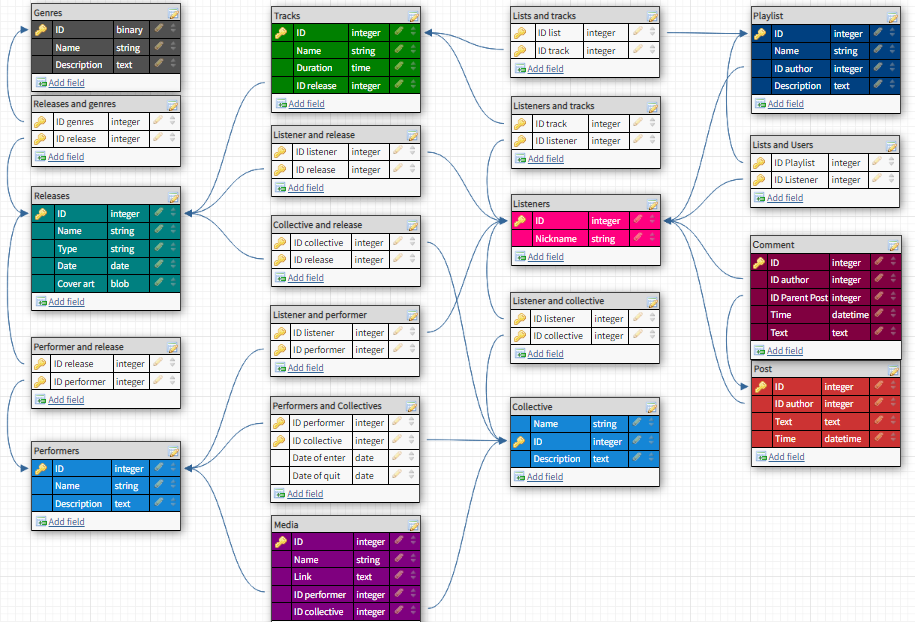
|  |  |  |
| --- | --- | --- |
| Лабораторная работа № 5  Индексы, хранимые процедуры, представления, триггеры | Ф.И.О. | Белоусов Дмитрий Вадимович |
| Группа | ИВТ-260 |
| Преподаватель | Соколов Александр Александрович |
| Дата сдачи | 14.06.2022 |

# Выполнение лабораторной работы

## Реляционная схема:



## Код:

[Ссылка на проект в Github, где можно найти файл с sql-скриптами](https://github.com/coldousedbird/DataBasesProject.git)

## Индексы

1. -- Показать «любимые» треки

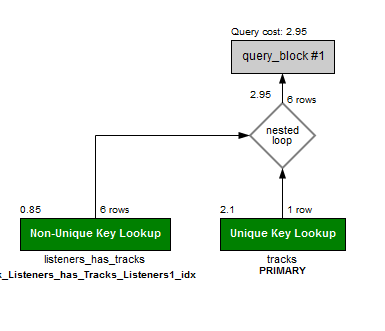
SELECT tracks.Name FROM listeners\_has\_tracks INNER JOIN tracks

ON idTracks = Tracks\_idTracks

WHERE Listeners\_idListeners = 1;

-- EXEC TIME = 0.017 sec

CREATE INDEX TracksName ON Tracks(Name);

-- EXEC TIME AFTER INDEX ADD = 0.0004 sec

2. -- Найти трек по части названия

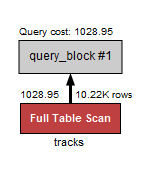
SELECT idTracks, Name FROM tracks

WHERE Name LIKE "%patella%";

-- EXEC TIME = 0.016 sec.

CREATE INDEX TracksName ON Tracks(Name);

-- EXEC TIME AFTER INDEX ADD = 0.006 sec



3. -- Просмотреть дату выпуска релиза

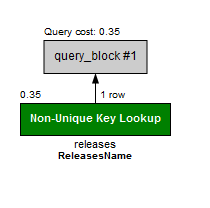
SELECT Date, Name FROM releases

WHERE Name = "Cavalcade";

-- EXEC TIME = 0.016 sec.

CREATE INDEX ReleasesName ON releases(Name);

-- EXEC TIME AFTER INDEX ADD = 0.00046 sec



4. -- Показать количество слушателей коллектива

SELECT collectives.Name, COUNT(listeners\_has\_collectives.Listeners\_idListeners) AS Adds FROM listeners\_has\_collectives

JOIN collectives ON listeners\_has\_collectives.Collectives\_idCollective = collectives.idCollective

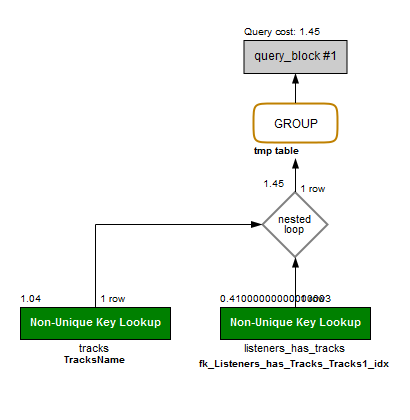
WHERE collectives.Name = "black midi"

GROUP BY Collectives\_idCollective;

-- EXEC TIME = 0.046 sec.

CREATE INDEX CollectivesName ON collectives(Name);

-- EXEC TIME AFTER INDEX ADD = 0.00414 sec



-- 5. Вывести все посты и их комментарии за заданный промежуток времени и приджойнить авторов

SELECT "Post" as "", listeners.Nickname, post.text, post.DateTime, post.idPost as idPost FROM post

INNER JOIN listeners ON listeners.idListeners = post.Listeners\_idListeners

UNION

SELECT "Comment", listeners.Nickname, comments.text, comments.DateTime, comments.Post\_idPost as idPost FROM comments

INNER JOIN listeners ON listeners.idListeners = comments.Listeners\_idListeners

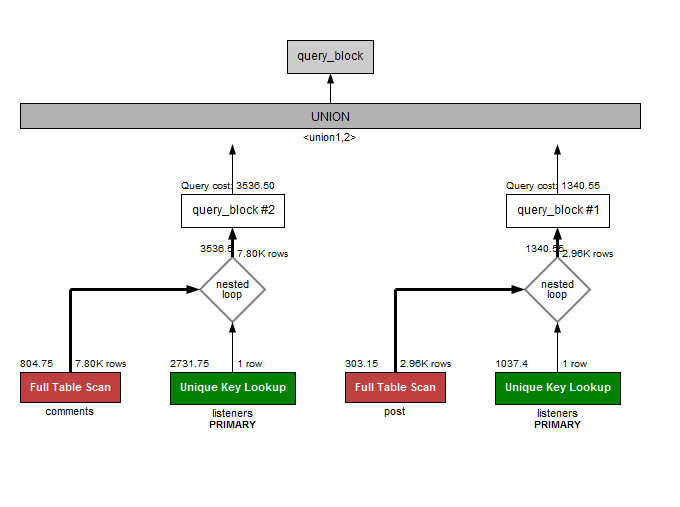
ORDER BY idPost, DateTime;

-- EXEC TIME = 0.217 sec.

CREATE INDEX PostDateTime ON Post(DateTime);

CREATE INDEX CommentDateTime ON Comments(DateTime);

-- EXEC TIME AFTER INDEX ADD = 0.153 sec



-- 6. Для всех коллективов вывести всех участников, через group concat, которые на текущий момент состоят в группе

SELECT collectives.Name, GROUP\_CONCAT(performers.Name) as "Members" FROM collectives

INNER JOIN collectives\_has\_performers ON collectives\_has\_performers.Collectives\_idCollective = collectives.idCollective

INNER JOIN performers ON collectives\_has\_performers.Performers\_idPerformer = performers.idPerformer

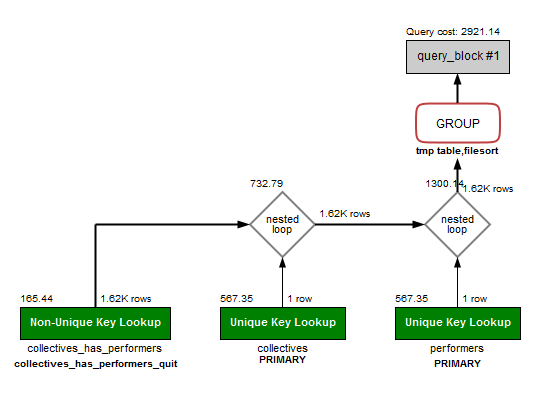
WHERE collectives\_has\_performers.QuitDate = 0

GROUP BY collectives.Name;

-- EXEC TIME = 0.016 sec.

CREATE INDEX collectives\_has\_performers\_quit ON collectives\_has\_performers(QuitDate);

-- EXEC TIME AFTER INDEX ADD = 0.000 sec



## Хранимые процедуры и функции

1. Написать для своей БД 3 процедуры и 3 функции, а также написать как минимум по 1 SQL-запросу вызова каждой из них для тестирования результатов их работы.
2. В процедуры и функции обязательно передавать параметры, различных видов, как минимум в 3 из них использовать локальные переменные.
3. В процедурах и функциях можно использовать созданные в 4-й лабораторной работе запросы. Как минимум в 3 из них обязательно использовать операторы IF, CASE, LOOP, ISNULL и др.

В протокол включить для каждой процедуры и функции:

* краткое описание того, что процедура или функция делает
* код процедуры или функции
* скриншот с результатами вызова процедуры или функции с данными из вашей БД

## Процедуры

1. Выводит все посты и комменты заданного полязователя

-- 1 Output All user's posts and comments

DELIMITER $$

CREATE PROCEDURE users\_posts\_n\_comments(IN idListener INT)

BEGIN

SELECT "Post" as "", listeners.Nickname, post.text, post.DateTime, post.idPost as idPost FROM post

INNER JOIN listeners ON listeners.idListeners = post.Listeners\_idListeners

WHERE listeners.idListeners = idListener

UNION

SELECT "Comment", listeners.Nickname, comments.text, comments.DateTime, comments.Post\_idPost as idPost FROM comments

INNER JOIN listeners ON listeners.idListeners = comments.Listeners\_idListeners

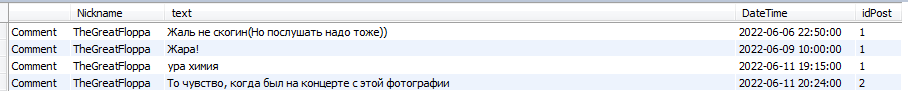
WHERE listeners.idListeners = idListener

ORDER BY idPost, DateTime;

END $$

DELIMITER ;

CALL users\_posts\_n\_comments(2);



2. Выводит всех авторов трека (и коллективы, и отдельных исполнителей)

-- 2 output all authors of track (both collectives and performers)

DELIMITER $$

CREATE PROCEDURE tracks\_authors(IN track\_name VARCHAR(100))

BEGIN

SELECT tracks.Name, performers.Name FROM tracks

INNER JOIN performers\_has\_releases ON performers\_has\_releases.Releases\_idReleases = tracks.Releases\_idReleases

INNER JOIN performers ON performers.idPerformer = performers\_has\_releases.performers\_idPerformers

WHERE tracks.Name = track\_name

UNION ALL

SELECT tracks.Name, collectives.Name FROM tracks

INNER JOIN collectives\_has\_releases ON collectives\_has\_releases.Releases\_idReleases = tracks.Releases\_idReleases

INNER JOIN collectives ON collectives.idCollective = collectives\_has\_releases.Collectives\_idCollective

WHERE tracks.Name = track\_name;

END $$

DELIMITER ;

CALL tracks\_authors('Back To Back');



3. Выводит топ жанров/треков/релизов с размерностью N

-- 3 OUTPUT TOP-N

DELIMITER $$

CREATE PROCEDURE top(IN tableName VARCHAR(50), N int)

BEGIN

IF (tableName = 'genres') THEN

(SELECT \* FROM genres\_adds

ORDER BY Adds DESC LIMIT N);

end if;

IF (tableName = 'tracks') THEN

(SELECT \* FROM tracks\_adds

ORDER BY Adds DESC LIMIT N);

end if;

IF (tableName = 'releases') THEN

(SELECT \* FROM releases\_adds

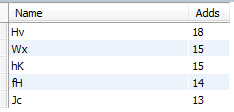
ORDER BY Adds DESC LIMIT N);

end if;

END $$

DELIMITER ;

CALL top ('releases', 5);



## Функции:

1. Считает длительность релиза

-- 1 Counts release's whole duration

delimiter $$

CREATE FUNCTION release\_duration( ReleaseName VARCHAR (45) )

RETURNS TIME

DETERMINISTIC

BEGIN

RETURN (SELECT SEC\_TO\_TIME(SUM(TIME\_TO\_SEC(Duration))) as Duration FROM releases

INNER JOIN tracks ON tracks.Releases\_idReleases = releases.idReleases

WHERE releases.Name = ReleaseName

);

end $$

delimiter ;

SELECT release\_duration('The Fiancee');



2. Считает количество треков в релизе

-- 2 Counts num of tracks in release

delimiter $$

CREATE FUNCTION release\_num\_of\_tracks( ReleaseName VARCHAR (45) )

RETURNS INT

DETERMINISTIC

BEGIN

RETURN (SELECT COUNT(\*) as Num\_Of\_Tracks FROM releases

INNER JOIN tracks ON tracks.Releases\_idReleases = releases.idReleases

WHERE releases.Name = ReleaseName

);

end $$

delimiter ;

SELECT release\_num\_of\_tracks('The Fiancee');



3. Рассчитывает текущее количество участников в коллективе

-- 3 count current numbers of members in collective

delimiter $$

CREATE FUNCTION current\_members\_in\_collective( collective\_name VARCHAR (100) )

RETURNS INT

DETERMINISTIC

BEGIN

RETURN (

SELECT COUNT(\*) FROM collectives\_has\_performers

INNER JOIN collectives ON collectives.idCollective = collectives\_has\_performers.Collectives\_idCollective

WHERE collectives.Name = collective\_name AND collectives\_has\_performers.QuitDate = 0

);

end $$

delimiter ;

DROP FUNCTION track\_find;

SELECT current\_members\_in\_collective('black midi');



SELECT current\_members\_in\_collective('The Pink Floyd');



SELECT current\_members\_in\_collective('Norma Jean');



## Представления

1. Количество добавлений жанров

-- 1 number of genres adds

CREATE VIEW genres\_adds AS

SELECT genres.name, COUNT(listeners\_has\_releases.Listeners\_idListeners) as Adds FROM genres

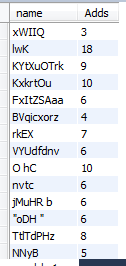
INNER JOIN releases\_has\_genres ON releases\_has\_genres.Genres\_idGenres = genres.idGenres

INNER JOIN releases ON releases.idReleases = releases\_has\_genres.Releases\_idReleases

INNER JOIN listeners\_has\_releases ON listeners\_has\_releases.Releases\_idReleases = releases.idReleases

GROUP BY genres.name;

SELECT \* FROM genres\_adds;



2. Количество добавлений треков

-- 2 number of tracks adds

CREATE VIEW tracks\_adds AS

SELECT tracks.Name, COUNT(Listeners\_idListeners) as Adds FROM tracks

INNER JOIN listeners\_has\_tracks ON listeners\_has\_tracks.Tracks\_idTracks = tracks.idTracks

GROUP BY tracks.Name;

SELECT \* FROM tracks\_adds;



3. Количество добавлений релизов

-- 3 number of releases adds

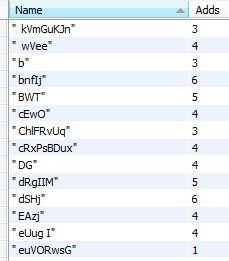
CREATE VIEW releases\_adds AS

SELECT releases.Name, COUNT(listeners\_has\_releases.Listeners\_idListeners) as Adds FROM releases

INNER JOIN listeners\_has\_releases ON listeners\_has\_releases.Releases\_idReleases = releases.idReleases

GROUP BY releases.Name;

SELECT \* FROM releases\_adds;



## Триггер

При присоединении релиза к коллективу, триггер присоединит его ко всем активным участникам группы

-- If release belong to collective, so it belongs to it’s members in this time

-- TRIGGER

CREATE TRIGGER if\_release\_by\_collective\_than\_release\_by\_all\_members

AFTER INSERT

ON collectives\_has\_releases

FOR EACH ROW

INSERT INTO performers\_has\_releases (performers\_idPerformers, Releases\_idReleases)

SELECT collectives\_has\_performers.Performers\_idPerformer, inserted.Releases\_idReleases FROM inserted

INNER JOIN collectives\_has\_performers ON inserted.Collectives\_idCollective = collectives\_has\_performers.Collectives\_idCollective

LEFT JOIN releases ON inserted.Releases\_idReleases = releases.idReleases

WHERE (collectives\_has\_performers.QuitDate = 0 AND releases.Date BETWEEN collectives\_has\_performers.EnterDate AND current\_date())

OR releases.Date BETWEEN collectives\_has\_performers.EnterDate AND collectives\_has\_performers.QuitDate;

DROP TRIGGER if\_like\_release\_by\_collective\_than\_release\_by\_all\_members;

## Модификация:

1. Триггер. Когда слушатель добавляет (лайкает) трек, пусть автоматом лайкает его релиз этого трека

-- 1. Триггер - когда слушатель добавляет (лайкает) трек, пусть автоматом лайкает его релиз этого трека

delimiter $$

CREATE TRIGGER if\_like\_track\_then\_like\_release

AFTER INSERT

ON listeners\_has\_tracks

FOR EACH ROW

BEGIN

INSERT INTO listeners\_has\_releases (Listeners\_idListeners, Releases\_idReleases)

(

SELECT new.Listeners\_idListeners, tracks.Releases\_idReleases FROM tracks

WHERE tracks.idTracks = new.Tracks\_idTracks

);

END$$

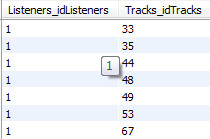
delimiter ;

INSERT INTO listeners\_has\_tracks(Listeners\_idListeners, Tracks\_idTracks) VALUES

(1, 67); 

/\*

В listeners\_has\_tracks появилась запись (1;67) – пользователю 1 добавился трек 67 – John L из альбома Cavalcade



Триггер сработал и в listeners\_has\_releases появилась запись (1;3) – пользователю 1 добавился альбом 3 – Cavalcade



\*/

2. Процедура. Принимает: idListeners или Nickname. Возвращает: количество лайков этого пользователя (4 параметра: треки, релизы, коллективы + перформеры, плейлисты).

-- принимает: idListeners

-- возвращает: количество лайков этого пользователя (4 параметра: треки, релизы, коллективы + перформеры, плейлисты)

DELIMITER $$

CREATE PROCEDURE listeners\_likes(

IN ListenerID INT,

OUT tracks\_likes INT,

OUT releases\_likes INT,

OUT collectives\_likes INT,

OUT performers\_likes INT

)

BEGIN

SELECT COUNT(\*) as Num FROM listeners\_has\_tracks

WHERE listeners\_has\_tracks.Listeners\_idListeners = ListenerID

INTO tracks\_likes;

SELECT COUNT(\*) as Num FROM listeners\_has\_releases

WHERE listeners\_has\_releases.Listeners\_idListeners = ListenerID

INTO releases\_likes;

SELECT COUNT(\*) as Num FROM listeners\_has\_collectives

WHERE listeners\_has\_collectives.Listeners\_idListeners = ListenerID

INTO collectives\_likes;

SELECT COUNT(\*) as Num FROM listeners\_has\_performers

WHERE listeners\_has\_performers.Listeners\_idListeners = ListenerID

INTO performers\_likes;

END $$

DELIMITER ;

SET @tracks\_likes := 0,

@releases\_likes := 0,

@collectives\_likes := 0,

@performers\_likes := 0;

CALL listeners\_likes(3, @tracks\_likes, @releases\_likes, @collectives\_likes, @performers\_likes);

SELECT @tracks\_likes, @releases\_likes, @collectives\_likes, @performers\_likes;



3. Составной индекс использовать в запросе, выводящем названия релизов заданного типа, выпущенные в промежутке между некоторыми датами

SELECT \* FROM releases

WHERE Type = 'Album' AND Date BETWEEN "1970-01-01" AND "1980-01-01";



CREATE INDEX type\_n\_date ON releases(Type, Date);

