МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Базы данных»

**Тема: «База данных система отчетности для образовательного учреждения»**

**Исполнитель**

студент 2 курса 1 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. Р. Угоренко

подпись, дата

**Руководитель**

ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. Г. Савельева

должность, учен. степень, ученое звание подпись, дата

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. Г. Савельева

подпись дата инициалы и фамилия

**Содержание**

[Введение 5](#_Toc197306016)

[1 Постановка задачи 6](#_Toc197306017)

[1.1 Обзор аналогичных решений 6](#_Toc197306018)

[1.1.1 Moodle 6](#_Toc197306019)

[1.1.2 Blackboard 6](#_Toc197306020)

[1.1.3 Google Classroom 7](#_Toc197306021)

[1.1.4 Microsoft SQL Server Reporting Services (SSRS) 7](#_Toc197306022)

[1.2 Требования по проекту 7](#_Toc197306023)

[1.3 Вывод по разделу 8](#_Toc197306024)

[2 Проектирование базы данных 9](#_Toc197306025)

[2.1 Определение вариантов использования 9](#_Toc197306026)

[2.2 Диаграммы UML, взаимодействие всех компонентов 12](#_Toc197306027)

[2.3 Вывод по разделу 13](#_Toc197306028)

[3 Разработка объектов базы данных 14](#_Toc197306029)

[3.1 Разработка таблиц базы данных 14](#_Toc197306030)

[3.2 Разработка таблиц базы данных 15](#_Toc197306031)

[3.3 Разработка процедур базы данных 15](#_Toc197306032)

[3.4 Разработка функций базы данных 16](#_Toc197306033)

[3.5 Разработка представлений базы данных 16](#_Toc197306034)

[3.6 Разработка триггеров базы данных 17](#_Toc197306035)

[3.7 Роли и пользователи 17](#_Toc197306036)

[3.8 Вывод по разделу 17](#_Toc197306037)

[4 Описание процедур экспорта и импорта данных 18](#_Toc197306038)

[4.1 Процедура импорта данных из JSON-файла 18](#_Toc197306039)

[4.2 Процедура экспорта данных из JSON-файла 19](#_Toc197306040)

[4.3 Вывод по разделу 19](#_Toc197306041)

[5 Тестирование производительности 20](#_Toc197306042)

[5.1 Нагрузочное тестирование базы данных 20](#_Toc197306043)

[5.2 Вывод по разделу 21](#_Toc197306044)

[6 Описание технологии и её применение в базе данных 22](#_Toc197306045)

[6.1 Применение Reporting-технология СУБД 22](#_Toc197306046)

[6.2 Вывод по разделу 22](#_Toc197306047)

[Заключение 23](#_Toc197306048)

[Список использованных источников 24](#_Toc197306049)

[Приложение A 25](#_Toc197306050)

[Приложение Б 29](#_Toc197306051)

[Приложение В 30](#_Toc197306052)

[Приложение Г 31](#_Toc197306053)

[Приложение Д 32](#_Toc197306054)

[Приложение Е 33](#_Toc197306055)

# Введение

Современные образовательные учреждения сталкиваются с необходимостью эффективного управления большими объемами данных, что требует внедрения современных информационных систем. Актуальность разработки базы данных для системы отчетности обусловлена ростом требований к прозрачности, автоматизации и аналитике образовательного процесса. Существующие решения, такие как Moodle, Blackboard и Google Classroom, имеют ряд ограничений: сложность настройки, недостаточная поддержка аналитики или высокая стоимость внедрения. Поэтому создание собственной базы данных позволяет учесть индивидуальные потребности учреждения и повысить эффективность управления.

Цель курсового проекта — разработать базу данных для автоматизации учета успеваемости, управления данными студентов и преподавателей, а также формирования отчетности. Для достижения цели поставлены следующие задачи:

* Определение ролей: студент, преподаватель, администрация, отдел статистики.
* Общие действия для всех ролей: сортировка и фильтрация данных.
* Действия для студента: просмотр своей успеваемости, анализ оценок.
* Действия для преподавателя: просмотр списка студентов, добавление оценок, анализ среднего балла, отчеты по посещаемости.
* Действия для администрации: управление студентами и преподавателями, назначение ролей, отчеты по успеваемости, выявление студентов в зоне риска.
* Действия для отдела статистики: сводные отчеты, анализ динамики успеваемости, обработка данных за годы.

В качестве СУБД выбрана PostgreSQL, отличающаяся надежностью, расширяемостью и поддержкой сложных запросов. Анализ аналогов показал, что ролевой доступ, масштабируемость и поддержка аналитических функций наиболее эффективно реализуются на реляционной модели. В проекте предусмотрено создание таблиц, представлений, процедур и функций для реализации всех требований. Ожидается, что внедрение разработанной системы позволит оптимизировать процессы управления успеваемостью, повысить качество образовательного процесса и обеспечить прозрачность отчетности.

**1 Постановка задачи**

1.1 Обзор аналогичных решений

Для успешной реализации проекта важно изучить существующие аналоги и решения, используемые в образовательных учреждениях для управления отчетностью и успеваемостью студентов. Рассмотрим несколько распространенных систем с точки зрения их принципа работы, структуры базы данных, преимуществ и недостатков.

1.1.1 Moodle

Moodle — открытая система управления обучением с реляционной моделью данных и модульной архитектурой, где таблицы определяются через XML-файлы [1].

Основные таблицы:

* user — данные пользователей;
* course — курсы и категории;
* role — роли (студент, преподаватель);
* context — контексты доступа;
* grade\_items и grade\_grades — элементы оценивания и оценки студентов.

Эта структура обеспечивает гибкость управления учебным процессом и отчетностью. Использует MySQL/MariaDB. Преимущества: гибкая структура, масштабируемые отчеты, импорт/экспорт данных. Недостатки: сложность настройки при больших нагрузках, ограниченная аналитика и визуализация, высокая нагрузка на сервер.

1.1.2 Blackboard

Blackboard — платформа дистанционного обучения с централизованным хранилищем данных и архитектурой «черной доски» [2].

Основные таблицы:

* users — информация о пользователях;
* courses — курсы;
* enrollments — связи пользователей с курсами;
* assignments — задания;
* grades — оценки;
* content — учебные материалы.

Структура позволяет эффективно управлять учебным контентом и успеваемостью.

Преимущества: параллельная обработка данных, поддержка версионирования и контроля доступа.

Недостатки: жесткая связь между хранилищем и модулями, сложности синхронизации при распределенной нагрузке.

1.1.3 Google Classroom

Google Classroom — облачная платформа, интегрированная с Google Drive, использующая NoSQL-подобные структуры (документы, таблицы) [3].

Основные сущности:

* Users — пользователи;
* Courses — курсы;
* Assignments — задания;
* Submissions — сдачи заданий;
* Announcements — объявления.

Несмотря на NoSQL-структуру, эти сущности отражают ключевые элементы управления учебным процессом. В реляционной базе их можно реализовать для структурированного хранения.

Преимущества: простота интеграции с Google-сервисами, совместное редактирование в реальном времени.

Недостатки: отсутствие реляционной структуры усложняет аналитику, нет контроля версий на уровне СУБД.

1.1.4 Microsoft SQL Server Reporting Services (SSRS)

SSRS — инструмент для создания и управления отчетами, интегрированный с Microsoft SQL Server [4].

Основные таблицы:

* Catalog — информация об отчетах и источниках данных
* DataSource — источники данных;
* Subscriptions — подписки на отчеты;
* Users — пользователи с доступом;
* Roles — роли пользователей.

Эти таблицы обеспечивают управление отчетами, метаданными и подписками, что важно для аналитики и отчетности. Использует RDL-файлы (XML-описания отчетов).

Преимущества: поддержка сложных отчетов и визуализаций, интеграция с BI (Power BI), экспорт в различные форматы.

Недостатки: зависимость от лицензированного SQL Server, сложность администрирования, ограниченная гибкость интеграции.

1.2 Требования по проекту

Для реализации проекта по созданию базы данных для данной темы необходимо учитывать следующие требования:

Определение ролей и прав доступа: Система должна поддерживать четыре основные роли: студенты, преподаватели, администрация и отдел статистики. Функциональные требования для ролей:

* Для студентов: возможность просмотра своей успеваемости и оценок.
* Для преподавателей: доступ к списку студентов, возможность добавления оценок и анализ среднего балла.
* Для администрации: управление данными о студентах и преподавателях, назначение ролей, формирование отчетов по успеваемости и выявление студентов, находящихся в зоне риска.
* Для отдела статистики: создание сводных отчетов, анализ динамики успеваемости и обработка данных за несколько лет.

Требования к базе данных: База данных должна быть спроектирована в PostgreSQL. Данная СУБД выбрана благодаря надежности, расширяемости, соответствию стандартам SQL и поддержке сложных запросов. Поддержка аналитических расширений и работы с JSON снижают затраты и обеспечивают гибкость настройки под нужды образовательного учреждения. Необходимо создать таблицы, представления, последовательности, материализованные представления, процедуры, функции и триггеры для реализации функциональных требований.

Требования к импорту и экспорту данных: Система должна поддерживать импорт данных из JSON файлов и экспорт данных в формат JSON. Процедуры импорта и экспорта должны быть документированы и протестированы.

Требования к производительности: База данных должна быть протестирована на производительность с использованием набора данных, содержащего не менее 100 000 строк для одной из таблиц.

Требования к отчетности: в системе должны быть реализованы Reporting-технологии СУБД для создания отчетов и анализа данных. Выбор данной технологии обусловлен тесной интеграцией с базой данных, что упрощает создание отчетов благодаря прямому доступу к данным через SQL и встроенные функции. Это обеспечивает высокую производительность и удобные средства визуализации, а также снижает требования к дополнительному ПО и упрощает администрирование.

1.3 Вывод по разделу

Анализ существующих решений выявил необходимость гибкой и масштабируемой системы. Реляционная модель базы данных, такая как PostgreSQL оптимальна для обеспечения целостности и безопасности данных. Разработка базы данных на PostgreSQL, учитывающей специфику образовательного учреждения, позволит создать эффективную систему отчетности с высоким уровнем управления данными, поддержкой ролей и инструментами анализа.

**2 Проектирование базы данных**

2.1 Определение вариантов использования

В данном разделе определяются возможные сценарии использования системы с учетом ролей пользователей и их функциональных обязанностей. Для наглядного представления вариантов использования разработаны диаграммы UML.

Студенты используют систему для просмотра своей успеваемости и анализа полученных оценок. Диаграмма вариантов использования для данной роли должна отражать эти основные функции.

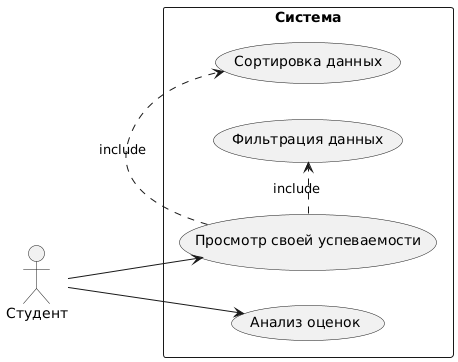


Рисунок 2.1 – Диаграмма вариантов использования для роли «Студент»

Описание прецедентов использования для роли «Студент».

Студент: Просмотр своей успеваемости. Основной поток: студент входит в систему, переходит в раздел «Успеваемость», и отображается список предметов, оценок и средний балл. Альтернативный поток: у студента нет оценок, и система отображает сообщение об отсутствии данных.

Студент: Анализ оценок. Основной поток: студент выбирает предмет, и система отображает график изменения оценок. Альтернативный поток: по предмету недостаточно оценок для построения графика, и система отображает сообщение об этом.

Преподаватели используют систему для управления списком студентов, добавления оценок, анализа успеваемости и формирования отчетов по посещаемости.

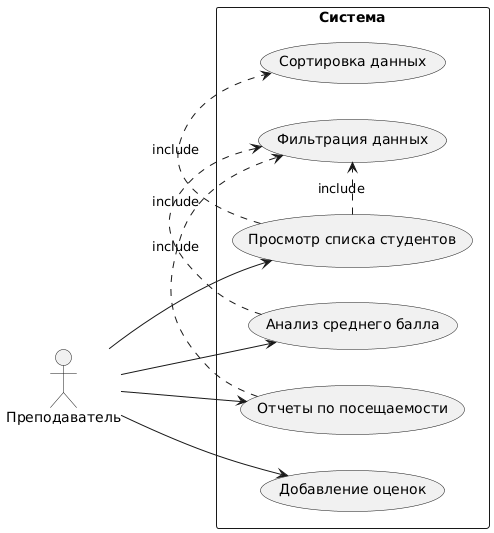


Рисунок 2.2 – Диаграмма вариантов использования для роли «Преподаватель»

Описание прецедентов использования для роли «Преподаватель».

Преподаватель: Просмотр списка студентов. Основной поток: преподаватель входит в систему, выбирает предмет. Отображается список студентов, записанных на этот предмет. Альтернативный поток: на предмет не записано ни одного студента. Система отображает сообщение об отсутствии данных.

Преподаватель: Добавление оценок. Основной поток: преподаватель выбирает студента и предмет, вводит оценку. Система сохраняет оценку. Альтернативный поток: введенная оценка не соответствует допустимому диапазону. Система отображает сообщение об ошибке.

Преподаватель: Анализ среднего балла. Основной поток: преподаватель выбирает предмет. Система отображает средний балл. Альтернативный поток: по предмету нет оценок. Система отображает сообщение об отсутствии данных.

Преподаватель: Отчеты по посещаемости. Основной поток: преподаватель выбирает предмет и период. Система формирует отчет по посещаемости. Альтернативный поток: данные о посещаемости отсутствуют. Система отображает сообщение об этом.

Администраторы используют систему для управления данными о студентах и преподавателях, назначения ролей и т.д.

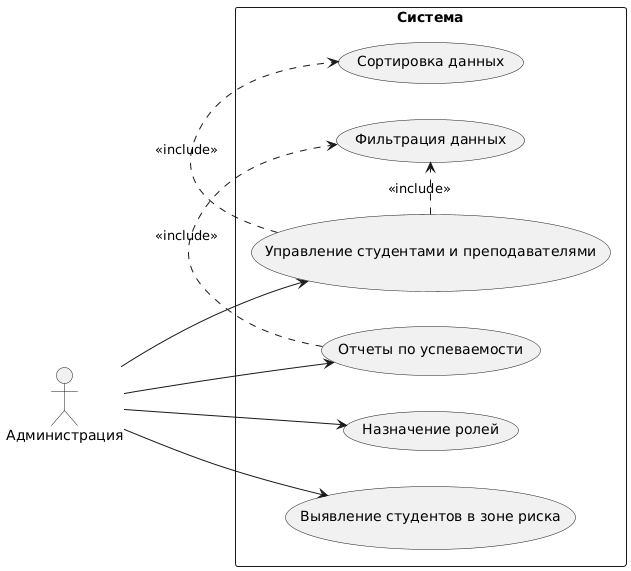


Рисунок 2.3 – Диаграмма вариантов использования для роли «Администрация»

Описание прецедентов использования для роли «Администрация».

Администрация: Управление студентами и преподавателями. Основной поток: администратор манипулирует данными. Альтернативный поток: введенные данные не соответствуют требованиям. Система отображает сообщение об ошибке.

Администрация: Назначение ролей. Основной поток: администратор назначает или изменяет роли пользователей. Альтернативный поток: пользователь с указанным ID не найден. Система отображает сообщение об ошибке.

Администрация: Отчеты по успеваемости. Основной поток: администратор формирует отчеты по критериям (факультет, курс). Альтернативный поток: данные для отчета отсутствуют. Система отображает сообщение об этом.

Администрация: Выявление студентов в зоне риска. Основной поток: система анализирует успеваемость и выявляет студентов в зоне риска. Администратор получает список. Альтернативный поток: в зоне риска нет студентов. Система отображает соответствующее сообщение.

Отдел статистики использует систему для формирования сводных отчетов, анализа динамики успеваемости и обработки данных за годы.

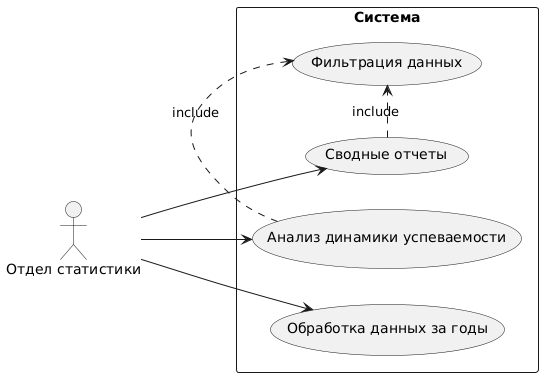


Рисунок 2.4 – Диаграмма вариантов использования для роли «Отдел статистики»

Описание прецедентов использования для роли «Отдел статистики».

Отдел статистики: Сводные отчеты. Основной поток: отдел статистики формирует сводные отчеты. Альтернативный поток: данные для отчета отсутствуют. Система отображает сообщение об этом.

Отдел статистики: Анализ динамики успеваемости. Основной поток: система анализирует динамику успеваемости. Альтернативный поток: данных для анализа недостаточно. Система отображает соответствующее сообщение.

Отдел статистики: Обработка данных за годы. Основной поток: отдел статистики обрабатывает данные за предыдущие годы. Альтернативный поток: данные за указанный период отсутствуют. Система отображает сообщение об этом.

Этот раздел предоставляет подробное описание вариантов использования системы, а также диаграммы UML для визуализации этих вариантов для каждой роли пользователя.

2.2 Диаграммы UML, взаимодействие всех компонентов

Логическая схема базы данных – это абстрактное представление структуры данных, определяющее, как данные будут организованы и связаны между собой, без учета физической реализации предоставлена в приложении А. Она описывает таблицы, столбцы, типы данных и отношения между таблицами. В данном случае логическая схема включает таблицы, необходимые для системы отчетности образовательного учреждения: Faculties, Accounts, StudyGroups, Students, Teachers, Employees, Subjects, Grades, Attendance, Reports. Каждая таблица содержит определенные поля, например, таблица Faculties включает faculty\_id (первичный ключ), faculty\_name и description. Связи между таблицами устанавливаются с помощью внешних ключей, например, поле faculty\_id в таблице StudyGroups является внешним ключом, ссылающимся на таблицу Faculties. Определение логической схемы – важный шаг в проектировании базы данных, обеспечивающий эффективное хранение, управление и извлечение данных.

Таблица 2.1 – Сведения о таблицах базы данных

|  |  |
| --- | --- |
| Имя таблицы | Назначение таблицы |
| Accounts | Хранение информации о всех аккаунтах. |
| Students | Хранение информации о студентах. |
| Employees | Хранение информации о сотрудниках. |
| Teachers | Хранение информации о преподавателях. |
| Faculties | Хранение информации о факультетах. |
| Subjects | Хранение информации о предметах. |
| StudyGroups | Хранение информации о группах. |
| Grades | Хранение информации об оценках студентов по предметам. |
| Attendance | Хранение информации о посещаемости студентов. |
| Reports | Хранение данных для формирования отчетов. |

Данная таблица определяет структуру базы данных, где каждая таблица предназначена для хранения информации об аккаунтах, студентах, сотрудниках, преподавателях, факультетах, предметах и т. д. обеспечивая комплексное управление данными образовательного учреждения.

2.3 Вывод по разделу

В результате проектирования базы данных была создана структура, которая соответствует требованиям системы отчетности образовательного учреждения. Определены основные сущности (студенты, преподаватели, предметы, оценки и т.д.) и связи между ними. Разработаны варианты использования для каждой роли пользователя.

**3 Разработка объектов базы данных**

3.1 Разработка схемы базы данных

Перед оформлением физической базы данных необходимо сформировать точное количество таблиц базы данных и определить необходимые поля, типы данных для этих полей и связи между этими таблицами. Подробное описание всех таблиц предоставлено в приложении А.

Таблица 3.1 – Сведения о связях между таблицами базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1 | Связь | Таблица 2 | Описание связи |
| Students | 1:1 | Accounts | Each student has one record in the Accounts table. |
| Teachers | 1:1 | Accounts | У каждого преподавателя есть одна запись в таблице Accounts. |
| Employees | 1:1 | Accounts | У каждого сотрудника есть одна запись в таблице Accounts. |
| Students | M:1 | StudyGroups | В одной группе может быть много студентов. |
| StudyGroups | M:1 | Faculties | К одному факультету может принадлежать много групп. |
| Subjects | M:1 | Faculties | К одному факультету может принадлежать много предметов. |
| Employees | M:1 | Faculties | В одном факультете может работать много сотрудников. |
| Grades | M:1 | Students | Одному студенту может быть выставлено много оценок. |
| Grades | M:1 | Subjects | По одному предмету может быть выставлено много оценок. |
| Grades | M:1 | Teachers | Одному преподавателю может быть выставлено много оценок. |
| Attendance | M:1 | Students | У одного студента может быть много записей о посещаемости. |
| Attendance | M:1 | Subjects | К одному предмету может относиться много записей о посещаемости. |

Данные таблицы формируют основу базы данных системы отчетности для образовательного учреждения, обеспечивая хранение и обработку информации об успеваемости, посещаемости и пользователях системы.

3.2 Разработка таблиц базы данных

В данном разделе будет рассмотрена разработка таблиц базы данных. Всего было создано 10 таблиц. В листинге 3.1 представлен SQL-запрос для создания одной из них.

CREATE TABLE Subjects (

subject\_id SERIAL PRIMARY KEY,

subject\_name VARCHAR NOT NULL,

hours INT NOT NULL,

form\_of\_assessment VARCHAR NOT NULL,

faculty\_id INT NOT NULL,

CONSTRAINT fk\_subjects\_faculty FOREIGN KEY (faculty\_id) REFERENCES Faculties(faculty\_id) ON DELETE CASCADE

);

Листинг 3.1 – SQL-запрос для создания

таблицы Subjects

Другие примеры SQL-запросов для создания таблиц представлены в приложении Б.

3.3 Разработка процедур базы данных

В рамках данного проекта были разработаны хранимые процедуры для управления формированием и обработкой отчетности, а также для автоматизации ключевых процессов образовательного учреждения.

Процедура для генерации индивидуального отчета по успеваемости студента представлена в листинге 3.2

CREATE PROCEDURE proc\_add\_grade(

p\_student\_id INT, p\_subject\_id INT, p\_teacher\_id INT, p\_grade INT, p\_date DATE)

LANGUAGE plpgsql AS $$

BEGIN

INSERT INTO Grades(student\_id,subject\_id,teacher\_id,grade,date)

VALUES(p\_student\_id,p\_subject\_id,p\_teacher\_id,p\_grade,p\_date);

END; $$;

Листинг 3.2 – SQL-запрос для создания процедуры для добавления оценки

Также реализованы процедуры для формирования отчетов по успеваемости, посещаемости и другие. Все остальные процедуры, созданные в рамках данного курсового проекта представлены в приложении В.

3.4 Разработка функций базы данных

Функции в PostgreSQL используются для получения агрегированных и аналитических данных, а также для фильтрации информации по различным критериям. В рамках данного курсового проекта были реализованы функции, позволяющие получать статистику по студентам, преподавателям, предметам, а также анализировать посещаемость и успеваемость. Пример создания одной из таких функций приведён ниже.

CREATE OR REPLACE FUNCTION fn\_get\_student\_grades(p\_student INT)

RETURNS TABLE(subject VARCHAR, grade INT, date DATE) LANGUAGE sql AS $$

SELECT sub.subject\_name, g.grade, g.date

FROM Grades g

JOIN Subjects sub ON g.subject\_id=sub.subject\_id

WHERE g.student\_id = p\_student

ORDER BY g.date;

$$;

Листинг 3.3 – SQL-запрос для создания функции получения всех оценок студента

Также реализованы процедуры для формирования отчетов по успеваемости, посещаемости, а также анализа динамики оценок. Все остальные процедуры, разработанные в рамках данного курсового проекта, представлены в приложении Г.

3.5 Разработка представлений базы данных

Представления позволяют упрощать доступ к данным, формируя удобные виртуальные таблицы для анализа и отчетов. В рамках данного курсового проекта реализовано представление для выявления студентов, находящихся в зоне риска по результатам успеваемости и посещаемости. SQL-запрос для создания приведён в листинге 3.4

CREATE OR REPLACE VIEW view\_students\_risk AS

SELECT s.student\_id, a.last\_name, a.first\_name, AVG(g.grade)::NUMERIC AS avg\_grade

FROM Grades g

JOIN Students s ON g.student\_id=s.student\_id

JOIN Accounts a ON s.account\_id=a.account\_id

GROUP BY s.student\_id,a.last\_name,a.first\_name

HAVING AVG(g.grade)<6;

Листинг 3.4 – SQL-запрос для создания представления «Студенты в зоне риска»

SQL-запросы для создания остальных представлений приведены в приложении Д.

3.6 Разработка триггеров базы данных

Триггеры в базе данных позволяют автоматизировать выполнение действий при изменении данных, обеспечивая целостность и корректность информации. В рамках проекта были разработаны следующие триггеры;

* Триггер trg\_check\_grade\_range гарантирует соответствие оценок установленному диапазону (1–10), предотвращая ввод некорректных значений;
* Триггер trg\_check\_grade\_on\_class\_day контролирует временную согласованность данных, разрешая выставление оценок только в день проведения занятий;
* Триггер trg\_teacher\_access реализует систему разграничения доступа, позволяя преподавателям работать исключительно с собственными предметами.

Такая система триггеров способствует поддержанию высокого уровня достоверности данных, автоматизации ключевых процессов и облегчает контроль за учебной деятельностью в образовательном учреждении.

3.7 Роли и пользователи

В рамках проекта для обеспечения разграничения доступа и реализации системы прав пользователей были созданы четыре основные роли:

* student\_role- роль для студентов,
* teacher\_role- роль для преподавателей,
* admin\_role- роль для административного персонала,
* analytics\_role- роль для сотрудников отдела статистики.

Для каждой из этих ролей созданы тестовые пользователи, которые позволяют проверить корректность настройки прав и функционала системы. Примеры пользователей:

* Для роли student: student\_user;
* Для роли teacher: teacher\_user;
* Для роли administration: admin\_user;
* Для роли statistics\_department: analytics\_user.

Листинг SQL-запросов по созданию ролей и пользователей приведён в приложении Е.

3.8 Вывод по разделу

В данном разделе были разработаны основные объекты базы данных, включая таблицы, представления и триггеры, обеспечивающие функциональность системы отчетности образовательного учреждения. Особое внимание уделено реализации триггеров, которые автоматизируют контроль корректности данных и поддерживают целостность учебного процесса.

**4 Описание процедур экспорта и импорта данных**

В данном разделе представлены процедуры для экспорта и импорта данных между базой данных образовательного учреждения и файлами в формате JSON. Использование формата JSON обеспечивает удобство обмена данными, интеграцию с внешними системами и возможность резервного копирования информации. Процедуры реализованы средствами PostgreSQL и позволяют автоматизировать процесс переноса данных.

4.1 Процедура импорта данных из JSON-файла

Для загрузки данных из JSON-файла в таблицу (например, Students) реализована процедура, которая читает содержимое файла, преобразует его в объекты и вставляет их в таблицу. При необходимости существующие записи обновляются.

CREATE OR REPLACE PROCEDURE proc\_import\_students(p\_path text)

LANGUAGE plpgsql AS $$

DECLARE

json\_data json;

student\_record json;

BEGIN

json\_data := pg\_read\_file(p\_path)::json;

INSERT INTO Accounts (...) -- поля таблицы

SELECT ... FROM json\_array\_elements(json\_data) AS account\_data -- преобразование полей

ON CONFLICT (account\_id) DO UPDATE SET ...; -- обновляемые поля

FOR student\_record IN SELECT \* FROM json\_array\_elements(json\_data) LOOP

IF student\_record->>'student\_id' IS NULL OR ... THEN

RAISE WARNING 'Пропущена запись: %', student\_record;

CONTINUE;

END IF;

INSERT INTO Students (...) VALUES (...) -- поля и значения

ON CONFLICT (student\_id) DO UPDATE SET ...; -- обновляемые поля

END LOOP;

END;

$$;

Листинг 4.1 – Процедура импорта данных из JSON-файла

Данная процедура позволяет быстро загружать данные из внешних источников, обеспечивая актуальность информации в базе данных. Это особенно удобно при первичном наполнении системы или интеграции с другими образовательными платформами.

4.2 Процедура экспорта данных из JSON-файла

Для выгрузки данных из таблицы в JSON-файл реализована процедура, формирующая JSON-массив на основе содержимого таблицы и сохраняющая его в указанный файл.

CREATE OR REPLACE PROCEDURE proc\_export\_students(p\_path TEXT)

LANGUAGE plpgsql AS $$

DECLARE

js TEXT;

BEGIN

SELECT json\_agg(json\_build\_object(...))::TEXT INTO js -- поля для экспорта

FROM Students s JOIN Accounts a ON s.account\_id = a.account\_id;

EXECUTE format('COPY (SELECT %L) TO %L', js, p\_path);

END;

$$;

Листинг 4.2 – Процедура экспорта данных в JSON-файл

Экспорт данных в формате JSON позволяет создавать резервные копии, а также обмениваться информацией с внешними системами. Такой подход облегчает перенос данных между разными экземплярами базы данных и интеграцию с аналитическими инструментами.

4.3 Вывод по разделу

Реализация процедур экспорта и импорта данных в формате JSON обеспечивает гибкость интеграции базы данных с внешними системами и автоматизацию процессов резервного копирования и восстановления. Использование стандартных средств PostgreSQL позволяет упростить администрирование и повысить надежность работы с большими объемами данных. Данные процедуры могут быть адаптированы для любых таблиц проекта, что делает систему универсальной и удобной в эксплуатации.

**5 Тестирование производительности**

Тестирование производительности базы данных является важным этапом проверки её эффективности. В рамках данного проекта проведено тестирование скорости выполнения запросов к таблице Grades, чтобы оценить влияние индексации на производительность при работе с большими объёмами данных.

5.1 Нагрузочное тестирование базы данных

Для имитации реальной нагрузки в таблицу Grades было добавлено 100 000 новых оценок, учитывая, что изначально в таблице уже содержалось 60 записей. В качестве идентификаторов использовались существующие студенты (student\_id от 1 до 10), предметы (subject\_id от 1 до 10) и преподаватели (teacher\_id от 11 до 20). Для каждой новой оценки случайным образом выбирались значения из допустимого диапазона.

DO $$

DECLARE

i INT := 1;

BEGIN

WHILE i <= 100000 LOOP

INSERT INTO Grades (student\_id, subject\_id, teacher\_id, grade, date)

VALUES (

(random() \* 9 + 1)::INT,

(random() \* 9 + 1)::INT,

(random() \* 9 + 11)::INT,

(random() \* 9 + 1)::INT,

(CURRENT\_DATE - (random() \* 365)::INT)

);

i := i + 1;

END LOOP;

END $$;

Листинг 5.1 – SQL-запрос для массового заполнения таблицы Grades

После заполнения таблицы была проведена фильтрация оценок по определённому студенту и предмету. На листинге 5.2 продемонстрирован запрос.

EXPLAIN ANALYZE

SELECT \* FROM Grades WHERE student\_id = 5 AND subject\_id = 3;

Листинг 5.2 – SQL-запрос для выполнения фильтрации без индекса

На рисунке 5.1 показан пример результата выполнения запроса без индекса:

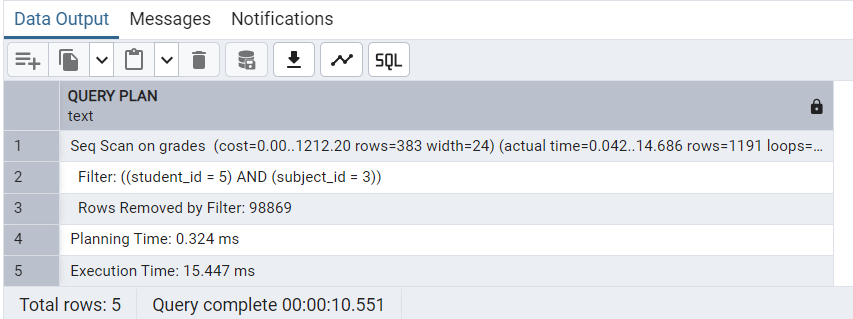


Рисунок 5.1 – Выполнение запроса без индекса

Время выполнения запроса составило 15 миллисекунд, при этом база данных выполнила полное сканирование таблицы Grades.

Далее был создан составной индекс для ускорения поиска:

CREATE INDEX idx\_grades\_student\_subject ON Grades(student\_id, subject\_id);

Листинг 5.3 – SQL-запрос для создания индекса

Повторное выполнение того же запроса показало значительное снижение времени выполнения – до 3 миллисекунд, так как теперь используется индексированная структура поиска (см. рисунок 5.2).

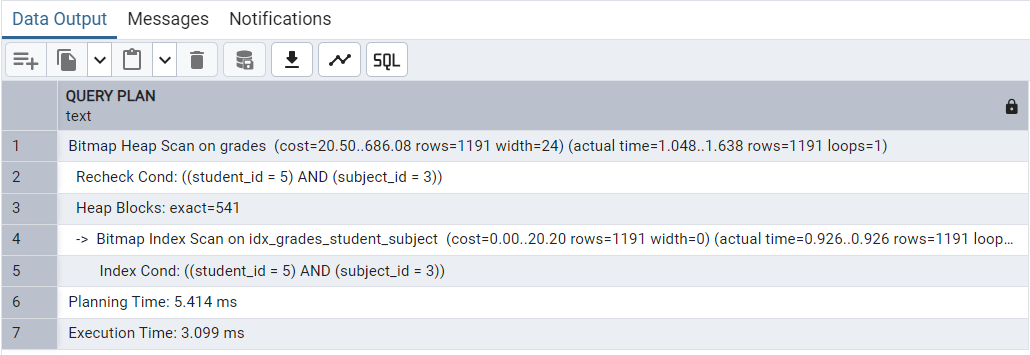


Рисунок 5.2 – Выполнение запроса с индексом

Таким образом, использование индекса существенно повышает скорость выборки данных даже при большом объёме информации.

5.2 Вывод по разделу

Результаты тестирования показывают, что индексирование значительно улучшает производительность поиска данных. Без индекса PostrgeSQL просматривает всю таблицу, тогда как индекс позволяет находить нужные записи быстрее, снижая нагрузку на систему.

**6 Описание технологии и её применение в базе данных**

В данном разделе рассматриваются возможности Reporting-технологий современных СУБД и описывается необходимость их применение в рамках проекта системы отчетности для образовательного учреждения.

6.1 Применение Reporting-технология СУБД

Reporting-технологии СУБД – это встроенные средства для создания, автоматизации и визуализации отчетов непосредственно на уровне базы данных. Они позволяют формировать различные виды отчетности (например, сводные таблицы, аналитические выборки, динамические представления), используя стандартные SQL-запросы, представления (VIEW), материализованные представления и процедуры.

В проекте на базе PostgreSQL Reporting-технологии используются для решения следующих задач:

* Автоматизация отчетности: Благодаря триггерам, например, log\_attendance\_changes, формирование отчетов по успеваемости и посещаемости студентов происходит автоматически при добавлении новых данных.
* Аналитика и мониторинг: Представления, такие как view\_students\_risk, позволяют администраторам и отделу статистики оперативно выявлять студентов с низкой успеваемостью или высокой пропускной активностью.
* Гибкость доступа: Различные роли пользователей (студент, преподаватель, администрация, отдел статистики) получают доступ к релевантным отчетам и аналитическим данным через специально созданные представления и процедуры.

Таким образом, Reporting-технологии СУБД в проекте позволяют получать актуальные и точные отчеты в режиме реального времени, автоматизировать процессы анализа и контроля данных и снижать нагрузку на пользователей за счет автоматизации формирования отчетности.

6.2 Вывод по разделу

Reporting-технологии СУБД значительно повышают эффективность работы системы отчетности, обеспечивая своевременное получение аналитических данных для всех ролей пользователей. Их применение в проекте позволило автоматизировать формирование отчетов, повысить прозрачность образовательного процесса и упростить анализ больших объемов информации.

Заключение

Разработанная база данных системы отчетности для образовательного учреждения успешно решает задачи, поставленные в начале проекта, обеспечивая эффективное управление информацией об успеваемости, посещаемости и учебных процессах. В ходе работы была реализована многоуровневая система ролей и прав доступа, что гарантирует безопасность данных и разграничение функционала для различных категорий пользователей – студентов, преподавателей, администрации и отдела статистики. Использование возможностей СУБД PostgreSQL, таких как триггеры, представления и процедуры, позволило автоматизировать ключевые бизнес-процессы и повысить надежность работы системы. Проведённое тестирование подтвердило высокую производительность и стабильность базы данных при работе с большими объемами информации, что обеспечивает масштабируемость и адаптивность решения под нужды образовательного учреждения. В целом, созданная система способствует повышению прозрачности учебного процесса, облегчает формирование аналитической отчетности и оптимизирует управление образовательной деятельностью.

Список использованных источников

1. Аналог «Moodle» [Электронный ресурс]. – Режим доступа: https://moodle.org – Дата доступа: 28.03.2025.

2. Аналог «Blackboard» [Электронный ресурс]. – Режим доступа: https://www.anthology.com/products/teaching-and-learning/learning-effectiveness/blackboard– Дата доступа: 28.03.2025.

3. Аналог «Google Classroom» [Электронный ресурс]. – Режим доступа: https://classroom.google.com/?pli=1 Дата доступа: 28.03.2025.

4. Аналог «SSRS» [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/sql/reporting-services/create-deploy-and-manage-mobile-and-paginated-reports?view=sql-server-ver16> – Дата доступа: 28.03.2025.

5. Сайт для разработки диаграмм «PlantText» [Электронный ресурс]. – Режим доступа: https://www.planttext.com

Приложение A

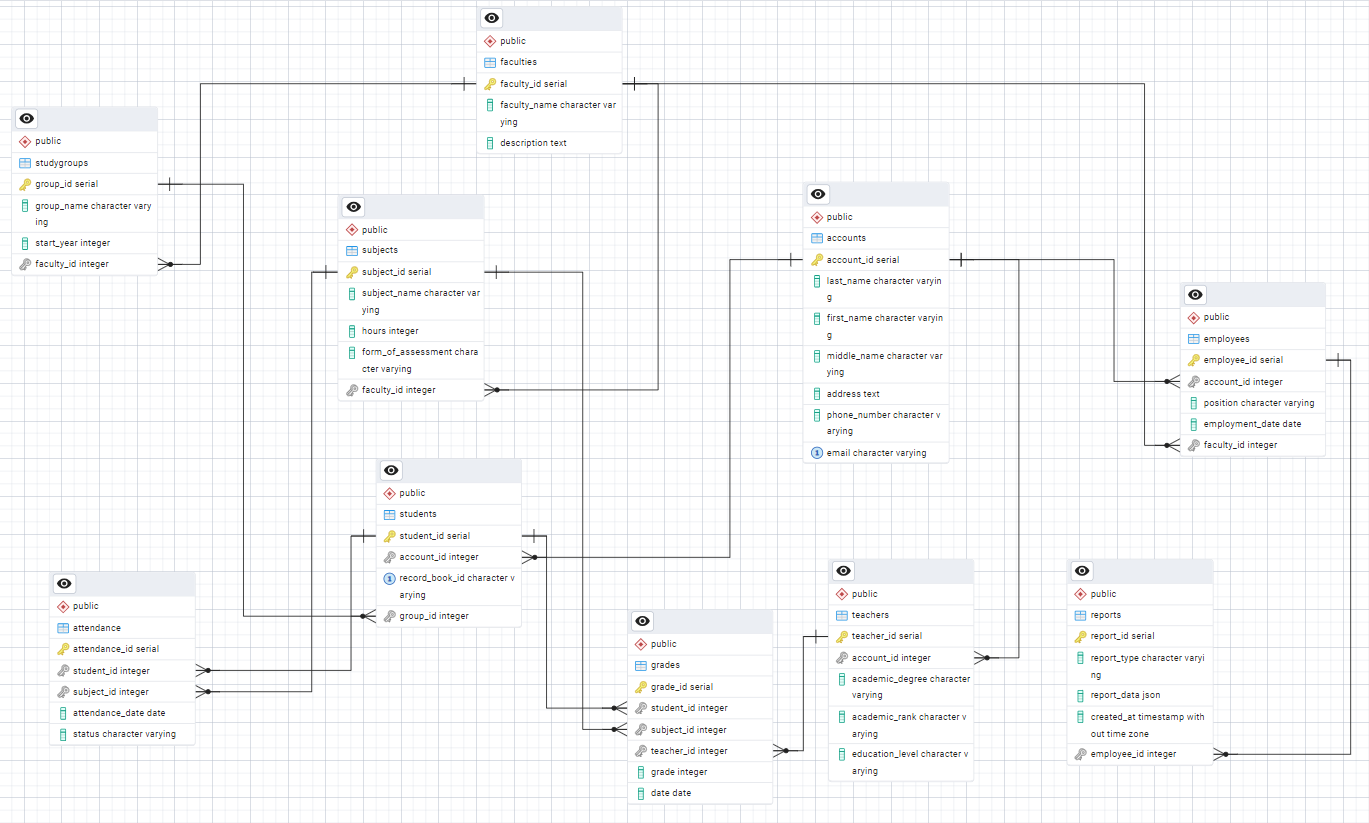


Рисунок А.1 – Логическая схема базы данных

Таблица A.1 – Таблица «Аккаунты» (Accounts)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| person\_id | SERIAL PRIMARY KEY | Уникальный идентификатор личности. |
| last\_name | VARCHAR(255) NOT NULL | Фамилия. |
| first\_name | VARCHAR(255) NOT NULL | Имя. |
| middle\_name | VARCHAR(255) | Отчество. |
| address | VARCHAR(255) | Адрес проживания. |
| phone\_number | VARCHAR(20) | Номер телефона. |
| email | VARCHAR(255) UNIQUE | Электронная почта |

Таблица A.2 – Таблица «Студенты» (Students)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| student\_id | SERIAL PRIMARY KEY | Уникальный идентификатор студента |
| account\_id | INTEGER REFERENCES Accounts(account\_id) | Ссылка на таблицу Accounts |
| record\_book\_id | VARCHAR(20) UNIQUE | Номер зачетной книжки |
| group\_id | INTEGER REFERENCES StudyGroups (group\_id) | Ссылка на таблицу StudyGroups |

Таблица A.3 – Таблица «Сотрудники» (Employees)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| employee\_id | SERIAL PRIMARY KEY | Уникальный идентификатор сотрудника |
| account\_id | INTEGER REFERENCES Accounts(account\_id) | Ссылка на таблицу Accounts |
| position | VARCHAR(255) | Должность |
| employment\_date | DATE | Дата приема на работу |
| faculty\_id | INTEGER REFERENCES Faculties(faculty\_id) | Ссылка на таблицу Faculties |

Таблица A.4 – Таблица «Преподаватели» (Teachers)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| teacher\_id | SERIAL PRIMARY KEY | Уникальный идентификатор преподавателя |
| account\_id | INTEGER REFERENCES Accounts(account\_id) | Ссылка на таблицу Accounts |
| academic\_degree | VARCHAR(255) | Ученая степень |
| academic\_rank | VARCHAR(255) | Ученое звание |
| education\_level | VARCHAR(255) | Уровень образования |

Таблица A.5 – Таблица «Факультеты» (Faculties)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| faculty\_id | SERIAL PRIMARY KEY | Уникальный идентификатор факультета |
| faculty\_name | VARCHAR(255) NOT NULL | Название факультета |
| description | VARCHAR(255) | Описание факультета |

Таблица А.6 – Таблица «Предметы» (Subjects)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| subject\_id | SERIAL PRIMARY KEY | Уникальный идентификатор предмета |
| subject\_name | VARCHAR(255) NOT NULL | Название предмета |
| hours | INTEGER | Количество часов |
| form\_of\_assessment | VARCHAR(255) | Форма зачета (экзамен, зачет и т.д.) |
| faculty\_id | INTEGER REFERENCES Faculties(faculty\_id) | Ссылка на таблицу Faculties |

Таблица А.7 – Таблица «Группы» (StudyGroups)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| group\_id | SERIAL PRIMARY KEY | Уникальный идентификатор группы |
| group\_name | VARCHAR(255) NOT NULL | Название группы |
| start\_year | INTEGER | Год начала обучения |
| faculty\_id | INTEGER REFERENCES Faculties(faculty\_id) | Ссылка на таблицу Faculties |

Таблица А.8 – Таблица «Оценки» (Grades)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| grade\_id | SERIAL PRIMARY KEY | Уникальный идентификатор оценки |
| student\_id | INTEGER REFERENCES Students(student\_id) | Ссылка на таблицу Students |
| subject\_id | INTEGER REFERENCES Subjects(subject\_id) | Ссылка на таблицу Subjects |
| teacher\_id | INTEGER REFERENCES Teachers(teacher\_id) | Ссылка на таблицу Teachers |
| grade | INTEGER | Оценка |
| date | DATE | Дата оценки |

Таблица А.9 – Таблица «Посещаемость» (Attendance)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| attendance\_id | SERIAL PRIMARY KEY | Уникальный идентификатор записи о посещаемости. |
| student\_id | INTEGER REFERENCES Students(student\_id) | ID студента (внешний ключ к таблице Students). |
| subject\_id | INTEGER REFERENCES Subjects(subject\_id) | ID предмета (внешний ключ к таблице Subjects). |
| attendance\_date | DATE | Дата занятия. |
| status | VARCHAR(50) | Статус посещения (например, «присутствовал», «отсутствовал»). |

Таблица А.10 – Таблица «Отчеты» (Reports)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| report\_id | SERIAL PRIMARY KEY | Уникальный идентификатор отчета. |
| report\_type | VARCHAR(255) | Тип отчета (например, «успеваемость», «посещаемость»). |
| report\_data | JSONB | Данные отчета в формате JSON. |
| created\_at | TIMESTAMP | Дата и время создания отчета. |
| employee\_id | INTEGER REFERENCES Employees(employee\_id) | ID сотрудника (внешний ключ к таблице Сотрудники). |

Приложение Б

CREATE TABLE Faculties (

faculty\_id SERIAL PRIMARY KEY,

faculty\_name VARCHAR NOT NULL,

description TEXT

);

CREATE TABLE Accounts (

account\_id SERIAL PRIMARY KEY,

last\_name VARCHAR NOT NULL,

first\_name VARCHAR NOT NULL,

middle\_name VARCHAR,

address TEXT,

phone\_number VARCHAR,

email VARCHAR UNIQUE

);

CREATE TABLE StudyGroups (

group\_id SERIAL PRIMARY KEY,

group\_name VARCHAR NOT NULL,

start\_year INT NOT NULL,

faculty\_id INT NOT NULL,

CONSTRAINT fk\_groups\_faculty FOREIGN KEY (faculty\_id) REFERENCES Faculties(faculty\_id) ON DELETE CASCADE

);

CREATE TABLE Students (

student\_id SERIAL PRIMARY KEY,

account\_id INT NOT NULL,

record\_book\_id VARCHAR UNIQUE NOT NULL,

group\_id INT NOT NULL,

CONSTRAINT fk\_students\_account FOREIGN KEY (account\_id) REFERENCES Accounts(account\_id) ON DELETE CASCADE,

CONSTRAINT fk\_students\_group FOREIGN KEY (group\_id) REFERENCES StudyGroups(group\_id) ON DELETE CASCADE

);

CREATE TABLE Teachers (

teacher\_id SERIAL PRIMARY KEY,

account\_id INT NOT NULL,

academic\_degree VARCHAR,

academic\_rank VARCHAR,

education\_level VARCHAR,

CONSTRAINT fk\_teachers\_account FOREIGN KEY (account\_id) REFERENCES Accounts(account\_id) ON DELETE CASCADE

);

Листинг Б – Листинг SQL-запросов для создания таблиц базы данных

Приложение В

-- Обновление оценки

CREATE OR REPLACE PROCEDURE proc\_update\_grade(

p\_grade\_id INT, p\_new\_grade INT)

LANGUAGE plpgsql AS $$

BEGIN

UPDATE Grades SET grade = p\_new\_grade WHERE grade\_id = p\_grade\_id;

END; $$;

-- Удаление оценки

CREATE OR REPLACE PROCEDURE proc\_delete\_grade(p\_grade\_id INT)

LANGUAGE plpgsql AS $$

BEGIN

DELETE FROM Grades WHERE grade\_id = p\_grade\_id;

END; $$;

-- Добавление посещаемости

CREATE OR REPLACE PROCEDURE proc\_add\_attendance(

p\_student\_id INT, p\_subject\_id INT, p\_date DATE, p\_status VARCHAR)

LANGUAGE plpgsql AS $$

BEGIN

INSERT INTO Attendance(student\_id,subject\_id,attendance\_date,status)

VALUES(p\_student\_id,p\_subject\_id,p\_date,p\_status);

END; $$;

-- Обновление статуса посещаемости

CREATE OR REPLACE PROCEDURE proc\_update\_attendance(

p\_att\_id INT, p\_new\_status VARCHAR)

LANGUAGE plpgsql AS $$

BEGIN

UPDATE Attendance SET status = p\_new\_status WHERE attendance\_id = p\_att\_id;

END; $$;

-- Удаление записи посещаемости

CREATE OR REPLACE PROCEDURE proc\_delete\_attendance(p\_att\_id INT)

LANGUAGE plpgsql AS $$

BEGIN

DELETE FROM Attendance WHERE attendance\_id = p\_att\_id;

END; $$;

Листинг В – Листинг SQL-запросов для создания процедур базы данных

Приложение Г

-- Средняя оценка студента

CREATE OR REPLACE FUNCTION fn\_get\_student\_avg(p\_student INT)

RETURNS NUMERIC LANGUAGE sql AS $$

SELECT AVG(grade)::NUMERIC FROM Grades WHERE student\_id = p\_student;

$$;

-- Средняя оценка студента по семестрам

CREATE OR REPLACE FUNCTION fn\_get\_student\_avg\_by\_sem(p\_student INT)

RETURNS TABLE(semester TEXT, avg\_grade NUMERIC) LANGUAGE sql AS $$

SELECT

CASE WHEN EXTRACT(MONTH FROM date)<=6 THEN 'Spring' ELSE 'Fall' END,

AVG(grade)

FROM Grades

WHERE student\_id = p\_student

GROUP BY 1;

$$;

-- Список студентов преподавателя

CREATE OR REPLACE FUNCTION fn\_get\_teacher\_students(p\_teacher INT)

RETURNS TABLE(student\_id INT, last\_name VARCHAR, first\_name VARCHAR) LANGUAGE sql AS $$

SELECT s.student\_id, a.last\_name, a.first\_name

FROM Grades g

JOIN Students s ON g.student\_id = s.student\_id

JOIN Accounts a ON s.account\_id = a.account\_id

WHERE g.teacher\_id = p\_teacher

GROUP BY s.student\_id,a.last\_name,a.first\_name;

$$;

-- Средний балл по группе по предмету (для преподавателя)

CREATE OR REPLACE FUNCTION fn\_get\_avg\_by\_group(

p\_teacher INT, p\_subject INT)

RETURNS TABLE(group\_name VARCHAR, avg\_grade NUMERIC) LANGUAGE sql AS $$

SELECT sg.group\_name, AVG(g.grade)

FROM Grades g

JOIN Students s ON g.student\_id = s.student\_id

JOIN StudyGroups sg ON s.group\_id = sg.group\_id

WHERE g.teacher\_id=p\_teacher AND g.subject\_id=p\_subject

GROUP BY sg.group\_name;

$$;

Листинг Г – Листинг SQL-запросов для создания функций базы данных

Приложение Д

-- Успеваемость студента

CREATE OR REPLACE VIEW view\_student\_grades AS

SELECT s.student\_id, a.last\_name, a.first\_name,

sub.subject\_name, g.grade, g.date

FROM Grades g

JOIN Students s ON g.student\_id=s.student\_id

JOIN Accounts a ON s.account\_id=a.account\_id

JOIN Subjects sub ON g.subject\_id=sub.subject\_id;

-- Средний балл студента

CREATE OR REPLACE VIEW view\_student\_avg AS

SELECT student\_id, AVG(grade)::NUMERIC AS avg\_grade

FROM Grades GROUP BY student\_id;

-- Средний балл по семестрам

CREATE OR REPLACE VIEW view\_student\_avg\_sem AS

SELECT student\_id,

CASE WHEN EXTRACT(MONTH FROM date)<=6 THEN 'Spring' ELSE 'Fall' END AS semester,

AVG(grade)::NUMERIC AS avg\_grade

FROM Grades GROUP BY student\_id, CASE WHEN EXTRACT(MONTH FROM date)<=6 THEN 'Spring' ELSE 'Fall' END;

-- Список студентов преподавателя (группировка по факультету и группе)

CREATE OR REPLACE VIEW view\_teacher\_students AS

SELECT t.teacher\_id, f.faculty\_name, sg.group\_name, s.student\_id, a.last\_name, a.first\_name

FROM Grades g

JOIN Teachers t ON g.teacher\_id=t.teacher\_id

JOIN Students s ON g.student\_id=s.student\_id

JOIN Accounts a ON s.account\_id=a.account\_id

JOIN StudyGroups sg ON s.group\_id=sg.group\_id

JOIN Faculties f ON sg.faculty\_id=f.faculty\_id

ORDER BY f.faculty\_name, sg.group\_name;

Листинг Д – Листинг SQL-запросов для создания представлений базы данных

Приложение Е

-- Создание ролей

CREATE ROLE student\_role;

CREATE ROLE teacher\_role;

CREATE ROLE admin\_role;

CREATE ROLE analytics\_role;

-- пользователи

CREATE USER student\_user WITH PASSWORD 'student\_pass';

CREATE USER teacher\_user WITH PASSWORD 'teacher\_pass';

CREATE USER admin\_user WITH PASSWORD 'admin\_pass';

CREATE USER analytics\_user WITH PASSWORD 'analytics\_pass';

-- привязка ролей к пользователям

GRANT student\_role TO student\_user;

GRANT teacher\_role TO teacher\_user;

GRANT admin\_role TO admin\_user;

GRANT analytics\_role TO analytics\_user;

-- студент: только чтение своих view-функций

GRANT SELECT ON view\_student\_grades, view\_student\_avg, view\_student\_avg\_sem, view\_students\_risk TO student\_role;

GRANT EXECUTE ON FUNCTION fn\_get\_student\_grades(INT), fn\_get\_student\_avg(INT), fn\_get\_student\_avg\_by\_sem(INT) TO student\_role;

-- преподаватель: чтение + добавление/обновление оценок и отчётов по своим предметам

GRANT SELECT ON view\_teacher\_students, view\_avg\_by\_group\_subject, view\_attendance\_pct TO teacher\_role;

GRANT EXECUTE ON PROCEDURE proc\_add\_grade(INT,INT,INT,INT,DATE), proc\_update\_grade(INT,INT) TO teacher\_role;

GRANT EXECUTE ON FUNCTION fn\_get\_teacher\_students(INT), fn\_get\_avg\_by\_group(INT,INT), fn\_get\_attendance\_pct(INT,INT) TO teacher\_role;

-- администратор: полный доступ ко всему

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO admin\_role;

GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA public TO admin\_role;

GRANT EXECUTE ON ALL PROCEDURES IN SCHEMA public TO admin\_role;

-- аналитик: доступ только к отчётным view и функциям

GRANT SELECT ON view\_admin\_performance, view\_stats\_summary, view\_stats\_dynamics TO analytics\_role;

GRANT EXECUTE ON FUNCTION fn\_students\_at\_risk(), fn\_group\_summary(INT), fn\_performance\_dynamics(INT,INT), fn\_get\_reports(TEXT) TO analytics\_role;

Листинг Е – Листинг SQL-запросов для создания пользователей и ролей базы данных