Санкт-Петербургский государственный политехнический
университет Петра Великого
Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе
Дисциплина: Базы данных
Тема: Ознакомление с основами SQL-DML

Выполнил
студент гр. 43501/3 _____ Д. А. Зобков
(подпись)

Преподаватель _____ А. В. Мяснов
(подпись)

“___”_____ 2016 г.

Санкт-Петербург
2016 г.

1 Цель работы

Познакомиться с языком создания запросов управления данными SQL-DML.

2 Программа работы

1. Изучить SQL-DML;
2. Выполнить все запросы из списка стандартных запросов. Продемонстрировать результаты преподавателю;
3. Получить у преподавателя и реализовать SQL-запросы в соответствии с индивидуальным заданием. Продемонстрировать результаты преподавателю;
4. Выполненные запросы SELECT сохранить в БД в виде представлений, запросы INSERT, UPDATE или DELETE — в виде ХП.

3 Список стандартных запросов

1. Сделать выборку всех данных из каждой таблицы;
2. Сделать выборку данных из одной таблицы при нескольких условиях, с использованием логических операций LIKE, BETWEEN, IN (не менее 3-х разных примеров);
3. Создать в запросе вычисляемое поле;
4. Сделать выборку всех данных с сортировкой по нескольким полям;
5. Создать запрос, вычисляющий несколько совокупных характеристик таблиц;
6. Сделать выборку данных из связанных таблиц (не менее двух примеров);
7. Создать запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки;
8. Придумать и реализовать пример использования вложенного запроса;
9. С помощью оператора INSERT добавить в каждую таблицу по одной записи;
10. С помощью оператора UPDATE изменить значения нескольких полей у всех записей, отвечающих заданному условию;
11. С помощью оператора DELETE удалить запись, имеющую максимальное (минимальное) значение некоторой совокупной характеристики;
12. С помощью оператора DELETE удалить записи в главной таблице, на которые не ссылается подчиненная таблица (используя вложенный запрос).

4 Ход работы

По итогам предыдущей работы была разработана модифицированная SQL-схема БД для музыкальной библиотеки (рис. 4.1 на следующей странице).

Будем использовать данную БД для выполнения работы. В процессе будем создавать представления и ХП в соответствии с заданием, за исключением тех случаев, где в этом нет необходимости.

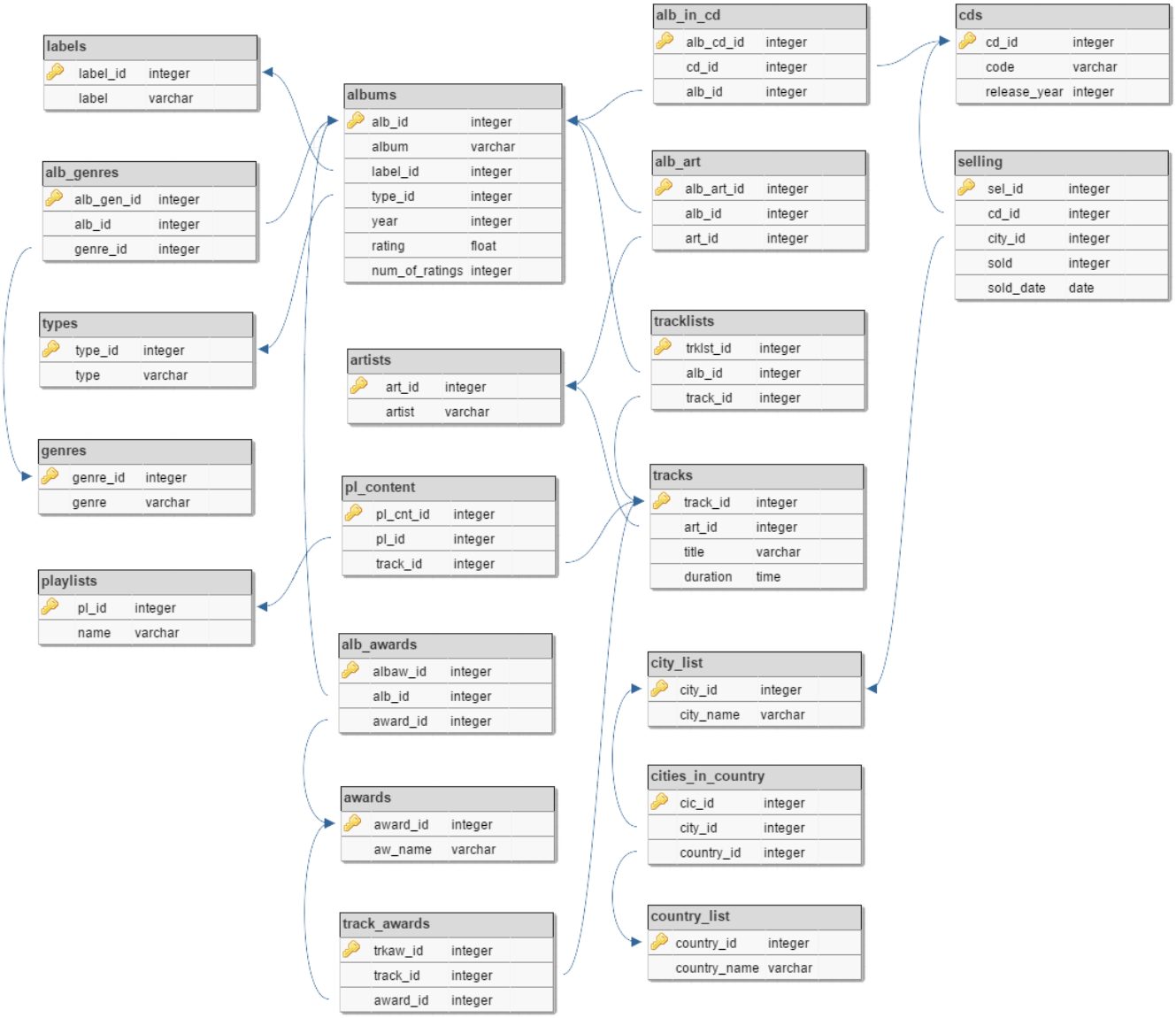

Рис. 4.1. SQL-схема БД

4.1 Стандартные запросы

4.1.1 Сделать выборку всех данных из каждой таблицы

```
SELECT * FROM ALBUMS;
SELECT * FROM ALB_ART;
SELECT * FROM ALB_AWARDS;
SELECT * FROM ALB_GENRES;
SELECT * FROM ALB_IN_CD;
SELECT * FROM ARTISTS;
SELECT * FROM AWARDS;
SELECT * FROM CDS;
SELECT * FROM CITIES_IN_COUNTRY;
SELECT * FROM CITY_LIST;
SELECT * FROM COUNTRY_LIST;
SELECT * FROM GENRES;
SELECT * FROM LABELS;
SELECT * FROM PLAYLISTS;
```

```
SELECT * FROM PL_CONTENT;
SELECT * FROM SELLING;
SELECT * FROM TRACKLISTS;
SELECT * FROM TRACKS;
SELECT * FROM TRACK_AWARDS;
SELECT * FROM TYPES;
```

4.1.2 Сделать выборку данных из одной таблицы при нескольких условиях, с использованием логических операций LIKE, BETWEEN, IN (не менее 3-х разных примеров)

```
-- Вывод альбомов, начинающихся с "J-N"
CREATE OR ALTER VIEW t2_1 AS
SELECT * FROM ALBUMS WHERE ALBUM LIKE 'J-N%';
-- Вывод альбомов, выпущенных в период 2007-2009 годов
CREATE OR ALTER VIEW t2_2 AS
SELECT * FROM ALBUMS WHERE "YEAR" BETWEEN 2007 AND 2009;
-- Вывод альбомов, в которых 50, 100, 150 проголосовавших для рейтинга
CREATE OR ALTER VIEW t2_3 AS
SELECT * FROM ALBUMS WHERE NUM_OF_RATINGS IN (50,100,150);
-- Завершение транзакции
COMMIT;
```

4.1.3 Создать в запросе вычисляемое поле

```
-- Вычисляется дополнительное поле длительности песни в NightCore жанре
-- (ускорение на 30%)
-- С помощью DATEDIFF() получаем количество секунд,
-- выполняем умножение и вычитаем из начальной длины
CREATE OR ALTER VIEW t3 AS
SELECT TITLE, DURATION,
    (DURATION - (DATEDIFF(SECOND FROM TIME '00:00' TO DURATION)*0.3))
   AS NIGHTCORE_DURATION
        FROM TRACKS;
-- Завершение транзакции
COMMIT;
```

4.1.4 Сделать выборку всех данных с сортировкой по нескольким полям

```
-- Сортировка таблицы ALBUMS по годам по возрастанию
-- и по кол-ву проголосовавших по убыванию
CREATE OR ALTER VIEW t4 AS
SELECT * FROM ALBUMS ORDER BY "YEAR" ASC, NUM_OF_RATINGS DESC;
-- Завершение транзакции
COMMIT;
```

4.1.5 Создать запрос, вычисляющий несколько совокупных характеристик таблиц

```
-- Вывод минимального, среднего и максимального значения времени треков
-- CAST() преобразует строку "00:00" в тип TIME
CREATE OR ALTER VIEW t5 ("MIN", "AVG", "MAX") AS
SELECT MIN(DURATION),
    CAST('00:00' AS TIME) + AVG(DATEDIFF(SECOND FROM TIME '00:00' TO
    DURATION)),
    MAX(DURATION) FROM TRACKS;
-- Завершение транзакции
COMMIT;
```

4.1.6 Сделать выборку данных из связанных таблиц (не менее двух примеров)

```
-- Объединение таблиц TRACKS и ARTISTS
CREATE OR ALTER VIEW t6_1(TRACK_ID, ART_ID, TITLE, DURATION, ART_ID1,
    ARTIST) AS
SELECT * FROM TRACKS
    JOIN ARTISTS ON TRACKS.ART_ID = ARTISTS.ART_ID
    ORDER BY TRACK_ID ASC;
-- Подстановка реальных данных вместо внешних ключей в таблицу
    TRACKLISTS
CREATE OR ALTER VIEW t6_2 AS
SELECT TRACKLISTS.TRKLST_ID, ALBUMS.ALBUM, TRACKS.TITLE FROM TRACKLISTS
    JOIN ALBUMS ON ALBUMS.ALB_ID = TRACKLISTS.ALB_ID
    JOIN TRACKS ON TRACKS.TRACK_ID = TRACKLISTS.TRACK_ID;
-- Завершение транзакции
COMMIT;
```

4.1.7 Создать запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки

```
-- Выборка артистов, которым принадлежит менее 4300 треков
CREATE OR ALTER VIEW t7(ARTIST, "COUNT") AS
SELECT  ARTISTS.ARTIST, COUNT(TRACKS.ART_ID) FROM TRACKS
    JOIN ARTISTS ON ARTISTS.ART_ID = TRACKS.ART_ID
    GROUP BY ARTISTS.ARTIST HAVING COUNT(TRACKS.ART_ID)<4300;
-- Завершение транзакции
COMMIT;
```

4.1.8 Придумать и реализовать пример использования вложенного запроса

```
-- Вывести названия треков из альбомов с названиями,
-- начинающимися с "J-N"
CREATE OR ALTER VIEW t8 AS
SELECT TRACK_ID, TITLE FROM TRACKS WHERE TRACK_ID IN
```

```
    (SELECT TRACK_ID FROM TRACKLISTS WHERE ALB_ID IN
        (SELECT ALB_ID FROM ALBUMS WHERE ALBUM LIKE 'J-N%'));
-- Завершение транзакции
COMMIT;
```

4.1.9 С помощью оператора INSERT добавить в каждую таблицу по одной записи

```
-- Добавление записи в каждую таблицу
CREATE OR ALTER PROCEDURE t9 AS
BEGIN
    INSERT INTO CDS VALUES (7, 'TESTCD', 2016);
    INSERT INTO LABELS VALUES (4, 'TESTLABEL');
    INSERT INTO TYPES VALUES (9, 'TESTTYPE');
    INSERT INTO GENRES VALUES (7, 'TESTGENRE');
    INSERT INTO ARTISTS VALUES (24, 'TESTARTIST');
    INSERT INTO TRACKS VALUES (100074, 24, 'TESTTITLE', '00:03:00');
    INSERT INTO PLAYLISTS VALUES (2, 'TESTPLAYLIST');
    INSERT INTO PL_CONTENT VALUES (24, 2, 100074);
    INSERT INTO ALBUMS VALUES (100008, 'TESTALBUM', 4, 9, 2016, 5, 1);
    INSERT INTO ALB_GENRES VALUES (10, 100008, 7);
    INSERT INTO ALB_IN_CD VALUES (7, 7, 100008);
    INSERT INTO ALB_ART VALUES (8, 100008, 24);
    INSERT INTO TRACKLISTS VALUES (100074, 100008, 100074);
    INSERT INTO AWARDS VALUES (1, 'TESTAWARD');
    INSERT INTO TRACK_AWARDS VALUES (1, 100074, 1);
    INSERT INTO ALB_AWARDS VALUES (1, 100008, 1);
    INSERT INTO CITY_LIST VALUES (1, 'SPB');
    INSERT INTO COUNTRY_LIST VALUES (1, 'RUSSIA');
    INSERT INTO CITIES_IN_COUNTRY VALUES (1, 1, 1);
    INSERT INTO SELLING VALUES (1, 7, 1, 1000, 'NOW');
END
-- Завершение транзакции
COMMIT;
```

4.1.10 С помощью оператора UPDATE изменить значения нескольких полей у всех записей, отвечающих заданному условию

```
-- Инкремент количества проголосовавших
CREATE OR ALTER PROCEDURE t10 AS
BEGIN
    UPDATE ALBUMS SET NUM_OF_RATINGS=NUM_OF_RATINGS+1;
END
-- Завершение транзакции
COMMIT;
```

4.1.11 С помощью оператора DELETE удалить запись, имеющую максимальное (минимальное) значение некоторой совокупной характеристики

```
-- Создание трека с самым большим временем и его удаление
CREATE OR ALTER PROCEDURE t11 AS
BEGIN
    INSERT INTO TRACKS VALUES ((SELECT MAX(TRACK_ID) FROM TRACKS)+1, 1,
    'TEST', '23:59:59');
    DELETE FROM TRACKS WHERE DURATION=(SELECT MAX(DURATION) FROM TRACKS
   );
END
-- Завершение транзакции
COMMIT;
```

4.1.12 С помощью оператора DELETE удалить записи в главной таблице, на которые не ссылается подчиненная таблица (используя вложенный запрос)

```
-- Удаление записей, на которые не ссылаются подчиненные таблицы
CREATE OR ALTER PROCEDURE t12 AS
BEGIN
    INSERT INTO TRACKS VALUES ((SELECT MAX(TRACK_ID) FROM TRACKS)+1, 1,
    'TEST', '23:59:59');
    DELETE FROM TRACKS WHERE TRACK_ID IN (
        SELECT TRACK_ID FROM TRACKS
            WHERE TRACK_ID NOT IN(SELECT TRACK_ID FROM TRACKLISTS)
            AND TRACK_ID NOT IN(SELECT TRACK_ID FROM PL_CONTENT)
            AND TRACK_ID NOT IN(SELECT TRACK_ID FROM TRACK_AWARDS));
END
-- Завершение транзакции
COMMIT;
```

4.2 Индивидуальное задание

4.2.1 Вывести 5 лейблов, треки, с альбомов которых, наиболее часто встречаются в плейлистах

```
CREATE OR ALTER VIEW ind1(LABEL, "COUNT") AS
SELECT FIRST 5 LABELS.LABEL, COUNT(LABELS.LABEL_ID) FROM PL_CONTENT
    JOIN TRACKLISTS ON TRACKLISTS.TRACK_ID = PL_CONTENT.TRACK_ID
    JOIN ALBUMS ON TRACKLISTS.ALB_ID = ALBUMS.ALB_ID
    JOIN LABELS ON LABELS.LABEL_ID = ALBUMS.LABEL_ID
    GROUP BY LABELS.LABEL
    ORDER BY COUNT(LABELS.LABEL) DESC;
-- Завершение транзакции
COMMIT;
```

4.2.2 Вывести треки, которые содержатся в альбомах, записанных на заданных лейблах или относящихся к заданным жанрам

```
CREATE OR ALTER PROCEDURE ind2(
    GENRE VARCHAR(40),
    LABEL VARCHAR(100)
)
RETURNS(
    ID INTEGER,
    TITLE VARCHAR(100)
)
AS
BEGIN
    FOR SELECT TRACKLISTS.TRACK_ID, TRACKS.TITLE FROM TRACKLISTS
        JOIN TRACKS ON TRACKS.TRACK_ID = TRACKLISTS.TRACK_ID
        WHERE ALB_ID IN (
            SELECT ALB_ID FROM ALB_GENRES WHERE GENRE_ID IN (
                SELECT GENRE_ID FROM GENRES WHERE GENRE IN (:GENRE)))
   OR
        ALB_ID IN (
            SELECT ALB_ID FROM ALBUMS WHERE LABEL_ID IN (
                SELECT LABEL_ID FROM LABELS WHERE LABEL IN (:LABEL)))
        GROUP BY TRACKLISTS.TRACK_ID, TRACKS.TITLE
        ORDER BY TRACKLISTS.TRACK_ID ASC
        INTO :ID, :TITLE
    DO
    BEGIN
        SUSPEND;
    END
END
-- Завершение транзакции
COMMIT;
```

4.2.3 Для каждого жанра вывести отношение количества появления треков в плейлистах к количеству альбомов, на которых содержатся треки данного жанра

```
CREATE OR ALTER VIEW ind3 AS
SELECT GENRES.GENRE, (CAST(TRK AS FLOAT)/NULLIF(CAST(ALB AS FLOAT), 0))
    AS RATIO FROM GENRES
JOIN (
    SELECT ALB_GENRES.GENRE_ID, COUNT(PL_CONTENT.TRACK_ID) AS TRK FROM
   PL_CONTENT
        JOIN TRACKLISTS ON TRACKLISTS.TRACK_ID = PL_CONTENT.TRACK_ID
        RIGHT JOIN ALB_GENRES ON TRACKLISTS.ALB_ID = ALB_GENRES.ALB_ID
        GROUP BY ALB_GENRES.GENRE_ID)
    NUM ON NUM.GENRE_ID = GENRES.GENRE_ID
JOIN (
    SELECT GENRE_ID, COUNT(ALB_ID) AS ALB FROM ALB_GENRES
        GROUP BY GENRE_ID)
    DENOM ON DENOM.GENRE_ID = GENRES.GENRE_ID
ORDER BY GENRES.GENRE_ID ASC;
-- Завершение транзакции
COMMIT;
```

5 Выводы

В ходе выполнения данной работы было проведено ознакомление с языком SQL-DML, позволяющий производить операции над данными в БД. К основным операциям SQL-DML можно отнести:

- INSERT — добавление записи;
- SELECT — выборка данных;
- DELETE — удаление записей;
- UPDATE — обновление значений записей;
- JOIN — объединение таблиц для выборки;
- WHERE — условия отбора в выборку;
- ORDER BY — группировка выбоки.

Использование данных операторов позволяет производить широкий круг действий над БД, таких как выборка данных, объединение таблиц, вложенные запросы, наложение условий для выборки и т.д.

Использование представлений позволяет сохранить частые запросы в БД как виртуальные таблицы для более простого их вызова. Хранимые процедуры являются аналогом функций в языках программирования и позволяют сохранить часто используемые однотипные операции сложной выборки с использованием переменных в БД. Данные инструменты упрощают выполнение рутинных действий с базой данных.