Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Фундаментальные науки»

КАФЕДРА «Математическое моделирование»

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент Тренёв Иван Сергеевич
фамилия, имя, отчество

Группа ФН12-21М

Тип практики научно-исследовательская работа

Название предприятия

Студент _____ **Тренёв И.С.**
подпись, дата

Руководитель практики _____ _____
подпись, дата *фамилия, и.о.*

Оценка _____

2022 г.

«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой _____
_____ _____
« ___ » _____ 20 __ г.

З А Д А Н И Е
на прохождение производственной практики
научно-исследовательская работа
Тип практики

Студент

Тренёв Иван Сергеевич 1 курса группы ФН12-21М
Фамилия Имя Отчество № курса индекс группы

в период с _____. _____.20_____ г. по _____. _____.20_____ г.

Предприятие:

_____

Подразделение:

_____
(отдел/сектор/цех)

Руководитель практики от предприятия (наставник):

_____
(Фамилия Имя Отчество полностью, должность)

Руководитель практики от кафедры:

_____
(Фамилия Имя Отчество полностью, должность)

Задание:

1. Спроектировать ER-модель базы данных "Отель".

2. Преобразовать спроектированную ER-модель в реляционную.

3. Реализовать спроектированную базу данных средствами СУБД SQL Server.

Дата выдачи задания « ____ » _____ 20_ _ г.

Руководитель практики от предприятия _____/_____/

Руководитель практики от кафедры _____/_____/

Студент _____/ Тренёв И.С. /

Содержание

1. Введение

База данных и система управления базой данных являются неотъемлемой частью информационных систем предприятия. Процесс проектирования базы данных представляет собой последовательность переходов от словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели. В общем случае выделяют следующие этапы проектирования: анализ и описание предметной области информационной системы, концептуальное моделирование, построение логической модели базы данных, физическое проектирование базы данных.

Целью данной работы является изучение метода моделирования данных «сущность-связь» и реализация этой модели.

Для решения этой задачи, необходимо сделать следующее:

1. выбрать предметную область, соответствующую 4-5 сущностям;
2. сформировать требования к предметной области;
3. создать модель «сущность-связь» для предметной области с обоснованием выбора кардинальных чисел связей;
4. преобразовать модель «сущность-связь» в реляционную модель согласно процедуре преобразования;
5. обосновать выбор типов данных, ключей, правил обеспечения ограничений минимальной кардинальности;
6. реализовать спроектированную базу данных с помощью средств СУБД SQL Server.
7. создать базу данных с использованием средств SQL Server 2012:
   (a) поддержания создания и физической организации базы данных;
   (b) различных категорий целостности;
   (c) представления и индексы;
   (d) хранимые процедуры, функции и триггеры;
8. создание объектов базы данных должно быть осуществлено средствами DDL (CREATE / ALTER / DROP), иллюстрирующими следующие аспекты:
   (a) добавление и изменение полей;
   (b) назначение типов данных;
   (c) назначение ограничений целостности (PRIMARY KEY, NULL / NOT NULL / UNIQUE, CHECK и т.п.);
   (d) определение значений по умолчанию;
9. в рассматриваемой базе данных должны быть созданы запросы DML для:
   (a) выборки записей (команда SELECT);

(b) добавления новых записей (команда INSERT), как с помощью непосредственного указания значений, так и с помощью команды SELECT;

(c) модификации записей (команда UPDATE);

(d) удаления записей (команда DELETE);

10. запросы, созданные в рамках пп.2, 3 должны иллюстрировать следующие возможности языка:

(a) удаление повторяющихся записей (DISTINCT);

(b) выбор, упорядочивание и именование полей (создание псевдонимов для полей и таблиц / представлений);

(c) соединение таблиц (INNER JOIN / LEFT JOIN / RIGHT JOIN / FULL OUTER JOIN);

(d) условия выбора записей (в том числе, условия / LIKE / BETWEEN / IN / EXISTS);

(e) сортировка записей (ORDER BY - ASC, DESC);

(f) группировка записей (GROUP BY + HAVING, использование функций агрегирования – COUNT / AVG / SUM / MIN / MAX);

(g) объединение результатов нескольких запросов (UNION / UNION ALL / EXCEPT / INTERSECT);

(h) вложенные запросы.

2. Проектирование и преобразование ER-модели

2.1. Предметная область и требования

Для реализации модели, в качестве предметной области была выбрана системы заказа такси. Предполагается, что каждый клиент может сформировать заказ, с учетом своих индивидуальных предпочтений и наличий скидок. В связи с этим к предметной области были сформулированы следующие требования:

- Каждый клиент может сделать сколько угодно заказов;
- У каждого заказа может быть только один клиент и не более одного водителя;
- Клиент может иметь несколько скидок;
- Водитель имеет в доступе только один автомобиль.

2.2. Модель «сущность-связь»

На основе описанной в предыдущем пункте предметной области была синтезирована модель «сущность-связь» (1), включающая 5 сущностей:

- Клиент – сущность, являющаяся абстракцией покупателя услуги извоза, с идентификатором *ID клиента* и атрибутами: Имя, Фамилия, Отчество.

- Скидка – сущность, являющаяся абстракцией наличия скидок у клиента с идентификатором *ID скидочной карты* и атрибутами: Номер карты, Текущая скидка, Срок действия.
- Заказ – сущность, являющаяся абстракцией формируемого заказа с идентификатором *ID заказа* и атрибутами: Дата и время, Улица, Дом, Багаж, Стоимость, Комментарий водителю, Способ оплаты, Класс автомобиля.
- Водитель – сущность, являющаяся абстракцией исполнителя услуги извоза с идентификатором *ID водителя* и атрибутами: Имя, Фамилия, Отчество, Номер телефон, Статус.
- Автомобиль – сущность, определяющая специфику автомобиля для извоза, с составным идентификатором *ID водителя, ID автомобиля* и атрибутами: ГОС номер, Марка, Цвет, Год выпуска, Класс автомобиля, Топливо.

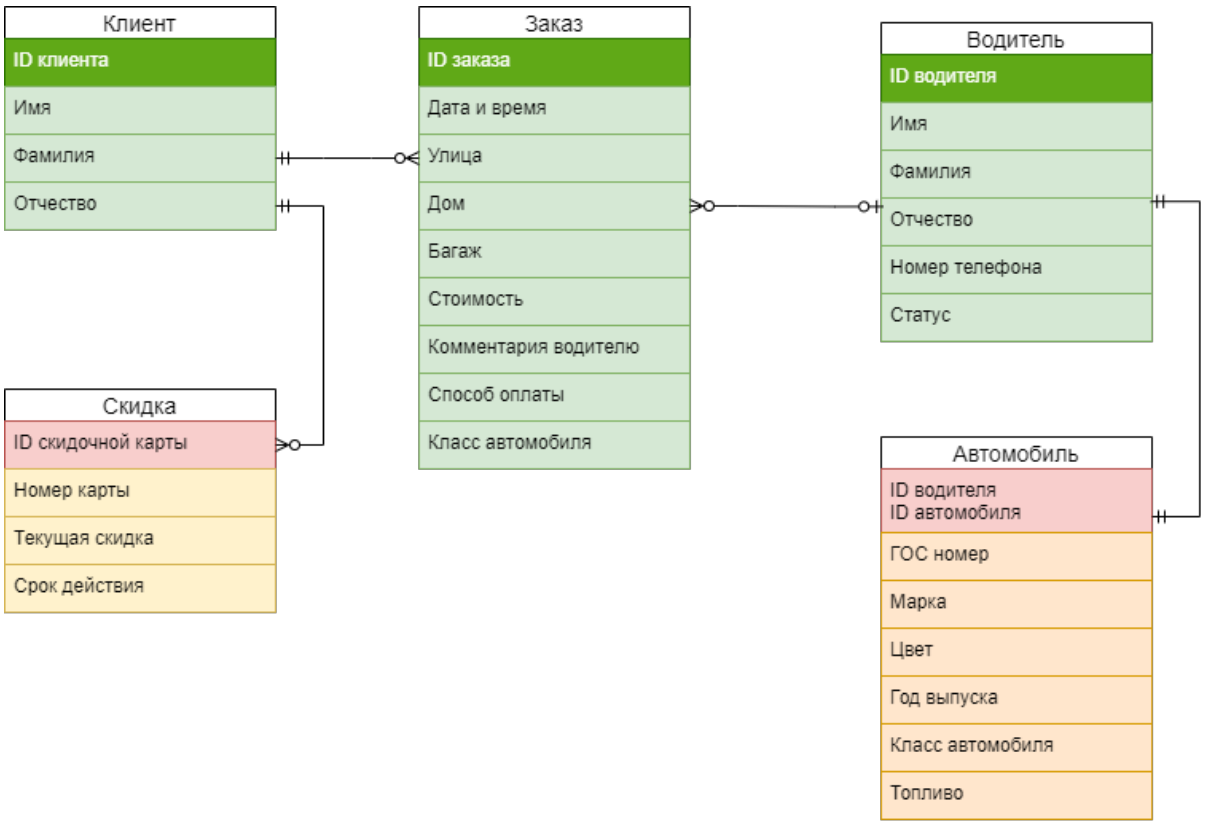

Рис. 1. ER модель.

Между описанными сущностями были построены связи, согласующиеся с особенностями предметной области, описанными в предыдущем пункте. Так как каждый клиент может сделать несколько заказов сразу, но у каждого заказа должен быть ровно один клиент, то связь сущностей *Клиент* и *Заказ* «один-ко-многим», а минимальные и максимальные кардинальные числа равны 1:1 и 0:N. Аналогичная связь между сущностями *Клиент* и *Скидка*, так как каждой скидке соответствует ровно один владелец, но у клиента может быть несколько скидок. Между сущностями *Водитель* и *Заказ* так же связь «один-ко-многим», однако кардинальные числа равны

0:1 и 0:N, так на заказ может быть назначен только один водитель (а может и не быть назначен), а у водителя может быть несколько заказов (а может и не быть не одного). Между сущностями *Водитель* и *Автомобиль* связь «один-к-одному», кардинальные числа равны 1:1 для обоих случаев, поскольку у водителя имеется ровно один автомобиль, и у автомобиля есть только один водитель.

2.3. Преобразование модели «сущность-связь» в реляционную модель

Для идентификационно-зависимых сущностей обновляются внешние ключи, в соответствии с первичными ключами сильных сущностей, также добавляются внешние ключи, которые неявно присутствовали в модели «сущность-связь». Нормализация не требуется, так как удовлетворяет условиям БКНФ (нормальной формы Бойса-Кодда). Полученная модель изображена на рисунке 2

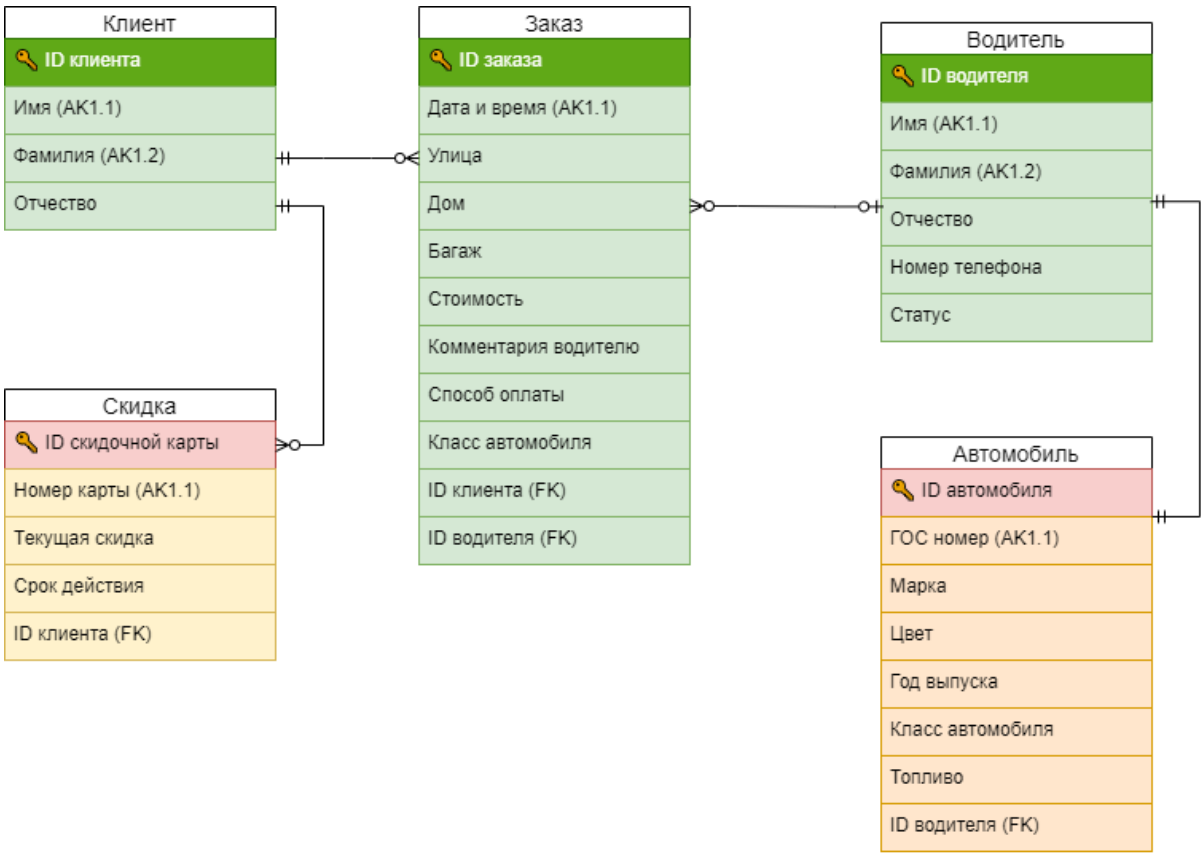

Рис. 2. Реляционная модель.

2.4. Обоснование выбора типов данных, ключей, правил обеспечения ограничений минимальной кардинальности

Данные таблицы, соответствуют требованиям к предметной области.

Автомобиль			
Column	Type	Key	NULL Status
ID автомобиля	int	Primary key	NOT NULL
ID водителя	int	Primary key, Foreign key	NOT NULL
ГОС номер	char(10)	Alternate key	NOT NULL
Марка	char(20)	No	NOT NULL
Цвет	char(20)	No	NOT NULL
Год выпуска	int	No	NULL
Класс автомобиля	char(20)	No	NOT NULL

Рис. 3. Обоснование выбора типов данных и ключей для сущности *Автомобиль*.

Клиент			
Column	Type	Key	NULL Status
ID клиента	int	Primary key	NOT NULL
Имя	nvarchar(30)	Alternate key	NOT NULL
Фамилия	nvarchar(30)	Alternate key	NOT NULL
Отчество	nvarchar(30)	No	NOT NULL

Рис. 4. Обоснование выбора типов данных и ключей для сущности *Клиент*.

Скидка			
Column	Type	Key	NULL Status
ID скидочной карты	int	Primary key	NOT NULL
Номер карты	int	Alternate key	NOT NULL
Текущая скидка	int	No	NULL
Срок действия	datetime	No	NOT NULL
ID клиента	int	Foreign key	NOT NULL

Рис. 5. Обоснование выбора типов данных и ключей для сущности *Скидка*.

Водитель			
Column	Type	Key	NULL Status
ID водителя	int	Primary key	NOT NULL
Имя	nvarchar(30)	Alternate key	NOT NULL
Фамилия	nvarchar(30)	Alternate key	NOT NULL
Отчество	nvarchar(30)	No	NOT NULL
Номер телефона	char(10)	no	NOT NULL
Статус	bit	no	NOT NULL

Рис. 6. Обоснование выбора типов данных и ключей для сущности *Водитель*.

Заказ			
Column	Type	Key	NULL Status
ID заказа	int	Primary key	NOT NULL
Дата и время	datetime	Alternate key	NOT NULL
Улица	nvarchar(30)	No	NOT NULL
Дом	int	No	NOT NULL
Багаж	bit	No	NULL
Стоимость	int	No	NOT NULL
Комментарий водителю	nvarchar(300)	No	NULL
Способ оплаты	bit	No	NOT NULL
Класс автомобиля	int	No	NULL
ID клиента	int	Foreign key, Alternate key	NOT NULL
ID заказа	int	Foreign key	NOT NULL

Рис. 7. Обоснование выбора типов данных и ключей для сущности *Заказ*.

M-O	Action on КЛИЕНТ (Parent)	Action on СКИДКА (Child)
Insert	None	Get a parent
Modify key or foreign key	Prohibit	Prohibit
Delete	Cascade delete	None

Рис. 8. Обоснование выбора правил обеспечения ограничений минимальной кардинальности для связи *Клиент-Скидка*.

M-O	Action on КЛИЕНТ (Parent)	Action on ЗАКАЗ (Child)
Insert	None	Get a parent
Modify key or foreign key	Prohibit	Prohibit
Delete	Cascade delete	None

Рис. 9. Обоснование выбора правил обеспечения ограничений минимальной кардинальности для связи *Клиент-Заказ*.

M-M	Action on ВОДИТЕЛЬ (Parent)	Action on АВТОМОБИЛЬ (Child)
Insert	Insert trigger on ВОДИТЕЛЬ to create row in АВТОМОБИЛЬ	Will be created by Insert trigger on ВОДИТЕЛЬ
Modify key or foreign key	Prohibit	Prohibit
Delete	Prohibit	Prohibit

Рис. 10. Обоснование выбора правил обеспечения ограничений минимальной кардинальности для связи *Водитель-Авто*.

3. Заключение

За время прохождения научно-исследовательской практики были выполнены следующие задачи:

1. Выбрана предметная область, соответствующая 4-5 сущностям, к которой были сформированы соответствующие требования.
2. Создана модель «сущность-связь» для предметной области с обоснованием выбора кардинальных чисел связей.
3. Модель «сущность-связь» была преобразована в реляционную модель согласно процедуре преобразования.
4. Обоснован выбор типов данных, ключей, правил обеспечения ограничений минимальной кардинальности.
5. Создана база данных с использованием средств SQL Server 2012:
   (a) поддержания создания и физической организации базы данных;
   (b) различных категорий целостности;
   (c) представления и индексы;
   (d) хранимые процедуры, функции и триггеры.
6. Создание объектов базы данных было осуществлено средствами DDL (CREATE / ALTER / DROP), иллюстрирующими следующие аспекты:
   (a) добавление и изменение полей;
   (b) назначение типов данных;
   (c) назначение ограничений целостности (PRIMARY KEY, NULL / NOT NULL / UNIQUE, CHECK и т.п.);
   (d) определение значений по умолчанию.
7. В рассматриваемой базе данных были созданы запросы DML для:
   (a) выборки записей (команда SELECT);
   (b) добавления новых записей (команда INSERT), как с помощью непосредственного указания значений, так и с помощью команды SELECT;
   (c) модификации записей (команда UPDATE);
   (d) удаления записей (команда DELETE).
8. Запросы, созданные в рамках пп.2, 3 иллюстрируют следующие возможности языка:
   (a) удаление повторяющихся записей (DISTINCT);
   (b) выбор, упорядочивание и именование полей (создание псевдонимов для полей и таблиц / представлений);

(c) соединение таблиц (INNER JOIN / LEFT JOIN / RIGHT JOIN / FULL OUTER JOIN);

(d) условия выбора записей (в том числе, условия / LIKE / BETWEEN / IN / EXISTS);

(e) сортировка записей (ORDER BY - ASC, DESC);

(f) группировка записей (GROUP BY + HAVING, использование функций агрегирования – COUNT / AVG / SUM / MIN / MAX);

(g) объединение результатов нескольких запросов (UNION / UNION ALL / EXCEPT / INTERSECT);

(h) вложенные запросы.

4. Реализация с помощью средств СУБД SQL Server

Ниже приведен листинг кода для формирования базы, и создания всех запросов.

```
USE master
GO

-- Проверка на пустоту
IF DB_ID(N'TAXI_PROJECT') IS NOT NULL
    DROP DATABASE TAXI_PROJECT;
GO
-- Создание БД
CREATE DATABASE TAXI_PROJECT
ON
( NAME = Taxi_dat,
    FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\DATA\taxiproject.mdf',
    SIZE = 10,
    MAXSIZE = 50,
    FILEGROWTH = 5 )
LOG ON
( NAME = Taxi_log,
    FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\DATA\taxiprojectlog.ldf',
    SIZE = 5MB,
    MAXSIZE = 25MB,
    FILEGROWTH = 5MB );
GO

USE TAXI_PROJECT
GO


--============================================КЛИЕНТ=============================================

-- Проверка на пустоту
IF OBJECT_ID(N'Client_table') is NOT NULL
    DROP TABLE Client_table
GO
-- Создание таблицы
CREATE TABLE Client_table
(
 id_client int IDENTITY NOT NULL,
 cl_first_name nvarchar(30) DEFAULT 'Vasiliy' NOT NULL,
 cl_last_name nvarchar(30) DEFAULT 'Pupkin' NOT NULL,
 cl_patronymic nvarchar(30) DEFAULT 'Petrovich' NOT NULL,
 CONSTRAINT Uniq_client UNIQUE (cl_first_name, cl_last_name, cl_patronymic),
 CONSTRAINT PK_id_client PRIMARY KEY (id_client)
)
GO


--============================================СКИДКА=============================================

-- Проверка на пустоту
IF OBJECT_ID(N'Sale_table') is NOT NULL
    DROP TABLE Sale_table
GO
-- Создание таблицы
CREATE TABLE Sale_table
(
 id_sale uniqueidentifier DEFAULT NEWID(),
 cart_number int DEFAULT 00000 NOT NULL,
 current_sale int DEFAULT 0 NULL,
 CONSTRAINT PK_id_sale PRIMARY KEY (id_sale),
 CONSTRAINT Uniq_sale UNIQUE (cart_number),
 id_name int default 1 NOT NULL,
 FOREIGN KEY (id_name) REFERENCES Client_table (id_client)
   ON DELETE CASCADE
   ON UPDATE CASCADE
)
GO
```

```
-- Добавление элемента в таблицу (CHECK)
ALTER TABLE Sale_table
   ADD validity date DEFAULT CONVERT(date, '1/1/2025') NOT NULL
   CONSTRAINT CHK_validity
   CHECK (validity > CONVERT(date, CURRENT_TIMESTAMP) );
GO


--=============================================ВОДИТЕЛЬ=============================================

-- Проверка на пустоту
IF OBJECT_ID(N'Driver_table') is NOT NULL
    DROP TABLE Driver_table
GO
-- Создание таблицы
CREATE TABLE Driver_table
(
 id_driver int PRIMARY KEY NOT NULL,
 first_name nvarchar(30) DEFAULT 'Islam' NOT NULL,
 last_name nvarchar(30) DEFAULT 'Magomed' NOT NULL,
 patronymic nvarchar(30) DEFAULT 'Amirovich' NOT NULL,
 telephone_number char(10) DEFAULT '9993658723' NOT NULL,
 CONSTRAINT Uniq_driver UNIQUE (telephone_number),
 status_drive bit DEFAULT 0 NOT NULL
)
GO

-- Формирование последовательности
CREATE SEQUENCE count_by
 START WITH 1
 INCREMENT BY 1;
GO


--=========================================АВТОМОБИЛЬ=========================================

-- Проверка на пустоту
IF OBJECT_ID(N'Auto_table') is NOT NULL
    DROP TABLE Auto_table
GO
-- Создание таблицы
CREATE TABLE Auto_table
(
 id_auto int IDENTITY NOT NULL,
 state_number nvarchar(10) DEFAULT 'a123мр77' NOT NULL,
 brand nvarchar(20) DEFAULT 'toyota' NOT NULL,
 color nvarchar(20) DEFAULT 'gray' NOT NULL,
 year_of_issue int DEFAULT 2015 NULL,
 car_class nvarchar(20) DEFAULT 'comfort' NOT NULL,
 CONSTRAINT PK_id_auto PRIMARY KEY (id_auto),
 CONSTRAINT Uniq_auto UNIQUE (state_number),
 driver_id int default 1 NOT NULL,
 FOREIGN KEY (driver_id) REFERENCES Driver_table (id_driver)
 -- ON DELETE NO ACTION
 -- ON DELETE SET DEFAULT
 -- ON DELETE SET NULL
ON DELETE CASCADE
ON UPDATE CASCADE
)


--=========================================ЗАКАЗ=========================================

-- Проверка на пустоту
IF OBJECT_ID(N'Order_table') is NOT NULL
    DROP TABLE Order_table
GO

-- Создание таблицы
CREATE TABLE Order_table
(
```

```
 id_order int IDENTITY NOT NULL,
 order_time DATETIME DEFAULT CURRENT_TIMESTAMP NOT NULL,
 street nvarchar(30) DEFAULT 'Mira' NOT NULL,
 build int DEFAULT 1 NOT NULL,
 luggage bit NULL,
 cost int DEFAULT 300 NOT NULL,
 comment nvarchar(300) NULL,
 payment_method bit DEFAULT 0 NOT NULL,
 car_class nvarchar(20) NULL,
 CONSTRAINT PK_id_order PRIMARY KEY (id_order),
 fk_client_id int default 1 NOT NULL,
 FOREIGN KEY (fk_client_id) REFERENCES Client_table (id_client)
 -- ON DELETE NO ACTION
 -- ON DELETE SET DEFAULT
 -- ON DELETE SET NULL
ON DELETE CASCADE
ON UPDATE CASCADE,
 fk_driver_id int default 1 NOT NULL,
 FOREIGN KEY (fk_driver_id) REFERENCES Driver_table (id_driver)
 -- ON DELETE NO ACTION
 -- ON DELETE SET DEFAULT
 -- ON DELETE SET NULL
ON DELETE CASCADE
ON UPDATE CASCADE
)
GO


--=========================================ПРЕДСТАВЛЕНИЯ=============================================

-- Проверка на пустоту
IF OBJECT_ID(N'Drive_view') is NOT NULL
    DROP VIEW Drive_view
GO

-- Представление NEWID (для функции)
CREATE VIEW get_newID AS SELECT newid() AS new_id
GO

-- Представление на основе таблиц ВОДИТЕЛЬ и АВТОМОБИЛЬ
CREATE VIEW Drive_view AS
SELECT dl.first_name, dl.last_name, dl.patronymic, dl.telephone_number,
al.state_number, al.brand, al.color, al.year_of_issue, al.car_class
FROM Driver_table dl INNER JOIN Auto_table al
ON al.driver_id = dl.id_driver
GO

-- Представление на основе таблиц ЗАКАЗ и ВОДИТЕЛЬ
CREATE VIEW Order_Driver_view AS
SELECT dt.first_name, dt.last_name, dt.patronymic, dt.telephone_number,
ot.order_time, ot.fk_driver_id
FROM Driver_table dt INNER JOIN Order_table ot
ON ot.fk_driver_id = dt.id_driver
GO

CREATE VIEW Full_order_view AS
SELECT O.order_time as order_time, O.street as street, O.build as build, O.car_class,
   C.cl_first_name as cl_first_name, C.cl_last_name as cl_last_name,
    C.cl_patronymic as cl_patronymic,
   D.first_name as dr_first_name, D.last_name as dr_last_name, D.patronymic as dr_patronymic,
D.telephone_number as dr_telephone_number,
   A.state_number as car_state_number, A.brand as car_brand, A.Color as car_color
FROM Order_table as O
LEFT JOIN Client_table as C ON O.fk_client_id = C.id_client
LEFT JOIN Driver_table as D ON O.fk_driver_id = D.id_driver
LEFT JOIN Auto_table as A ON D.id_driver = A.driver_id
GO


--=========================================ПРОЦЕДУРЫ=============================================
```

```
-- Проверка на пустоту
IF OBJECT_ID(N'suitable_car') is NOT NULL
    DROP PROCEDURE suitable_car
GO
-- Процедура для заполнения таблицы заказов
CREATE PROCEDURE suitable_car
@order_time DATETIME,
@street nvarchar(30),
@build int,
@luggage bit,
@cost int,
@payment_method bit,
@car_class nvarchar(20),
@fk_client_id int
AS
declare @suitable_driver int
-- Получаем id свободных водителей с подходящим классом автомобилей
SELECT TOP(1) @suitable_driver = dt.id_driver
FROM Auto_table au INNER JOIN Driver_table dt
ON au.driver_id = dt.id_driver
WHERE dt.status_drive = 0 AND au.car_class = @car_class
ORDER BY (SELECT new_id FROM get_newID)
-- Получаем id свободных водителей с любым классом
IF @suitable_driver is NULL
BEGIN
SELECT TOP(1) @suitable_driver = dt.id_driver
FROM Auto_table au INNER JOIN Driver_table dt
ON au.driver_id = dt.id_driver
WHERE dt.status_drive = 0
ORDER BY (SELECT new_id FROM get_newID)
END

-- Обновляем статус водителя
UPDATE Driver_table
SET status_drive = 1
WHERE id_driver = @suitable_driver

-- Заполняем таблицу
INSERT INTO Order_table(order_time, street, build, luggage, cost,
payment_method, car_class, fk_client_id, fk_driver_id)
VALUES(@order_time, @street, @build, @luggage, @cost,
@payment_method, @car_class, @fk_client_id, @suitable_driver)
GO


--=========================================ТРИГЕРЫ=========================================

-- Создаем тригер, запрещающий обновлять ФИО водителя
CREATE TRIGGER driver_fio_update_trig ON Driver_table
FOR UPDATE AS
 IF UPDATE (first_name)
    OR UPDATE (last_name)
    OR UPDATE (patronymic)
  BEGIN
   PRINT 'Нельзя изменять фамилию, имя и отчество'
   ROLLBACK TRANSACTION
  END
GO
-- Создаем тригер, запрещающий обновлять ФИО клиента
CREATE TRIGGER client_fio_update_trig ON Client_table
FOR UPDATE AS
 IF UPDATE (cl_first_name)
    OR UPDATE (cl_last_name)
    OR UPDATE (cl_patronymic)
  BEGIN
   PRINT 'Нельзя изменять фамилию, имя и отчество'
   ROLLBACK TRANSACTION
  END
GO
```

```
-- Тригер на вставку водителя через представление
CREATE TRIGGER insert_drive_view ON Drive_view
INSTEAD OF INSERT
AS BEGIN
-- Создаем временную таблицу
DECLARE @temp_table TABLE (
id int DEFAULT NEXT VALUE FOR count_by,
first_name nvarchar(30),
last_name nvarchar(30),
patronymic nvarchar(30),
telephone_number char(10),
state_number nvarchar(10),
brand nvarchar(20),
color nvarchar(20),
year_of_issue int,
car_class nvarchar(20)
)
-- Добавление во временную таблицу
INSERT INTO @temp_table(first_name, last_name, patronymic, telephone_number,
state_number, brand, color, year_of_issue, car_class)
SELECT first_name, last_name, patronymic, telephone_number, i.state_number,
i.brand, i.color, i.year_of_issue, i.car_class
FROM inserted i
-- Добавление водителя
INSERT INTO Driver_table(id_driver, first_name, last_name, patronymic, telephone_number)
SELECT id, first_name, last_name, patronymic, telephone_number
FROM @temp_table
-- Добавление автомобиль
INSERT INTO Auto_table(driver_id, state_number, brand, color, year_of_issue, car_class)
SELECT i.id, i.state_number, i.brand, i.color, i.year_of_issue, i.car_class
FROM @temp_table i, Driver_table dl WHERE i.id = dl.id_driver
 END
GO


-- Тригер на удаление водителя через представление
CREATE TRIGGER delete_view_trig ON Drive_view
INSTEAD OF DELETE
AS BEGIN
DELETE FROM Driver_table
WHERE telephone_number IN (SELECT telephone_number FROM deleted)
PRINT 'Минус водила'
   END
GO

-- Тригер на удаление заказа и смену статуса водителя
CREATE TRIGGER delete_update_trig ON Order_Driver_view
INSTEAD OF DELETE
AS
  BEGIN
DELETE FROM Order_table
WHERE fk_driver_id IN (SELECT fk_driver_id FROM deleted)
UPDATE Driver_table
SET status_drive = 0
WHERE id_driver IN (SELECT fk_driver_id FROM deleted)
  END
GO


--=========================================ЗАПОЛНЕНИЕ=========================================

-- Заполнение таблицы клиентов
INSERT INTO Client_table(cl_first_name, cl_last_name, cl_patronymic)
VALUES ('Тарасов', 'Алексей', 'Артёмович'),
('Казакова', 'Дарья', 'Георгиевна'),
('Морозов', 'Адольф', 'Яковлевич'),
('Тимофеева', 'Адель', 'Натановна'),
('Селезнёв', 'Арсений', 'Максович'),
('Белоусова', 'Альбина', 'Сергеевна'),
('Котов', 'Степан', 'Витальевич'),
('Медведева', 'Любовь', 'Антоновна'),
```

```
('Прохоров', 'Семен', 'Витальевич'),
('Воронова', 'Владислава', 'Вадимовна')


-- Заполнение таблицы скидок
INSERT INTO Sale_table(cart_number, current_sale, id_name, validity)
VALUES (296399, 10, 1, '24/08/2022'),
(304922, 12, 2, '13/04/2024'),
(492512, 8, 3, '13/04/2024'),
(370620, 2, 4, '12/08/2024'),
(237861, 5, 5, '15/02/2023'),
(304275, 9, 6, '16/02/2025'),
(104737, 11, 7, '11/10/2023'),
(251188, 14, 8, '06/07/2022'),
(349955, 3, 9, '29/05/2024'),
(211157, 7, 10, '14/10/2023')

-- Заполнение таблиц водителей и автомобилей
INSERT INTO Drive_view
VALUES('Andrew', 'Tkachenko', 'Alexandrovich', '9147856723',
'к674ту', 'toyota', 'white', 2016, 'comfort'),
('Daniil', 'Devyatkin', 'Dmitrievich', '9993097439',
'м239ор', 'kia', 'green', 2014, 'economy'),
('Anastasia', 'Piskunova', 'Eduardovna', '9147853402',
'у982оо', 'volkswagen', 'gray', 2015, 'economy'),
('Vasily', 'Koreshkov', 'Romanovich', '9641592385',
'м712сс', 'hyundai', 'yellow', 2015, 'comfort'),
('Ivan', 'Trenev', 'Sergeevich', '9999567832',
'с321тт', 'audi', 'black', 2017, 'business'),
('Nadezhda', 'Chaplinskaya', 'Vasilevna', '9184591402',
'к048тр', 'skoda', 'black', 2018, 'comfort'),
('Alexey', 'Komlev', 'Alexeyevich', '9149024567',
'н533пу', 'renault', 'silver', 2014, 'economy'),
('Irina', 'Vankina', 'Nikolaevna', '9647245602',
'а190ор', 'mitsubishi', 'gray', 2013, 'economy')
GO

-- Заполнение таблицы заказов (через процедуру)
DECLARE @time DATETIME
--(order_time, street, build, luggage, cost, payment_method, car_class, fk_client_id, fk_driver_id)
SET @time = CURRENT_TIMESTAMP
EXEC suitable_car @time, 'Street_1', 1, 1, 500, 1, 'economy', 1
SET @time = CURRENT_TIMESTAMP
EXEC suitable_car @time, 'Street_2', 42, 1, 1600, 0, 'business', 2
SET @time = CURRENT_TIMESTAMP
EXEC suitable_car @time, 'Street_3', 13, 0, 800, 1, 'economy', 3
SET @time = CURRENT_TIMESTAMP
EXEC suitable_car @time, 'Street_4', 29, 1, 2000, 1, 'comfort', 4
SET @time = CURRENT_TIMESTAMP
EXEC suitable_car @time, 'Street_5', 33, 0, 250, 0, 'economy', 5

-- Второго бизнеса нет, даем, что есть
/*
SET @time = CURRENT_TIMESTAMP
EXEC suitable_car @time, 'Street_6', 56, 0, 1000, 0, 'business', 6
*/

--==========================================ДЕМОНСТРАЦИЯ==========================================

SELECT * FROM Client_table
SELECT * FROM Sale_table
SELECT * FROM Driver_table
SELECT * FROM Auto_table
SELECT * FROM Order_table
GO


--=======================================ВСЯКИЕ ПРИКОЛЫ=======================================

-- Выводит количество уникальных значений текущих скидок
SELECT COUNT(DISTINCT current_sale) AS count_distinct_sale FROM Sale_table
```

```
GO

-- Возвращает null-записи
SET ANSI_NULLS ON
SELECT * FROM Order_table WHERE comment IS NULL
GO

-- Вывод автомобилей по году выпуска
SELECT * FROM Auto_table
ORDER BY year_of_issue ASC
GO

-- Вывод средней стоимости заказа для каждого класса автомобиля (если всего было заработано больше 1500)
SELECT AVG(cost) as online_money_by_car_class
FROM Order_table
GROUP BY car_class
HAVING SUM(cost) >= 1500
GO

-- Вывод информации о заказе, клиенте, водителе и авто (LEFT JOIN)
SELECT * FROM Full_order_view
GO


--=======================================УДАЛЕНИЕ, ИЗМЕНЕНИЕ=======================================

-- Типа выполнили заказы
DELETE FROM Order_Driver_view WHERE fk_driver_id in (1, 2, 5)
GO

-- Удаляем водителя
--DELETE FROM Driver_table WHERE telephone_number LIKE '914%'
--GO
-- Удаляем авто
--DELETE FROM Auto_table WHERE year_of_issue < 2015
--GO

-- Увеличиваем скидку тем, у кого заканчивается карта через 6 месяцев
UPDATE Sale_table SET current_sale = current_sale + 2
WHERE validity BETWEEN CONVERT(date, CURRENT_TIMESTAMP) AND
CONVERT(date, DATEADD(month, 6, CURRENT_TIMESTAMP ))
go
--=============================================ДЕМОНСТРАЦИЯ=============================================

SELECT * FROM Sale_table
SELECT * FROM Driver_table
SELECT * FROM Auto_table
SELECT * FROM Order_table
GO
```