Тестовое задание.

Часть 1.

1. Установить Oracle Database Express Edition.

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

Автоматически созданное описание

1. Разработать структуру и создать БД для сервиса сдачи автомобилей в аренду.

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

Автоматически созданное описание

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

Автоматически созданное описание

1. Основные сущности: клиенты, автомобили, заказы.

Учитывая, что используются только три сущности структура для сервиса сдачи автомобилей в аренду, будет выглядеть следующим образом:

**Таблица Customers (Клиенты):**

* customer\_id (Уникальный идентификатор клиента);
* first\_name (Имя клиента);
* last\_name (Фамилия клиента);
* email (Адрес электронной почты клиента);
* phone (Номер телефона клиента);
* address (Адрес клиента).

**Таблица Cars (Автомобили):**

* car\_id (Уникальный идентификатор автомобиля);
* make (Марка автомобиля);
* model (Модель автомобиля);
* year (Год выпуска автомобиля);
* registration (Регистрационный номер автомобиля);
* daily\_rate (Дневная ставка аренды);
* status (Статус автомобиля, например, "доступен", "арендован").

**Таблица Orders (Заказы):**

* order\_id (Уникальный идентификатор заказа);
* customer\_id (Ссылка на клиента, сделавшего заказ);
* car\_id (Ссылка на арендованный автомобиль);
* start\_date (Дата начала аренды);
* end\_date (Дата окончания аренды);
* total\_cost (Итоговая стоимость аренды).

Следующим шагом создадим таблицу Customers, в соответствии с условием:

*CREATE TABLE Customers (*

*customer\_id NUMBER PRIMARY KEY,*

*first\_name VARCHAR2(50),*

*last\_name VARCHAR2(50),*

*email VARCHAR2(100),*

*phone VARCHAR2(20),*

*address VARCHAR2(255)*

*);*

Проверим создание таблицы:

*SELECT \* FROM Customers*

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

Автоматически созданное описание

Создадим аналогичным образом таблицу Cars, в соответствии с условием:

*CREATE TABLE Cars (*

*car\_id NUMBER PRIMARY KEY,*

*make VARCHAR2(50),*

*model VARCHAR2(50),*

*year NUMBER,*

*registration VARCHAR2(20) UNIQUE,*

*daily\_rate NUMBER,*

*status VARCHAR2(20)*

*);*

Проверим создание таблицы:

*SELECT \* FROM Cars*

Изображение выглядит как текст, Шрифт, линия, программное обеспечение

Автоматически созданное описание

Создадим таблицу Orders с внешними ключами, что позволит хранить данные о заказах и связывать их с соответствующими клиентами и автомобилями:

*CREATE TABLE Orders (*

*order\_id NUMBER PRIMARY KEY,*

*customer\_id NUMBER,*

*car\_id NUMBER,*

*start\_date DATE,*

*end\_date DATE,*

*total\_cost NUMBER,*

*CONSTRAINT fk\_customer*

*FOREIGN KEY (customer\_id)*

*REFERENCES Customers(customer\_id),*

*CONSTRAINT fk\_car*

*FOREIGN KEY (car\_id)*

*REFERENCES Cars(car\_id)*

*);*

Проверим создание таблицы:

*SELECT \* FROM Orders*

Изображение выглядит как текст, программное обеспечение, Шрифт, Значок на компьютере

Автоматически созданное описание

1. Реализовать:
   1. Скрипты для наполнения БД тестовыми данными.

Для наполнения базы данных тестовыми данными в таблицу Customers внесем следующие данные:

*INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)*

*VALUES (1, 'Иван', 'Иванов', 'ivan.ivanov@email.com', '8-800-123-4567', 'Московская улица, 1, Ростов-на-Дону');*

*INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)*

*VALUES (2, 'Екатерина', 'Петрова', 'ekaterina.petrova@email.com', '8-800-987-6543', 'Санкт-Петербургская улица, 2, Ростов-на-Дону');*

*INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)*

*VALUES (3, 'Павел', 'Смирнов', 'pavel.smirnov@email.com', '8-800-555-5555', 'Красная площадь, 3, Ростов-на-Дону');*

*INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)*

*VALUES (4, 'Анна', 'Козлова', 'anna.kozlova@email.com', '8-800-888-1234', 'Пресненская набережная, 4, Ростов-на-Дону');*

*INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)*

*VALUES (5, 'Дмитрий', 'Медведев', 'dmitry.medvedev@email.com', '8-800-444-7777', 'Большая Дмитровка, 5, Ростов-на-Дону');*

*INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)*

*VALUES (6, 'Светлана', 'Крылова', 'svetlana.krylova@email.com', '8-800-222-9999', 'Тверская улица, 6, Ростов-на-Дону');*

*INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)*

*VALUES (7, 'Алексей', 'Соколов', 'alexei.sokolov@email.com', '8-800-111-3333', 'Казанская улица, 7, Ростов-на-Дону');*

*INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)*

*VALUES (8, 'Мария', 'Волкова', 'maria.volkova@email.com', '8-800-999-2222', 'Невский проспект, 8, Ростов-на-Дону');*

*INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)*

*VALUES (9, 'Сергей', 'Зайцев', 'sergei.zaitsev@email.com', '8-800-777-4444', 'Ленинский проспект, 9, Ростов-на-Дону');*

*INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)*

*VALUES (10, 'Ольга', 'Лебедева', 'olga.lebedeva@email.com', '8-800-666-8888', 'Арбатская площадь, 10, Ростов-на-Дону');*

Изображение выглядит как текст, Шрифт, число, программное обеспечение

Автоматически созданное описание

Для наполнения базы данных тестовыми данными в таблицу Cars внесем следующие данные:

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (1, 'Toyota', 'Camry', 2020, 'А111АА123', 50.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (2, 'Honda', 'Civic', 2019, 'А222ВВ123', 45.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (3, 'Ford', 'Escape', 2021, 'Е777КК123', 55.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (4, 'Chevrolet', 'Malibu', 2018, 'М333ОО456', 40.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (5, 'Volkswagen', 'Passat', 2022, 'Т555ЕЕ123', 60.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (6, 'Nissan', 'Altima', 2019, 'А999АА123', 45.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (7, 'Kia', 'Sorento', 2021, 'Р777УУ123', 55.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (8, 'Hyundai', 'Tucson', 2020, 'О123ОО123', 52.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (9, 'Mercedes-Benz', 'E-Class', 2022, 'У555УУ123', 80.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (10, 'Audi', 'A4', 2023, 'О999ОО123', 75.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (11, 'BMW', 'X5', 2021, 'Р888УУ123', 70.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (12, 'Lexus', 'RX', 2020, 'А777АА123', 65.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (13, 'Volvo', 'XC90', 2022, 'М111ММ123', 70.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (14, 'Subaru', 'Outback', 2019, 'Н444НН123', 55.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (15, 'Mazda', 'CX-5', 2020, 'О555ОО123', 60.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

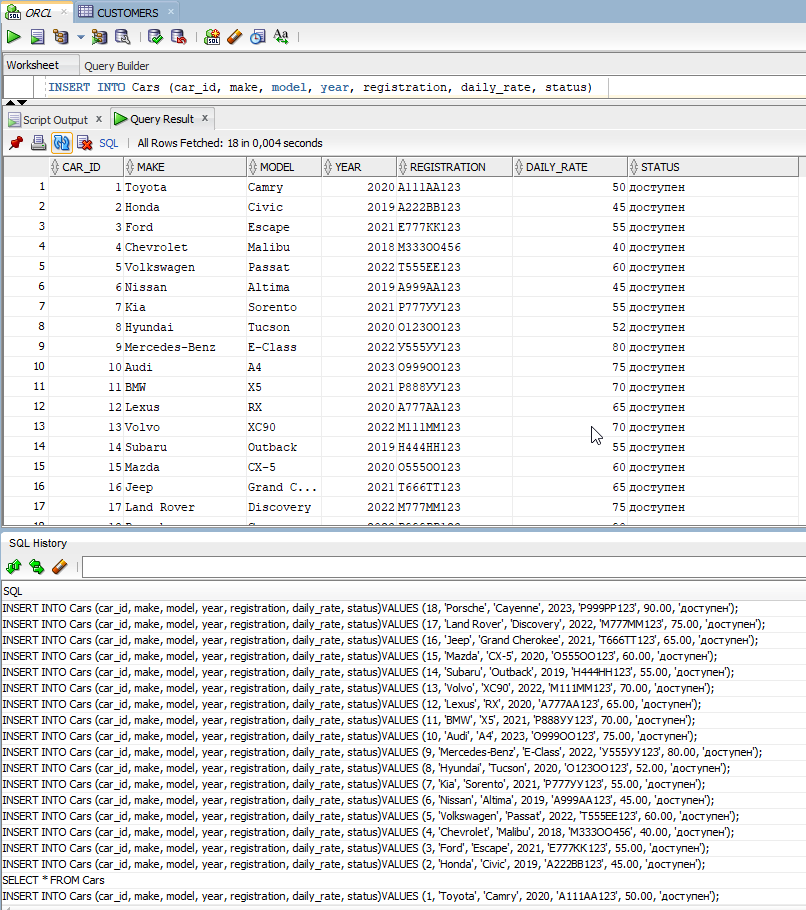
*VALUES (16, 'Jeep', 'Grand Cherokee', 2021, 'Т666ТТ123', 65.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (17, 'Land Rover', 'Discovery', 2022, 'М777ММ123', 75.00, 'доступен');*

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (18, 'Porsche', 'Cayenne', 2023, 'Р999РР123', 90.00, 'доступен');*



Для наполнения базы данных тестовыми данными в таблицу Orders внесем следующие данные:

*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

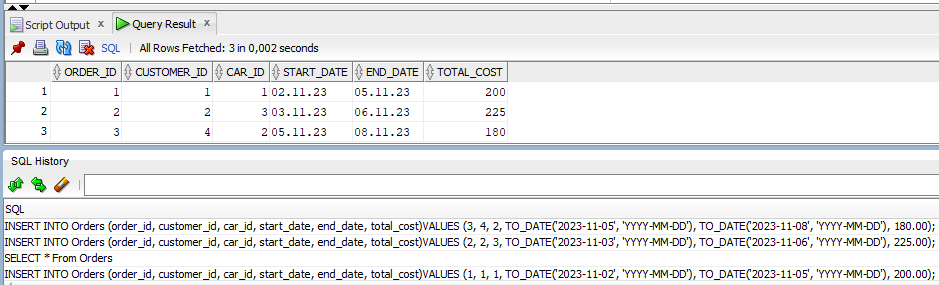
*VALUES (1, 1, 1, TO\_DATE('2023-11-02', 'YYYY-MM-DD'), TO\_DATE('2023-11-05', 'YYYY-MM-DD'), 200.00);*

*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

*VALUES (2, 2, 3, TO\_DATE('2023-11-03', 'YYYY-MM-DD'), TO\_DATE('2023-11-06', 'YYYY-MM-DD'), 225.00);*

*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

*VALUES (3, 4, 2, TO\_DATE('2023-11-05', 'YYYY-MM-DD'), TO\_DATE('2023-11-08', 'YYYY-MM-DD'), 180.00);*



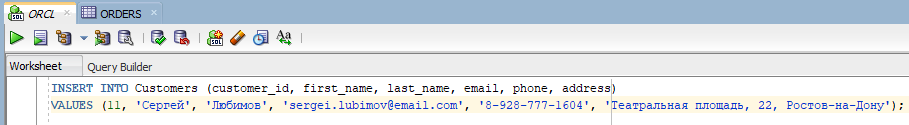
* 1. CRUD операции для всех сущностей.

CRUD операции **для сущности Customers**:

Create (Создание):

*INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address)*

*VALUES (11, 'Сергей', 'Любимов', 'sergei.lubimov@email.com', '8-928-777-1604', 'Театральная площадь, 22, Ростов-на-Дону');*



Read (Чтение):

*SELECT \* FROM Customers WHERE last\_name = 'Любимов';*

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

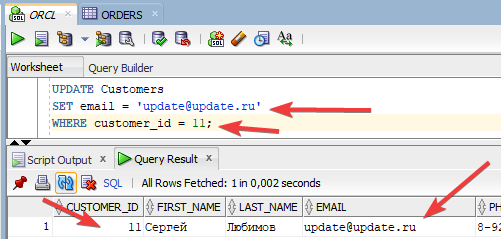
Автоматически созданное описание

Update (Обновление):

*UPDATE Customers*

*SET email = 'update@update.ru'*

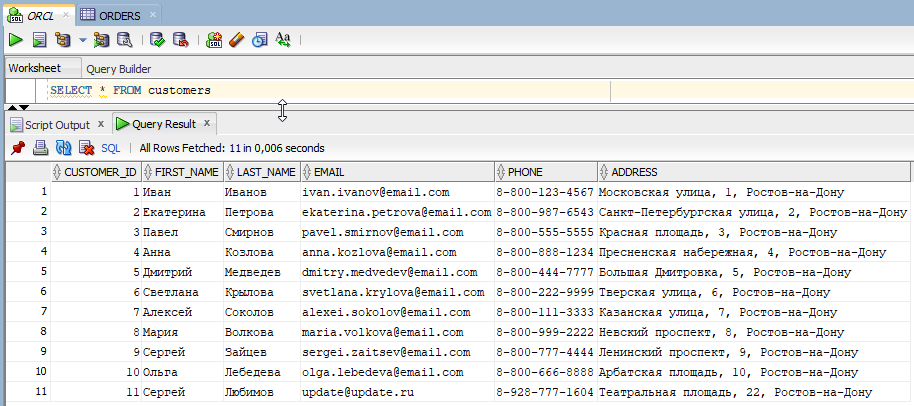
*WHERE customer\_id = 11;*

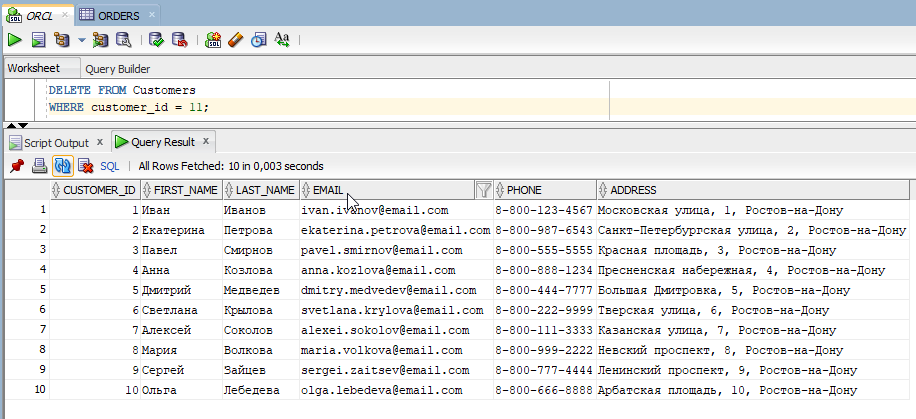


Delete (Удаление):

*DELETE FROM Customers*

*WHERE customer\_id = 11;*



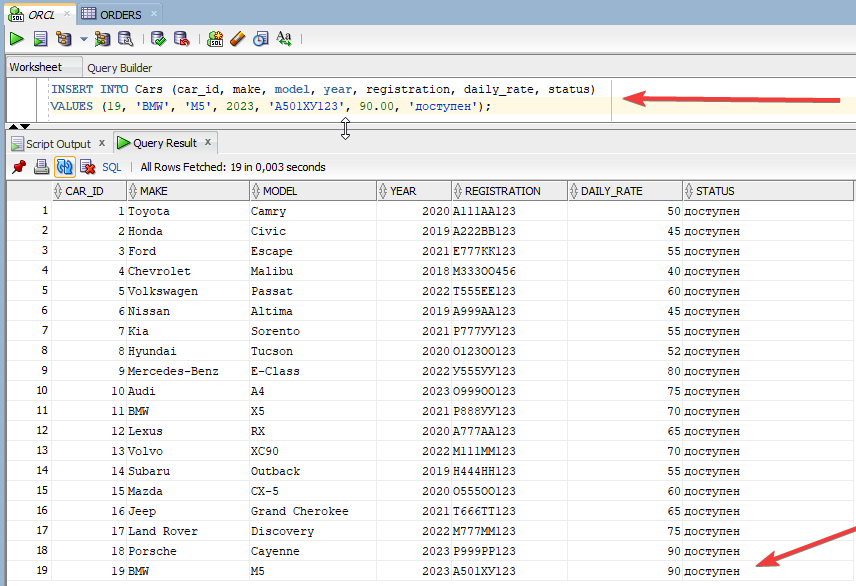


CRUD операции **для сущности Cars:**

Create (Создание):

*INSERT INTO Cars (car\_id, make, model, year, registration, daily\_rate, status)*

*VALUES (19, 'BMW', 'M5', 2023, 'А501ХУ123', 90.00, 'доступен');*



Read (Чтение):

*SELECT \* FROM Customers WHERE make = 'BMW’;*

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

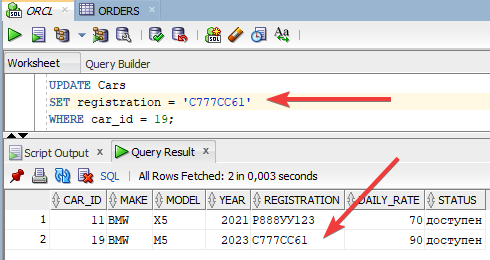
Автоматически созданное описание

Update (Обновление):

*UPDATE Cars*

*SET registration = С777СС61*

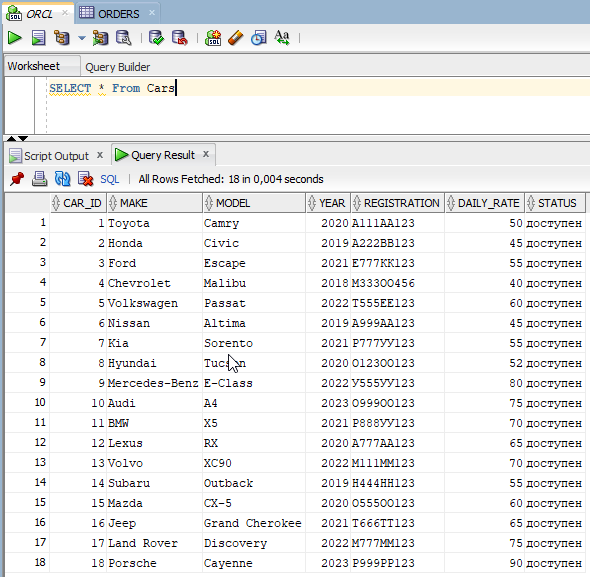
*WHERE car\_id = 19;*



Delete (Удаление):

*DELETE FROM Cars*

*WHERE car\_id = 19;*

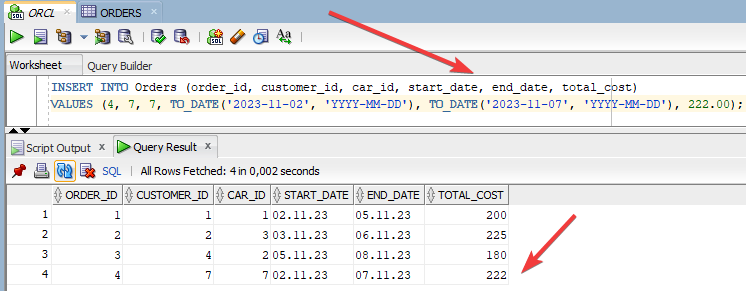


CRUD операции **для сущности Orders:**

Create (Создание):

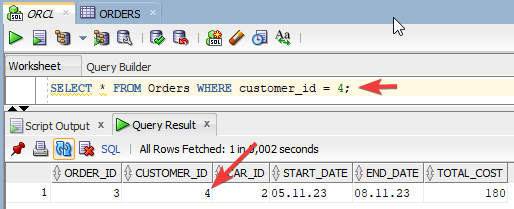
*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

*VALUES (4, 7, 7, TO\_DATE('2023-11-02', 'YYYY-MM-DD'), TO\_DATE('2023-11-07', 'YYYY-MM-DD'), 222.00);*



Read (Чтение):

*SELECT \* FROM Orders WHERE customer\_id = 4;*

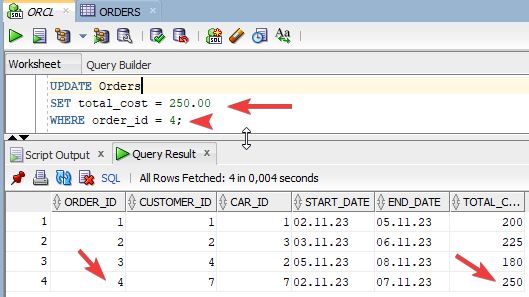


Update (Обновление):

*UPDATE Orders*

*SET total\_cost = 250.00*

*WHERE order\_id = 4;*



Delete (Удаление):

*DELETE FROM Orders*

*WHERE order\_id = 4;*

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

Автоматически созданное описание

* 1. сохранение истории изменения заказов.

Для сохранения истории изменений заказов в базе данных нам потребуется создать таблицу для хранения истории изменений, создать последовательности и триггер, который будет записывать изменения в эту таблицу:

Создание таблицы:

*CREATE TABLE OrderHistory (*

*history\_id NUMBER PRIMARY KEY,*

*customer\_id NUMBER,*

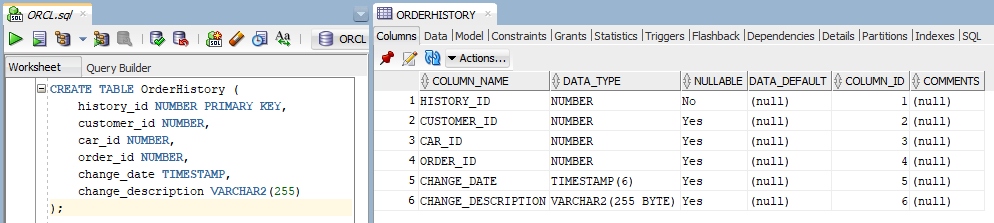
*car\_id NUMBER,*

*order\_id NUMBER,*

*change\_date TIMESTAMP,*

*change\_description VARCHAR2(255)*

*);*



Создание последовательности:

*CREATE SEQUENCE OrderHistory\_seq*

*START WITH 1*

*INCREMENT BY 1*

*NOCACHE*

*NOCYCLE;*

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Создание триггера для отслеживания изменений:

*CREATE OR REPLACE TRIGGER ORDERSHISTORYTRIGGER*

*AFTER INSERT OR UPDATE OR DELETE ON Orders*

*FOR EACH ROW*

*BEGIN*

*IF INSERTING THEN*

*INSERT INTO OrderHistory (history\_id, order\_id, customer\_id, car\_id, change\_date, change\_description)*

*VALUES (OrderHistory\_seq.NEXTVAL, :new.order\_id, :new.customer\_id, :new.car\_id, SYSTIMESTAMP, 'Order created');*

*ELSIF UPDATING THEN*

*INSERT INTO OrderHistory (history\_id, order\_id, customer\_id, car\_id, change\_date, change\_description)*

*VALUES (OrderHistory\_seq.NEXTVAL, :new.order\_id, :new.customer\_id, :new.car\_id, SYSTIMESTAMP, 'Order updated');*

*ELSIF DELETING THEN*

*INSERT INTO OrderHistory (history\_id, order\_id, customer\_id, car\_id, change\_date, change\_description)*

*VALUES (OrderHistory\_seq.NEXTVAL, :old.order\_id, :old.customer\_id, :old.car\_id, SYSTIMESTAMP, 'Order deleted');*

*END IF;*

*END;*

*/*

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Чтобы проверить сохранение истории заказов добавим информацию об аренде:

*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

*VALUES (4, 7, 7, TO\_DATE('2023-11-02', 'YYYY-MM-DD'), TO\_DATE('2023-11-07', 'YYYY-MM-DD'), 222.00);*

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Убедимся, что изменения, сделанные в таблице Orders отображены в таблице OrderHistory:

Изображение выглядит как текст, программное обеспечение, Шрифт, линия

Автоматически созданное описание

* 1. отчет об истории изменения заказов.

Выполним запрос:

*SELECT*

*oh.history\_id,*

*oh.change\_date,*

*o.order\_id,*

*o.customer\_id,*

*o.car\_id,*

*oh.change\_description*

*FROM*

*OrderHistory oh*

*JOIN*

*Orders o ON oh.order\_id = o.order\_id*

*ORDER BY*

*oh.change\_date DESC;*

Чтобы проверить работу отчета об истории изменения заказов добавим информацию об аренде:

*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

*VALUES (5, 3, 4, TO\_DATE('2023-11-03', 'YYYY-MM-DD'), TO\_DATE('2023-11-08', 'YYYY-MM-DD'), 100.00);*

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

Автоматически созданное описание

* 1. сохранение истории сдачи в аренду авто.

Для реализации сохранения истории сдачи автомобилей в аренду мы создадим таблицу «RentalHistory» и создадим триггер для записи изменений в эту таблицу:

Создание таблицы для истории аренды:

*CREATE TABLE RentalHistory (*

*rental\_id NUMBER PRIMARY KEY,*

*car\_id NUMBER,*

*customer\_id NUMBER,*

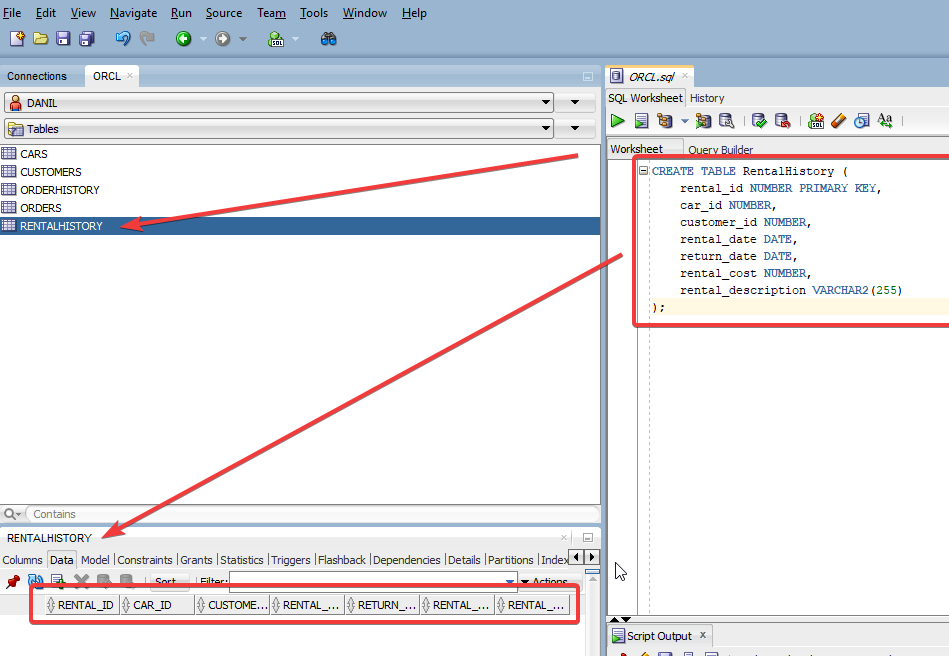
*rental\_date DATE,*

*return\_date DATE,*

*rental\_cost NUMBER,*

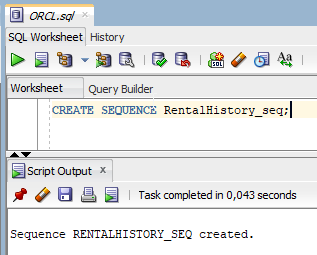
*rental\_description VARCHAR2(255)*

*);*



Создание последовательности «RentalHistory\_seq», ее мы будем использовать для генерации уникальных идентификаторов:

*CREATE SEQUENCE RentalHistory\_seq;*



Создадим триггер для записи сдачи аренды авто:

*CREATE OR REPLACE TRIGGER RentalHistoryTrigger*

*AFTER INSERT ON Orders*

*FOR EACH ROW*

*BEGIN*

*INSERT INTO RentalHistory (rental\_id, car\_id, customer\_id, rental\_date, return\_date, rental\_cost, rental\_description)*

*VALUES (*

*RentalHistory\_seq.NEXTVAL,*

*:new.car\_id,*

*:new.customer\_id,*

*:new.start\_date,*

*:new.end\_date,*

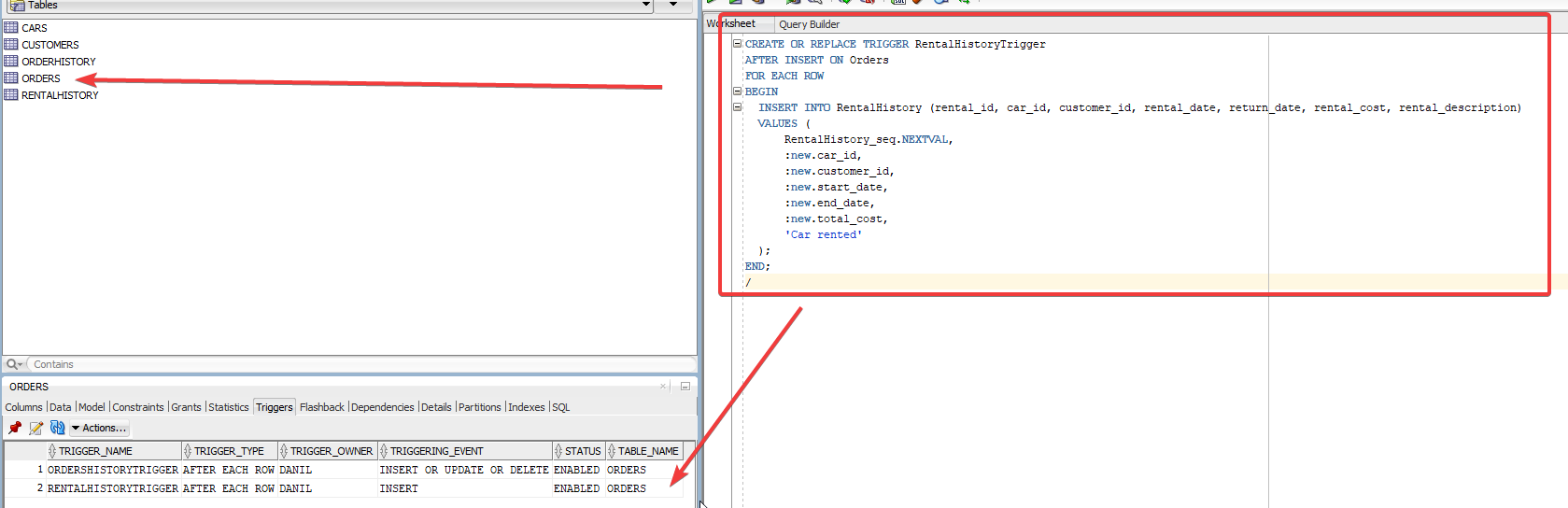
*:new.total\_cost,*

*'Car rented'*

*);*

*END;*

*/*



Создадим триггер для записи возврата аренды авто:

*CREATE OR REPLACE TRIGGER ReturnHistoryTrigger*

*AFTER UPDATE ON Orders*

*FOR EACH ROW*

*BEGIN*

*IF :new.end\_date IS NOT NULL AND :old.end\_date IS NULL THEN*

*INSERT INTO RentalHistory (rental\_id, car\_id, customer\_id, rental\_date, return\_date, rental\_cost, rental\_description)*

*VALUES (*

*RentalHistory\_seq.NEXTVAL,*

*:new.car\_id,*

*:new.customer\_id,*

*:new.start\_date,*

*:new.end\_date,*

*:new.total\_cost,*

*'Car returned'*

*);*

*END IF;*

*END;*

*/*

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

Автоматически созданное описание

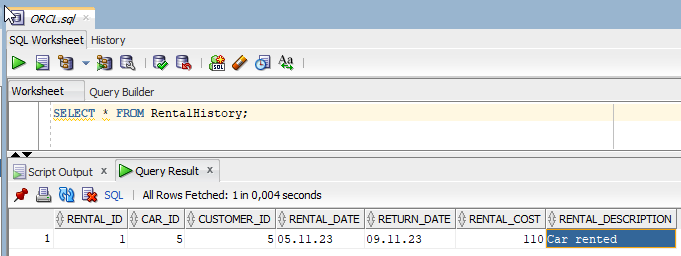
Чтобы проверить сохранение истории сдачи в аренду авто, создадим тестовую ситуацию, выполним команду:

*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

*VALUES (6, 5, 5, TO\_DATE('2023-11-05', 'YYYY-MM-DD'), TO\_DATE('2023-11-09', 'YYYY-MM-DD'), 100.00);*

Для проверки сохранения истории сдачи в аренду автомобиля выполним SQL-запрос, который вернет записи из таблицы «RentalHistory»:

*SELECT \* FROM RentalHistory;*



* 1. отчет об истории сдачи в аренду авто.

Выполним запрос:

*SELECT*

*rental\_id,*

*car\_id,*

*customer\_id,*

*rental\_date,*

*return\_date,*

*rental\_cost,*

*rental\_description*

*FROM*

*RentalHistory*

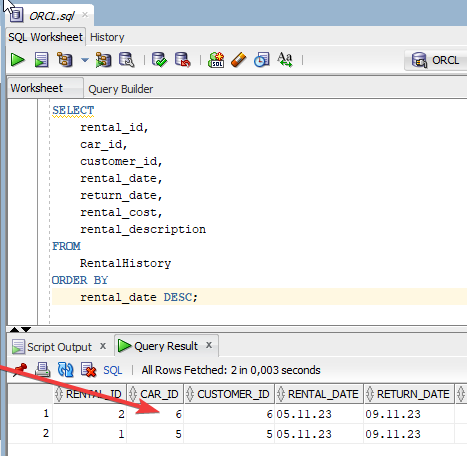
*ORDER BY*

*rental\_date DESC;*

Чтобы проверить работу отчета об истории сдачи в аренду авто добавим информацию об аренде:

*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

*VALUES (7, 6, 6, TO\_DATE('2023-11-05', 'YYYY-MM-DD'), TO\_DATE('2023-11-08', 'YYYY-MM-DD'), 120.00);*



* 1. отчет о финансовой деятельности сервиса за период (для простоты будем считать, что стоимость сдачи в аренду на сутки одного автомобиля константна).

Для создания отчета о финансовой деятельности сервиса сдачи автомобилей в аренду за определенный период будем использовать SQL-запрос, который учитывает стоимость аренды на сутки и агрегирует данные из таблицы «RentalHistory».

Подготовка данных:

Прежде чем создавать отчет, должна быть таблица «RentalHistory» с данными о сдаче в аренду автомобилей.

SQL-запрос:

*SELECT*

*TO\_CHAR(rental\_date, 'YYYY-MM') AS rental\_month,*

*SUM(rental\_cost) AS total\_revenue*

*FROM*

*RentalHistory*

*WHERE*

*rental\_date >= TO\_DATE('2023-11-01', 'YYYY-MM-DD')*

*AND rental\_date < TO\_DATE('2023-12-01', 'YYYY-MM-DD')*

*GROUP BY*

*TO\_CHAR(rental\_date, 'YYYY-MM')*

*ORDER BY*

*rental\_month;*

Для проверки работы SQL-запроса удалю все записи из таблиц Orders, OderHistory, Rentalhistory.

После чего внесу данные о заказах снова:

*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

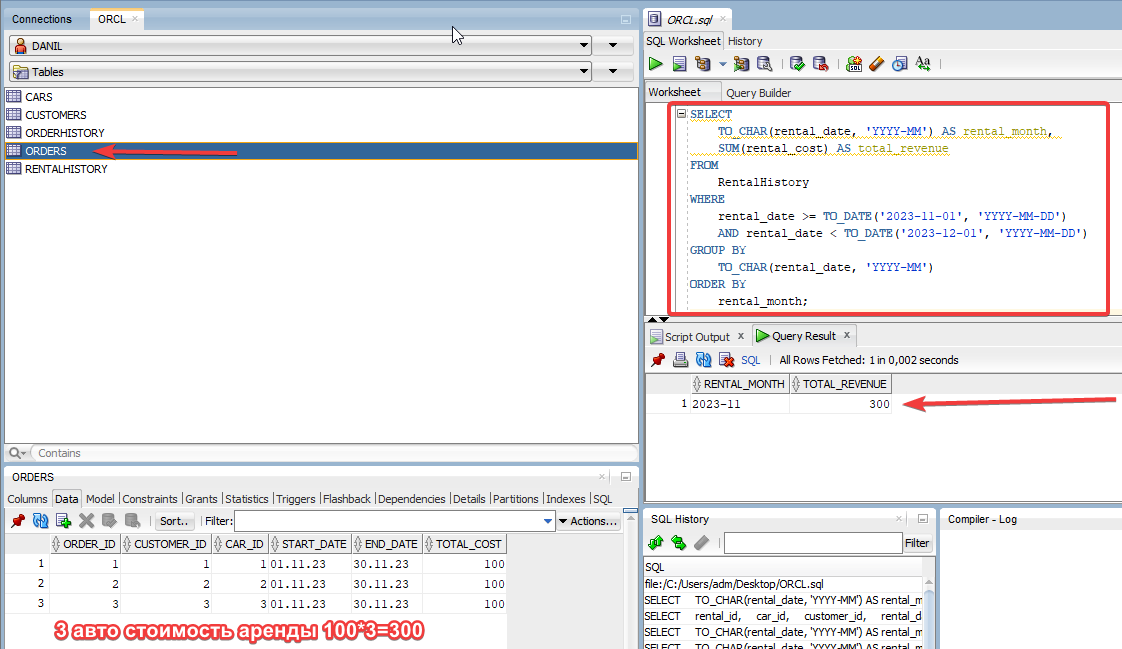
*VALUES (1,1, 1, TO\_DATE('2023-11-01', 'YYYY-MM-DD'), TO\_DATE('2023-11-30', 'YYYY-MM-DD'), 100.00);*

*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

*VALUES (2,2, 2, TO\_DATE('2023-11-01', 'YYYY-MM-DD'), TO\_DATE('2023-11-30', 'YYYY-MM-DD'), 100.00);*

*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

*VALUES (3,3, 3, TO\_DATE('2023-11-01', 'YYYY-MM-DD'), TO\_DATE('2023-11-30', 'YYYY-MM-DD'), 100.00);*



* 1. Отчет о клиентах: когда какие авто брал в аренду, сколько заплатил и т.д.

Для создания отчета о клиентах, их заказах, арендованных автомобилях и общих расходах мы будем использовать SQL-запрос:

*SELECT*

*C.customer\_id,*

*C.first\_name,*

*C.last\_name,*

*O.order\_id,*

*CA.make,*

*CA.model,*

*O.start\_date,*

*O.end\_date,*

*O.total\_cost*

*FROM*

*Customers C*

*JOIN*

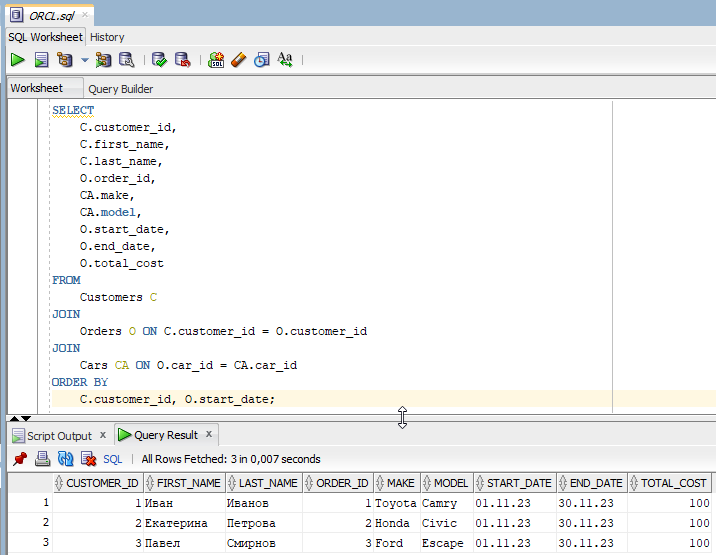
*Orders O ON C.customer\_id = O.customer\_id*

*JOIN*

*Cars CA ON O.car\_id = CA.car\_id*

*ORDER BY*

*C.customer\_id, O.start\_date;*



* 1. Отчет о текущем состоянии автопарка: сколько в аренде, сколько свободны, сколько на ТО и т.д.

Для создания отчета о клиентах, их заказах, арендованных автомобилях и общих расходах мы будем использовать SQL-запрос:

*SELECT*

*COUNT(CASE WHEN status = 'в аренде' THEN 1 ELSE NULL END) AS cars\_in\_rent,*

*COUNT(CASE WHEN status = 'свободен' THEN 1 ELSE NULL END) AS cars\_available,*

*COUNT(CASE WHEN status = 'на ТО' THEN 1 ELSE NULL END) AS cars\_under\_maintenance*

*FROM Cars;*

На данном этапе была установлена проблема, что после добавления информации в таблицу Orders, у нас не обновляется информация в таблице Cars (не меняется статус), для обновления мы будем использовать триггер:

*CREATE OR REPLACE TRIGGER UpdateCarStatusTrigger*

*AFTER INSERT OR UPDATE OR DELETE ON Orders*

*FOR EACH ROW*

*BEGIN*

*IF INSERTING THEN*

*UPDATE Cars*

*SET status = 'в аренде'*

*WHERE car\_id = :new.car\_id;*

*ELSIF DELETING THEN*

*UPDATE Cars*

*SET status = 'доступен'*

*WHERE car\_id = :old.car\_id;*

*END IF;*

*END;*

*/*

Для проверки работы SQL-запроса удалю все записи из таблиц Orders, OderHistory, Rentalhistory.

После чего внесу данные о заказах снова:

*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

*VALUES (1,1, 1, TO\_DATE('2023-11-01', 'YYYY-MM-DD'), TO\_DATE('2023-11-30', 'YYYY-MM-DD'), 100.00);*

*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

*VALUES (2,2, 2, TO\_DATE('2023-11-01', 'YYYY-MM-DD'), TO\_DATE('2023-11-30', 'YYYY-MM-DD'), 100.00);*

*INSERT INTO Orders (order\_id, customer\_id, car\_id, start\_date, end\_date, total\_cost)*

*VALUES (3,3, 3, TO\_DATE('2023-11-01', 'YYYY-MM-DD'), TO\_DATE('2023-11-30', 'YYYY-MM-DD'), 100.00);*

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

* 1. Подбор(поиск) авто по параметрам

Для реализации поиска автомобилей по параметрам воспользуемся SQL-запросами и WHERE-условиями, чтобы фильтровать данные в таблице "Cars" в соответствии с заданными параметрами:

*SELECT \**

*FROM Cars*

*WHERE*

*make = 'Toyota'*

*AND year >= 2020*

*AND daily\_rate <= 100.00*

*AND status = 'в аренде';*

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

*SELECT \**

*FROM Cars*

*WHERE*

*make = 'BMW'*

*AND year >= 2021*

*AND daily\_rate <= 100.00*

*AND status = 'доступен';*

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

Автоматически созданное описание