|  |  |  |
| --- | --- | --- |
| **Проектирование баз данных** Лабораторная работа № **5** Индексы,хранимые процедуры, представления, триггеры **Вариант 27**  Тема: Компьютерные игры | Ф.И.О. | Клычёв Д. С. |
| Группа | ИВТ-262 |
| Преподаватель | Гаис Аль - Мерри |
| Дата сдачи |  |

1. Добавить в большинство основных таблиц в БД **минимум по 10 000 записей, лучше - больше**(если в другие по логике невозможно) через файлы **CSV** с помощью генераторов данных, **Excel** или других инструментов сохранения в CSV-файлы, а также команды **COPY** (**LOAD DATA** в MySQL и т.п. в других СУБД).

2. Запустить самые сложные, составные запросы из лабораторной работы №4 и **измерить их время выполнения** с помощью приведённых в лекции инструментов.

3. Выбрать из них **5-10 самых затратных по времени** и **проанализировать** их с помощью команды **EXPLAIN**. Также использовать **визуализатор планов** в графической оболочке СУБД. Построить **визуальные планы выполнения запросов**.

Также можно выбрать те запросы, которые будут потенциально самыми **часто используемыми**, **популярными** у клиентов вашей БД (приложений), которые будет рационально ускорить для ускорения работы приложения. Например, запрос поиска книги по названию и/или автору (скорее всего, чаще книги будут искать именно так). 

**Для каждого из выбранных запросов по очереди:**

4. Приложить в протокол **код выбранного SQL-запроса**с пояснением, что он делает.

5. Добавить**скриншот плана выполнения запроса** в протокол, результаты вызовов функции **EXPLAIN**(особенно важно для PostgreSQL, т.к. на графическом плане не отображаются дополнительные данные)и время выполнения**до**дальнейших оптимизаций.

6. С помощью **добавления индексов** в свою БД ускорить этот запрос. Приложить **код добавления индексов**, дать комментарии, почему были выбраны именно такие индексы.

7. **Построить новый визуальный план** выполнения запроса с индексами и **повторно использовать команду EXPLAIN**, добавить их в протокол, а также новую измеренную скорость выполнения запроса. Проанализировать изменения в скорости выполнения запроса, сделать выводы.

8. Переходим к следующему запросу.

#### 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.

4. **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

#### Представления

1. Создать **3 представления** с помощью оператора **VIEW**, желательно объединяющих данные из **нескольких таблиц.**

2. Написать **SQL-запросы вызова** всех созданных представлений.

3. **В протокол включить:**

* краткое описание того, что представление хранит
* код создания представления
* скриншот результата вызова SQL-запроса обращения к представлению с данными из вашей БД

#### Триггеры

1. Создать **минимум 1 триггер.**

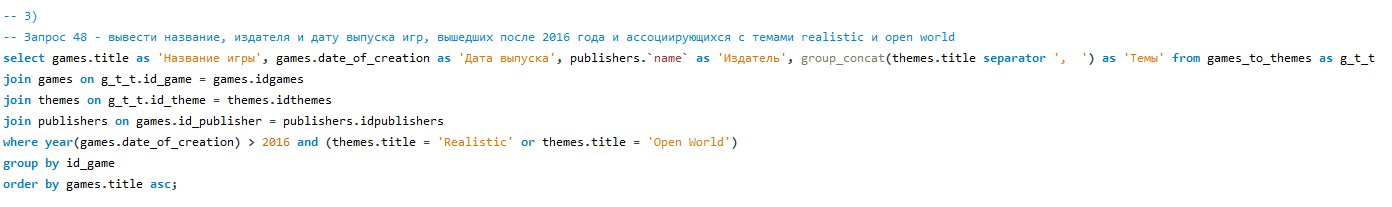
2. **В протокол включить:**

* краткое описание того, что делает триггер
* код создания триггера
* скриншот с примером результата работы триггера

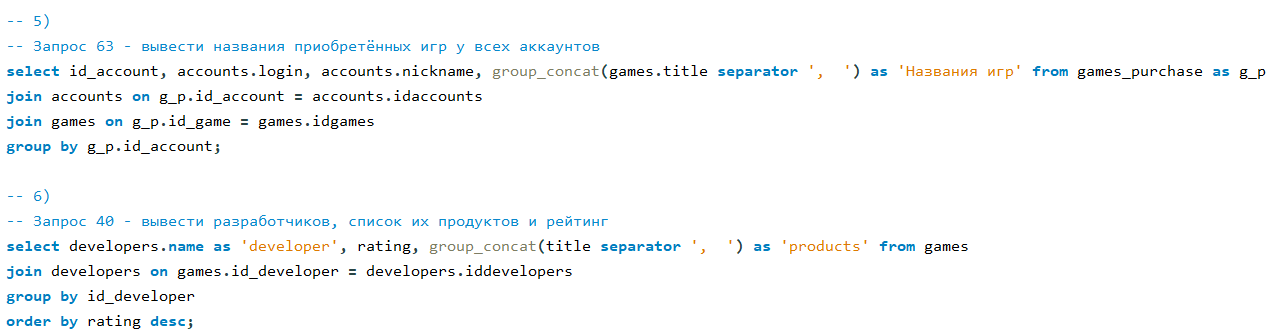
**Индексы**:

Запросы, в первую очередь, я выбирал по принципу актуальности для моей базы данных (игровая платформа):









Все они связаны с игровой тематикой и соответствующими поисковыми запросами соответственно.

Теперь **проанализируем** порядок выполнения каждого запроса:

1 запрос:

-- 1)

-- Запрос 46 - вывести игры с жанром Role-playing и Action

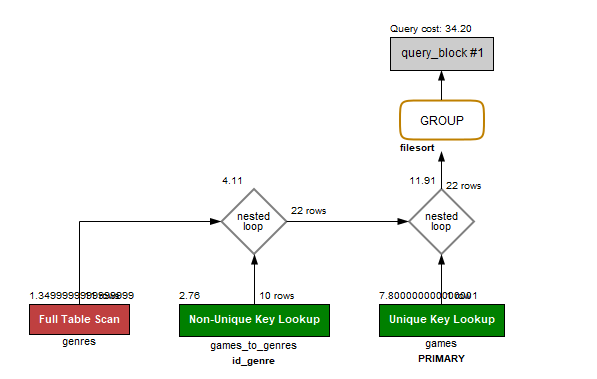
select genres.title as 'Название жанра', group\_concat(games.title separator ', ') as 'Игры, соответствующие выбранному жанру' from games\_to\_genres

join games on games\_to\_genres.id\_game = games.idgames

join genres on games\_to\_genres.id\_genre = genres.idgenres

where genres.title = 'Role-Playing' or genres.title = 'Action'

group by genres.title;





Как мы видим, используется метод full table scan, перебирающий все строки, что будет неэффективно при большом наборе данных.

2 запрос:

-- 2)

-- Запрос 47 - вывести игры, которые являются многопользовательскими

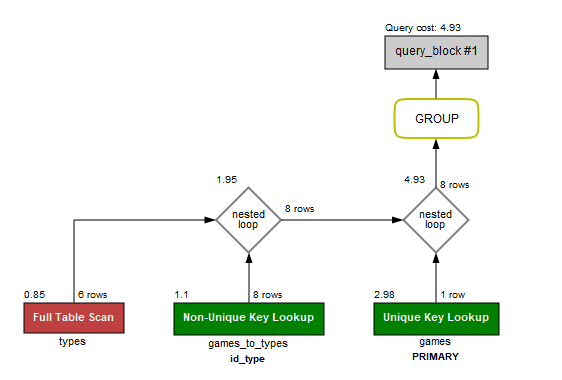
select types.title as 'Тип', group\_concat(games.title separator ', ') as 'Игры, соответствующие выбранному типу' from games\_to\_types

join games on games\_to\_types.id\_game = games.idgames

join types on games\_to\_types.id\_type = types.idtypes

where types.title = 'Multiplayer'

group by types.title;





Используется всё тот же метод full table scan, перебирающий все строки, что будет неэффективно при большом наборе данных.

3 запрос:

-- 3)

-- Запрос 48 - вывести название, издателя и дату выпуска игр, вышедших после 2016 года и ассоциирующихся с темами realistic и open world

select games.title as 'Название игры', games.date\_of\_creation as 'Дата выпуска', publishers.`name` as 'Издатель', group\_concat(themes.title separator ', ') as 'Темы' from games\_to\_themes as g\_t\_t

join games on g\_t\_t.id\_game = games.idgames

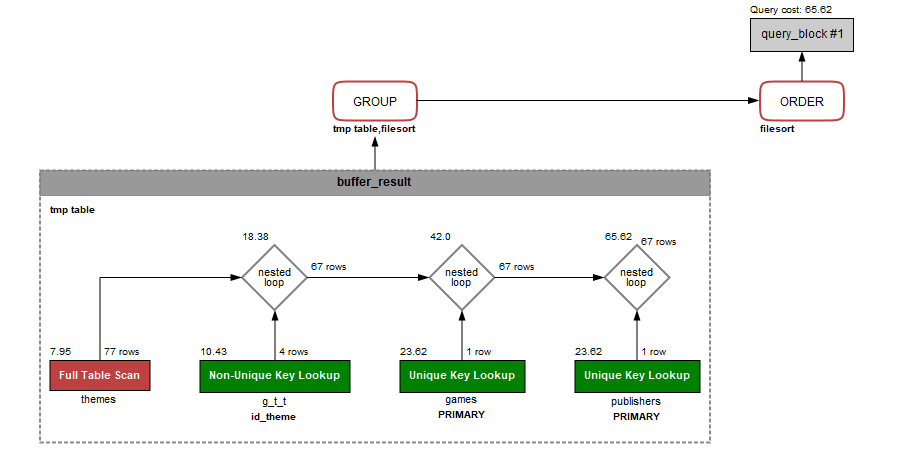
join themes on g\_t\_t.id\_theme = themes.idthemes

join publishers on games.id\_publisher = publishers.idpublishers

where year(games.date\_of\_creation) > 2016 and (themes.title = 'Realistic' or themes.title = 'Open World')

group by id\_game

order by games.title asc;





Запрос 4:

-- 4)

-- Запрос 57 - показать список рекомендаций у каждого аккаунта

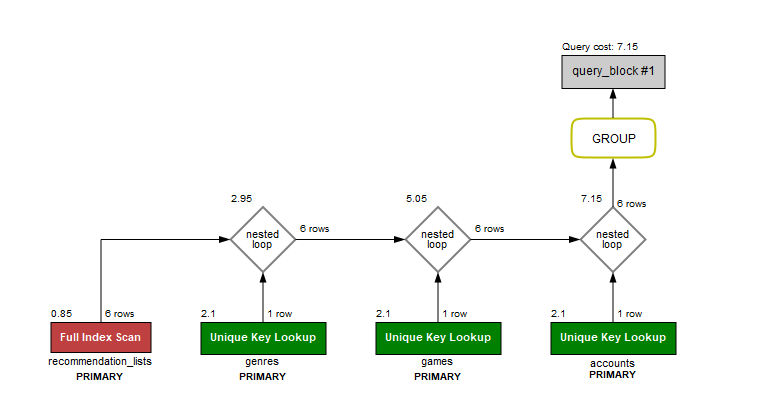
select accounts.login as 'Логин', accounts.idaccounts as 'id списка рекомендаций (соответствует id аккаунта)',genres.title as 'Преобладающий жанр', games.title as 'Игра, рекомендуемая к приобретению', recommendation\_lists.date\_of\_creation as 'Последнее обновление' from recommendation\_lists

join accounts on recommendation\_lists.id\_account = accounts.idaccounts

join games on recommendation\_lists.id\_proposed\_game = games.idgames

join genres on recommendation\_lists.id\_prevailing\_genre = genres.idgenres

group by  recommendation\_lists.id\_account;





Запрос 5:

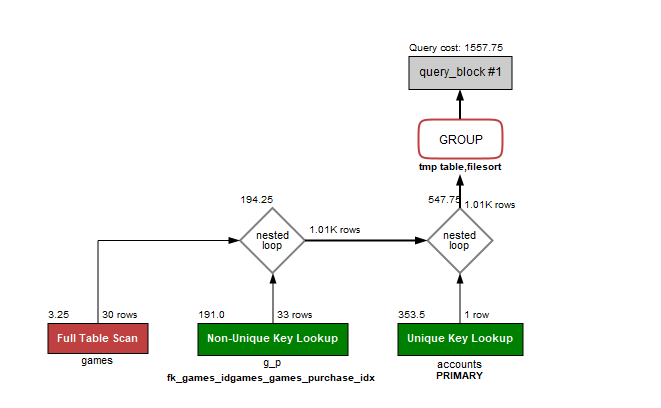
-- 5)

-- Запрос 63 - вывести названия приобретённых игр у всех аккаунтов

select id\_account, accounts.login, accounts.nickname, group\_concat(games.title separator ', ') as 'Названия игр' from games\_purchase as g\_p

join accounts on g\_p.id\_account = accounts.idaccounts

join games on g\_p.id\_game = games.idgames

group by g\_p.id\_account;



Запрос 6:

-- 6)

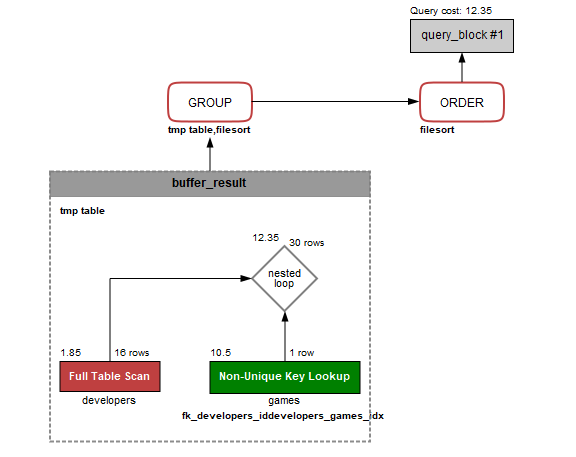
-- Запрос 40 - вывести разработчиков, список их продуктов и рейтинг

select developers.name as 'developer', rating, group\_concat(title separator ', ') as 'products' from games

join developers on games.id\_developer = developers.iddevelopers

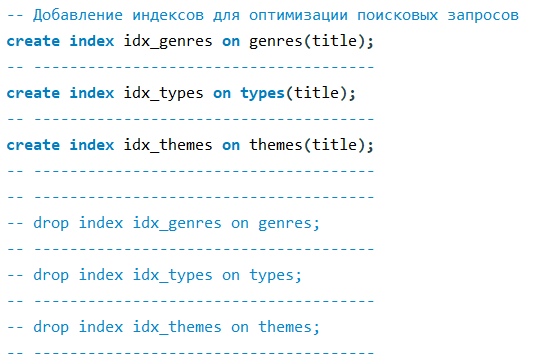
group by id\_developer

order by rating desc;





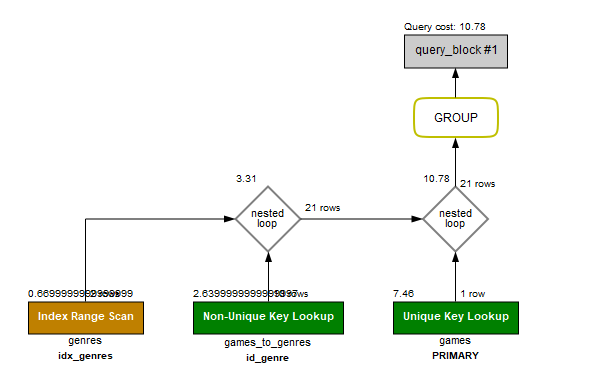
Добавление **индексов** для ускорения запросов:



Добавление данных индексов значительно ускоряет первые три поисковых запроса.

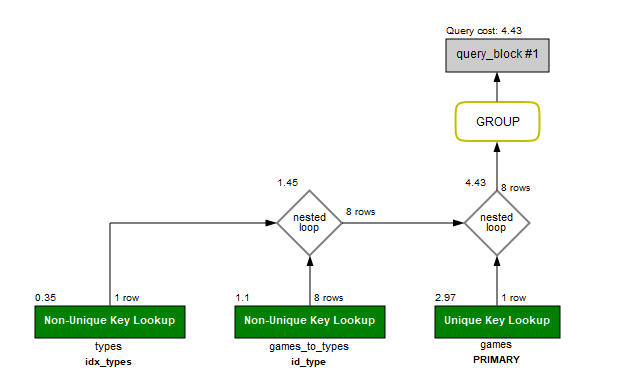
**Проанализируем** порядок выполнения запроса путём построения новой схемы:

**1 запрос** (обновлённый):



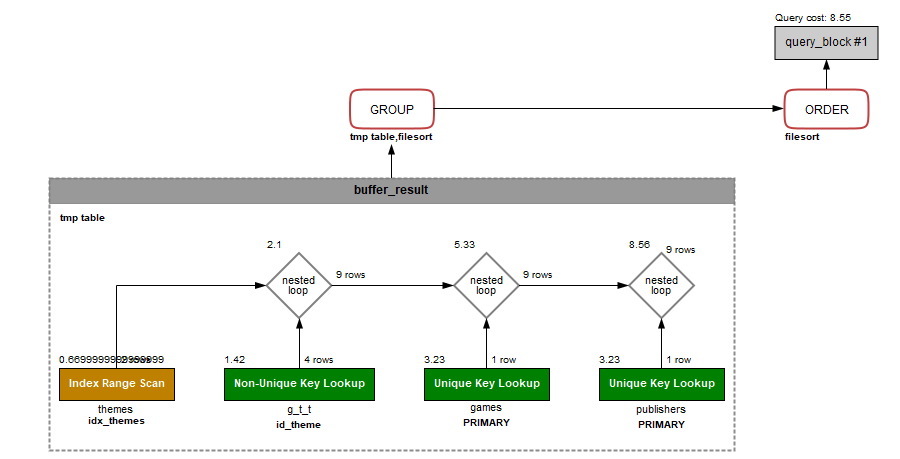


**2 запрос** (обновлённый):





**3 запрос** (обновлённый):

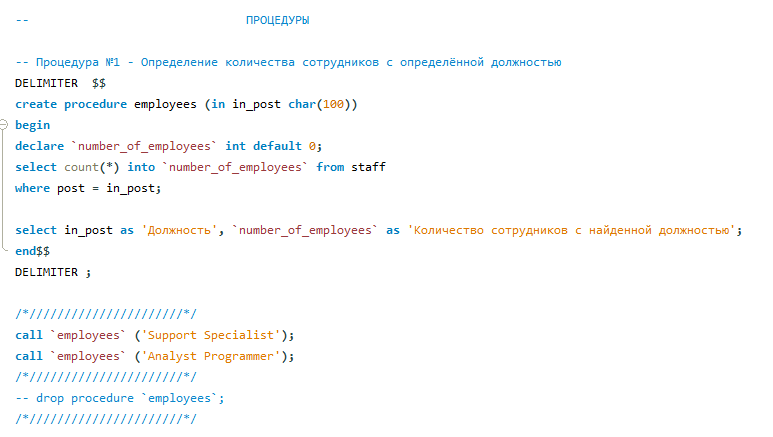


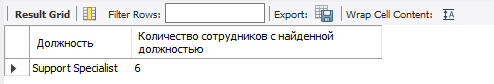


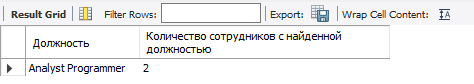
Мы видим **значительное** ускорение выполнения запросов. Full table scan сменился на index range scan, путём добавления индексов, опирающихся на название жанра, темы и типа игры, соответственно.

#### Хранимые процедуры и функции:

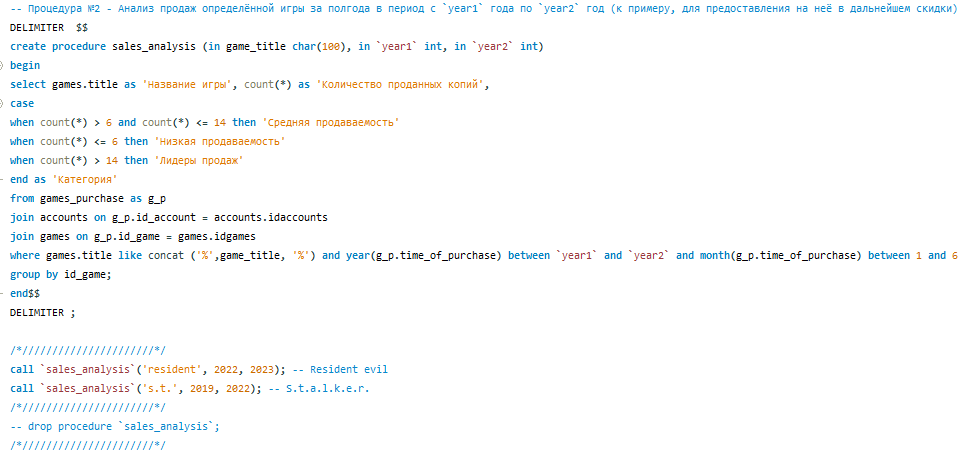
1 процедура:

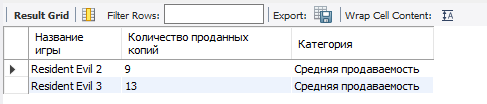


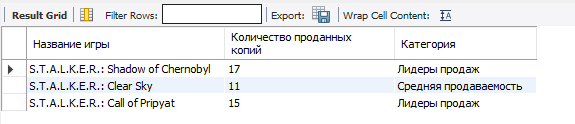




2 процедура:

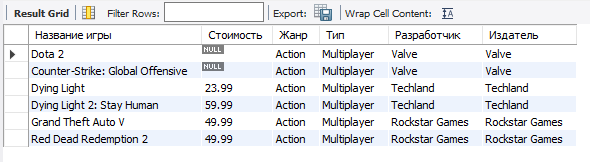


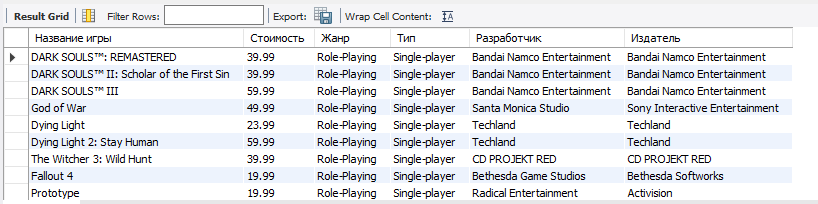




3 процедура:

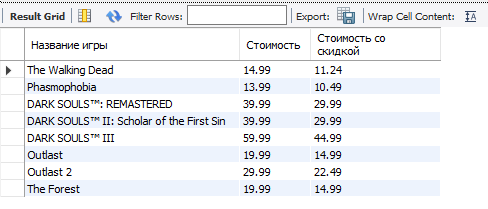




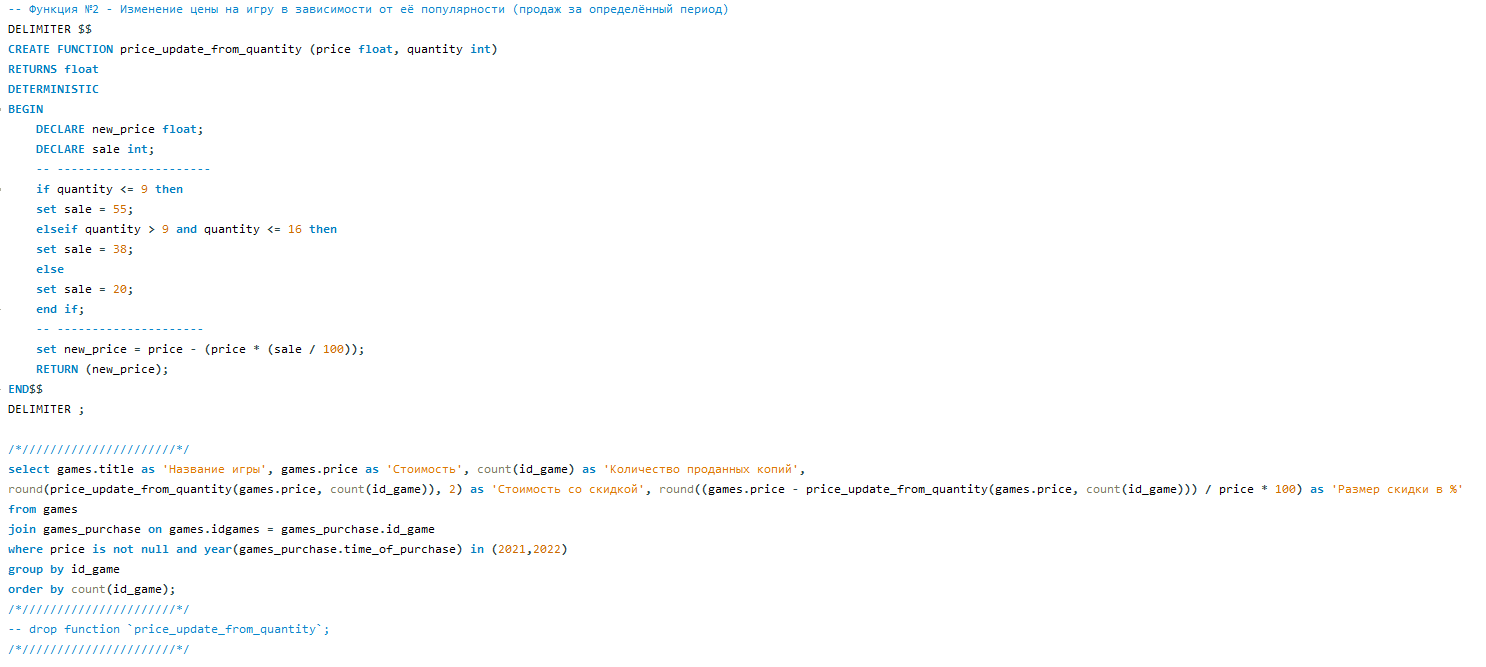


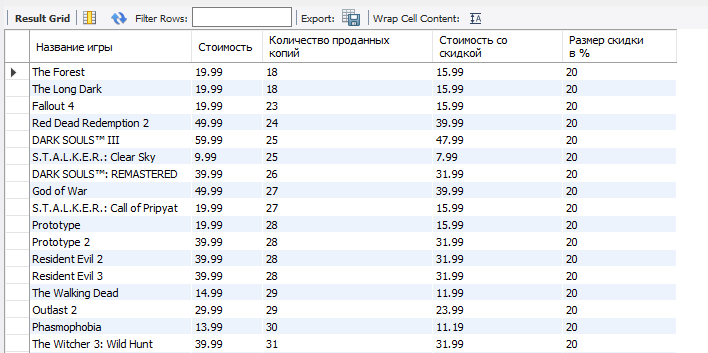
1 функция:



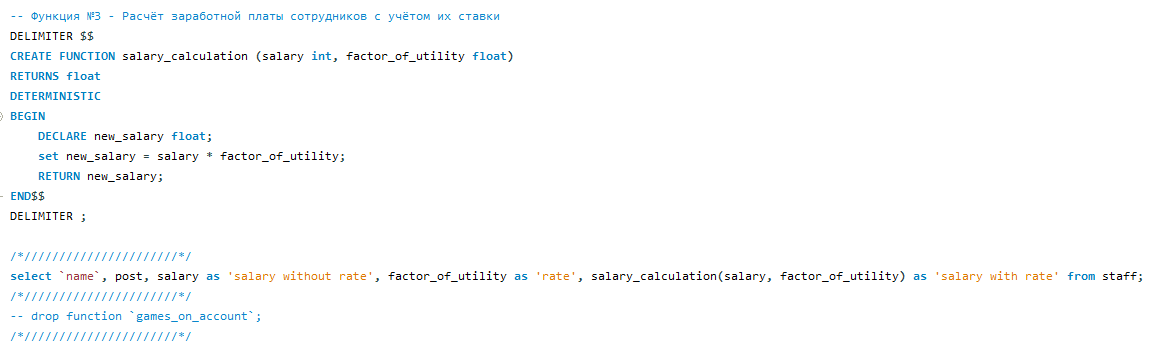


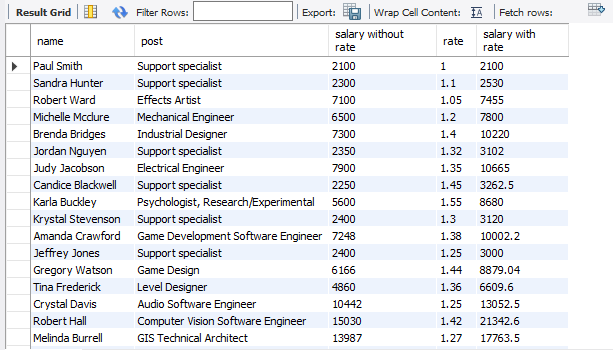
2 функция:





3 функция:

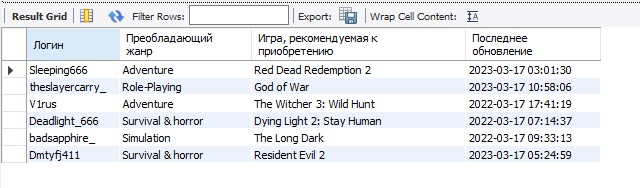




**Представления**:

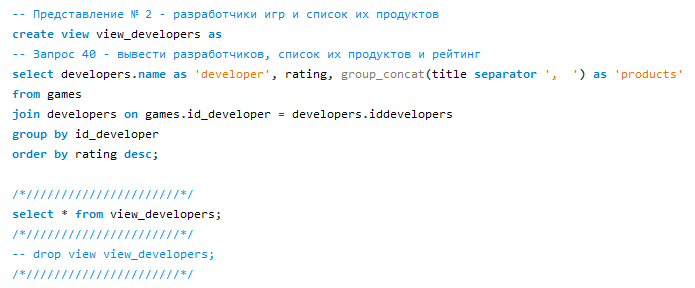
Представление 1:





Представление view\_recommendation\_list выводит список рекомендаций каждого пользователя.

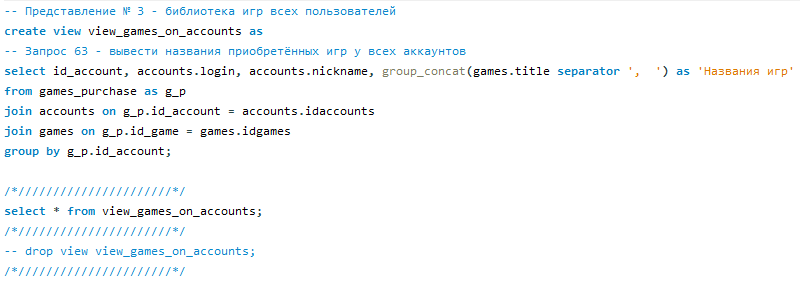
Представление 2:

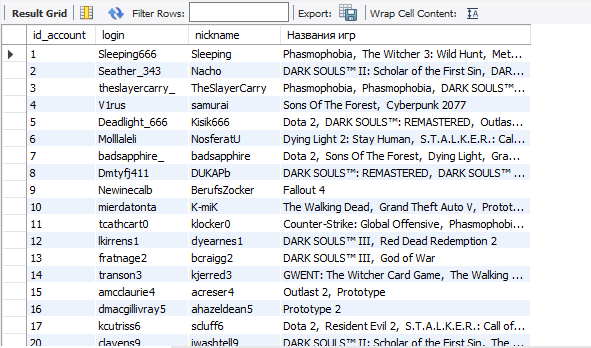




Представление view\_developers выводит всех разработчиков игр и список их продуктов.

Представление 3:





Представление view\_games\_on\_accounts выводит библиотеку игр всех пользователей (список всех игр на аккаунте).

**Триггер:**