МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»
Институт компьютерных наук и кибербезопасности
Высшая школа технологий искусственного интеллекта
Направление **02.03.01** : Математика и компьютерные науки

ОТЧЕТ

О выполнении курсовой работы

"Реализация базы данных для предметной области «Составление расписания на конференции»"

по дисциплине «Теоретические основы баз данных»

Обучающийся: _____ Яшнова Дарья Михайловна
группа 5130201/20002

Руководитель: _____ Попов Сергей Геннадьевич

«___»_____ 2024г

Санкт-Петербург, 2024

Содержание

1 Описание предметной области

1.1 Описание особенностей конференции

Данная предметная область я буду рассматривается, на основе информации о конкретной конференции - Всероссийских школьных Харитоновских чтениях.

- Организатором конференции выступает Всероссийский научно-исследовательский институт экспериментальной физики;
- Конференция проводится в 2 этапа: прослушивание докладов в каждой секции, олимпиада.
- Конференция проводится по нескольким секциям: Физика, Математика, Биология, Русский язык, Химия, История. Количество секций, в которых может принимать участие один школьник не ограничено;
- Существует отборочный тур, после прохождения которого лучшие из участников отбираются на конференцию;
- Баллы за каждый этап усредняются и по ним выявляются участники, занявшие 1,2,3 место;
- Участниками конференции могут быть школьники с 7 по 11 класс;
- Каждый год на конференцию приезжают около 100 участников, 20 сопровождающих и 20 членов жюри со всей России;
- Проводится в закрытом городе, всем иногородним участникам оформляются временные пропуска;
- Для проведения конференции используются помещения одной из городских школ;

Рассмотрим подробнее процесс формирования расписания конференции для всех участников:

- В конференции участвует участник и члены жюри. У каждого участника есть сопровождающий, причем один сопровождающий может сопровождать несколько участников;
- Участники и члены жюри делятся на секции. Секция включает в себя олимпиаду и доклад. Участники в каждой секции пишут олимпиаду и делают доклад. При этом можно рассказывать несколько докладов, но олимпиада может быть только одна;
- Сопровождающий регистрируется на секции, чтобы следить за порядком и слушать доклады. Члены жюри оценивают секцию, причем для одной секции может быть несколько членов жюри. Также члены жюри проверяют олимпиаду, несколько членов жюри могут проверять одну олимпиаду.

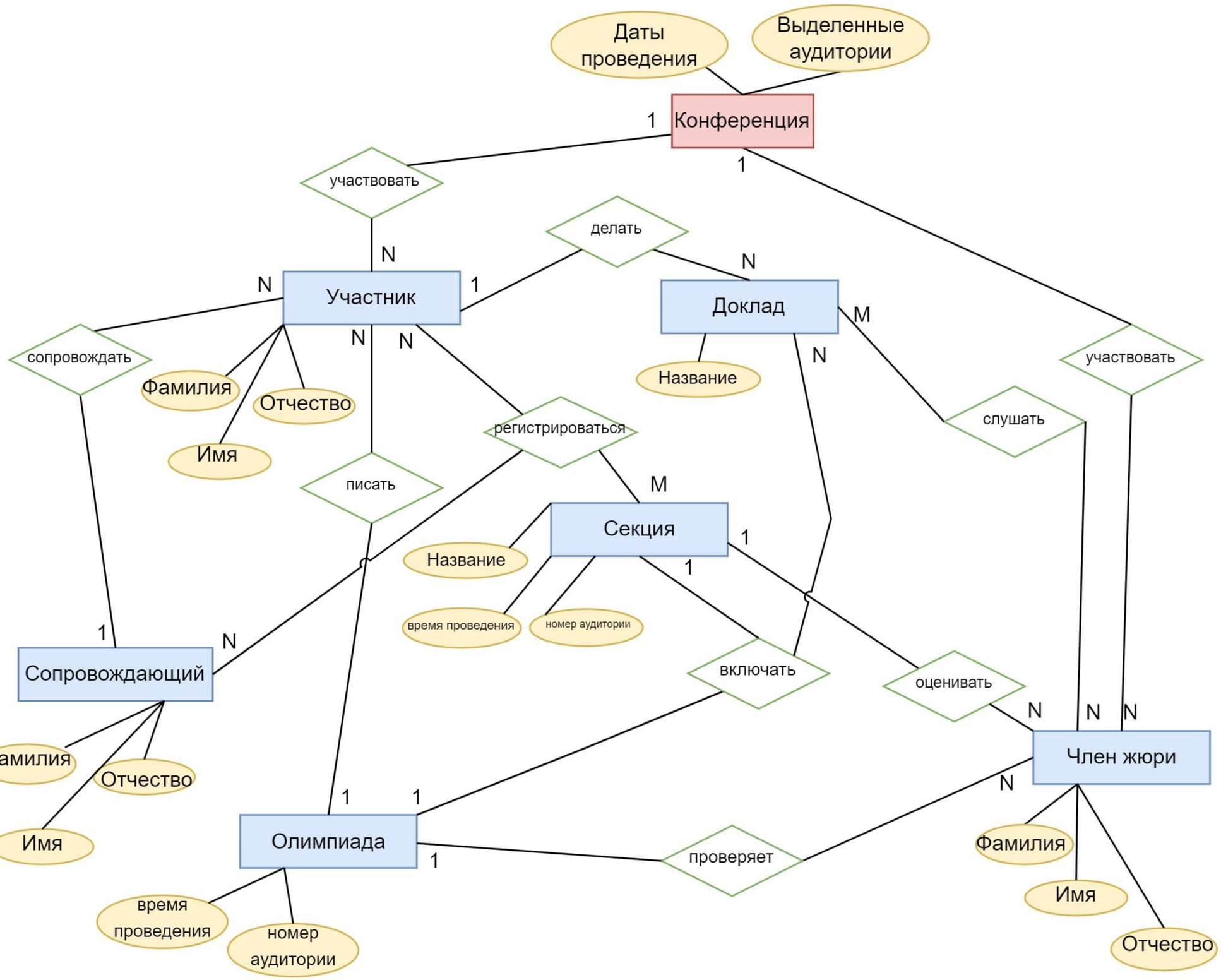


Рис.1 ER - диаграмма базы данных

1.2 ER-диаграмма

- Сопровождающий сопровождает участника;
- Сопровождающий регистрируется в секции;
- Участник участвует в конференции;
- Секция включает олимпиаду;
- Секция включает доклад;
- Участник пишет олимпиаду;
- Участник регистрируется в секции;
- Участник рассказывает в секции;
- Участник делает доклад;
- Член жюри участвует в конференции;
- Член жюри оценивает в секции;
- Член жюри слушает доклад;
- Член жюри проверяет олимпиаду;

1.3 Цели функционирования базы данных

- Информирование членов жюри о месте и времени проведения их секции;
- Предоставление личной траектории конференции для каждого участника;
- Расписание отдельных дней конференции;

1.4 Роли, и их задачи

Среди участников процесса можно выделить несколько ролей и их задачи:

- Участник.
  Задача: предоставить в оргкомитет сведения о секции доклада и названии доклада. Если докладов несколько, определиться с секцией, в которой участник будет писать олимпиаду.
- Сопровождающий.
  Задача: заявить о своем желании слушать доклады в той или иной секции.
- Член жюри.
  Задача: Явиться в секцию в день чтения докладов и проверить олимпиаду.

1.5 Сущности и их аттрибуты

- Конференция - мероприятие, включающее в себя несколько секций, на каждой из которых выступает несколько докладчиков и их оценивает несколько членов жюри.
  Аттрибуты: выделенные аудитории, даты проведения.

- Участник - школьник, который читает доклад на одной или нескольких секциях.
  Атрибуты: фамилия, имя, отчество.

- Сопровождающий - представитель школьника в рамках конференции.
  Атрибуты: фамилия, имя, отчество.

- Член жюри - эксперт, оценивающий олимпиаду и доклады.
  Атрибуты: фамилия, имя, отчество.

- Секция - мероприятие, включающее в себы чтение докладов несколькими участниками, написание олимпиады, а также оценку доклада и проверку олимпиады членами жюри.
  Атрибуты: название секции, время проведения, номер аудитории.

- Доклад - устное выступление в определенной секции.
  Атрибуты: название доклада.

- Олимпиада - письменная проверка знаний учащихся по данному предмету.
  Атрибуты: время проведения, номер аудитории.

1.6 Схема объектов базы данных

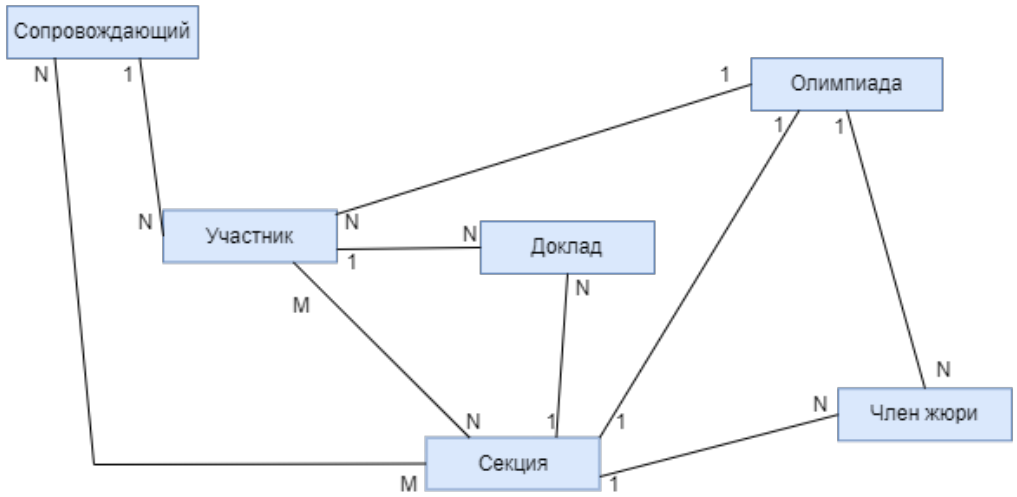


Рис. 2: Схема объектов базы данных

2 Диаграмма базы данных на русском и английском языке

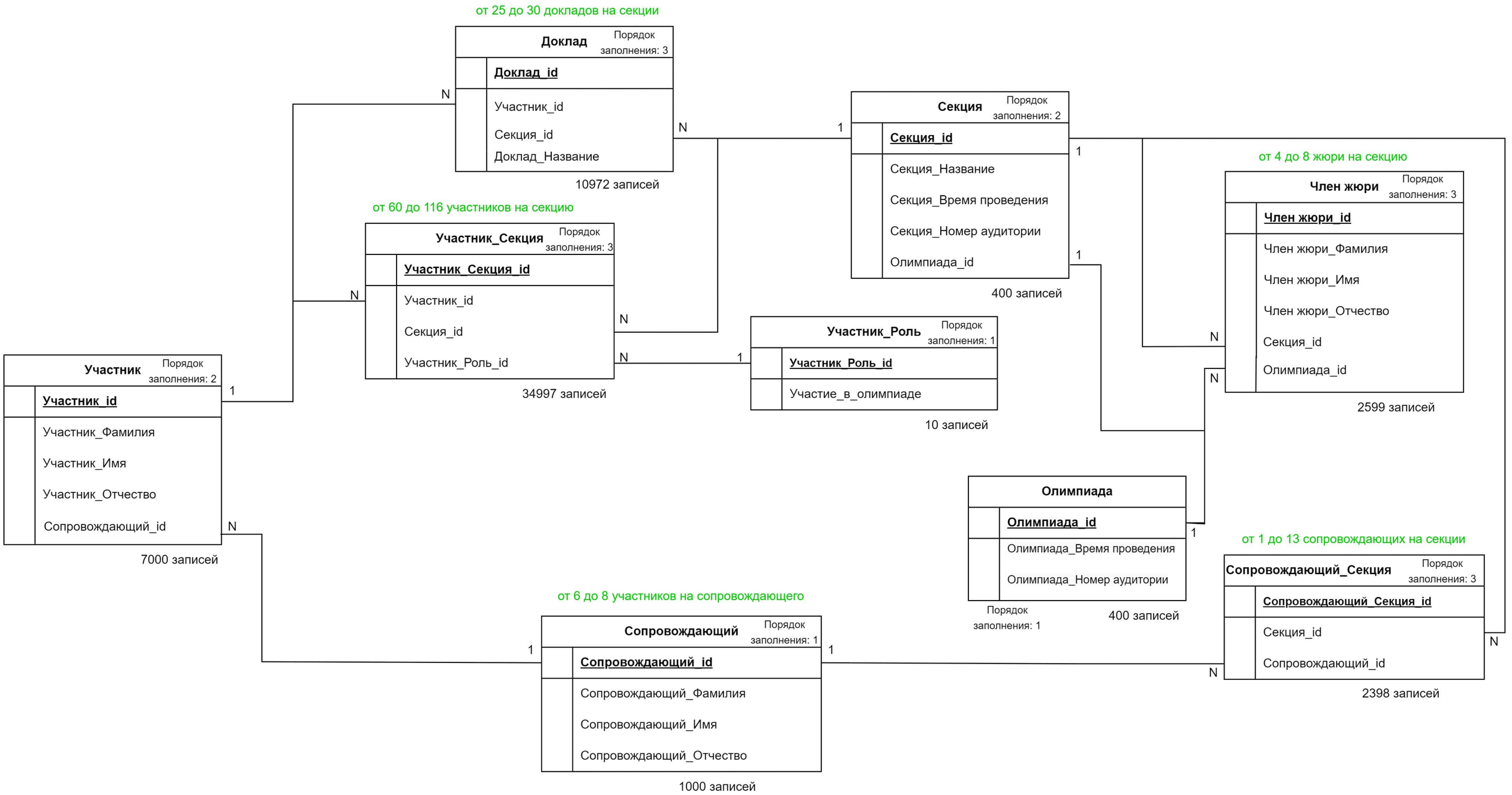


Рис.3 Диаграмма базы данных на русском языке

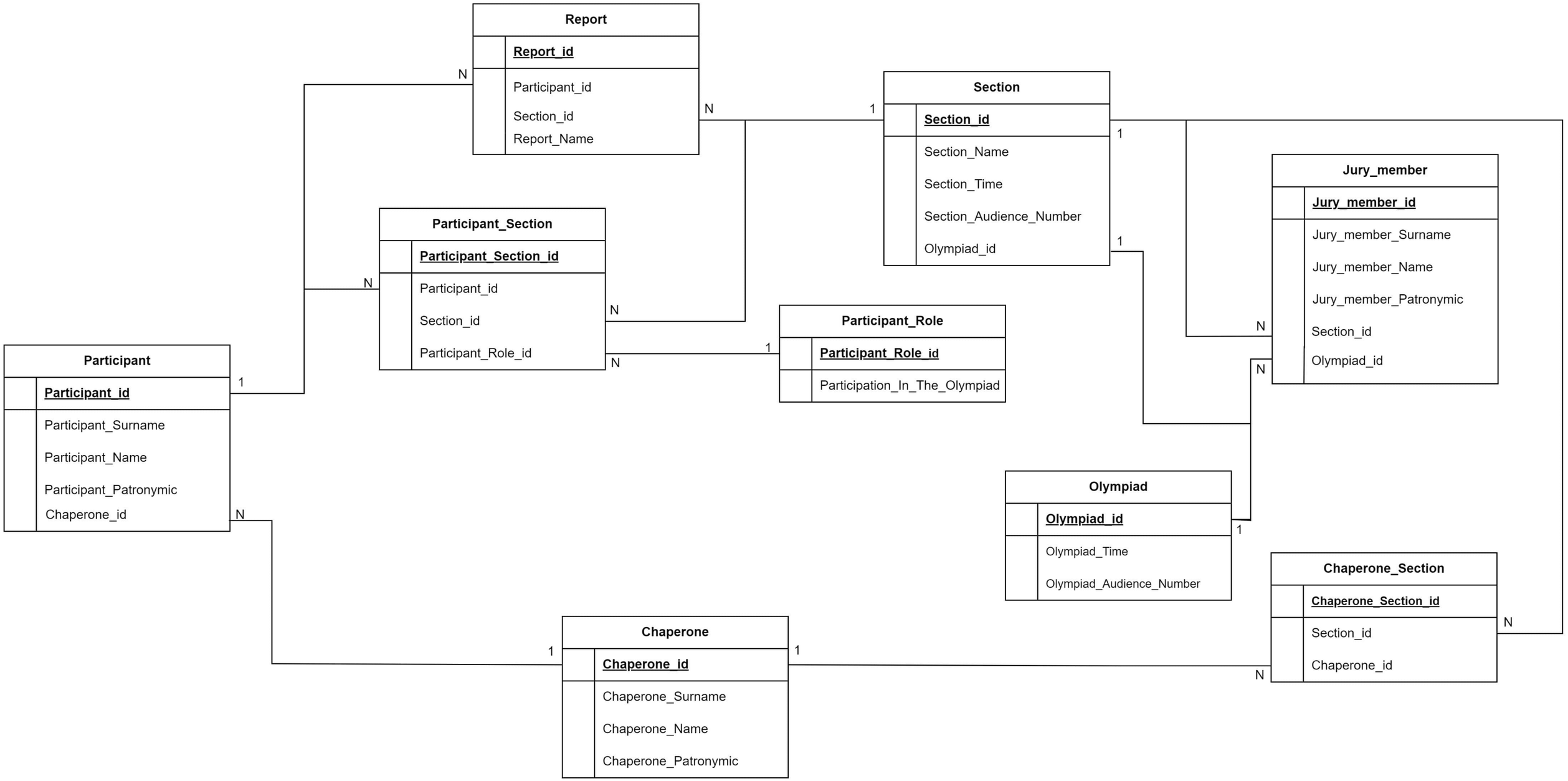


Рис.4 Диаграмма базы данных на английском языке

2.1 Таблицы метаданных

Все персональные ключи имеют тип INT, так как с добавлением записи в базу данных идентификаторы автоматически увеличиваются. Для каждого типа атрибута хранящегося в таблице есть обоснование. Фамилии, имена и отчества хранятся в строке длины до 45 символов, поскольку наибольшая длина фамилии и отчества почти совпадает и равно 43-46 символов. Так как у человека может не быть отчества, то значение этого атрибута может быть не определено. Дата и время проведения секции и олимпиады имеют тип DATETIME, поскольку этот тип как раз предназначен для манипуляций с датами и временем.

Скрипт создания таблиц базы данных находится в приложении А(4).

	Jury_member	**Член жюри**			
	Название_EN	Название_RU	Тип атрибута	NULL/NOT NULL	
PK	Jury_Member_id	Член_Жюри_id	INT	NOT NULL	
	Jury_Member_Surname	Член_Жюри_Фамилия	VARCHAR(45)	NOT NULL	
	Jury_Member_Name	Член_Жюри_Имя	VARCHAR(45)	NOT NULL	
	Jury_Member_Patronymic	Член_Жюри_Отчество	VARCHAR(45)	NULL	
	Section_id	Секция_id	INT	NOT NULL	Внешний ключ -> Секция_id
	Olympiad_id	Олимпиада_id	INT	NOT NULL	Внешний ключ -> Олимпиада_ic

Рис. 5: Таблица «Член жюри»

	Participant	**Участник**			
	Название_EN	Название_RU	Тип атрибута	NULL/NOT NULL	
PK	Participant_id	Участник_id	INT	NOT NULL	
	Participant_Surname	Участник_Фамилия	VARCHAR(45)	NOT NULL	
	Participant_Name	Участник_Имя	VARCHAR(45)	NOT NULL	
	Participant_Patronymic	Участник_Отчество	VARCHAR(45)	NULL	
	Chaperone_id	Сопровождающий_id	INT	NOT NULL	Внешний ключ -> Сопровождающий_id

Рис. 6: Таблица «Участник»

	Chaperone	**Сопровождающий**		
	Название_EN	Название_RU	Тип атрибута	NULL/NOT NULL
PK	Chaperone_id	Участник_id	INT	NOT NULL
	Chaperone_Surname	Участник_Фамилия	VARCHAR(45)	NOT NULL
	Chaperone_Name	Участник_Имя	VARCHAR(45)	NOT NULL
	Chaperone_Patronymic	Участник_Отчество	VARCHAR(45)	NULL

Рис. 7: Таблица «Сопровождающий»

	Olympiad	**Олимпиада**		
	Название_EN	Название_RU	Тип атрибута	NULL/NOT NULL
PK	Olympiad_id	Олимпиада_id	INT	NOT NULL
	Olympiad_Time	Олимпиада_Время	DATETIME	NOT NULL
	Olympiad_Audience_Number	Олимпиада_Номер_аудитории	INT	NOT NULL

Рис. 8: Таблица «Олимпиада»

	Report	Доклад			
	Название_EN	Название_RU	Тип атрибута	NULL/NOT NULL	
PK	Report_id	Доклад_id	INT	NOT NULL	
	Participant_id	Участник_id	INT	NOT NULL	
	Section_id	Секция_id	INT	NOT NULL	Внешний ключ -> Секция_id
	Report_Name	Доклад_Название	VARCHAR(100)	NOT NULL	

Рис. 9: Таблица «Доклад»

	Section	Секция			
	Название_EN	Название_RU	Тип атрибута	NULL/NOT NULL	
PK	Section_id	Секция_id	INT	NOT NULL	
	Section_Name	Секция_Название	VARCHAR(45)	NOT NULL	
	Section_Time	Секция_Время	DATETIME	NOT NULL	
	Section_Audience_Number	Секция_Номер_Аудитории	INT	NOT NULL	
	Olympiad_id	Олимпиада_id	INT	NOT NULL	Внешний ключ -> Олимпиада_id

Рис. 10: Таблица «Секция»

	Chaperone_Section	Сопровождающий_Секция			
	Название_EN	Название_RU	Тип атрибута	NULL/NOT NULL	
PK	Chaperone_Section_id	Сопровождающий_Секция_id	INT	NOT NULL	
	Section_id	Секция_id	INT	NOT NULL	Внешний ключ -> Секция_id
	Chaperone_id	Сопровождающий_id	INT	NOT NULL	Внешний ключ -> Сопровождающий_id

Рис. 11: Таблица «Сопровождающий-Секция»

	Participant_Section	Участник_Секция			
	Название_EN	Название_RU	Тип атрибута	NULL/NOT NULL	
PK	Participant_Section_id	Участник_Секция_id	INT	NOT NULL	
	Participant_id	Участник_id	INT	NOT NULL	Внешний ключ -> Участник_id
	Section_id	Секция_id	INT	NOT NULL	Внешний ключ -> Секция_id
	Participant_Role_id	Участник_Роль_id	INT	NOT NULL	Внешний ключ -> Участник_Роль_id

Рис. 12: Таблица «Участник-Секция»

	Participant_Role	Участник_Роль		
	Название_EN	Название_RU	Тип атрибута	NULL/NOT NULL
PK	Participant_Role_id	Участник_Роль_id	INT	NOT NULL
	Participation_in_the_olympiad	Участие_в_олимпиаде	BOOLEAN	NOT NULL

Рис. 13: Таблица «Участник-Роль»

2.2 Генерация записей

Для генерации записей в базе данных был написан код на языке Python, который наполняет текстовый файл командами вставки INSERT. Также в начале создается соединение с базой данных.

Для генерации текстовых полей имени, фамилии и отчества используется функция rand_name(), которая генерирует случайную строку длинной от 2 до 44 из букв латинского алфавита.

Для генерации названия доклада используется функция rand_report(int n), генерирующая случайную строку длины n, состоящую из букв латинского алфавита.

Код программы, используемой для заполнения базы данных представлен в приложении B(4).

3 Выполнение запросов

3.1 Запрос 1

Найти участников олимпиады, которых сопровождал A, и на секции судил член жюри B.

```
select Participant_Name, Participant_Surname, Chaperone_id, Section_id
from participant a
    join participant_section b
    on a.Participant_id = b.Participant_id
    where a.Chaperone_id = 61 and  b.Section_id in(
                Select Section_id
                    from jury_member
                    where Jury_Member_id = 45)
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	jury_member	NULL	const	PRIMARY,Section_idx	PRIMARY	4	const	1	100.00	NULL
1	SIMPLE	a	NULL	ref	PRIMARY,Chaperone_idx	Chaperone_idx	4	const	6	100.00	NULL
1	SIMPLE	b	NULL	ref	Participant_idx,Section_idx	Participant_idx	4	mydb.a.Participant_id	5	0.99	Using where

Рис. 14: Explain запроса 1

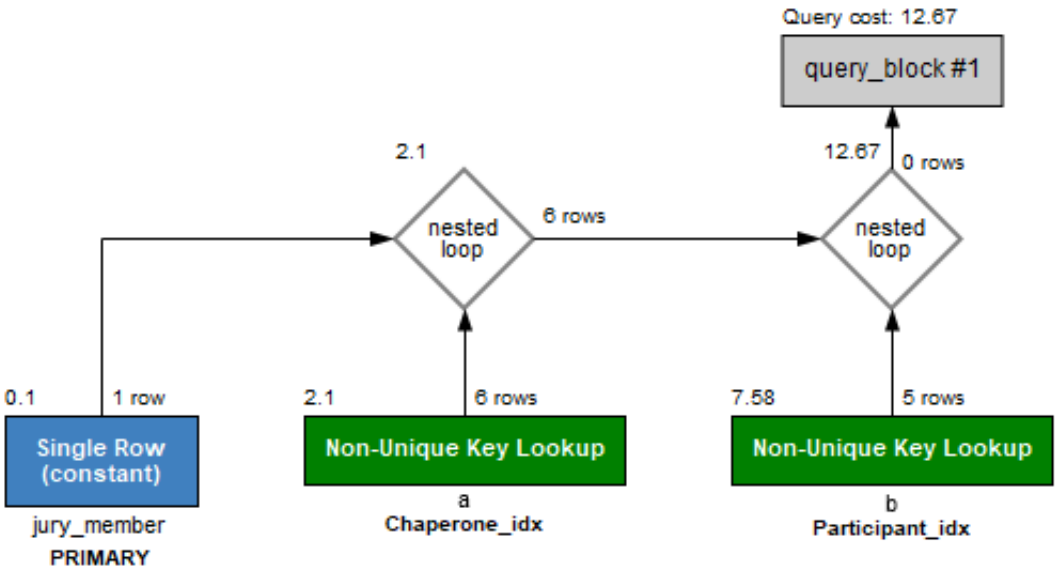


Рис. 15: Visual explain запроса 1

Данный запрос выбирает поля Participant_Name, Participant_Surname, Chaperone_id, Section_id из таблицы participant и таблицы participant_section. Затем, выполняется соединение двух таблиц по полю Participant_id.

Запрос далее фильтрует результаты, выбирая только те строки, где Chaperone_id равно 61 и Section_id находится в подзапросе, который выбирает Section_id из таблицы jury_member, где Jury_Member_id равно 45.

Время выполнения менее 0.001 сек. В результате исполнения запроса была возвращена одна запись.

Participant_Name	Participant_Surname	Chaperone_id	Section_id
MGFCUDDBBZXOYJD	NNTXEDBJFEZEHBCCCKSI	61	8

Рис. 16: Результат выполнения запроса №1

3.2 Запрос 2

Посчитать количество сопровождающих на секции A, на которой делал доклад участник B.

```
SELECT count(*)
FROM participant_section
join chaperone_section
    on chaperone_section.Section_id = participant_section.Section_id
                and participant_section.Participant_id = 5
                and participant_section.Section_id  = 167
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	participant_section	NULL	index_merge	Participant_idx,Section_idx	Participant_idx,Section_idx	4,4	NULL	1	100.00	Using intersect(Participant_idx,Section_
1	SIMPLE	chaperone_section	NULL	ref	Section_idx	Section_idx	4	const	6	100.00	Using index

Рис. 17: Explain запроса 2

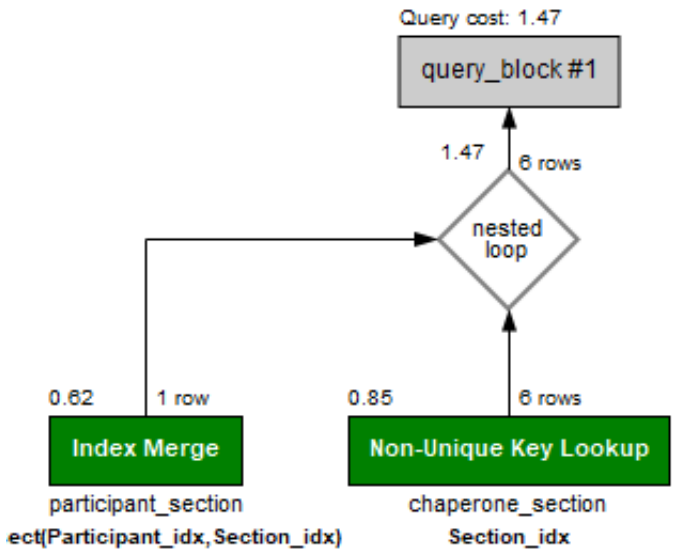
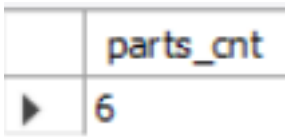


Рис. 18: Visual explain запроса 2

Данный запрос выполняет подсчет количества строк, соответствующих определенным условиям. Он соединяет таблицы participant_section и chaperone_section по трем условиям:

chaperone_section.Section_id = participant_section.Section_id participant_section.Participant_id = 5 и participant_section.Section_id = 167 Затем он подсчитывает количество строк, удовлетворяющих этим условиям, в таблице participant_section. Таким образом, данный запрос выведет количество строк, где Participant_id равен 5 и Section_id равен 167, а также соответствующие этим значениям записи из таблицы chaperone_section.

Время выполнения менее 0.001 сек.



	parts_cnt
▶	6

Рис. 19: Результат выполнения запроса №2

В результате исполнения запроса была возвращена одна запись.

3.3 Запрос 3

Для каждой роли участника посчитать число секций, в которых он участвует.

```
1  select participant_role.Participation_in_the_olympiad,
2          COUNT(DISTINCT participant_section.Section_id) as cnt_sects
3  from participant_role
4  join participant_section
5      on participant_role.Participant_Role_id = participant_section.
           Participant_Role_id
6  group by Participation_in_the_olympiad
```

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	participant_role	NULL	ALL	PRIMARY	NULL	NULL	NULL	10	100.00	Using filesort
	1	SIMPLE	participant_section	NULL	ref	Participant_Role_idx	Participant_Role_idx	4	mydb.participant_role.Participant_Role_id	3212	100.00	NULL

Рис. 20: Explain запроса 3

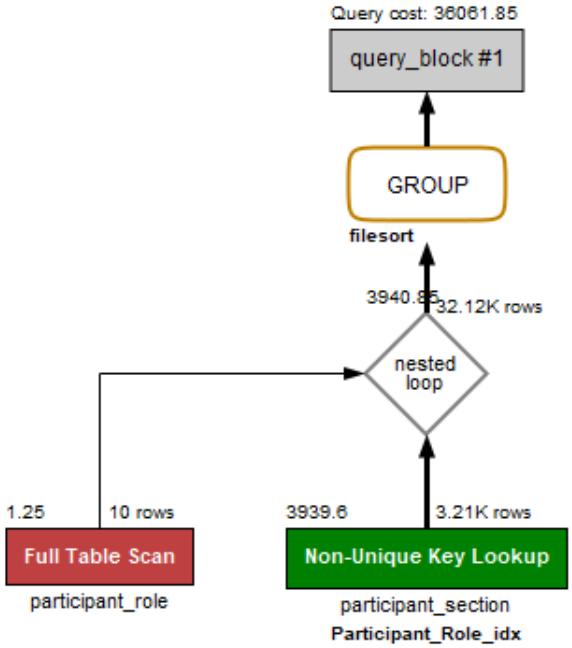


Рис. 21: Visual explain запроса 3

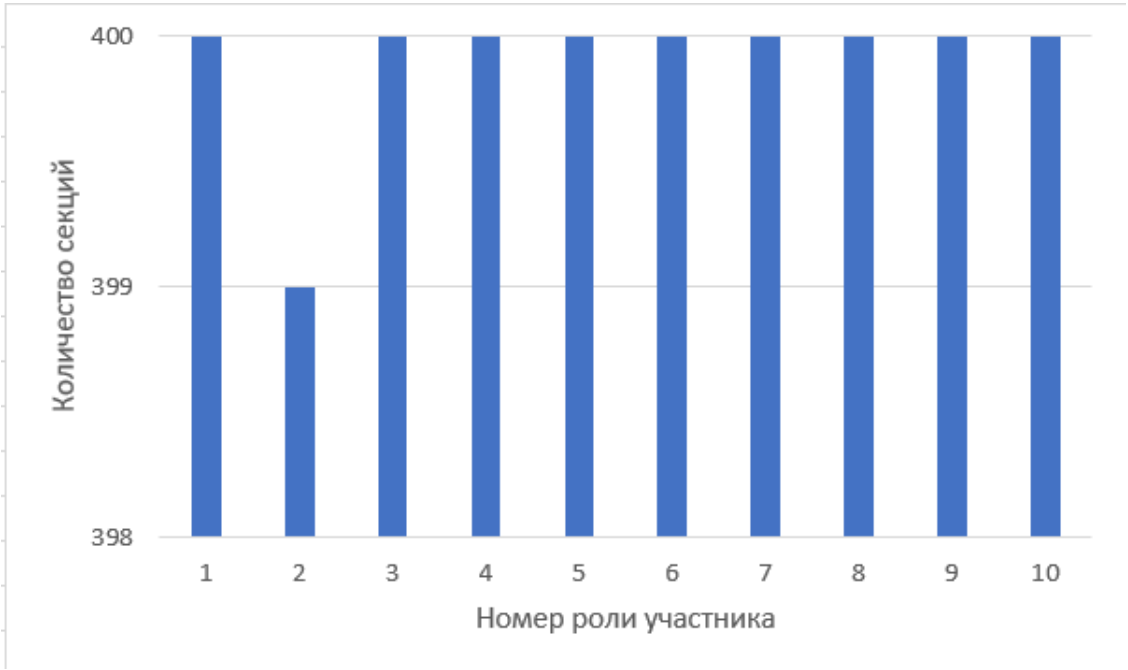


Рис. 22: Диаграмма результатов запроса 3

В этом запросе выбирается столбец Participation_in_the_olympiad из таблицы participant_role. Затем подсчитывается количество уникальных Section_id для каждой роли участника с помощью функции COUNT и ключевого слова DISTINCT. Далее происходит объединение таблиц participant_role и participant_section по ключевому столбцу Participant_Role_id. Наконец, результат сгруппирован по роли участника Participation_in_the_olympiad. Таким образом, данный запрос позволяет получить количество участников и количество разделов, в которых они участвуют, сгруппированных по их ролям в олимпиаде.

Время выполнения менее 0.001 сек.

Participation_in_the_olympiad	cnt_sects
1	400
2	399
3	400
4	400
5	400
6	400
7	400
8	400
9	400
10	400

Рис. 23: Результат выполнения запроса №3

В результате исполнения запроса было возвращено 10 записей.

3.4 Запрос 4

Посчитать число секций с одинаковым числом участников.

3.4.1 Вариант 1

```
1  with report_num as(
2                   select  report.Section_id,
3                           count(report.Report_id) as rep_num
4                   from report
5                   join section on section.Section_id = report.Section_id
6                   group by report.Section_id
7  )
8  select rep_num, count(report_num.Section_id)
9  from report_num
10 group by rep_num
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	11126	100.00	Using temporary
2	DERIVED	section	NULL	index	PRIMARY	Olympiad_idx	4	NULL	400	100.00	Using index; Using temporary
2	DERIVED	report	NULL	ref	Section_idx	Section_idx	4	mydb.section.Section_id	27	100.00	Using index

Рис. 24: Explain запроса 4.1

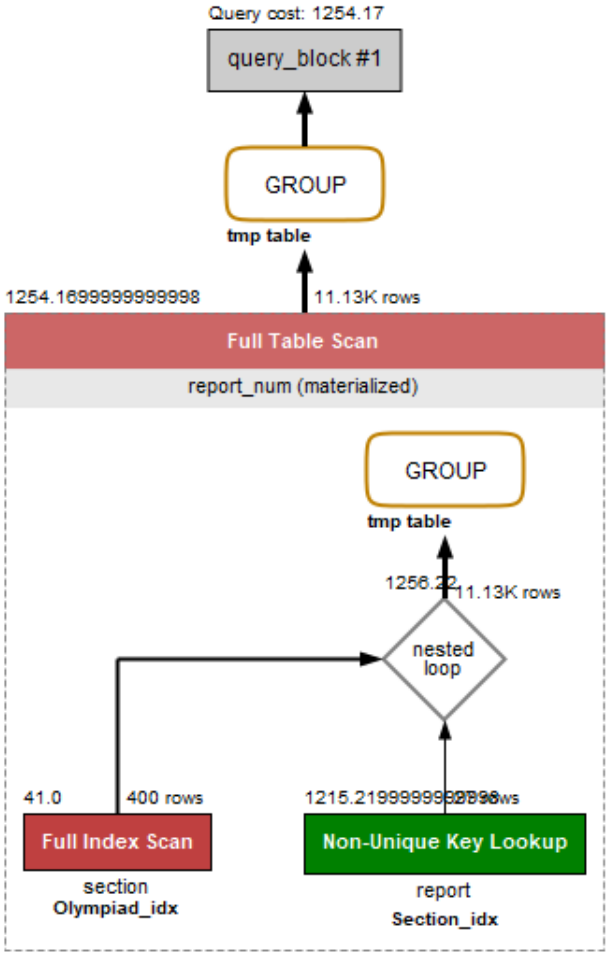


Рис. 25: Visual explain запроса 4.1

С помощью конструкции WITH создается временная таблица report_num. В этой таблице сохраняются результаты подзапроса, который выбирает Section_id из таблицы report и считается количество уникальных докладов для каждой секции. После этого результаты группируются по идентификатору секции.

Далее основной запрос выбирает количество докладов (rep_num) из временной таблицы report_num и считает количество Section_id, к которым они относятся. Результаты этого запроса также группируются по количеству докладов.

Время выполнения 0.125 сек.

3.4.2 Вариант 2

```
1  select part_number, count(Section_id)
2  from(
3       select Section_id, count(Participant_id) as part_number
4       from participant_section
5       group by Section_id) as tb
6  group by part_number
```

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	32121	100.00	Using temporary
	2	DERIVED	participant_section	NULL	index	Section_idx	Section_idx	4	NULL	32121	100.00	NULL

Рис. 26: Explain запроса 4.2

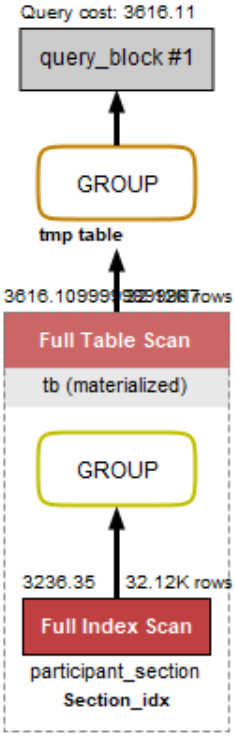


Рис. 27: Visual explain запроса 4.2

Сначала внутренний подзапрос вычисляет количество участников в каждой секции. Внешний запрос выбирает столбцы part_number и подсчитывает количество различных Section_id, которые имеют одинаковое количество участников. В результате мы получим подсчет секций, сгруппированных по количеству участников.

Время выполнения 0.172 сек. В результате исполнения запроса было возвращено 49 записей.

Рис. 28: Диаграмма результатов запроса 4

part_number	sections_num
86	14
83	10
93	12
63	1
92	18
75	9
76	10
94	13
78	11
105	6
87	24
84	14
85	16
98	4
74	4
100	4
89	21
97	11
96	7

Рис. 29: Результат выполнения запроса №4

3.5 Запрос 5

Найти олимпиады, в которых участников больше, чем у олимпиады A.

3.5.1 Вариант 1

```
1  with olymp_pts as(
2                   select Section_Name, count(Participant_id) pt_number
3                   from participant_section
4                   join section on section.Section_id =
5                           participant_section.Section_id
```

```
6                   group by participant_section.Section_id),
7
8   need_pts as(
9               select pt_number
10              from olymp_pts
11              where Section_Name = "HSDB")
12
13  select * from olymp_pts
14  where pt_number > (select * from need_pts)
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	32120	33.33	Using where
3	SUBQUERY	<derived2>	NULL	ref	<auto_key0>	<auto_key0>	137	const	10	100.00	NULL
2	DERIVED	section	NULL	ALL	PRIMARY	NULL	NULL	NULL	400	100.00	Using temporary
2	DERIVED	participant_section	NULL	ref	Section_idx	Section_idx	4	mydb.section.Section_id	80	100.00	NULL

Рис. 30: Explain запроса 5.1

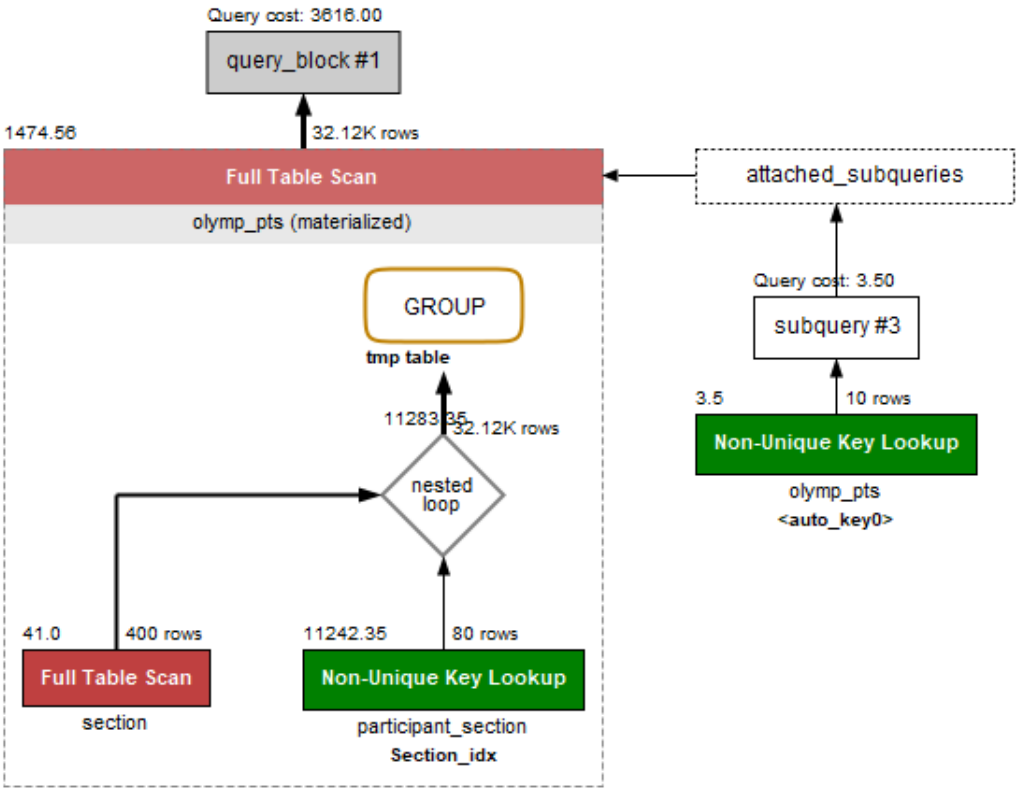


Рис. 31: Visual explain запроса 5.1

Создается временная таблица olymp_pts, которая содержит название секции и количество участников в каждой секции. Для этого используется объединение таблиц participant_section и section по соответствующим идентификаторам секции. Результат сгруппирован по идентификатору секции.

Создается временная таблица need_pts, которая содержит количество участников в олимпиаде с названием «HSDB» из таблицы olymp_pts.

Выбираются все записи из таблицы olymp_pts, где количество участников (pt_number) превышает количество участников в олимпиаде «HSDB» из таблицы need_pts.

Итак, запрос возвращает все секции, в которых количество участников больше, чем количество участников в секции «HSDB».

Время выполнения 0.141 сек.

3.5.2 Вариант 2

```
select Section_Name, count(Participant_id) pt_number
from participant_section
join section on section.Section_id =
                participant_section.Section_id
group by participant_section.Section_id
having pt_number > (select count(Participant_id) pt_number
                        from participant_section
                        join section on section.Section_id =
                                    participant_section.Section_id and
                                        Section_Name = "HSDB"
                        group by participant_section.Section_id)
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	section	NULL	ALL	PRIMARY	NULL	NULL	NULL	400	100.00	Using temporary
1	PRIMARY	participant_section	NULL	ref	Section_idx	Section_idx	4	mydb.section.Section_id	80	100.00	NULL
2	SUBQUERY	section	NULL	ALL	PRIMARY	NULL	NULL	NULL	400	10.00	Using where; Using temporary
2	SUBQUERY	participant_section	NULL	ref	Section_idx	Section_idx	4	mydb.section.Section_id	80	100.00	NULL

Рис. 32: Explain запроса 5.2

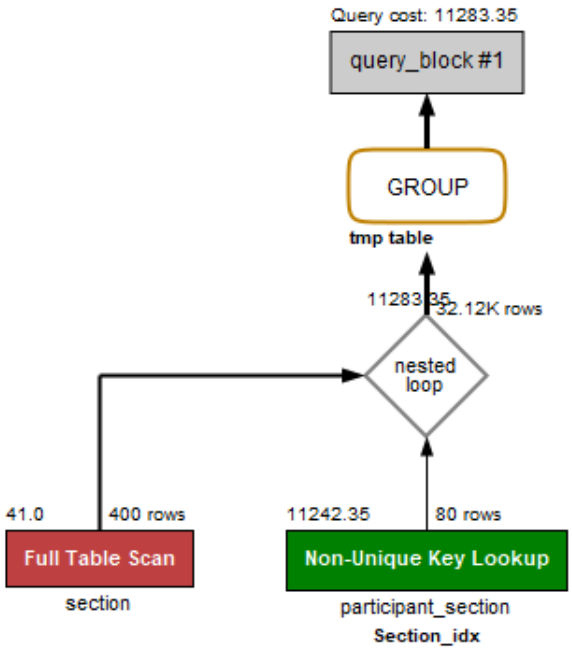


Рис. 33: Visual explain запроса 5.2

В строке SELECT выбираются столбцы Section_Name и count(Participant_id) AS pt_number из таблицы participant_section. Далее в строке FROM происходит объединение таблиц participant_section и section по условию section.Section_id = participant_section.Section_id. Затем данные группируются

по participant_section.Section_id с помощью выражения GROUP BY participant_section.Section_id. Далее используется условие HAVING, которое фильтрует результаты запроса по условию:

```
pt_number > (
        SELECT count(Participant_id) pt_number
        FROM participant_section
        JOIN section
        ON section.Section_id = participant_section.Section_id
        AND Section_Name = "HSDB"
        GROUP BY participant_section.Section_id).
```

Вложенный подзапрос внутри HAVING выбирает количество участников в разделе с названием «HSDB», группируя по Section_id. Таким образом, запрос выберет только те секции, в которых количество участников больше, чем количество участников в секции «HSDB». Время выполнения 0.125 секунд. В результате исполнения запроса было возвращено 398 записей.

Section_Name	pt_number
IQPUINHBTHZSKPHOQZIPHKYOIYOIYOKGRBKY...	86
ZYFRELSUJFWLDQSUQSBBISWVAZNTNPBAFPYEY	83
DGTIYLCRQDSEMQLBZEABWIIMYZBBM	83
XMESHWJPNCBWSXXSWTKKXBFVTRFIXGECFDKLY	93
GHDCBFLKNEMXLDE	92
IANXSZWLYNONRRQEBM	75
KYXLZMDGEAXMFF	76
LBLUACYOSKFUQTHMUVXCZZLUJKWSAXIJDNIG	94
HVFK	78
WDXGSCEXDJTDVXSUWKDDAGSSIKYSYVCWWI...	105
PPMBIKAYGYNPOQSFGHFUI	87
EWIONNDGNLDWF	84
OSHIXKNFNLAYTUXZLMPXFYFVFBFD	85
UKNJXBRLKMGQJSGVVIXNHYSPKZILXWWYZUMP...	98
IWXPOPJO	74
FVMZVC	100

Рис. 34: Результат выполнения запроса №5

3.6 Запрос 6

3.6.1 Запрос 6.1

Найти членов жюри с наибольшим числом проверенных докладов.

Вариант 1

```
1  with jury_reps as(
2                   select  Jury_Member_Surname, Jury_Member_Name,
3                   count(Report_id) as reps_num
4                   from jury_member
5                   join report
6                   on jury_member.Section_id = report.Section_id
7                   group by Jury_Member_id
8  )
9  select * from jury_reps
10 where jury_reps.reps_num = (
11 select MAX(reps_num) from jury_reps)
```

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	PRIMARY	<derived2>	NULL	ref	<auto_key0>	<auto_key0>	8	const	10	100.00	Using where
	3	SUBQUERY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	72291	100.00	NULL
	2	DERIVED	jury_member	NULL	index	PRIMARY,Section_idx,Olympiad_idx	PRIMARY	4		2599	100.00	NULL
	2	DERIVED	report	NULL	ref	Section_idx	Section_idx	4	mydb.jury_member.Section_id	27	100.00	Using index

Рис. 35: Explain запроса 6

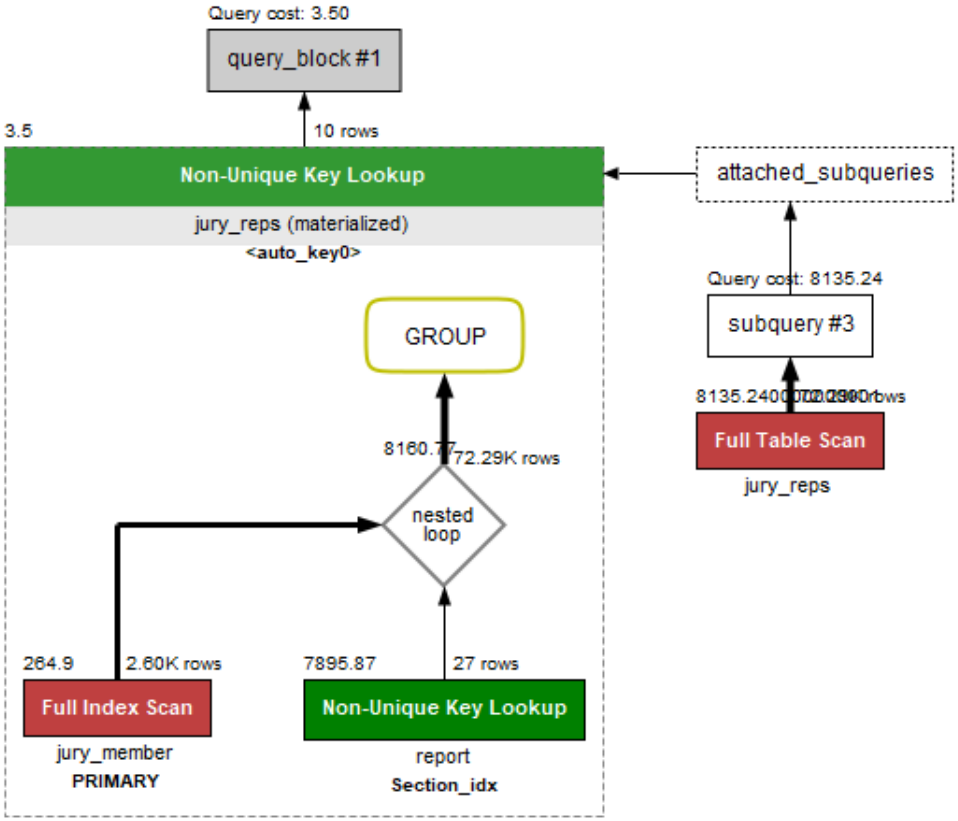


Рис. 36: Visual explain запроса 6.1

Создает временную таблицу jury_reps, где для каждого члена жюри подсчитывается количество докладов, которые он прослушал. Для этого выбираются Jury_Member_Surname, Jury_Member_Name и подсчитывается количество докладов для данного члена жюри (reps_num), сгруппированных по Jury_Member_id.

В основной части запроса выводятся все записи из временной таблицы jury_reps, где количество докладов (reps_num) равно максимальному значению, найденному во временной таблице jury_reps.

Таким образом, данный запрос возвращает информацию о членах жюри, которые прослушали максимальное количество докладов.

Время выполнения 0.063 сек.

Вариант 2

```
select Jury_Member_id, Jury_Member_Surname,
                Jury_Member_Name,
                count(Report_id) as reps_num
from jury_member
join report on jury_member.Section_id = report.Section_id
group by Jury_Member_id
```

```
7  having reps_num = (select
8                              count(Report_id) as reps_num
9                          from jury_member
10                         join report on jury_member.Section_id =
11                                        report.Section_id
12                         group by Jury_Member_id
13                         order by reps_num desc
14                         limit 1)
```

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	PRIMARY	jury_member	NULL	index	PRIMARY,Section_idx,Olympiad_idx	PRIMARY	4	NULL	2599	100.00	NULL
	1	PRIMARY	report	NULL	ref	Section_idx	Section_idx	4	mydb.jury_member.Section_id	27	100.00	Using index
	2	SUBQUERY	jury_member	NULL	index	PRIMARY,Section_idx,Olympiad_idx	PRIMARY	4	NULL	2599	100.00	Using temporary; Using filesort
	2	SUBQUERY	report	NULL	ref	Section_idx	Section_idx	4	mydb.jury_member.Section_id	27	100.00	Using index

Рис. 37: Explain запроса 6.1.2

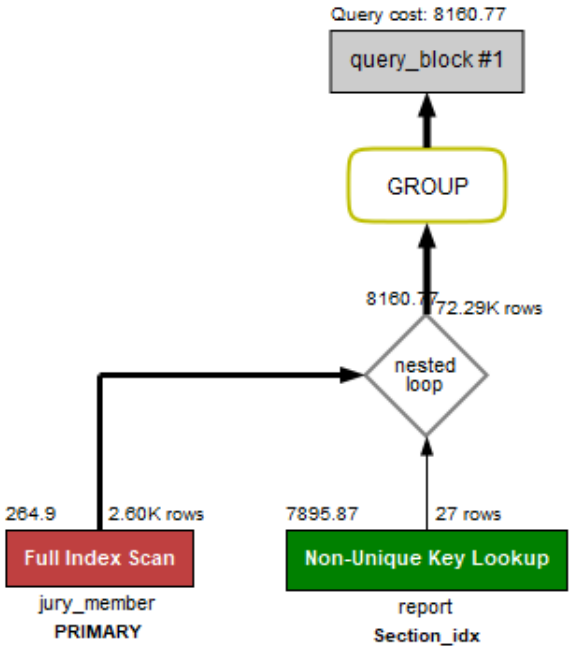


Рис. 38: Visual explain запроса 6.1.2

В первой части запроса мы выбираем данные о идентификаторе члена жюри, фамилии, имени и подсчитываем количество докладов (reps_num). Мы объединяем таблицу jury_member с таблицей report по идентификатору секции.

Затем мы группируем результат по идентификатору члена жюри.

В части «having» мы фильтруем результаты таким образом, чтобы отобрать только тех членов жюри, у которых reps_num равно максимальному количеству отчетов среди всех членов жюри.

Подзапрос в скобках возвращает максимальное количество докладов (reps_num), подсчитанное в рамках каждого члена жюри. Затем мы упорядочиваем результаты по убыванию количества отчетов и выбираем только верхнюю запись с помощью «limit 1».

Время выполнения 0.062 сек. В результате исполнения запроса было возвращено 383 записи.

	Jury_Member_id	Jury_Member_Surname	Jury_Member_Name	reps_num
▶	1175	CXSUXGYNGZLGAQRIXOLZBSTCYZUBJNRBAGNK...	ZWPSTGNQRTLRJEEQ	30
	155	ILXZGGCEXCC	ZWNRGSMXJPTPHFNGSUDYUWLTNVQDURLDM...	30
	430	GBDANDMELZCIDCOMCSLGZLMHXIUJIOODIHF...	ZUWOGKMYQIEXEBPRJQVNNKAUNZTFWSGFOZ...	30
	662	EYAVQTERCKBUVJURXSUCNRFTIVEUSXCWKHZI...	ZUJALGKZWSIDPQSZK	30
	2111	OBQGFPGKROADTBHVFHEDPBMYERWKERHQIV	ZUHZJJXCFSV	30
	107	AQQHEPLPWXU	ZOHTYTN	30
	546	XOXRZZFFNXDVMMLUFAGPWLNTAFBRKJNJPOK...	ZNAAZDEIVTWDLADFLUASQAEZW	30
	1588	PXKVDLQWUDPCKVDNWDXJYOGMHLUYVTOLDIJ...	ZLBCZAFKKOFMFSDUVYVKZGKNCVULEHD	30
	885	GTMNKJIQUGXO	ZJBDHXDMBRVYQAFFVUCAAVAEQJJPCHULYSS...	30
	963	IKXOZPIXWYWKXKAHQIVXQNUXYUONICSYX	ZHJCBCWNEZZACTJYDRGKIQMEHSUSIAGFI	30
	411	KUFLWQVDHHGR	ZGVDWAIJIBXQYQWOHZKEKH	30
	2068	LQNIUHULFXOHWAELZWIVJSQOJATIYKRAGX	ZFUQNYAXJOZFDJKXRVQEVYQB	30
	2105	JBYOSFNWOSTRA	ZDPEMUD	30
	1579	OJAGRODGUELMWERLLWKRDQPUKEYOM	ZCIFPPXZLIVQUKWXUOKRVZTQXRRTUILGC	30
	52	BYBIXZWWLFZYFEWIMZGUTHJCOOTOUAABAW...	ZAJXZFEWQEGMNANULXMZ	30
	1536	ULZJPIBMBWLPQHWIXSCHWIUERVLCMOQVPJGX	YYWFZXGHFMBWPACJGSI	30
	54	SBWFQDXCT	YYSZLOQJKOYNHQYXVMWZAGJWKVPXWJYNCGO	30

Рис. 39: Результат выполнения запроса №6.1

3.6.2 Запрос 6.2

Найти членов жюри с наименьшим числом проверенных докладов.

```
select Jury_Member_id, Jury_Member_Surname,
                Jury_Member_Name,
                count(Report_id) as reps_num
from jury_member
join report on jury_member.Section_id = report.Section_id
group by Jury_Member_id
having reps_num = (select
                           count(Report_id) as reps_num
                    from jury_member
                    join report on jury_member.Section_id =
                                   report.Section_id
                    group by Jury_Member_id
                    order by reps_num asc
                    limit 1)
```

	Jury_Member_id	Jury_Member_Surname	Jury_Member_Name	reps_num
▶	2357	ZBHFSBNDNKTHK	ZVVSB	25
	485	YRPXOLGKMTP	ZTXWKOCLCKAMHGOBSQMRBMVOUVR	25
	118	SGZIYXARJUELTMGWCMGWQWOUMOHFRK	ZROICQULYOYTWEHAEHPWAISAZLZ	25
	915	BE	ZRKHJKEGAFECCEFMGJFBDJZIBRMPA	25
	212	CWCBNMENJKVKLKCBONBZKHZGHHE	ZLBQBDKQIT	25
	2442	LVKMZVEURMWRJVPPAHJGPZBQPJVHOPMSOYX...	ZKMNWSPAELWSLAZMFPXLRRMNNUKKYG	25
	713	XPZJWJCYIRFCRTJCDSVXOPQEZNFVGAZCCWV	ZJGDJWHUHTASEDTPRUIRR	25
	879	VEFQWALEGLZKWSZQSGQKAHWYGIEVHZSMHS	ZHELJ	25
	704	BFNMIKIULUUEH	ZEGNHQFAQKDBIRQCKBNXQTTR	25
	2045	ITJDOVMNGTJ	ZCEAWUEIOTWMJEJYRUKTLYDOZBK	25
	2427	AZYFRSCAHSBKK	ZBYMMLSZVTOOLBRFWBMXLYKEFWGGHAGKYF...	25
	2182	ZMHOPQEBWQUOXYNPSOQZJYSUCZLUNTXHGA...	ZBIBLIAJLVTFGCA	25
	2041	RHQFPNVJXJYSHMBJOMRHTJJJBWGQHMJJCYC...	ZBANLTVSOXTSXXAUEQMFSFFEARSVSUOKQIJS...	25
	228	ARBTOP	YYJDEQSHJWEUCXIK	25
	1153	FUTDYEYPQ	YWK	25
	1544	ESHOBSKZ	YWEGMMNZYVAAM	25
	279	UIYRRYCGRZLUXDZMBLJGAH	YUIAGQYWPFMYGGTAOAKRWP	25
	1353	UFMHN	YSQXJDYHWREFDLFJJMPLJIFPLRPJLHBCQXUO	25

Рис. 40: Результат выполнения запроса №6.2

В результате исполнения запроса было возвращено 483 записи.

3.7 Запрос 7

Найти сопровождающих, которые не участвовали в секциях, в которых участвовали их участники.

3.7.1 Вариант 1

```
with chap_sect as(
    select chaperone.Chaperone_id, Chaperone_Name,
           Chaperone_Surname, Section_id
    from chaperone
    join chaperone_section on chaperone.Chaperone_id =
         chaperone_section.Chaperone_id
),
chap_sect_part as(
    select chap_sect.Chaperone_id, chap_sect.Chaperone_name,
            chap_sect.Chaperone_Surname,
            chap_sect.Section_id as Chap_Sect, Participant_id
    from chap_sect
    join participant on participant.Chaperone_id =
                        chap_sect.Chaperone_id
),
chap_sect_part_sect as (
    select chap_sect_part.Chaperone_id, chap_sect_part.Chaperone_name,
            chap_sect_part.Chaperone_Surname, Chap_Sect,
            chap_sect_part.Participant_id,
            participant_section.Section_id as Part_Sect
    from chap_sect_part
    join participant_section on participant_section.Participant_id =
                                chap_sect_part.Participant_id
),
same_sects as(
        select Chaperone_id from chap_sect_part_sect
        where Chap_Sect = Part_Sect)

select Chaperone_Name, Chaperone_Surname, Section_id
from chap_sect
where chap_sect.Chaperone_id not in (
                            select * from same_sects)
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	chaperone	NULL	ALL	PRIMARY	NULL	NULL	NULL	1000	100.00	NULL
1	SIMPLE	chaperone	NULL	eq_ref	PRIMARY	PRIMARY	4	mydb.chaperone.Chaperone_id	1	100.00	Using where; Not exists; Using index
1	SIMPLE	chaperone_section	NULL	ref	Section_idx,Chaperone_idx	Chaperone_idx	4	mydb.chaperone.Chaperone_id	2	100.00	NULL
1	SIMPLE	participant	NULL	ref	PRIMARY,Chaperone_idx	Chaperone_idx	4	mydb.chaperone.Chaperone_id	6	100.00	Using index
1	SIMPLE	participant_section	NULL	ref	Participant_idx,Section_idx	Participant_idx	4	mydb.participant.Participant_id	5	100.00	Using where
1	SIMPLE	chaperone_section	NULL	ref	Chaperone_idx	Chaperone_idx	4	mydb.chaperone.Chaperone_id	2	100.00	NULL

Рис. 41: Explain запроса 7.1

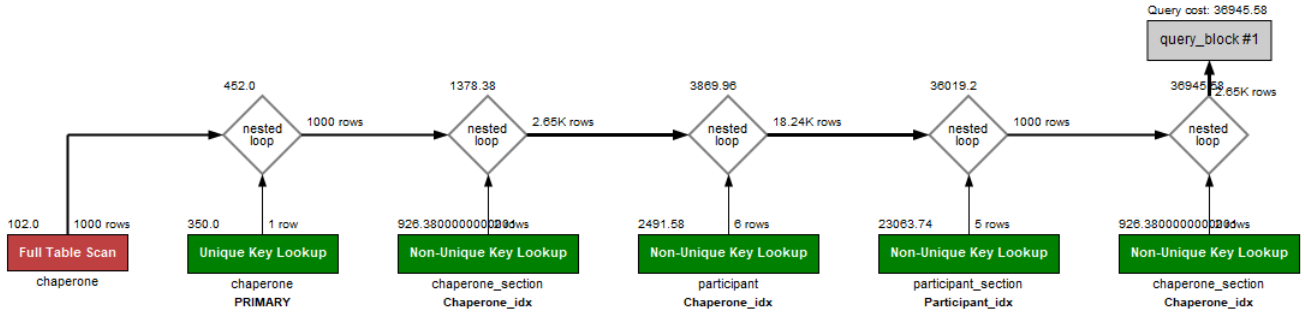


Рис. 42: Visual explain запроса 7.2

Создается временная таблица chap_sect, которая содержит информацию о сопровождающих и секциях, в которых они зарегистрированы. Далее создается временная таблица chap_sect_part, которая содержит информацию о сопровождающих, участвующих в секциях, и участниках, которых они сопровождают. После этого создается временная таблица chap_sect_part_sect, которая содержит информацию о слпровождающих, участвующих в секциях, участниках и секциях, в которых они участвуют.

Далее создается временная таблица same_sects, которая содержит идентификаторы сопровождающих, у которых секции совпадают с секциями участников, которых они сопровождают и извлекается информация о сопровождающих (имя, фамилия, идентификатор секции), которые не сопровождают участников в тех же секциях, в которых они участвуют.

Время выполнения 0.031 сек.

3.7.2 Вариант 2

```
1  select distinct Chaperone_id
2  from chaperone
3  where Chaperone_id not in(
4                  select distinct participant.Chaperone_id
5                  from participant
6                  join participant_section
7                  on participant_section.Participant_id =
8                     participant.Participant_id
9                  join chaperone_section
10                        on chaperone_section.Chaperone_id =
11                        participant.Chaperone_id
12                 and  participant_section.Section_id =
13                 chaperone_section.Section_id)
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	chaperone	NULL	index	PRIMARY	PRIMARY	4	NULL	1000	100.00	Using index; Using temporary
1	SIMPLE	chaperone_section	NULL	ref	Section_idx,Chaperone_idx	Chaperone_idx	4	mydb.chaperone.Chaperone_id	2	100.00	Using where; Not exists; Distinc
1	SIMPLE	participant	NULL	ref	PRIMARY,Chaperone_idx	Chaperone_idx	4	mydb.chaperone.Chaperone_id	6	100.00	Using index; Distinct
1	SIMPLE	participant_section	NULL	ref	Participant_idx,Section_idx	Participant_idx	4	mydb.participant.Participant_id	5	100.00	Using where; Distinct

Рис. 43: Explain запроса 7.2

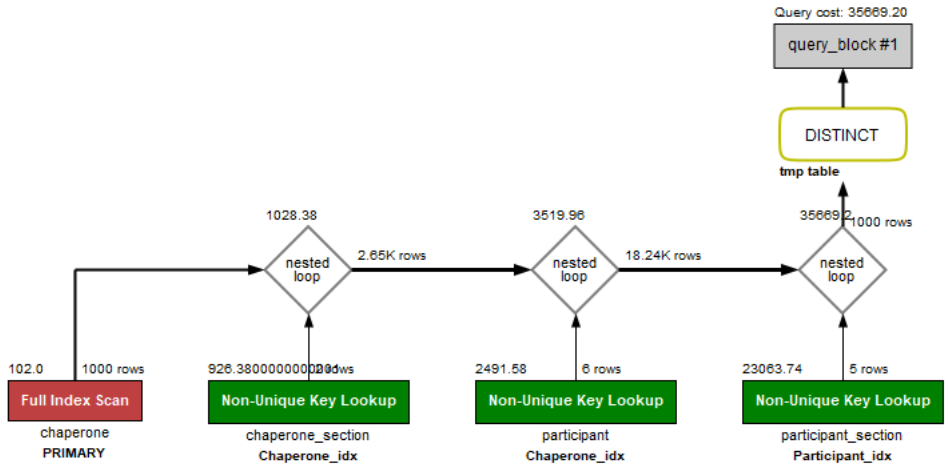


Рис. 44: Visual explain запроса 7.2

Сначала идет запрос, который выбирает уникальные значения поля Chaperone_id из таблицы chaperone. Затем идет подзапрос, который выбирает уникальные комбинации полей Chaperone_id, Participant_id, Section_id из таблиц participant, participant_section и chaperone_section. Здесь мы присоединяем таблицы participant_section и chaperone_section к таблице participant по полям Participant_id и Chaperone_id соответственно, а также по полю Section_id. После этого, в условии not in мы проверяем, что Chaperone_id из таблицы chaperone не присутствует в полученном наборе данных.

Время выполнения запроса 0.016 секунд. В результате исполнения запроса было возвращено 833 записи.

Chaperone_Name	Chaperone_Surname
GTISIVFUBQLXNADADQ	RMAKQSZWDFDJZSKGHTUU
PXUFQDD	NMXSG
CCXFEEPBSJVWTIDIQGIXZBQVWDEMUCM	IZTHIQZBUZUWF
KCWDNUKFKMVITEFOAFKYXFTRVUAQ	JZCITURSQPVSJDISVYCLAVMWNWZX
DCYIUTZR	IKPTSTIIIGNQLACE
ANLQKWGHPEETQDZILGSYHQJDBKDFZCGCYKO...	VQMMQPZNGJHPQSWLNYFGUOYYXCWRVTBUX...
CBIXQKCQSJNXZZDRLBW	XLRGVN
XRWVYENSNITACBNZQLHIZIBFSRPDSYWUXUXX...	SCIUKVJFNGKWQZDJS
YPQZARBXWVUQHX	DKTYSUVMXGUESYBGXHSFGDXUWFNJMAYVPFQ...
SAILGDHURMAJSUURIXCOINHUIKICVJEDLLNFQ...	KJAELLCCZVHWYHFX
VXUDSZWWXGWO	KXYSQJJOBCGHFJAEGETKCWRPTE
CLLDHPHWAZRMQCZMBCGSQDMPIHTWOBBZX...	WOXAKUQTHAEVPRQETCIURGJRMCAVAXCDM...
MKDQCWERNVFLACTPVHPJTAHKYSQBVM	JJQJLWRWYEHSJBKBJDYEGFXVFTSLSHLPPEEZX
QEJLFXDEEPLTNVBYGJAME	FRTNICMDHNTGEBNBQDEWUNWGESNSWP
PXWZJDMNV	XDQRRUGHRKOIDYMVXCNCIICNWGZYVICP

Рис. 45: Результат выполнения запроса №7

3.8 Запрос 8

Для каждой роли участника и каждой олимпиады посчитать число участников.

```
1  select participant_role.Participant_Role_id as Role_, section.Section_
       id as sect,
2                          coalesce((select count(Participant_id) as part
                               _cnt
3                          from participant_section
```

```
4                join participant_role on participant_role.Participant_Role
                    _id = participant_section.Participant_Role_id
5                where sect = participant_section.Section_id and
                    participant_role.Participant_Role_id = Role_
6            )
7                                , 0)as count_parts
8   from participant_role, section
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	participant_role	NULL	index	NULL	PRIMARY	4	NULL	10	100.00	Using index
1	PRIMARY	section	NULL	index	NULL	Olympiad_idx	4	NULL	400	100.00	Using index; Using
1	PRIMARY	<derived2>	NULL	ref	<auto_key0>	<auto_key0>	8	mydb.participant_role.Participant_Role_id,myd...	321	100.00	NULL
2	DERIVED	participant_role	NULL	index	PRIMARY	PRIMARY	4	NULL	10	100.00	Using index; Using
2	DERIVED	participant_section	NULL	ref	Participant_Role_idx	Participant_Role_idx	4	mydb.participant_role.Participant_Role_id	3212	100.00	NULL

Рис. 46: Explain запроса 8

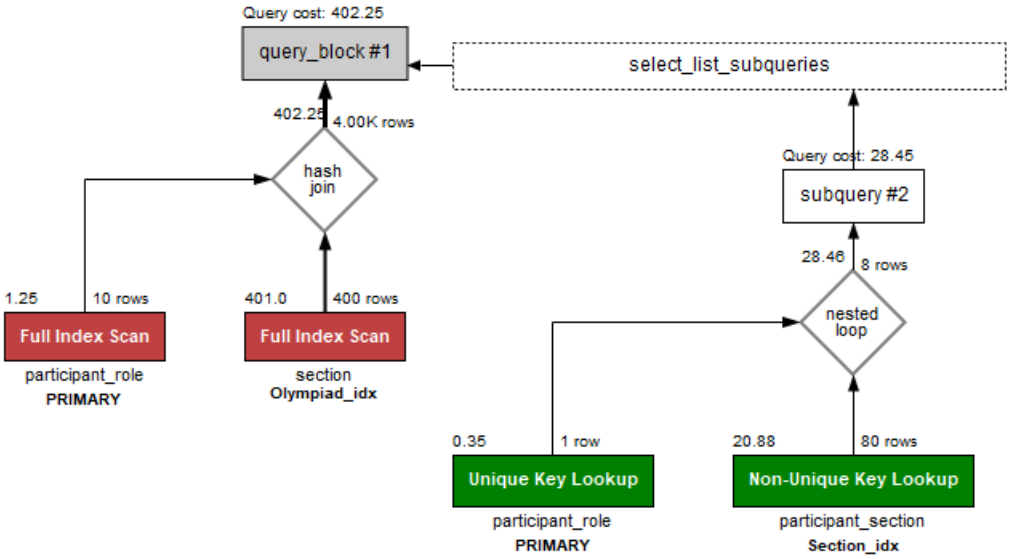


Рис. 47: Visual explain запроса 8

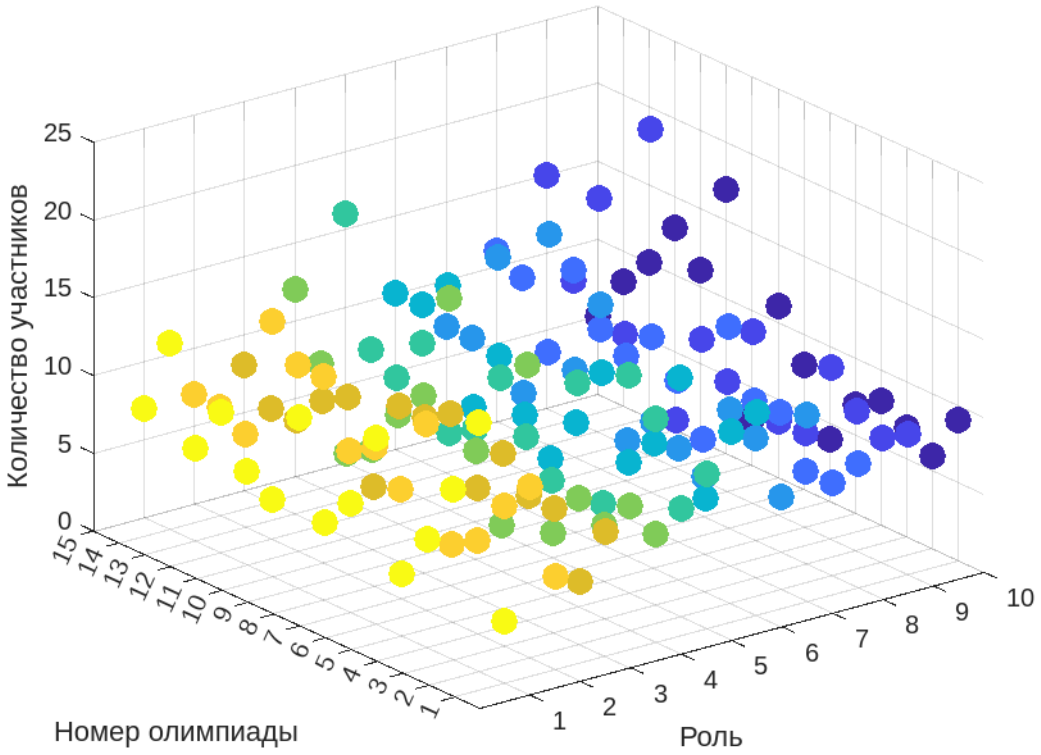


Рис. 48: Диаграмма результатов запроса 8

Сначала происходит декартово произведение таблиц participant_role и section, чтобы получить все возможные комбинации значений из обеих таблиц.

Затем происходит левое объединение с подзапросом, который сначала получает количество участников для определенной роли участника и секции. Затем результат этого подзапроса соединяется с основным запросом по секции и роли участника.

Время выполнения 0.328 сек. В результате исполнения запроса было возвращено 4000 записей.

Role_	olympiad_number	part_number
10	1	9
9	1	9
8	1	8
7	1	12
6	1	13
5	1	10
4	1	7
3	1	8
2	1	6
1	1	4
10	2	6
9	2	8
8	2	6
7	2	6
6	2	11
5	2	7
4	2	8
3	2	4
2	2	11

Рис. 49: Результат выполнения запроса №8

4 Заключение

В рамках исследования был проведён анализ выбранной предметной области, после чего были составлены ER-диаграмма, диаграмма объектов базы данных и таблицы метаданных. На основе диаграммы объектов базы данных были созданы таблицы в MySQL, которые затем были заполнены с использованием программы на языке Python. После этого было выполнено восемь запросов к базе данных, некоторые из них представлены в 2 вариантах и были оптимизированы путём использования вложенных SELECT, удаления временных таблиц и лишних операторов SELECT. В запросе 6.2 вместо функции MAX использовалась конструкция с LIMIT 1 и сортировкой таблицы.

Работа выполнялась с использованием диалекта MySQL, среды программирования Visual Studio Code, Python версии 3.8.1, командная строки и MySQL Workbench 8.0 для работы с таблицами. Диаграммы для запросов были построены в Excel и Matlab.

Приложение А. Код создания таблиц

```
1
2  CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8mb3 ;
3  USE `mydb` ;
4
5
6  CREATE TABLE IF NOT EXISTS `mydb`.`chaperone` (
7    `Chaperone_id` INT NOT NULL AUTO_INCREMENT,
8    `Chaperone_Surname` VARCHAR(45) NOT NULL,
9    `Chaperone_Name` VARCHAR(45) NOT NULL,
10   `Chaperone_Patronymic` VARCHAR(45) NULL DEFAULT NULL,
11   PRIMARY KEY (`Chaperone_id`))
12 ENGINE = InnoDB
13 AUTO_INCREMENT = 671
14 DEFAULT CHARACTER SET = utf8mb3;
15
16
17 CREATE TABLE IF NOT EXISTS `mydb`.`olympiad` (
18   `Olympiad_id` INT NOT NULL AUTO_INCREMENT,
19   `Olympiad_Time` DATETIME NOT NULL,
20   `Olympiad_Audience_Number` INT NOT NULL,
21   PRIMARY KEY (`Olympiad_id`))
22 ENGINE = InnoDB
23 DEFAULT CHARACTER SET = utf8mb3;
24
25
26 CREATE TABLE IF NOT EXISTS `mydb`.`section` (
27   `Section_id` INT NOT NULL AUTO_INCREMENT,
28   `Section_Name` VARCHAR(45) NOT NULL,
29   `Section_Time` DATETIME NOT NULL,
30   `Section_Audience_Number` INT NOT NULL,
31   `Olympiad_id` INT NOT NULL,
32   PRIMARY KEY (`Section_id`),
33   INDEX `Olympiad_idx` (`Olympiad_id` ASC) VISIBLE,
34   CONSTRAINT `Sec-Olymp`
35     FOREIGN KEY (`Olympiad_id`)
36     REFERENCES `mydb`.`olympiad` (`Olympiad_id`)
37     ON DELETE CASCADE
38     ON UPDATE CASCADE)
39 ENGINE = InnoDB
40 DEFAULT CHARACTER SET = utf8mb3;
41
42
43 CREATE TABLE IF NOT EXISTS `mydb`.`chaperone_section` (
44   `Chaperone_Section_id` INT NOT NULL AUTO_INCREMENT,
```

```
45    `Section_id` INT NOT NULL,
46    `Chaperone_id` INT NOT NULL,
47    PRIMARY KEY (`Chaperone_Section_id`),
48    INDEX `Section_idx` (`Section_id` ASC) VISIBLE,
49    INDEX `Chaperone_idx` (`Chaperone_id` ASC) VISIBLE,
50    CONSTRAINT `Chap-Sec`
51      FOREIGN KEY (`Section_id`)
52      REFERENCES `mydb`.`section` (`Section_id`)
53      ON DELETE CASCADE
54      ON UPDATE CASCADE,
55    CONSTRAINT `Sec-Chap`
56      FOREIGN KEY (`Chaperone_id`)
57      REFERENCES `mydb`.`chaperone` (`Chaperone_id`)
58      ON DELETE CASCADE
59      ON UPDATE CASCADE)
60  ENGINE = InnoDB
61  DEFAULT CHARACTER SET = utf8mb3;
62
63
64
65  CREATE TABLE IF NOT EXISTS `mydb`.`jury_member` (
66    `Jury_Member_id` INT NOT NULL AUTO_INCREMENT,
67    `Jury_Member_Surname` VARCHAR(45) NOT NULL,
68    `Jury_Member_Name` VARCHAR(45) NOT NULL,
69    `Jury_Member_Patronymic` VARCHAR(45) NULL DEFAULT NULL,
70    `Section_id` INT NOT NULL,
71    `Olympiad_id` INT NOT NULL,
72    PRIMARY KEY (`Jury_Member_id`),
73    INDEX `Section_idx` (`Section_id` ASC) VISIBLE,
74    INDEX `Olympiad_idx` (`Olympiad_id` ASC) VISIBLE,
75    CONSTRAINT `Jury-Olymp`
76      FOREIGN KEY (`Olympiad_id`)
77      REFERENCES `mydb`.`olympiad` (`Olympiad_id`)
78      ON DELETE CASCADE
79      ON UPDATE CASCADE,
80    CONSTRAINT `Jury-Sec`
81      FOREIGN KEY (`Section_id`)
82      REFERENCES `mydb`.`section` (`Section_id`)
83      ON DELETE CASCADE
84      ON UPDATE CASCADE)
85  ENGINE = InnoDB
86  DEFAULT CHARACTER SET = utf8mb3;
87
88
89  CREATE TABLE IF NOT EXISTS `mydb`.`participant` (
90    `Participant_id` INT NOT NULL AUTO_INCREMENT,
91    `Participant_Surname` VARCHAR(45) NOT NULL,
92    `Participant_Name` VARCHAR(45) NOT NULL,
93    `Participant_Patronymic` VARCHAR(45) NULL DEFAULT NULL,
94    `Olympiad_id` INT NOT NULL,
95    `Chaperone_id` INT NOT NULL,
96    PRIMARY KEY (`Participant_id`, `Olympiad_id`, `Chaperone_id`),
97    INDEX `Olympiad_idx` (`Olympiad_id` ASC) VISIBLE,
98    INDEX `Chaperone_idx` (`Chaperone_id` ASC) VISIBLE,
99    CONSTRAINT `Part-Chap`
100     FOREIGN KEY (`Chaperone_id`)
101     REFERENCES `mydb`.`chaperone` (`Chaperone_id`)
102     ON DELETE CASCADE
103     ON UPDATE CASCADE,
104   CONSTRAINT `Part-Olymp`
```

```
105       FOREIGN KEY (`Olympiad_id`)
106       REFERENCES `mydb`.`olympiad` (`Olympiad_id`)
107       ON DELETE CASCADE
108       ON UPDATE CASCADE)
109   ENGINE = InnoDB
110   DEFAULT CHARACTER SET = utf8mb3;
111
112
113
114   CREATE TABLE IF NOT EXISTS `mydb`.`participant_role` (
115     `Participant_Role_id` INT NOT NULL AUTO_INCREMENT,
116     `Participation_in_the_olympiad` TINYINT NOT NULL,
117     PRIMARY KEY (`Participant_Role_id`))
118   ENGINE = InnoDB
119   DEFAULT CHARACTER SET = utf8mb3;
120
121
122   CREATE TABLE IF NOT EXISTS `mydb`.`participant_section` (
123     `Participant_Section_id` INT NOT NULL AUTO_INCREMENT,
124     `Participant_id` INT NOT NULL,
125     `Section_id` INT NOT NULL,
126     `Participant_Role_id` INT NOT NULL,
127     PRIMARY KEY (`Participant_Section_id`),
128     INDEX `Participant_Role_idx` (`Participant_Role_id` ASC) VISIBLE,
129     INDEX `Participant_idx` (`Participant_id` ASC) VISIBLE,
130     INDEX `Section_idx` (`Section_id` ASC) VISIBLE,
131     CONSTRAINT `Part-Sec`
132       FOREIGN KEY (`Section_id`)
133       REFERENCES `mydb`.`section` (`Section_id`)
134       ON DELETE CASCADE
135       ON UPDATE CASCADE,
136     CONSTRAINT `Participant_Role`
137       FOREIGN KEY (`Participant_Role_id`)
138       REFERENCES `mydb`.`participant_role` (`Participant_Role_id`)
139       ON DELETE CASCADE
140       ON UPDATE CASCADE,
141     CONSTRAINT `Sec-Part`
142       FOREIGN KEY (`Participant_id`)
143       REFERENCES `mydb`.`participant` (`Participant_id`)
144       ON DELETE CASCADE
145       ON UPDATE CASCADE)
146   ENGINE = InnoDB
147   DEFAULT CHARACTER SET = utf8mb3;
148
149
150   CREATE TABLE IF NOT EXISTS `mydb`.`report` (
151     `Report_id` INT NOT NULL AUTO_INCREMENT,
152     `Participant_id` INT NOT NULL,
153     `Section_id` INT NOT NULL,
154     `Report_name` VARCHAR(100) NOT NULL,
155     PRIMARY KEY (`Report_id`),
156     INDEX `Participant_idx` (`Participant_id` ASC) VISIBLE,
157     INDEX `Section_idx` (`Section_id` ASC) VISIBLE,
158     CONSTRAINT `Rep-Part`
159       FOREIGN KEY (`Participant_id`)
160       REFERENCES `mydb`.`participant` (`Participant_id`)
161       ON DELETE CASCADE
162       ON UPDATE CASCADE,
163     CONSTRAINT `Rep-Sec`
164       FOREIGN KEY (`Section_id`)
```

```
165       REFERENCES `mydb`.`section` (`Section_id`)
166       ON DELETE CASCADE
167       ON UPDATE CASCADE)
168   ENGINE = InnoDB
169   DEFAULT CHARACTER SET = utf8mb3;
```

Приложение В.Код заполнения таблиц

```
1       import mysql.connector
2   import random
3   from random import choice
4   from string import ascii_uppercase
5   import datetime
6   import time
7
8   def rand_name():
9       n = random.randint(2,44)
10      s=''.join(choice(ascii_uppercase) for i in range(n))
11      return s
12
13  def rand_report():
14      n = random.randint(30,99)
15      s=''.join(choice(ascii_uppercase) for i in range(n))
16      return s
17
18  mydb = mysql.connector.connect(
19      host = 'localhost',
20      port = 3306,
21      user='root',
22      passwd = '18273645',
23      database = 'mydb'
24  )
25  mycursor = mydb.cursor()
26
27
28
29
30
31  for i in range(1000):
32      myFormula = 'INSERT INTO chaperone (Chaperone_Surname, Chaperone_
            Name, Chaperone_Patronymic) VALUES (%s,%s,%s)'
33      s_name = rand_name()
34      s_sur = rand_name()
35      s_pat = rand_name()
36      user = (s_name, s_sur, s_pat)
37      mycursor.execute(myFormula, user)
38      mydb.commit()
39
40  for i in range(400):
41      myFormula = 'INSERT INTO olympiad (Olympiad_Time, Olympiad_
            Audience_Number) VALUES (%s,%s)'
42      times = [datetime.datetime(2024, 3, 20, 10, 0, 0),datetime.
            datetime(2024, 3, 20, 12, 0, 0),datetime.datetime(2024, 3, 20,
            14, 0, 0) ]
43      i = random.randint(0,2)
44      time = times[i].strftime('%Y-%m-%d %H:%M:%S')
45      au =str(random.randint(300, 700))
46      elem = (time, au)
```

```
47      mycursor.execute(myFormula, elem)
48      mydb.commit()
49
50
51
52
53  for i in range(400):
54      myFormula = 'INSERT INTO section (Section_Name, Section_Time,
            Section_Audience_Number, Olympiad_id) VALUES (%s, %s, %s, %s)'
55      nm = rand_name()
56      times = [datetime.datetime(2024, 3, 19, 10, 0, 0), datetime.
            datetime(2024, 3, 19, 12, 0, 0), datetime.datetime(2024, 3, 19,
             14, 0, 0) ]
57      m = random.randint(0,2)
58      time_s = times[m].strftime('%Y-%m-%d %H:%M:%S')
59      au = str(random.randint(701, 1000))
60      elem = (nm, time_s, au, str(i+1))
61      mycursor.execute(myFormula, elem)
62      mydb.commit()
63
64  next_chap = 1
65
66  for i in range(7000):
67      myFormula = 'INSERT INTO participant (Participant_Surname,
            Participant_Name, Participant_Patronymic,  Chaperone_id) VALUES
             ( %s, %s, %s, %s)'
68      if(i%21==6 or i%21==13 or (i%21==0 and not i==0)):
69          next_chap+=1
70      nm = rand_name()
71      sur = rand_name()
72      patr = rand_name()
73      elem = (sur, nm, patr, str(next_chap))
74      mycursor.execute(myFormula, elem)
75      mydb.commit()
76
77
78  for i in range(10):
79      myFormula = "INSERT INTO participant_role (Participation_in_the_
            olympiad) VALUES (%s)"
80
81      elem = (str(i+1), )
82      mycursor.execute(myFormula, elem)
83      mydb.commit()
84
85  part=1
86  while(part<7000):
87      myFormula = "INSERT INTO participant_section (Participant_id,
            Section_id, Participant_Role_id) VALUES (%s, %s, %s)"
88
89      it = random.randint(3,7)
90      for i in range(it):
91          sect = random.randint(1, 400)
92          role = random.randint(1,10)
93          elem = (str(part), str(sect), str(role))
94          mycursor.execute(myFormula, elem)
95          mydb.commit()
96      part+=1
97
98
99
```

```
100 it = 1
101 sect = 1
102
103 while sect<=400:
104     myFormula = "INSERT INTO jury_member (Jury_Member_Surname, Jury_
        Member_Name, Jury_Member_Patronymic, Section_id, Olympiad_id)
        VALUES (%s, %s, %s, %s, %s)"
105     if(it%26==0 or it%26==5 or it%26==11 or it%26==18):
106         sect+=1
107
108     elem = (rand_name(), rand_name(),rand_name(), str(sect), str(sect)
        )
109     mycursor.execute(myFormula, elem)
110     mydb.commit()
111     it+=1
112
113
114
115
116 it = 1
117 sect = 1
118 for i in range(1,10001):
119     myFormula = "INSERT INTO chaperone_section (Section_id, Chaperone_
        id) VALUES (%s, %s)"
120     if(i%18==0 or i%18==5 or i%15==11):
121         sect+=1
122     it = random.randint(1,1000);
123     elem = (str(sect), str(it))
124     mycursor.execute(myFormula, elem)
125     mydb.commit()
126
127
128 sect = 1
129 while sect<=400:
130     myFormula = "INSERT INTO report (Participant_id, Section_id,
        Report_name) VALUES (%s, %s, %s);"
131     n = random.randint(25,30)
132     for i in range(n):
133         part = random.randint(1,6999)
134         elem = (str(part),str(sect), rand_report())
135         mycursor.execute(myFormula, elem)
136         mydb.commit()
137     sect+=1
```