Университет ИТМО   
Мегафакультет компьютерных технологий и управления   
Факультет программной инженерии и компьютерной техники



Лабораторная работа №4

по Информационным системам и базам данных

Вариант: 9381

Группа: Р33111

Выполнил: Козлов Виктор Николаевич

Преподаватель: Николаев Владимир Вячеславович

Санкт-Петербург

2021 год

## Текст задания

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?

Для запросов 1-2 необходимо добавить в отчет вывод команды EXPLAIN ANALYZE [запрос]

Подробные ответы на все вышеперечисленные вопросы должны присутствовать в отчете (планы выполнения запросов должны быть нарисованы, ответы на вопросы - представлены в текстовом виде).

## Реализация запросов

### Задание 1

Изображение выглядит как текст

Автоматически созданное описание

Запрос

Select "ФАМИЛИЯ", "ДАТА"

From "Н\_ЛЮДИ" peop

RIGHT JOIN "Н\_СЕССИЯ" sess on peop."ИД" = sess."ЧЛВК\_ИД"

WHERE peop."ИМЯ" = 'Николай' and sess."УЧГОД" = '2001/2002' and sess."УЧГОД" > '2011/2012';

Индексы

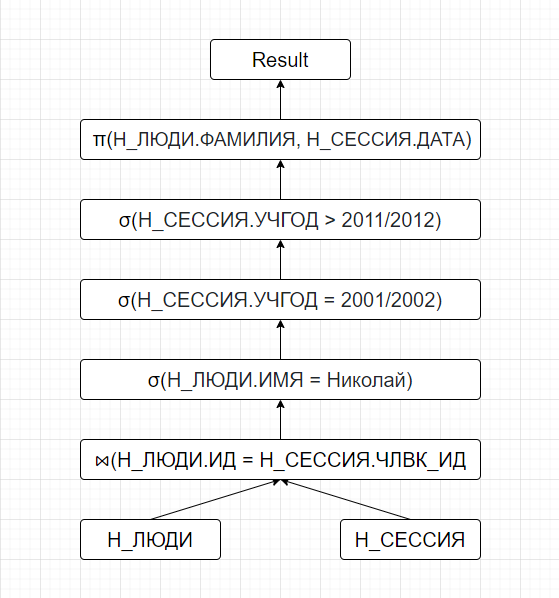
CREATE INDEX "ИНД\_ИМЯ" ON "Н\_ЛЮДИ" ("ИМЯ");

CREATE INDEX "ИНД\_УЧГОД" ON "Н\_СЕССИЯ" ("УЧГОД");

CREATE INDEX "ИНД\_ЧЛВК\_ИД" ON "Н\_СЕССИЯ" USING HASH("ЧЛВК\_ИД");

Возможные планы выполнения запроса

А)



Б)

Изображение выглядит как стол

Автоматически созданное описание

SELECT COUNT(\*) FROM "Н\_ЛЮДИ"; -- 5118

SELECT COUNT(\*) FROM "Н\_СЕССИЯ"; -- 3752

SELECT COUNT(\*) FROM "Н\_ЛЮДИ"

RIGHT JOIN "Н\_СЕССИЯ" sess on "Н\_ЛЮДИ"."ИД" = sess."ЧЛВК\_ИД"; -- 3752

SELECT COUNT(\*) FROM "Н\_СЕССИЯ"

RIGHT JOIN "Н\_ЛЮДИ" peop on peop."ИД" = "Н\_СЕССИЯ"."ЧЛВК\_ИД"; --8177

SELECT COUNT(\*) FROM "Н\_ЛЮДИ" WHERE "ИМЯ" = 'Николай'; -- 74

SELECT COUNT(\*) FROM "Н\_СЕССИЯ" WHERE "УЧГОД" = '2001/2002' and "УЧГОД" > '2011/2012'; -- 0

SELECT COUNT(\*) From "Н\_ЛЮДИ" peop

RIGHT JOIN "Н\_СЕССИЯ" sess on peop."ИД" = sess."ЧЛВК\_ИД"

WHERE peop."ИМЯ" = 'Николай' and sess."УЧГОД" = '2001/2002' and sess."УЧГОД" > '2011/2012';-- 0

SELECT COUNT(\*) From "Н\_СЕССИЯ" sess

RIGHT JOIN "Н\_ЛЮДИ" peop on peop."ИД" = sess."ЧЛВК\_ИД"

WHERE peop."ИМЯ" = 'Николай' and sess."УЧГОД" = '2001/2002' and sess."УЧГОД" > '2011/2012'; --0

Я отобразил основные планы выполнения этого запроса. Я думаю, что оптимальным планом будет Б), т. к. сначала мы выбираем строки по указанным условиям, а уже потом отобранные строчки объединяем. При проверке выяснилось, что при объединении обеих таблиц без условий у нас получается 3752 или 8177 строк в зависимости от положения таблиц (Делаем вывод, что выгоднее поставить справа "Н\_СЕССИЯ"). Причем в условии на УЧГОД в таблице Н\_СЕССИЯ мы видим, что проходит 0 строк таким образом можно сделать вывод, что объединение после двух фильтров будет оптимальным решением.

Мы используем RIGHT OUTER JOIN, поэтому у нас в результат идут только строки, в которых выполняется условие объединения, а также все строки из правой таблицы, которым не соответствуют никакие строки из левой таблицы. Как мы видим, результат не зависит от того, какая таблица присоединяется, потому что в условии на УЧГОД в таблице Н\_СЕССИЯ получается 0, поэтому менять Join нет смысла.

Выгодно использовать Nested Loop при соединении, т.к. у нас соединяются небольшие таблицы (особенно отфильтрованные). А с учетом того, что у нас один из фильтров просто даст 0, то Hash Join только сделает лишние расчеты.

При добавлении предложенных индексов план останется эффективным, т.к. использование BTREE индекс для операций сравнения ускоряет выборку, а операция сравнения при соединении таблиц ускорится, если использовать HASH на ЧЛВК\_ИД в таблице Н\_СЕССИЯ.

Вывод Explain Analyze

Изображение выглядит как текст

Автоматически созданное описание

Мы видим, что мои предположения практически полностью совпали с действительностью. У нас идет фильтрация, а уже затем объединение таблиц. Кроме того, мы видим, что сначала просматривается (Seq Scan) таблица Н\_ЛЮДИ. Для каждой её строки просматриваются совпадения на "УЧГОД" = '2001/2002' и УЧГОД" > '2011/2012'. Затем сразу идет соединение таблиц, так как по факту на выходе будет 0 строк. И в конце проверяется "ИМЯ" = 'Николай', который уже ничего не меняет.  
  
Задание 2

Изображение выглядит как текст

Автоматически созданное описание

Запрос

SELECT peop."ИМЯ", ob."ЧЛВК\_ИД", pup."НАЧАЛО"

FROM "Н\_ЛЮДИ" peop

JOIN "Н\_ОБУЧЕНИЯ" ob on peop."ИД" = ob."ЧЛВК\_ИД"

JOIN "Н\_УЧЕНИКИ" pup on peop."ИД" = pup."ЧЛВК\_ИД"

WHERE

peop."ОТЧЕСТВО" < 'Владимирович'

AND ob."ЧЛВК\_ИД" < 163484;

Индексы

CREATE INDEX "ИНД\_ОТЧЕСТВО" ON "Н\_ЛЮДИ" ("ОТЧЕСТВО");

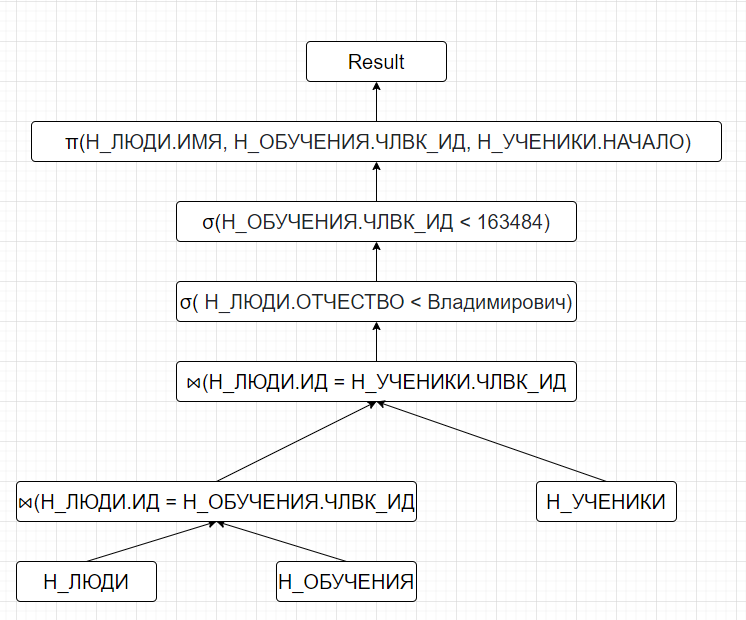
CREATE INDEX " ИНД\_ОБУЧЕНИЕ\_ЧЛВК\_ИД " ON ""Н\_ОБУЧЕНИЯ " ("ЧЛВК\_ИД ");

CREATE INDEX "ИНД\_ОБУЧЕНИЕ\_ЧЛВК\_ИД" ON "Н\_СЕССИЯ" ("ЧЛВК\_ИД");

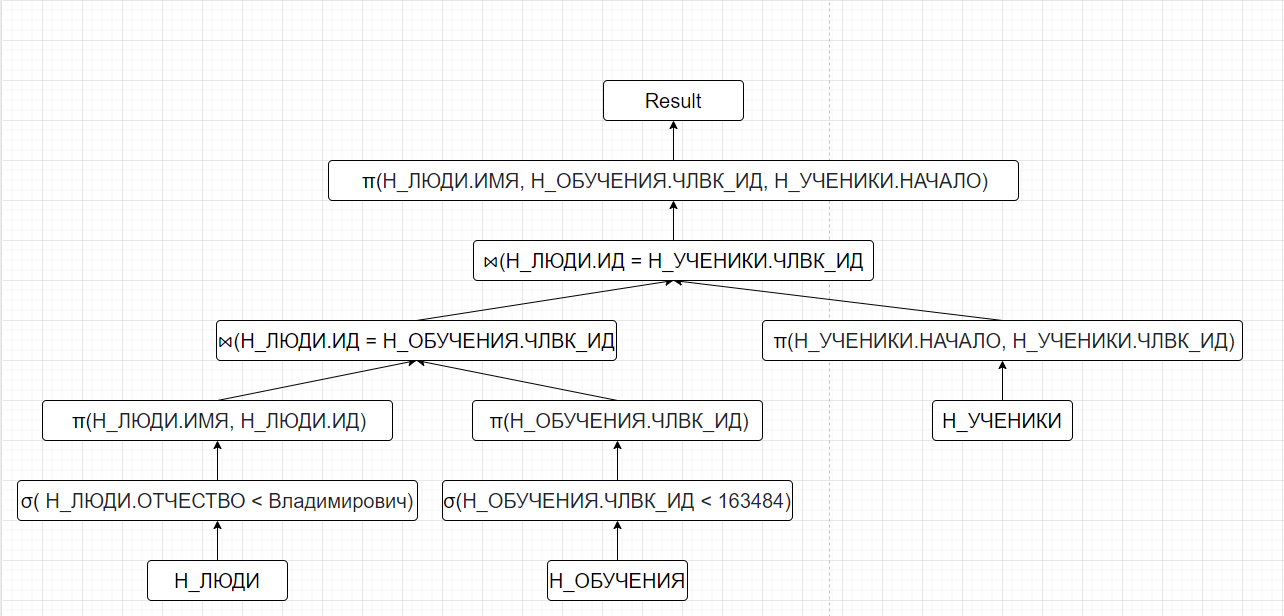
CREATE INDEX "ИНД\_УЧЕНИКИ\_ЧЛВК\_ИД" ON "Н\_ВЕДОМОСТИ" USING HASH("ЧЛВК\_ИД");

Возможные планы выполнения запроса

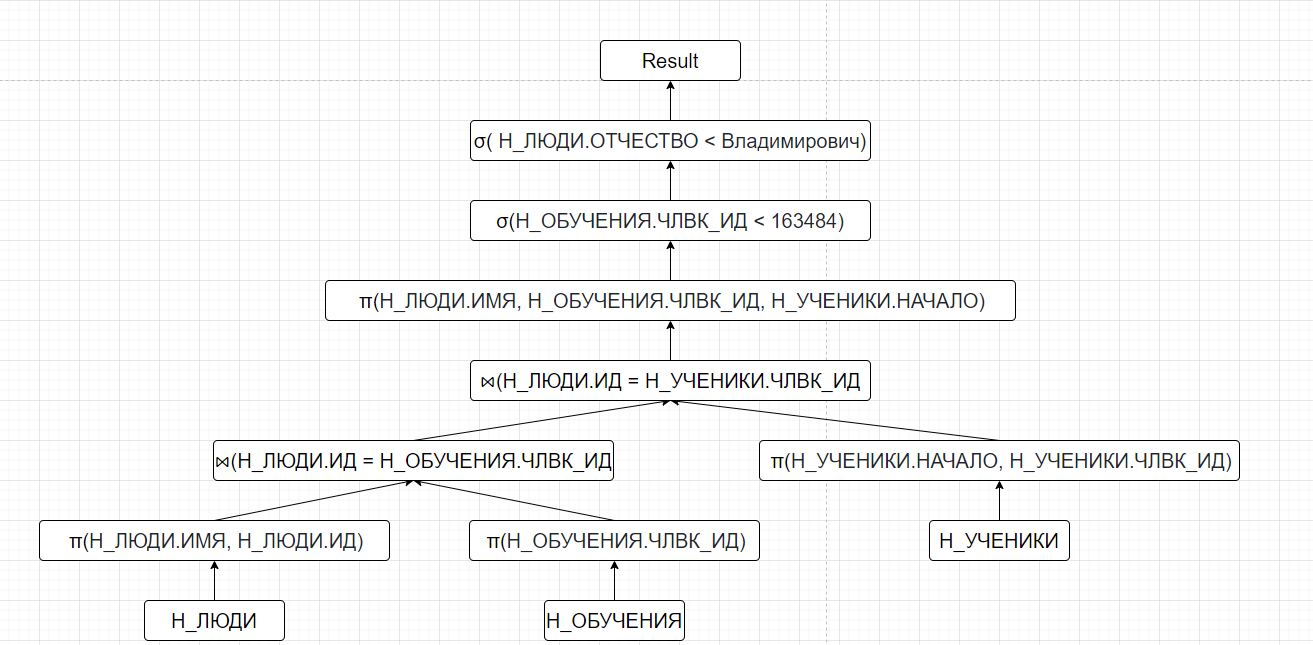
А)



Б)



В)



SELECT COUNT(\*) FROM "Н\_ЛЮДИ"; -- 5118

SELECT COUNT(\*) FROM "Н\_ОБУЧЕНИЯ"; -- 5021

SELECT COUNT(\*) FROM "Н\_УЧЕНИКИ"; -- 23311

SELECT COUNT(\*) FROM "Н\_ЛЮДИ" WHERE "Н\_ЛЮДИ"."ОТЧЕСТВО" < 'Владимирович'; -- 2104

SELECT COUNT(\*) FROM "Н\_ОБУЧЕНИЯ" WHERE "ЧЛВК\_ИД" < 163484; -- 5020

SELECT COUNT(\*) FROM "Н\_ЛЮДИ"

JOIN "Н\_УЧЕНИКИ" pup on "Н\_ЛЮДИ"."ИД" = pup."ЧЛВК\_ИД"; -- 23311

SELECT COUNT(\*) FROM "Н\_ЛЮДИ"

JOIN "Н\_ОБУЧЕНИЯ" ob on "Н\_ЛЮДИ"."ИД" = ob."ЧЛВК\_ИД"; -- 5021

SELECT COUNT(\*) FROM "Н\_ЛЮДИ"

JOIN "Н\_ОБУЧЕНИЯ" ob on "Н\_ЛЮДИ"."ИД" = ob."ЧЛВК\_ИД"

JOIN "Н\_УЧЕНИКИ" pup on "Н\_ЛЮДИ"."ИД" = pup."ЧЛВК\_ИД"; -- 23636

Оптимальным планом будет Б), т.к. в нем после выполнения последнего соединения останется очень мало строк от их общего числа. С помощью проекций и фильтрации на начальных этапах мы обеспечиваем соединение только нужных атрибутов, поэтому промежуточные данные малы.   
  
Мы используем INNER JOIN, поэтому у нас в результат идут только строки, в которых выполняется условие объединения. Как мы видим, результат не зависит от того, левая таблица или правая, потому что мы находим соответствия между таблицами, поэтому использовать LEFT JOIN или RIGHT JOIN не имеет смысла.  
  
Лучше всего использовать Hash Join, т.к. у нас у нас сравнения по id, к тому же этот вид отлично подходят для больших соединений, а так как после фильтрации остается достаточное количество строк, я думаю, что будет использоваться Hash Join.

Предложенные мною хэш индексы ускорят работу за счет того, что приходится постоянно сравнивать числа, а также BTREE поможет нам делать фильтрацию значений с оператором <.

Вывод Explain Analyze

Изображение выглядит как текст

Автоматически созданное описание

Как мы видим, план СУБД не совсем совпадает с моими рассуждениями. PostgreSQL выбрал Hash Join и сам выбор таблиц совпадает с моим, но мне не понятен Seq Scan на первую таблицу, а потом уже вычисления hash функции для строк, хотя с ними ничего кроме соединения не происходит. Вместо того, чтобы делать проверки сразу, СУБД сначала соединяет таблицы, а потом делает проверки с помощью фильтров и использованием BTREE.

## Выводы

Выполнив лабораторную работу, я изучил 2 SQL запроса и разобрался в планах их выполнения. Кроме того, я изучил особенности составления планов в PostgreSQL с помощью или без индексов, а также я понял, какие бывают индексы и стратегии соединения таблиц.