«Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем

Кафедра №42 «Криптология и кибербезопасность»

Отчёт по лабораторной работе № 2\_3

"Безопасность базы данных"

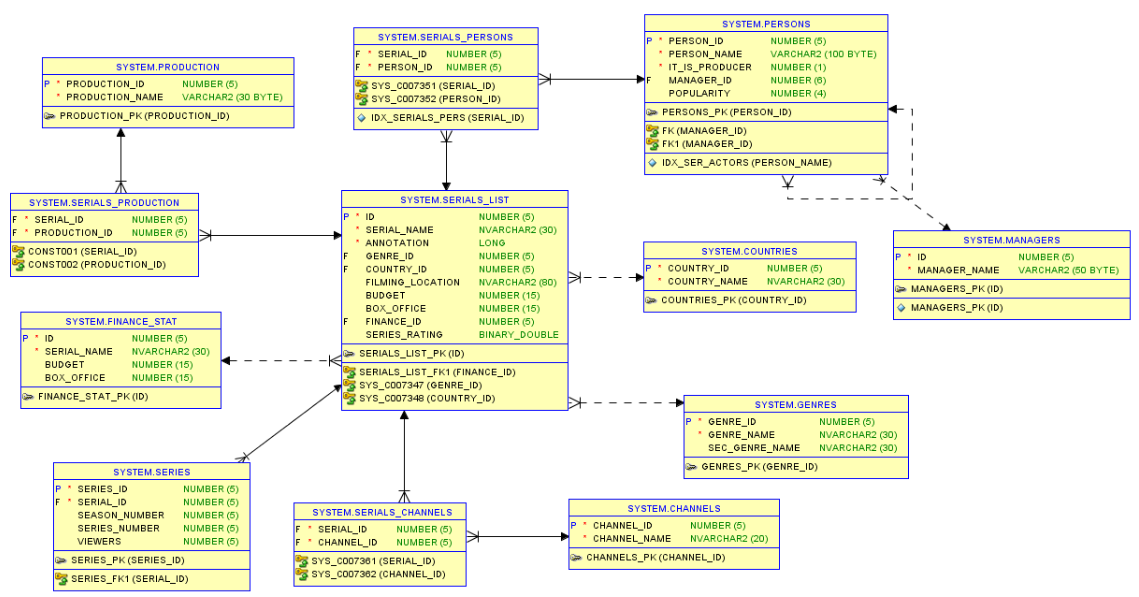
Студент: Нурматов С.Р.

Группа: Б17-565

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

Москва 2020

**Диаграмма сущностей**

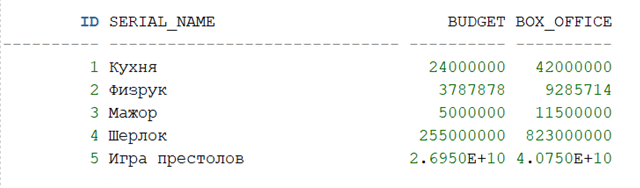
****

**0. Продемонстрировать обновление таблицы посредством операции слияния (инструкция MERGE). Таблицы для слияния должны быть выбраны таким образом, чтобы продемонстрировать и добавление, и обновление рядов базовой таблицы. В случае, если в схеме данных нет подходящей таблицы, необходимо создать новую таблицу специально для этой цели;**

Для выполнения этого задания была создана таблица finance\_stat, чтобы продемонстрировать назначение инструкции MERGE.

Назначение инструкции MERGE — обновить таблицу A при помощи новых сведений из таблицы B, где таблица А – finance\_stat, в которой хранятся промежуточные результаты бюджета (budget) и кассовых сборов (box\_office) сериалов, таблица B – serials\_list, в которой хранятся обновленные значения, это может означать, что вышла новая серия или новый сезон того или иного сериала или сериал завершился. Например, сериал «Острые козырьки» не имел данных о сборах и бюджете, как показано в таблице finance\_stat.

select \* from finance\_stat



После выхода нового сезона сериала, по достоверной информации MASS MEDIA, стали известны бюджет и кассовые сборы сериала. Данные были занесены в таблицу serials\_list. После чего было выполнено слияние таблиц finance\_stat и serials\_list, чтобы обновить таблицу «А». Данный запрос так же может быть полезен в сфере финансовой аналитики, каждый год компании делают ревизию за год, чтобы проанализировать продуктивность, затраты и доходы и на основе этого составить план на будущее. Кроме того, ревизия выполняется каждый месяц, чтобы иметь промежуточные результаты.

Суть в том, что промежуточная таблица «B» в базе данных хранит информацию, которая набиралась в течении месяца, после чего данные обновляют в таблице «А» в течении года каждый месяц.

MERGE INTO finance\_stat D

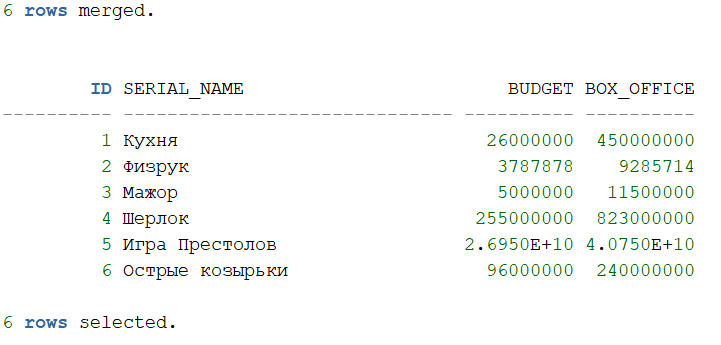
USING (SELECT id, serial\_name, budget, box\_office FROM serials\_list) S

ON (D.id = S.id)

WHEN MATCHED THEN UPDATE SET D.serial\_name = S.serial\_name, D.budget = S.budget, D.box\_office = S.box\_office

WHEN NOT MATCHED THEN INSERT (D.id, D.serial\_name, D.budget, D.box\_office) VALUES(S.id, S.serial\_name, S.budget, S.box\_office)

WHERE (S.id = 6);



Как уже было сказано раннее, в данном примере сериал «Острые козырьки» не был законченным, поэтому в таблице Finance\_stat нету информации о его бюджете и кассовых сборах. А в таблице serials\_list хранится свежая информация о бюджете и сборах, в результате чего, после завершения сериала, информация обновляется в таблице finance\_stat посредством слияния.

**1. Спроектировать и составить отчёт по разработанной базе данных с промежуточными итогами на основе расширенной группировки GROUP BY ROLLUP;**

**ROLLUP** – оператор Transact-SQL, который формирует промежуточные итоги для каждого указанного элемента и общий итог.

* 1. Для демонстрации оператора ROLLUP была выполнена сортировка по сериалам, где абстрактно показаны промежуточные итоги по количеству зрителей по сериалам (а также по сезонам).

SELECT serial\_name,

NVL(CAST(season\_number as VARCHAR2(30)),

CASE WHEN GROUPING(season\_number) = 1 AND GROUPING(serial\_name) = 0

THEN 'Subtotal'

ELSE 'GrandTotal' END) AS season\_number,

SUM(viewers) as ITOG,

GROUPING(serial\_name) AS grouping\_serial\_name,

GROUPING(season\_number) AS grouping\_season\_name

FROM serials\_list

JOIN series

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

GROUP BY

ROLLUP (serial\_name, season\_number);

Данный запрос позволяет провезти анализ по количеству зрителей и сделать выводы о том, какой сериал более просматриваемый. Так же можно увидеть количество зрителей по сезонам, что не мало важно для производителей и аналитиков. Но чтобы показать подробную информацию о количестве зрителей, по сезонам, можно выполнить следующий запрос.

Результаты запроса сохранены в файл «Res\_query 1.1».

* 1. Данный запрос демонстрирует номер сезона сериала «Кухня», номер серии каждого сезона, а также промежуточный итог по количеству зрителей каждого сезона и общее количество зрителей сериала.

SELECT serial\_name, season\_number, series\_number, SUM(viewers) as itog

FROM serials\_list

JOIN series

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

where id = 2

GROUP BY

ROLLUP (serial\_name, season\_number, series\_number);

Запрос можно считать логическим продолжением предыдущего запроса.

Есть много критериев, по которым можно определить популярность сериала или какой сезон считается лучшим. Один из них количество зрителей или просмотров.

Результаты запроса сохранены в файл «Res\_query 1.2».

**2. Спроектировать и составить отчёт по разработанной базе данных на основе расширенной группировки «гиперкуб» (GROUP BY CUBE) или GROUPING SETS. На примере этого отчёта показать, как работают данные режимы группировки;**

**GROUPING SETS** – оператор Transact-SQL, который формирует результаты нескольких группировок в один набор данных

Для демонстрации работы режима группировки GROUPING SETS была создана таблица PRODUCTION и SERIALS\_PRODUCTION, чтобы наглядно показать результаты группировки по кинокомпаниям. Так же можно было добавить и группировку по странам, так как может быть несколько сериалов из одной страны.

SELECT production\_name, SUM(box\_office - budget) as net\_income

FROM serials\_list

JOIN serials\_production

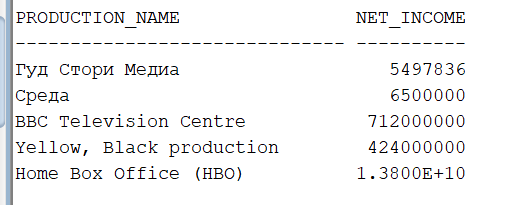
ON serials\_list.id = serials\_production.serial\_id

JOIN production

ON serials\_production.production\_id = production.production\_id

GROUP BY

GROUPING SETS (production\_name);



Данный запрос показывает объем прибыли по кинокомпаниям. Кроме того, объем прибыли говорит от том, что сериалы кинокомпаний пользуются огромной популярностью и обладают высоким рейтингом.

**3. Спроектировать и составить матричный отчёт по разрабатываемой базе данных с применением оператора PIVOT. В случае, если в базе данных нет подходящей таблицы, можно создать её специально. Также допустимо реализовать вместо этого запрос на базе оператора UNPIVOT. На примере этого отчёта (запроса) показать, как работает этот оператор;**

**PIVOT** – это оператор Transact-SQL, который поворачивает результирующий набор данных, т.е. происходит транспонирование таблицы, и данные соответственно группируются. Другими словами, значения, которые расположены по вертикали, выстраиваются по горизонтали

Для составления матричного отчета были использованы поля serial\_name из таблицы serials\_list, season\_number и series\_number из таблицы series, где результатом запроса является таблица со столбцами «SEASON\_N», где N номер сезона, а по строкам –– названия сериалов.

select \* from

(

select a.serial\_name, b.season\_number, b.series\_number

FROM series b

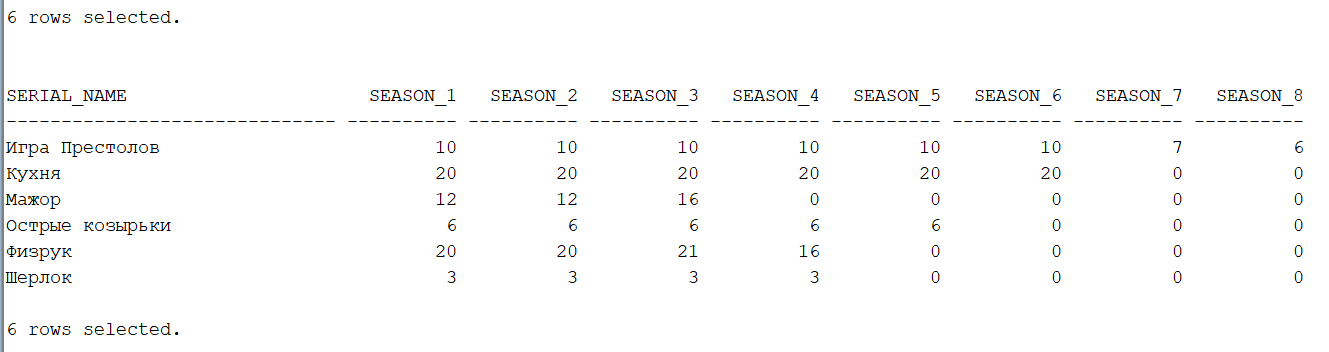
INNER join serials\_list a

on b.serial\_id = a.id

)

PIVOT (count(series\_number) FOR season\_number IN (1 as season\_1, 2 as season\_2, 3 as season\_3, 4 as season\_4, 5 as season\_5, 6 as season\_6, 7 as season\_7, 8 as season\_8))

order by serial\_name;

****

Данную таблицу можно назвать универсальной, так как она наглядно показывает сколько серий в каждом сезоне. Как уже многим известно, большинство людей прежде чем смотреть сериал смотрят не только на рейтинг, страну выпуска или кинокомпанию, но и на количество сезонов, серий и продолжительность (20 или 40-20 минут). Многие просто не в состоянии просмотреть сериал, который имеет свыше 5 – 8 сезонов, такие как Ходячие Мертвецы, Игра Престолов или Сверхестественное.

**4. Выделить в базе данных таблицы фактов (FACT TABLE) и таблицы измерений (DIMENSION TABLE). Обосновать этот выбор. В случае, если в базе данных нет выраженной таблицы фактов, допускается использовать похожую таблицу или создать таблицу фактов специально. На основе этих таблиц создать измерения (CREATE DIMENSION) и иерархии. Использовать их для создания аналитического представления. Убедиться в работоспособности аналитического представления при помощи нескольких запросов.**

В базе данных была выделена таблица фактов. За такую таблицу была принята таблица serials\_list так она является центральной и связывает почти все остальные таблицы.

Для создания аналитического в первую очередь было создано атрибутное измерение (ATTRIBUTE DIMENSION).

CREATE OR REPLACE ATTRIBUTE DIMENSION serials\_attr\_dim

USING serials\_List

ATTRIBUTES

(

id,

serial\_name,

budget,

box\_office,

series\_rating

)

LEVEL serials\_list

KEY id

ORDER BY id

ALL MEMBER NAME 'ALL SERIALS';

Затем была создана иерархия для представления

CREATE OR REPLACE HIERARCHY serials\_hierar

USING serials\_attr\_dim

(serials\_list);

После создания измерения, было создано аналитическое представление.

CREATE OR REPLACE ANALYTIC VIEW names\_anl\_views

CLASSIFICATION caption VALUE 'names\_anl\_views'

CLASSIFICATION description VALUE 'names\_anl\_views'

USING finance\_stat

dimension by

(serials\_attr\_dim

KEY id references id

HIERARCHIES(

serials\_hierar)

)

measures

(

box\_office fact box\_office

)

default measure box\_office;

**Заклюние**

В данной лабораторной работе, были получены навыки работы

1. С инструкцией MERGE для слияния таблиц
2. С расширенными группировками GROUP BY ROLLUP, CUBE, GROUPING SETS
3. С инструкцией PIVOT для формирования матричных отчетов
4. Создания измерений, иерархий и аналитических представлений.

Результаты запросов и информация об изменении таблиц хранится на GitHub.

Ссылка: <https://github.com/Nurmatov-07/DataBase2020>