

Санкт-Петербургский национальный исследовательский университет ИТМО

Факультет Инфокоммуникационных технологий

Отчет по итоговому проекту

Выполнили:

Студент 2 курса группы К32421

Козлов Всеволод Денисович

Студент 2 курса группы К32402

Горбатов Дмитрий Алексеевич

Санкт-Петербург 2023

Оглавление

Описание предметной области	6
Выполнение	7
Архитектура БД.....	7
Описание атрибутов сущностей и ограничений на данные	8
Запросы к базе данных	16
Запрос №1:	16
Команда:	16
Результат (рис 2.):.....	16
Запрос №2:	17
Команда:	17
Результат(рис. 3):.....	17
Запрос №3:	18
Команда:	18
Результат (рис 4.):.....	18
Запрос №4:	19
Команда:	19
Результат (рис. 5):.....	19
Запрос №5:	20
Команда:	20
Результат (рис. 6):.....	20
Запрос №6:	21
Команда:	21
Результат (рис. 7):.....	21
Запрос №7:	22
Команда:	22
Результат (рис. 8):.....	22
Создание представлений.....	23
Представление №1:	23
Команда:	23
Результат (рис. 9):.....	23
Представление №2:	24
Команда:	24

Результат (рис. 10):	24
Запросы на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.....	25
Запрос №1 на update:	25
Команда:	25
Скриншот до (рис. 11):.....	25
Скриншот после (рис.12):.....	26
Запрос №2 на update:	26
Команда:	26
Скриншот до (рис. 13):.....	27
Скриншот после (рис. 14):.....	27
Запрос №1 на удаление:	28
Команда:	28
Скриншот до (рис. 15):.....	28
Скриншот после (рис. 16):.....	29
Запрос №2 на удаление:	30
Команда:	30
Скриншот до (рис. 17):.....	30
Скриншот после (рис.18):.....	31
Запрос №1 на вставку:	31
Команда:	31
Скриншот до (рис. 19):.....	32
Скриншот после (рис. 20):.....	32
Создание индексов	33
Запрос с составным индексом:	33
Команда:	33
План операции до создания индексов (рис. 21):	34
Время до создания индексов:	35
437ms	35
Добавление индексов:	35
create index try_mark_idx on record_book(try_number, mark);.....	35
План операции после создания индексов (рис. 22):	35
Время после создания индексов:	35

Запрос без составных индексов:	36
Команда:	36
Время до создания индексов:	36
План операции до создания индексов (рис. 23):	36
Добавление индексов:	37
Время после создания индексов:	37
План операции после добавления индексов:	37
План после добавления 10000 студентов (рис. 25):	39
Методы:	40
№ . 1	40
Описание:	40
Код метода:	40
Выполнение метода:	40
№ . 2	41
Описание:	41
Код метода:	41
Выполнение метода:	42
№ 3	42
Описание:	42
Код метода:	42
Выполнение метода:	43
№4. Бонус	44
Описание:	44
Код метода:	44
Триггеры:	45
№1	45
Описание:	45
Код:	45
Выполнение:	46
№2	47
Описание:	47
Код:	47
Выполнение:	47

Выводы	48
---------------------	-----------

Цели проекта

- Создать архитектуру БД, которая может быть использована для ведения учебного процесса в высшем учебном заведении.
- Продумать ограничения на столбцы в БД, чтобы поддерживать ее целостность
 - Реализовать архитектуру БД, используя PostgreSQL
 - Заполнить БД тестовыми данными для проверки работоспособности и дальнейшей оптимизации БД при помощи индексов
 - Проверить работоспособность БД при помощи запросов на получение, вставку, изменение и удаление данных
 - Реализовать представления
 - Оптимизировать работу БД при помощи индексов
 - Реализовать функции
 - Реализовать триггеры

Описание предметной области

БД содержит сведения о сдаче сессии студентами. Номер зачетной книжки однозначно идентифицирует студента. Каждый студент обучается в группе, причем номера групп меняются каждый очередной учебный год. Дисциплины, по которым студенты сдают промежуточную аттестацию, соотнесены с учебным планом образовательной программы, которая в свою очередь относится к направлению подготовки, реализуемому в определенном подразделении вуза. По каждой дисциплине могут проводиться лекционные, лабораторные/практические занятия и практика определенного объема часов. По каждой дисциплине и практике проводится аттестация в формате экзамен/дифзачет/зачет. Одна дисциплина может соотноситься с несколькими учебными планами разных направлений подготовки. Каждый учебный план относится к определенному году приема. Экзамены проходят на различных площадках вуза, территориально расположенных в разных частях города или страны.

БД должна содержать следующий минимальный набор сведений: Номер зачетной книжки. Фамилия студента. Имя студента. Отчество студента. Курс. Группа. Учебный год. Семестр. Код дисциплины/практики. Название дисциплины/практики. Код направления. Название направления. Оценка. Фамилия преподавателя. Имя преподавателя. Отчество преподавателя. Должность. Код подразделения. Подразделение. Дата сдачи экзамена/зачета/дифзачета. Аудитория. Площадка (адрес). Номер попытки (максимально 3).

Дополните исходные данные информацией: по расписанию сессии, по назначению базовой и повышенной стипендии.

Выполнение

Архитектура БД

В нотации idf1x (рис 1.). Архитектура также есть в формате PDF в архиве.

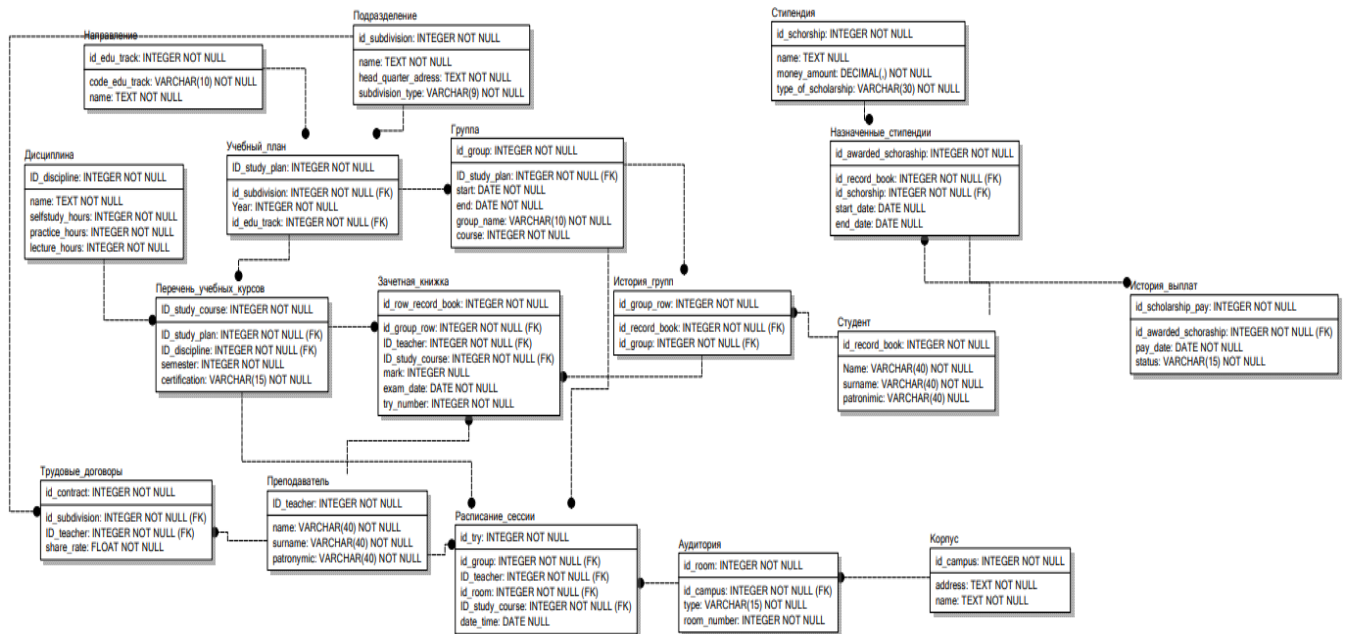


Рисунок 1 – нотация idf1x

Описание атрибутов сущностей и ограничений на данные

Таблица 1 – Описание атрибутов сущностей

Наименование атрибута	Тип	Первичный ключ		Внешний ключ	Обязательность	Ограничения целостности
		Собственный атрибут	Внешний ключ			
Subdivisions (подразделения)						
id_subdivision	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения
name	VARCHAR(40)				+	
headquarter_addresses (адрес головного офиса)	VARCHAR(50)				+	
type	VARCHAR(9)				+	Значение должно выбираться из списка (основной, филиал)
Study_plans (Учебный план)						
id_study_plan	INTEGER	+			+	Значение соответствует первичному ключу сущности учебный план
id_edu_track(Код направления)	INTEGER			+	+	Значение соответствует первичному ключу сущности направление

Id_subdivision (ID подразделения)	INTEGER			+	+	Значение соответствует первичному ключу сущности подра зделение
Year (Год)	INTEGER				+	1900>val>2030
Edu_tracks (Направление)						
Id_edu_track	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения
Code_edu_track (Код направления)	VARCHAR R(10)				+	
Name (Название направления)	VARCHAR R(50)				+	
Disciplines (Дисциплина)						
Id_discipline	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения
Name	VARCHAR R(70)				+	
Selfstudy_hours	INTEGER				+	>0
Practice_hours	INTEGER				+	>0
Lecture_hours	INTEGER				+	>0
Study_courses (Перечень учебных курсов)						
Id_study_course	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения
Semester	INTEGER					1<=val<=8
Certification	VARCHAR R(9)				+	Значение должно выбираться из списка (зачет, диф.зачет, экзамен)

Id_study_plan	INTEGER			+	+	Значение соответствует первичному ключу сущности учебный план
Id_discipline	INTEGER			+	+	Значение соответствует первичному ключу сущности дисциплина
Record_book (Зачетная книжка)						
Id_row_record_book	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения
Id_study_course	INTEGER			+	+	Значение соответствует первичному ключу сущности учебный курс
Id_group_row	INTEGER			+	+	Значение соответствует первичному ключу сущности студент
Id_teacher	INTEGER			+	+	Значение соответствует первичному ключу сущности преподаватель
Mark	INTEGER				+	2<=val<=5
Try_number	INTEGER				+	1<=val <= 3
Students (Студент)						
Id_record_book	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения
Name	VARCHAR(40)				+	

Surname	VARCHAR(40)				+	
Patronymic	VARCHAR(40)				-	
Student_groups (История групп)						
Id_group_row	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения
Id_group	INTEGER			+	+	Значение соответствует первичному ключу сущности группа
Id_record_book	INTEGER			+	+	Значение соответствует первичному ключу сущности студент
Groups (Группа)						
Id_group	INTEGER	+				Уникален, необходимо обеспечить автоматическую генерацию значения
Id_study_plan	INTEGER			+	+	Значение соответствует первичному ключу сущности учебный план
Start_date	DATE				+	
End_date	DATE				+	val > Начало обучения
Group_name	VARCHAR(10)				+	
Course	INTEGER				+	0 <= val <= 4
Awarded_scholarships (Назначенные стипендии)						
Id_awarded_scholarship	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения

Id_scholarship	INTEGER			+	+	Значение соответствует первичному ключу сущности стипендия
Id_record_book	INTEGER				+	Значение соответствует первичному ключу сущности студент
Start_date	DATE				+	
End_date	DATE				+	val > C
Scholarships (Стипендия)						
Id_scholarship	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения
name	VARCHAR(100)				+	
Money_amount	MONEY				+	val>0
Type_of_scholarship	VARCHAR(13)				+	Значение должно выбираться из списка (социальная, академическая, именная)
Scholarship_payments (История выплат)						
Id_scholarship_pay	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения
Id_awarded_scholarship	INTEGER			+	+	Значение соответствует первичному ключу сущности назначенные стипендии

Pay_date	DATE				+	
Status	VARCHAR(14)				+	Значение должно выбираться из списка (выплачено, не выплачено, обрабатывается)
Campuses (Корпус)						
Id_campus	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения
Address	VARCHAR(70)				+	
name	VARCHAR(30)				+	
Rooms (Аудитория)						
Id_room	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения
Id_campus	INTEGER			+	+	Значение соответствует первичному ключу сущности корпус
Room_type	VARCHAR(12)				+	Значение должно выбираться из списка (лекционная, практическая, лаборатория)
Room_number	INTEGER				+	val>0
Session_schedule (Расписание сессии)						
Id_schedule (id сдачи)	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения

Id_group	INTEGER			+	+	Значение соответствует первичному ключу сущности группа
Id_teacher	INTEGER			+	+	Значение соответствует первичному ключу сущности преподаватель
Id_room	INTEGER			+	+	Значение соответствует первичному ключу сущности аудитория
Id_study_course	INTEGER			+	+	Значение соответствует первичному ключу сущности перечень учебных курсов
Date_time	DATE				+	
Teachers (Преподаватель)						
Id_teacher	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения
Name	VARCHAR(40)				+	
Surname	VARCHAR(40)				+	
Patronymic	VARCHAR(40)				-	
Employment_contracts (Трудовые договоры)						
Id_contract	INTEGER	+			+	Уникален, необходимо обеспечить автоматическую генерацию значения
Id_teachet	INTEGER			+	+	Значение соответствует первичному ключу сущности

						преподаватель
Share_rate (Доля ставки)	FLOAT				+	0.1, 0.25, 0.5, 0.75, 1
Id_subdivision	INTEGER			+	+	Значение соответствует первичному ключу сущности подразделения

Запросы к базе данных

Запрос №1:

Список дисциплин, которые должны быть сданы заданной группой с указанием дат сдачи и фамилий преподавателей.

Команда:

```
select date_time, surname
from session_schedule join teachers
    on session_schedule.id_teacher = teachers.id_teacher
where id_group = (select id_group from groups where
group_name='K10');
```

Результат (рис 2.):

	<input type="checkbox"/> date_time	<input type="checkbox"/> surname
1	2012-01-11 10:00:00	teacher_surname0
2	2012-01-11 10:00:00	teacher_surname1
3	2012-01-04 10:00:00	teacher_surname2
4	2012-01-28 10:00:00	teacher_surname3
5	2012-01-23 10:00:00	teacher_surname4
6	2012-09-25 10:00:00	teacher_surname5
7	2012-09-15 10:00:00	teacher_surname6
8	2012-09-13 10:00:00	teacher_surname7
9	2012-09-15 10:00:00	teacher_surname8
10	2012-09-01 10:00:00	teacher_surname9

Рисунок 2 – результат работы 1-ого запроса

Запрос №2:

Список студентов, получивших двойки на первой попытке с указанием фамилии преподавателя, которым они должны пересдать экзамен.

Команда:

```
select s.surname as student_name, t.surname as teacher_surname
from record_book
join student_groups sg
  on sg.id_group_row = record_book.id_group_row
join students s
  on sg.id_record_book = s.id_record_book
join teachers t
  on record_book.id_teacher = t.id_teacher
where try_number=1 and mark=2;
```

Результат(рис. 3):

	student_name	teacher_surname
1	student_surname0	teacher_surname0
2	student_surname0	teacher_surname2
3	student_surname0	teacher_surname1
4	student_surname0	teacher_surname7
5	student_surname0	teacher_surname6
6	student_surname0	teacher_surname4
7	student_surname0	teacher_surname18
8	student_surname0	teacher_surname16
9	student_surname0	teacher_surname19
10	student_surname0	teacher_surname15
11	student_surname0	teacher_surname14
12	student_surname0	teacher_surname13
13	student_surname0	teacher_surname12

Рисунок 3 – результат работы 2-ого запроса

Запрос №3:

Фамилии студентов, получивших оценки по дисциплине, которые выше среднего балла по этой дисциплине.

Команда:

```
from record_book
join student_groups sg
  on sg.id_group_row = record_book.id_group_row
join students s
  on sg.id_record_book = s.id_record_book
join (select id_study_course, avg(mark) as avg_mark from
record_book where mark<>2 group by id_study_course) avg_mark
  on record_book.id_study_course = avg_mark.id_study_course
where record_book.mark > avg_mark.avg_mark;
```

Результат (рис 4.):

	stud_surname	id_study_course
1	student_surname0	7
2	student_surname0	0
3	student_surname0	5
4	student_surname0	3
5	student_surname0	8
6	student_surname0	6
7	student_surname0	15
8	student_surname0	19
9	student_surname0	11
10	student_surname1	2
11	student_surname1	5
12	student_surname1	6
13	student_surname1	10
14	student_surname1	12
15	student_surname1	15
16	student_surname1	16

Рисунок 4 – результат работы 3-его запроса

Запрос №4:

Рейтинговый список групп по заданному направлению по результатам сдачи сессии, упорядочить его по убыванию.

Команда:

```
select sg.group_name, avg(mark) from record_book
join
  (select id_group_row, group_name from student_groups
  join groups g
   on g.id_group = student_groups.id_group
  where g.id_study_plan = 0) sg
on sg.id_group_row = record_book.id_group_row
where mark <> 2
group by sg.group_name
order by avg(mark) desc;
```

Результат (рис. 5):

	group_name	avg
1	K10	3.9504950495049505
2	K20	3.95

Рисунок 5 – результат работы 4-ого запроса

Запрос №5:

Списки студентов, упорядоченные по группам и фамилиям студентов, назначении на стипендии. Студент получает стипендию, если он сдал сессию без троек. Если студент не назначен на стипендию, указать 0, если назначен – 1.

Команда:

```
select s.surname, sg.group_name,
       case
         when min(mark) > 3 then 1
         else 0
       end as is_awarder
from record_book
join (select id_record_book, id_group_row, group_name
      from student_groups
      join groups g
      on g.id_group = student_groups.id_group) sg
on sg.id_group_row = record_book.id_group_row
join students s
on sg.id_record_book = s.id_record_book
group by (s.surname, sg.group_name)
order by surname, group_name;
```

Результат (рис. 6):

Console 53 ms	name	group_name	is_awarder
1	student_surname0	K10	0
2	student_surname0	K20	0
3	student_surname1	K10	0
4	student_surname1	K20	0
5	student_surname10	K11	0
6	student_surname10	K21	0
7	student_surname11	K11	0
8	student_surname11	K21	0
9	student_surname12	K11	0
10	student_surname12	K21	0

Рисунок 6 – результат работы 5-ого запроса

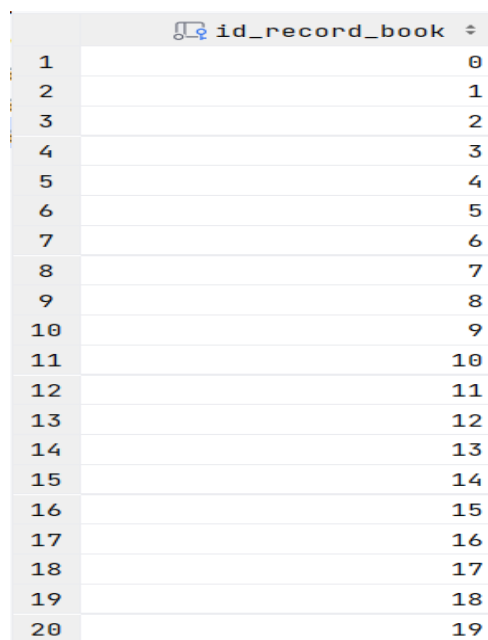
Запрос №6:

Список студентов, сдавших все положенные экзамены.

Команда:

```
select id_record_book from record_book
join (select id_group_row, id_study_course, max(try_number) as
last_try
      from record_book r2
      group by (id_group_row, id_study_course)) as trys
on record_book.try_number=last_try and
   record_book.id_group_row = trys.id_group_row and
   record_book.id_study_course = trys.id_study_course
join student_groups sg
on record_book.id_group_row = sg.id_group_row
group by sg.id_record_book
having min(mark) > 2;
```

Результат (рис. 7):



	id_record_book
1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
11	10
12	11
13	12
14	13
15	14
16	15
17	16
18	17
19	18
20	19

Рисунок 7 – результат работы 6-ого запроса

Запрос №7:

Список студентов, получивших максимальный средний балл в своей группе.

Команда:

```
select sg.id_record_book, sg.id_group
from
  (select id_group_row, avg(mark) as avg_st
   from record_book
   where mark<>2
   group by id_group_row) avg_st
join
  student_groups sg
on
  sg.id_group_row = avg_st.id_group_row
join
  (select id_group, max(avg_st) as max_group_avg from record_book
 as r1
 join student_groups sg on
   r1.id_group_row = sg.id_group_row
 join (select id_group_row, avg(mark) as avg_st
   from record_book
   where mark<>2
   group by id_group_row) s_av
 on r1.id_group_row = s_av.id_group_row
 group by id_group) as max_group_avg
on
  sg.id_group = max_group_avg.id_group
where avg_st=max_group_avg;
```

Результат (рис. 8):



	 id_record_book	 id_group
1	2	0
2	15	1
3	10	3
4	8	2

Рисунок 8 – результат работы 7-ого запроса

Создание представлений

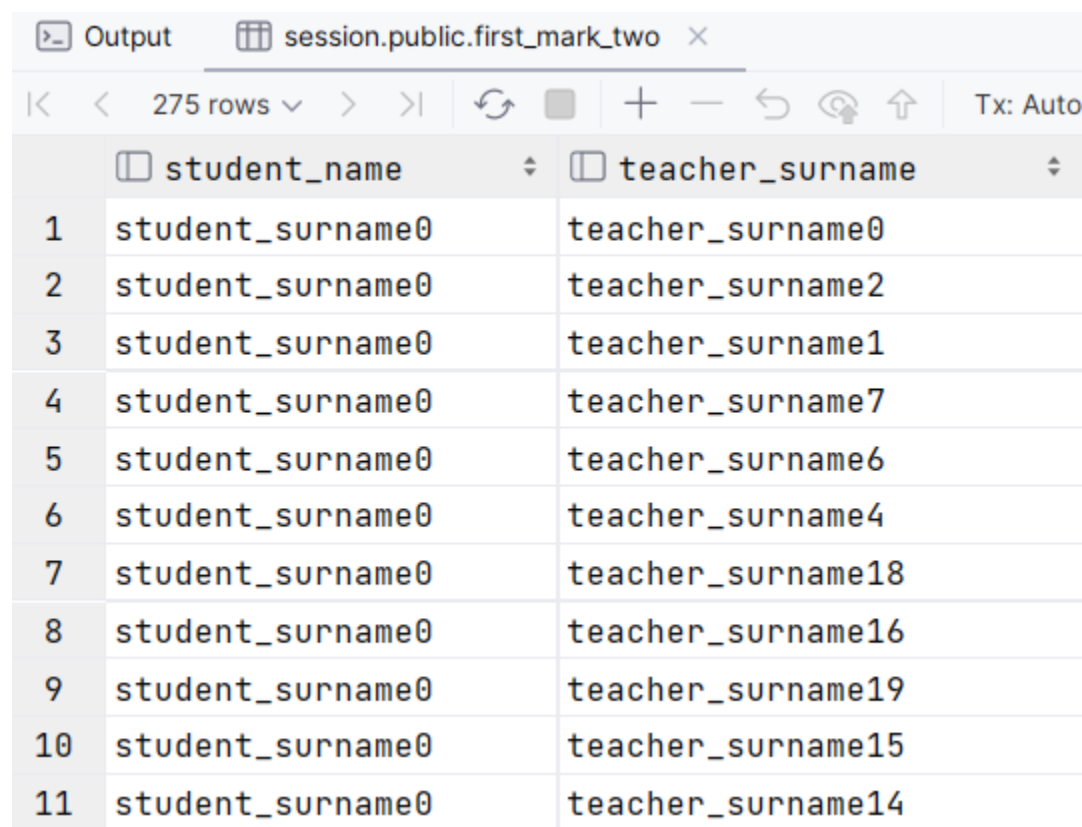
Представление №1:

Список студентов, получивших двойки на первой попытке с указанием фамилии преподавателя, которым они должны пересдать экзамен.

Команда:

```
create or replace view first_mark_two as
select s.surname as student_name, t.surname as teacher_surname
from record_book
join student_groups sg
  on sg.id_group_row = record_book.id_group_row
join students s
  on sg.id_record_book = s.id_record_book
join teachers t
  on record_book.id_teacher = t.id_teacher
where try_number=1 and mark=2;
```

Результат (рис. 9):



The screenshot shows a database interface with a query result table. The table has two columns: 'student_name' and 'teacher_surname'. There are 11 rows of data. The first column contains the value 'student_surname0' for all rows. The second column contains various teacher surnames, such as 'teacher_surname0', 'teacher_surname2', 'teacher_surname1', etc.

	student_name	teacher_surname
1	student_surname0	teacher_surname0
2	student_surname0	teacher_surname2
3	student_surname0	teacher_surname1
4	student_surname0	teacher_surname7
5	student_surname0	teacher_surname6
6	student_surname0	teacher_surname4
7	student_surname0	teacher_surname18
8	student_surname0	teacher_surname16
9	student_surname0	teacher_surname19
10	student_surname0	teacher_surname15
11	student_surname0	teacher_surname14

Рисунок 9 – представление №1

Представление №2:

Создать представление для студентов, побывавших на ппа2

Команда:

```
create or replace view students_ppa2 as
select s.surname as student_name, t.surname as teacher_surname
from record_book
join student_groups sg
on sg.id_group_row = record_book.id_group_row
join students s
on sg.id_record_book = s.id_record_book
join teachers t
on record_book.id_teacher = t.id_teacher
where try_number=2 and mark=2;
```

Результат (рис. 10):

	student_name	teacher_surname
1	student_surname0	teacher_surname7
2	student_surname0	teacher_surname6
3	student_surname0	teacher_surname2
4	student_surname0	teacher_surname12
5	student_surname0	teacher_surname11
6	student_surname0	teacher_surname13
7	student_surname0	teacher_surname16
8	student_surname0	teacher_surname14
9	student_surname0	teacher_surname19
10	student_surname0	teacher_surname17
11	student_surname0	teacher_surname10
12	student_surname1	teacher_surname4
13	student_surname1	teacher_surname3
14	student_surname1	teacher_surname13
15	student_surname1	teacher_surname12
16	student_surname1	teacher_surname10
17	student_surname1	teacher_surname11
18	student_surname1	teacher_surname17
19	student_surname2	teacher_surname8
20	student_surname2	teacher_surname0

Рисунок 10 – представление №2

Запросы на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов

Запрос №1 на update:

Заменить название предмета на "nice discipline", по которому student_surname4 получил больше всего 5.

Команда:

```
create temporary table cnt_5_marks as
  select id_discipline, count(*) cnt_5 from record_book
  join study_courses sc
    on record_book.id_study_course = sc.id_study_course
  where
    id_group_row in
      (select id_group_row from student_groups where
id_record_book in
      (select id_record_book from students where
surname='student_surname4'))
    and
      mark=5
  group by id_discipline;

update disciplines
set name='nice discipline'
where id_discipline=(select id_discipline from cnt_5_marks
where cnt_5=(select max(cnt_5) from cnt_5_marks));
select * from disciplines where id_discipline=8;

drop table cnt_5_marks;
```

Скриншот до (рис. 11):

	id_discipline	name	selfstudy_h...	practice_hours	lecture_hours
1	8	name8	85	85	85

Рисунок 11 – запрос №1 – до

Скриншот после (рис.12):

	id_discipline	name	selfstudy_h...	practice_hours	lecture_hours
1	8	nice discipline	85	85	85

Рисунок 12 – запрос №1 – после

Запрос №2 на update:

Увеличить количество selfstudy_hours на 10 для дисциплин с самой низкой средней успеваемостью

Команда:

```
create temporary table discipline_avg as
select d.id_discipline, avg(mark) avg_mark from record_book
join study_courses sc
  on record_book.id_study_course = sc.id_study_course
join
  disciplines d on sc.id_discipline = d.id_discipline
where mark <> 2
group by d.id_discipline;

update disciplines
  set selfstudy_hours = selfstudy_hours + 10
where id_discipline in
  (select id_discipline from discipline_avg
   where avg_mark >= all(select avg_mark from discipline_avg)
   limit 1);

drop table discipline_avg;
```

Скриншот до (рис. 13):

	id_discipline	name	selfstudy_...	practice_h...	lecture_hours
1	0	name0	5	5	5
2	1	name1	15	15	15
3	2	name2	25	25	25
4	3	name3	35	35	35
5	4	name4	45	45	45
6	5	name5	55	55	55
7	6	name6	65	65	65
8	7	name7	75	75	75
9	9	name9	95	95	95
10	10	name10	105	105	105
11	11	name11	115	115	115
12	12	name12	125	125	125
13	13	name13	135	135	135
14	14	name14	145	145	145
15	15	name15	155	155	155
16	16	name16	165	165	165

Рисунок 13 – запрос №2 – до

Скриншот после (рис. 14):

	id_discip... ^ 1	name	selfstudy_...	practice_h...	lecture_hours
1	0	name0	5	5	5
2	1	name1	15	15	15
3	2	name2	25	25	25
4	3	name3	35	35	35
5	4	name4	55	45	45
6	5	name5	55	55	55
7	6	name6	65	65	65
8	7	name7	75	75	75
9	8	nice discipline	85	85	85
10	9	name9	95	95	95
11	10	name10	105	105	105
12	11	name11	115	115	115
13	12	name12	125	125	125
14	13	name13	135	135	135
15	14	name14	145	145	145
16	15	name15	155	155	155

Рисунок 14 – запрос №2 – после

Запрос №1 на удаление:

Удалить подразделение, в котором не обучается ни один студент и не трудоустроен ни один преподаватель

Команда:

```
delete from subdivisions
where id_subdivision not in
    (select distinct id_subdivision from study_plans
     where id_study_plan in
       (select distinct id_study_plan from groups
        where id_group in
          (select distinct id_group from student_groups)))

union

select distinct id_subdivision from employment_contracts);
```

Скриншот до (рис. 15):





	 id_subdivi...	 name	 headquarte...	 type
1	0	name 0	address 0	основной
2	1	name 1	address 1	основной
3	2	name 2	address 2	основной
4	3	name 3	address 3	основной
5	4	name 4	address 4	основной
6	5	name 5	address 5	филиал
7	6	name 6	address 6	филиал
8	7	name 7	address 7	филиал
9	8	name 8	address 8	филиал
10	9	name 9	address 9	филиал

Рисунок 15 – запрос №3 – до

Скриншот после (рис. 16):





	 id_subdivision ▾	 name ▾	 headquarter... ▾	 type ▾
1	0	name 0	address 0	основной
2	1	name 1	address 1	основной
3	2	name 2	address 2	основной
4	3	name 3	address 3	основной
5	4	name 4	address 4	основной
6	5	name 5	address 5	филиал
7	6	name 6	address 6	филиал
8	7	name 7	address 7	филиал
9	8	name 8	address 8	филиал
10	9	name 9	address 9	филиал

Рисунок 16 – запрос №3 – после

Запрос №2 на удаление:

Удалить учебные планы, которые были созданы позже всего и не реализует ни одной дисциплины

Команда:

```
delete from study_plans
where id_study_plan in
    (select id_study_plan from study_plans
     where id_study_plan not in
         (select distinct id_study_plan from study_courses)
     and year>=all(select year from study_plans));
```

Скриншот до (рис. 17):





	 id_study_plan ▾	 id_subdivi... ▾	 year ▾	 id_edu_track ▾
1	0	0	2010	0
2	1	0	2010	0
3	2	0	2011	0
4	3	0	2011	1
5	4	0	2012	1
6	5	1	2012	1
7	6	1	2013	2
8	7	1	2013	2
9	8	1	2014	2
10	9	1	2014	3

Рисунок 17 – запрос №4 – до

Скриншот после (рис.18):

	id_study_plan	id_subdivision	year	id_edu_track
1	0	0	2010	0
2	1	0	2010	0
3	2	0	2011	0
4	3	0	2011	1
5	4	0	2012	1
6	5	1	2012	1
7	6	1	2013	2
8	7	1	2013	2

Рисунок 18 – запрос №4 – после

Запрос №1 на вставку:

Для студента с id_record_book=2 во время его обучения в группе с id_group=0. Если по дисциплине была получена 3 на 1-й попытке, то добавить 4 со второй попытки.

Команда:

```
insert into record_book
(id_group_row, id_teacher, id_study_course, mark, try_number)
select id_group_row, id_teacher, id_study_course, 4, 2 from
record_book
where
    id_group_row =
    (select id_group_row from student_groups
     where id_group=0 and id_record_book=2)
and
    mark=3
and
    try_number=1;
```

Скриншот до (рис. 19):

	id_row_record_book	id_group_row	id_teacher	id_study_course	mark	try_number
7	46	4	6	6	3	1
8	47	4	7	7	5	1
9	48	4	8	8	5	3
10	49	4	9	9	5	1
11	448	4	0	0	2	1
12	449	4	0	0	2	2
13	450	4	1	1	2	1
14	451	4	3	3	2	1
15	452	4	5	5	2	1
16	453	4	8	8	2	1
17	454	4	8	8	2	2

Рисунок 19 – запрос №5 – до

Скриншот после (рис. 20):

	id_row_record_book	id_group_row	id_teacher	id_study_course	mark	try_number
1	40	4	0	0	3	3
2	41	4	1	1	5	2
3	42	4	2	2	5	1
4	43	4	3	3	5	2
5	44	4	4	4	5	1
6	45	4	5	5	3	2
7	46	4	6	6	3	1
8	47	4	7	7	5	1
9	48	4	8	8	5	3
10	49	4	9	9	5	1
11	817	4	6	6	4	2

Рисунок 20 – запрос №5 – после

Создание индексов

Запрос с составным индексом:

Список студентов, получивших двойки на первой попытке с указанием фамилии преподавателя, которым они должны пересдать экзамен.

Команда:

```
select s.surname as student_name, t.surname as teacher_surname
from record_book
join student_groups sg
    on sg.id_group_row = record_book.id_group_row
join students s
    on sg.id_record_book = s.id_record_book
join teachers t
    on record_book.id_teacher = t.id_teacher
where try_number=1 and mark=2;
```

План операции до создания индексов (рис. 21):

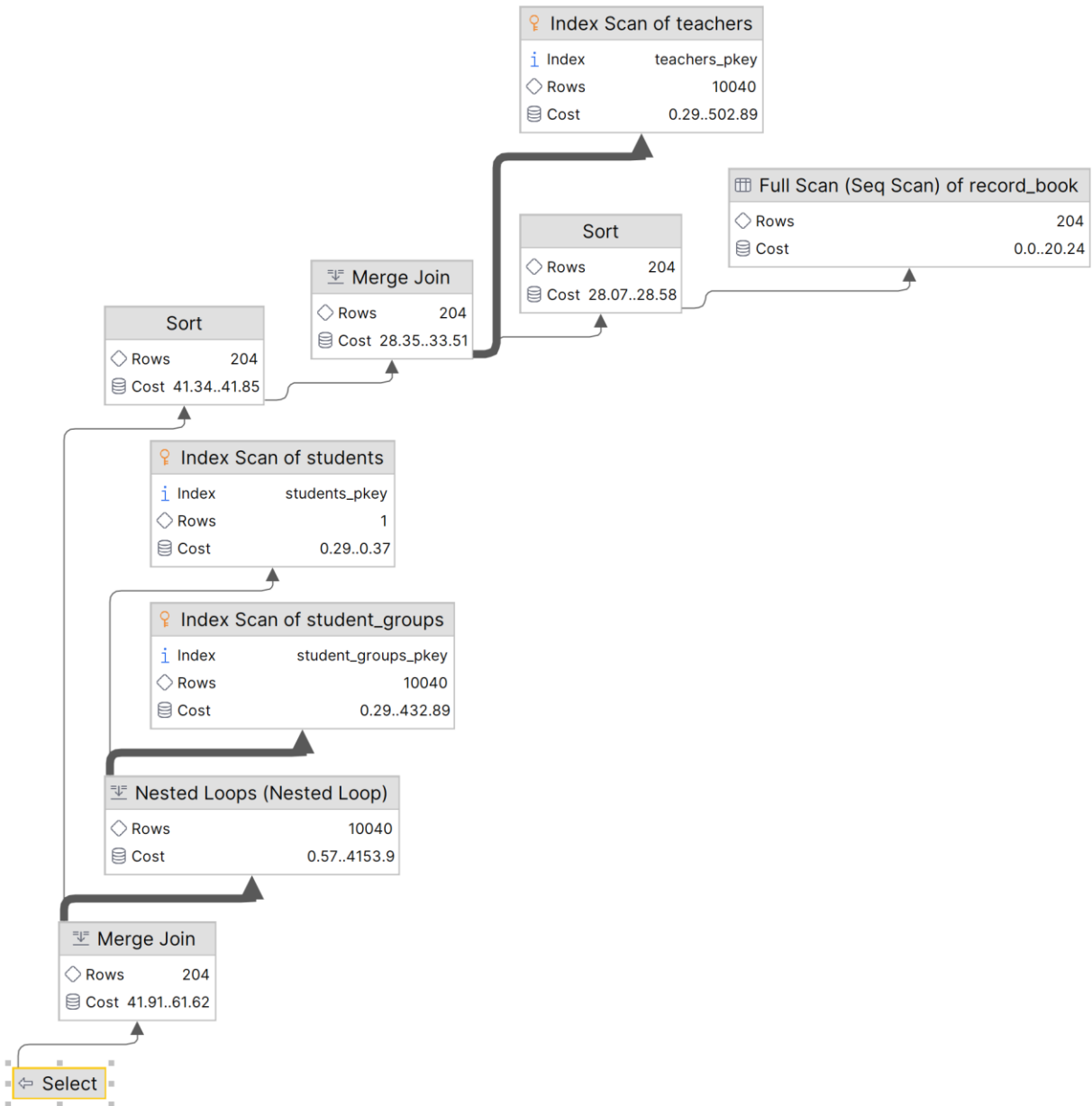


Рисунок 21 – план операции до создания индексов

Время до создания индексов:

437ms

Добавление индексов:

```
create index try_mark_idx on record_book(try_number, mark);  
create index record_book_idteacher_idx on  
record_book(id_teacher);  
create index record_book_group_row_idx on  
record_book(id_group_row);  
create index group_recordbook_idx on  
student_groups(id_record_book);
```

План операции после создания индексов (рис. 22):

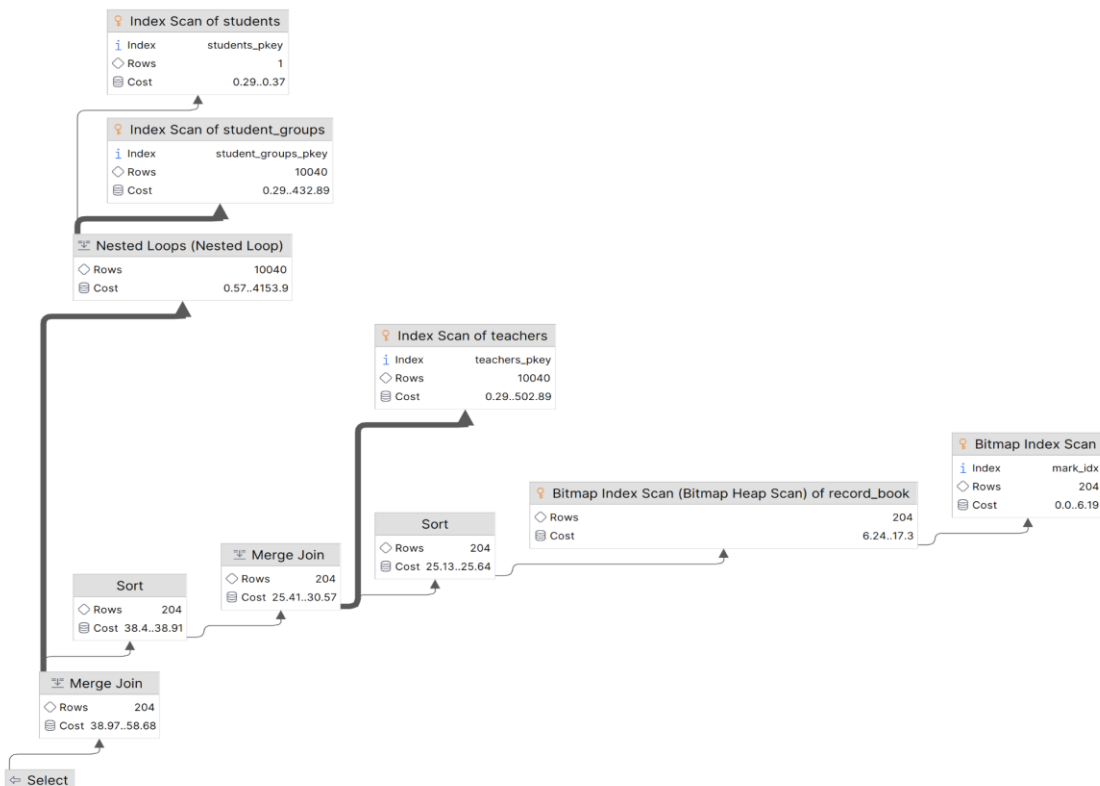


Рисунок 22 – план операции после создания индексов

Время после создания индексов:

58ms

Запрос без составных индексов:

Получить количество 5-к по дисциплинам у студента с surname=student_surname4

Команда:

```
select id_discipline, count(*) cnt_5 from record_book
join study_courses sc
  on record_book.id_study_course = sc.id_study_course
where
  id_group_row in
    (select id_group_row from student_groups where id_record_book
in
      (select id_record_book from students where
surname='student_surname4'))
  and
    mark=5
group by id_discipline;
```

Время до создания индексов:

68ms

План операции до создания индексов (рис. 23):

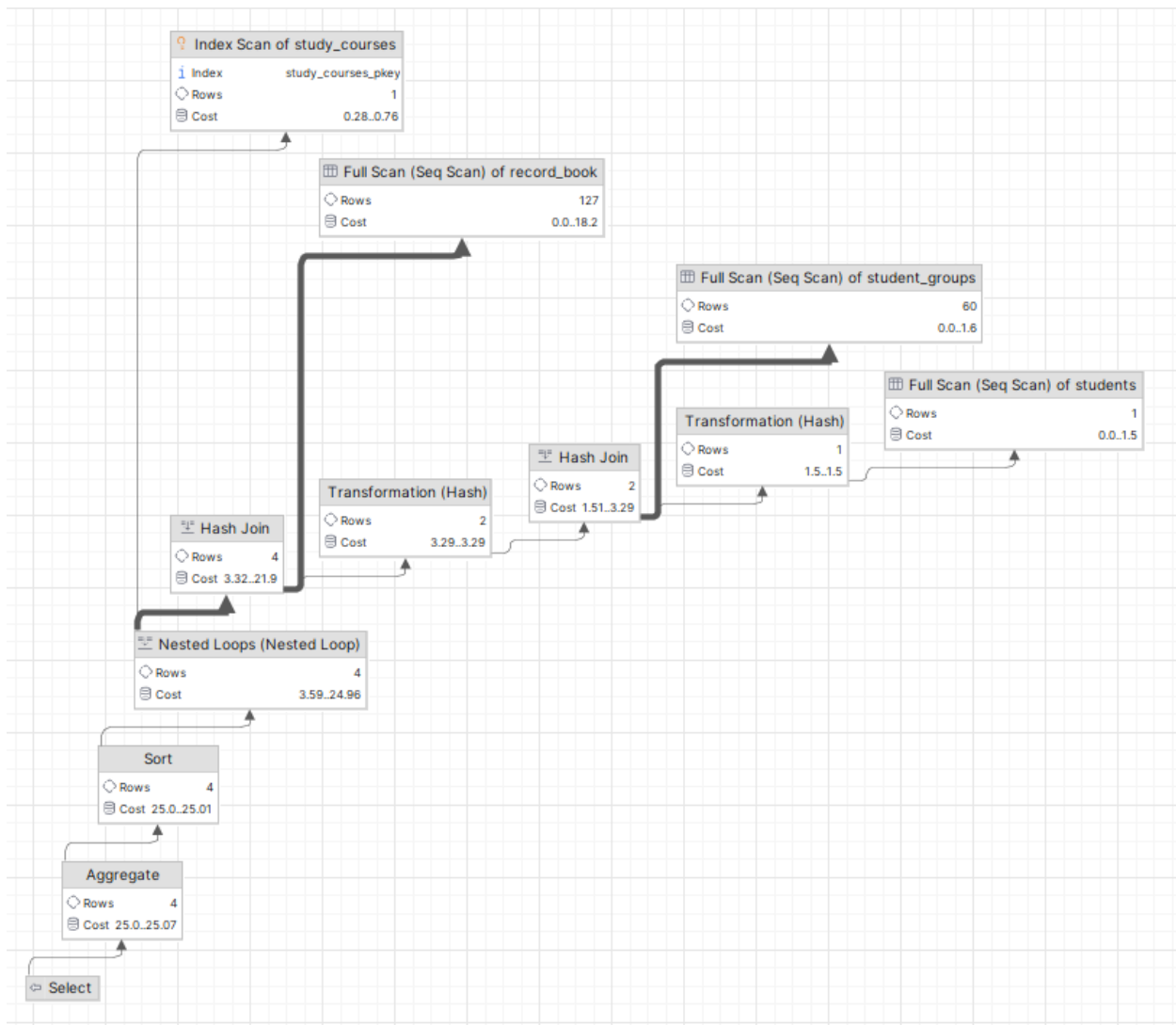


Рисунок 23 – план операции до создания индексов

Добавление индексов:

```
create index mark_idx on record_book(mark);
create index student_surname_idx on students(surname);
create index group_recordbook_idx on student_groups(id_record_book);
```

Время после создания индексов:

48ms

План операции после добавления индексов:

План до добавления дополнительных студентов (рис. 24):

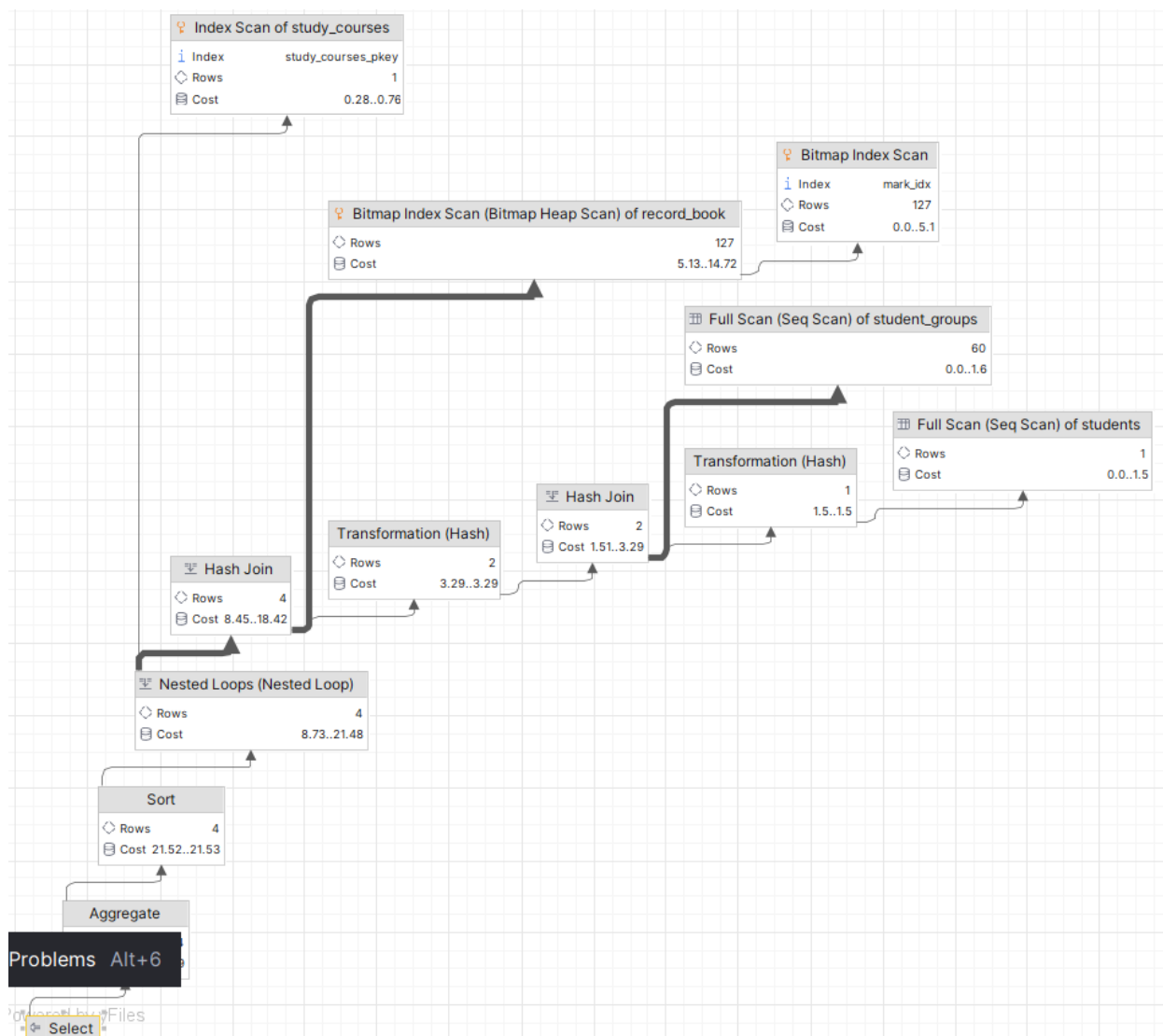


Рисунок 24 – план операции после создания индексов и добавления дополнительных студентов

План после добавления 10000 студентов (рис. 25):

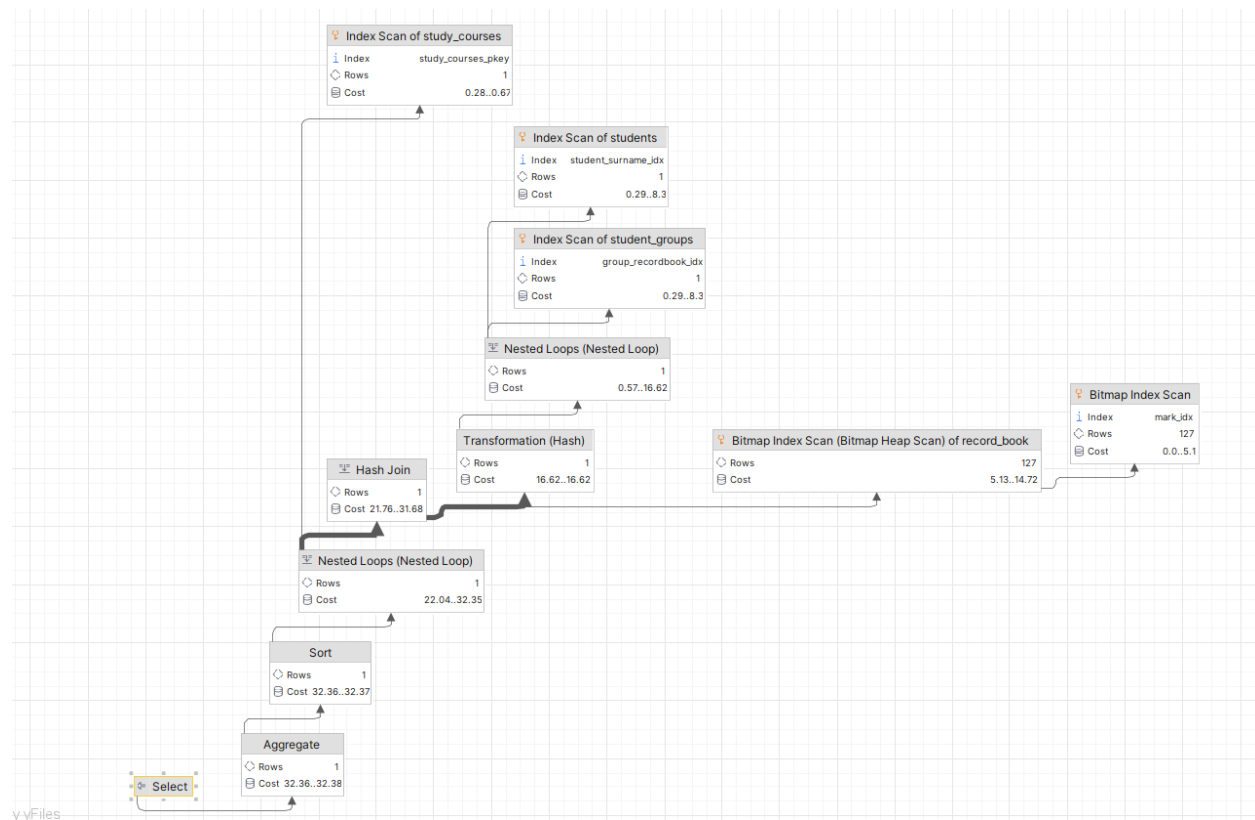


Рисунок 25 – план операции после создания индексов и добавления 10000 дополнительных студентов

Методы

№ . 1

Описание:

Повысить социальную стипендию на 10%

Код метода:

```
create or replace function increase_social_scholarship()
returns void as
$$
declare
    prev_amount int;
    scholarship_name text;
begin
    scholarship_name := 'социальная имя';
    select money_amount into prev_amount
    from scholarships
    where name=scholarship_name;
    update scholarships
    set
        money_amount = prev_amount * 1.1
    where
        name = scholarship_name;
end;
$$ language plpgsql;
```

Выполнение метода:

До (рис. 26):

	id_scholarship	name	money_amount	type_of_scholarship
1	0	социальная имя	30000.00	социальная
2	1	академическая имя	2000.00	академическая
3	2	именная имя	100000.90	именная

Рисунок 26 – выполнение метода до

После (рис. 27):

	id_scholarship	name	money_amount	type_of_scholarship
1	1	академическая имя	2000.00	академическая
2	2	именная имя	100000.90	именная
3	0	социальная имя	33000.00	социальная

Рисунок 27 – выполнение метода после

№ . 2

Описание:

Добавить новую группу в историю групп.

Код метода:

```
create or replace function
    plus_course(id_record_book_in int, id_new_group_in int)
returns void as
$$
    declare
        current_course int;
        new_group_course int;
    begin
        select max(course) into current_course from groups
        where id_group in
            (select id_group from student_groups where
id_record_book=id_record_book_in);

        select course into new_group_course from groups
        where id_group = id_new_group_in;

        if new_group_course <> current_course+1 then
            raise exception 'new course of group is not larger than
current course of group by one';
        end if;

        insert into student_groups(id_record_book, id_group)
        values (id_record_book_in, id_new_group_in);
    end;
$$ language plpgsql;
```

Выполнение метода:

Выведем историю групп студента 20. Он сейчас находится на 1-м курсе в группе 4 (рис. 28):




	 id_group_row ▾	 id_record_book ▾	 id_group ▾
1	40	20	4

Рисунок 28 – выполнение запроса

Попытаемся добавить в группу 0 с курсом 1:

```
select plus_course(20, 0)
```

Получаем ошибку:

[2023-05-06 12:26:54] [P0001] ОШИБКА: new course of group is not larger than current course of group by one

[2023-05-06 12:26:54] Где: функция PL/pgSQL plus_course(integer,integer), строка 14, оператор RAISE

Попытаемся добавить в группу 5 с курсом 2:

```
select plus_course(20, 5);
```

Посмотрим на результат (рис. 29):




	 id_group_row ▾	 id_record_book ▾	 id_group ▾
1	10042	20	5
2	40	20	4

Рисунок 29 – выполнение метода

№ 3

Описание:

Метод для изменения оценки при успешной пересдаче экзамена

Код метода:

```
create or replace function change_mark(input_id_study_course int,
input_id_group_row int, new_mark int) returns void as
$$
    DECLARE
        last_try_number int;
        var_id_teacher int;
    begin
        select max(try_number) into last_try_number from record_book
        where
```

```

        id_study_course = input_id_study_course and
        id_group_row = input_id_group_row;

select id_teacher into var_id_teacher from record_book
where
    id_study_course = input_id_study_course and
    id_group_row = input_id_group_row and
    try_number=last_try_number;

if last_try_number = 3 then
    raise exception 'Невозможно пересдать экзамен, так как были
использованы все попытки';
end if;

insert into record_book(id_group_row, id_teacher,
id_study_course, mark, try_number)
values (input_id_group_row, var_id_teacher,
input_id_study_course, new_mark, last_try_number+1);
end;
$$ language plpgsql;

```

Выполнение метода:

Посмотрим оценки для `id_group_row = 0` and `id_study_course = 1` (рис. 30):

	id_row_record_book	id_group_row	id_teacher	id_study_course	mark	try_number
1	1	0	1	1	4	2
2	401	0	1	1	2	1

Рисунок 30 – выполнение запроса

Выполним метод:

```
select change_mark(2, 0, 5);
```

Результат (рис. 31):

	id_row_record_book	id_group_row	id_teacher	id_study_course	mark	try_number
1	1	0	1	1	4	2
2	819	0	1	1	5	3
3	401	0	1	1	2	1

Рисунок 31 – выполнение метода

Посмотрим оценки для `id_group_row = 0` and `id_study_course = 2`.

Результат (рис. 32):

	<code>id_row_record_book</code>	<code>id_group_row</code>	<code>id_teacher</code>	<code>id_study_course</code>	<code>mark</code>	<code>try_number</code>
1	2	0	2	2	3	3
2	402	0	2	2	2	1
3	403	0	2	2	2	2

Рисунок 32 – результат запроса

Видим, что у студента использованы все попытки => ему нельзя пересдать предмета

Выполним метод:

```
select change_mark(2, 0, 5);
```

Получили ошибку:

[2023-05-06 15:49:24] [P0001] ERROR: Невозможно пересдать экзамен, так как были использованы все попытки

[2023-05-06 15:49:24] Где: PL/pgSQL function change_mark(integer,integer,integer) line 18 at RAISE

№4.

Описание:

У нас почему-то сбились последовательности для primary key. Мы решили написать метод, который устанавливает значение последовательностей на максимальное значение PK + 1.

Код метода:

```
CREATE OR REPLACE FUNCTION reset_serial_sequences() RETURNS void AS $$
DECLARE
    name_of_table text;
    name_of_column text;
    sequence_name text;
    max_value bigint;
BEGIN
    FOR name_of_table, name_of_column IN
        SELECT table_name, column_name FROM information_schema.columns
        WHERE column_default LIKE 'nextval%'
    LOOP
        sequence_name      :=      pg_get_serial_sequence(name_of_table,
name_of_column);

        if name_of_column is not null then
            EXECUTE format('SELECT max(%I) FROM %I', name_of_column,
name_of_table) INTO max_value;
```

```

        EXECUTE format('SELECT setval(%L, %s)', sequence_name,
max_value + 1);
    end if;

END LOOP;
END;
$$ LANGUAGE plpgsql;

```

Триггеры

№1

Описание:

Триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL

Код:

```

-- Создание таблицы для логов
create table subdivision_logs(
    id_log serial primary key,
    operation_type cud_operation not null,
    operation_time timestamp without time zone,
    affected_id_subdivision int null,
    affected_subdivision_name text null
);
-- Создание метода для триггера
CREATE OR REPLACE FUNCTION subdivision_add_to_log() RETURNS TRIGGER AS
$$
DECLARE
    var_operation_type cud_operation;
BEGIN
    IF TG_OP = 'INSERT' THEN
        var_operation_type := 'insert';
        insert into
            subdivision_logs(operation_type, operation_time,
affected_id_subdivision, affected_subdivision_name)
VALUES
            (var_operation_type, now(), new.id_subdivision, new.name);
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN

```

```

var_operation_type := 'update';
insert into
    subdivision_logs(operation_type,          operation_time,
affected_id_subdivision, affected_subdivision_name)
VALUES
    (var_operation_type, now(), old.id_subdivision, old.name);
RETURN NEW;
ELSIF TG_OP = 'DELETE' THEN
var_operation_type := 'delete';
insert into
    subdivision_logs(operation_type,          operation_time,
affected_id_subdivision, affected_subdivision_name)
VALUES
    (var_operation_type, now(), old.id_subdivision, old.name);
RETURN OLD;
END IF;
END
$$ LANGUAGE plpgsql;
-- Создание триггера
create trigger log_trigger
    after insert or update or delete on subdivisions
    for each row execute procedure subdivision_add_to_log();

```

Выполнение:

Выполнение insert (рис. 33):

	id_log	operation_type	operation_time	affected_id_subdivision	affected_subdivision_name
1	1	insert	2023-05-06 13:29:18.406493	11	new subdivision

Рисунок 33 – выполнение insert

Выполнение update (рис. 34):

	id_log	operation_type	operation_time	affected_id_subdivision	affected_subdivision_name
1	1	insert	2023-05-06 13:29:18.406493	11	new subdivision
2	2	update	2023-05-06 13:30:22.449229	9	fancy name
3	3	update	2023-05-06 13:31:02.105800	9	fancy name
4	4	update	2023-05-06 13:31:35.469823	9	name 9

Рисунок 34 – выполнение update

Выполнение delete (рис. 35):

	id_log	operation_type	operation_time	affected_id_subdivision	affected_subdivision_name
1	1	insert	2023-05-06 13:29:18.406493	11	new subdivision
2	2	update	2023-05-06 13:30:22.449229	9	fancy name
3	3	update	2023-05-06 13:31:02.105800	9	fancy name
4	4	update	2023-05-06 13:31:35.469823	9	name 9
5	5	delete	2023-05-06 13:32:23.247818	11	new subdivision

Рисунок 35 – выполнение delete

№2

Описание:

Триггер, который предотвращает вставку оценки, если дисциплина отсутствует в учебном плане группы.

Код:

```
-- Метод для триггера
create or replace function check_study_course() returns trigger as
$$
declare
    var_id_study_courses int[];
begin
    select array_agg(id_study_course) into var_id_study_courses from
study_courses
    where id_study_plan =
        (select id_study_plan from groups where id_group =
            (select id_group from student_groups where
id_group_row=new.id_group_row));

    if new.id_study_course <> any(var_id_study_courses) then
        raise EXCEPTION 'Нет такого учебного курса в учебном плане
группы';
    end if;

    return new;
end
$$ language plpgsql;
-- Создание триггера
create trigger study_course_trigger
before insert on record_book
for each row execute procedure check_study_course();
```

Выполнение:

Попытаемся поставить оценку по дисциплине, которой нет у студента в учебном плане:

```
insert into record_book(id_group_row, id_teacher, id_study_course, mark, try_number)
VALUES (0, 10, 10, 5, 1);
```

Получаем ошибку:

```
[2023-05-06 17:18:10] [P0001] ERROR: Нет такого учебного курса в учебном плане группы
[2023-05-06 17:18:10] Где: PL/pgSQL function check_study_course() line 11 at RAISE
```

Выводы

В результате выполнения проекта мы успешно реализовали базу данных для ведения учебного процесса в высшем учебном заведении. Были реализованы функции, триггеры, представления и индексы, которые значительно расширили возможности базы данных и улучшили ее эффективность.

Благодаря выполнению проекта мы расширили свои знания в области проектирования и реализации баз данных, используя PostgreSQL