ч «Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем

Кафедра №42 «Криптология и кибербезопасность»

Отчёт по лабораторной работе №4

"Безопасность базы данных"

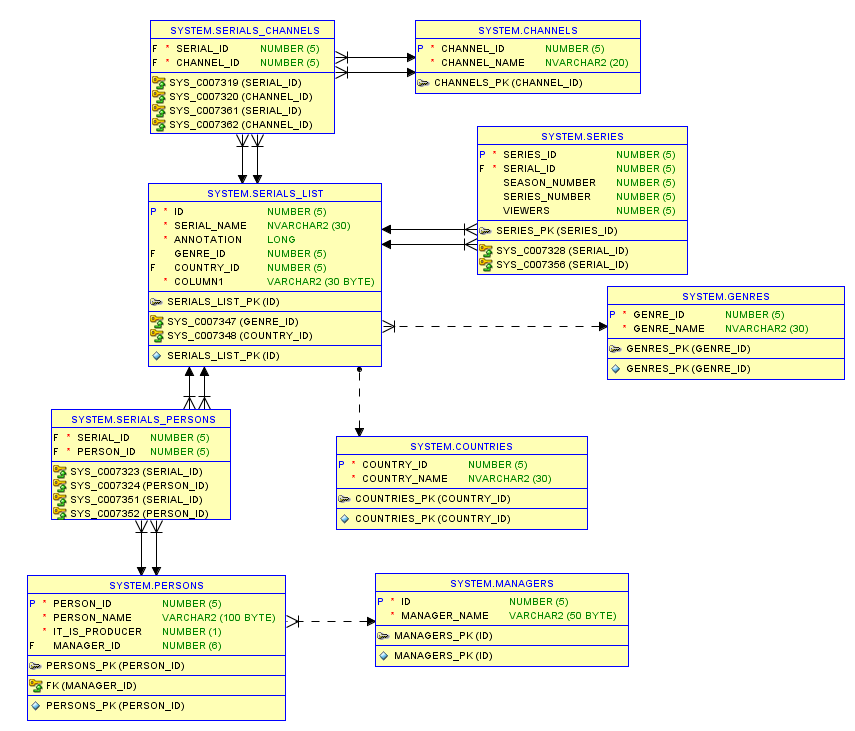
Студент: Нурматов С.Р.

Группа: Б17-565

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2020

**Диаграмма сущностей**

****

**Описание предметной области**

**База данных сериалов.**

В компании BestTVS появился новый сайт, который предоставляет доступ к просмотру сериалов на любой вкус и выбор. BestTVS предоставляет посетителям всю информацию о сериалах: актеры, продюсеры, страна, год выпуска, описание фильма, количество сезонов и серий, а так же каналы, на которых могут транслироваться данные сериалы. Более того, можно увидеть количество зрителей, просматривающих ту или иную серию определенного сериала.

**Разработанные представления**

1. Serials\_actors – В данном представлении выделены главные актеры каждого сериала
2. Serials\_stat – Статистика сериалов с id < 6, имеющие не менее 10 зрителей, просматриваемых какую-либо серию, того или иного сезона и каналы транслирующие тот или иной сериал.

**Цель работы**

Приобретение навыков создания вспомогательных объектов схемы данных, целью которых является упрощение и регулирование работы пользователей базы данных.

**Задание**

1. **Разработать и проверить не менее двух SQL-запросов с использованием разных теоретико-множественных операций: UNION [ALL], INTERSECT, MINUS. Описать их назначение;**
   1. UNION (Оператор UNION используется для объединения результатов двух операторов SELECT)

SELECT person\_name FROM persons

where persons.person\_id < 40

UNION

SELECT person\_name FROM persons, serials\_persons

where serials\_persons.person\_id < 20

* 1. INTERSECT (Оператор INTERSECT имеет противоположный эффект оператора UNION, он объединит результаты двух операторов SELECT и выберет, какие записи дублируются).

SELECT serial\_name

FROM serials\_list

WHERE country\_id = 5

INTERSECT

SELECT serial\_name

FROM serials\_list

WHERE genre\_id = 4;

Этот запрос позволяет получить список сериалов serial\_name из serials\_list где идентификатор страны равен 5 и при этом идентификатор жанра должен быть равен 4;

**Модифицировнный запрос 1\_2**

SELECT serial\_name, genre\_name

FROM serials\_list

JOIN genres

ON serials\_list.id = genres.genre\_id

where id > 4 and id < 6

INTERSECT

SELECT serial\_name, genre\_name

FROM serials\_list

JOIN genres

ON serials\_list.id = genres.genre\_id

Из-за не полного совпадения индексов genre\_id и id, при выполнении запроса состоящего из последниъх 4 строчек результатом будут сериалы с genre\_id 4, 5, 6, так как у жанров индексация начинается с 4 а у сериалов с 1 до 6. При выполении запроса результатом будет сериал с id = 5. (Ссылка на гитхаб в заключении)

* 1. MINUS (MINUS объединит и выберет записи только в первом операторе SELECT, но не во втором операторе SELECT)

SELECT person\_name FROM persons

MINUS

SELECT person\_name FROM persons, serials\_persons

where serials\_persons.person\_id = persons.person\_id

Этот запрос позволяет получить список person\_name, находящихся в таблице persons, но не находящихся в связующей таблице serials\_persons.

1. **Разработать систему представлений, основанную на (возможно расширенном) списке запросов, реализованных ранее в ходе работ 1-1 и 1-2. Важно, чтобы представления не содержали служебных данных, в первую очередь — суррогатных ключей. Для этого необходимо изучить базовые возможности SQL-инструкции CREATE VIEW, а также представление USER INDEXES;**
   1. Это представление всех актеров каждого сериала

CREATE VIEW serials\_actors AS

SELECT serials\_persons.serial\_id, persons.person\_id, serials\_list.serial\_name, persons.person\_name

FROM serials\_list

JOIN serials\_persons

ON serials\_list.id = serials\_persons.serial\_id

JOIN persons

ON serials\_persons.person\_id = persons.person\_id

where persons.it\_is\_producer = 0

* 1. Это представление представляет из себя статистику всех сериалов, которое включает в себя количество зрителей, название сериала, канал, на котором транслируется тот или иной сериал, а так же номер сезона.

CREATE VIEW serial\_stat AS

SELECT series.serial\_id, serials\_list.serial\_name, series.season\_number, channels.channel\_name, series.viewers

FROM series

JOIN serials\_list

ON series.serial\_id = serials\_list.id

JOIN serials\_channels

ON serials\_list.id = serials\_channels.serial\_id

JOIN channels

ON serials\_channels.channel\_id = channels.channel\_id

where series.viewers > 10 and series.serial\_id < 6

ORDER BY series.viewers DESC;

1. **Принять обоснованные меры по ускорению работы этих представлений: рассмотреть возможность создания дополнительных индексов на основе B-дерева или битовых карт, создание кластеров. Можно воспользоваться возможностями autotrace, чтобы убедиться в полезности реализованных индексов. Необходимо изучить базовые возможности инструкций CREATE INDEX, CREATE CLUSTER, а также настройку и использование инструмента autotrace в SQL\*Plus;**
   1. Создание индекса на основе B-дерева:

CREATE INDEX idx\_serials

ON serials\_list(serial\_name);

SELECT serial\_name, annotation FROM serials\_list

WHERE serial\_name != 'Кухня'

3.2.1. Выполнение запроса перед созданием индекса:

SELECT serial\_name, country\_name, genre\_name, annotation

FROM countries

JOIN serials\_list

ON countries.country\_id = serials\_list.country\_id

JOIN genres

ON serials\_list.genre\_id = genres.genre\_id

WHERE serial\_name != 'Кухня'

ORDER BY serials\_list.serial\_name

3.2.2. Создание индекса на основе B-дерева

CREATE INDEX idx\_serials

ON serials\_list(serial\_name);

SELECT serial\_name, country\_name, genre\_name, annotation

FROM countries

JOIN serials\_list

ON countries.country\_id = serials\_list.country\_id

JOIN genres

ON serials\_list.genre\_id = genres.genre\_id

WHERE serial\_name != 'Кухня'

ORDER BY serials\_list.serial\_name;

3.2.3. Выполнение запроса перед созданием индекса

SET AUTOTRACE ON EXPLAIN

SELECT serial\_id, person\_id

FROM serials\_persons

where person\_id < 50

Order by serial\_id

3.2.4. Создание индекса для таблицы на которое опирается представление.

CREATE INDEX idx\_serials\_pers

ON serials\_persons(serial\_id);

SELECT serial\_id, person\_id

FROM serials\_persons

where person\_id < 50

Order by serial\_id

В результате можно увидеть (файлы 3.2.3 и 3.2.4 по ссылке на гитхаб), что после создания индекса поиск быстрее чем раньше.

3.2.5. Создание индекса для таблицы на которое опирается представление.

CREATE INDEX idx\_ser\_actors

ON persons(person\_name);

SELECT id, person\_name as actor\_name

FROM serials\_list

JOIN serials\_persons

ON serials\_list.id = serials\_persons.serial\_id

JOIN persons

ON serials\_persons.person\_id = persons.person\_id

where it\_is\_producer = 0 and id < 6

order by id;

3.3. Невозможно создать индекс для представления, который является просто запросом. Вместо этого создайте индекс над материализованным представлением. Материализованное представление - это таблица, которая создается путем оценки представления, так что вы можете создать индекс по нему.

CREATE MATERIALIZED VIEW producers(Serial\_id, Serial\_name, Country, Producer)

AS

SELECT serial\_id, serial\_name, country\_name, person\_name

FROM countries

JOIN serials\_list

ON countries.country\_id = serials\_list.country\_id

JOIN serials\_persons

ON serials\_list.id = serials\_persons.serial\_id

JOIN persons

ON serials\_persons.person\_id = persons.person\_id

WHERE persons.it\_is\_producer = 1

ORDER BY serial\_id;

В данных запросах использовалось autotrace, чтобы убедиться в полезности реализованных индексов.

SET AUTOTRACE ON EXPLAIN

1. **Где это является осмысленным, реализовать операцию добавления/удаления строк из разработанных представлений. Для этого использовать возможности языка PL/SQL. Например, при добавлении нового заказа в представление с заказами можно проверить, содержится ли заказчик в списке клиентов и добавить его, если нет — требования нормальной формы не должны отягощать конечного пользователя! Необходимо изучить основы PL/SQL, а также возможности инструкции CREATE TRIGGER;**

Задание - реализовать операцию добавления/удаления строк из представления

4.1. Файл для добавления

CREATE OR REPLACE TRIGGER insert\_actors

INSTEAD OF INSERT ON serials\_actors

REFERENCING NEW AS cnt

DECLARE

rowcnt number;

BEGIN

SELECT COUNT(\*) INTO rowcnt FROM serials\_list

WHERE id =: cnt.serial\_id;

IF rowcnt = 0 THEN

INSERT INTO serials\_list (id, serial\_name) VALUES (:cnt.serial\_id, :cnt.serial\_name);

DBMS\_OUTPUT.put\_line ('New serial added to serials\_list successfully!');

END IF;

SELECT COUNT(\*) INTO rowcnt FROM persons WHERE person\_id=:cnt.person\_id;

IF rowcnt = 0 THEN

INSERT INTO persons(person\_id, person\_name) VALUES (:cnt.person\_id, :cnt.person\_name);

DBMS\_OUTPUT.put\_line ('New actor added to persons successfuly!');

INSERT INTO serials\_persons (serial\_id, person\_id) VALUES (:cnt.serial\_id, :cnt.person\_id);

END IF;

END;

Результатом данного запроса будет срабатывание триггера на вставку в представление той или иной информации. То есть при вставке в представление проверяется, есть ли такие сериалы или нет, если нету, то добавляется.

4.2. Файл для удаления

CREATE OR REPLACE TRIGGER delete\_stat

INSTEAD OF DELETE ON serial\_stat

BEGIN

DELETE FROM series

WHERE serial\_id = :old.serial\_id;

END;

1. **Разработать один или несколько SQL-сценариев для составления отчётов. В том случае, если используется клиент SQL\*Plus, необходимо изучить его параметры командной строки, а также ряд инструкций SQL\*Plus, в первую очередь set spool; set markup html; set feedback; set termoutput; set serveroutput; set echo. Также, могут быть полезны функции PL/SQL пакета DBMS OUTPUT;**

set spool on;

set sqlformat csv

spool 'C:\Users\HP\file1\_4.html'

select \* from serials\_actors;

spool off;

Результатом данного запроса будет запись ответа в выбранный html файл по директории. Для того, чтобы в файле отчет был не закодирован был использован формат csv.

**Заключение**

В данной были разработаны:

SQL-запросы с использованием разных теоретико-множественных операций: UNION, INTERSECT, MINUS;

Представления, основанные на списке запросов, реализованных ранее в ходе работ 1-1 и 1-2;

Создание дополнительных индексов на основе B-дерева;

Операции добавления/удаления строк из разработанных представлений.

Ссылка на результаты запросов:

https://github.com/Nurmatov-07/DataBase2020/tree/master/Lab1\_4