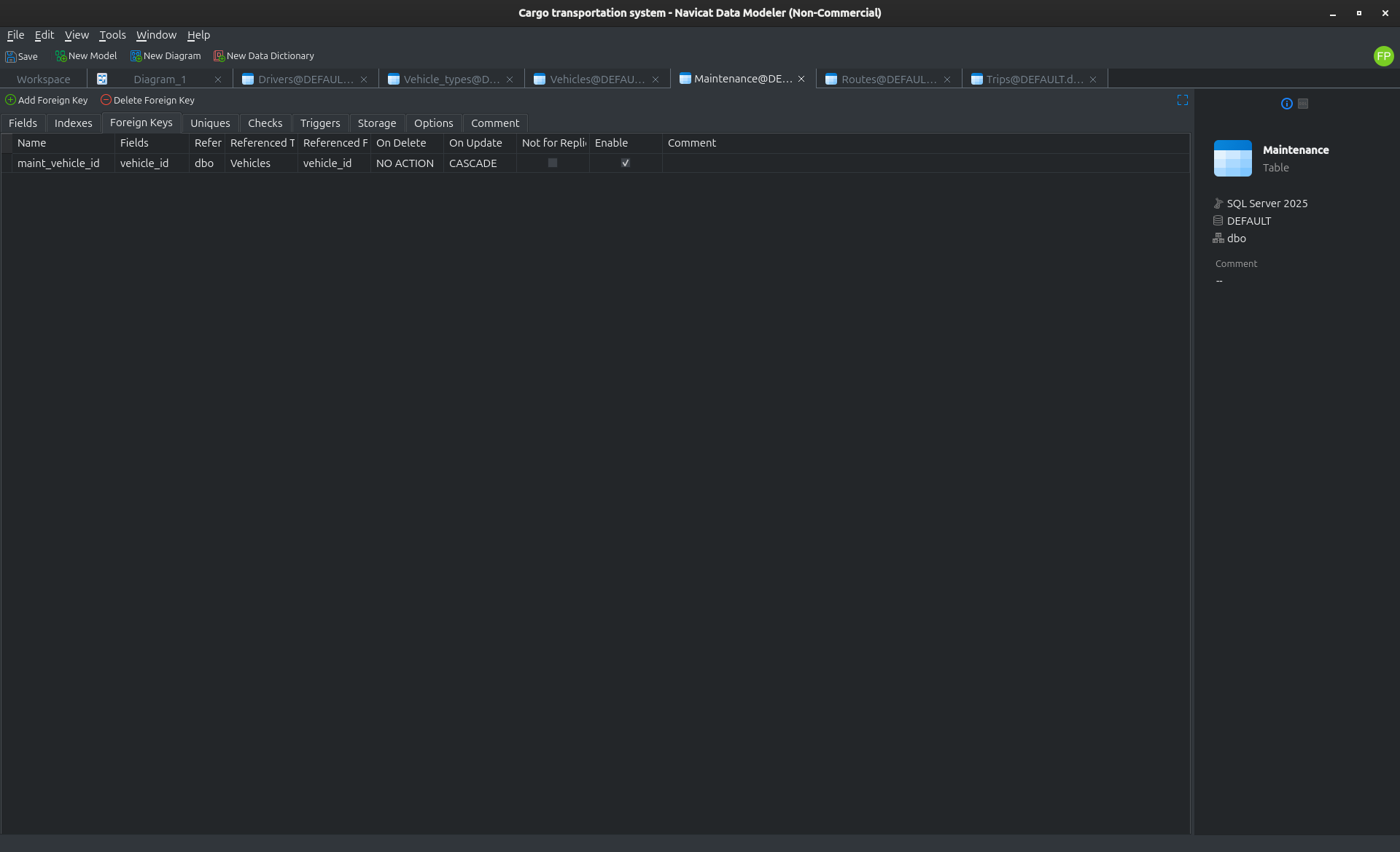
После добавления внешнего ключа на схеме появилась связь. Настраиваю кардинальность связи: со стороны Vehicle\_types один и только один, со стороны Vehicles – ноль или много. Логически это означает, что может не быть, а может быть много машин какого-то указанного типа, но у каждой машины тип обязательно указан и обязательно один.



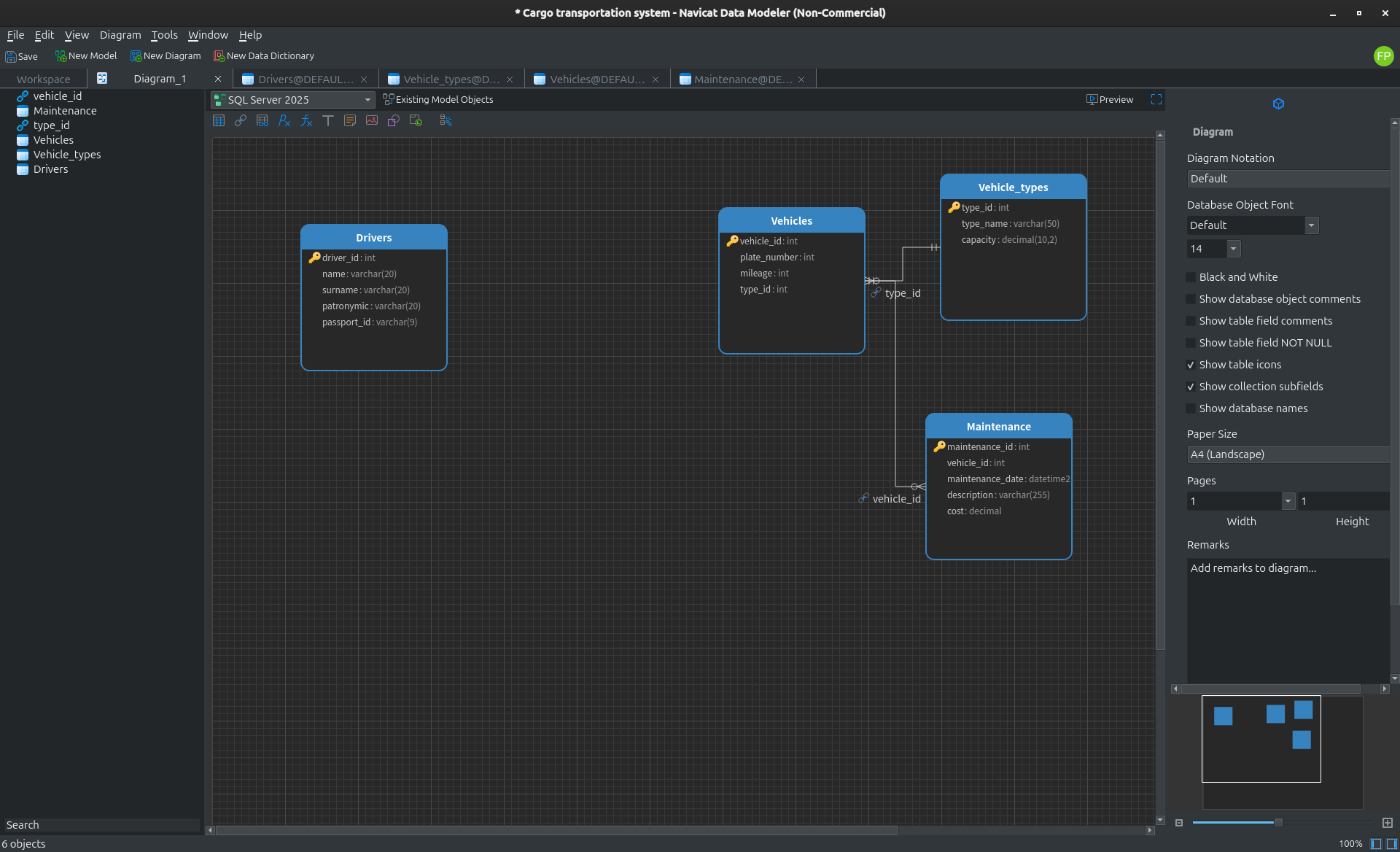
Добавляю следующую сущность – Maintenance. Она будет хранить записи об обслуживании транспортных средств.



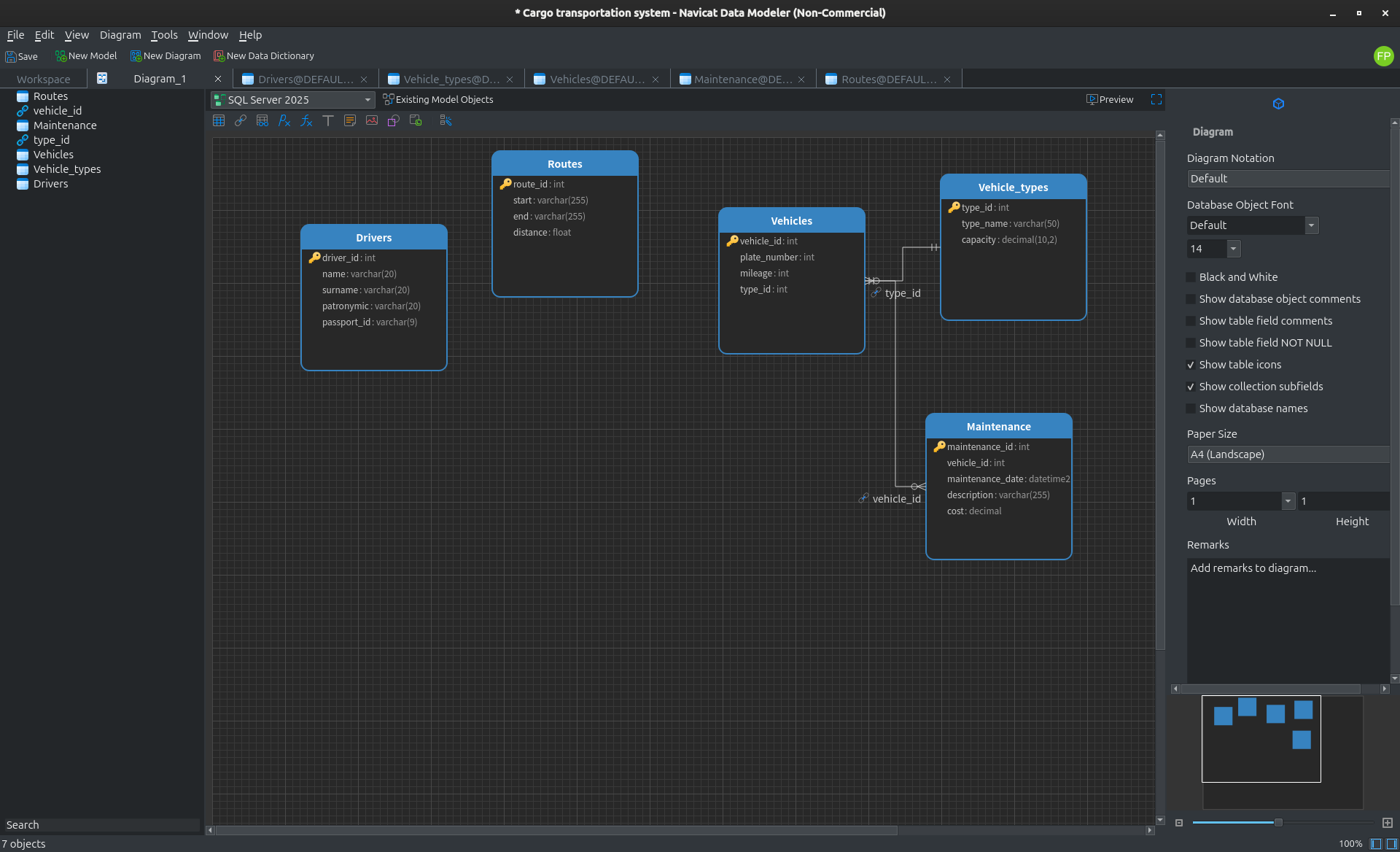
Добавляю внешний ключ – vehicle\_id – для связи с таблицей Vehicles.



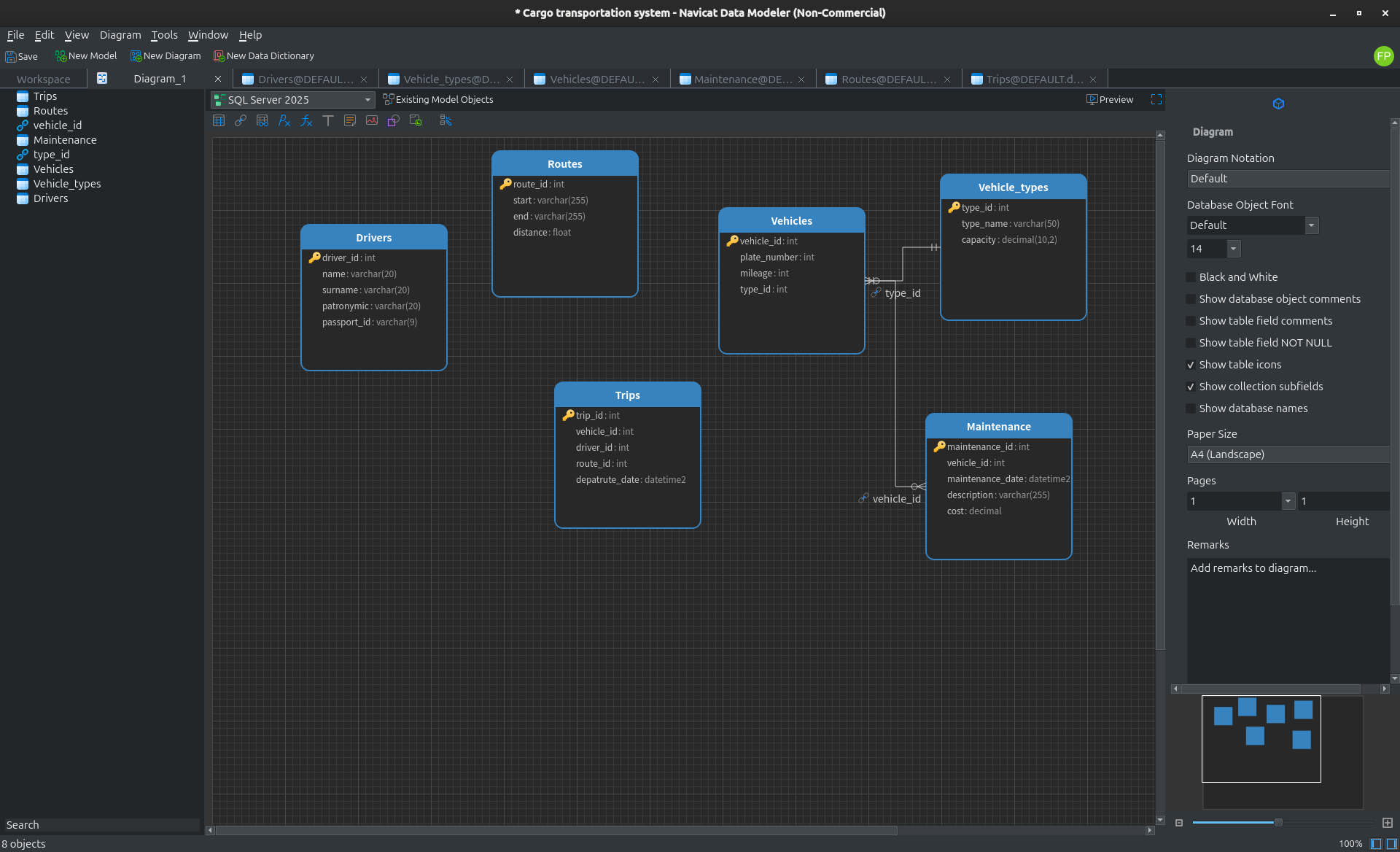
После добавления внешнего ключа на схеме появилась связь. Аналогично настраиваем кардинальность – Vehicles один и только один, Maintenance – ноль или много.



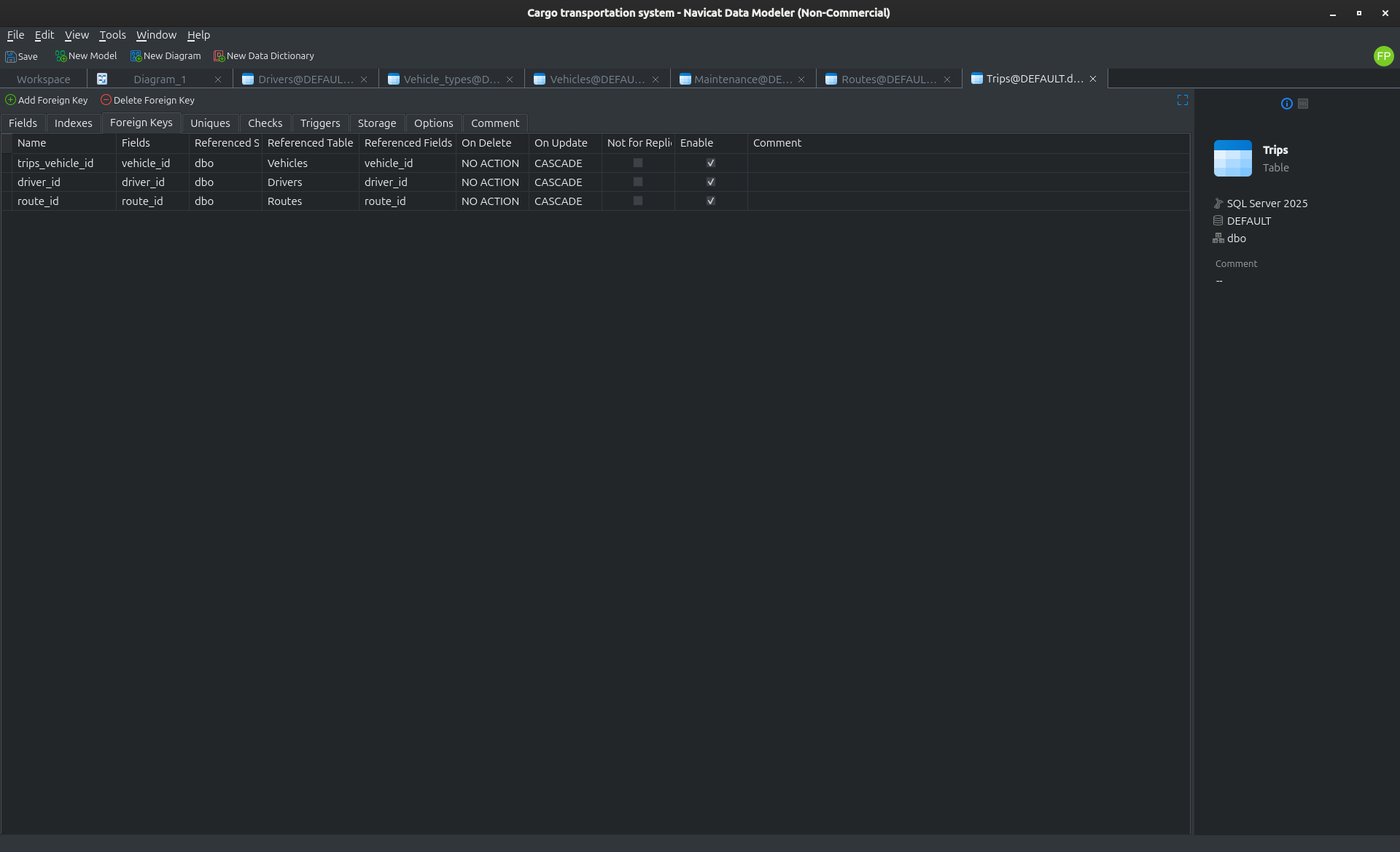
Следующая сущность – Routes. Будет хранить доступные маршруты.



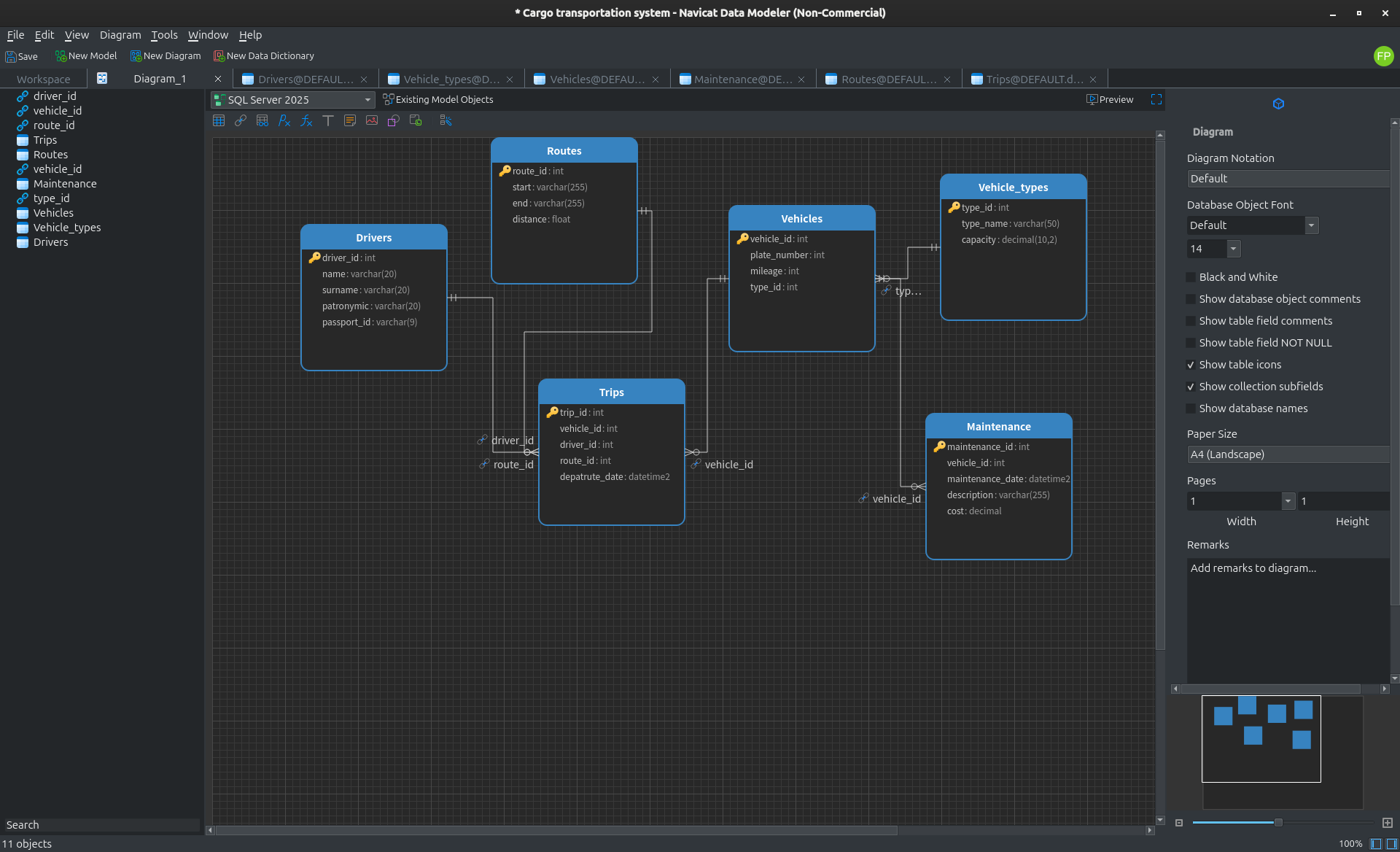
Последняя сущность – Trips. Она будет хранить информацию о поездках конкретных водителей, на конкретных машинах по конкретным маршрутам.



Добавляю 3 внешних ключа, которые обеспечат нужную мне связь

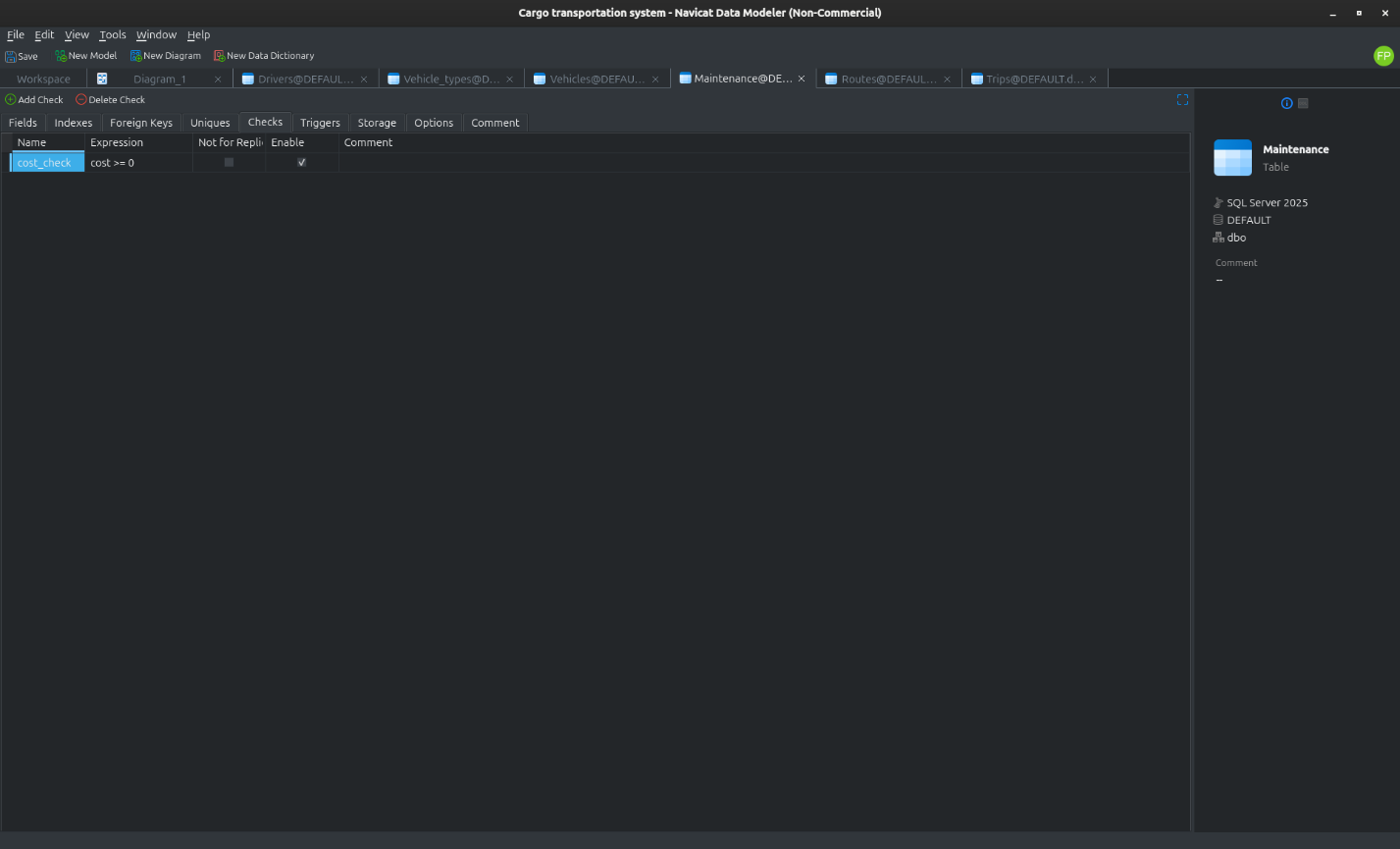


После добавления внешних ключей на схеме отобразились связи. Снова настраиваем кардинальность: Routes, Drivers, Trips – один и только один, Trips – ноль или много. Логика та же – в маршруте может быть только один маршрут, водитель или транспортное средство, но у каждого водителя, транспортного средства или маршрута может быть ноль или много поездок.



Далее добавляю проверки на атрибуты. Сначала ввожу ограничения для численных параметров через вкладку Checks в программе:

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

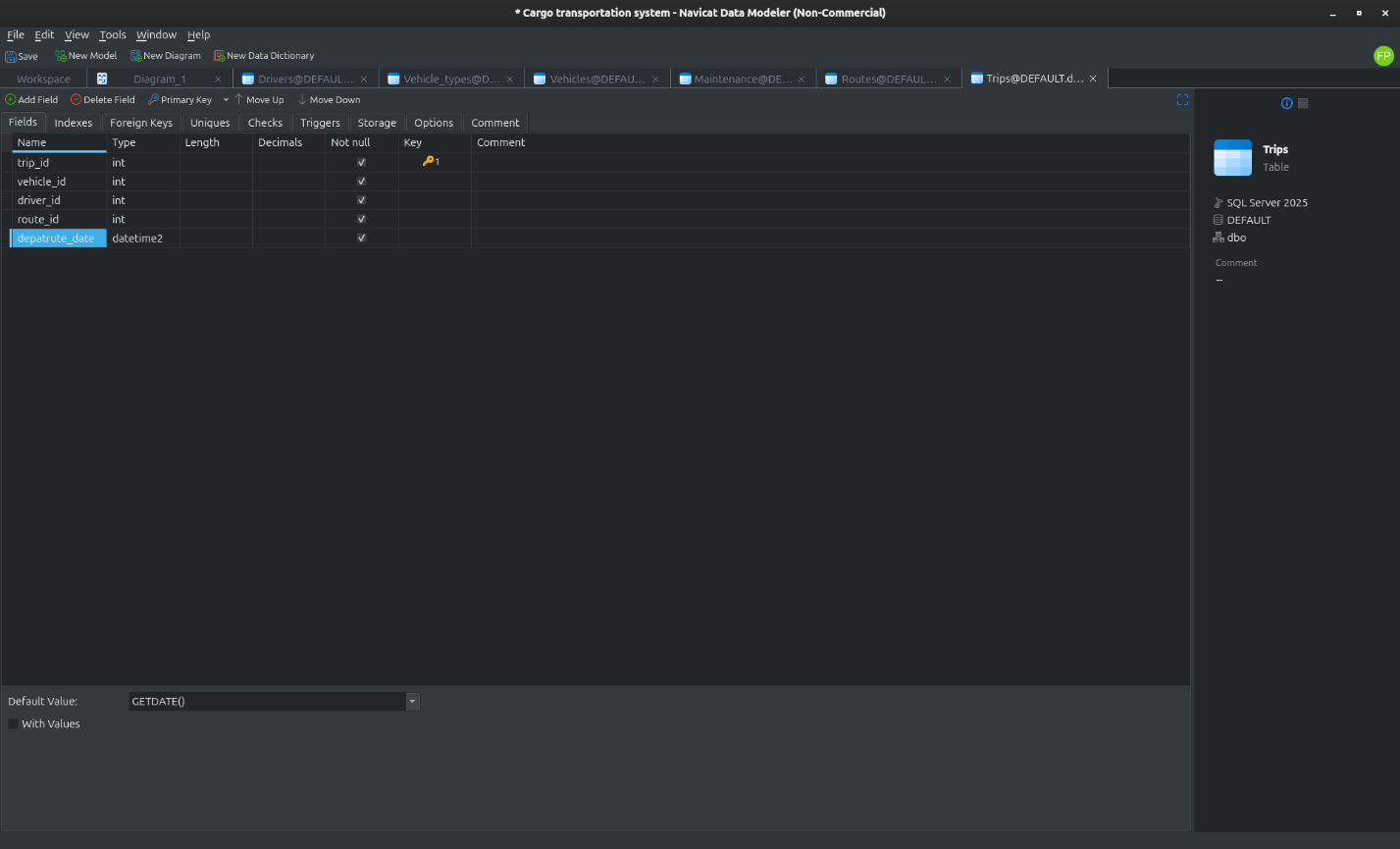
Далее для соответствия условию лабораторной добавляю два форматированных поля – passport\_id и plate\_number:

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

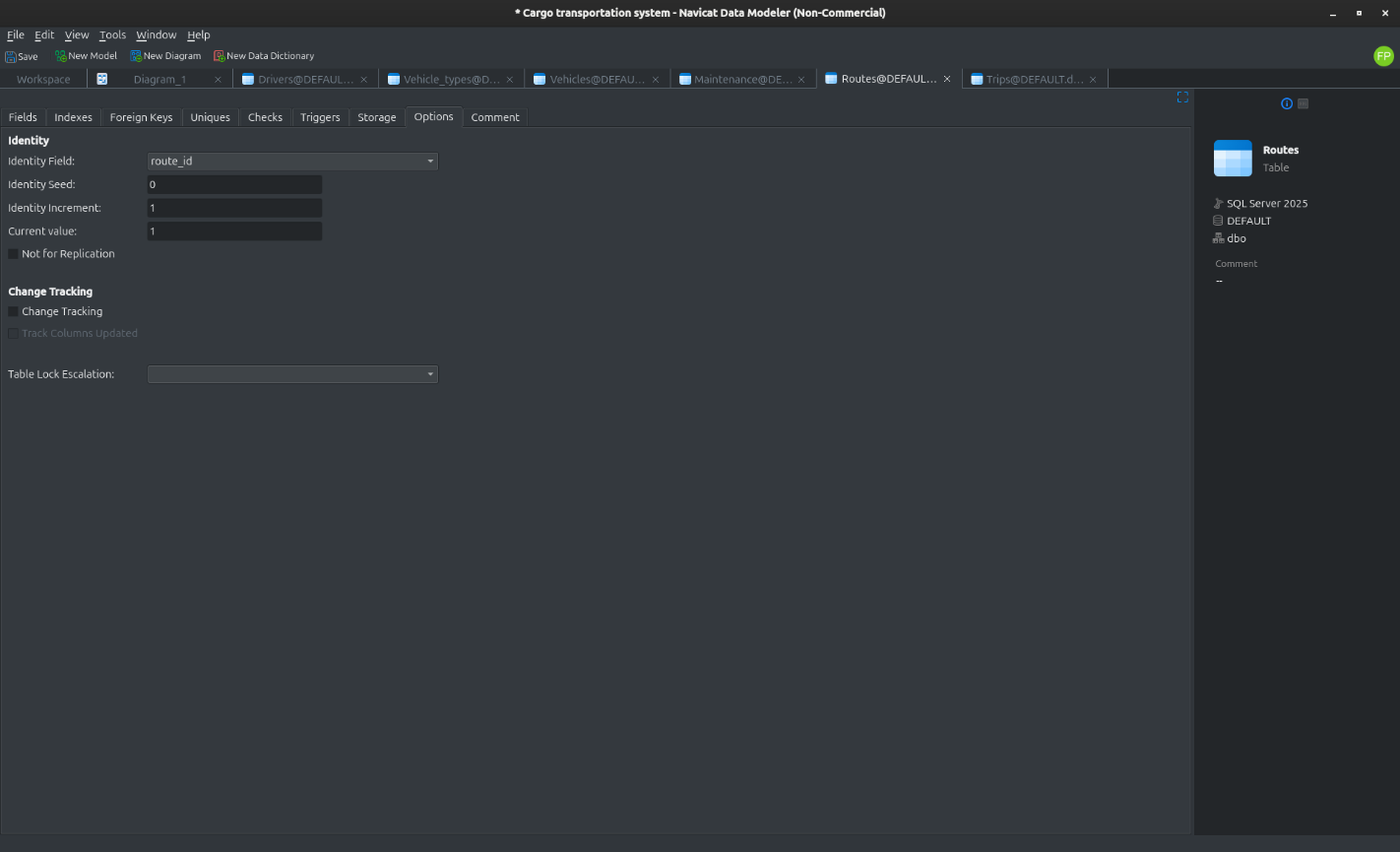
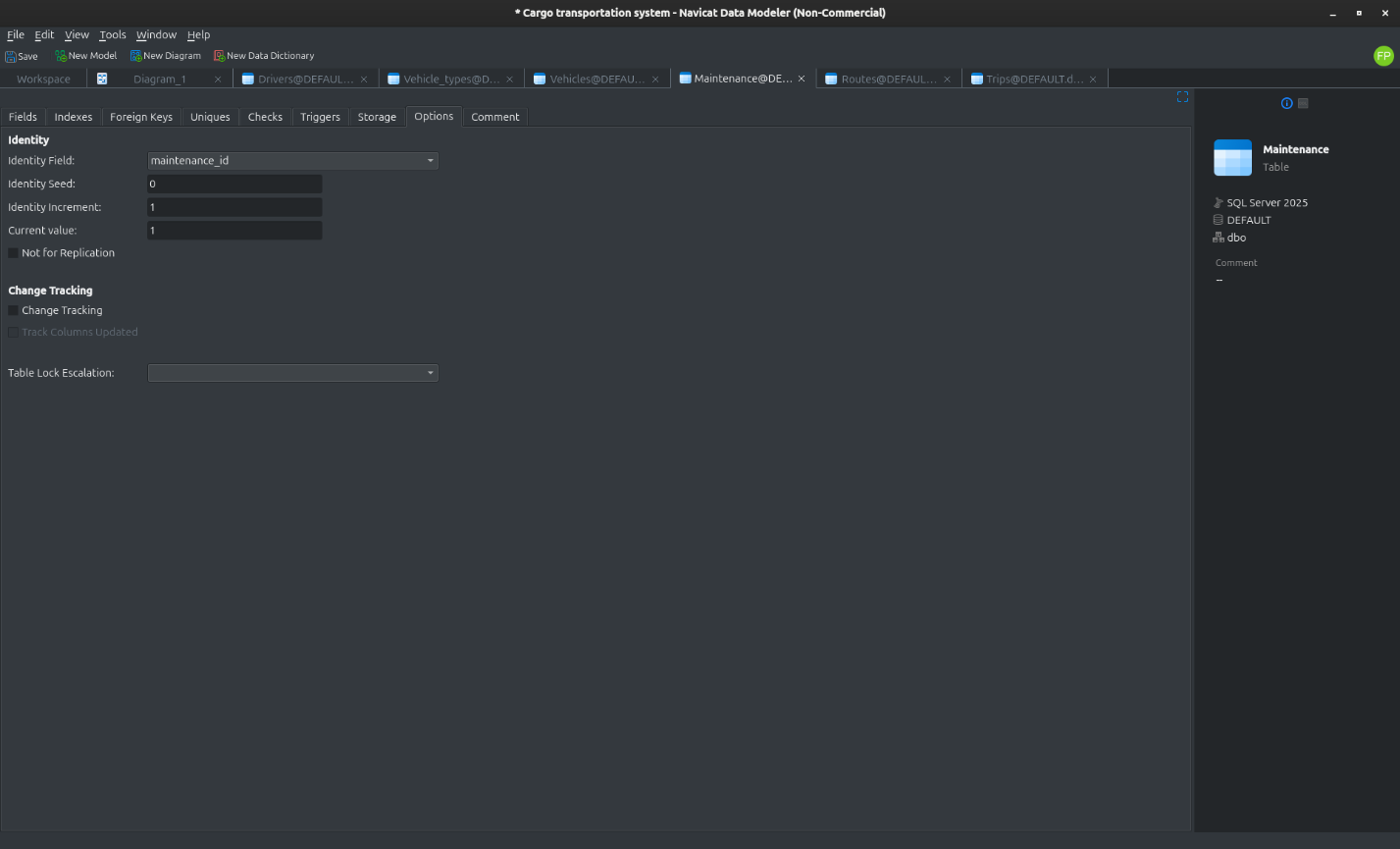
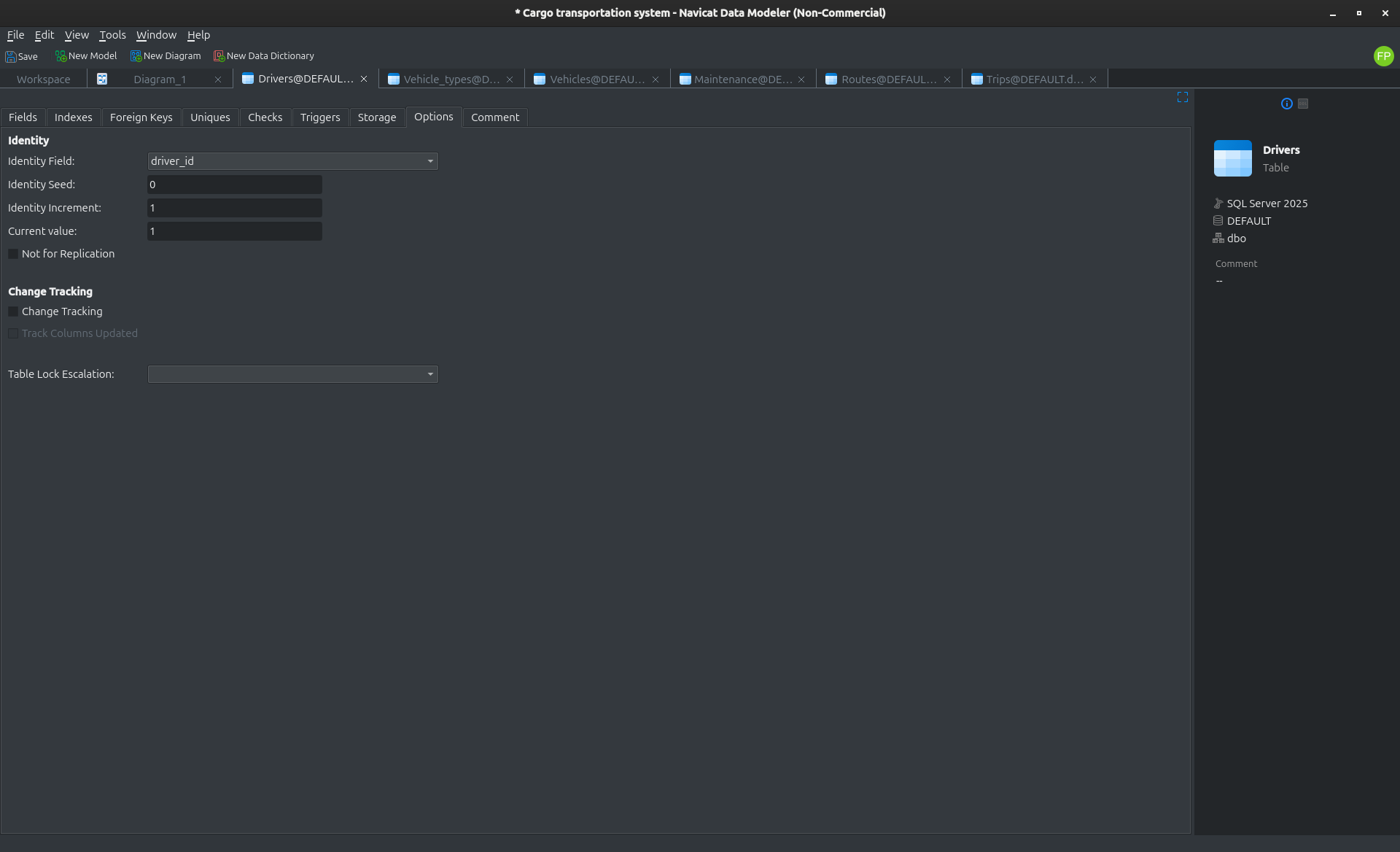
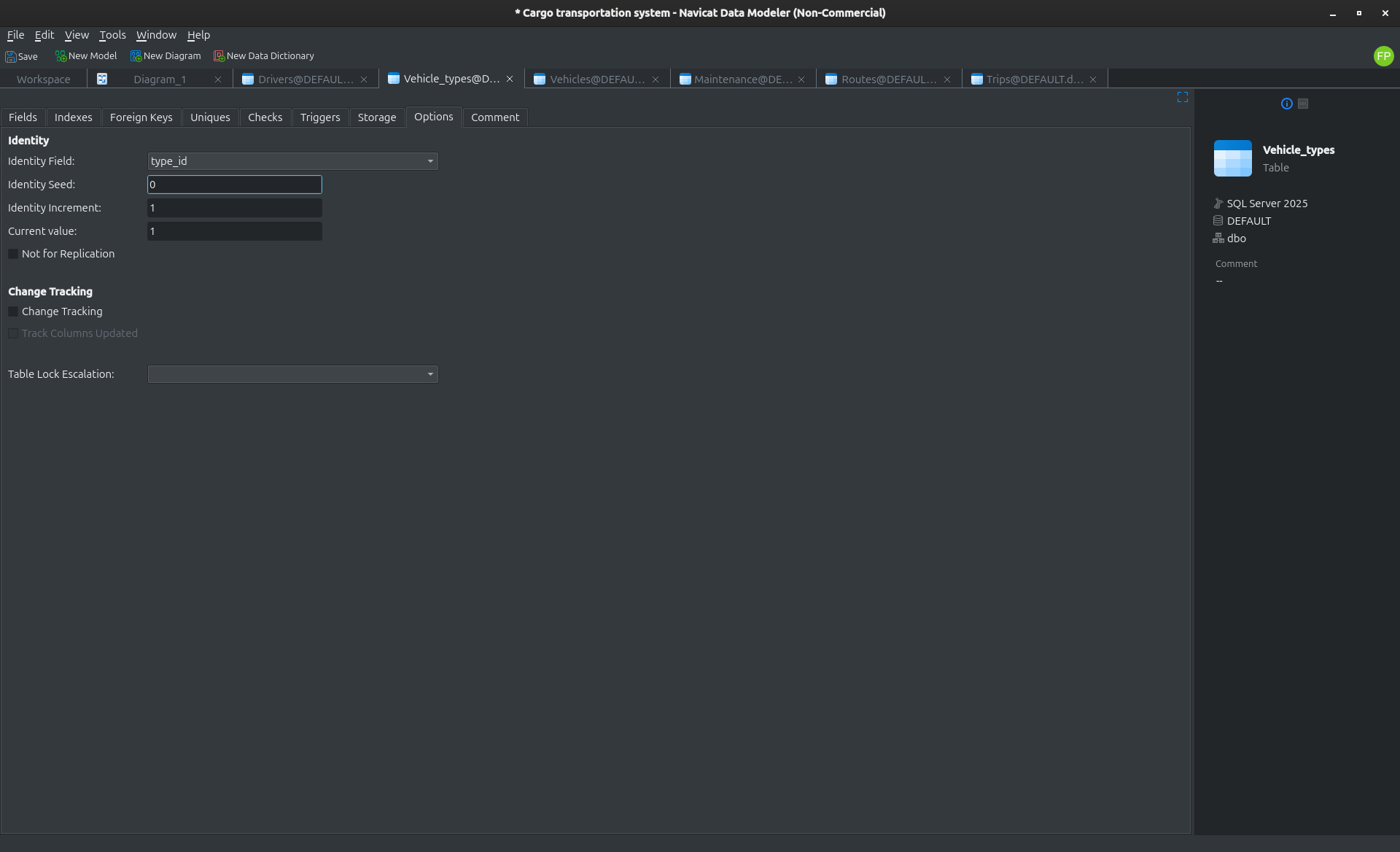
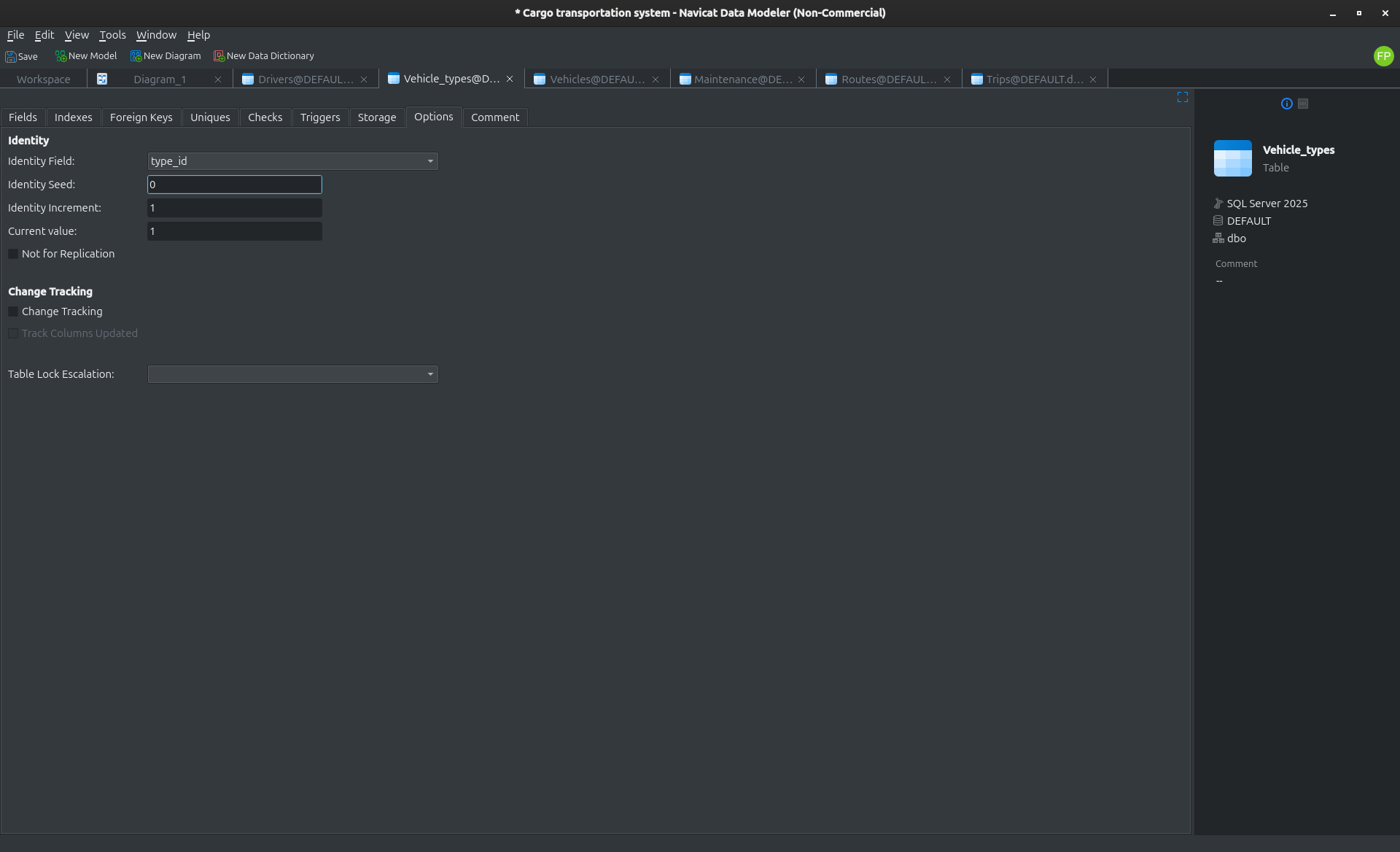
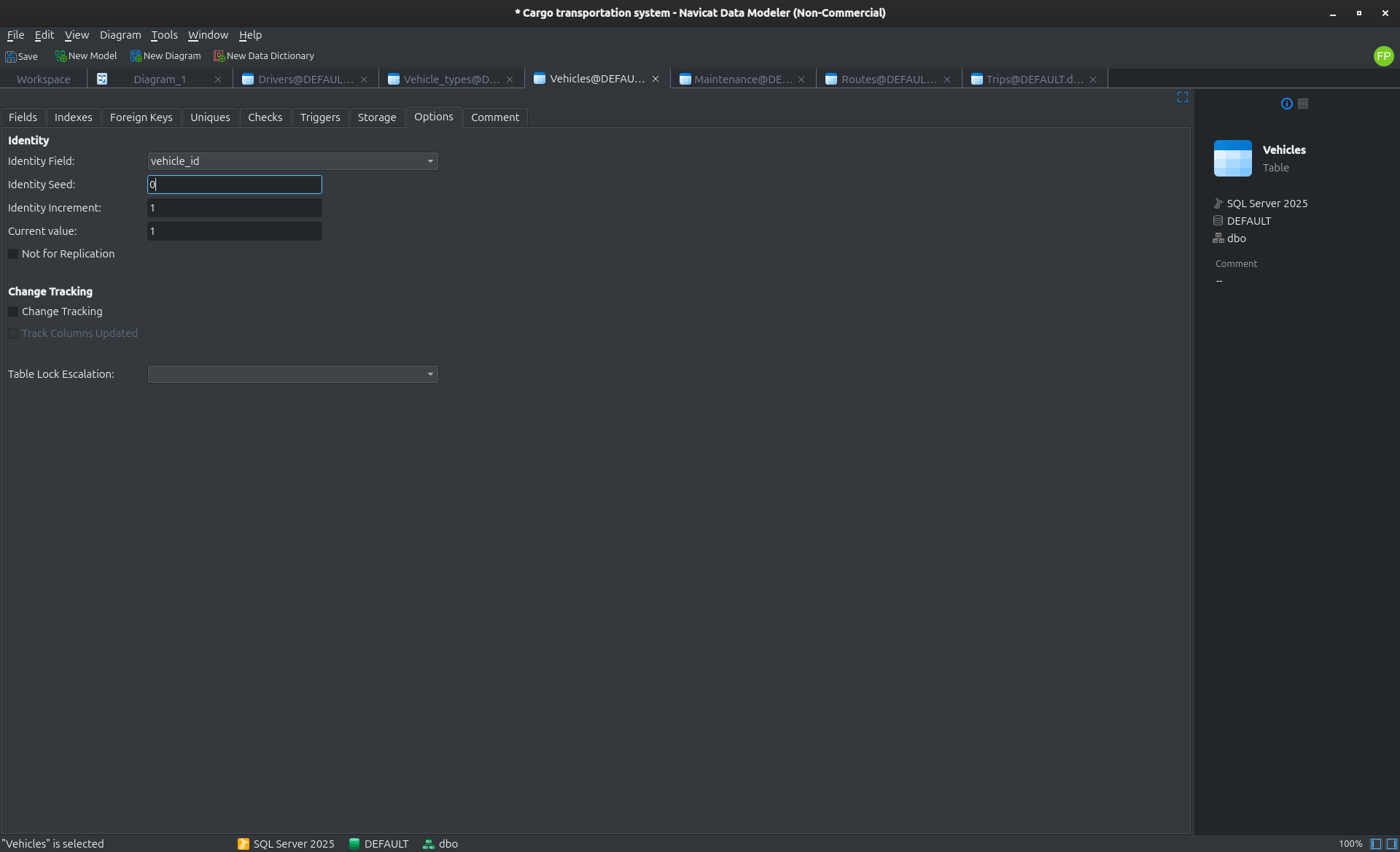
Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

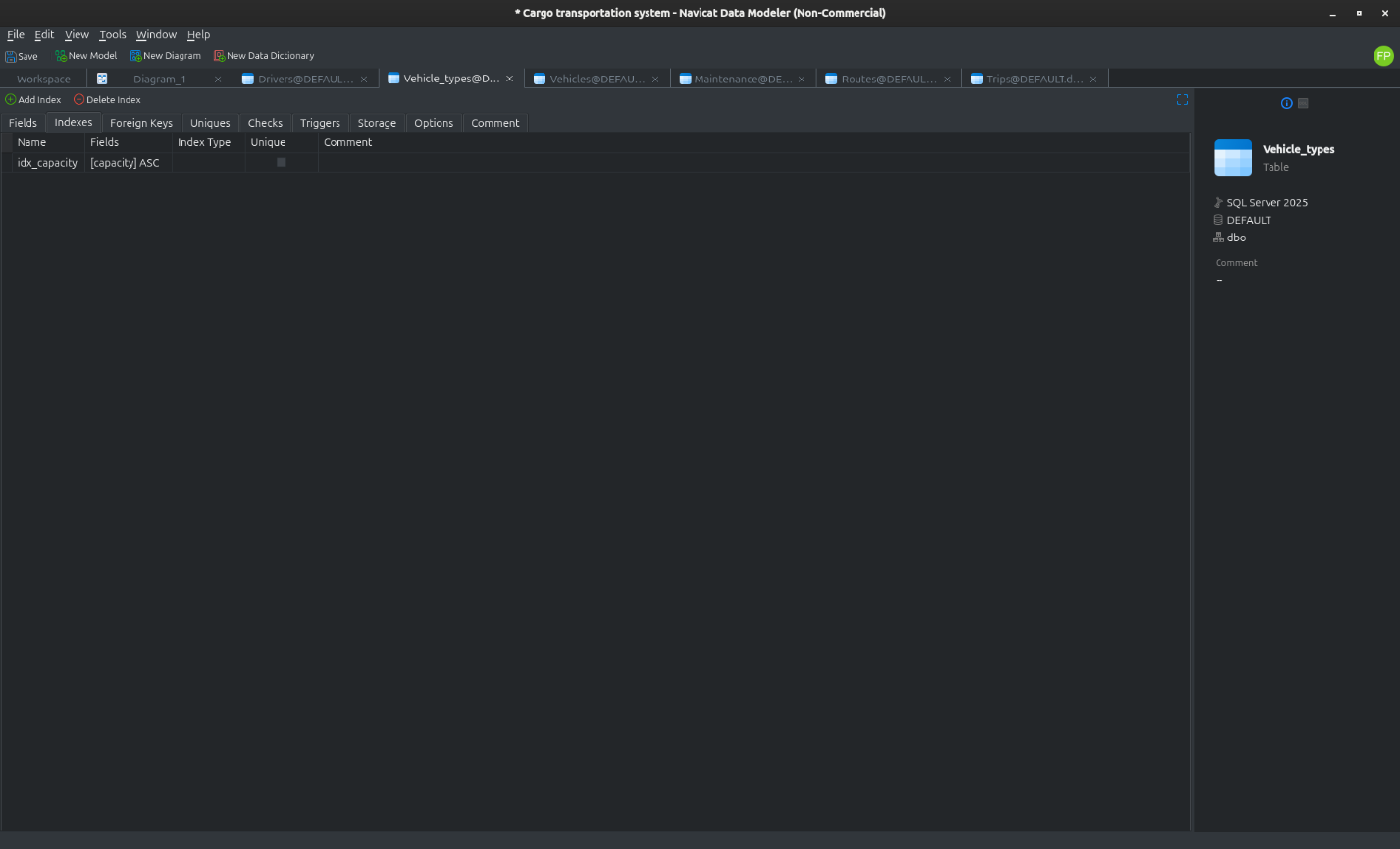
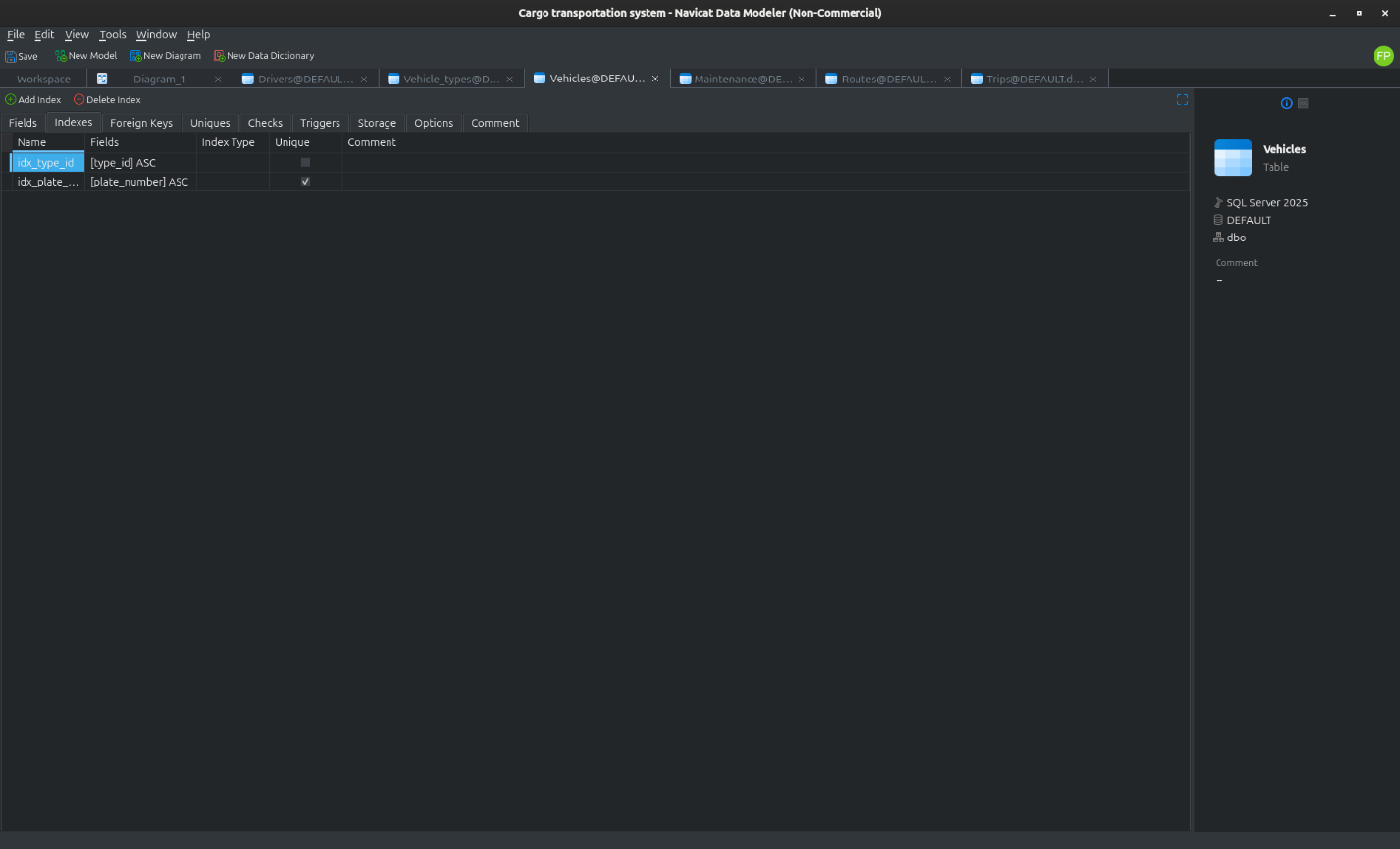
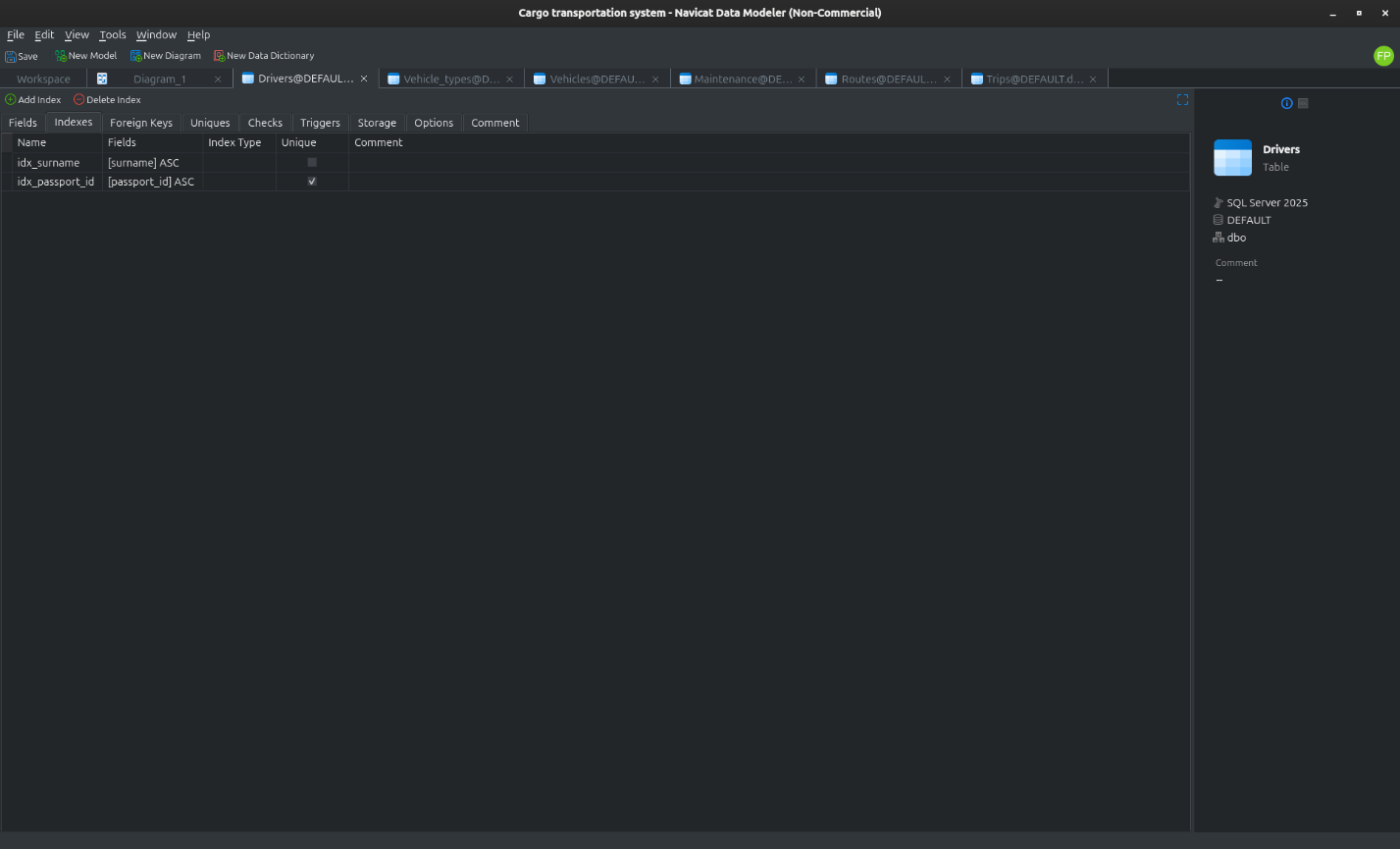
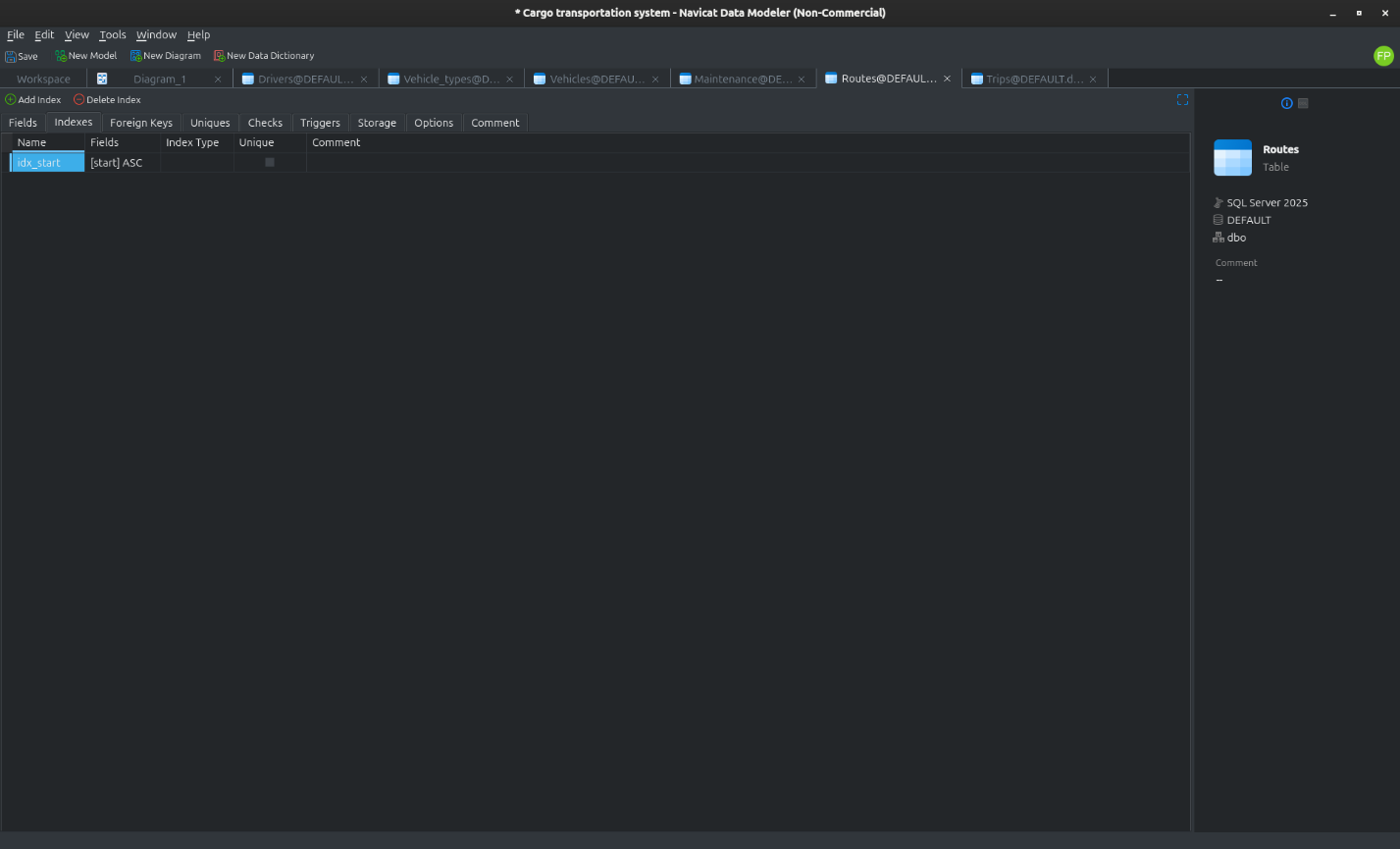
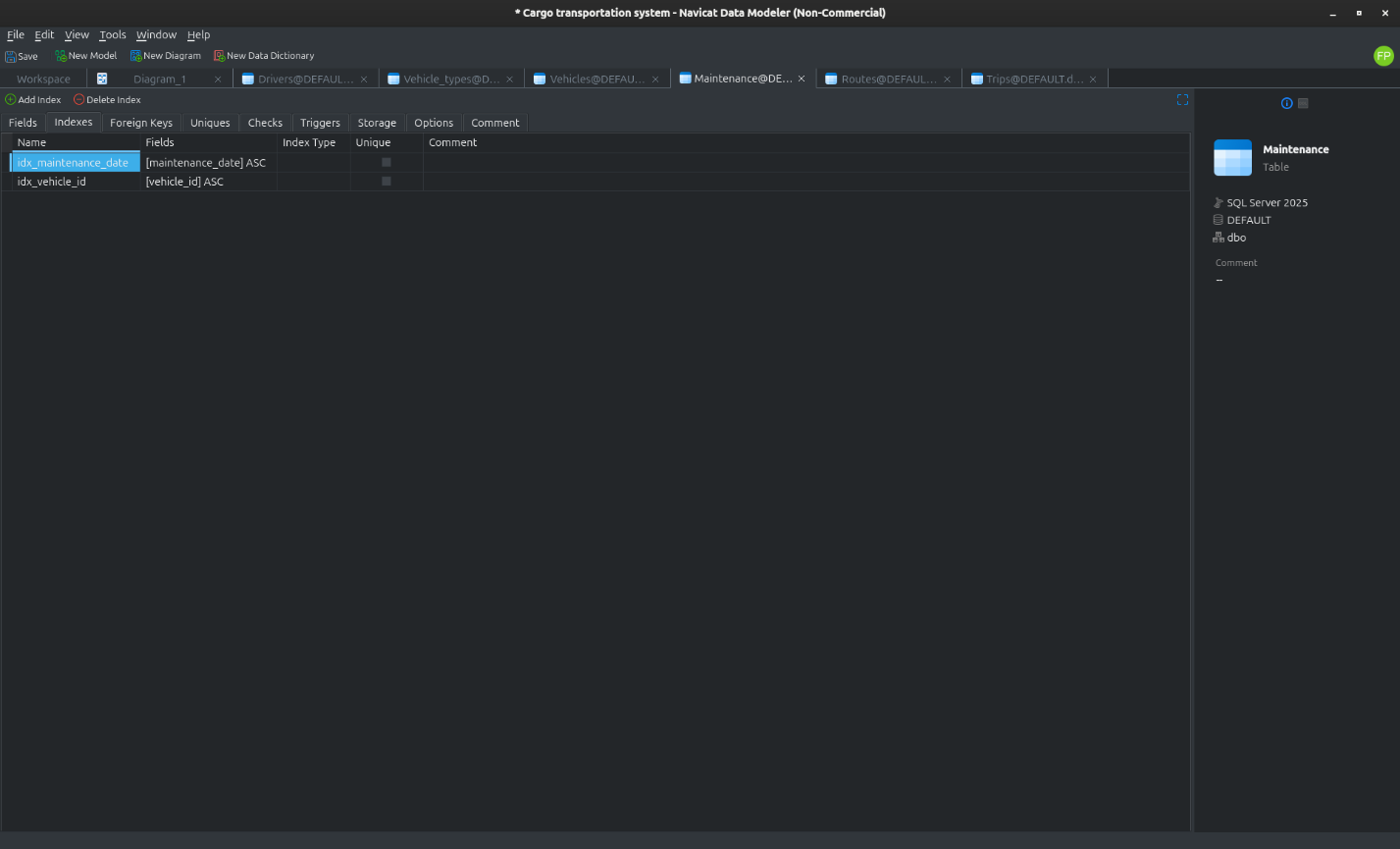
Далее следуя условию лабораторной, я добавил значение по умолчанию для одного из полей – departure\_date:



Следующий шаг – добавление автоматической инкрементации основных ключей таблиц. Через специально отведенную в программе вкладку выполняю этот шаг:



Следующий шаг – добавление индексов. Согласно условию необходимо добавить индексы всем достаточно большим таблицам:

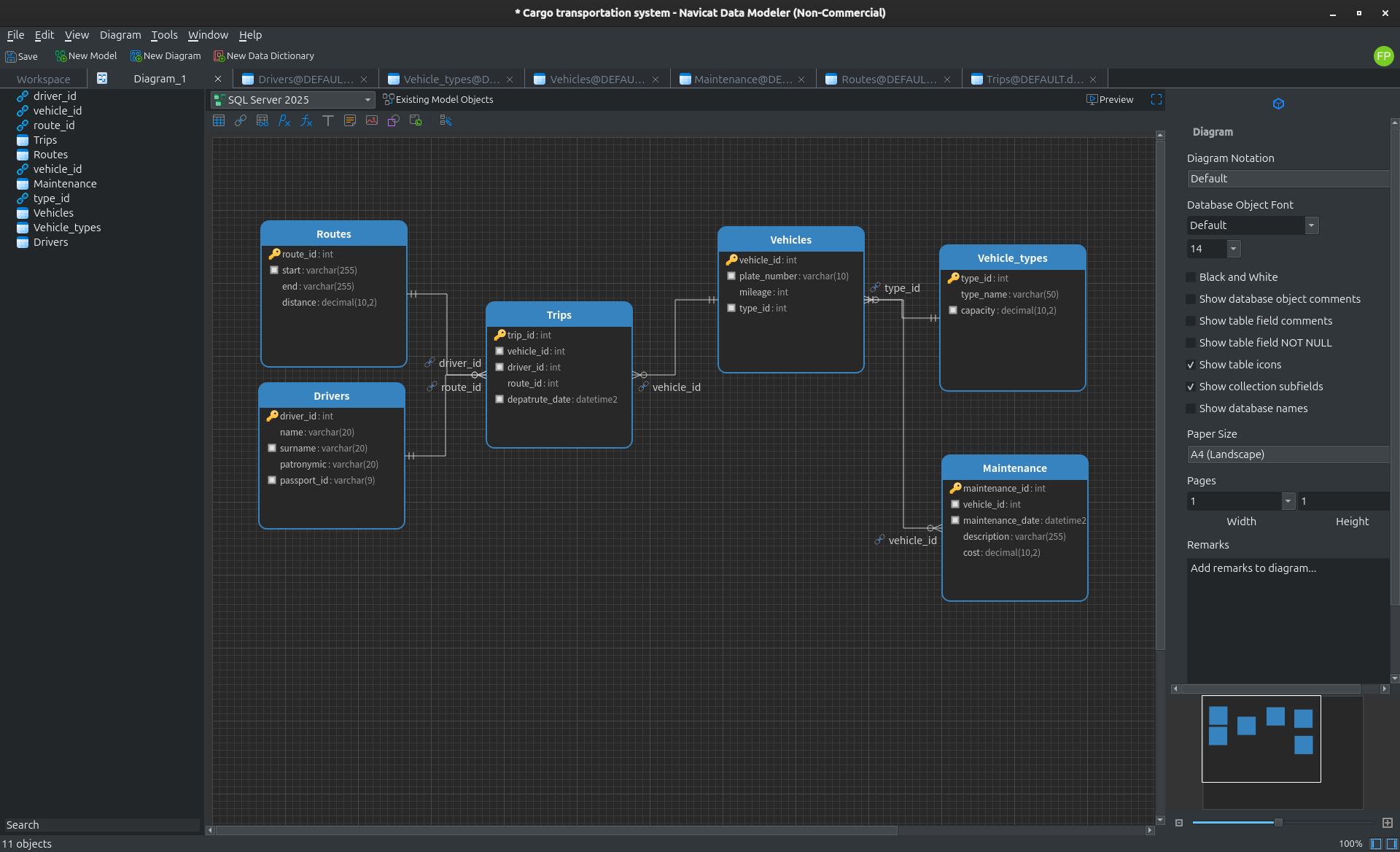


Также хотя бы один составной индекс. Я выбрал составной индекс для таблицы Trips на основе атрибутов driver\_id и departure\_date. Этот индекс позволит более эффективно выполнять запросы, в которых фигурируют указанные атрибуты, т. е. делать выборку маршрутов конкретного транспортного средства в конкретную дату:

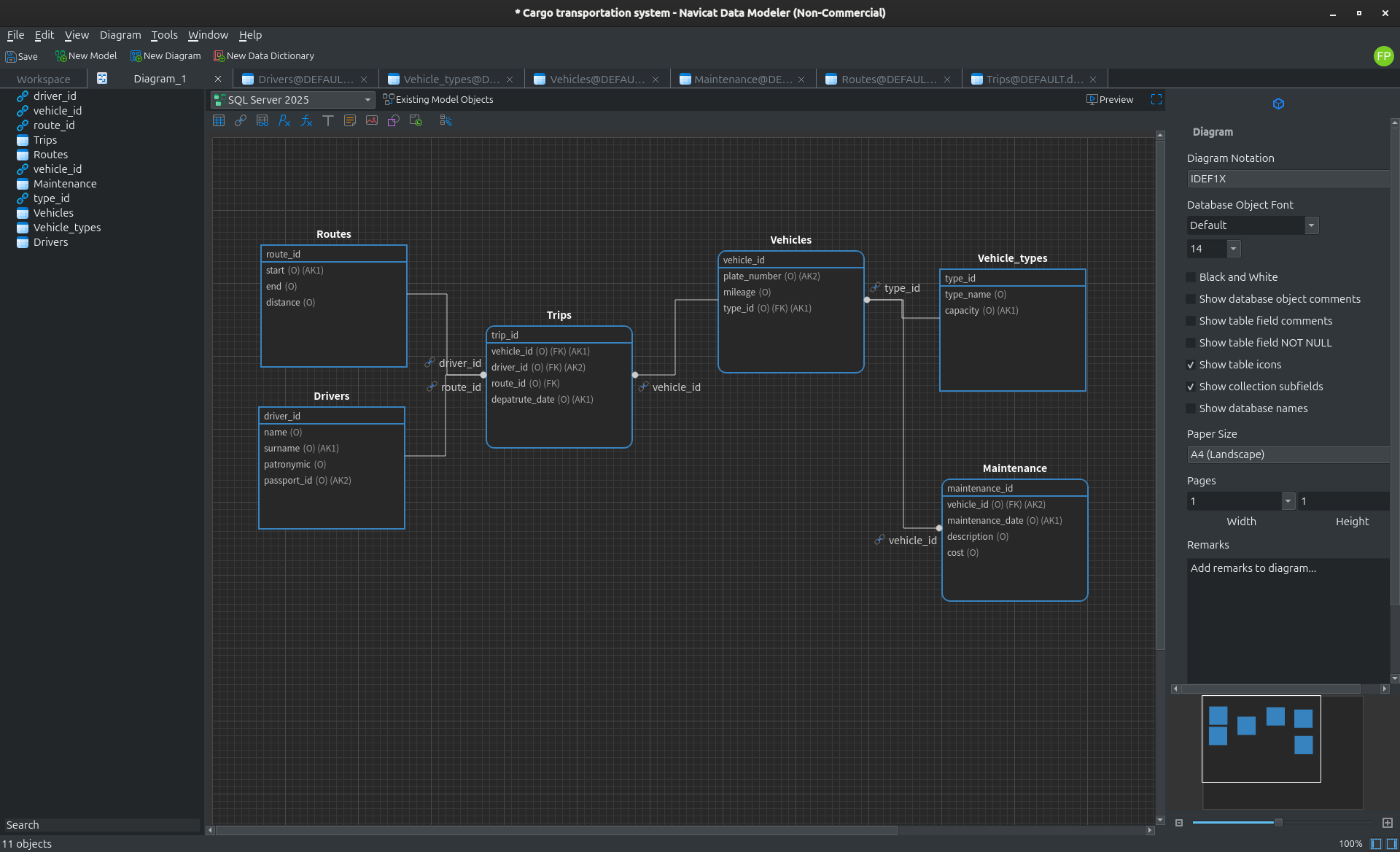
Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

В итоге получаю следующую схему:

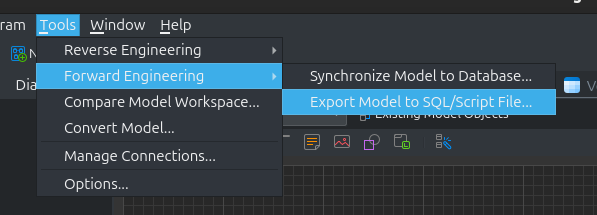


*Рис 1. Представление схемы в виде, стандартном для Navicat Data Modeler 4*

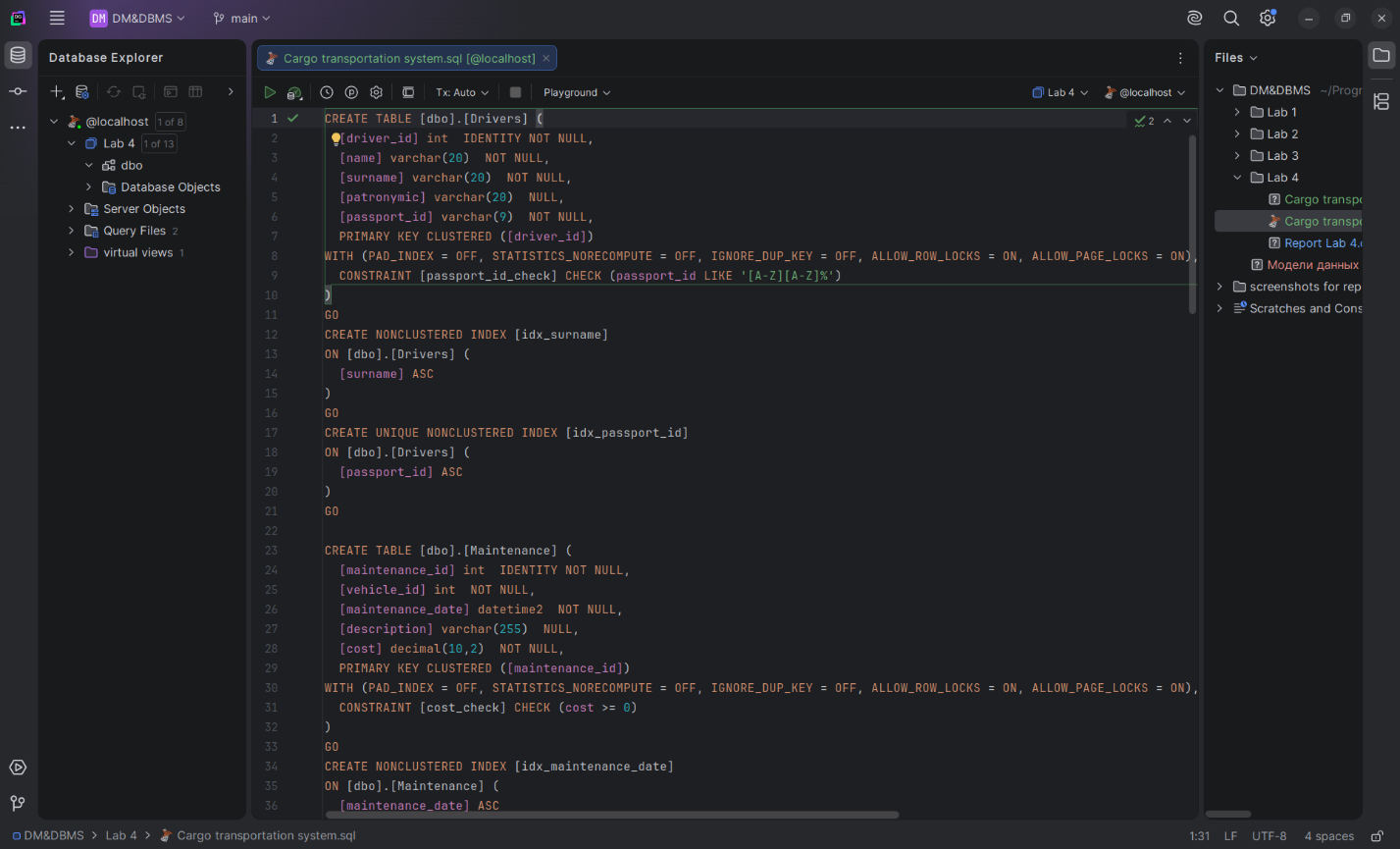
**

*Рис 2. Представление схемы в виде IDEF1X*

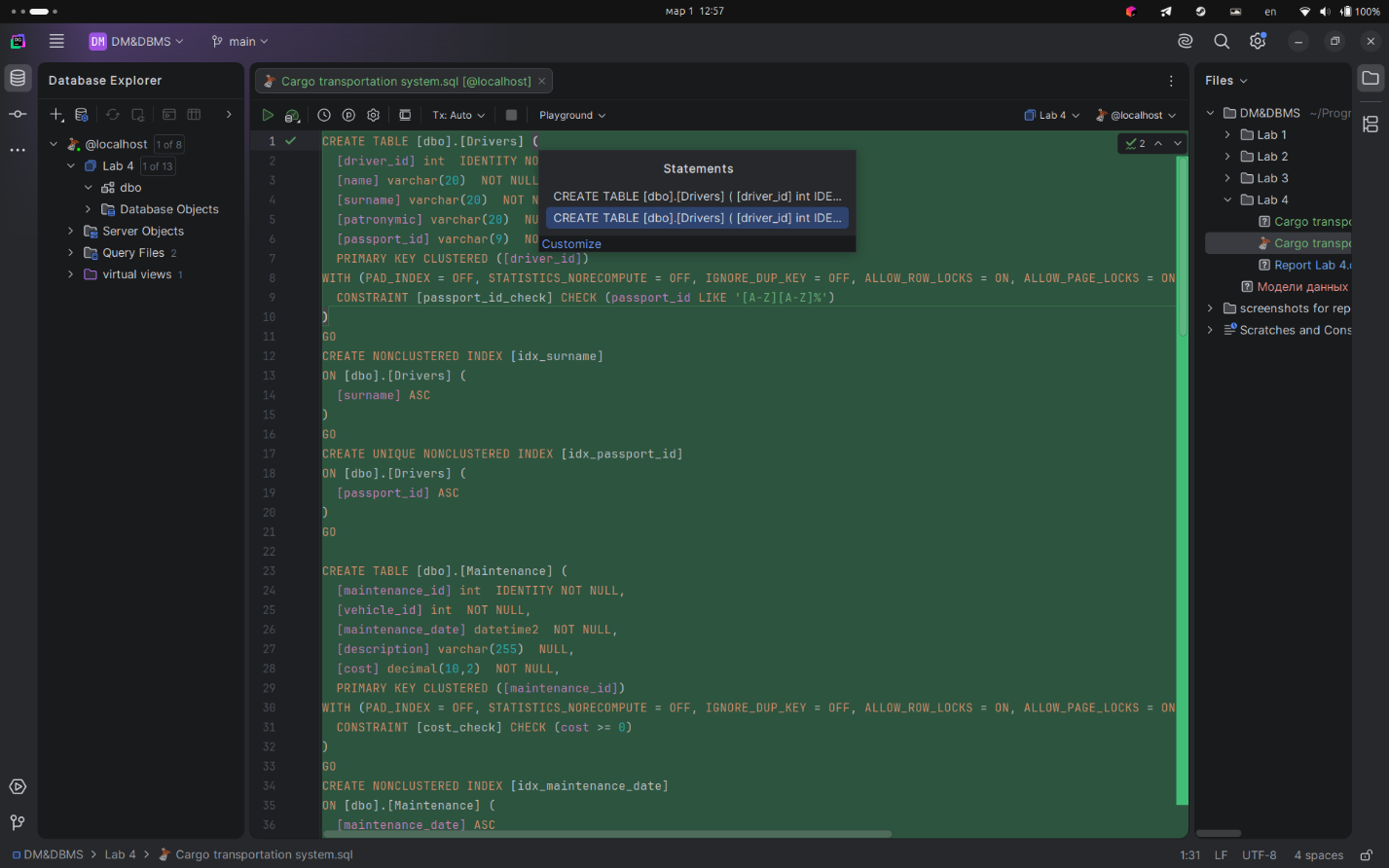
Схема готова. Следующий шаг – переход от схемы к SQL коду для непосредственного создания таблиц. Для этого воспользуюсь опцией Forward Engineering – Export Model to SQL/Script File…:



После сохранения файла получаем готовый для запуска SQL код. Открываю файл с кодом через DataGrip:



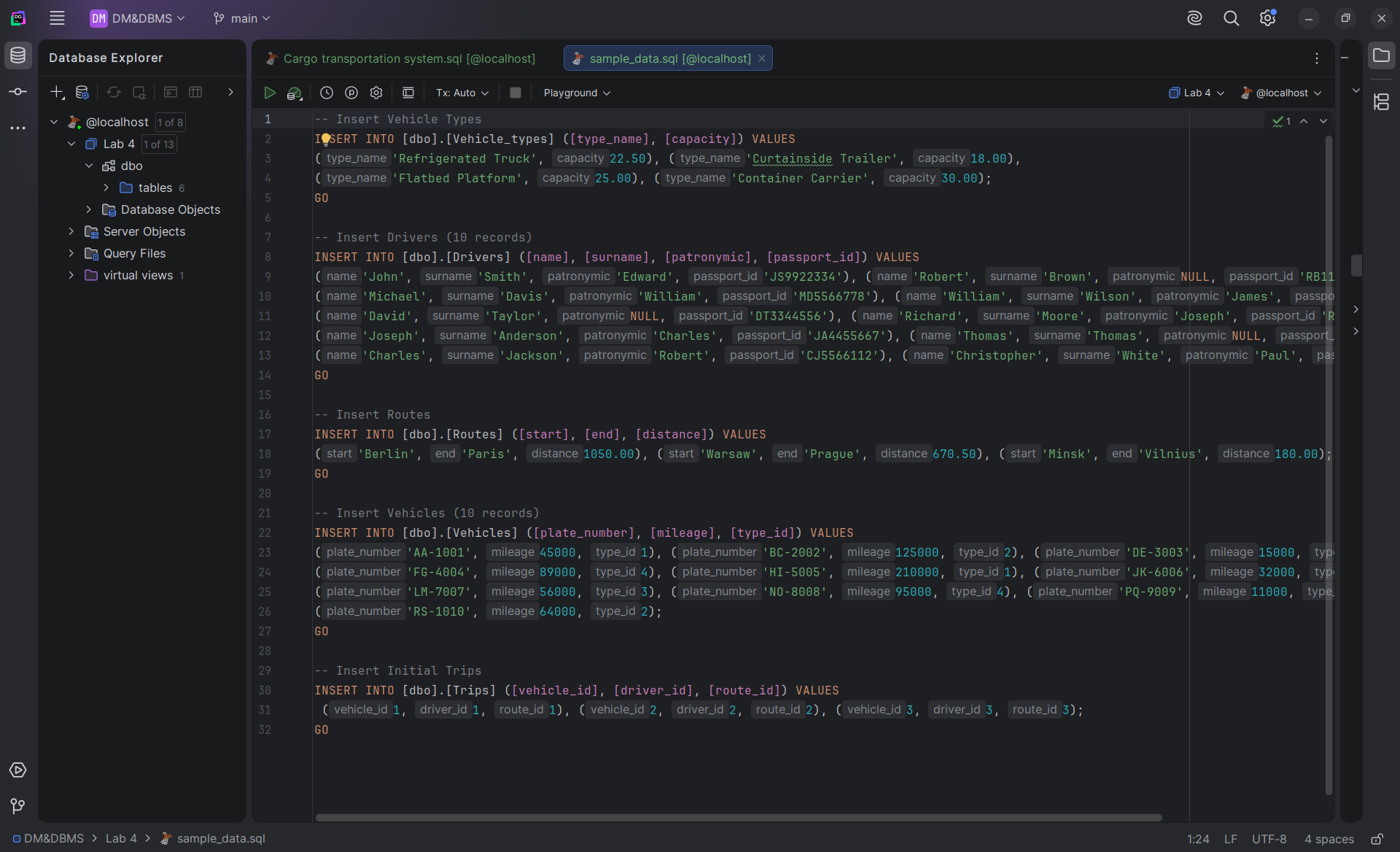
Далее в заранее созданной базе – Lab 4 – запускаю код:



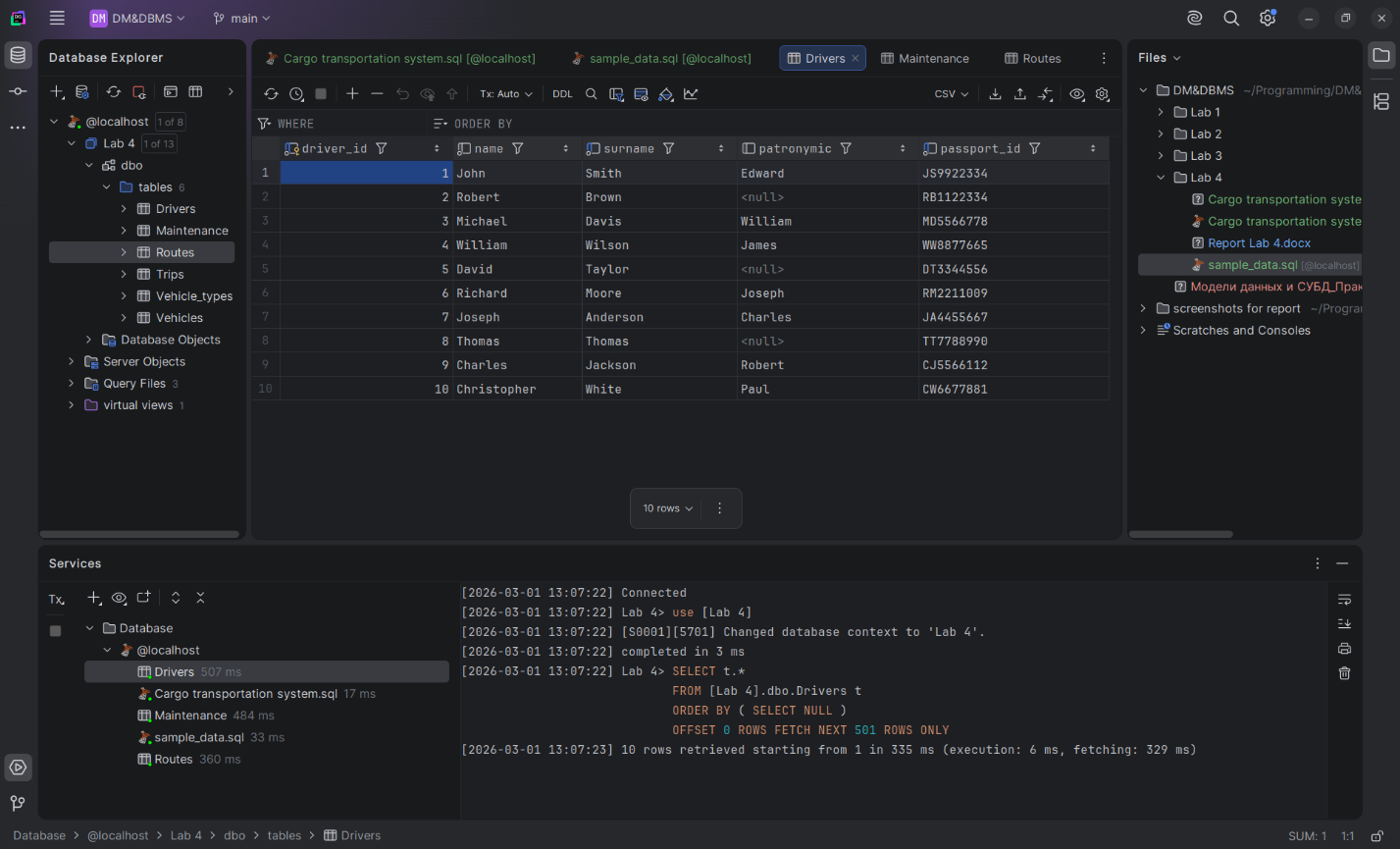
Как видно из скриншота, таблицы были успешно созданы. При двойном нажатии на какую-либо таблицу в центральной части экрана видно результат запроса – колонки таблицы, в моем случае пустые, т. к. никаких данных не было добавлено. Обратив внимание на нижнюю часть экрана, можно заметить, что при двойном нажатии на таблицу в самом деле исполняется SQL код, который выбирает первые 501 записей из таблицы:



Для проверки работоспособности базы данных я обратился в нейросеть NotebookLM для генерации SQL кода с целью заполнения базы записями. Полученный код приложен в архиве с лабораторной:



После запуска кода вижу, что таблица в действительности заполнилась указанными записями:



Напишу и запущу простой SQL запрос, который выберет 5 первых записей из таблицы Drivers:



*Рис 3. В нижней части скриншота виден результат запроса – первые 5 записей из таблицы*