

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

**ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2**

По теме: Запросы на выборку и модификацию данных,
представления и индексы в PostgreSQL

Специальность:

3. Мобильные и сетевые технологии

Проверил:
Говорова М.М. _____
Дата: «__» _____ 20__ г.
Оценка _____

Выполнил:
студент группы К3241
Балцат К.

Санкт-
Петербург 2022

ЦЕЛЬ РАБОТЫ

Овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL 1X, pgAdmin 4.

Практическое задание:

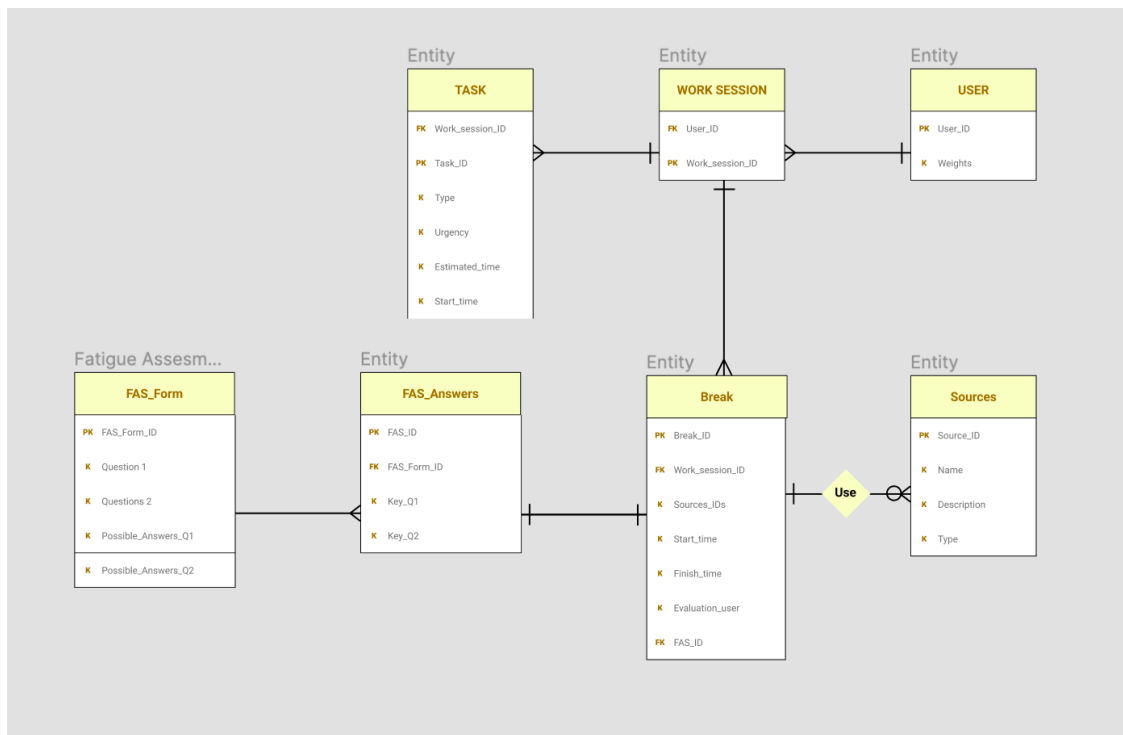
1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию, часть 2 и 3).
2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.
3. Изучить графическое представление запросов и просмотреть историю запросов
4. Создать простой и составной индексы для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

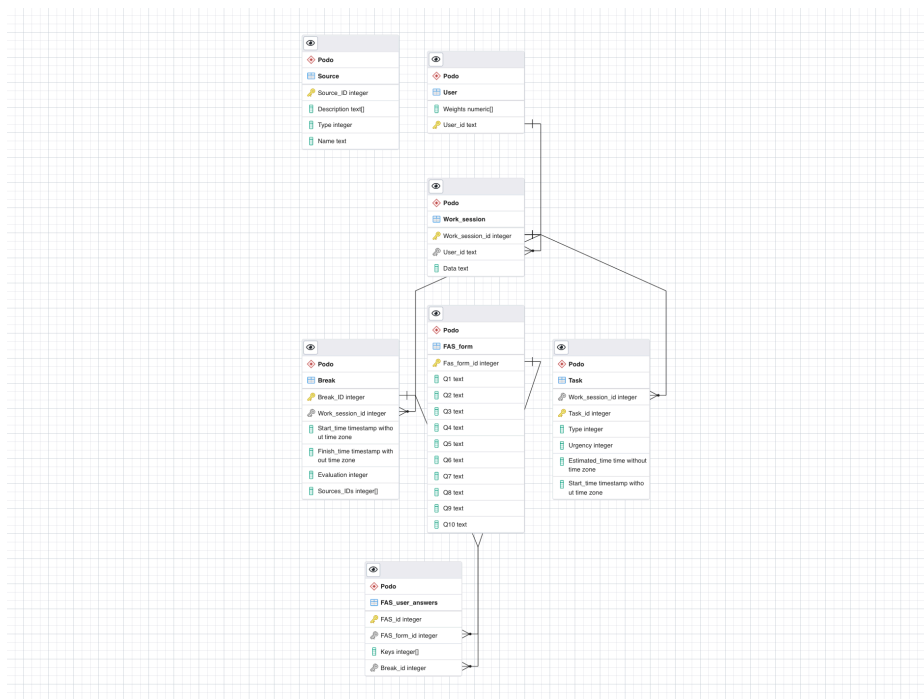
ХОД РАБОТЫ

1) Наименование БД:

Podo Data Base

2) Схема логической модели:





3) Запросы к БД

1. Список всех пользователей, их перерывов и ответов на этот перерыв

Query Editor
Query History

```

1 SELECT "Work_session"."User_id" as "Username", "Break"."Break_ID" as "Break_id", array_to_string("FAS_user_answers"."Keys", ',', '*') as "FAS_answers"
2 FROM "Podo"."Work_session"
3 INNER JOIN "Podo"."Break" ON "Break"."Work_session_id" = "Work_session"."Work_session_id"
4 INNER JOIN "Podo"."FAS_user_answers" ON "FAS_user_answers"."Break_id" = "Break"."Break_ID"
5 ORDER BY "Work_session"."User_id";
  
```

Data Output
Explain
Messages
Notifications

	Username text	Break_id integer	FAS_answers text
1	ProductivityH...	4	2,2,2,2,2,2,2,2
2	ProductivityH...	10	2,2,2,2,2,2,2,2
3	ProductivityH...	6	2,2,0,2,2,2,4,2,2
4	botan	2	2,2,2,2,2,2,2,2
5	botan	7	2,2,2,2,2,2,2,2
6	botan	1	2,2,2,2,2,2,2,2
7	botan	3	2,3,2,3,2,3,2,3
8	botan	5	2,2,2,2,2,2,2,2
9	crystall12	8	2,2,0,2,2,4,2,2
10	crystall12	9	2,1,1,2,2,3,2,2,3,2

SELECT "Work_session"."User_id" as "Username", "Break"."Break_ID" as

```

"Break_id", array_to_string("FAS_user_answers"."Keys", ',', '*') as "FAS_answers"
FROM "Podo"."Work_session"
INNER JOIN "Podo"."Break" ON "Break"."Work_session_id" =
"Work_session"."Work_session_id"
INNER JOIN "Podo"."FAS_user_answers" ON "FAS_user_answers"."Break_id" =
"Break"."Break_ID"
ORDER By «Work_session"."User_id";

```

2. Последние задачи пользователей и время их начала

Query Editor Query History

```

1 SELECT "Work_session"."User_id" as "Username", "Task"."Type" as "Task_type", "Task"."Urgency" as "Task_urgency", "Task"."Start_time" as "Start_time" FROM "Podo"."Work_session"
2 INNER JOIN "Podo"."Task" ON "Task"."Work_session_id" = "Work_session"."Work_session_id"
3 Order by "Task"."Start_time" DESC;

```

Data Output Explain Messages Notifications

	Username text	Task_type integer	Task_urgency integer	Start_time timestamp without time zone
1	botan	1	5	2022-03-17 12:10:58
2	crystal12	2	2	2022-01-01 20:23:05
3	botan	1	3	2022-01-01 00:00:00
4	botan	3	4	2021-01-01 14:16:00
5	ProductivityHero	2	3	2011-01-02 00:20:00
6	botan	1	1	2011-01-01 00:00:00

```

SELECT "Work_session"."User_id" as "Username", "Task"."Type" as "Task_type",
"Task"."Urgency" as "Task_urgency", "Task"."Start_time" as "Start_time" FROM
"Podo"."Work_session"
INNER JOIN "Podo"."Task" ON "Task"."Work_session_id" =
"Work_session"."Work_session_id"
Order by "Task"."Start_time" DESC;

```

3. Статистика по каждому пользователю: общее количество выполненных задач и среднее оцениваемое время выполнения одной задачи

Query Editor Query History

```

1 SELECT "Work_session"."User_id" as "Username", COUNT("Task"."Start_time") as "Number of tasks completed",
2 AVG("Task"."Estimated_time") as "AVG time to complete a task"
3 FROM "Podo"."Work_session"
4 INNER JOIN "Podo"."Task" ON "Task"."Work_session_id" = "Work_session"."Work_session_id"
5 GROUP By "Work_session"."User_id"
6 ORDER By COUNT("Task"."Start_time") DESC
7 LIMIT ALL;

```

Data Output Explain Messages Notifications

	Username text	Number of tasks completed bigint	AVG time to complete a task interval
1	botan	4	00:30:00
2	crystal12	1	00:27:00
3	ProductivityHero	1	01:26:00

```

SELECT "Work_session"."User_id" as "Username", COUNT("Task"."Start_time")

```

```

as "Number of tasks completed",
AVG("Task"."Estimated_time") as "AVG time to complete a task"
FROM "Podo"."Work_session"
INNER JOIN "Podo"."Task" ON "Task"."Work_session_id" =
"Work_session"."Work_session_id"
GROUP By "Work_session"."User_id"
ORDER By COUNT("Task"."Start_time") DESC
LIMIT ALL;

```

4. Рабочие сессии пользователей, в которых последние выполняли задание

Query Editor Query History

```

1 SELECT "Work_session"."Work_session_id", "Work_session"."User_id"
2 FROM "Podo"."Work_session"
3 WHERE "Work_session_id" in (
4     SELECT "Work_session"."Work_session_id" FROM "Podo"."Work_session", "Podo"."Task"
5     WHERE "Work_session"."Work_session_id" = "Task"."Task_id"
6 );
7

```

Data Output Explain Messages Notifications

	Work_session_id [PK] integer	User_id text
1	3	ProductivityHero
2	5	botan
3	1	botan
4	2	botan
5	4	ProductivityHero
6	6	botan

4) Графические представления.

1. Создать представление для продуктового менеджера о кумулятивных метриках компании

```

CREATE VIEW Cumulative_data as
SELECT COUNT("Work_session"."User_id") as "Number of users",
COUNT("Task"."Start_time") as "Number of tasks completed",
AVG("Task"."Estimated_time") as "AVG time to complete a task",
SUM(("Break"."Finish_time" - "Break"."Start_time")) as "Cumulative time of
users' breaks"
FROM "Podo"."Work_session"
INNER JOIN "Podo"."Task" ON "Task"."Work_session_id" =
"Work_session"."Work_session_id"
INNER JOIN "Podo"."Break" ON "Break"."Work_session_id" =
"Work_session"."Work_session_id"
LIMIT ALL;

```

Podo/postgres@PostgreSQL 14				
Query Editor Query History				
1 SELECT * FROM Cumulative_data;				
Data Output Explain Messages Notifications				
	Number of users bigint	Number of tasks completed bigint	AVG time to complete a task interval	Cumulative time of users' breaks interval
1	1	4	00:22:30	00:04:00

2. Создать представление по каждому из пользователей о его типах задач в среднем

```
CREATE VIEW Avg_tasks_per_user as
SELECT "Work_session"."User_id", COUNT(DISTINCT "Task"."Type") as
"Different task types", AVG("Task"."Type") as "Average task type",
COUNT(DISTINCT "Task"."Urgency") as "Different urgency types",
AVG("Task"."Urgency") as "Average task urgency"
FROM "Podo"."Work_session", "Podo"."Task"
GROUP BY «Work_session"."User_id";
```

Query Editor Query History						
1 SELECT * FROM Avg_tasks_per_user;						
Data Output Explain Messages Notifications						
	User_id text	Different task types bigint	Average task type numeric	Different urgency types bigint	Average task urgency numeric	
1	ProductivityHero	3	1.6666666666666667	5	3.0000000000000000	
2	botan	3	1.6666666666666667	5	3.0000000000000000	
3	crystal12	3	1.6666666666666667	5	3.0000000000000000	

5) Запросы на модификацию данных

Update - добавить в последние рабочие сессии информацию о кумулятивном времени перерыва там, где это возможно

Query Editor

Query History

```
1 UPDATE "Podo"."Work_session"
2 SET "Data" = ('Cumulative Break time')
3 where "Work_session"."Work_session_id" in (
4     SELECT "Work_session"."Work_session_id"
5     -- SUM("Break"."Finish_time" - "Break"."Start_time")
6     FROM "Podo"."Work_session", "Podo"."Break"
7     WHERE "Work_session"."Work_session_id" = "Break"."Work_session_id"
8     GROUP BY "Work_session"."Work_session_id"
9     ORDER BY "Work_session"."Work_session_id");
```

Data Output

Explain

Messages

Notifications

UPDATE 6

Query returned successfully in 47 msec.

	Work_session_id [PK] integer	User_id text	Data text
1	3	ProductivityHero	Data
2	5	botan	Work
3	8	crystal12	Empty
4	10	crystal12	[null]
5	1	botan	Cumulat...
6	2	botan	Cumulat...
7	4	ProductivityHero	Cumulat...
8	6	botan	Cumulat...
9	7	ProductivityHero	Cumulat...
10	9	crystal12	Cumulat...

Insert - создать нового пользователя с уникальным ником «task_destroyer» и начальными весами по умолчанию
INSERT INTO "Podo"."User"("User_id", "Weights")
VALUES('task_destroyer', DEFAULT);

Query Editor

Query history

1

INSERT INTO "Podo"."User"("User_id", "Weights") VALUES('task_destroyer', DEFAULT);

Data Output

Explain

Messages

Notifications

INSERT 0 1

Query returned successfully in 51 msec.

После:

	Weights numeric[]	User_id [PK] text
1	{0,0,1,2}	botan
2	{0,1,2,3}	crystal12
3	{0,1,10,4}	ProductivityHero
4	{0,0,0,1}	task_destroyer

Delete - удалить дыхательную практику «Lion’s Breath» из ресурсов к перерывам

До:

	Source_ID [PK] integer	Description text[]	Type integer	Name text
1	1	{<html>}	1	Yoga
2	2	{<html>}	2	Chi
3	3	{<html>}	2	Meditation
4	4	{<html>}	3	Eye Movement
5	5	{<html>}	1	7 Min HIIT
6	6	{<html>}	4	Breathing Fast
7	7	{<html>}	4	Wim Hoff Breathing
8	8	{<html>}	4	Lion's Breath

После:

	Source_ID [PK] integer	Description text[]	Type integer	Name text
1	1	{<html>}	1	Yoga
2	2	{<html>}	2	Chi
3	3	{<html>}	2	Meditation
4	4	{<html>}	3	Eye Movement
5	5	{<html>}	1	7 Min HIIT
6	6	{<html>}	4	Breathing Fast
7	7	{<html>}	4	Wim Hoff Breathing

Создание индексов


1. Запросы без индексов 1-й

```
SELECT "Work_session"."Work_session_id", "Work_session"."User_id" FROM
"Podo"."Work_session"
WHERE "Work_session_id" in (
    SELECT "Work_session"."Work_session_id" FROM "Podo"."Work_session",
    "Podo"."Task"
    WHERE "Work_session"."Work_session_id" = "Task"."Work_session_id"
)
```

Query Editor Query History

```
1 EXPLAIN SELECT "Work_session"."Work_session_id", "Work_session"."User_id"
2 FROM "Podo"."Work_session"
3 WHERE "Work_session_id" in (
4     SELECT "Work_session"."Work_session_id" FROM "Podo"."Work_session", "Podo"."Task"
5     WHERE "Work_session"."Work_session_id" = "Task"."Task_id"
6 )
7
```

Data Output Explain Messages Notifications

QUERY PLAN		
	text	
1	Hash Semi Join (cost=66.93..97.12 rows=850 width=36)	
2	[...] Hash Cond: ("Work_session"."Work_session_id" = "Work_session_1"."Work_session_id")	
3	[...] -> Seq Scan on "Work_session" (cost=0.00..18.50 rows=850 width=36)	
4	[...] -> Hash (cost=56.31..56.31 rows=850 width=8)	
5	[...] -> Hash Join (cost=29.12..56.31 rows=850 width=8)	
6	[...] Hash Cond: ("Task"."Task_id" = "Work_session_1"."Work_session_id")	
7	[...] -> Seq Scan on "Task" (cost=0.00..23.60 rows=1360 width=4)	
8	[...] -> Hash (cost=18.50..18.50 rows=850 width=4)	
9	[...] -> Seq Scan on "Work_session" "Work_session_1" (cost=0.00..18.50 rows=850 width=4)	

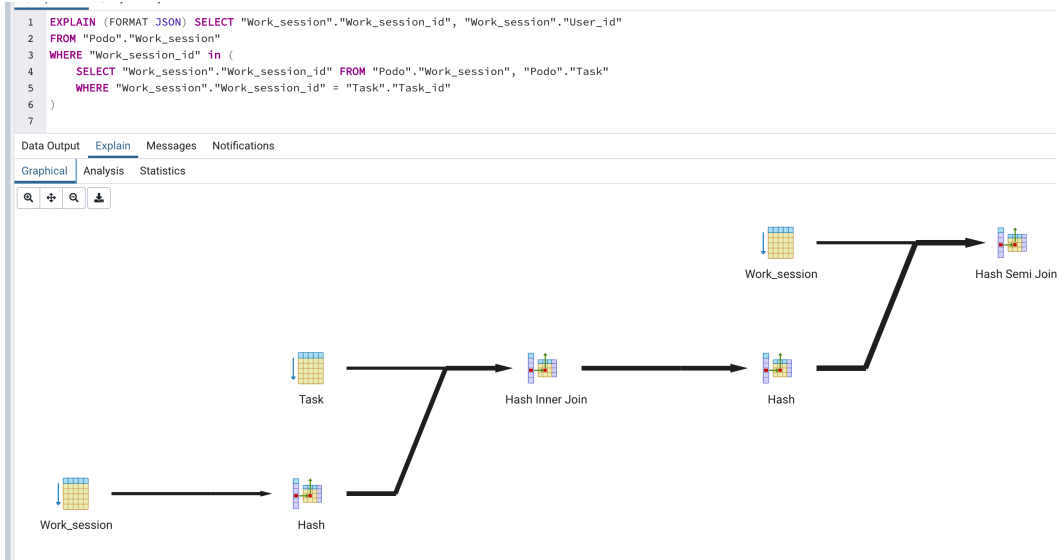
Podo/postgres@PostgreSQL 14

Query Editor Query History

```
1 EXPLAIN SELECT "Work_session"."Work_session_id", "Work_session"."User_id"
2 FROM "Podo"."Work_session"
3 WHERE "Work_session_id" in (
4     SELECT "Work_session"."Work_session_id" FROM "Podo"."Work_session", "Podo"."Task"
5     WHERE "Work_session"."Work_session_id" = "Task"."Task_id"
6 )
7
```

Data Output Explain Messages Notifications

Successfully run. Total query runtime: 60 msec.
9 rows affected.

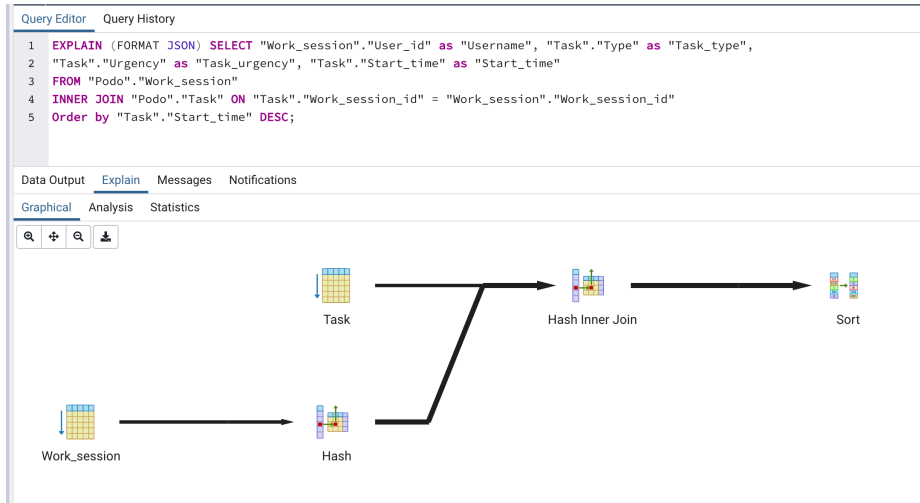


2. Запросы без индексов 2-й

```
1 EXPLAIN SELECT "Work_session"."User_id" as "Username", "Task"."Type" as "Task_type",
2 "Task"."Urgency" as "Task_urgency", "Task"."Start_time" as "Start_time"
3 FROM "Podo"."Work_session"
4 INNER JOIN "Podo"."Task" ON "Task"."Work_session_id" = "Work_session"."Work_session_id"
5 Order by "Task"."Start_time" DESC;
```

Data Output Explain Messages Notifications

QUERY PLAN	
text	
1 Sort (cost=127.10..130.50 rows=1360 width=48)	
2 [...] Sort Key: "Task"."Start_time" DESC	
3 [...] -> Hash Join (cost=29.12..56.32 rows=1360 width=48)	
4 [...] Hash Cond: ("Task"."Work_session_id" = "Work_session"."Work_session_id")	
5 [...] -> Seq Scan on "Task" (cost=0.00..23.60 rows=1360 width=20)	
6 [...] -> Hash (cost=18.50..18.50 rows=850 width=36)	
7 [...] -> Seq Scan on "Work_session" (cost=0.00..18.50 rows=850 width=36)	



Data Output Explain **Messages** Notifications

Successfully run. Total query runtime: 30 msec.
1 rows affected.

3. Запросы с индексами 1-й

```
1 CREATE INDEX work_session_index ON "Podo"."Work_session"("Work_session_id")
```

Data Output Explain Messages Notifications

CREATE INDEX

Query returned successfully in 48 msec.

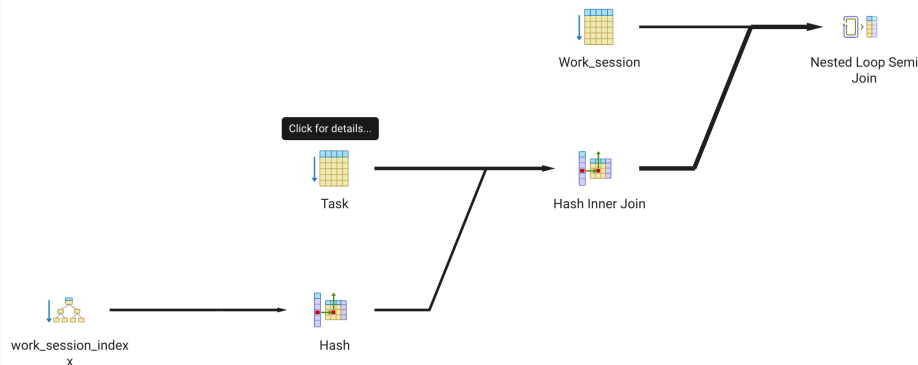
Data Output Explain Messages Notifications

Successfully run. Total query runtime: 74 msec.
5 rows affected.

```
1 EXPLAIN (FORMAT JSON) SELECT "Work_session"."Work_session_id", "Work_session"."User_id" FROM "Podo"."Work_session"  
2 WHERE "Work_session_id" in (  
3 SELECT "Work_session"."Work_session_id" FROM "Podo"."Work_session", "Podo"."Task"  
4 WHERE "Work_session"."Work_session_id" = "Task"."Work_session_id"  
5 )
```

Data Output Explain Messages Notifications

Graphical Analysis Statistics



4. Запросы с индексами 2-й

Query Editor Query History

```
1 CREATE UNIQUE INDEX work_session_composed on "Podo"."Work_session"("Work_session_id", "User_id")
```

Data Output Explain Messages Notifications

CREATE INDEX

Query returned successfully in 55 msec.

Podo/postgres@PostgreSQL 14

Query Editor Query History

```
1 EXPLAIN (FORMAT JSON) SELECT "Work_session"."User_id" as "Username", "Task"."Type" as "Task_type",  
2 "Task"."Urgency" as "Task_urgency", "Task"."Start_time" as "Start_time"  
3 FROM "Podo"."Work_session"  
4 INNER JOIN "Podo"."Task" ON "Task"."Work_session_id" = "Work_session"."Work_session_id"  
5 Order by "Task"."Start_time" DESC;
```

Data Output Explain Messages Notifications

Successfully run. Total query runtime: 42 msec.
1 rows affected.

Вывод:

В ходе лабораторной работы реализованы запросы к БД и созданы графические представления, которые полезны для бизнес-аналитики приложения. Я создал запросы и представления на выборку данных, запросы на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов. Изучил графическое представление запросов. Создал простой и составной индексы для двух запросов и сравнил время выполнения запросов без индексов и с индексами: с индексами время выполнения больше, так как моя база данных маленькая.

Также выполнены задания на тренажере-SQL на hackerrank.

Prepare > Sql

FAIR

85 more points to get your next star!
Rank: 851591 | Points: 90/175

<div>Revising the Select Query I</div> <div>Easy, SQL (Basic), Max Score: 10, Success Rate: 96.33%</div> <div>★</div> <div>Solved </div>	<div>STATUS</div> <div><input checked="" type="checkbox"/> Solved</div> <div><input type="checkbox"/> Unsolved</div> <div>SKILLS</div> <div><input type="checkbox"/> SQL (Basic)</div> <div><input type="checkbox"/> SQL (Intermediate)</div> <div><input type="checkbox"/> SQL (Advanced)</div> <div>DIFFICULTY</div> <div><input type="checkbox"/> Easy</div> <div><input type="checkbox"/> Medium</div> <div><input type="checkbox"/> Hard</div> <div>SUBDOMAINS</div> <div><input type="checkbox"/> Basic Select</div> <div><input type="checkbox"/> Advanced Select</div> <div><input type="checkbox"/> Aggregation</div> <div><input type="checkbox"/> Basic Join</div> <div><input type="checkbox"/> Advanced Join</div> <div><input type="checkbox"/> Alternative Queries</div>
<div>Revising the Select Query II</div> <div>Easy, SQL (Basic), Max Score: 10, Success Rate: 98.84%</div> <div>★</div> <div>Solved </div>	
<div>Select All</div> <div>Easy, SQL (Basic), Max Score: 10, Success Rate: 99.66%</div> <div>★</div> <div>Solved </div>	
<div> Stand out from the crowd</div> <div>Take the HackerRank Skills Certification Test and make your profile stand out to peers & employers</div> <div>Get Certified</div>	
<div>Select By ID</div> <div>Easy, SQL (Basic), Max Score: 10, Success Rate: 99.70%</div> <div>★</div> <div>Solved </div>	
<div>Type of Triangle</div> <div>Easy, SQL (Basic), Max Score: 20, Success Rate: 96.52%</div> <div>★</div> <div>Solved </div>	
<div>Population Census</div> <div>Easy, SQL (Basic), Max Score: 10, Success Rate: 98.65%</div> <div>★</div> <div>Solved </div>	
<div>African Cities</div> <div>Easy, SQL (Basic), Max Score: 10, Success Rate: 99.40%</div> <div>★</div> <div>Solved </div>	
<div>Average Population of Each Continent</div> <div>Easy, SQL (Basic), Max Score: 10, Success Rate: 98.14%</div> <div>★</div> <div>Solved </div>	