**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 2](#_Toc30437157)

[1. Проектирование логической схемы базы данных 3](#_Toc30437158)

[2. Таблицы базы данных 3](#_Toc30437159)

[3. Представления базы данных 3](#_Toc30437160)

[4. XML и пространственные объекты 3](#_Toc30437161)

[5. Пользователи базы данных и безопасность 3](#_Toc30437162)

[6. Индексы базы данных 3](#_Toc30437163)

[7. Программные объекты базы данных 3](#_Toc30437164)

[8. Резервное копирование базы данных 3](#_Toc30437165)

# ВВЕДЕНИЕ

Тема **Организация факультативов для студентов (вариант 5)**

В высшем учебном заведении организованы факультативы. Существует минимальный объем часов предметов, которые должен прослушать каждый студент. Информационные поля: ФИО студента, Адрес, Телефон, Название предмета, Объем лекций, Объем практических занятий, Объем лабораторных работ, Оценка, Дата экзамена.

Цель: контроль успеваемости студентов на факультативных занятиях.

Задачи:

.

# 1. Проектирование логической схемы базы данных

Логическая схема базы данных приведена на рисунке 1. ниже.

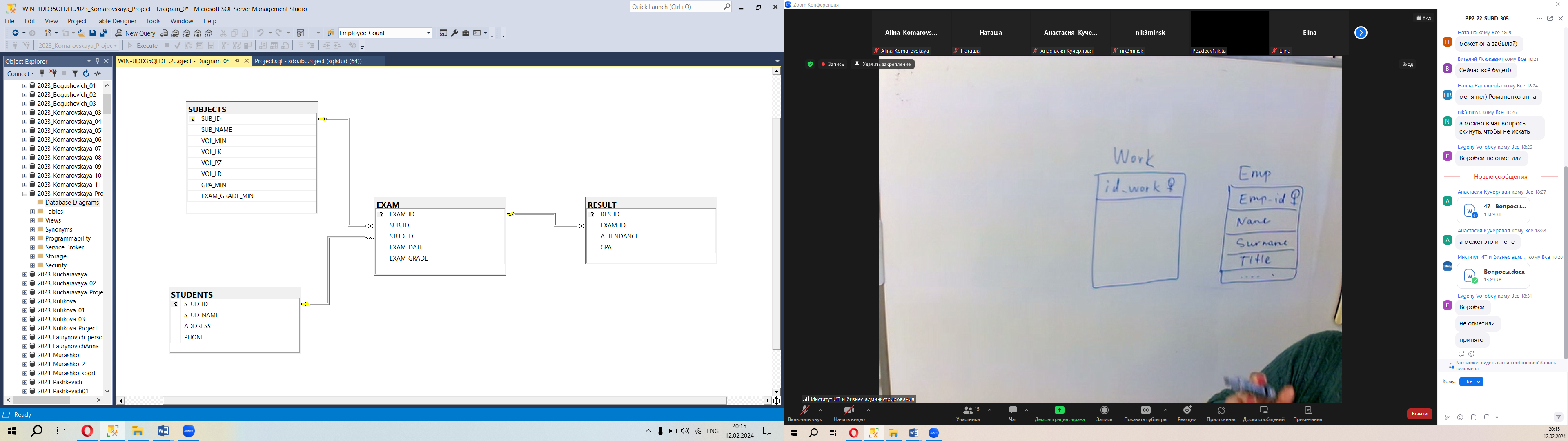


Рисунок 1. Логическая схема базы данных

# 2. Таблицы базы данных

Таблицы базы данных: наименование, поля, типы данных, ограничения.

Таблица STUDENTS (студенты).

Поля: STUD\_ID CHAR(3) NOT NULL,

STUD\_NAME VARCHAR(20) NOT NULL,

ADDRESS VARCHAR(50) NOT NULL,

PHONE INTEGER NOT NULL,

PRIMARY KEY (STUD\_ID)

Таблица SUBJECTS (предметы).

Поля: SUB\_ID CHAR(4) NOT NULL,

SUB\_NAME VARCHAR(20) NOT NULL,

VOL\_MIN DECIMAL(9,2),

VOL\_LK INT,

VOL\_PZ INT,

VOL\_LR INT,

GPA\_MIN DECIMAL(9,2), --минимальный ср.балл

CHECK (GPA\_MIN BETWEEN 1 AND 10),

EXAM\_GRADE\_MIN DECIMAL(9,2) NOT NULL,

CHECK (EXAM\_GRADE\_MIN BETWEEN 1 AND 10),

PRIMARY KEY (SUB\_ID)

Таблица EXAM (экзамен).

Поля: EXAM\_ID CHAR(10) NOT NULL,

SUB\_ID CHAR(4) NOT NULL,

STUD\_ID CHAR(3) NOT NULL,

EXAM\_DATE DATE NOT NULL,

EXAM\_GRADE DECIMAL(9,2),

CHECK (EXAM\_GRADE BETWEEN 1 AND 10),

PRIMARY KEY (EXAM\_ID),

CONSTRAINT EXAMSUB FOREIGN KEY (SUB\_ID)

REFERENCES SUBJECTS(SUB\_ID),

CONSTRAINT EXAMSTUD FOREIGN KEY (STUD\_ID)

REFERENCES STUDENTS(STUD\_ID)

Таблица RESULT (результат обучения).

RES\_ID CHAR(10) NOT NULL,

EXAM\_ID CHAR(10) NOT NULL,

ATTENDANCE DECIMAL(9,2), --посещаемость

GPA DECIMAL(9,2), --ср.балл

CHECK (GPA BETWEEN 1 AND 10),

PRIMARY KEY (RES\_ID),

CONSTRAINT RESEXAM FOREIGN KEY (EXAM\_ID)

REFERENCES EXAM(EXAM\_ID)

# 3. Представления базы данных

Представления базы данных: наименование, поля, типы данных, ограничения.

Представления (views) - это виртуальные таблицы, которые формируются на основе запросов к одной или нескольким таблицам. Вы можете использовать представления для:

- Сокрытия сложности запросов.  
- Обеспечения уровня абстракции и безопасности, позволяя пользователям доступ к данным без предоставления доступа к самим таблицам.  
- Объединения данных из различных таблиц для упрощения запросов.

--1. студенты, которые не сдали экзамены

CREATE VIEW View1 AS

SELECT E.STUD\_ID, E.SUB\_ID, R.EXAM\_ID, R.GPA, E.EXAM\_GRADE

FROM Result R join Exam E

ON R.EXAM\_ID = E.EXAM\_ID

WHERE E.EXAM\_GRADE is null;

GO

--показать представление

SELECT \* FROM View1;

--2.Экзамены, которые проходят в 2024

create view View2 as

select EXAM\_ID, SUB\_ID, STUD\_ID, EXAM\_DATE

from EXAM

where year(EXAM\_DATE) = 2024;

--ИЛИ

--where EXAM\_DATE between '2024-01-01' and '2024-12-31';

go

--обновить представление (год изменен на 2023 и добавить оценки EXAM\_GRADE)

--Если представление состоит из нескольких таблиц,

--его нельзя обновить из-за неоднозначности отношений между таблицами

ALTER VIEW View2 AS

select EXAM\_ID, SUB\_ID, STUD\_ID, EXAM\_DATE, EXAM\_GRADE

from EXAM

where year(EXAM\_DATE) = 2023;

--ИЛИ

--where EXAM\_DATE between '2023-01-01' and '2023-12-31';

go

SELECT \* FROM View2;

--удалить представление

DROP VIEW View1;

DROP VIEW View2;

# 4. XML и пространственные объекты

Использование пространственных данных и импорт/экспорт в/из XML.

Пространственные данные:

--Geography type point object (Los Angeles)

DECLARE @g geography;

SET @g = geography::STGeomFromText('POINT (-118.2423 34.0225)', 4326)

SELECT @g

--Geometry type point object

DECLARE @h geometry;

SET @h = geometry::STGeomFromText('POINT (-118.2423 34.0225)', 4326)

SELECT @h

--Экспорт данных из MS SQL в XML

SELECT \*

FROM RESULT

FOR XML PATH('row'), ROOT('rows') --FOR XML позволяет форматировать результаты запроса XML напрямую в SQL запросе

--Импорт XML в MS SQL

INSERT INTO RESULT (XmlCol)

SELECT \*

FROM OPENROWSET(

BULK 'C:\Users\Alina\Microsoft SQL Server Management Studio 19\SQLServer\fileFROM.xml',

SINGLE\_BLOB) --SINGLE\_BLOB - исользуя это можно избежать несоответствия между кодировкой XML-документа и строковой кодовой страницой, подразумеваемой сервером

AS x;

--7. Импортировать данные из внешних источников.

--Импорт должен быть отражен в скрипте или в пояснительной записке.

--1. Откройте SQL Server Management Studio (SSMS)\*\* и подключитесь к серверу, на который хотите импортировать данные.

--2. Щелкните правой кнопкой мыши по базе данных, в которую хотите импортировать данные, выберите "Tasks" > "Import Data...".

--3. Мастер импорта данных запустится. Щелкните "Next" для продолжения.

--4. Выберите источник данных из предоставленного списка и настройте его параметры.

--5. Выберите целевую базу данных и таблицу.

--6. Можно настроить отображение (mapping) столбцов, если это необходимо.

--7. Последние шаги мастера позволяют выполнить импорт сразу или сохранить его как SSIS пакет для последующего использования.

--8. Просмотрите сводку и нажмите "Finish", чтобы выполнить импорт.

--BULK INSERT: для импорта данных из файла в SQL Server таблицу.

--8. Создать процедуру для экспорта данных.

--Экспорт должен быть отражен в скрипте или в пояснительной записке.

USE [2023\_Komarovskaya\_Project];

GO

-- Создаем процедуру экспорта

CREATE PROCEDURE ExportData

@tableName VARCHAR(256), -- имя таблицы, данные которой вы хотите экспортировать

@exportPath NVARCHAR(256) --путь файла, куда будет произведен экспорт данных

AS

BEGIN

SET NOCOUNT ON;

DECLARE @sqlCmd NVARCHAR(MAX) = 'sqlcmd -S '

+ @@SERVERNAME

+ ' -d '

+ DB\_NAME()

+ ' -E -Q "SET NOCOUNT ON; SELECT \* FROM '

+ @tableName

+ '" -o "'

+ @exportPath

+ '" -h-1 -s"," -w 700';

EXEC xp\_cmdshell @sqlCmd;

SET NOCOUNT OFF;

END;

GO

--Перед тем, как использовать эту процедуру, удостоверьтесь, что служба 'xp\_cmdshell' включена:

-- Активируем xp\_cmdshell

--НЕ НАЖИМАТЬ

EXEC sp\_configure 'show advanced options', 1;

RECONFIGURE;

EXEC sp\_configure 'xp\_cmdshell', 1;

RECONFIGURE;

--выполняе процедуру

EXEC ExportData @tableName = 'Export', @exportPath = 'C:\Users\Alina\Microsoft SQL Server Management Studio 19\SQLServer\Export.csv';

# 5. Пользователи базы данных и безопасность

Пользователи базы данных: наименование, гранты, роли.

CREATE ROLE ADMIN

CREATE ROLE STUDENT

CREATE ROLE TEACHER;

GO

ALTER ROLE db\_datareader ADD MEMBER STUDENT

GO

-- предоставление права на чтение для STUDENT на таблицу RESULT

GRANT SELECT ON RESULT TO STUDENT;

GO

ALTER ROLE db\_datareader ADD MEMBER TEACHER

GO

ALTER ROLE db\_datawriter ADD MEMBER TEACHER

GO

-- предоставление права на чтение и редактирование для TEACHER на таблицу RESULT и EXAM

GRANT SELECT ON RESULT TO STUDENT;

GO

GRANT SELECT ON EXAM TO STUDENT;

GO

ALTER ROLE db\_datareader ADD MEMBER ADMIN

GO

ALTER ROLE db\_datawriter ADD MEMBER ADMIN

GO

ALTER ROLE db\_securityadmin ADD MEMBER ADMIN --изменять членство в роли (только для настраиваемых ролей) и управлять разрешениями

GO

ALTER ROLE db\_accessadmin ADD MEMBER ADMIN --добавлять или удалять доступ к базе данных

GO

ALTER ROLE db\_backupoperator ADD MEMBER ADMIN --могут создавать резервные копии базы данных

GO

--предоставление права для ADMIN

GRANT SELECT ON STUDENTS TO ADMIN;

GO

GRANT SELECT ON SUBJECTS TO ADMIN;

GO

GRANT SELECT ON EXAM TO ADMIN;

GO

GRANT SELECT ON RESULT TO ADMIN;

GO

-- удаление права на чтение для STUDENT на таблицу RESULT

REVOKE SELECT ON RESULT FROM STUDENT;

GO

--5.7. Выдать привилегии на запуск процедур и функций.

GRANT EXECUTE ON

dbo.Procedure1 TO ADMIN;

# 6. Индексы базы данных

Индексы базы данных: наименование, тип, цель создания.

Индексы используются для ускорения поиска данных в таблицах.

--найти студентов, которые на экзамене по матем получили больше проходного балла (5,5)

select \* from EXAM;

select \* from SUBJECTS;

select \* from RESULT --0.0032985 --0.0032985

where GPA > 6.00;

--средняя по группе

select RES\_ID, avg(GPA) --0.0033 --0.0033

from RESULT

group by RES\_ID;

create index idx\_RES\_GPA on RESULT (GPA);

--select E.STUD\_ID, S.SUB\_ID, S.SUB\_NAME, S.EXAM\_GRADE\_MIN, E.EXAM\_GRADE --0.0066022 --0.013

--from EXAM E join SUBJECTS S

--on E.SUB\_ID = S.SUB\_ID

--where E.SUB\_ID = 'MATH' and E.EXAM\_GRADE > S.EXAM\_GRADE\_MIN;

--create index idx\_EXAM\_EXAM\_GRADE\_SUB\_ID on EXAM(EXAM\_GRADE) include (SUB\_ID);

--create index idx\_SUB\_EXAM\_GRADE\_MIN on SUBJECTS(EXAM\_GRADE\_MIN);

# 7. Программные объекты базы данных

Программные объекты базы данных: наименование, параметры, возвращаемое значение, тип (процедуры, функции, триггеры).

Процедуры (подпрограммы) принимают параметры, возвращают значения, и производят все операции, которые можно реализовать с помощью T-SQL, включая выполнение транзакций, обработку ошибок и т.д.:

CREATE PROCEDURE Procedure1

AS

BEGIN

BEGIN TRANSACTION -- Начало транзакции

BEGIN TRY

-- Объявление курсора для построчного чтения из таблицы

DECLARE myCursor CURSOR FOR

SELECT \* FROM RESULT WHERE GPA >= 9.00

OPEN myCursor -- Открытие курсора и начало чтения

DECLARE @GPA DECIMAL

FETCH NEXT FROM myCursor INTO @GPA -- Чтение первой строки

WHILE @@FETCH\_STATUS = 0 -- Проход по строкам таблицы (читает, пока 0; перестает читать, когда 1)

BEGIN

-- обновление ATTENDANCE

UPDATE RESULT

SET ATTENDANCE = 2000.00

WHERE GPA = @GPA

FETCH NEXT FROM myCursor INTO @GPA -- Чтение следующей строки

END

CLOSE myCursor -- Закрытие курсора и освобождение ресурсов

DEALLOCATE myCursor --освобождение ресурсов

COMMIT TRANSACTION-- Если все прошло успешно, подтвердить транзакцию

END TRY

BEGIN CATCH

-- Если произошла ошибка, закрыть курсор и откатить транзакцию

IF CURSOR\_STATUS('global','myCursor') >= 0

BEGIN

CLOSE myCursor

DEALLOCATE myCursor

END

ROLLBACK TRANSACTION-- Откат транзакции

-- Получение информации об ошибке и её выброс

DECLARE @ErrorNumber INT = ERROR\_NUMBER()

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE()

DECLARE @ErrorSeverity INT = ERROR\_SEVERITY()

DECLARE @ErrorState INT = ERROR\_STATE()

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState)

END CATCH

END

GO

Функции — это программные единицы, которые могут принимать параметры, выполнять вычисления или действия и возвращать результат. Существуют скалярные функции, возвращающие одно значение, и табличные функции, возвращающие таблицу.

--1. Скалярная функция (возвращает одно значение):

CREATE FUNCTION dbo.GetFormattedDate(@Date DATETIME)

RETURNS VARCHAR(30)

AS

BEGIN

-- Возвращает дату в формате 'DD-MM-YYYY'

RETURN CONVERT(VARCHAR(10), @Date, 105)

END

GO

--Чтобы использовать эту функцию, надо ее вызвать:

SELECT dbo.GetFormattedDate(GETDATE()) AS FormattedDate

--2. Табличная функция (возвращает набор строк):

CREATE FUNCTION dbo.GetListStudents(@ListStudents NVARCHAR(50))

RETURNS TABLE

AS

RETURN

(

SELECT \*

FROM STUDENTS

);

GO

--Использовать табличную функцию:

SELECT \* FROM dbo.GetListStudents('');

Триггеры – это специальный тип хранимой процедуры, который автоматически выполняется (срабатывает) в ответ на определенные события на таблице или представлении. Существуют разные типы триггеров, включая “AFTER” триггеры (срабатывают после операции), “INSTEAD OF” триггеры (заменяют операцию) и триггеры DDL (срабатывают в ответ на изменения в определении данных, например, создание или изменение таблицы).

--создание таблицы, в которую триггером добавляются записи об изменениях

CREATE TABLE StudentsAudit

( STUD\_ID CHAR(3) NOT NULL,

STUD\_NAME VARCHAR(20) NOT NULL,

ADDRESS VARCHAR(50) NOT NULL,

PHONE INTEGER NOT NULL,

PRIMARY KEY (STUD\_ID));

GO

select \* from STUDENTS;

select \* from StudentsAudit;

GO

-- создание AFTER INSERT триггера (добавление)

CREATE TRIGGER trg\_Insert ON STUDENTS

AFTER INSERT

AS

BEGIN

-- при добавлении данных в таблице STUDENTS, старые данные сохраняются в таблицу StudentsAudit

INSERT INTO StudentsAudit (STUD\_ID, STUD\_NAME, ADDRESS)

SELECT i.STUD\_ID, 'INSERT', GETDATE()

FROM inserted i; --inserted - таблица, доступная внутри триггера, которая содержит копии всех строк, подвергающихся операции INSERT

END;

GO

-- создание AFTER UPDATE триггера (обновление)

CREATE TRIGGER trg\_Update ON STUDENTS

AFTER UPDATE

AS

BEGIN

-- при обновлении данных в таблице STUDENTS, старые данные сохраняются в таблицу StudentsAudit

INSERT INTO StudentsAudit (STUD\_ID, STUD\_NAME, ADDRESS, PHONE)

SELECT d.STUD\_ID, 'UPDATE', GETDATE(), CONCAT(',', d.STUD\_ID, d.STUD\_NAME, d.ADDRESS, d.PHONE)

FROM deleted d;

END;

GO

-- создание AFTER DELETE триггера (удаление)

CREATE TRIGGER trg\_Delete ON STUDENTS

AFTER DELETE

AS

BEGIN

-- при удалении данных в таблице STUDENTS, старые данные сохраняются в таблицу StudentsAudit

INSERT INTO StudentsAudit (STUD\_ID, STUD\_NAME, ADDRESS)

SELECT d.STUD\_ID, 'DELETE', GETDATE()

FROM deleted d;

END;

GO

--Удаление триггера

DROP TRIGGER trg\_Insert;

DROP TRIGGER trg\_Update;

DROP TRIGGER trg\_Delete;

--Отключение и включение триггера

DISABLE TRIGGER trg\_Insert ON STUDENTS; --выкл

ENABLE TRIGGER trg\_Insert ON STUDENTS; --вкл

# 8. Резервное копирование базы данных

Настройка резервного копирования.

BACKUP DATABASE [2023\_Komarovskaya\_08]

TO DISK = 'C:\Users\Alina\Microsoft SQL Server Management Studio 19\SQLServer\BACKUP.bak'

WITH NOFORMAT, -- NOFORMAT говорит о том, что не нужно форматировать накопитель, использовать существующий.

NOINIT,-- указывает, что файл резервной копии не должен инициализироваться заново, но новая резервная копия будет добавляться в существующий файл.

SKIP, -- означает пропустить проверку метки резервной медии и метки окончания ленты.

-- `NOREWIND` и `NOUNLOAD` применяются в случае использования ленточных накопителей, инструктируя SQL Server не перематывать и не извлекать ленту после окончания резервного копирования.

NOREWIND,

NOUNLOAD,

STATS = 10;-- определяет процент выполнения операции, который будет отображаться во время резервного копирования.

-- STATS = 10 отображает прогресс выполнения каждые 10%.

**Проверка для созданных программных объектов**

6. Добавить в скрипт проверку для всех созданных программных объектов.

--1. Проверка существования таблицы:

IF EXISTS (SELECT \* FROM sys.tables WHERE name = 'STUDENTS')

BEGIN

-- Таблица существует

PRINT 'Таблица существует.'

END

ELSE

BEGIN

-- Таблица не найдена

PRINT 'Таблица не найдена.'

END

--2. Проверка существования хранимой процедуры:

IF EXISTS (SELECT \* FROM sys.procedures WHERE name = 'Procedure1')

BEGIN

-- Хранимая процедура существует

PRINT 'Хранимая процедура существует.'

END

ELSE

BEGIN

-- Хранимая процедура не найдена

PRINT 'Хранимая процедура не найдена.'

END

--3. Проверка существования функции:

IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID('dbo.GetFormattedDate'))

BEGIN

-- Функция существует

PRINT 'Функция существует.'

END

ELSE

BEGIN

-- Функция не найдена

PRINT 'Функция не найдена.'

END

--4. Проверка существования триггера:

IF EXISTS (SELECT \* FROM sys.triggers WHERE name = 'trg\_Update')

BEGIN

-- Триггер существует

PRINT 'Триггер существует.'

END

ELSE

BEGIN

-- Триггер не найден

PRINT 'Триггер не найден.'

END

--5. Проверка существования представления:

IF EXISTS (SELECT \* FROM sys.views WHERE name = 'View2')

BEGIN

-- Представление существует

PRINT 'Представление существует.'

END

ELSE

BEGIN

-- Представление не найдено

PRINT 'Представление не найдено.'

END