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

Отчёт по лабораторной работе

Дисциплина: Базы данных

Тема: Язык SQL-DML

Выполнил студент гр. 43501/1 _____ Г. А. Жемелев
(подпись)

Преподаватель _____ А. В. Мяснов
(подпись)

“11” декабря 2016 г.

Санкт-Петербург
2016

Цель работы

Познакомиться с языком создания запросов управления данными SQL-DML.

Программа работы

1. Изучить SQL-DML.
2. Выполнить все запросы из списка стандартных запросов и 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 хранимых процедур.

Ход работы

Список стандартных запросов:

- Сделайте выборку всех данных из каждой таблицы.
- Сделайте выборку данных из одной таблицы при нескольких условиях, с использованием логических операций, LIKE, BETWEEN, IN (не менее 3-х разных примеров).
- Создайте в запросе вычисляемое поле.
- Сделайте выборку всех данных с сортировкой по нескольким полям.
- Создайте запрос, вычисляющий несколько совокупных характеристик таблиц.
- Сделайте выборку данных из связанных таблиц (не менее двух примеров).
- Создайте запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки.
- Придумайте и реализуйте пример использования вложенного запроса.
- С помощью оператора INSERT добавьте в каждую таблицу по одной

записи.

- С помощью оператора UPDATE измените значения нескольких полей у всех записей, отвечающих заданному условию.
- С помощью оператора DELETE удалите запись, имеющую максимальное (минимальное) значение некоторой совокупной характеристики.
- С помощью оператора DELETE удалите записи в главной таблице, на которые не ссылается подчиненная таблица (используя вложенный запрос).

Для выполнения запросов SELECT из заданного списка запросов был написан скрипт, приведенный далее. Все запросы сразу оформлены в виде представлений.

```
CREATE VIEW all_arrivals AS     SELECT * FROM arrivals;
CREATE VIEW all_cat AS          SELECT * FROM categories;
CREATE VIEW all_cat_arr AS      SELECT * FROM cat_arrangement;
CREATE VIEW all_man AS          SELECT * FROM manufacturers;
CREATE VIEW all_od AS           SELECT * FROM orders_distribution;
CREATE VIEW all_specs AS        SELECT * FROM specifications;
CREATE VIEW all_storages AS     SELECT * FROM storages;
CREATE VIEW all_avail AS        SELECT * FROM availability;
CREATE VIEW all_csd AS          SELECT * FROM catspec_distribution;
CREATE VIEW all_goods AS        SELECT * FROM goods;
CREATE VIEW all_orders AS       SELECT * FROM orders;
CREATE VIEW all_reviews AS      SELECT * FROM reviews;
CREATE VIEW all_specarr AS      SELECT * FROM spec_arrangement;

CREATE VIEW goodsGTX AS
    SELECT * FROM goods WHERE Name LIKE '%GTX%';
CREATE VIEW mans56 AS
    SELECT * FROM goods WHERE man_ID IN (5, 6);
CREATE VIEW price1020k AS
    SELECT * FROM goods WHERE Price BETWEEN 10000.00 AND 20000.00;

CREATE VIEW totalAsusCost AS
SELECT SUM(Price * Quantity) AS "Total cost of goods by ASUS"
FROM goods AS g NATURAL JOIN availability JOIN manufacturers AS m ON
m.ID = g.man_ID WHERE m.ID = (SELECT ID from manufacturers WHERE Name LIKE 'Asus');

CREATE VIEW sortGoodsExample AS
SELECT * FROM goods ORDER BY man_ID ASC, Name ASC;

CREATE VIEW avgsumExample AS
SELECT AVG(Quantity) AS "Average quantity in arrival",
SUM(Quantity) AS "Total items arrived" FROM arrivals;

-- Вывести все категории с их характеристиками
CREATE VIEW catspecHumanReadable AS
SELECT c.Name AS "Category", s.Name AS "Specification"
```

```
FROM categories c JOIN catspec_distribution csd ON c.ID = csd.catID
    JOIN specifications s ON csd.specID = s.ID;
-- Вывести названия товаров вместе с названиями их производителей
CREATE VIEW manGoods AS
SELECT m.Name AS "Manufacturer", g.Name AS "Item"
FROM goods g JOIN manufacturers m ON g.man_ID = m.ID;

-- Вывести количество товаров, поставляемых каждым производителем
CREATE VIEW manItemsCount AS
SELECT m.Name AS "Manufacturer", COUNT(*) AS "Items count"
FROM goods g JOIN manufacturers m ON g.man_ID = m.ID GROUP BY m.Name;
/* Вывести количество товаров от каждого производителя, поставляющего
   больше одного наименования */
CREATE VIEW manItemsCountMore1 AS
SELECT m.Name AS "Manufacturer", COUNT(*) AS "Items count"
FROM goods g JOIN manufacturers m ON g.man_ID = m.ID GROUP BY m.Name
HAVING COUNT(*) > 1;

-- Узнать, что купил клиент с номером телефона +79119786196
CREATE VIEW boughtByPhone AS
SELECT Name FROM goods WHERE Article = (
    SELECT article FROM orders NATURAL JOIN orders_distribution
    WHERE Phone LIKE '%9119786196'
);
```

Запросы, модифицирующие данные в таблицах:

```
-- Добавление по одной записи в каждую таблицу
INSERT INTO manufacturers
VALUES (NEXT VALUE FOR Seq_manID, 'Thermaltake');
INSERT INTO storages
VALUES (NEXT VALUE FOR Seq_stoID, 'Москва, ул. Строителей, д. 25');
INSERT INTO goods VALUES
(NEXT VALUE FOR Seq_Article, 'Core V1', 3500.00, (SELECT ID FROM manufacturers WHERE
Name = 'Thermaltake'));
INSERT INTO arrivals VALUES
(NEXT VALUE FOR Seq_arrID, '01.12.2016', 13, 3, 32);
INSERT INTO orders VALUES
(NEXT VALUE FOR Seq_OrderNo, '01.12.2016', '+79119786196', 0);
INSERT INTO orders_distribution VALUES
(2, 13, 1, 3500.00);
INSERT INTO reviews VALUES
(NEXT VALUE FOR Seq_revID, 8, '11.11.2016 14:00:19', 1, 'Плохая вебка.');
INSERT INTO categories VALUES
(NEXT VALUE FOR Seq_catID, 'Корпуса');
INSERT INTO specifications VALUES
(NEXT VALUE FOR Seq_specID, 'Число слотов PCI');
INSERT INTO catspec_distribution VALUES (4, 6);
INSERT INTO cat_arrangement VALUES (13, 4);
INSERT INTO spec_arrangement VALUES (13, 6, '2');
COMMIT;
-- Изменение нескольких полей у всех записей, удовлетворяющих заданному условию
UPDATE goods SET Price = 0.8*Price, Name = Name || ' (Sale! -20%)'
WHERE man_ID = (SELECT ID FROM manufacturers WHERE Name = 'Intel');
-- Удаление записи, имеющей минимальное значение некоторой совокупной характеристики
DELETE FROM reviews ORDER BY Rating ASC ROWS 1;
-- Удаление записей в главное таблице, на которые не ссылается подчиненная
DELETE FROM manufacturers m WHERE NOT EXISTS
(SELECT man_ID FROM goods g WHERE m.ID = g.man_ID);
```

Реализация всех приведенных выше запросов в виде хранимых процедур:

```
SET TERM ^;
CREATE PROCEDURE insertEverywhere AS
    DECLARE VARIABLE newManID INTEGER;
    DECLARE VARIABLE newArticle INTEGER;
    DECLARE VARIABLE newStoID INTEGER;
    DECLARE VARIABLE newOrderNo INTEGER;
    DECLARE VARIABLE newCatID INTEGER;
    DECLARE VARIABLE newSpecID INTEGER;
BEGIN
    INSERT INTO manufacturers VALUES
        (NEXT VALUE FOR Seq_manID, 'Thermaltake')
        RETURNING ID INTO :newManID;
    INSERT INTO storages VALUES
        (NEXT VALUE FOR Seq_stoID, 'Москва, ул. Строителей, д. 25')
        RETURNING ID INTO :newStoID;
    INSERT INTO goods VALUES
        (NEXT VALUE FOR Seq_Article, 'Core V1', 3500.00, :newManID)
        RETURNING Article INTO :newArticle;
    INSERT INTO arrivals VALUES
        (NEXT VALUE FOR Seq_arrID, '01.12.2016', :newArticle, :newStoID, 32);
    INSERT INTO orders VALUES
        (NEXT VALUE FOR Seq_OrderNo, '01.12.2016', '+79119786196', 0)
        RETURNING Order_No INTO :newOrderNo;
    INSERT INTO orders_distribution VALUES
        (:newOrderNo, :newArticle, 1, 3500.00);
    INSERT INTO reviews VALUES
     (NEXT VALUE FOR Seq_revID, (SELECT Article FROM goods WHERE Name = 'Webcam C170'),
        '11.11.2016 14:00:19', 1, 'Плохая вебка.');
    INSERT INTO categories VALUES
        (NEXT VALUE FOR Seq_catID, 'Корпуса')
        RETURNING ID INTO :newCatID;
    INSERT INTO specifications VALUES
        (NEXT VALUE FOR Seq_specID, 'Число слотов PCI')
        RETURNING ID INTO :newSpecID;
    INSERT INTO catspec_distribution VALUES (:newCatID, :newSpecID);
    INSERT INTO cat_arrangement VALUES (:newArticle, :newCatID);
    INSERT INTO spec_arrangement VALUES (:newArticle, :newSpecID, '2');
END^

CREATE PROCEDURE startIntelSale AS
BEGIN
    UPDATE goods SET Price = 0.8*Price, Name = Name || ' (Sale! -20%)'
    WHERE man_ID = (SELECT ID FROM manufacturers WHERE Name = 'Intel');
END^

CREATE PROCEDURE deleteWorstReview AS
BEGIN
    DELETE FROM reviews ORDER BY Rating ASC ROWS 1;
END^

CREATE PROCEDURE deleteMansWithoutGoods AS
BEGIN
    DELETE FROM manufacturers m WHERE NOT EXISTS
    (SELECT man_ID FROM goods g WHERE m.ID = g.man_ID);
END^
SET TERM ;^
COMMIT;
```

После демонстрации результатов преподавателю было получено следующее индивидуальное задание:

1. Для заданного производителя вывести суммарные продажи (в деньгах) за заданный промежуток времени по каждой категории.
2. Вывести 5 заказов с максимальной суммарной стоимостью, в которых содержатся позиции из разных категорий.
3. Вывести товары, которые соответствуют набору заданных значений характеристик.

Для того чтобы можно было проиллюстрировать результаты выполнения заданных запросов, в базу данных были добавлены новые записи:

```
INSERT INTO categories VALUES
(NEXT VALUE FOR Seq_catID, 'Видеокарты');
INSERT INTO cat_arrangement VALUES (1, 4);
INSERT INTO cat_arrangement VALUES (2, 4);
INSERT INTO cat_arrangement VALUES (3, 4);
INSERT INTO cat_arrangement VALUES (4, 4);

INSERT INTO categories VALUES
(NEXT VALUE FOR Seq_catID, 'Ноутбуки');
INSERT INTO categories VALUES
(NEXT VALUE FOR Seq_catID, 'Материнские платы');
INSERT INTO categories VALUES
(NEXT VALUE FOR Seq_catID, 'Мониторы');

INSERT INTO goods VALUES
(NEXT VALUE FOR Seq_Article, 'Transformer Book Trio TX201LA', 31990.00, (SELECT ID FROM
manufacturers WHERE Name = 'Asus'));
INSERT INTO goods VALUES
(NEXT VALUE FOR Seq_Article, 'Zenbook UX303UA', 46036.00, (SELECT ID FROM manufacturers
WHERE Name = 'Asus'));
INSERT INTO goods VALUES
(NEXT VALUE FOR Seq_Article, 'K501UX', 50000.00, (SELECT ID FROM manufacturers WHERE
Name = 'Asus'));
INSERT INTO goods VALUES
(NEXT VALUE FOR Seq_Article, 'XPS 13 9360', 86440.00, (SELECT ID FROM manufacturers
WHERE Name = 'Dell'));
INSERT INTO goods VALUES
(NEXT VALUE FOR Seq_Article, 'VX279Q', 21500.00, (SELECT ID FROM manufacturers WHERE
Name = 'Asus'));
INSERT INTO goods VALUES
(NEXT VALUE FOR Seq_Article, 'VC239H', 11365.00, (SELECT ID FROM manufacturers WHERE
Name = 'Asus'));
INSERT INTO goods VALUES
(NEXT VALUE FOR Seq_Article, 'Z170 Pro Gaming', 13700.00, (SELECT ID FROM manufacturers
WHERE Name = 'Asus'));

INSERT INTO cat_arrangement VALUES (13, 5);
INSERT INTO cat_arrangement VALUES (14, 5);
INSERT INTO cat_arrangement VALUES (15, 5);
INSERT INTO cat_arrangement VALUES (16, 5);
INSERT INTO cat_arrangement VALUES (17, 7);
INSERT INTO cat_arrangement VALUES (18, 7);
INSERT INTO cat_arrangement VALUES (19, 6);
COMMIT;
```

Кроме того, в таблицы ORDERS и ORDERS_DISTRIBUTION было добавлено два десятка случайно сгенерированных записей с помощью IBExpert. В связи с этим цена в столбце ORDERS_DISTRIBUTION.Fixed_price большинства заказов оказалась не связана с содержащимися в этом заказе товарами, а содержимое столбца ORDERS.Fixed_cost не равно сумме цен входящих в заказ товаров. Для исправления ситуации все значения этого столбца были пересчитаны с помощью следующей процедуры:

```
CREATE PROCEDURE fillFP AS
    DECLARE VARIABLE ordNo INTEGER;
    DECLARE VARIABLE cost D_money;
BEGIN
    FOR SELECT Order_No, SUM(Fixed_price*Quantity) FROM orders_distribution
    GROUP BY Order_No
    INTO :ordNo, :cost
    DO
      UPDATE orders SET Fixed_cost = :cost
      WHERE Order_No = :ordNo;
END^
```

Окончательно, содержимое таблицы заказов приняло вид (рис. 1):

ORDER_NO	ORDER_DATE	PHONE	STATUS	FIXED_COST
1	2016-09-30	+79119786196	3	99439.00
2	2016-11-04	96644463569	3	13700.00
3	2016-10-25	81822446654	2	100000.00
4	2016-11-18	0035827243959	1	50500.00
5	2016-10-12	29640768061	4	178512.00
6	2016-11-07	38149531508	2	11365.00
9	2016-10-09	7302511865126	3	5340.00
10	2016-10-20	87525985522	4	76740.00
12	2016-12-02	5888368906157	3	0.00
13	2016-11-28	041198425569	4	136000.00
14	2016-12-05	5551432639779	2	0.00
15	2016-11-14	1093602865941	3	0.00
17	2016-10-05	058755223562	2	0.00
20	2016-10-09	8896021928870	3	86440.00
22	2016-11-23	57281780001	3	190880.00
23	2016-10-20	11712974702	3	76200.00
24	2016-11-20	144667467565	3	0.00
25	2016-11-01	59569127770	3	31500.00

Рис. 1. Содержимое таблицы заказов ORDERS

Заметим, что заказы с нулевой стоимостью возникли как следствие случайного выбора номеров заказов в таблице ORDERS_DISTRIBUTION (рис. 2).

ORDER_NO	ARTICLE	QUANTITY	FIXED_PRICE
1	1	1	12999.00
5	8	2	46036.00
25	5	2	5000.00
23	4	2	13700.00
1	19	1	86440.00
9	9	1	5340.00
3	9	2	50000.00
23	15	1	24400.00
4	11	2	18000.00
13	12	2	18000.00
22	8	2	86440.00
20	19	1	86440.00
13	11	2	50000.00
2	19	1	13700.00
6	10	1	11365.00
10	6	2	31990.00
22	18	1	18000.00
25	1	1	21500.00
23	3	1	24400.00
10	18	1	12760.00

ORDER_NO	ARTICLE	QUANTITY	FIXED_PRICE
5	1	1	86440.00
4	14	1	14500.00

Рис. 2. Содержимое таблицы ORDERS_DISTRIBUTION

Тогда реализация запросов согласно индивидуальному заданию может иметь следующий вид:

```
/* Для заданного производителя вывести суммарные продажи (в деньгах)
   за заданный промежуток времени по каждой категории. */

SELECT c.Name AS "Категория", SUM(od.Quantity * od.Fixed_price) AS "Продажи Asus"
FROM orders_distribution od NATURAL JOIN goods g
JOIN manufacturers m ON g.man_ID = m.ID
JOIN cat_arrangement ca ON ca.article = g.Article
JOIN categories c ON ca.catID = c.ID
WHERE od.order_No IN
    (SELECT Order_No FROM orders WHERE Order_Date BETWEEN '01.01.2016' AND '20.11.2016'
    AND Status = 3)
AND m.Name = 'Asus'
GROUP BY c.Name;

/* Вывести 5 заказов с максимальной суммарной стоимостью,
   в которых содержатся позиции из разных категорий. */

SELECT * FROM orders
WHERE Order_No IN
    (SELECT od.order_No FROM orders_distribution od NATURAL JOIN cat_arrangement ca
    GROUP BY od.order_No HAVING COUNT(ca.catID) > 1)
ORDER BY Fixed_cost DESC ROWS 5;
```

```
/* Вывести товары, которые соответствуют набору заданных значений характеристик. */

SELECT * FROM goods WHERE Article IN
(SELECT Article FROM goods NATURAL JOIN spec_arrangement sa JOIN specifications s ON
s.ID = sa.specID
WHERE (s.Name LIKE '%CAPACITY%' AND Spec_value LIKE '250%'))
AND Article IN
(SELECT Article FROM goods NATURAL JOIN spec_arrangement sa JOIN specifications s ON
s.ID = sa.specID
WHERE (s.Name LIKE '%IOPS%' AND Spec_value LIKE '89000'));
```

Результаты выполнения запросов приведены на рис. 3-5 соответственно:

Категория	Продажи Asus
Видеокарты	86299.00
Материнские платы	186580.00
Ноутбуки	24400.00

Рис. 3. Суммарные продажи товаров, произведенных Asus, по категориям за период с 01.01.2016 по 20.11.2016

ORDER_NO	ORDER_DATE	PHONE	STATUS	FIXED_COST
13	2016-11-28	041198425569	4	136000.00
3	2016-10-25	81822446654	2	100000.00
1	2016-09-30	+79119786196	3	99439.00
23	2016-10-20	11712974702	3	76200.00
4	2016-11-18	0035827243959	1	50500.00

Рис. 4. Пять заказов с максимальной суммарной стоимостью, в которых содержатся позиции из разных категорий

ARTICLE	NAME	PRICE	MAN_ID
11	850 EVO 250Gb MZ-N5E250BW	7920.00	5

Рис. 5. Товар, соответствующий набору заданных значений характеристик: объем = 250 Гб, число операций ввода-вывода в секунду = 89000.

В последнем случае (п. 3), если помимо характеристик необходимо выполнить проверку на принадлежность той или иной категории, то к запросу следует добавить условия вида:

```
AND Article IN
(SELECT Article FROM goods NATURAL JOIN cat_arrangement ca JOIN categories c ON
c.ID = ca.catID
WHERE c.Name = 'SSD');
```

Выводы

Язык SQL-DML предназначен для манипулирования данными: их добавления (INSERT), изменения (UPDATE), удаления (DELETE) и извлечения (SELECT). Это язык позволяет строить сложные запросы к БД, включающие множество условий, подзапросы, вычисление агрегатных функций и др.

Результаты выполнения оператора SELECT можно сохранять в виде представлений (VIEW) – виртуальных именованных таблиц. В отличие от обычных таблиц, представление не является самостоятельным набором данных, хранящимся в БД. Результат в виде набора данных динамически создается при обращении к представлению.

Операторы SQL-DML часто используются в хранимых процедурах. Это позволяет еще больше расширить возможности манипулирования данными, а также упростить описание запросов за счет использования локальных переменных, циклов, условных операторов и других возможностей PSQL.