Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Проектирование и разработка баз данных интернет-приложений

Студент: Трубач Д.С.

ФИТ 3 курс 5 группа

Преподаватель: Нистюк О.А.

Минск 2024

# **Лабораторная работа № 2**

MSSQL

Список таблиц с указанием атрибутов:

Таблица Пользователь:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **Ограничения** | **Назначение** |
| ID | INT | PRIMARY KEY, IDENTITY(1,1) | Уникальный идентификатор пользователя |
| Имя | NVARCHAR(100) | NOT NULL | Имя пользователя |
| Фамилия | NVARCHAR(100) | NOT NULL | Фамилия пользователя |
| email | NVARCHAR(100) | NOT NULL | Электронная почта пользователя |
| Пароль | NVARCHAR(100) | NOT NULL | Пароль пользователя |
| Роль | NVARCHAR(20) | CHECK (Роль IN (‘Студент’, ‘Преподаватель’, ‘Администратор’)), NOT NULL | Роль пользователя |
| Данные | NVARCHAR(100) |  | Дополнительные данные пользователя |

Таблица Тест:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **Ограничения** | **Назначение** |
| ID | INT | PRIMARY KEY | Уникальный идентификатор теста |
| Название | NVARCHAR(100) | NOT NULL | Название теста |
| Описание | NVARCHAR(255) | NOT NULL | Описание теста |
| Предмет | NVARCHAR(100) | NOT NULL | Предмет теста |
| ID\_пользователя | INT | FOREIGN KEY REFERENCES Пользователь(ID), NOT NULL | Ссылка на пользователя, создавшего тест |

Таблица Вопрос:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **Ограничения** | **Назначение** |
| ID | INT | PRIMARY KEY | Уникальный идентификатор вопроса |
| Текст | NVARCHAR(255) | NOT NULL | Текст вопроса |
| ID\_теста | INT | FOREIGN KEY REFERENCES Тест(ID), NOT NULL | Ссылка на тест, которому принадлежит вопрос |

Таблица Ответ:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **Ограничения** | **Назначение** |
| ID | INT | PRIMARY KEY | Уникальный идентификатор ответа |
| Текст | NVARCHAR(255) | NOT NULL | Текст ответа |
| Правильный | BIT | NOT NULL | Флаг, указывающий, является ли ответ правильным |
| ID\_вопроса | INT | FOREIGN KEY REFERENCES Вопрос(ID), NOT NULL | Ссылка на вопрос, которому принадлежит ответ |

Таблица Результат\_теста:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **Ограничения** | **Назначение** |
| ID | INT | PRIMARY KEY, IDENTITY(1,1) | Уникальный идентификатор результата теста |
| ID\_пользователя | INT | FOREIGN KEY REFERENCES Пользователь(ID), NOT NULL | Ссылка на пользователя, проходившего тест |
| ID\_теста | INT | FOREIGN KEY REFERENCES Тест(ID), NOT NULL | Ссылка на пройденный тест |
| Дата\_и\_время | DATETIME | NOT NULL | Дата и время прохождения теста |
| Итоговый\_балл | FLOAT | NOT NULL | Итоговый балл за тест |

Список прочих объектов:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип объекта** | **Назначение** | **Зависимые объекты** |
| IX\_Пользователь\_Имя\_Фамилия | Индекс | Ускоряет поиск по имени и фамилии пользователя | Таблица Пользователь |
| IX\_Тест\_Предмет | Индекс | Ускоряет поиск тестов по предмету | Таблица Тест |
| IX\_Вопрос\_ID\_теста | Индекс | Ускоряет поиск вопросов по ID теста | Таблица Вопрос |
| IX\_Ответ\_ID\_вопроса | Индекс | Ускоряет поиск ответов по ID вопроса | Таблица Ответ |
| IX\_Результат\_теста\_ID\_пользователя | Индекс | Ускоряет поиск результатов теста по ID пользователя | Таблица Результат\_теста |
| Регистрация | Процедура | Регистрация нового пользователя | Таблица Пользователь |
| Авторизация | Процедура | Авторизация пользователя | Таблица Пользователь |
| Проверка\_Количества\_Вопросов | Триггер | Проверка количества вопросов в тесте при добавлении или изменении вопроса | Таблицы Тест, Вопрос |
| Добавить\_тест | Процедура | Добавление нового теста | Таблицы Пользователь, Тест |
| Редактировать\_тест | Процедура | Редактирование существующего теста | Таблицы Пользователь, Тест |
| Удалить\_тест | Процедура | Удаление существующего теста | Таблицы Пользователь, Тест |
| Добавить\_Вопрос | Процедура | Добавление нового вопроса в тест | Таблицы Пользователь, Тест, Вопрос |
| Добавить\_Ответ | Процедура | Добавление нового ответа на вопрос | Таблицы Пользователь, Тест, Вопрос, Ответ |
| ПроверитьОтветы | Процедура | Проверка ответов пользователя | Таблицы Пользователь, Тест, Вопрос, Ответ, Результат\_теста |
| ИзменитьДанныеПользователя | Процедура | Изменение данных пользователя | Таблица Пользователь |
| ИзменитьРольПользователя | Процедура | Изменение роли пользователя | Таблица Пользователь |
| Все\_тесты | Представление | Представление всех тестов | Таблица Тест |
| Информация\_О\_Тесте | Функция | Получение информации о тесте | Таблица Тест |
| Вопросы\_Теста | Функция | Получение вопросов теста | Таблица Вопрос |
| ВсеРезультаты | Функция | Получение всех результатов тестов | Таблица Результат\_теста |
| РезультатыПользователя | Функция | Получение результатов тестов конкретного пользователя | Таблицы Пользователь, Результат\_теста |
| РезультатыТеста | Функция | Получение результатов конкретного теста | Таблицы Тест, Результат\_теста |

ORACLE

Список таблиц с указанием атрибутов:

Таблица Пользователь:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **Ограничения** | **Назначение** |
| ID | NUMBER | PRIMARY KEY | Уникальный идентификатор пользователя |
| Имя | NVARCHAR2(100) | NOT NULL | Имя пользователя |
| Фамилия | NVARCHAR2(100) | NOT NULL | Фамилия пользователя |
| email | NVARCHAR2(100) | NOT NULL | Электронная почта пользователя |
| Пароль | NVARCHAR2(100) | NOT NULL | Пароль пользователя |
| Роль | NVARCHAR2(20) | CHECK (Роль IN (‘Студент’, ‘Преподаватель’, ‘Администратор’)), NOT NULL | Роль пользователя |
| Данные | NVARCHAR2(100) |  | Дополнительные данные пользователя |

Таблица Тест:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **Ограничения** | **Назначение** |
| ID | NUMBER | PRIMARY KEY | Уникальный идентификатор теста |
| Название | NVARCHAR2(100) | NOT NULL | Название теста |
| Описание | NVARCHAR2(255) | NOT NULL | Описание теста |
| Предмет | NVARCHAR2(100) | NOT NULL | Предмет теста |
| ID\_пользователя | NUMBER | FOREIGN KEY REFERENCES Пользователь(ID), NOT NULL | Ссылка на пользователя, создавшего тест |

Таблица Вопрос:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **Ограничения** | **Назначение** |
| ID | NUMBER | PRIMARY KEY | Уникальный идентификатор вопроса |
| Текст | NVARCHAR2(255) | NOT NULL | Текст вопроса |
| ID\_теста | NUMBER | FOREIGN KEY REFERENCES Тест(ID), NOT NULL | Ссылка на тест, которому принадлежит вопрос |

Таблица Ответ:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **Ограничения** | **Назначение** |
| ID | NUMBER | PRIMARY KEY | Уникальный идентификатор ответа |
| Текст | NVARCHAR(255) | NOT NULL | Текст ответа |
| Правильный | BIT | NOT NULL | Флаг, указывающий, является ли ответ правильным |
| ID\_вопроса | NUMBER | FOREIGN KEY REFERENCES Вопрос(ID), NOT NULL | Ссылка на вопрос, которому принадлежит ответ |

Таблица Результат\_теста:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **Ограничения** | **Назначение** |
| ID | NUMBER | PRIMARY KEY | Уникальный идентификатор результата теста |
| ID\_пользователя | NUMBER | FOREIGN KEY REFERENCES Пользователь(ID), NOT NULL | Ссылка на пользователя, проходившего тест |
| ID\_теста | NUMBER | FOREIGN KEY REFERENCES Тест(ID), NOT NULL | Ссылка на пройденный тест |
| Дата\_и\_время | DATETIME | NOT NULL | Дата и время прохождения теста |
| Итоговый\_балл | FLOAT | NOT NULL | Итоговый балл за тест |

Список прочих объектов:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип объекта** | **Назначение** | **Зависимые объекты** |
| seq\_Пользователь | Последовательность | Генерация уникальных идентификаторов для таблицы Пользователь | Таблица Пользователь |
| seq\_Тест | Последовательность | Генерация уникальных идентификаторов для таблицы Тест | Таблица Тест |
| seq\_Вопрос | Последовательность | Генерация уникальных идентификаторов для таблицы Вопрос | Таблица Вопрос |
| seq\_Ответ | Последовательность | Генерация уникальных идентификаторов для таблицы Ответ | Таблица Ответ |
| seq\_Результат\_теста | Последовательность | Генерация уникальных идентификаторов для таблицы Результат\_теста | Таблица Результат\_теста |
| IX\_Пользователь\_Имя\_Фамилия | Индекс | Ускоряет поиск по имени и фамилии пользователя | Таблица Пользователь |
| IX\_Тест\_Предмет | Индекс | Ускоряет поиск тестов по предмету | Таблица Тест |
| IX\_Вопрос\_ID\_теста | Индекс | Ускоряет поиск вопросов по ID теста | Таблица Вопрос |
| IX\_Ответ\_ID\_вопроса | Индекс | Ускоряет поиск ответов по ID вопроса | Таблица Ответ |
| IX\_Результат\_теста\_ID\_пользователя | Индекс | Ускоряет поиск результатов теста по ID пользователя | Таблица Результат\_теста |
| Регистрация | Процедура | Регистрация нового пользователя | Таблица Пользователь |
| Авторизация | Процедура | Авторизация пользователя | Таблица Пользователь |
| trg\_Пользователь | Триггер | Автоматическая генерация уникального идентификатора для нового пользователя | Таблица Пользователь, Последовательность seq\_Пользователь |
| trg\_Тест | Триггер | Автоматическая генерация уникального идентификатора для нового теста | Таблица Тест, Последовательность seq\_Тест |
| trg\_Вопрос | Триггер | Автоматическая генерация уникального идентификатора для нового вопроса | Таблица Вопрос, Последовательность seq\_Вопрос |
| trg\_Ответ | Триггер | Автоматическая генерация уникального идентификатора для нового ответа | Таблица Ответ, Последовательность seq\_Ответ |
| trg\_Результат\_теста | Триггер | Автоматическая генерация уникального идентификатора для нового результата теста | Таблица Результат\_теста, Последовательность seq\_Результат\_теста |
| Проверка\_Количества\_Вопросов | Триггер | Проверка количества вопросов в тесте при добавлении или изменении вопроса | Таблицы Тест, Вопрос |
| Добавить\_тест | Процедура | Добавление нового теста | Таблицы Пользователь, Тест |
| Редактировать\_тест | Процедура | Редактирование существующего теста | Таблицы Пользователь, Тест |
| Удалить\_тест | Процедура | Удаление существующего теста | Таблицы Пользователь, Тест |
| Добавить\_Вопрос | Процедура | Добавление нового вопроса в тест | Таблицы Пользователь, Тест, Вопрос |
| Добавить\_Ответ | Процедура | Добавление нового ответа на вопрос | Таблицы Пользователь, Тест, Вопрос, Ответ |
| ПроверитьОтветы | Процедура | Проверка ответов пользователя | Таблицы Пользователь, Тест, Вопрос, Ответ, Результат\_теста |
| ИзменитьДанныеПользователя | Процедура | Изменение данных пользователя | Таблица Пользователь |
| ИзменитьРольПользователя | Процедура | Изменение роли пользователя | Таблица Пользователь |
| Все\_тесты | Представление | Представление всех тестов | Таблица Тест |
| Информация\_О\_Тесте | Функция | Получение информации о тесте | Таблица Тест |
| Вопросы\_Теста | Функция | Получение вопросов теста | Таблица Вопрос |
| ВсеРезультаты | Функция | Получение всех результатов тестов | Таблица Результат\_теста |
| РезультатыПользователя | Функция | Получение результатов тестов конкретного пользователя | Таблицы Пользователь, Результат\_теста |
| РезультатыТеста | Функция | Получение результатов конкретного теста | Таблицы Тест, Результат\_теста |

|  |
| --- |
| CREATE DATABASE TESTING;  USE TESTING;  --USE master;  --DROP DATABASE TESTING;  --drop table Пользователь;  --drop table Тест;  --drop table Вопрос;  --drop table Ответ;  --drop table Результат\_теста;  CREATE TABLE Пользователь (  ID INT IDENTITY(1,1) PRIMARY KEY,  Имя NVARCHAR(100) NOT NULL,  Фамилия NVARCHAR(100) NOT NULL,  email NVARCHAR(100) NOT NULL,  Пароль NVARCHAR(100) NOT NULL,  Роль NVARCHAR(20) CHECK (Роль IN ('Студент', 'Преподаватель', 'Администратор')) NOT NULL,  Данные NVARCHAR(100)  );  CREATE TABLE Тест (  ID INT PRIMARY KEY,  Название NVARCHAR(100) NOT NULL,  Описание NVARCHAR(255) NOT NULL,  Предмет NVARCHAR(100) NOT NULL,  ID\_пользователя INT FOREIGN KEY REFERENCES Пользователь(ID) NOT NULL  );  CREATE TABLE Вопрос (  ID INT PRIMARY KEY,  Текст NVARCHAR(255) NOT NULL,  ID\_теста INT FOREIGN KEY REFERENCES Тест(ID) NOT NULL  );  CREATE TABLE Ответ (  ID INT PRIMARY KEY,  Текст NVARCHAR(255) NOT NULL,  Правильный BIT NOT NULL,  ID\_вопроса INT FOREIGN KEY REFERENCES Вопрос(ID) NOT NULL  );  CREATE TABLE Результат\_теста (  ID INT IDENTITY(1,1) PRIMARY KEY,  ID\_пользователя INT FOREIGN KEY REFERENCES Пользователь(ID) NOT NULL,  ID\_теста INT FOREIGN KEY REFERENCES Тест(ID) NOT NULL,  Дата\_и\_время DATETIME NOT NULL,  Итоговый\_балл FLOAT NOT NULL  );  CREATE INDEX IX\_Пользователь\_Имя\_Фамилия ON Пользователь (Имя, Фамилия);  CREATE INDEX IX\_Тест\_Предмет ON Тест (Предмет);  CREATE INDEX IX\_Вопрос\_ID\_теста ON Вопрос (ID\_теста);  CREATE INDEX IX\_Ответ\_ID\_вопроса ON Ответ (ID\_вопроса);  CREATE INDEX IX\_Результат\_теста\_ID\_пользователя ON Результат\_теста (ID\_пользователя);  SELECT OBJECT\_NAME(i.object\_id) AS table\_name, i.name AS index\_name FROM sys.indexes i WHERE i.name LIKE 'IX\_%';  -- ============================== Регистрация ==============================  CREATE OR ALTER PROCEDURE Регистрация  @Имя NVARCHAR(100),  @Фамилия NVARCHAR(100),  @Email NVARCHAR(100),  @Пароль NVARCHAR(100),  @Роль NVARCHAR(20),  @Данные NVARCHAR(100)  AS  BEGIN  IF NOT EXISTS (SELECT 1 FROM Пользователь WHERE Email = @Email)  BEGIN  IF @Email LIKE '%@%.%' AND @Email NOT LIKE '%@%@%'  BEGIN  IF @Роль IN ('Студент', 'Преподаватель', 'Администратор')  BEGIN  -- Проверка, что пользователь с таким email уже не существует  DECLARE @ПользовательСуществует BIT;  SET @ПользовательСуществует = 0;  SELECT @ПользовательСуществует = 1 FROM Пользователь WHERE Email = @Email;  IF @ПользовательСуществует = 0  BEGIN  INSERT INTO Пользователь (Имя, Фамилия, Email, Пароль, Роль, Данные)  VALUES (@Имя, @Фамилия, @Email, HASHBYTES('SHA2\_256', @Пароль), @Роль, @Данные);  SELECT 1 AS Result, 'Регистрация прошла успешно.' AS Message;  END  ELSE  BEGIN  SELECT 0 AS Result, 'Пользователь с таким email уже существует.' AS Message;  END;  END  ELSE  BEGIN  SELECT 0 AS Result, 'Недопустимая роль. Роль должна быть либо Студент, либо Преподаватель, либо Администратор.' AS Message;  END;  END  ELSE  BEGIN  SELECT 0 AS Result, 'Неправильный формат email.' AS Message;  END;  END  ELSE  BEGIN  SELECT 0 AS Result, 'Пользователь с таким email уже существует.' AS Message;  END;  END;  GO  -- ============================== Авторизация ==============================  CREATE OR ALTER PROCEDURE Авторизация  @Email NVARCHAR(100),  @Пароль NVARCHAR(100)  AS  BEGIN  IF EXISTS (SELECT 1 FROM Пользователь WHERE Email = @Email AND Пароль = HASHBYTES('SHA2\_256', @Пароль))  BEGIN  SELECT 1 AS Result, 'Авторизация прошла успешно.' AS Message;  END  ELSE  BEGIN  SELECT 0 AS Result, 'Неверный email или пароль.' AS Message;  END  END;  GO  -- ============================== Провека регистрации и авторизации ==============================  EXEC Регистрация @Имя = 'Dmitry', @Фамилия = 'Trubach', @Email = 'dima1@mail.ru', @Пароль = 'dima123', @Роль = 'Студент', @Данные = 'Тест 1';  EXEC Регистрация @Имя = 'Prepod', @Фамилия = 'Prepod', @Email = 'dima2@mail.ru', @Пароль = 'dima1234', @Роль = 'Преподаватель', @Данные = 'Тест 2';  EXEC Регистрация @Имя = 'Тестовый', @Фамилия = 'Препод', @Email = 'dima3@mail.ru', @Пароль = 'dima12345', @Роль = 'Преподаватель', @Данные = 'Тест 3';  EXEC Регистрация @Имя = 'admin', @Фамилия = 'admin', @Email = 'dima4@mail.ru', @Пароль = 'dima123456', @Роль = 'Администратор', @Данные = 'Тест 3';  SELECT \* FROM Пользователь;  EXEC Авторизация @Email = 'dima1@mail.ru', @Пароль = 'dima123';  -- ============================== Добавление, редактирование и удаление теста ==============================  -- ============================== Проверка количества вопросов (<=5) ==============================  CREATE OR ALTER TRIGGER Проверка\_Количества\_Вопросов  ON Вопрос  AFTER INSERT, UPDATE  AS  BEGIN  IF EXISTS (  SELECT T.ID, COUNT(\*) AS Количество\_Вопросов  FROM Тест T  JOIN Вопрос V ON T.ID = V.ID\_теста  WHERE T.ID IN (SELECT ID\_теста FROM inserted)  GROUP BY T.ID  HAVING COUNT(\*) > 5  )  BEGIN  RAISERROR ('Максимальное количество вопросов в тесте составляет 5.', 16, 1);  ROLLBACK TRANSACTION;  END;  END;  GO  -- ============================== Добавить тест ==============================  CREATE OR ALTER PROCEDURE Добавить\_тест  @ID INT,  @Название NVARCHAR(100),  @Описание NVARCHAR(255),  @Предмет NVARCHAR(100),  @ID\_пользователя INT  AS  BEGIN  IF EXISTS (SELECT 1 FROM Пользователь WHERE ID = @ID\_пользователя AND Роль = 'Преподаватель')  BEGIN  INSERT INTO Тест (ID, Название, Описание, Предмет, ID\_пользователя)  VALUES (@ID, @Название, @Описание, @Предмет, @ID\_пользователя);  SELECT 1 AS Result, 'Тест успешно добавлен.' AS Message;  END  ELSE  BEGIN  SELECT 0 AS Result, 'Недостаточно прав для добавления теста.' AS Message;  END  END;  GO  -- ============================== Редактировать тест ==============================  CREATE OR ALTER PROCEDURE Редактировать\_тест  @ID INT,  @Название NVARCHAR(100),  @Описание NVARCHAR(255),  @Предмет NVARCHAR(100),  @ID\_пользователя INT  AS  BEGIN  IF EXISTS (SELECT 1 FROM Пользователь WHERE ID = @ID\_пользователя AND Роль = 'Преподаватель')  BEGIN  UPDATE Тест  SET Название = @Название, Описание = @Описание, Предмет = @Предмет  WHERE ID = @ID;  SELECT 1 AS Result, 'Тест успешно отредактирован.' AS Message;  END  ELSE  BEGIN  SELECT 0 AS Result, 'Недостаточно прав для редактирования теста.' AS Message;  END  END;  GO  -- ============================== Удалить тест ==============================  CREATE OR ALTER PROCEDURE Удалить\_тест  @ID INT,  @ID\_пользователя INT  AS  BEGIN  IF EXISTS (SELECT 1 FROM Пользователь WHERE ID = @ID\_пользователя AND Роль = 'Преподаватель')  BEGIN  DELETE FROM Тест  WHERE ID = @ID;  SELECT 1 AS Result, 'Тест успешно удален.' AS Message;  END  ELSE  BEGIN  SELECT 0 AS Result, 'Недостаточно прав для удаления теста.' AS Message;  END  END;  GO  -- ============================== Просмотр всех тестов ==============================  CREATE VIEW Все\_тесты  AS  SELECT ID, Название, Описание, Предмет  FROM Тест;  GO  select \* from Тест; -- для просмотра ID  SELECT \* FROM Все\_тесты;  -- Добавление теста  EXEC Добавить\_тест @ID = 1, @Название = 'Тест №1', @Описание = 'Тесттест', @Предмет = 'Математика', @ID\_пользователя = 2;  -- Редактирование теста  EXEC Редактировать\_тест @ID = 1, @Название = 'Измененный тест 1', @Описание = 'Измененное описание теста 1', @Предмет = 'Информатика', @ID\_пользователя = 1;  -- Удаление теста  EXEC Удалить\_тест @ID = 1, @ID\_пользователя = 1;  -- ============================== Добавление вопроса ==============================  CREATE OR ALTER PROCEDURE Добавить\_Вопрос  @ID INT,  @Текст NVARCHAR(100),  @ID\_теста INT,  @ID\_пользователя INT  AS  BEGIN  IF EXISTS (SELECT 1 FROM Пользователь WHERE ID = @ID\_пользователя AND Роль = 'Преподаватель')  BEGIN  IF EXISTS (SELECT 1 FROM Тест WHERE ID = @ID\_теста AND ID\_пользователя = @ID\_пользователя)  BEGIN  INSERT INTO Вопрос (ID, Текст, ID\_теста)  VALUES (@ID, @Текст, @ID\_теста);    SELECT 1 AS Result, 'Вопрос успешно добавлен.' AS Message;  END  ELSE  BEGIN  SELECT 0 AS Result, 'Тест не принадлежит данному пользователю.' AS Message;  END  END  ELSE  BEGIN  SELECT 0 AS Result, 'Недопустимая роль. Только преподаватель может добавлять вопросы.' AS Message;  END  END;  GO  -- ============================== Добавление ответа ==============================  CREATE OR ALTER PROCEDURE Добавить\_Ответ  @ID INT,  @Текст NVARCHAR(100),  @Правильный BIT,  @ID\_вопроса INT,  @ID\_пользователя INT  AS  BEGIN  IF EXISTS (SELECT 1 FROM Пользователь WHERE ID = @ID\_пользователя AND Роль = 'Преподаватель')  BEGIN  IF EXISTS (SELECT 1 FROM Вопрос INNER JOIN Тест ON Вопрос.ID\_теста = Тест.ID WHERE Вопрос.ID = @ID\_вопроса AND Тест.ID\_пользователя = @ID\_пользователя)  BEGIN  INSERT INTO Ответ (ID, Текст, Правильный, ID\_вопроса)  VALUES (@ID, @Текст, @Правильный, @ID\_вопроса);    SELECT 1 AS Result, 'Ответ успешно добавлен.' AS Message;  END  ELSE  BEGIN  SELECT 0 AS Result, 'Вопрос не принадлежит тесту данного пользователя.' AS Message;  END  END  ELSE  BEGIN  SELECT 0 AS Result, 'Недопустимая роль. Только преподаватель может добавлять ответы.' AS Message;  END  END;  GO  EXEC Добавить\_Вопрос  @ID = 1,  @Текст = 'Вопрос 1, ответ да',  @ID\_теста = 1,  @ID\_пользователя = 2;  EXEC Добавить\_Ответ  @ID = 1,  @Текст = 'Да',  @Правильный = 1,  @ID\_вопроса = 1,  @ID\_пользователя = 2;  EXEC Добавить\_Ответ  @ID = 2,  @Текст = 'Нет',  @Правильный = 0,  @ID\_вопроса = 1,  @ID\_пользователя = 2;  EXEC Добавить\_Вопрос  @ID = 2,  @Текст = 'Вопрос 2, ответ да',  @ID\_теста = 1,  @ID\_пользователя = 2;  EXEC Добавить\_Ответ  @ID = 3,  @Текст = 'Да',  @Правильный = 1,  @ID\_вопроса = 2,  @ID\_пользователя = 2;  EXEC Добавить\_Ответ  @ID = 4,  @Текст = 'Нет',  @Правильный = 0,  @ID\_вопроса = 2,  @ID\_пользователя = 2;  EXEC Добавить\_Вопрос  @ID = 3,  @Текст = 'Вопрос 3, ответ нет',  @ID\_теста = 1,  @ID\_пользователя = 2;  EXEC Добавить\_Ответ  @ID = 5,  @Текст = 'Да',  @Правильный = 0,  @ID\_вопроса = 3,  @ID\_пользователя = 2;  EXEC Добавить\_Ответ  @ID = 6,  @Текст = 'Нет',  @Правильный = 1,  @ID\_вопроса = 3,  @ID\_пользователя = 2;  EXEC Добавить\_Вопрос  @ID = 4,  @Текст = 'Вопрос 4, ответ нет',  @ID\_теста = 1,  @ID\_пользователя = 2;  EXEC Добавить\_Ответ  @ID = 7,  @Текст = 'Да',  @Правильный = 0,  @ID\_вопроса = 4,  @ID\_пользователя = 2;  EXEC Добавить\_Ответ  @ID = 8,  @Текст = 'Нет',  @Правильный = 1,  @ID\_вопроса = 4,  @ID\_пользователя = 2;  EXEC Добавить\_Вопрос  @ID = 5,  @Текст = 'Вопрос 5, ответ нет',  @ID\_теста = 1,  @ID\_пользователя = 2;  EXEC Добавить\_Ответ  @ID = 9,  @Текст = 'Да',  @Правильный = 0,  @ID\_вопроса = 5,  @ID\_пользователя = 2;  EXEC Добавить\_Ответ  @ID = 10,  @Текст = 'Нет',  @Правильный = 1,  @ID\_вопроса = 5,  @ID\_пользователя = 2;  EXEC Добавить\_Вопрос  @ID = 6,  @Текст = 'Вопрос 6, ответ нет',  @ID\_теста = 1,  @ID\_пользователя = 2;  EXEC Добавить\_Ответ  @ID = 11,  @Текст = 'Да',  @Правильный = 0,  @ID\_вопроса = 6,  @ID\_пользователя = 2;  EXEC Добавить\_Ответ  @ID = 12,  @Текст = 'Нет',  @Правильный = 1,  @ID\_вопроса = 6,  @ID\_пользователя = 2;  -- ============================== Вывод информации о тесте ==============================  CREATE OR ALTER FUNCTION Информация\_О\_Тесте(@ID\_теста INT)  RETURNS TABLE  AS  RETURN  SELECT Название, Описание  FROM Тест  WHERE ID = @ID\_теста;  GO  -- ============================== Вывод вопросов теста ==============================  CREATE OR ALTER FUNCTION Вопросы\_Теста(@ID\_теста INT)  RETURNS TABLE  AS  RETURN  SELECT ID, Текст AS Вопрос  FROM Вопрос  WHERE ID\_теста = @ID\_теста;  GO  SELECT \* FROM Информация\_О\_Тесте(1);  SELECT \* FROM Вопросы\_Теста(1);  GO  -- ============================== Проверка ответов ==============================  CREATE OR ALTER PROCEDURE ПроверитьОтветы (  @ID\_теста INT,  @ID\_пользователя INT,  @Ответы NVARCHAR(MAX)  )  AS  BEGIN  -- Создаем таблицу для хранения ответов  CREATE TABLE #Ответы (ID\_вопроса INT, Ответ NVARCHAR(3));  -- Разделяем строку ответов на отдельные ответы  DECLARE @ID\_вопроса INT = 1;  DECLARE @Ответ NVARCHAR(3);  WHILE LEN(@Ответы) > 0  BEGIN  IF CHARINDEX(',', @Ответы) > 0  BEGIN  SET @Ответ = SUBSTRING(@Ответы, 1, CHARINDEX(',', @Ответы) - 1);  SET @Ответы = SUBSTRING(@Ответы, CHARINDEX(',', @Ответы) + 1, LEN(@Ответы));  END  ELSE  BEGIN  SET @Ответ = @Ответы;  SET @Ответы = '';  END  INSERT INTO #Ответы VALUES (@ID\_вопроса, @Ответ);  SET @ID\_вопроса = @ID\_вопроса + 1;  END  -- Проверяем, что количество ответов соответствует количеству вопросов в тесте  DECLARE @КоличествоВопросов INT;  SELECT @КоличествоВопросов = COUNT(\*) FROM Вопрос WHERE ID\_теста = @ID\_теста;  IF @КоличествоВопросов <> @ID\_вопроса - 1  BEGIN  PRINT 'Ошибка: количество ответов не соответствует количеству вопросов в тесте.';  RETURN;  END  -- Сравниваем ответы с правильными ответами  SELECT  Вопрос.ID AS ID\_вопроса,  Вопрос.Текст AS Текст\_вопроса,  Ответ.Текст AS Текст\_ответа,  CASE  WHEN Ответ.Правильный = 1 THEN 'Правильно'  ELSE 'Неправильно'  END AS Результат  INTO #Результаты  FROM Вопрос  INNER JOIN Ответ ON Вопрос.ID = Ответ.ID\_вопроса  INNER JOIN #Ответы ON Вопрос.ID = #Ответы.ID\_вопроса  WHERE Вопрос.ID\_теста = @ID\_теста  AND Ответ.Правильный = CASE  WHEN #Ответы.Ответ = 'Да' THEN 1  WHEN #Ответы.Ответ = 'Нет' THEN 0  ELSE NULL  END;  -- Вычисляем оценку  DECLARE @ПравильныеОтветы INT;  DECLARE @Оценка FLOAT;  SELECT  @ПравильныеОтветы = COUNT(\*) FROM #Результаты WHERE Результат = 'Правильно';  IF @ПравильныеОтветы = @КоличествоВопросов  SET @Оценка = 10.0;  ELSE IF @ПравильныеОтветы = 0  SET @Оценка = 0.0;  ELSE  SET @Оценка = 10.0 - ((1.0 / @КоличествоВопросов) \* 10.0 \* (@КоличествоВопросов - @ПравильныеОтветы));  -- Выводим результаты и оценку  SELECT \* FROM #Результаты;  PRINT 'Оценка: ' + CAST(@Оценка AS VARCHAR);  -- Добавляем результаты в таблицу Результат\_теста  INSERT INTO Результат\_теста (ID\_пользователя, ID\_теста, Дата\_и\_время, Итоговый\_балл)  VALUES (@ID\_пользователя, @ID\_теста, GETDATE(), @Оценка);  -- Удаляем временные таблицы  DROP TABLE #Ответы;  DROP TABLE #Результаты;  END;  GO  EXEC ПроверитьОтветы @ID\_теста = 1, @ID\_пользователя = 1, @Ответы = 'Да,Нет,Да,Да,Да';  select \* from Результат\_теста;  CREATE OR ALTER FUNCTION ВсеРезультаты()  RETURNS TABLE  AS  RETURN  SELECT ID, ID\_пользователя, ID\_теста, Дата\_и\_время, Итоговый\_балл  FROM Результат\_теста;  GO  CREATE OR ALTER FUNCTION РезультатыПользователя(@ID\_пользователя INT)  RETURNS TABLE  AS  RETURN  SELECT ID, ID\_пользователя, ID\_теста, Дата\_и\_время, Итоговый\_балл  FROM Результат\_теста  WHERE ID\_пользователя = @ID\_пользователя;  GO  CREATE OR ALTER FUNCTION РезультатыТеста(@ID\_теста INT)  RETURNS TABLE  AS  RETURN  SELECT ID, ID\_пользователя, ID\_теста, Дата\_и\_время, Итоговый\_балл  FROM Результат\_теста  WHERE ID\_теста = @ID\_теста;  GO  SELECT \* FROM ВсеРезультаты();  SELECT \* FROM РезультатыПользователя(1);  SELECT \* FROM РезультатыТеста(1);  -- ============================== Изменение данных о пользователе ==============================  CREATE OR ALTER PROCEDURE ИзменитьДанныеПользователя  @ID\_пользователя INT,  @ID\_администратора INT,  @НовоеИмя NVARCHAR(100),  @НоваяФамилия NVARCHAR(100),  @НовыйEmail NVARCHAR(100),  @НовыйПароль NVARCHAR(100),  @НовыеДанные NVARCHAR(100)  AS  BEGIN  IF EXISTS (SELECT 1 FROM Пользователь WHERE ID = @ID\_администратора AND Роль = 'Администратор')  BEGIN  UPDATE Пользователь  SET Имя = @НовоеИмя,  Фамилия = @НоваяФамилия,  email = @НовыйEmail,  Пароль = HASHBYTES('SHA2\_256', @НовыйПароль),  Данные = @НовыеДанные  WHERE ID = @ID\_пользователя;  SELECT 1 AS Result, 'Данные пользователя успешно обновлены.' AS Message;  END  ELSE  BEGIN  SELECT 0 AS Result, 'Ошибка: только администратор может изменять данные пользователя.' AS Message;  END  END;  GO  EXEC ИзменитьДанныеПользователя @ID\_пользователя = 1, @ID\_администратора = 4, @НовоеИмя = 'Dmitry', @НоваяФамилия = 'Trubach',  @НовыйEmail = 'dima1@mail.ru', @НовыйПароль = 'dima123', @НовыеДанные = 'Тест 1';  SELECT \* FROM Пользователь;  -- ============================== Изменение роли ==============================  CREATE OR ALTER PROCEDURE ИзменитьРольПользователя  @ID\_пользователя INT,  @ID\_администратора INT,  @НоваяРоль NVARCHAR(20)  AS  BEGIN  -- Проверяем, является ли пользователь с @ID\_администратора администратором  IF EXISTS (SELECT 1 FROM Пользователь WHERE ID = @ID\_администратора AND Роль = 'Администратор')  BEGIN  -- Если пользователь является администратором, обновляем роль пользователя  UPDATE Пользователь  SET Роль = @НоваяРоль  WHERE ID = @ID\_пользователя;  SELECT 1 AS Result, 'Роль пользователя успешно обновлена.' AS Message;  END  ELSE  BEGIN  -- Если пользователь не является администратором, возвращаем сообщение об ошибке  SELECT 0 AS Result, 'Ошибка: только администратор может изменять роль пользователя.' AS Message;  END  END;  GO  EXEC ИзменитьРольПользователя @ID\_пользователя = 2, @ID\_администратора = 4, @НоваяРоль = 'Преподаватель'; |

Листинг – Создание объектов в MSSQL

|  |
| --- |
| CREATE SEQUENCE seq\_Пользователь START WITH 1;  CREATE SEQUENCE seq\_Тест START WITH 1;  CREATE SEQUENCE seq\_Вопрос START WITH 1;  CREATE SEQUENCE seq\_Ответ START WITH 1;  CREATE SEQUENCE seq\_Результат\_теста START WITH 1;  drop SEQUENCE seq\_Пользователь;  drop SEQUENCE seq\_Тест;  drop SEQUENCE seq\_Вопрос;  drop SEQUENCE seq\_Ответ;  drop SEQUENCE seq\_Результат\_теста;  drop table Результат\_теста;  drop table Ответ;  drop table Вопрос;  drop table Тест;  drop table Пользователь;  -- ERROR: Если возникает ошибка, что нельзя делать триггеры на объектах,  -- которые принадлежат SYS, то делаем:  -- 1) Подключаемся не под SYS (гениально)  -- 2) Можно либо создать юзера, либо взять с прошлых лаб по бд (мой случай)  -- 3) Роль указываем default (вроде с этим у меня стало нормас работать)  CREATE TABLE Пользователь (  ID NUMBER PRIMARY KEY,  Имя NVARCHAR2(100) NOT NULL,  Фамилия NVARCHAR2(100) NOT NULL,  email NVARCHAR2(100) NOT NULL,  Пароль NVARCHAR2(100) NOT NULL,  Роль NVARCHAR2(20) CHECK (Роль IN ('Студент', 'Преподаватель', 'Администратор')) NOT NULL,  Данные NVARCHAR2(100)  );  CREATE TRIGGER trg\_Пользователь  BEFORE INSERT ON Пользователь  FOR EACH ROW  BEGIN  SELECT seq\_Пользователь.NEXTVAL  INTO :new.ID  FROM dual;  END;  /  CREATE TABLE Тест (  ID NUMBER PRIMARY KEY,  Название NVARCHAR2(100) NOT NULL,  Описание NVARCHAR2(255) NOT NULL,  Предмет NVARCHAR2(100) NOT NULL,  ID\_пользователя NUMBER REFERENCES Пользователь(ID) NOT NULL  );  CREATE TRIGGER trg\_Тест  BEFORE INSERT ON Тест  FOR EACH ROW  BEGIN  SELECT seq\_Тест.NEXTVAL  INTO :new.ID  FROM dual;  END;  /  CREATE TABLE Вопрос (  ID NUMBER PRIMARY KEY,  Текст NVARCHAR2(255) NOT NULL,  ID\_теста NUMBER REFERENCES Тест(ID) NOT NULL  );  CREATE TRIGGER trg\_Вопрос  BEFORE INSERT ON Вопрос  FOR EACH ROW  BEGIN  SELECT seq\_Вопрос.NEXTVAL  INTO :new.ID  FROM dual;  END;  /  CREATE TABLE Ответ (  ID NUMBER PRIMARY KEY,  Текст NVARCHAR2(255) NOT NULL,  Правильный NUMBER(1) NOT NULL,  ID\_вопроса NUMBER REFERENCES Вопрос(ID) NOT NULL  );  CREATE TRIGGER trg\_Ответ  BEFORE INSERT ON Ответ  FOR EACH ROW  BEGIN  SELECT seq\_Ответ.NEXTVAL  INTO :new.ID  FROM dual;  END;  /  CREATE TABLE Результат\_теста (  ID NUMBER PRIMARY KEY,  ID\_пользователя NUMBER REFERENCES Пользователь(ID) NOT NULL,  ID\_теста NUMBER REFERENCES Тест(ID) NOT NULL,  Дата\_и\_время DATE NOT NULL,  Итоговый\_балл FLOAT NOT NULL  );  CREATE TRIGGER trg\_Результат\_теста  BEFORE INSERT ON Результат\_теста  FOR EACH ROW  BEGIN  SELECT seq\_Результат\_теста.NEXTVAL  INTO :new.ID  FROM dual;  END;  /  CREATE INDEX IX\_Пользователь\_Имя\_Фамилия ON Пользователь (Имя, Фамилия);  CREATE INDEX IX\_Тест\_Предмет ON Тест (Предмет);  CREATE INDEX IX\_Вопрос\_ID\_теста ON Вопрос (ID\_теста);  CREATE INDEX IX\_Ответ\_ID\_вопроса ON Ответ (ID\_вопроса);  CREATE INDEX IX\_Результат\_теста\_ID\_пользователя ON Результат\_теста (ID\_пользователя);  -- INFO: Может вы в лабах уже делали хэширование паролей, но если нет, то  -- пропишите команду ниже с учетом имени вашего юзера (выполняем от SYS)  GRANT EXECUTE ON SYS.DBMS\_CRYPTO TO C##TDS;  -- Проверка (если 1 - то нормально, если 0 - то нет)  SELECT COUNT(\*)  FROM all\_objects  WHERE object\_name = 'DBMS\_CRYPTO' AND object\_type = 'PACKAGE';  -- ============================== Регистрация ==============================  CREATE OR REPLACE PROCEDURE Регистрация (  p\_Имя IN Пользователь.Имя%TYPE,  p\_Фамилия IN Пользователь.Фамилия%TYPE,  p\_Email IN Пользователь.Email%TYPE,  p\_Пароль IN Пользователь.Пароль%TYPE,  p\_Роль IN Пользователь.Роль%TYPE,  p\_Данные IN Пользователь.Данные%TYPE,  p\_Result OUT NUMBER,  p\_Message OUT NVARCHAR2  ) AS  v\_ПользовательСуществует NUMBER;  v\_Хеш\_Пароля VARCHAR2(100); -- Изменено на VARCHAR2 с максимальной длиной 100 символов  BEGIN  SELECT COUNT(\*)  INTO v\_ПользовательСуществует  FROM Пользователь  WHERE Email = p\_Email;  IF v\_ПользовательСуществует = 0 THEN  IF REGEXP\_LIKE(p\_Email, '^[A-Za-z0-9.\_%-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$') THEN  IF p\_Роль IN ('Студент', 'Преподаватель', 'Администратор') THEN  v\_Хеш\_Пароля := SUBSTR(UTL\_RAW.CAST\_TO\_RAW(DBMS\_CRYPTO.HASH(UTL\_I18N.STRING\_TO\_RAW(p\_Пароль, 'AL32UTF8'), DBMS\_CRYPTO.HASH\_SH256)), 1, 100);  INSERT INTO Пользователь (Имя, Фамилия, Email, Пароль, Роль, Данные)  VALUES (p\_Имя, p\_Фамилия, p\_Email, v\_Хеш\_Пароля, p\_Роль, p\_Данные);  p\_Result := 1;  p\_Message := 'Регистрация прошла успешно.';  ELSE  p\_Result := 0;  p\_Message := 'Недопустимая роль. Роль должна быть либо Студент, либо Преподаватель, либо Администратор.';  END IF;  ELSE  p\_Result := 0;  p\_Message := 'Неправильный формат email.';  END IF;  ELSE  p\_Result := 0;  p\_Message := 'Пользователь с таким email уже существует.';  END IF;  EXCEPTION  WHEN OTHERS THEN  p\_Result := 0;  p\_Message := SQLERRM;  END Регистрация;  /  -- ============================== Авторизация ==============================  CREATE OR REPLACE PROCEDURE Авторизация (  p\_Email IN Пользователь.Email%TYPE,  p\_Пароль IN Пользователь.Пароль%TYPE,  p\_Result OUT NUMBER,  p\_Message OUT NVARCHAR2  ) AS  v\_Хеш\_Пароля VARCHAR2(100); -- Изменено на VARCHAR2 с максимальной длиной 100 символов  BEGIN  SELECT Пароль  INTO v\_Хеш\_Пароля  FROM Пользователь  WHERE Email = p\_Email;  IF SUBSTR(UTL\_RAW.CAST\_TO\_RAW(DBMS\_CRYPTO.HASH(UTL\_I18N.STRING\_TO\_RAW(p\_Пароль, 'AL32UTF8'), DBMS\_CRYPTO.HASH\_SH256)), 1, 100) = v\_Хеш\_Пароля THEN  p\_Result := 1;  p\_Message := 'Авторизация прошла успешно.';  ELSE  p\_Result := 0;  p\_Message := 'Неверный email или пароль.';  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  p\_Result := 0;  p\_Message := 'Неверный email или пароль.';  WHEN OTHERS THEN  p\_Result := 0;  p\_Message := SQLERRM;  END Авторизация;  /  -- ============================== Провека регистрации и авторизации ==============================  DECLARE  v\_Result NUMBER;  v\_Message NVARCHAR2(100);  BEGIN  Регистрация('Dmitry', 'Trubach', 'dima1@mail.ru', 'dima123', 'Студент', 'Тест 1', v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);    v\_Result := NULL; -- Обнуляем переменные перед следующим вызовом  v\_Message := NULL;    Регистрация('Prepod', 'Prepod', 'dima2@mail.ru', 'dima1234', 'Преподаватель', 'Тест 2', v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);    v\_Result := NULL;  v\_Message := NULL;    Регистрация('Тестовый', 'Препод', 'dima3@mail.ru', 'dima12345', 'Преподаватель', 'Тест 3', v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);    v\_Result := NULL;  v\_Message := NULL;    Регистрация('admin', 'admin', 'dima4@mail.ru', 'dima123456', 'Администратор', 'Тест 3', v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  END;  /  SELECT \* FROM Пользователь;  DECLARE  v\_Result NUMBER;  v\_Message NVARCHAR2(100);  BEGIN  Авторизация('dima1@mail.ru', 'dima123', v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  END;  /  -- ============================== Добавление, редактирование и удаление теста ==============================  -- ============================== Проверка количества вопросов (<=5) ==============================  CREATE OR REPLACE TRIGGER Проверка\_Количества\_Вопросов  BEFORE INSERT ON Вопрос  FOR EACH ROW  DECLARE  Количество\_Вопросов NUMBER;  BEGIN  SELECT COUNT(\*)  INTO Количество\_Вопросов  FROM Вопрос  WHERE ID\_теста = :NEW.ID\_теста;  IF Количество\_Вопросов >= 5 THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Максимальное количество вопросов в тесте составляет 5.');  END IF;  END;  /  -- ============================== Добавить тест ==============================  CREATE OR REPLACE PROCEDURE Добавить\_тест (  p\_ID IN Тест.ID%TYPE,  p\_Название IN Тест.Название%TYPE,  p\_Описание IN Тест.Описание%TYPE,  p\_Предмет IN Тест.Предмет%TYPE,  p\_ID\_пользователя IN Тест.ID\_пользователя%TYPE,  p\_Result OUT NUMBER,  p\_Message OUT NVARCHAR2  ) AS  v\_Роль Пользователь.Роль%TYPE;  BEGIN  SELECT Роль  INTO v\_Роль  FROM Пользователь  WHERE ID = p\_ID\_пользователя;  IF v\_Роль = 'Преподаватель' THEN  INSERT INTO Тест (ID, Название, Описание, Предмет, ID\_пользователя)  VALUES (p\_ID, p\_Название, p\_Описание, p\_Предмет, p\_ID\_пользователя);  p\_Result := 1;  p\_Message := 'Тест успешно добавлен.';  ELSE  p\_Result := 0;  p\_Message := 'Недостаточно прав для добавления теста.';  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  p\_Result := 0;  p\_Message := 'Недостаточно прав для добавления теста.';  WHEN OTHERS THEN  p\_Result := 0;  p\_Message := SQLERRM;  END Добавить\_тест;  /  -- ============================== Редактировать тест ==============================  CREATE OR REPLACE PROCEDURE Редактировать\_тест (  p\_ID IN Тест.ID%TYPE,  p\_Название IN Тест.Название%TYPE,  p\_Описание IN Тест.Описание%TYPE,  p\_Предмет IN Тест.Предмет%TYPE,  p\_ID\_пользователя IN Тест.ID\_пользователя%TYPE,  p\_Result OUT NUMBER,  p\_Message OUT NVARCHAR2  ) AS  v\_Роль Пользователь.Роль%TYPE;  BEGIN  SELECT Роль  INTO v\_Роль  FROM Пользователь  WHERE ID = p\_ID\_пользователя;  IF v\_Роль = 'Преподаватель' THEN  UPDATE Тест  SET Название = p\_Название, Описание = p\_Описание, Предмет = p\_Предмет  WHERE ID = p\_ID;  p\_Result := 1;  p\_Message := 'Тест успешно отредактирован.';  ELSE  p\_Result := 0;  p\_Message := 'Недостаточно прав для редактирования теста.';  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  p\_Result := 0;  p\_Message := 'Недостаточно прав для редактирования теста.';  WHEN OTHERS THEN  p\_Result := 0;  p\_Message := SQLERRM;  END Редактировать\_тест;  /  -- ============================== Удалить тест ==============================  CREATE OR REPLACE PROCEDURE Удалить\_тест (  p\_ID IN Тест.ID%TYPE,  p\_ID\_пользователя IN Тест.ID\_пользователя%TYPE,  p\_Result OUT NUMBER,  p\_Message OUT NVARCHAR2  ) AS  v\_Роль Пользователь.Роль%TYPE;  BEGIN  SELECT Роль  INTO v\_Роль  FROM Пользователь  WHERE ID = p\_ID\_пользователя;  IF v\_Роль = 'Преподаватель' THEN  DELETE FROM Тест  WHERE ID = p\_ID;  p\_Result := 1;  p\_Message := 'Тест успешно удален.';  ELSE  p\_Result := 0;  p\_Message := 'Недостаточно прав для удаления теста.';  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  p\_Result := 0;  p\_Message := 'Недостаточно прав для удаления теста.';  WHEN OTHERS THEN  p\_Result := 0;  p\_Message := SQLERRM;  END Удалить\_тест;  /  -- ============================== Просмотр всех тестов ==============================  CREATE OR REPLACE VIEW Все\_тесты AS  SELECT ID, Название, Описание, Предмет  FROM Тест;  select \* from Тест; -- для просмотра ID  SELECT \* FROM Все\_тесты;  DECLARE  v\_Result NUMBER;  v\_Message NVARCHAR2(100);  BEGIN  Добавить\_тест(1, 'Тест №1', 'Тесттест', 'Математика', 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  END;  /  DECLARE  v\_Result NUMBER;  v\_Message NVARCHAR2(100);  BEGIN  Редактировать\_тест(1, 'Измененный тест 1', 'Измененное описание теста 1', 'Информатика', 1, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  END;  /  DECLARE  v\_Result NUMBER;  v\_Message NVARCHAR2(100);  BEGIN  Удалить\_тест(1, 1, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  END;  /  -- ============================== Добавление вопроса ==============================  CREATE OR REPLACE PROCEDURE Добавить\_Вопрос (  p\_ID IN Вопрос.ID%TYPE,  p\_Текст IN Вопрос.Текст%TYPE,  p\_ID\_теста IN Вопрос.ID\_теста%TYPE,  p\_ID\_пользователя IN Пользователь.ID%TYPE,  p\_Result OUT NUMBER,  p\_Message OUT NVARCHAR2  ) AS  v\_Роль Пользователь.Роль%TYPE;  v\_Тест\_принадлежит\_пользователю NUMBER;  BEGIN  SELECT Роль  INTO v\_Роль  FROM Пользователь  WHERE ID = p\_ID\_пользователя;  IF v\_Роль = 'Преподаватель' THEN  SELECT COUNT(\*)  INTO v\_Тест\_принадлежит\_пользователю  FROM Тест  WHERE ID = p\_ID\_теста AND ID\_пользователя = p\_ID\_пользователя;  IF v\_Тест\_принадлежит\_пользователю > 0 THEN  INSERT INTO Вопрос (ID, Текст, ID\_теста)  VALUES (p\_ID, p\_Текст, p\_ID\_теста);  p\_Result := 1;  p\_Message := 'Вопрос успешно добавлен.';  ELSE  p\_Result := 0;  p\_Message := 'Тест не принадлежит данному пользователю.';  END IF;  ELSE  p\_Result := 0;  p\_Message := 'Недопустимая роль. Только преподаватель может добавлять вопросы.';  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  p\_Result := 0;  p\_Message := 'Недопустимая роль. Только преподаватель может добавлять вопросы.';  WHEN OTHERS THEN  p\_Result := 0;  p\_Message := SQLERRM;  END Добавить\_Вопрос;  /  -- ============================== Добавление ответа ==============================  CREATE OR REPLACE PROCEDURE Добавить\_Ответ (  p\_ID IN Ответ.ID%TYPE,  p\_Текст IN Ответ.Текст%TYPE,  p\_Правильный IN Ответ.Правильный%TYPE,  p\_ID\_вопроса IN Ответ.ID\_вопроса%TYPE,  p\_ID\_пользователя IN Пользователь.ID%TYPE,  p\_Result OUT NUMBER,  p\_Message OUT NVARCHAR2  ) AS  v\_Роль Пользователь.Роль%TYPE;  v\_Вопрос\_принадлежит\_тесту NUMBER;  BEGIN  SELECT Роль  INTO v\_Роль  FROM Пользователь  WHERE ID = p\_ID\_пользователя;  IF v\_Роль = 'Преподаватель' THEN  SELECT COUNT(\*)  INTO v\_Вопрос\_принадлежит\_тесту  FROM Вопрос  JOIN Тест ON Вопрос.ID\_теста = Тест.ID  WHERE Вопрос.ID = p\_ID\_вопроса AND Тест.ID\_пользователя = p\_ID\_пользователя;  IF v\_Вопрос\_принадлежит\_тесту > 0 THEN  INSERT INTO Ответ (ID, Текст, Правильный, ID\_вопроса)  VALUES (p\_ID, p\_Текст, p\_Правильный, p\_ID\_вопроса);  p\_Result := 1;  p\_Message := 'Ответ успешно добавлен.';  ELSE  p\_Result := 0;  p\_Message := 'Вопрос не принадлежит тесту данного пользователя.';  END IF;  ELSE  p\_Result := 0;  p\_Message := 'Недопустимая роль. Только преподаватель может добавлять ответы.';  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  p\_Result := 0;  p\_Message := 'Недопустимая роль. Только преподаватель может добавлять ответы.';  WHEN OTHERS THEN  p\_Result := 0;  p\_Message := SQLERRM;  END Добавить\_Ответ;  /  select \* from Вопрос;  DECLARE  v\_Result NUMBER;  v\_Message NVARCHAR2(200);  BEGIN  -- ID, текст, ID\_теста, ID\_препода  Добавить\_Вопрос(1, 'Вопрос 1, ответ да', 1, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;    -- ID, текст, Правильность, ID\_вопроса, ID\_препода  Добавить\_Ответ(1, 'Да', 1, 1, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;  Добавить\_Ответ(2, 'Нет', 0, 1, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;    -- ID, текст, ID\_теста, ID\_препода  Добавить\_Вопрос(2, 'Вопрос 2, ответ да', 1, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;    -- ID, текст, Правильность, ID\_вопроса, ID\_препода  Добавить\_Ответ(3, 'Да', 1, 2, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;  Добавить\_Ответ(4, 'Нет', 0, 2, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;    -- ID, текст, ID\_теста, ID\_препода  Добавить\_Вопрос(3, 'Вопрос 3, ответ нет', 1, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;    -- ID, текст, Правильность, ID\_вопроса, ID\_препода  Добавить\_Ответ(5, 'Да', 0, 3, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;  Добавить\_Ответ(6, 'Нет', 1, 3, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;    -- ID, текст, ID\_теста, ID\_препода  Добавить\_Вопрос(4, 'Вопрос 4, ответ нет', 1, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;    -- ID, текст, Правильность, ID\_вопроса, ID\_препода  Добавить\_Ответ(7, 'Да', 0, 4, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;  Добавить\_Ответ(8, 'Нет', 1, 4, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;    -- ID, текст, ID\_теста, ID\_препода  Добавить\_Вопрос(5, 'Вопрос 5, ответ да', 1, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;    -- ID, текст, Правильность, ID\_вопроса, ID\_препода  Добавить\_Ответ(9, 'Да', 1, 5, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;  Добавить\_Ответ(10, 'Нет', 0, 5, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;  END;  /  -- Добавление 6-го вопроса (проверка триггера)  DECLARE  v\_Result NUMBER;  v\_Message NVARCHAR2(200);  BEGIN  -- ID, текст, ID\_теста, ID\_препода  Добавить\_Вопрос(6, 'Вопрос 6, ответ нет', 1, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;    -- ID, текст, Правильность, ID\_вопроса, ID\_препода  Добавить\_Ответ(11, 'Да', 0, 6, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;  Добавить\_Ответ(12, 'Нет', 1, 6, 2, v\_Result, v\_Message);  DBMS\_OUTPUT.PUT\_LINE('Result: ' || v\_Result || ', Message: ' || v\_Message);  v\_Result := NULL;  v\_Message := NULL;  END;  /  -- ============================== Вывод информации о тесте ==============================  CREATE OR REPLACE FUNCTION Информация\_О\_Тесте (p\_ID\_теста IN NUMBER)  RETURN SYS\_REFCURSOR IS  v\_результат SYS\_REFCURSOR;  BEGIN  OPEN v\_результат FOR  SELECT Название, Описание  FROM Тест  WHERE ID = p\_ID\_теста;    RETURN v\_результат;  END Информация\_О\_Тесте;  /  -- ============================== Вывод вопросов теста ==============================  CREATE OR REPLACE FUNCTION Вопросы\_Теста (p\_ID\_теста IN NUMBER)  RETURN SYS\_REFCURSOR IS  v\_результат SYS\_REFCURSOR;  BEGIN  OPEN v\_результат FOR  SELECT ID, Текст AS "Вопрос"  FROM Вопрос  WHERE ID\_теста = p\_ID\_теста;    RETURN v\_результат;  END Вопросы\_Теста;  /  -- Вызов функций  DECLARE  v\_результат SYS\_REFCURSOR;  v\_название Тест.Название%TYPE;  v\_описание Тест.Описание%TYPE;  v\_ID Вопрос.ID%TYPE;  v\_вопрос Вопрос.Текст%TYPE;  BEGIN  -- Вызов функции Информация\_О\_Тесте  v\_результат := Информация\_О\_Тесте(1);  FETCH v\_результат INTO v\_название, v\_описание;  CLOSE v\_результат;  DBMS\_OUTPUT.PUT\_LINE('Название: ' || v\_название || ', Описание: ' || v\_описание);  -- Вызов функции Вопросы\_Теста  v\_результат := Вопросы\_Теста(1); -- замените 1 на ID теста  LOOP  FETCH v\_результат INTO v\_ID, v\_вопрос;  EXIT WHEN v\_результат%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE('ID: ' || v\_ID || ', Вопрос: ' || v\_вопрос);  END LOOP;  CLOSE v\_результат;  END;  /  -- ============================== Проверка ответов ==============================  CREATE OR REPLACE PROCEDURE ПроверитьОтветы (  ID\_теста IN NUMBER,  ID\_пользователя IN NUMBER,  Ответы IN VARCHAR2  ) AS  КоличествоВопросов NUMBER;  ПравильныеОтветы NUMBER;  Оценка NUMBER;  BEGIN  -- Проверяем, что количество ответов соответствует количеству вопросов в тесте  SELECT COUNT(\*) INTO КоличествоВопросов FROM Вопрос WHERE ID\_теста = ID\_теста;  IF КоличествоВопросов <> REGEXP\_COUNT(Ответы, ',') + 1 THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка: количество ответов не соответствует количеству вопросов в тесте.');  RETURN;  END IF;  -- Сравниваем ответы с правильными ответами  FOR i IN 1..КоличествоВопросов LOOP  SELECT COUNT(\*)  INTO ПравильныеОтветы  FROM Вопрос  INNER JOIN Ответ ON Вопрос.ID = Ответ.ID\_вопроса  WHERE Вопрос.ID\_теста = ID\_теста  AND Ответ.Правильный = CASE  WHEN REGEXP\_SUBSTR(Ответы, '[^,]+', 1, i) = 'Да' THEN 1  WHEN REGEXP\_SUBSTR(Ответы, '[^,]+', 1, i) = 'Нет' THEN 0  ELSE NULL  END;  END LOOP;  -- Вычисляем оценку  IF ПравильныеОтветы = КоличествоВопросов THEN  Оценка := 10.0;  ELSIF ПравильныеОтветы = 0 THEN  Оценка := 0.0;  ELSE  Оценка := 10.0 - ((1.0 / КоличествоВопросов) \* 10.0 \* (КоличествоВопросов - ПравильныеОтветы));  END IF;  -- Выводим результаты и оценку  DBMS\_OUTPUT.PUT\_LINE('Оценка: ' || TO\_CHAR(Оценка));  -- Добавляем результаты в таблицу Результат\_теста  INSERT INTO Результат\_теста (ID\_пользователя, ID\_теста, Дата\_и\_время, Итоговый\_балл)  VALUES (ID\_пользователя, ID\_теста, SYSDATE, Оценка);  END;  /  BEGIN  ПроверитьОтветы(1, 1, 'Да,Нет,Да,Да,Да');  END;  /  SELECT \* FROM Результат\_теста;  CREATE OR REPLACE FUNCTION ВсеРезультаты RETURN SYS\_REFCURSOR IS  rc SYS\_REFCURSOR;  BEGIN  OPEN rc FOR SELECT ID, ID\_пользователя, ID\_теста, Дата\_и\_время, Итоговый\_балл FROM Результат\_теста;  RETURN rc;  END;  /  CREATE OR REPLACE FUNCTION РезультатыПользователя (p\_ID\_пользователя INT) RETURN SYS\_REFCURSOR IS  rc SYS\_REFCURSOR;  BEGIN  OPEN rc FOR SELECT ID, ID\_пользователя, ID\_теста, Дата\_и\_время, Итоговый\_балл FROM Результат\_теста WHERE ID\_пользователя = p\_ID\_пользователя;  RETURN rc;  END;  /  CREATE OR REPLACE FUNCTION РезультатыТеста (p\_ID\_теста INT) RETURN SYS\_REFCURSOR IS  rc SYS\_REFCURSOR;  BEGIN  OPEN rc FOR SELECT ID, ID\_пользователя, ID\_теста, Дата\_и\_время, Итоговый\_балл FROM Результат\_теста WHERE ID\_теста = p\_ID\_теста;  RETURN rc;  END;  /  DECLARE  rc SYS\_REFCURSOR;  row Результат\_теста%ROWTYPE;  BEGIN  rc := ВсеРезультаты();  DBMS\_OUTPUT.PUT\_LINE('ID, ID\_пользователя, ID\_теста, Дата\_и\_время, Итоговый\_балл');  LOOP  FETCH rc INTO row;  EXIT WHEN rc%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE(row.ID || ', ' || row.ID\_пользователя || ', ' || row.ID\_теста || ', ' || row.Дата\_и\_время || ', ' || row.Итоговый\_балл);  END LOOP;  CLOSE rc;  END;  /  DECLARE  rc SYS\_REFCURSOR;  row Результат\_теста%ROWTYPE;  BEGIN  rc := РезультатыПользователя(1);  DBMS\_OUTPUT.PUT\_LINE('ID, ID\_пользователя, ID\_теста, Дата\_и\_время, Итоговый\_балл');  LOOP  FETCH rc INTO row;  EXIT WHEN rc%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE(row.ID || ', ' || row.ID\_пользователя || ', ' || row.ID\_теста || ', ' || row.Дата\_и\_время || ', ' || row.Итоговый\_балл);  END LOOP;  CLOSE rc;  END;  /  DECLARE  rc SYS\_REFCURSOR;  row Результат\_теста%ROWTYPE;  BEGIN  rc := РезультатыТеста(1);  DBMS\_OUTPUT.PUT\_LINE('ID, ID\_пользователя, ID\_теста, Дата\_и\_время, Итоговый\_балл');  LOOP  FETCH rc INTO row;  EXIT WHEN rc%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE(row.ID || ', ' || row.ID\_пользователя || ', ' || row.ID\_теста || ', ' || row.Дата\_и\_время || ', ' || row.Итоговый\_балл);  END LOOP;  CLOSE rc;  END;  /  -- ============================== Изменение данных о пользователе ==============================  CREATE OR REPLACE PROCEDURE ИзменитьДанныеПользователя(  p\_ID\_пользователя INT,  p\_ID\_администратора INT,  p\_НовоеИмя NVARCHAR2,  p\_НоваяФамилия NVARCHAR2,  p\_НовыйEmail NVARCHAR2,  p\_НовыйПароль NVARCHAR2,  p\_НовыеДанные NVARCHAR2  ) AS  v\_count INT;  v\_Хеш\_Пароля VARCHAR2(100);  BEGIN  SELECT COUNT(\*) INTO v\_count FROM Пользователь WHERE ID = p\_ID\_администратора AND Роль = 'Администратор';    IF v\_count = 1 THEN  v\_Хеш\_Пароля := SUBSTR(UTL\_RAW.CAST\_TO\_RAW(DBMS\_CRYPTO.HASH(UTL\_I18N.STRING\_TO\_RAW(p\_НовыйПароль, 'AL32UTF8'), DBMS\_CRYPTO.HASH\_SH256)), 1, 100);  UPDATE Пользователь  SET Имя = p\_НовоеИмя,  Фамилия = p\_НоваяФамилия,  email = p\_НовыйEmail,  Пароль = v\_Хеш\_Пароля,  Данные = p\_НовыеДанные  WHERE ID = p\_ID\_пользователя;    DBMS\_OUTPUT.PUT\_LINE('1, Данные пользователя успешно обновлены.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('0, Ошибка: только администратор может изменять данные пользователя.');  END IF;  END;  /  DECLARE  BEGIN  ИзменитьДанныеПользователя(1, 4, 'Dmitry', 'Trubach', 'dima1@mail.ru', 'dima123', 'Тест 1');  END;  /  SELECT \* FROM Пользователь;  -- ============================== Изменение роли ==============================  CREATE OR REPLACE PROCEDURE ИзменитьРольПользователя(  p\_ID\_пользователя IN Пользователь.ID%TYPE,  p\_ID\_администратора IN Пользователь.ID%TYPE,  p\_НоваяРоль IN Пользователь.Роль%TYPE  ) AS  v\_count INT;  BEGIN  SELECT COUNT(\*) INTO v\_count FROM Пользователь WHERE ID = p\_ID\_администратора AND Роль = 'Администратор';    IF v\_count = 1 THEN  UPDATE Пользователь  SET Роль = p\_НоваяРоль  WHERE ID = p\_ID\_пользователя;    DBMS\_OUTPUT.PUT\_LINE('1, Роль пользователя успешно обновлена.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('0, Ошибка: только администратор может изменять роль пользователя.');  END IF;  END;  /  DECLARE  v\_Result NUMBER;  v\_Message NVARCHAR2(100);  BEGIN  ИзменитьРольПользователя(2, 4, 'Преподаватель');  END;  /  SELECT \* FROM Пользователь; |

Листинг – Создание объектов в Oracle