МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

# ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине

«Базы данных»

Вариант № 5656

***Выполнил:***

Студент группы P3115

Чимирев Игорь Олегович

***Проверил:***

Райла Мартин

Санкт-Петербург, 2025

## Задание

Составить запросы на языке SQL (пункты 1-2).

Для каждого запроса предложить индексы, добавление которых уменьшит время выполнения запроса (указать таблицы/атрибуты, для которых нужно добавить индексы, написать тип индекса; объяснить, почему добавление индекса будет полезным для данного запроса).

Для запросов 1-2 необходимо составить возможные планы выполнения запросов. Планы составляются на основании предположения, что в таблицах отсутствуют индексы. Из составленных планов необходимо выбрать оптимальный и объяснить свой выбор. Изменятся ли планы при добавлении индекса и как?

1. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:
   Таблицы: Н_ЛЮДИ, Н_СЕССИЯ.
   Вывести атрибуты: Н_ЛЮДИ.ИД, Н_СЕССИЯ.УЧГОД.
   Фильтры (AND):
   a) Н_ЛЮДИ.ИД = 100012.
   b) Н_СЕССИЯ.ИД < 1975.
   Вид соединения: INNER JOIN.
. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:
Таблицы: Н_ЛЮДИ, Н_ВЕДОМОСТИ, Н_СЕССИЯ.
Вывести атрибуты: Н_ЛЮДИ.ОТЧЕСТВО, Н_ВЕДОМОСТИ.ИД, Н_СЕССИЯ.ДАТА.
Фильтры (AND):
a) Н_ЛЮДИ.ИД = 152862.
b) Н_ВЕДОМОСТИ.ЧЛВК_ИД < 105590.
Вид соединения: LEFT JOIN.

## Запросы

### 1 Запрос

```
SELECT Н_ЛЮДИ.ИД, Н_СЕССИЯ.УЧГОД

FROM Н_ЛЮДИ

INNER JOIN Н_СЕССИЯ ON Н_ЛЮДИ.ИД = Н_СЕССИЯ.ЧЛВК_ИД

WHERE Н_ЛЮДИ.ИД = 100012

  AND Н_СЕССИЯ.ИД < 1975;
```

**Возможные индексы**

Для таблицы Н_ЛЮДИ:
CREATE INDEX idx_н_люди_ид ON Н_ЛЮДИ (ИД);

**Обоснование:**

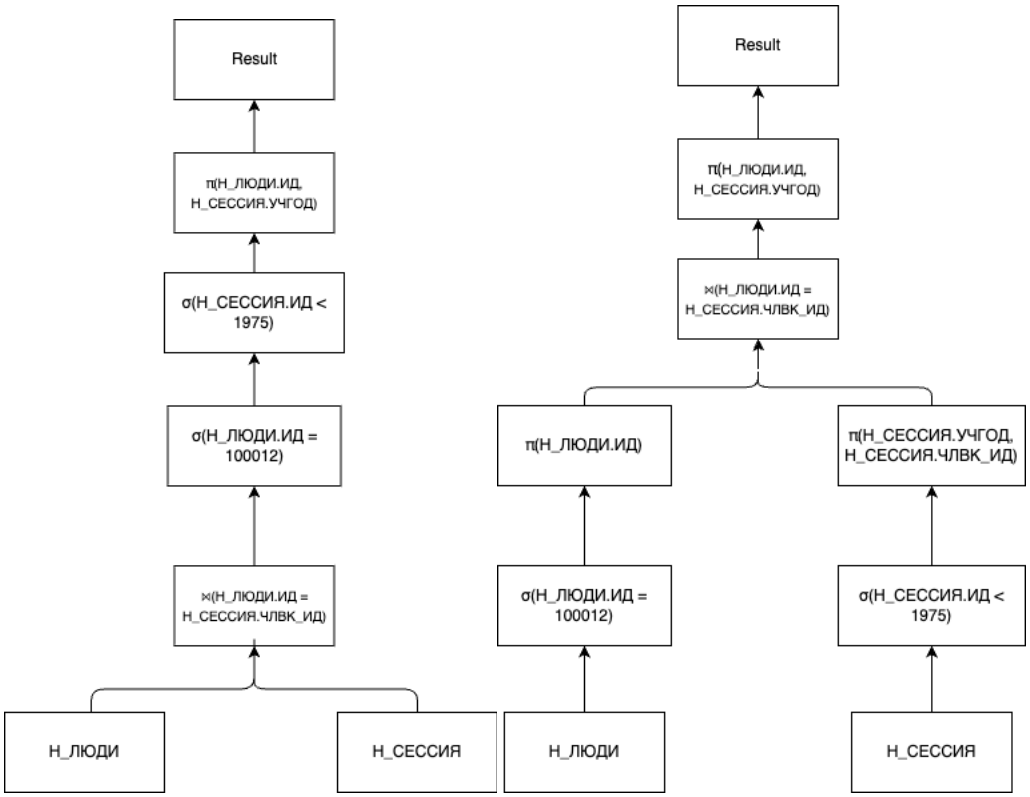
Условие Н_ЛЮДИ.ИД = 100012 требует быстрого поиска по полю ИД.
Индекс типа B-tree позволяет найти строку за время $O(\log n)$, избегая полного сканирования таблицы.
Включение дополнительных полей (если требуется) снижает нагрузку на таблицу (например, INCLUDE (ОТЧЕСТВО) для покрывающего индекса).

Для таблицы Н_СЕССИЯ:
CREATE INDEX idx_н_сессия_ид_члвк_ид ON Н_СЕССИЯ (ИД, ЧЛВК_ИД) INCLUDE (УЧГОД);

**Обоснование:**

Условие Н_СЕССИЯ.ИД < 1975 требует доступа к диапазону значений. Индекс на ИД ускоряет фильтрацию.
Включение ЧЛВК_ИД позволяет выполнить соединение с Н_ЛЮДИ без обращения к таблице.
INCLUDE (УЧГОД) добавляет поле в листовые узлы индекса, что исключает дополнительный поиск в таблице для получения УЧГОД.

**План запроса без индексов:**



Оптимальным является второй вариант (справа), поскольку мы сначала выполняем выборку, проекцию по обеим таблицам и только потом их соединяем, когда в первом случае происходит соединение двух таблиц целиком.

При наличии индексов выборка станет более эффективной.

### Вывод EXPLAIN ANALYZE по запросу до добавления индексов

```
Nested Loop  (cost=4.61..27.49 rows=1 width=14) (actual time=0.368..0.562 rows=6 loops=1)
  ->  Index Only Scan using "ЧЛВК_PK" on "Н_ЛЮДИ"  (cost=0.28..4.30 rows=1 width=4) (actual time=0.130..0.132 rows=1 loops=1)
        Index Cond: ("ИД" = 100012)
        Heap Fetches: 0
  ->  Bitmap Heap Scan on "Н_СЕССИЯ"  (cost=4.33..23.18 rows=1 width=14) (actual time=0.226..0.415 rows=6 loops=1)
        Recheck Cond: ("ЧЛВК_ИД" = 100012)
        Filter: ("ИД" < 1975)
        Heap Blocks: exact=5
        ->  Bitmap Index Scan on "SYS_C003500_IFK"  (cost=0.00..4.33 rows=6 width=0) (actual time=0.122..0.122 rows=6 loops=1)
              Index Cond: ("ЧЛВК_ИД" = 100012)
 Planning Time: 2.333 ms
 Execution Time: 0.931 ms
```

### Вывод EXPLAIN ANALYZE по запросу после добавления индексов

```
Nested Loop  (cost=0.56..26.94 rows=1 width=14) (actual time=0.197..0.306 rows=6 loops=1)
  ->  Index Only Scan using "idx_н_люди_ид" on "Н_ЛЮДИ"  (cost=0.28..4.30 rows=1 width=4) (actual time=0.108..0.109 rows=1
loops=1)
        Index Cond: ("ИД" = 100012)
        Heap Fetches: 0
  ->  Index Only Scan using "idx_н_сессия_ид_члвк_ид" on "Н_СЕССИЯ"  (cost=0.28..22.63 rows=1 width=14) (actual
time=0.086..0.193 rows=6 loops=1)
        Index Cond: (("ИД" < 1975) AND ("ЧЛВК_ИД" = 100012))
        Heap Fetches: 0
 Planning Time: 1.091 ms
 Execution Time: 0.359 ms
```

ЗАПРОС 2

```
SELECT

      Н_ЛЮДИ.ОТЧЕСТВО, Н_ВЕДОМОСТИ.ИД, Н_СЕССИЯ.ДАТА

FROM

      Н_ЛЮДИ

LEFT JOIN

      Н_ВЕДОМОСТИ ON Н_ВЕДОМОСТИ.ЧЛВК_ИД = Н_ЛЮДИ.ИД
```

```
LEFT JOIN

     Н_СЕССИЯ ON Н_СЕССИЯ.ЧЛВК_ИД = Н_ЛЮДИ.ИД

WHERE

     Н_ЛЮДИ.ИД = 152682 AND

     Н_ВЕДОМОСТИ.ЧЛВК_ИД < 105590;
```

**Возможные индексы**

```
     CREATE INDEX idx_н_люди_ид ON Н_ЛЮДИ (ИД)
INCLUDE(ОТЧЕСТВО);

     CREATE INDEX idx_н_ведомости_члвк_ид ON Н_ВЕДОМОСТИ
(ЧЛВК_ИД, ИД);

     CREATE INDEX idx_н_сессия_члвк_ид ON Н_СЕССИЯ (ЧЛВК_ИД)
INCLUDE(ДАТА);
```

1. **idx_н_люди_ид**

**Эффект:**

Ускоряет доступ к строке в Н_ЛЮДИ по условию Н_ЛЮДИ.ИД = 152682 и
Колонка ОТЧЕСТВО включена в индекс (покрывающий индекс), что позволяет избежать обращения к основной таблице после поиска по ИД и сразу получить значение ОТЧЕСТВО из индекса

**Оптимизация:**

Фильтрация по Н_ЛЮДИ.ИД становится мгновенной (O(1) для уникального ИД)
Устраняет чтение всей строки Н_ЛЮДИ

2. **idx_н_ведомости_члвк_ид**

**Эффект:**

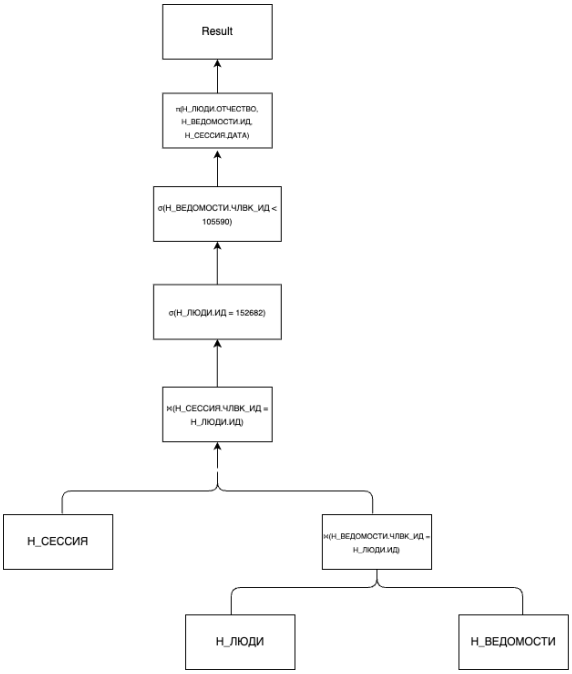
Оптимизирует:
JOIN между Н_ЛЮДИ и Н_ВЕДОМОСТИ через ЧЛВК_ИД
Фильтр Н_ВЕДОМОСТИ.ЧЛВК_ИД < 105590
Позволяет быстро находить все записи Н_ВЕДОМОСТИ для конкретного ЧЛВК_ИД и одновременно получить значение ИД без обращения к таблице

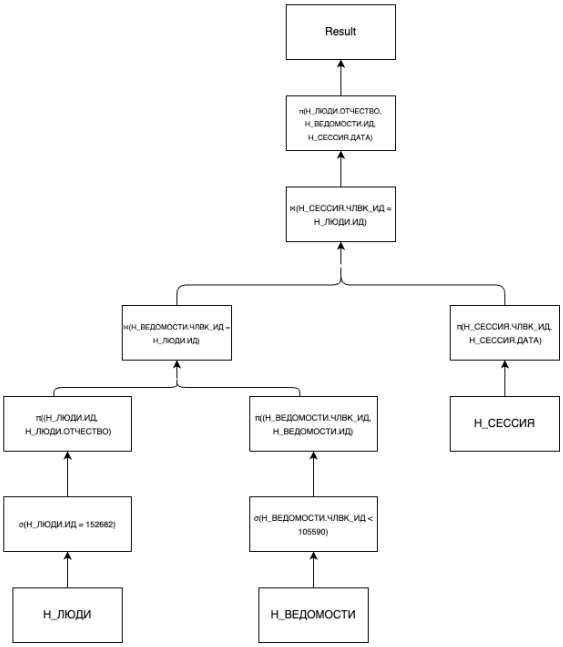
3. **idx_н_сессия_члвк_ид**

**Эффект:**

Ускоряет JOIN с Н_СЕССИЯ через ЧЛВК_ИД
Включение ДАТА позволяет получить значение прямо из индекса и избежать дополнительного чтения данных из таблицы Н_СЕССИЯ.

## План запроса без индексов:




Оптимальным также является второй вариант (справа), поскольку мы сначала выполняем выборку, проекцию по обеим таблицам и только потом их соединяем, когда в первом случае происходит соединение двух таблиц целиком.

При наличии индексов выборка станет более эффективной.

## Вывод `EXPLAIN ANALYZE` по запросу до добавления индексов:

Nested Loop Left Join  (cost=4.87..26.91 rows=2 width=32) (actual time=0.012..0.013 rows=0 loops=1)

-> Nested Loop  (cost=0.58..15.58 rows=1 width=28) (actual time=0.011..0.012 rows=0 loops=1)

-> Index Scan using "ЧЛВК_PK" on "Н_ЛЮДИ"  (cost=0.28..8.30 rows=1 width=24) (actual time=0.011..0.011 rows=0 loops=1)

Index Cond: ("ИД" = 152682)

-> Index Scan using "ВЕД_ЧЛВК_FK_IFK" on "Н_ВЕДОМОСТИ"  (cost=0.29..7.27 rows=1 width=8) (never executed)

Index Cond: (("ЧЛВК_ИД" < 105590) AND ("ЧЛВК_ИД" = 152682))

-> Bitmap Heap Scan on "Н_СЕССИЯ"  (cost=4.30..11.31 rows=2 width=12) (never executed)

Recheck Cond: ("ЧЛВК_ИД" = 152682)

-> Bitmap Index Scan on "SYS_C003500_IFK" (cost=0.00..4.29 rows=2 width=0) (never executed)

Index Cond: ("ЧЛВК_ИД" = 152682)

Planning Time: 0.275 ms

Execution Time: 0.063 ms

## Вывод `EXPLAIN ANALYZE` по запросу после добавления индексов:

Nested Loop Left Join (cost=0.86..15.91 rows=2 width=32) (actual time=0.007..0.008 rows=0 loops=1)

-> Nested Loop (cost=0.58..11.57 rows=1 width=28) (actual time=0.007..0.008 rows=0 loops=1)

-> Index Only Scan using "idx_н_люди_ид_отчество" on "Н_ЛЮДИ" (cost=0.28..4.30 rows=1 width=24) (actual time=0.007..0.007 rows=0 loops=1)

Index Cond: ("ИД" = 152682)

Heap Fetches: 0

-> Index Scan using "ВЕД_ЧЛВК_FK_IFK" on "Н_ВЕДОМОСТИ" (cost=0.29..7.26 rows=1 width=8) (never executed)

Index Cond: (("ЧЛВК_ИД" < 105590) AND ("ЧЛВК_ИД" = 152682))

-> Index Only Scan using "idx_н_сессия_члвк_ид_дата" on "Н_СЕССИЯ" (cost=0.28..4.31 rows=2 width=12) (never executed)

Index Cond: ("ЧЛВК_ИД" = 152682)

Heap Fetches: 0

Planning Time: 0.457 ms

Execution Time: 0.030 ms

## Вывод

Узнал про индексы, принцип их работы, сферы использования, познакомился с EXPLAIN ANALYZE.

```
DROP TABLE night CASCADE;

CREATE TABLE night (
night_id SERIAL PRIMARY KEY,
time_of_day TIME NOT NULL,
stars_present BOOLEAN DEFAULT FALSE,
total_darkness BOOLEAN DEFAULT TRUE
);

INSERT INTO night (time_of_day, stars_present, total_darkness)
SELECT
'20:00:00'::time + (random() * 86400 * '1 second'::interval),
random() < 0.7,
random() < 0.3
FROM generate_series(1, 100000000000);
```

```
INSERT INTO night (time_of_day, stars_present, total_darkness)
SELECT
'20:00:00'::time + (random() * 86400 * '1 second'::interval),
random() < 0.7,
random() < 0.3
FROM generate_series(1, 1000000);

CREATE INDEX idx_night_stars_true
ON night(stars_present)
WHERE stars_present = TRUE;

CREATE INDEX idx_night_stars_false
ON night(stars_present)
WHERE stars_present = FALSE;

EXPLAIN ANALYZE
SELECT *
FROM night
WHERE stars_present = TRUE;
EXPLAIN ANALYZE
SELECT *
FROM night
WHERE stars_present = FALSE;
```

```
                                                                 QUERY
PLAN
------------------------------------------------------------------------
------------------------------------------------------------------------
 Bitmap Heap Scan on night  (cost=4325.43..15695.42 rows=500000
width=14) (actual time=33.905..79.589 rows=699579 loops=1)
   Recheck Cond: stars_present
   Heap Blocks: exact=6370
   ->  Bitmap Index Scan on idx_night_stars_true  (cost=0.00..4200.43
rows=500000 width=0) (actual time=32.643..32.643 rows=699579 loops=1)
 Planning Time: 0.895 ms
 Execution Time: 98.823 ms
(6 rows)



Bitmap Heap Scan on night  (cost=12948.05..59774.38 rows=1497833
width=14) (actual time=63.273..250.991 rows=1499799 loops=1)
   Recheck Cond: (NOT stars_present)
   Heap Blocks: exact=31848
   ->  Bitmap Index Scan on idx_night_stars_false
(cost=0.00..12573.59 rows=1497833 width=0) (actual time=58.337..58.337
rows=1499799 loops=1)
 Planning Time: 0.255 ms
 Execution Time: 279.618 ms
(6 rows)
```