Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

[Кафедра информационных](https://www.belstu.by/fakultety/fit/vm) систем и технологий

Специальность 1-40 01 01 Программное обеспечение информационных технологий

**Отчёт по лабораторной работе №2**

по дисциплине Проектирование и разработка баз данных интернет-приложений

Тема: Реализация базы данных

Исполнитель:

Студент 3 курса группы 6

Станчик Максим Андреевич

Руководитель:

Ассистент Нистюк О. А.

Минск, 2025

1. **Список таблиц в СУБД MS SQL Server**

**Таблица CLIENT**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | INT | Primary key, identity (1,1) | Уникальный идентификатор клиента |
| LOGIN | NVARCHAR(255) | Not null | Логин клиента |
| PASSWORD | NVARCHAR(255) | Not null | Пароль клиента |

**Таблица CLIENT\_INFO**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | INT | Primary key, identity (1, 1) | Уникальный идентификатор информации о клинте |
| CLINT\_ID | INT | Not null | Идентификатор клиента, внешний ключ |
| FIRST\_NAME | NVARCHAR(50) | Not null | Имя клиента |
| LAST\_NAME | NVARCHAR(50) | Not null | Фамилия клиента |
| MIDDLE\_NAME | NVARCHAR(50) | Not null | Отчество клиента |
| ADDRESS | NVARCHAR(255) | - | Адрес клиента |
| EMAIL | NVARCHAR(100) | - | Электронная почта клиента |
| PHONE\_NUMBER | NVARCHAR(50) | - | Номер телефона клиента |

**Таблица DRIVER**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | INT | Primary key, identity (1, 1) | Уникальный идентификатор водителя |
| LOGIN | NVARCHAR(255) | Not null | Логин водителя |
| PASSWORD | NVARCHAR(255) | Not null | Пароль водителя |

**Таблица DRIVER\_INFO**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | INT | Primary key, identity (1, 1) | Уникальный идентификатор таблицы |
| DRIVER\_ID | INT | Not null | Уникальный идентификатор информации о водителе, внешний ключ |
| FIRST\_NAME | NVARCHAR(50) | Not null | Имя водителя |
| LAST\_NAME | NVARCHAR(50) | Not null | Фамилия водителя |
| MIDDLE\_NAME | NVARCHAR(50) | Not null | Отчество водителя |
| ADDRESS | NVARCHAR(255) | - | Адрес водителя |
| BIRTH\_DATE | DATE | - | Дата рождения водителя |
| EMAIL | NVARCHAR(100) | - | Электронная почта водителя |
| PASSPORT\_NUM | NVARCHAR(50) | - | Номер паспорта водителя |
| PHONE\_NUMBER | NVARCHAR(50) | - | Номер телефона водителя |
| VEHICLE\_INFO | INT | Not null | Внешний ключ |

**Таблица CARGO**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | INT | Primary key, identity (1, 1) | Идентификатор груза |
| NAME | NVARCHAR(50) | Not null | Название груза |
| DESCRIPTION | NVARCHAR(255) | - | Описание груза |
| WEIGHT | DECIMAL(10, 2) | Not null | Вес груза |
| VOLUME | DECIMAL(10, 2) | Not null | Объем груза |
| CARGO\_TYPE | INT | - | Тип груза |
| HAZARDOUS | NVARCHAR(7) | Может иметь значение 'No', 'Class 1', 'Class 2', 'Class 3', 'Class 4', 'Class 5', 'Class 6', 'Class 7', 'Class 8', 'Class 9' | Опасность груза |

**Таблица ORDERS**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | INT | Primary key, identity (1, 1) | Уникальный идентификатор заказа |
| CLIENT\_ID | INT | Not null | Идентификатор клиента, не может быть NULL |
| DRIVER\_ID | INT | - | Идентификатор водителя, может быть NULL |
| SERVICE\_ID | INT | Not null | Идентификатор услуги, не может быть NULL |
| STATUS | NVARCHAR(50) | Может принимать значения 'Pending', 'In Transit', 'Delivered', 'Cancellation Requested', 'Cancelled', 'On Hold', 'Returned', 'Lost', 'Awaiting Pickup', 'Out for Delivery', 'Completed', 'Exception', 'Scheduled', 'In Warehouse', 'Awaiting Clearance', 'Delayed' | Статус заказа |
| CREATED\_AT | DATETIME | Not null | Дата и время создания заказа, не может быть NULL |
| ROUTE | INT | Not null | Идентификатор маршрута, не может быть NULL |
| DELIVERY\_TYPE | NVARCHAR(50) | Not null | Тип доставки |
| TOTAL\_WEIGHT | DECIMAL(10, 2) | - | Общий вес груза |
| TOTAL\_VOLUME | DECIMAL(10, 2) | - | Общий объем груза |
| DELIVERED­\_AT | DATETIME | - | Дата и время доставки |

**Таблица VEHICLE\_TYPE**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | INT | Primary key, identity (1, 1) | Уникальный идентификатор типа транспорта |
| NAME | NVARCHAR(255) | Not null | Название типа транспорта |
| DESCRIPTION | NVARCHAR(255) | Not null | Описание типа транспорта |

**Таблица VEHICLE**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | INT | Primary key, identity (1, 1) | Уникальный идентификатор типа транспорта |
| DRIVER\_ID | INT | Not null | Название типа транспорта, внешний ключ |
| VEHICLE\_TYPE | INT | Not null | Описание типа транспорта, внешний ключ |
| LICENSE\_PLATE | NVARCHCHAR(20) | Not null | Номерной знак |
| MODEL | NVARCHAR(50) | Not null | Модель транспорта |
| CAPACITY | DECIMAL(10, 2) | Not null | Вместимость транспорта |
| INSURANCE\_EXPIRY | DATE | Not null | Истечение срока страхования |
| STATUS | NVARCHAR(20) | Может принимать значения 'In transit', 'Arrived', 'At stop', 'Delayed', 'Cancelled', 'Under maintenance', 'Available', 'On route', 'Delivered', 'Waiting', 'Issue with cargo', 'Free', 'Busy', 'Searching', 'Completed', 'On platform' | Статус транспорта |

**Таблица SERVICE\_TYPE**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | INT | Primary key, identity (1, 1) | Уникальный идентификатор типа сервиса |
| NAME | NVARCHAR(255) | Not null | Название типа сервиса |
| DESCRIPTION | NVARCHAR(255) | Not null | Описание типа сервиса |

**Таблица SERVICES**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | INT | Primary key, identity (1, 1) | Уникальный идентификатор сервиса |
| NAME | NVARCHAR(255) | Not null | Название сервиса |
| DESCRIPTION | NVARCHAR(255) | Not null | Описание сервиса |
| BASE\_RATE | DECIMAL(10, 2) | - | Базовая ставка (стоимость) услуги |
| SERVICE\_TYPE\_ID | INT | - | Внешний ключ |
| CREATED\_AT | DATETIME | DEFAULT GETDATE() | Дата и время создания записи о сервисе |

**Таблица ROUTES**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | INT | Primary key, identity (1, 1) | Уникальный идентификатор типа груза |
| START\_LOCATION | NVARCHAR(255) | Not null | Стартовая локация |
| END\_LOCATION | NVARCHAR(255) | Not null | Конечная локация |
| DISTANCE | DECIMAL(10, 2) | - | Расстояние |

**Таблица CARGO\_TYPE**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | INT | Primary key, identity (1, 1) | Уникальный идентификатор типа груза |
| NAME | NVARCHAR(255) | Not null | Название типа груза |
| DESCRIPTION | NVARCHAR(255) | Not null | Описание типа груза |

1. **Прочие объекты в СУБД MS SQL Server**

**ClientOrdersSummary:**

* Тип: представление.
* Назначение: предоставляет информацию о клиентах и их заказах.
* Зависимые объекты: зависит от таблиц CLIENT и CLIENT\_INFO.

**DriverVehicleInfo:**

* Тип: представление.
* Назначение: предоставляет информацию о водителях и их транспортных средствах.
* Зависимые объекты: зависит от таблиц VEHICLE и VEHICLE\_TYPE.

**OrderStatusInfo:**

* Тип: представление.
* Назначение: предоставляет информацию о заказах и их статусах.
* Зависимые объекты: зависит от таблиц ORDERS и ROUTES.

**IX\_Orders\_ClientID:**

* Тип: индекс.
* Назначение: индексирует столбец CLIENT\_ID в таблице ORDERS для повышения производительности при выполнении запросов, которые используют сортировки по этому полю.

**IX\_Vehicle\_DriverID:**

* Тип: индекс.
* Назначение: индексирует столбец DRIVER\_ID в таблице VEHICLE для повышения производительности при выполнении запросов, которые используют эти поля.

**trg\_UpdateDeliveredAt:**

* Тип: триггер.
* Назначение: автоматически обновляет дату и время доставки для заказов, когда их статус меняется на 'Delivered'.
* Зависимые объекты: таблица ORDERS.

Скрипт для создания объектов приведен в листинге 2.1.

|  |
| --- |
| CREATE VIEW BOOK\_DETAILS AS  CREATE INDEX IX\_Orders\_ClientID ON ORDERS (CLIENT\_ID);  CREATE INDEX IX\_Vehicle\_DriverID ON VEHICLE (DRIVER\_ID);  EXEC sp\_helpindex 'ORDERS';  EXEC sp\_helpindex 'VEHICLE';  CREATE TRIGGER trg\_UpdateDeliveredAt  ON ORDERS  AFTER UPDATE  AS  BEGIN  SET NOCOUNT ON;  IF EXISTS (SELECT 1 FROM inserted WHERE STATUS = 'Delivered')  BEGIN  UPDATE ORDERS  SET DELIVERED\_AT = GETDATE()  WHERE ID IN (SELECT ID FROM inserted WHERE STATUS = 'Delivered');  END  END;  CREATE VIEW OrderStatusInfo AS  SELECT o.ID AS OrderID, o.STATUS, o.DELIVERY\_TYPE, r.START\_LOCATION, r.END\_LOCATION, o.CREATED\_AT, o.DELIVERED\_AT  from ORDERS o JOIN ROUTES r ON o.ROUTE = r.ID;  create view DriverVehicleInfo as select  d.ID as DriverID, di.FIRST\_NAME, di.LAST\_NAME, v.LICENSE\_PLATE, vt.NAME as VehicleType, v.STATUS as VehicleStatus  from DRIVER d join DRIVER\_INFO di On d.ID = di.DRIVER\_ID join VEHICLE v on d.ID = v.DRIVER\_ID join VEHICLE\_TYPE vt ON v.VEHICLE\_TYPE = vt.ID;  create view ClientOrdersSummary as  select c.ID as ClientID, ci.FIRST\_NAME, ci.LAST\_NAME, count(o.ID) as TotalOrders, sum(o.TOTAL\_WEIGHT) as TotalWeight  from client c join client\_info ci on c.ID = ci.client\_id left join orders o ON c.ID = o.CLIENT\_ID group by c.ID, ci.FIRST\_NAME, ci.LAST\_NAME; |

Листинг 2.1 – Скрипт для создания представлений, индексов и триггера

1. **Хранимые процедуры в СУБД MS SQL Server**

**Хранимая процедура GetAllCargo:**

* Параметры отсутствуют
* Назначение: выводит все грузы из базы данных.
* Зависимости: использует таблицу CARGO и выводит из нее все строки.

**Хранимая процедура DeleteCargo:**

* Параметры:
* @CargoID (INT): Идентификатор груза для удаления.
* Назначение: удаляет груз из базы данных.
* Зависимости: использует таблицу CARGO.

**Хранимая процедура AddOrder:**

* Параметры:
* @ClientID INT: Идентификатор клиента, который размещает заказ.
* @DriverID INT: Идентификатор водителя, который будет назначен для выполнения заказа.
* @ServiceID INT: Идентификатор услуги, связанной с заказом.
* @Status NVARCHAR(50): Статус заказа.
* @Route INT: Идентификатор маршрута, по которому будет осуществляться доставка.
* @DeliveryType NVARCHAR(50): Тип доставки (например, 'Standard', 'Express' и т. д.).
* @TotalWeight DECIMAL(10, 2): Общий вес груза в заказе.
* @TotalVolume DECIMAL(10, 2): Общий объем груза в заказе.
* Назначение: создавать новый заказ с заданными параметрами, проверяя при этом существование клиента, водителя, услуги и валидность статуса заказа. Если все проверки проходят успешно, новый заказ добавляется в таблицу ORDERS.
* Зависимости: использует таблицы CLIENT, DRIVER, SERVICES, ORDERS.

**Хранимая процедура UpdateOrderStatus:**

* Параметры:
* @OrderID (INT): Идентификатор заказа, статус которого необходимо обновить.
* @Status NVARCHAR(50): Новый статус, который необходимо установить для заказа.
* Назначение: обновить статус существующего заказа, проверив при этом, что заказ с указанным идентификатором существует, и что новый статус является допустимым.
* Зависимости: использует таблицу ORDERS.

**Хранимая процедура GetDriverAndVehicles:**

* Параметры:
* @DriverID (INT): Идентификатор водителя, информацию о котором необходимо получить.
* Назначение: извлечение данных о водителе и его транспортных средствах из базы данных, проверяя при этом существование водителя.
* Зависимости: использует таблицы VEHICLE и DRIVER.

Скрипт для создания хранимых процедур приведен в листинге 3.1.

|  |
| --- |
| CREATE PROCEDURE GetAllCargo  AS  BEGIN  SELECT \* FROM CARGO;  END;  CREATE PROCEDURE DeleteCargo  @CargoID INT  AS  BEGIN  BEGIN TRY  IF NOT EXISTS (SELECT 1 FROM CARGO WHERE ID = @CargoID)  BEGIN  RAISERROR('Cargo with ID %d does not exist.', 16, 1, @CargoID);  RETURN;  END  DELETE FROM CARGO WHERE ID = @CargoID;  SELECT 'Cargo deleted successfully.' AS Message;  END TRY  BEGIN CATCH  DECLARE @ErrorMessage NVARCHAR(4000);  DECLARE @ErrorSeverity INT;  DECLARE @ErrorState INT;  SELECT  @ErrorMessage = ERROR\_MESSAGE(),  @ErrorSeverity = ERROR\_SEVERITY(),  @ErrorState = ERROR\_STATE();  RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);  END CATCH  END;  CREATE PROCEDURE AddOrder  @ClientID INT,  @DriverID INT,  @ServiceID INT,  @Status NVARCHAR(50),  @Route INT,  @DeliveryType NVARCHAR(50),  @TotalWeight DECIMAL(10, 2),  @TotalVolume DECIMAL(10, 2)  AS  BEGIN  BEGIN TRY  IF NOT EXISTS (SELECT 1 FROM CLIENT WHERE ID = @ClientID)  BEGIN  RAISERROR('Client with ID %d does not exist.', 16, 1, @ClientID);  RETURN;  END  IF NOT EXISTS (SELECT 1 FROM DRIVER WHERE ID = @DriverID)  BEGIN  RAISERROR('Driver with ID %d does not exist.', 16, 1, @DriverID);  RETURN;  END  IF NOT EXISTS (SELECT 1 FROM SERVICES WHERE ID = @ServiceID)  BEGIN  RAISERROR('Service with ID %d does not exist.', 16, 1, @ServiceID);  RETURN;  END  DECLARE @ValidStatuses TABLE (Status NVARCHAR(50));  INSERT INTO @ValidStatuses (Status) VALUES ('Pending'), ('In Transit'), ('Delivered'), ('Cancelled'), ('On Hold'), ('Returned'), ('Lost');  IF NOT EXISTS (SELECT 1 FROM @ValidStatuses WHERE Status = @Status)  BEGIN  RAISERROR('Invalid status: %s. Valid statuses are: Pending, In Transit, Delivered, Cancelled, On Hold, Returned, Lost.', 16, 1, @Status);  RETURN;  END  INSERT INTO ORDERS (CLIENT\_ID, DRIVER\_ID, SERVICE\_ID, STATUS, ROUTE, DELIVERY\_TYPE, TOTAL\_WEIGHT, TOTAL\_VOLUME)  VALUES (@ClientID, @DriverID, @ServiceID, @Status, @Route, @DeliveryType, @TotalWeight, @TotalVolume);  SELECT 'Order added successfully.' AS Message;  END TRY  BEGIN CATCH  DECLARE @ErrorMessage NVARCHAR(4000);  DECLARE @ErrorSeverity INT;  DECLARE @ErrorState INT;  SELECT  @ErrorMessage = ERROR\_MESSAGE(),  @ErrorSeverity = ERROR\_SEVERITY(),  @ErrorState = ERROR\_STATE();  RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);  END CATCH  END;  CREATE PROCEDURE UpdateOrderStatus  @OrderID INT,  @Status NVARCHAR(50)  AS  BEGIN  BEGIN TRY  IF NOT EXISTS (SELECT 1 FROM ORDERS WHERE ID = @OrderID)  BEGIN  RAISERROR('Order with ID %d does not exist.', 16, 1, @OrderID);  RETURN;  END  DECLARE @ValidStatuses TABLE (Status NVARCHAR(50));  INSERT INTO @ValidStatuses (Status) VALUES ('Pending'), ('In Transit'), ('Delivered'), ('Cancelled'), ('On Hold'), ('Returned'), ('Lost');  IF NOT EXISTS (SELECT 1 FROM @ValidStatuses WHERE Status = @Status)  BEGIN  RAISERROR('Invalid status: %s. Valid statuses are: Pending, In Transit, Delivered, Cancelled, On Hold, Returned, Lost.', 16, 1, @Status);  RETURN;  END  UPDATE ORDERS SET STATUS = @Status WHERE ID = @OrderID;  SELECT 'Order status updated successfully.' AS Message;  END TRY  BEGIN CATCH  DECLARE @ErrorMessage NVARCHAR(4000);  DECLARE @ErrorSeverity INT;  DECLARE @ErrorState INT;  SELECT  @ErrorMessage = ERROR\_MESSAGE(),  @ErrorSeverity = ERROR\_SEVERITY(),  @ErrorState = ERROR\_STATE();  RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);  END CATCH  END;  CREATE PROCEDURE GetDriverAndVehicles  @DriverID INT  AS  BEGIN  BEGIN TRY  IF NOT EXISTS (SELECT 1 FROM DRIVER WHERE ID = @DriverID)  BEGIN  RAISERROR('Driver with ID %d does not exist.', 16, 1, @DriverID);  RETURN;  END  SELECT D.\*, V.\*  FROM DRIVER D  LEFT JOIN VEHICLE V ON D.ID = V.DRIVER\_ID  WHERE D.ID = @DriverID;  IF @@ROWCOUNT = 0  BEGIN  RAISERROR('No vehicles found for the driver with ID %d.', 16, 1, @DriverID);  END  END TRY  BEGIN CATCH  DECLARE @ErrorMessage NVARCHAR(4000);  DECLARE @ErrorSeverity INT;  DECLARE @ErrorState INT;  SELECT  @ErrorMessage = ERROR\_MESSAGE(),  @ErrorSeverity = ERROR\_SEVERITY(),  @ErrorState = ERROR\_STATE();  RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);  END CATCH  END; |

Листинг 3.1 – Скрипт для создания хранимых процедур

1. **Функции в СУБД MS SQL Server**

**Функция GetVehicleInfoByID:**

* Возвращает таблицу (тип данных TABLE), содержащую все записи из таблицы VEHICLE, где значение поля ID соответствует переданному идентификатору транспортного средства (@VehicleID).
* Параметры:
* @VehicleID INT - единственный входной параметр, принимает целочисленный ID транспортного средства
* Назначение: получение полной информации о транспортном средстве по его ID.
* Зависимости: использует таблицу VEHICLE.

**Функция GetServicesByType:**

* Возвращает таблицу (тип данных TABLE), содержащую все записи из таблицы SERVICES, где значение поля SERVICE\_TYPE\_ID соответствует переданному идентификатору типа услуги (@ServiceTypeID).
* Параметры:
* @ServiceTypeID INT - ID типа услуги (целое число)
* Назначение: возвращает общее количество заказов для указанного клиента.
* Зависимости: использует таблицу BOOK\_ORDER.

**Функция GetActiveVehicles:**

* Возвращает таблицу (тип данных TABLE), содержащую все записи из таблицы VEHICLE, где значение поля STATUS равно 'Available'.
* Параметры отсутствуют
* Назначение: получить таблицу с транспортными средствами, которые имеют статус "Available" (доступно).
* Зависимости: использует таблицу VEHICLE.

**Функция GetCargoByType:**

* Возвращает таблицу, содержащую все грузы, относящиеся к определенному типу груза.
* Параметры:
* @CargoTypeID INT - идентификатор типа груза, для которого нужно получить список грузов
* Назначение: получить таблицу с транспортными средствами, которые имеют статус "Available" (доступно).
* Зависимости: использует таблицу CARGO.

**Функция GetClientsWithOrdersAboveAmount:**

* Функция возвращает таблицу, содержащую следующие поля:
* CLIENT\_ID - идентификатор клиента.
* FIRST\_NAME - имя клиента.
* LAST\_NAME - фамилия клиента.
* TOTAL\_ORDER\_AMOUNT - общий вес заказов клиента.
* Параметры:
* @p\_Amount DECIMAL(10, 2) - параметр, представляющий сумму, выше которой нужно искать клиентов с заказами. Он должен быть положительным.
* Назначение: получение таблицы с клиентами, которые сделали заказы, общий вес которых превышает указанную сумму.
* Зависимости: использует таблицу BOOK\_ORDER.

Скрипт для создания функций приведен в листинге 4.1.

|  |
| --- |
| CREATE FUNCTION GetClientsWithOrdersAboveAmount(@p\_Amount DECIMAL(10, 2))  RETURNS TABLE  AS  RETURN (  SELECT  c.ID AS CLIENT\_ID,  ci.FIRST\_NAME,  ci.LAST\_NAME,  SUM(o.TOTAL\_WEIGHT) AS TOTAL\_ORDER\_AMOUNT  FROM  CLIENT c  JOIN  CLIENT\_INFO ci ON c.ID = ci.CLIENT\_ID  JOIN  ORDERS o ON c.ID = o.CLIENT\_ID  GROUP BY  c.ID, ci.FIRST\_NAME, ci.LAST\_NAME  HAVING  SUM(o.TOTAL\_WEIGHT) > @p\_Amount  AND @p\_Amount > 0  ); CREATE FUNCTION GetCargoByType  (  @CargoTypeID INT  )  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM CARGO  WHERE CARGO\_TYPE = @CargoTypeID  ); CREATE FUNCTION GetActiveVehicles()  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM VEHICLE  WHERE STATUS = 'Available'  ); CREATE FUNCTION GetServicesByType  @ServiceTypeID INT  RETURNS TABLE  AS  BEGIN  IF NOT EXISTS (SELECT 1 FROM SERVICE\_TYPE WHERE ID = @ServiceTypeID)  BEGIN  RAISERROR('Service type with ID %d does not exist.', 16, 1, @ServiceTypeID);  RETURN;  END  RETURN  (  SELECT \*  FROM SERVICES  WHERE SERVICE\_TYPE\_ID = @ServiceTypeID  );  END; CREATE FUNCTION dbo.GetVehicleInfoByID  (@VehicleID INT)  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM VEHICLE  WHERE ID = @VehicleID  ); |

Листинг 4.1 – Скрипт для создания функций

1. **Список таблиц в СУБД Oracle**

**Таблица CLIENT**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | number(4) | Generated by default on null as identity PRIMARY KEY | Уникальный идентификатор клиента |
| LOGIN | NVARCHAR2(255) | Not null | Логин клиента |
| PASSWORD | NVARCHAR2(255) | Not null | Пароль клиента |

**Таблица CLIENT\_INFO**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | number(4) | Generated by default on null as identity PRIMARY KEY | Уникальный идентификатор информации о клинте |
| CLINT\_ID | number(4) | Not null | Идентификатор клиента, внешний ключ |
| FIRST\_NAME | NVARCHAR2(50) | Not null | Имя клиента |
| LAST\_NAME | NVARCHAR2(50) | Not null | Фамилия клиента |
| MIDDLE\_NAME | NVARCHAR2(50) | Not null | Отчество клиента |
| ADDRESS | NVARCHAR2(255) | - | Адрес клиента |
| EMAIL | NVARCHAR2(100) | - | Электронная почта клиента |
| PHONE\_NUMBER | NVARCHAR2(50) | - | Номер телефона клиента |

**Таблица DRIVER**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | number(4) | Generated by default on null as identity PRIMARY KEY | Уникальный идентификатор водителя |
| LOGIN | NVARCHAR2(255) | Not null | Логин водителя |
| PASSWORD | NVARCHAR2(255) | Not null | Пароль водителя |

**Таблица DRIVER\_INFO**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | number(4) | Generated by default on null as identity PRIMARY KEY | Уникальный идентификатор таблицы |
| DRIVER\_ID | number(4) | Not null | Уникальный идентификатор информации о водителе, внешний ключ |
| FIRST\_NAME | NVARCHAR2(50) | Not null | Имя водителя |
| LAST\_NAME | NVARCHAR2(50) | Not null | Фамилия водителя |
| MIDDLE\_NAME | NVARCHAR2(50) | Not null | Отчество водителя |
| ADDRESS | NVARCHAR2(255) | - | Адрес водителя |
| BIRTH\_DATE | DATE | - | Дата рождения водителя |
| EMAIL | NVARCHAR2(100) | - | Электронная почта водителя |
| PASSPORT\_NUM | NVARCHAR2(50) | - | Номер паспорта водителя |
| PHONE\_NUMBER | NVARCHAR2(50) | - | Номер телефона водителя |
| VEHICLE\_INFO | number(4) | Not null | Внешний ключ |

**Таблица CARGO**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | number(4) | Generated by default on null as identity PRIMARY KEY | Идентификатор груза |
| NAME | NVARCHAR2(50) | not null | Название груза |
| DESCRIPTION | NVARCHAR2(255) | - | Описание груза |
| WEIGHT | number(10, 2) | not null | Вес груза |
| VOLUME | number(10, 2) | not null | Объем груза |
| CARGO\_TYPE | number(4) | - | Тип груза |
| HAZARDOUS | NVARCHAR2(7) | Может иметь значение 'No', 'Class 1', 'Class 2', 'Class 3', 'Class 4', 'Class 5', 'Class 6', 'Class 7', 'Class 8', 'Class 9' | Опасность груза |

**Таблица ORDERS**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | number(4) | Generated by default on null as identity PRIMARY KEY | Уникальный идентификатор заказа |
| CLIENT\_ID | number(4) | Not null | Идентификатор клиента, не может быть NULL |
| DRIVER\_ID | number(4) | - | Идентификатор водителя, может быть NULL |
| SERVICE\_ID | number(4) | Not null | Идентификатор услуги, не может быть NULL |
| STATUS | NVARCHAR2(50) | Может принимать значения 'Pending', 'In Transit', 'Delivered', 'Cancellation Requested', 'Cancelled', 'On Hold', 'Returned', 'Lost', 'Awaiting Pickup', 'Out for Delivery', 'Completed', 'Exception', 'Scheduled', 'In Warehouse', 'Awaiting Clearance', 'Delayed' | Статус заказа |
| CREATED\_AT | TIMESTAMP | Not null | Дата и время создания заказа, не может быть NULL |
| ROUTE | number(4) | Not null | Идентификатор маршрута, не может быть NULL |
| DELIVERY\_TYPE | NVARCHAR2(50) | Not null | Тип доставки |
| TOTAL\_WEIGHT | number(10, 2) | - | Общий вес груза |
| TOTAL\_VOLUME | number(10, 2) | - | Общий объем груза |
| DELIVERED­\_AT | TIMESTAMP | - | Дата и время доставки |

**Таблица VEHICLE\_TYPE**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | number(4) | Generated by default on null as identity PRIMARY KEY | Уникальный идентификатор типа транспорта |
| NAME | NVARCHAR2(255) | Not null | Название типа транспорта |
| DESCRIPTION | NVARCHAR2(255) | Not null | Описание типа транспорта |

**Таблица VEHICLE**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | number(4) | Generated by default on null as identity PRIMARY KEY | Уникальный идентификатор типа транспорта |
| DRIVER\_ID | number(4) | Not null | Название типа транспорта, внешний ключ |
| VEHICLE\_TYPE | number(4) | Not null | Описание типа транспорта, внешний ключ |
| LICENSE\_PLATE | NVARCHCHAR2(20) | Not null | Номерной знак |
| MODEL | NVARCHAR2(50) | Not null | Модель транспорта |
| CAPACITY | number(10, 2) | Not null | Вместимость транспорта |
| INSURANCE\_EXPIRY | DATE | Not null | Истечение срока страхования |
| STATUS | NVARCHAR2(20) | Может принимать значения 'In transit', 'Arrived', 'At stop', 'Delayed', 'Cancelled', 'Under maintenance', 'Available', 'On route', 'Delivered', 'Waiting', 'Issue with cargo', 'Free', 'Busy', 'Searching', 'Completed', 'On platform' | Статус транспорта |

**Таблица SERVICE\_TYPE**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | number(4) | Generated by default on null as identity PRIMARY KEY | Уникальный идентификатор типа сервиса |
| NAME | NVARCHAR2(255) | Not null | Название типа сервиса |
| DESCRIPTION | NVARCHAR2(255) | Not null | Описание типа сервиса |

**Таблица SERVICES**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | number(4) | Generated by default on null as identity PRIMARY KEY | Уникальный идентификатор сервиса |
| NAME | NVARCHAR2(255) | Not null | Название сервиса |
| DESCRIPTION | NVARCHAR2(255) | Not null | Описание сервиса |
| BASE\_RATE | number(10, 2) | - | Базовая ставка (стоимость) услуги |
| SERVICE\_TYPE\_ID | number(4) | - | Внешний ключ |
| CREATED\_AT | TIMESTAMP | DEFAULT GETDATE() | Дата и время создания записи о сервисе |

**Таблица ROUTES**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | number(4) | Generated by default on null as identity PRIMARY KEY | Уникальный идентификатор типа груза |
| START\_LOCATION | NVARCHAR2(255) | Not null | Стартовая локация |
| END\_LOCATION | NVARCHAR2(255) | Not null | Конечная локация |
| DISTANCE | number(10, 2) | - | Расстояние |

**Таблица CARGO\_TYPE**

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Тип данных | Ограничения | Назначение |
| ID | number(4) | Generated by default on null as identity PRIMARY KEY | Уникальный идентификатор типа груза |
| NAME | NVARCHAR2(255) | Not null | Название типа груза |
| DESCRIPTION | NVARCHAR2(255) | Not null | Описание типа груза |

1. **Прочие объекты в СУБД Oracle**

**CargoTypeInfo:**

* Тип: представление.
* Назначение: позволяет быстро получить всю необходимую информацию о грузах и их типах в одном запросе.
* Зависимые объекты: зависит от таблиц CARGO и CARGO\_TYPE.

**VehicleDriverInfo:**

* Тип: представление.
* Назначение: позволяет получить полную информацию о транспортных средствах и их водителях в одном запросе, что удобно для управления автопарком.
* Зависимые объекты: зависит от таблиц DRIVER и DRIVER\_INFO.

**ServiceTypeInfo:**

* Тип: представление.
* Назначение: позволяет легко получить информацию об услугах и их типах, что полезно для анализа и управления услугами.
* Зависимые объекты: зависит от таблиц SERVICES и SERVICE\_TYPE.

**IDX\_Orders\_ClientID:**

* Тип: индекс.
* Назначение: индексирует столбец CLIENT\_ID в таблице ORDERS для повышения производительности при выполнении запросов, которые используют сортировки по этому полю.

**IDX\_Vehicle\_DriverID:**

* Тип: индекс.
* Назначение: индексирует столбец DRIVER\_ID в таблице VEHICLE для повышения производительности при выполнении запросов, которые используют эти поля.

**trg\_vehicle\_status\_update:**

* Тип: триггер.
* Назначение: автоматическое обновление статуса транспортного средства при назначении/освобождении водителя.
* Зависимые объекты: таблицы DRIVER\_INFO, VEHICLE, VEHICLE\_INFO.

**trg\_vehicle\_driver\_update:**

* Тип: триггер.
* Назначение: автоматическая проверка назначен ли водитель.
* Зависимые объекты: таблицы DRIVER\_INFO и VEHICLE, VEHICLE\_INFO.

Скрипт для создания объектов приведен в листинге 6.1.

|  |
| --- |
| CREATE INDEX IDX\_Orders\_ClientID ON ORDERS (CLIENT\_ID);  CREATE INDEX IDX\_Vehicle\_DriverID ON VEHICLE (DRIVER\_ID);  CREATE VIEW ServiceTypeInfo AS  SELECT s.ID AS ServiceID, s.NAME AS ServiceName, st.NAME AS ServiceType, s.BASE\_RATE  FROM SERVICES s JOIN SERVICE\_TYPE st ON s.SERVICE\_TYPE\_ID = st.ID;  CREATE VIEW VehicleDriverInfo AS  SELECT v.ID AS VehicleID, v.LICENSE\_PLATE, v.MODEL, vt.NAME AS VehicleType, di.FIRST\_NAME AS DriverFirstName, di.LAST\_NAME AS DriverLastName FROM VEHICLE v  JOIN VEHICLE\_TYPE vt ON v.VEHICLE\_TYPE = vt.ID  JOIN DRIVER d ON v.DRIVER\_ID = d.ID  JOIN DRIVER\_INFO di ON d.ID = di.DRIVER\_ID;  CREATE VIEW CargoTypeInfo AS  SELECT c.ID AS CargoID, c.NAME AS CargoName, ct.NAME AS CargoType, c.HAZARDOUS FROM CARGO c  JOIN CARGO\_TYPE ct ON c.CARGO\_TYPE = ct.ID;  CREATE OR REPLACE TRIGGER trg\_vehicle\_driver\_update  BEFORE UPDATE OF DRIVER\_ID ON VEHICLE  FOR EACH ROW  BEGIN  IF :NEW.DRIVER\_ID IS NULL THEN  :NEW.STATUS := 'Available';  ELSE  IF :OLD.DRIVER\_ID IS NULL THEN  :NEW.STATUS := 'Busy';  END IF;  END IF;  END;  CREATE OR REPLACE TRIGGER trg\_vehicle\_status\_update  AFTER UPDATE ON DRIVER\_INFO  FOR EACH ROW  BEGIN  IF :NEW.VEHICLE\_INFO IS NOT NULL THEN  UPDATE VEHICLE  SET STATUS = 'Busy'  WHERE ID = :NEW.VEHICLE\_INFO;  END IF;  IF :OLD.VEHICLE\_INFO IS NOT NULL AND :NEW.VEHICLE\_INFO IS NULL THEN  UPDATE VEHICLE  SET STATUS = 'Available',  DRIVER\_ID = NULL  WHERE ID = :OLD.VEHICLE\_INFO;  END IF;  END; |

Листинг 6.1 – Скрипт для создания представлений, индексов и триггеров

1. **Хранимые процедуры в СУБД Oracle**

**Хранимая процедура GET\_ORDERS\_BY\_VEHICLE\_TYPE:**

* Параметры:
* p\_VehicleTypeID IN NUMBER: Идентификатор типа транспортного средства, по которому будет производиться фильтрация заказов.
* Назначение: получение списка заказов, которые были доставлены с использованием определенного типа транспортного средства.
* Зависимости: использует таблицы ORDERS, DRIVER, DRIVER\_INFO и VEHICLE.

**Хранимая процедура UpdateClientInfo:**

* Параметры:
* p\_ClientID: ID клиента для обновления
* p\_FirstName: Новое имя клиента
* p\_LastName: Новая фамилия клиента
* p\_MiddleName: Новое отчество клиента
* p\_Address: Новый адрес клиента
* p\_Email: Новый email клиента
* p\_PhoneNumber: Новый телефонный номер клиента
* Назначение: обновить данные клиента.
* Зависимости: использует таблицу CLIENT\_INFO.

**Хранимая процедура UpdateClientInfo:**

* Параметры:
* p\_ClientID: ID клиента для обновления
* p\_FirstName: Новое имя клиента
* p\_LastName: Новая фамилия клиента
* p\_MiddleName: Новое отчество клиента
* p\_Address: Новый адрес клиента
* p\_Email: Новый email клиента
* p\_PhoneNumber: Новый телефонный номер клиента
* Назначение: обновить данные клиента.
* Зависимости: использует таблицу CLIENT\_INFO.

**Хранимая процедура GetAllClients:**

* Параметры:
* p\_ClientID: ID клиента для обновления
* Назначение: получить информацию о всех клиентах.
* Зависимости: использует таблицу CLIENT\_INFO.

**Хранимая процедура GetAllServices:**

* Параметры отсутствуют.
* Назначение: получить все услуги.
* Зависимости: использует таблицу SERVICES.

Скрипт для создания хранимых процедур приведен в листинге 7.1.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE GetAllClients AS  BEGIN  FOR client\_rec IN (  SELECT ci.FIRST\_NAME, ci.LAST\_NAME, ci.EMAIL, ci.PHONE\_NUMBER  FROM CLIENT\_INFO ci  ) LOOP  DBMS\_OUTPUT.PUT\_LINE('Client Name: ' || client\_rec.FIRST\_NAME || ' ' || client\_rec.LAST\_NAME || ', Email: ' || client\_rec.EMAIL || ', Phone: ' || client\_rec.PHONE\_NUMBER);  END LOOP;  IF SQL%ROWCOUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('No clients found.');  END IF;  END GetAllClients;  CREATE OR REPLACE PROCEDURE GetAllServices AS  BEGIN  FOR service\_rec IN (SELECT \* FROM SERVICES) LOOP  DBMS\_OUTPUT.PUT\_LINE('Service ID: ' || service\_rec.ID || ', Name: ' || service\_rec.NAME || ', Base Rate: ' || service\_rec.BASE\_RATE);  END LOOP;  IF SQL%ROWCOUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('No services found.');  END IF;  END GetAllServices;  CREATE OR REPLACE PROCEDURE UpdateCargoInfo (  p\_CargoID IN NUMBER,  p\_Name IN NVARCHAR2,  p\_Description IN NVARCHAR2,  p\_Weight IN NUMBER,  p\_Volume IN NUMBER,  p\_CargoType IN NUMBER,  p\_Hazardous IN NVARCHAR2  ) AS  BEGIN  UPDATE CARGO  SET NAME = p\_Name,  DESCRIPTION = p\_Description,  WEIGHT = p\_Weight,  VOLUME = p\_Volume,  CARGO\_TYPE = p\_CargoType,  HAZARDOUS = p\_Hazardous  WHERE ID = p\_CargoID;  IF SQL%ROWCOUNT = 0 THEN  RAISE\_APPLICATION\_ERROR(-20003, 'Cargo with ID ' || p\_CargoID || ' not found.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Cargo info updated successfully.');  END IF;  END UpdateCargoInfo;  CREATE OR REPLACE PROCEDURE UpdateClientInfo (  p\_ClientID IN NUMBER,  p\_FirstName IN NVARCHAR2,  p\_LastName IN NVARCHAR2,  p\_MiddleName IN NVARCHAR2,  p\_Address IN NVARCHAR2,  p\_Email IN NVARCHAR2,  p\_PhoneNumber IN NVARCHAR2  ) AS  BEGIN  UPDATE CLIENT\_INFO  SET FIRST\_NAME = p\_FirstName,  LAST\_NAME = p\_LastName,  MIDDLE\_NAME = p\_MiddleName,  ADDRESS = p\_Address,  EMAIL = p\_Email,  PHONE\_NUMBER = p\_PhoneNumber  WHERE CLIENT\_ID = p\_ClientID;  IF SQL%ROWCOUNT = 0 THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Client with ID ' || p\_ClientID || ' not found.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Client info updated successfully.');  END IF;  END UpdateClientInfo;  CREATE OR REPLACE PROCEDURE GET\_ORDERS\_BY\_VEHICLE\_TYPE(  p\_VehicleTypeID IN NUMBER  )  IS  CURSOR orders\_cursor IS  SELECT O.ID AS ORDER\_ID,  O.CLIENT\_ID,  O.DRIVER\_ID,  O.SERVICE\_ID,  O.STATUS,  O.CREATED\_AT,  O.DELIVERED\_AT,  O.TOTAL\_WEIGHT,  O.TOTAL\_VOLUME  FROM ORDERS O  JOIN DRIVER D ON O.DRIVER\_ID = D.ID  JOIN DRIVER\_INFO DI ON D.ID = DI.DRIVER\_ID  JOIN VEHICLE V ON DI.VEHICLE\_INFO = V.ID  WHERE V.VEHICLE\_TYPE = p\_VehicleTypeID  AND O.STATUS = 'Delivered';  v\_row\_count NUMBER := 0;  BEGIN  IF p\_VehicleTypeID < 0 THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Vehicle type ID must be a non-negative number.');  END IF;  IF p\_VehicleTypeID != TRUNC(p\_VehicleTypeID) THEN  RAISE\_APPLICATION\_ERROR(-20003, 'Vehicle type ID must be an integer (non-fractional number).');  END IF;  IF p\_VehicleTypeID < 1 THEN  RAISE\_APPLICATION\_ERROR(-20004, 'Vehicle type ID must be greater than or equal to 1.');  END IF;  DBMS\_OUTPUT.PUT\_LINE('Orders delivered using vehicle type ID ' || p\_VehicleTypeID || ':');  DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------');  FOR order\_rec IN orders\_cursor LOOP  v\_row\_count := v\_row\_count + 1;  DBMS\_OUTPUT.PUT\_LINE(  'Order ID: ' || order\_rec.ORDER\_ID ||  ', Client ID: ' || order\_rec.CLIENT\_ID ||  ', Driver ID: ' || order\_rec.DRIVER\_ID ||  ', Service ID: ' || order\_rec.SERVICE\_ID ||  ', Status: ' || order\_rec.STATUS ||  ', Created At: ' || TO\_CHAR(order\_rec.CREATED\_AT, 'YYYY-MM-DD HH24:MI:SS') ||  ', Delivered At: ' || TO\_CHAR(order\_rec.DELIVERED\_AT, 'YYYY-MM-DD HH24:MI:SS') ||  ', Total Weight: ' || order\_rec.TOTAL\_WEIGHT || ' kg' ||  ', Total Volume: ' || order\_rec.TOTAL\_VOLUME || ' m^3'  );  END LOOP;  IF v\_row\_count = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('No orders found for the specified vehicle type.');  END IF;  EXCEPTION  WHEN OTHERS THEN  RAISE\_APPLICATION\_ERROR(-20002, 'An error occurred: ' || SQLERRM);  END; |

Листинг 7.1 – Скрипт для создания хранимых процедур

1. **Функции в СУБД Oracle**

**Функция COUNT\_DELIVERED\_ORDERS:**

* Параметры:
* startDate IN TIMESTAMP: Начальная дата для фильтрации заказов.
* endDate IN TIMESTAMP: Конечная дата для фильтрации заказов.
* Назначение: подсчет количества доставленных заказов
* Возвращает количество доставленных заказов в виде числа (тип NUMBER). Если возникают проблемы с датами (например, startDate больше endDate, или одна из дат равна NULL, или одна из дат раньше 2000 года), функция возвращает 0 и выводит соответствующее сообщение.
* Зависимости: использует таблицу ORDERS.

**Функция GET\_DRIVER\_INFO:**

* Возвращает строку (VARCHAR2), содержащую информацию о водителе (имя, фамилия и номер телефона). Если возникают проблемы (например, если driver\_id не положительное или водителя не существует), функция возвращает строку '0'.
* Параметры:
* driver\_id IN NUMBER: Идентификатор водителя, информацию о котором необходимо получить.
* Назначение: получение информации о водителе по ID.
* Зависимости: использует таблицу DRIVER\_INFO.

**Функция GetTotalCargo:**

* Функция возвращает строку (VARCHAR2), содержащую информацию о водителе (имя, фамилия и номер телефона). Если возникают проблемы (например, если driver\_id не положительное или водителя не существует), функция возвращает строку '0'.
* Параметры:
* driver\_id IN NUMBER: Идентификатор водителя, информацию о котором необходимо получить.
* Назначение: получение общего кол-ва грузов.
* Зависимости: использует таблицу DRIVER\_INFO.

**Функция GetTotalOrdersByDate:**

* Функция возвращает общее количество заказов за указанный период в виде числа (NUMBER). Если возникают проблемы с датами (например, одна из дат равна NULL или начальная дата позже конечной), функция возвращает 0 и выводит соответствующее сообщение.
* Параметры:
* p\_StartDate IN DATE: Начальная дата для фильтрации заказов.
* p\_EndDate IN DATE: Конечная дата для фильтрации заказов.
* Назначение: получение общего количества заказов за определенный период.
* Зависимости: использует таблицу ORDERS.

**Функция GetActiveDriversCount:**

* Функция возвращает количество активных водителей в виде числа (NUMBER).
* Параметры отсутствуют.
* Назначение: получение количества активных водителей.
* Зависимости: использует таблицы DRIVER, DRIVER\_INFO, VEHICLE.

Скрипт для создания функций приведен в листинге 8.1.

|  |
| --- |
| CREATE OR REPLACE FUNCTION COUNT\_DELIVERED\_ORDERS(startDate IN TIMESTAMP, endDate IN TIMESTAMP) RETURN NUMBER IS  delivered\_count NUMBER;  BEGIN  IF startDate > endDate THEN  dbms\_output.put\_line('Start date is higher then end date');  RETURN 0;  elsif startDate IS NULL OR endDate IS NULL THEN  dbms\_output.put\_line('Start date or end date is null');  RETURN 0;  ELSIF startDate < to\_date('2000-01-01', 'YYYY-MM-DD') OR endDate < to\_date('2000-01-01', 'YYYY-MM-DD') THEN  dbms\_output.put\_line('Date can not be earlier then "2000-01-01"');  RETURN 0;  ELSE  SELECT COUNT(\*) INTO delivered\_count  FROM ORDERS  WHERE STATUS = 'Delivered' AND DELIVERED\_AT BETWEEN startDate AND endDate;  RETURN delivered\_count;  END IF;  END COUNT\_DELIVERED\_ORDERS;  CREATE OR REPLACE FUNCTION GET\_DRIVER\_INFO(driver\_id IN NUMBER) RETURN VARCHAR2 IS  driver\_info VARCHAR2(255);  BEGIN  IF driver\_id <= 0 THEN  dbms\_output.put\_line('Number of driver must be a positive integer');  RETURN '0';  END IF;  DECLARE  driver\_count NUMBER;  BEGIN  SELECT COUNT(\*)  INTO driver\_count  FROM DRIVER\_INFO  WHERE DRIVER\_ID = driver\_id;  IF driver\_count = 0 THEN  RETURN '0';  END IF;  SELECT 'Name: ' || FIRST\_NAME || ' ' || LAST\_NAME || ', Phone: ' || PHONE\_NUMBER  INTO driver\_info  FROM DRIVER\_INFO  WHERE DRIVER\_ID = driver\_id  AND ROWNUM = 1;  RETURN driver\_info;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RETURN '0';  END;  EXCEPTION  WHEN TOO\_MANY\_ROWS THEN  RETURN 'Error: More than one driver found.';  END GET\_DRIVER\_INFO;  CREATE OR REPLACE FUNCTION GetTotalOrdersByDate(  p\_StartDate IN DATE,  p\_EndDate IN DATE  ) RETURN NUMBER AS  v\_TotalOrders NUMBER;  BEGIN  IF p\_StartDate IS NULL OR p\_EndDate IS NULL THEN  DBMS\_OUTPUT.PUT\_LINE('Start date or end date is null');  RETURN 0;  ELSIF p\_StartDate > p\_EndDate THEN  DBMS\_OUTPUT.PUT\_LINE('Start date is later than end date');  RETURN 0;  ELSE  SELECT COUNT(\*) INTO v\_TotalOrders  FROM ORDERS  WHERE CREATED\_AT BETWEEN p\_StartDate AND p\_EndDate;  RETURN v\_TotalOrders;  END IF;  END;  CREATE OR REPLACE FUNCTION GetActiveDriversCount RETURN NUMBER IS  v\_ActiveDriversCount NUMBER;  BEGIN  SELECT COUNT(\*) INTO v\_ActiveDriversCount  FROM DRIVER D  JOIN DRIVER\_INFO DI ON D.ID = DI.DRIVER\_ID  WHERE EXISTS (SELECT 1 FROM VEHICLE V WHERE V.DRIVER\_ID = D.ID AND V.STATUS = 'Available');    RETURN v\_ActiveDriversCount;  END GetActiveDriversCount; |

Листинг 8.1 – Скрипт для создания функций