

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра САПР**

**ОТЧЕТ**

**по ИДЗ**

**по дисциплине "Базы данных"**

**Тема: "Разработка базы данных для обеспечения работы приёмной  
комиссии университета"**

Студенты гр. 2311

Завьялов Н.С.

Преподаватель

Новакова Н.Е.

Санкт-Петербург

2024

1. Краткое описание предметной области

База данных для приёмной комиссии университета предназначена для автоматизации процессов приёма абитуриентов, хранения и обработки их заявлений, экзаменационных результатов, а также управления образовательными программами. Система обеспечивает:

- Регистрацию и учёт абитуриентов с уникальной идентификацией.
- Управление образовательными программами и их требованиями.
- Подачу заявок на поступление и отслеживание их статусов.
- Учёт результатов Единого государственного экзамена (ЕГЭ) абитуриентов.
- Контроль соответствия заявок требованиям образовательных программ.

Цель базы данных — повысить эффективность работы приёмной комиссии, обеспечить целостность и доступность данных, а также упростить процессы принятия решений о приёме абитуриентов.

2. Проектирование базы данных (структура данных)

Описание структуры таблицы БД		Наименование таблицы БД: Таблица Абитуриентов		Имя таблицы: Entrant	
Дата разработки: 20.11.2024					
Порядковый номер таблицы: 1					
№ п/п	Наименование поля	Спецификация данных			
		Имя поля	Тип данных	Ключ	Ограничения целостности
1	Идентификатор	ID	INT	P	Св-во: IDENTITY(1,1)
2	Фамилия	LastName	NVARCHAR(50)		NOT NULL
3	Имя	FirstName	NVARCHAR(50)		NOT NULL
4	Отчество	MiddleName	NVARCHAR(50)		
5	Дата рождения	DateOfBirth	DATE		NOT NULL
6	Пол	Gender	CHAR(1)		NOT NULL CHECK (Gender IN ('M', 'F'))
7	СНИЛС	SNILS	CHAR(11)		UNIQUE NOT NULL
8	Серия и номер паспорта	PassportNumber	CHAR(9)		UNIQUE NOT NULL

Описание структуры таблицы БД		Наименование таблицы БД: Таблица Образовательных Программ		Имя таблицы: EducationProgram	
Дата разработки: 20.11.2024					
Порядковый номер таблицы: 2					
№ п/п	Наименование поля	Спецификация данных			
		Имя поля	Тип данных	Ключ	Ограничения целостности
1	Идентификатор	EducationProgramID	INT	P	Св-во: IDENTITY(1,1)
2	Название программы	ProgramName	NVARCHAR(100)		NOT NULL
3	Факультет	Faculty	NVARCHAR(100)		NOT NULL
4	Уровень образования	EducationLevel	NVARCHAR(50)		
5	Количество бюджетных мест	BudgetPlaces	INT		NOT NULL

Описание структуры таблицы БД		Наименование таблицы БД: Таблица Заявок на Поступление		Имя таблицы: Application	
Дата разработки: 20.11.2024					
Порядковый номер таблицы: 3					
№ п/п	Наименование поля	Спецификация данных			
		Имя поля	Тип данных	Кл юч	Ограничения целостности
1	Идентификатор	ApplicationID	INT	P	Св-во: IDENTITY(1,1)
2	Связь с абитуриентом	EntrantID	INT	F	NOT NULL
3	Связь с образовательной программой	EducationProgramID	INT	F	NOT NULL
4	Дата подачи	ApplicationDate	DATE		NOT NULL
5	Статус заявки	Status	NVARCHAR(50)		

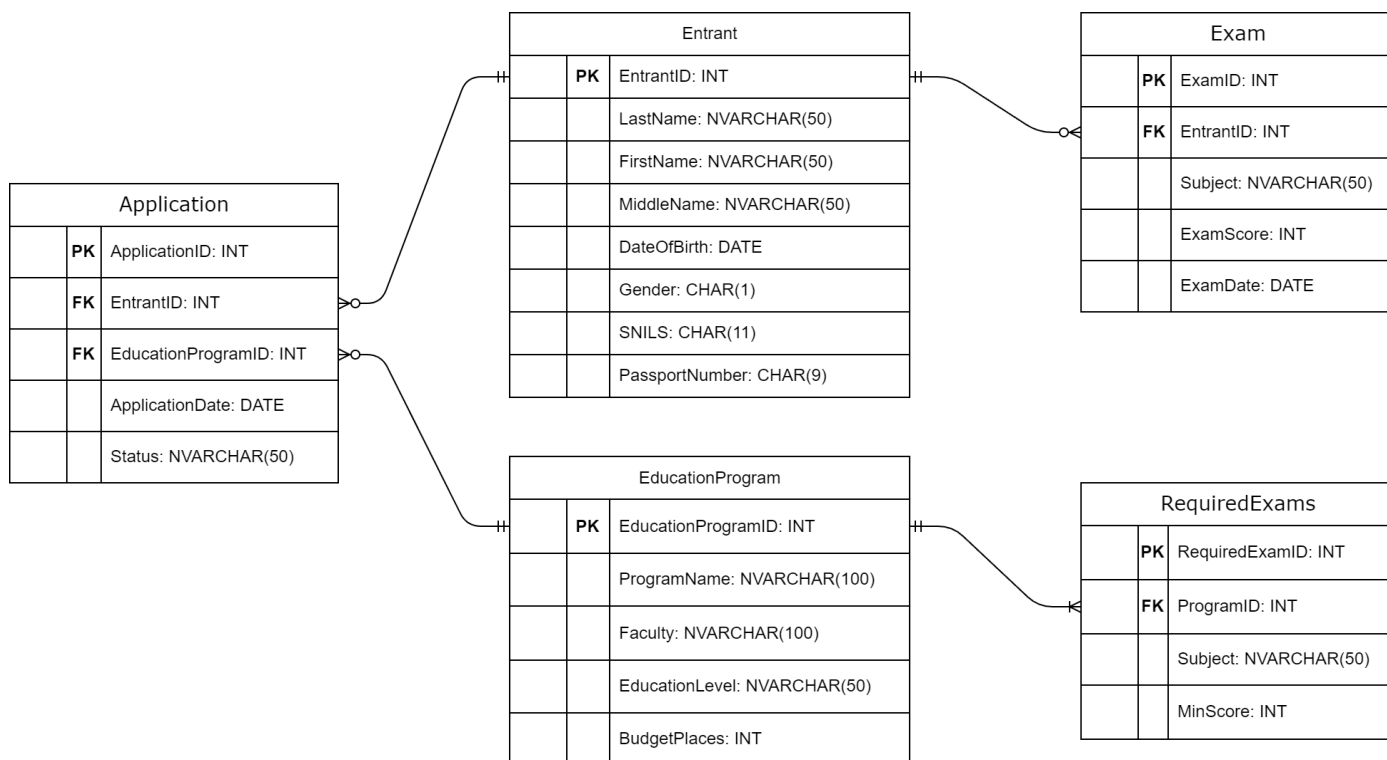
Описание структуры таблицы БД		Наименование таблицы БД: Таблица Сданных Экзаменов		Имя таблицы: Exam	
Дата разработки: 20.11.2024					
Порядковый номер таблицы: 4					

№ п/п	Наименование поля	Спецификация данных			
		Имя поля	Тип данных	Ключ	Ограничения целостности
1	Идентификатор	ExamID	INT	P	Св-во: IDENTITY(1,1)
2	Связь с абитуриентом	EntrantID	INT	F	NOT NULL
3	Название предмета	Subject	NVARCHAR(50)		NOT NULL
4	Результат экзамена	ExamScore	INT		CHECK (ExamScore BETWEEN 0 AND 100)
5	Дата сдачи экзамена	ExamDate	DATE		NOT NULL

Описание структуры таблицы БД		Наименование таблицы БД: Таблица Обязательных Экзаменов		Имя таблицы: RequiredExams	
Дата разработки: 20.11.2024					
Порядковый номер таблицы: 5					

№ п/п	Наименование поля	Спецификация данных			
		Имя поля	Тип данных	Ключ	Ограничения целостности
1	Идентификатор	RequiredExamID	INT	P	Св-во: IDENTITY(1,1)
2	Связь с программой	ProgramID	INT	F	NOT NULL
3	Название предмета	Subject	NVARCHAR(50)		NOT NULL
4	Проходной балл	MinScore	INT		CHECK (MinScore BETWEEN 0 AND 100)

## ER-диаграмма



## Связи между сущностями

- **Entrant ↔ Application: Один-ко-многим**
  - Один абитуриент может подать несколько заявок на разные программы, но не обязательно (может быть и не одной).
  - Каждая заявка обязательно связана с одним абитуриентом.
- **EducationProgram ↔ Application: Один-ко-многим**
  - Одна образовательная программа может иметь множество заявок от разных абитуриентов, но может быть и не одной заявки на программу.
  - Каждая заявка обязательно относится к одной программе.
- **EducationProgram ↔ RequiredExams: Один-ко-многим**
  - Одна программа должна иметь несколько обязательных экзаменов.
  - Каждый требуемый экзамен обязательно связан с одной программой.
- **Entrant ↔ Exam: Один-ко-многим**
  - Один абитуриент может сдавать несколько экзаменов, но может и ничего не сдавать вовсе.
  - Каждый экзамен обязательно связан с одним абитуриентом.

### 3. Создание базы данных

Для создания базы данных запустим следующий код:

```
CREATE DATABASE UniversityAdmission
ON PRIMARY (
    NAME = UniversityAdmission_Data,
    FILENAME = 'E:\SQLData\UniversityAdmission.mdf',
    SIZE = 100MB,
    MAXSIZE = 500MB,
    FILEGROWTH = 10%
)
LOG ON (
    NAME = UniversityAdmission_Log,
    FILENAME = 'D:\SQLData\UniversityAdmission.ldf',
    SIZE = 50MB,
    MAXSIZE = 250MB,
    FILEGROWTH = 5%
);
```

Использовать созданную базу данных:

```
USE UniversityAdmission;
GO
```

Создание схемы:

```
CREATE SCHEMA MySchema;
GO
```

#### 4. Создание таблиц и ограничений целостности

Ниже представлены SQL-команды для создания всех необходимых таблиц с учетом ограничений целостности.

##### Таблица "Entrant" (Абитуриент)

```
CREATE TABLE MySchema.Entrant (  
    EntrantID INT IDENTITY(1,1) PRIMARY KEY,  
    LastName NVARCHAR(50) NOT NULL,  
    FirstName NVARCHAR(50) NOT NULL,  
    MiddleName NVARCHAR(50) NULL,  
    DateOfBirth DATE NOT NULL,  
    Gender CHAR(1) NOT NULL CHECK (Gender IN ('M', 'F')),  
    SNILS CHAR(11) UNIQUE NOT NULL,  
    PassportNumber CHAR(9) UNIQUE NOT NULL  
);
```

##### Таблица "EducationProgram" (Образовательная программа)

```
CREATE TABLE MySchema.EducationProgram (  
    EducationProgramID INT IDENTITY(1,1) PRIMARY KEY,  
    ProgramName NVARCHAR(100) NOT NULL,  
    Faculty NVARCHAR(100) NOT NULL,  
    EducationLevel NVARCHAR(50) NULL,  
    BudgetPlaces INT NOT NULL DEFAULT 0  
);
```

##### Таблица "Application" (Заявка)

```
CREATE TABLE MySchema.Application (  
    ApplicationID INT IDENTITY(1,1) PRIMARY KEY,  
    EntrantID INT NOT NULL,
```

```
EducationProgramID INT NOT NULL,  
ApplicationDate DATE NOT NULL,  
Status NVARCHAR(50) NULL,  
UNIQUE (EntrantID, EducationProgramID),  
FOREIGN KEY (EntrantID) REFERENCES Entrant(EntrantID),  
FOREIGN KEY (EducationProgramID) REFERENCES  
EducationProgram(EducationProgramID)  
);
```

### Таблица "Exam" (ЕГЭ)

```
CREATE TABLE MySchema.Exam (  
    ExamID INT IDENTITY(1,1) PRIMARY KEY,  
    EntrantID INT NOT NULL,  
    Subject NVARCHAR(50) NOT NULL,  
    ExamScore INT CHECK (ExamScore BETWEEN 0 AND 100),  
    ExamDate DATE NOT NULL,  
    FOREIGN KEY (EntrantID) REFERENCES Entrant(EntrantID)  
);
```

### Таблица "RequiredExams" (Требуемые экзамены)

```
CREATE TABLE MySchema.RequiredExams (  
    RequiredExamID INT IDENTITY(1,1) PRIMARY KEY,  
    ProgramID INT NOT NULL,  
    Subject NVARCHAR(50) NOT NULL,  
    MinScore INT CHECK (MinScore BETWEEN 0 AND 100),  
    UNIQUE (ProgramID, Subject),  
    FOREIGN KEY (ProgramID) REFERENCES  
EducationProgram(EducationProgramID));
```



## 5. Заполнение таблиц данными

Для демонстрации работы базы данных, заполним таблицы примерными данными.

### Добавление абитуриентов

```
INSERT INTO MySchema.Entrant (LastName, FirstName,
MiddleName, DateOfBirth, Gender, SNILS, PassportNumber)
VALUES
('Иванов', 'Иван', 'Иванович', '2003-08-15', 'М',
'12345678901', '123456789'),
('Петрова', 'Анна', 'Сергеевна', '2002-05-22', 'F',
'23456789012', '234567890'),
('Сидоров', 'Пётр', 'Алексеевич', '2003-11-30', 'М',
'34567890123', '345678901');
```

### Добавление образовательных программ

```
INSERT INTO MySchema.EducationProgram (ProgramName,
Faculty, EducationLevel, BudgetPlaces)
VALUES
('Компьютерные науки', 'Факультет информационных
технологий', 'Бакалавриат', 25),
('Электронная инженерия', 'Факультет электроники',
'Бакалавриат', 20),
('Информационная безопасность', 'Факультет информационной
безопасности', 'Бакалавриат', 15),
('Автоматизация и управление', 'Факультет автоматизации',
'Бакалавриат', 18),
('Робототехника', 'Факультет робототехники', 'Магистратура',
12);
```

### Добавление требований по экзаменам для программ

```
INSERT INTO MySchema.RequiredExams (ProgramID, Subject,  
MinScore)
```

```
VALUES
```

```
(1, 'Математика', 70),  
(1, 'Русский язык', 60),  
(1, 'Информатика', 65),  
(2, 'Математика', 75),  
(2, 'Русский язык', 60),  
(3, 'Математика', 80),  
(3, 'Русский язык', 70),  
(3, 'Информатика', 75);
```

### Добавление экзаменов абитуриентов

```
INSERT INTO MySchema.Exam (EntrantID, Subject, ExamScore,  
ExamDate)
```

```
VALUES
```

```
(1, 'Математика', 78, '2024-06-15'),  
(1, 'Русский язык', 65, '2024-06-16'),  
(1, 'Информатика', 70, '2024-06-17'),  
(2, 'Математика', 82, '2024-06-15'),  
(2, 'Русский язык', 75, '2024-06-16'),  
(3, 'Математика', 68, '2024-06-15'),  
(3, 'Русский язык', 58, '2024-06-16'),  
(3, 'Информатика', 60, '2024-06-17');
```

### Подача заявок на образовательные программы

```
INSERT INTO MySchema.Application (EntrantID,  
EducationProgramID, ApplicationDate, Status)  
VALUES  
(1, 1, '2024-06-20', 'Подана'),  
(2, 2, '2024-06-21', 'Подана'),  
(3, 1, '2024-06-22', 'Подана');
```

## 6. Разработка объектов промежуточного слоя

Объекты промежуточного слоя включают представления (views), хранимые процедуры (stored procedures) и пользовательские функции (UDF). Они облегчают доступ к данным и выполняют часто используемые операции.

### Представления.

Представление *View\_Applications*

Отображает заявки с именами абитуриентов и названиями программ.

```
CREATE VIEW MySchema.View_Applications AS  
SELECT  
    a.ApplicationID,  
    e.LastName,  
    e.FirstName,  
    e.MiddleName,  
    p.ProgramName,  
    a.ApplicationDate,  
    a.Status
```

```
FROM
    MySchema.Application a
JOIN
    MySchema.Entrant e ON a.EntrantID = e.EntrantID
JOIN
    MySchema.EducationProgram p ON a.EducationProgramID =
p.EducationProgramID;
```

Пример использования:

```
SELECT * FROM MySchema.View_Applications;
```

Результат запроса:

	ApplicationID	LastName	FirstName	MiddleName	ProgramName	ApplicationDate	Status
1	1	Иванов	Иван	Иванович	Программная инженерия	2024-06-20	Принята
2	2	Петрова	Анна	Сергеевна	Прикладная математика	2024-06-21	Подана
3	3	Сидоров	Пётр	Алексеевич	Программная инженерия	2024-06-22	Подана
4	4	Иванов	Иван	Иванович	Прикладная математика	2024-06-23	Подана

(4 rows affected)

Completion time: 2024-11-21T00:31:16.8712579+03:00

Представление *View\_EntrantApplications*

Предназначена для отображения информации о заявках абитуриентов на образовательные программы, включая их экзаменационные баллы и ранжирование внутри каждой программы. Это позволяет быстро получить

сводные данные о том, как абитуриенты распределены по программам на основе их баллов.

```
CREATE VIEW MySchema.View_EntrantApplications AS
SELECT
    s.EntrantID,
    s.ApplicationID,
    s.ProgramName,
    s.Status,
    s.TotalScore,
    s.BudgetPlaces,
    DENSE_RANK() OVER (
        PARTITION BY s.EducationProgramID
        ORDER BY s.TotalScore DESC
    ) AS Ranking
FROM (
    SELECT
        e.EntrantID,
        a.ApplicationID,
        p.EducationProgramID,
        p.ProgramName,
        a.Status,
        p.BudgetPlaces,
        ISNULL(SUM(ex.ExamScore), 0) AS TotalScore
    FROM
        MySchema.Application a
    JOIN
        MySchema.Entrant e ON a.EntrantID = e.EntrantID
    JOIN
```

```

        MySchema.EducationProgram p ON
a.EducationProgramID = p.EducationProgramID
    LEFT JOIN
        MySchema.RequiredExams req ON req.ProgramID =
p.EducationProgramID
    LEFT JOIN
        MySchema.Exam ex ON ex.EntrantID = e.EntrantID AND
ex.Subject = req.Subject
    GROUP BY
        e.EntrantID, a.ApplicationID,
p.EducationProgramID, p.ProgramName, a.Status,
p.BudgetPlaces
) s;

```

Пример использования функции:

```

SELECT * FROM MySchema.View_EntrantApplications WHERE
EntrantID = 1

```

Результат запроса:

	EntrantID	ApplicationID	ProgramName	Status	TotalScore	BudgetPlaces	Ranking
1	100	95	Компьютерные науки	Подана	261	25	5
2	100	81	Электронная инженерия	Подана	261	20	2

(2 rows affected)

Completion time: 2024-11-28T02:14:45.4123194+03:00

## Пользовательские функции (UDF)

Функция *GetApplicationCount*

Возвращает количество заявок для конкретного абитуриента.

```
CREATE FUNCTION MySchema.GetApplicationCount(@EntrantID
INT)
RETURNS INT
AS
BEGIN
    DECLARE @Count INT;
    SELECT @Count = COUNT(*)
    FROM MySchema.Application
    WHERE EntrantID = @EntrantID;
    RETURN @Count;
END;
```

Пример использования функции:

```
SELECT MySchema.GetApplicationCount(1) AS
ApplicationCount;
```

Результат запроса:

ApplicationCount

-----

3

(1 row affected)

Completion time: 2024-11-21T00:33:32.3936496+03:00

### Функция *CheckExamRequirements*

Проверяет, соответствует ли абитуриент требованиям по экзаменам для выбранной программы.

```
CREATE FUNCTION MySchema.CheckExamRequirements(
    @EntrantID INT,
    @EducationProgramID INT)
RETURNS BIT
AS
BEGIN
    DECLARE @FailedSubjects INT;
    DECLARE @Result BIT;
    SET @FailedSubjects = (
        SELECT COUNT(*)
        FROM MySchema.RequiredExams req
        LEFT JOIN MySchema.Exam ex ON ex.EntrantID =
@EntrantID AND ex.Subject = req.Subject
        WHERE req.ProgramID = @EducationProgramID AND
(ex.ExamScore IS NULL OR ex.ExamScore < req.MinScore)
    );
    IF @FailedSubjects = 0
        SET @Result = 1;
    ELSE
        SET @Result = 0;
    RETURN @Result;
END;
```



## Хранимые процедуры.

Хранимая процедура *UpdateApplicationStatus*

Позволяет изменять статус заявки.

```
CREATE PROCEDURE MySchema.UpdateApplicationStatus
    @ApplicationID INT,
    @NewStatus NVARCHAR(50)
AS
BEGIN
    UPDATE MySchema.Application
    SET Status = @NewStatus
    WHERE ApplicationID = @ApplicationID;
END;
```

Пример вызова процедуры:

```
EXEC MySchema.UpdateApplicationStatus @ApplicationID = 1,
@NewStatus = 'Принята';
```

Результат запроса:

(1 row affected)

Completion time: 2024-11-21T00:32:11.6201600+03:00

Хранимая процедура *SubmitApplication*

Подача заявки с проверкой соответствия требованиям программы.

```
CREATE PROCEDURE MySchema.SubmitApplication
    @EntrantID INT,
```

```

    @EducationProgramID INT,
    @ApplicationDate DATE
AS
BEGIN
    -- Проверяем, есть ли уже поданная заявка
    IF EXISTS (
        SELECT 1 FROM Application
        WHERE EntrantID = @EntrantID AND
EducationProgramID = @EducationProgramID
    )
    BEGIN
        PRINT 'Заявка уже подана ранее.';
        RETURN;
    END

    -- Проверяем соответствие экзаменационным требованиям
    IF MySchema.CheckExamRequirements(@EntrantID,
@EducationProgramID) = 1
    BEGIN
        INSERT INTO MySchema.Application (EntrantID,
EducationProgramID, ApplicationDate, Status)
        VALUES (@EntrantID, @EducationProgramID,
@ApplicationDate, 'Подана');
        PRINT 'Заявка успешно подана.';
    END
    ELSE
    BEGIN

```

```
PRINT 'Заявка отклонена: не все требования по  
предметам выполнены.';
```

```
END
```

```
END;
```

Пример вызова процедуры:

```
EXEC MySchema.SubmitApplication @EntrantID = 1,  
@EducationProgramID = 2, @ApplicationDate = '2024-06-23';
```

Результат запроса:

(2 rows affected)

(1 row affected)

Заявка успешно подана.

Completion time: 2024-11-21T00:32:48.4072284+03:00

Хранимая процедура *GetProgramApplications*

Выводит заявки на конкретную образовательную программу,  
отсортированные по сумме баллов по требуемым экзаменам в порядке  
убывания.

```
CREATE PROCEDURE MySchema.GetProgramApplications
```

```
    @EducationProgramID INT
```

```
AS
```

```
BEGIN
```

```
    SELECT
```

```
        e.LastName,
```

```

        e.FirstName,
        e.MiddleName,
        p.ProgramName,
        ISNULL(SUM(ex.ExamScore), 0) AS TotalScore
FROM
    MySchema.Application a
JOIN
    MySchema.Entrant e ON a.EntrantID = e.EntrantID
JOIN
    MySchema.EducationProgram p ON
a.EducationProgramID = p.EducationProgramID
LEFT JOIN
    MySchema.RequiredExams req ON req.ProgramID =
p.EducationProgramID
LEFT JOIN
    MySchema.Exam ex ON ex.EntrantID = e.EntrantID AND
ex.Subject = req.Subject
WHERE
    p.EducationProgramID = @EducationProgramID
GROUP BY
    a.ApplicationID, e.EntrantID, e.LastName,
e.FirstName, e.MiddleName, p.ProgramName
ORDER BY
    TotalScore DESC;
END;

```

Пример вызова процедуры: EXEC MySchema.GetProgramApplications  
@EducationProgramID=1;

Результат запроса:

	LastName	FirstName	MiddleName	ProgramName	TotalScore
1	Фамилия33	Имя33	Отчество33	Программная инженерия	277
2	Фамилия81	Имя81	Отчество81	Программная инженерия	238
3	Фамилия57	Имя57	Отчество57	Программная инженерия	234
4	Фамилия11	Имя11	Отчество11	Программная инженерия	225
5	Иванов	Иван	Иванович	Программная инженерия	213
6	Попов	Сергей	Николаевич	Программная инженерия	203
7	Сидоров	Пётр	Алексеевич	Программная инженерия	186

(7 rows affected)

Completion time: 2024-11-27T20:12:23.2686138+03:00

Хранимая процедура *EnrollStudents*

Определяет и зачисляет n лучших абитуриентов на конкретную программу в зависимости от количества бюджетных мест.

```
CREATE PROCEDURE MySchema.EnrollStudents
    @EducationProgramID INT
AS
BEGIN
    DECLARE @BudgetPlaces INT;
    -- Получаем количество бюджетных мест
    SELECT @BudgetPlaces = BudgetPlaces
    FROM MySchema.EducationProgram
    WHERE EducationProgramID = @EducationProgramID;
    -- Обновляем статусы заявок
    WITH RankedApplicants AS (
        SELECT
            a.ApplicationID,
            e.EntrantID,
```

```

        ISNULL(SUM(ex.ExamScore), 0) AS TotalScore,
        ROW_NUMBER() OVER (ORDER BY
ISNULL(SUM(ex.ExamScore), 0) DESC) AS Ranking
FROM
    MySchema.Application a
JOIN
    MySchema.Entrant e ON a.EntrantID =
e.EntrantID
LEFT JOIN
    MySchema.RequiredExams req ON req.ProgramID =
a.EducationProgramID
LEFT JOIN
    MySchema.Exam ex ON ex.EntrantID = e.EntrantID
AND ex.Subject = req.Subject
WHERE
    a.EducationProgramID = @EducationProgramID
GROUP BY
    a.ApplicationID, e.EntrantID
)
UPDATE a
SET Status = CASE
    WHEN r.Ranking <= @BudgetPlaces THEN 'Зачислен'
    ELSE 'Отказано'
END
FROM MySchema.Application a
JOIN MySchema.RankedApplicants r ON a.ApplicationID =
r.ApplicationID
WHERE a.EducationProgramID = @EducationProgramID;

```

END;

Пример вызова процедуры:

```
EXEC MySchema.EnrollStudents @EducationProgramID = 1;
```

Результат запроса:

(7 rows affected)

Completion time: 2024-11-27T20:47:27.8318205+03:00

Хранимая процедура *GetSuitableProgramsForEntrant*

Создаёт рейтинг программ (список отсортирован начиная с самых требовательных по баллам программ), наиболее подходящих для абитуриента, учитывая его экзаменационные баллы и требования программ.

```
CREATE PROCEDURE MySchema.GetSuitableProgramsForEntrant
    @EntrantID INT
AS
BEGIN
    SET NOCOUNT ON;

    WITH SuitablePrograms AS (
        SELECT
            p.EducationProgramID,
            p.ProgramName,
            p.Faculty,
            p.EducationLevel,
```

```

        p.BudgetPlaces,
        SUM(req.MinScore) AS TotalMinScore
FROM
    MySchema.EducationProgram p
JOIN
    MySchema.RequiredExams req ON req.ProgramID =
p.EducationProgramID
LEFT JOIN
    MySchema.Exam e ON e.EntrantID = @EntrantID
AND e.Subject = req.Subject
GROUP BY
    p.EducationProgramID, p.ProgramName,
p.Faculty, p.EducationLevel, p.BudgetPlaces
HAVING
    COUNT(CASE WHEN e.ExamScore >= req.MinScore
THEN 1 END) = COUNT(req.Subject)
)
SELECT
    sp.EducationProgramID,
    sp.ProgramName,
    sp.Faculty,
    sp.EducationLevel,
    sp.BudgetPlaces,
    sp.TotalMinScore,
    DENSE_RANK() OVER (ORDER BY sp.TotalMinScore DESC)
AS Ranking
FROM
    MySchema.SuitablePrograms sp

```



```

ORDER BY
    sp.TotalMinScore DESC;
END;
GO

```

Пример вызова процедуры:

```

EXEC MySchema.GetSuitableProgramsForEntrant @EntrantID =
100;

```

Результат запроса:

	EducationProgramID	ProgramName	BudgetPlaces	TotalMinScore	Ranking
1	12	Робототехника	12	230	1
2	10	Информационная безопасность	15	225	2
3	3	Информационные системы	8	225	2
4	9	Электронная инженерия	20	200	3
5	11	Автоматизация и управление	18	200	3
6	1	Программная инженерия	3	195	4
7	8	Компьютерные науки	25	195	4

OP-0VRPS3O2 (15.0 RTM) | LAPTOP-0VRPS3O2\nikza ... | UniversityAdmission | 00:00:00 | 7 rows

Completion time: 2024-11-28T01:53:12.3516345+03:00

## Триггеры.

Триггер для предотвращения дублирования заявок

*trg\_PreventDuplicateApplications*

```
CREATE TRIGGER MySchema.trg_PreventDuplicateApplications
ON MySchema.Application
AFTER INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM MySchema.Application a
        JOIN inserted i ON a.EntrantID = i.EntrantID AND
a.EducationProgramID = i.EducationProgramID
    )
    BEGIN
        RAISERROR ('Duplicate application detected', 16,
1);
        ROLLBACK TRANSACTION;
    END
END;
```

## 7. Разработка стратегии резервного копирования

Для обеспечения сохранности данных необходимо регулярно выполнять резервное копирование базы данных. Рекомендуется использовать полное резервное копирование еженедельно и дифференциальное ежедневно, а также резервное копирование журнала транзакций каждые 6 часов.

**Пример выполнения полного резервного копирования:**

```
BACKUP DATABASE UniversityAdmission
TO DISK = 'C:\Backup\UniversityAdmission_Full.bak'
WITH FORMAT,
    INIT,
    NAME = 'Full Backup of UniversityAdmission',
    STATS = 10;
```

```
11 проц. обработано.
21 проц. обработано.
31 проц. обработано.
41 проц. обработано.
51 проц. обработано.
60 проц. обработано.
70 проц. обработано.
80 проц. обработано.
90 проц. обработано.
100 проц. обработано.
Обработано 496 страниц для базы данных "UniversityAdmission", файл "UniversityAdmission_Data" для файла 1.
Обработано 2 страниц для базы данных "UniversityAdmission", файл "UniversityAdmission_Log" для файла 1.
BACKUP DATABASE успешно обработал 498 страниц за 0.028 секунд (138.811 МБ/сек).

Completion time: 2024-12-05T01:17:40.1045525+03:00
```

**Пример выполнения дифференциального резервного копирования:**

```
BACKUP DATABASE UniversityAdmission
TO DISK = 'C:\Backup\UniversityAdmission_Diff.bak'
WITH DIFFERENTIAL,
    NAME = 'Differential Backup of UniversityAdmission',
    STATS = 10;
```

```
17 проц. обработано.
26 проц. обработано.
35 проц. обработано.
44 проц. обработано.
53 проц. обработано.
62 проц. обработано.
71 проц. обработано.
80 проц. обработано.
97 проц. обработано.
100 проц. обработано.
Обработано 104 страниц для базы данных "UniversityAdmission", файл "UniversityAdmission_Data" для файла 1.
Обработано 2 страниц для базы данных "UniversityAdmission", файл "UniversityAdmission_Log" для файла 1.
BACKUP DATABASE WITH DIFFERENTIAL успешно обработал 106 страниц за 0.008 секунд (103.027 МБ/сек).

Completion time: 2024-12-05T01:24:10.6897668+03:00
```

## Резервного копирования журнала транзакций

Для резервной копии журнала транзакций необходимо либо изменить модель восстановления базы данных на FULL или BULK\_LOGGED.

BACKUP LOG UniversityAdmission

TO DISK = 'C:\Backup\UniversityAdmission\_Log.bak'

WITH INIT,

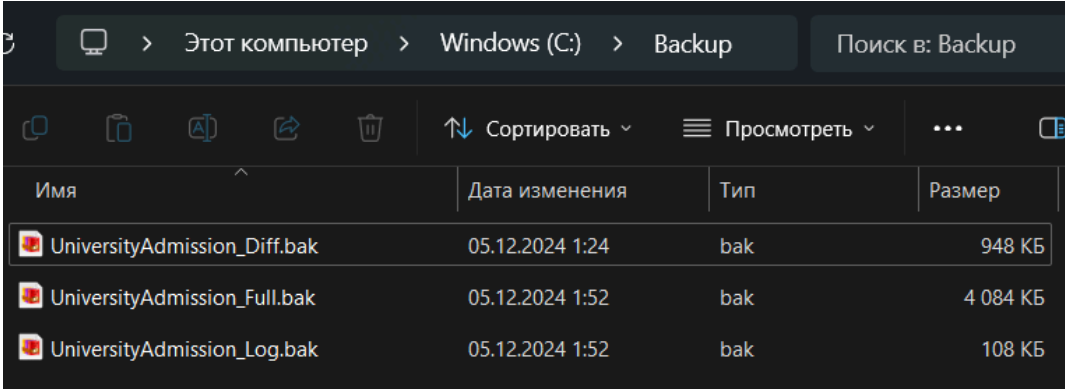
NAME = 'Transaction Log Backup of  
UniversityAdmission',

STATS = 10;

```
100 проц. обработано.
Обработано 3 страниц для базы данных "UniversityAdmission", файл "UniversityAdmission_Log" для файла 1.
BACKUP LOG успешно обработал 3 страниц за 0.002 секунд (11.718 МБ/сек).

Completion time: 2024-12-05T01:52:42.4672314+03:00
,
```

Результат создания файлов на диске:



The screenshot shows a Windows File Explorer window with the address bar set to 'C:\Backup'. The search bar contains 'Поиск в: Backup'. The file list shows three backup files:

Имя	Дата изменения	Тип	Размер
UniversityAdmission_Diff.bak	05.12.2024 1:24	bak	948 КБ
UniversityAdmission_Full.bak	05.12.2024 1:52	bak	4 084 КБ
UniversityAdmission_Log.bak	05.12.2024 1:52	bak	108 КБ

## **Заключение**

В результате выполнения индивидуального домашнего задания были закреплены теоретические знания и практические навыки, полученные в ходе изучения курса «Базы данных».

Была спроектирована и создана база данных «Прёмная комиссия», которая содержит информацию об абитуриентах, их заявках и результатах экзаменов, а также программах, и обязательных для поступления экзаменов. Также созданы таблицы с ограничениями целостности, они заполнены тестовыми данными, разработаны представления, хранимые процедуры и UDF, триггеры, разработана стратегия резервного копирования и созданы резервные копии базы данных.

## **Список использованных источников**

1. Горячев А. В., Новакова Н. Е. Распределенные базы данных. Мет. указания к лаб. работам., СПб. Изд-во СПбГЭТУ «ЛЭТИ», 2008
2. Горячев А. В, Новакова Н.Е. Особенности разработки и администрирования приложений баз данных: учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2016. 68 с.
3. Документация по Microsoft SQL:  
<https://learn.microsoft.com/en-us/sql/?view=sql-server-ver16>
4. Курс «базы данных» на учебном ресурсе moodle:  
<https://vec.etu.ru/moodle/course/view.php?id=19305>

### Дополнительная литература

5. Системы баз данных. Полный курс. / Гарсия-Молина Гектор, Ульман Джефери Д., Дженифер Уидом : пер. с англ., М.: Издательский дом «Вильямс», 2003, - 1088 с.