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

Отчёт
По лабораторной работе №4
Дисциплина: базы данных
Тема: язык SQL-DML

Выполнил студент группы 43501/1: Евсеев Е.П.
Проверил преподаватель: Мяснов А.В.

Санкт-Петербург
2016

Цели работы

Познакомиться с языком создания запросов управления данными БД SQL-DML.

Программа работы

1. Изучить SQL-DML
2. Выполните все запросы из списка стандартных запросов
3. Реализовать SQL-запросы согласно индивидуальному заданию
4. Выполненные запросы SELECT сохраните в БД в виде представлений, запросы INSERT, UPDATE или DELETE - в виде ХП

Ход работы

Язык SQL (Structured Query Language) -- язык структурированных запросов. Он позволяет формировать весьма сложные запросы к базам данных. В SQL определены два подмножества языка:

- SQL-DDL (Data Definition Language) -- язык определения структур и ограничений целостности баз данных. Сюда относятся команды создания и удаления баз данных; создания, изменения и удаления таблиц; управления пользователями и т.д.
- SQL-DML (Data Manipulation Language) -- язык манипулирования данными: добавление, изменение, удаление и извлечение данных, управления транзакциями

Выполнение стандартных запросов

1. Сделайте выборку всех данных из каждой таблицы

```
select * from CARS;
select * from CLIENTS;
select * from DETAILS;
select * from DETAILS_CATALOG;
select * from MANUFACTURERS;
select * from ORDERS;
select * from ORDERS_SOA;
select * from PURCHASE;
select * from REPAIR;
select * from REPAIR_ORDERS;
select * from TYPES_OF_REPAIR;
select * from WARRANTYS;
select * from WORKERS;
```

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

Предикат **LIKE** сравнивает строку, указанную в первом выражении, для вычисления значения строки, называемого проверяемым значением, с образцом, который определен во втором выражении для вычисления значения строки. В образце разрешается использовать два трафаретных символа:

- символ подчеркивания (_), который можно применять вместо любого единичного символа в проверяемом значении;
- символ процента (%) заменяет последовательность любых символов (число символов в последовательности может быть от 0 и более) в проверяемом значении.

Пример 1

```
SQL> select * from DETAILS where DETAIL_NAME like 'front%';

     ID_NAME DETAIL_NAME
============ ====================
           3 front spring
           6 front damper
           8 front left fara
```

Пример 2

```
SQL> select * from CLIENTS where PHONE like '%014';

   ID_CLIENT FULL_NAME                                                PHONE
============ ================================================== ==========
           4 SPIDER MAN                                         4072849014
         336 Яковлев                                            8020001014
        1130 Дмитриев                                           6225350014
        3934 Нестеров                                           6042110014
        3975 Кулагин                                            3705514014
        4099 Евсеев                                             3164470014
        4581 Кириллов                                           1774446014
        4739 Абрамов                                            3936440014
        4970 Родионов                                           4697086014
        5348 Карпов                                             1358813014
        5735 Летов                                              4265507014
        6864 Антонов                                            3628127014
        6891 Иванов                                             2009777014
        8663 Дементьев                                          6531401014
```

Пример 3

```
SQL> select * from CARS where VIN like 'R%WTA%';

ID_CAR   ID_CLIENT MARK                         MODEL                        COLOR      YEAR_OF_ISSUE VIN
======== ========== ============================ ============================ ========== ============= =================
46BVIDF         969 Chrysler                     VN:Q<57                      Yellow     1998          RWPSXSWTAUMJNJGKD
8FPHLPU4N      1905 Qvale                        @ZIEI54RIZF>MP:I0XO0;YZ      Dark coral 1985          RZAWTAAEOVLDVDRTZ
```

Предикат **BETWEEN** проверяет, попадают ли значения проверяемого выражения в диапазон, задаваемый пограничными выражениями, соединяемыми служебным словом **AND**. Естественно, как и для предиката сравнения, выражения в предикате **BETWEEN** должны быть совместимы по типам.

Пример 1

```
SQL> select * from CARS where ID_CLIENT between 1990 and 1999;

ID_CAR   ID_CLIENT MARK                         MODEL                        COLOR      YEAR_OF_ISSUE VIN
======== ========== ============================ ============================ ========== ============= =================
?VP??HY@Z      1991 Oldsmobile                   R:1E8LX6UD6F@:TX;UV          Arsenic    1991          KUJUDDHSYFRJHWKXL
PISMRK         1990 Hindustan                    ?RN?JYORR<                   Denim      1983          KTZCSJFDLMAQUSCUM
AM42J7         1999 Perodua                      J5OWH7TU17<2=A6RY>25U07IR>;: Copper     1991          EMJXQSZQOLGXMPLFV
S4W@FM;DE      1998 Marcos                       34RNE55ST=L>                 Aquamarine 1999          NRXZDIXSSOHDVHBKK
5<4K?Q9        1994 SEAT                         =8>;F46W<AZ=                 Bistre     2004          PQGMMXZMRLZXBYRYU
JUQNO91TU      1998 Wuling                       M2;V1<41Q7                   Mint Green 1987          AONLUSQQDVHPRIWCN
J6E4=4H?       1996 Fuqi                         :?1E6CKD34RPLLY2JM0QUEQE2QWL Olive      2010          VJBUPBAXHAVEIDVVR
HH<@@D;9       1990 FSC                          6VULP1=D?AOKLTWY4KMJOO>8Z4C@ Dark Olive 2009          TUEJAVEPIDOPKBJXG
Q0A964>K       1990 Westfield                    =YKRL2@KNV>YEDA<V3BE?S       Tea Green  2014          AUIUWFGCDJIDNENET
ADE3YXZ        1996 Austin                       GT>EVSGW;H1X4JH841F?P=FT     Tenne      1999          WUQMDFQEMBIUAGLHE
;CG03OZ        1992 Foton                        CS=4F5?VS9M?0=S:VBT1H        Pale mauve 2001          AFDQVFGGAPIJITAQN
O0IB0;HVU      1996 Renaissance Cars             UME6BEM                      Rust       2005          SCEXIFEXFFPDJMWYS
```

Пример 2

```
SQL> select * from DETAILS_CATALOG where COST between 10000 and 10100;

   ID_DETAIL NAME                 COST EXISTENCE_Y_OR_N       AMOUNT ID_MANUFACTURER ID_DETAIL_NAME ORIGINAL VIN
============ ============== ========== ================ ========== =============== ============== ======== =================

        2920 hpqvafeuksgovlt      10089 Y                         4              32              8 y        UUORPOVCIXGHIDUXQ
        3687 rmlfmld              10053 N                         5             123             53 n        CFKIKXWAPKEJFMHTK
        6804 rjkxoj               10046 N                         9             345             14 n        QVLDTYTIOFBJTWRNN
        7712 mgztfdv              10007 Y                        10             123             33 n        AGIODNJDUGHCMODYF
        7791 iqixctefvubnou       10046 Y                         1             123             39 n        IOHPXFPBGHEHADUYU
        8366 vvysgydhgbexci       10030 N                         3              32             60 y        NNHIQWKJDFPNLRMCP
```

Пример 3

```
SQL> select * from CLIENTS where ID_CLIENT between 1 and 10;

   ID_CLIENT FULL_NAME                                           PHONE
============ ================================================== ==========
           1 TONY STARK                                         4024864932
           2 CAPTAIN AMERICA                                    4012863920
           3 ANT MAN                                            4073920184
           4 SPIDER MAN                                         4072849014
           5 BLACK WIDOW                                        4092846710
           6 INCREDIBLE HULK                                    4001468242
           7 HOWK EYE                                           4081947436
           8 RED WITCH                                          4081394302
           9 WINTER SOLDIER                                     4001394824
          10 THOR THE SON OF ODIN                               4001847334
```

Предикат **IN** определяет, будет ли значение проверяемого выражения обнаружено в наборе значений, который либо явно определен, либо получен с помощью табличного подзапроса. Здесь табличный подзапрос – это обычный оператор **SELECT**, который создает одну или несколько строк для одного столбца, совместимого по типу данных со значением проверяемого выражения. Если целевой объект эквивалентен хотя бы одному из указанных в предложении **IN** значений, истинностное значение предиката **IN** будет равно **TRUE**. Если для каждого значения X в предложении **IN** целевой объект <> X, истинностное значение будет равно **FALSE**. Если подзапрос выполняется, и результат не содержит ни одной строки (пустая таблица), предикат принимает значение **FALSE**. Когда не соблюдается ни одно из упомянутых выше условий, значение предиката равно **UNKNOWN**.

Пример 1

```
SQL> select * from REPAIR where ID_TYPE in (1);

   ID_REPAIR      ID_TYPE NAME                                          ID_ORDER_SOA
============ ============ ================================================== ============
           1            1 complex                                                       1
           6            1 cleaning                                                      5
```

Пример 2

```
SQL> select * from ORDERS where ID_WARRANTY not in (0);

    ID_ORDER ID_CAR       BEGIN_AT   END_AT     ID_WARRANTY
============ ========= =========== =========== ============
           5 nf74ngfsd 2016-05-30  2016-05-31              2
           9 nf7cvcvbr 2013-03-15  2013-03-17              1
          10 nf74n8f42 2016-05-21  2016-06-13              3
```

Пример 3

```
SQL> select * from REPAIR_ORDERS where ID_WORKER in (1,4);

ID_REPAIR_ORDER    ID_REPAIR     ID_ORDER    ID_WORKER
=============== ============ ============ ============
              2            2            1            1
              4            4            5            4
              6            6            6            1
              7            2            7            4
```

3. Создайте в запросе вычисляемое поле

Пример 1

```
SQL> select max(COST) from DETAILS_CATALOG;

         MAX
============
      100000
```

Пример 2

```
SQL> select sum(PURCHASE_PRICE) from PURCHASE;

                  SUM
======================
                68800
```

Пример 3

```
SQL> select count(ID_MANUFACTURER) from MANUFACTURERS;

        COUNT
============
            3
```

4. Сделайте выборку всех данных с сортировкой по нескольким полям

```
SQL> select ID_DETAIL, NAME, ID_DETAIL_NAME, COST from DETAILS_CATALOG where ID_DETAIL_NAME in (5) order by COST;

   ID_DETAIL NAME            ID_DETAIL_NAME         COST
============ ============== ============== ============
         232 BAK                         5          500
         231 BAK                         5         1010
        3870 nkbndobmqfe                 5         1230
        4693 ckioxdzcn                   5         1440
        4478 punyzgb                     5         2042
        6304 spuqzhfs                    5         2646
        5172 zcomucovrqjib               5         3007
        7994 ielndhhpszj                 5         4043
        9374 rzwqdnhc                    5         4088
        4252 iuvwzearawbuh               5         5159
        2643 yuqzlnexvyxr                5         5738
        2128 ogdjdngvp                   5         5818
         787 gcmxugjzfffnau              5         6612
        1457 irxsdyxgtpg                 5         8009
        8874 jugmmneyg                   5         8795
        9515 obnwzvfd                    5         8949
        2612 axddboq                     5         9721
        8976 dflvgwuiwym                 5         9920
         527 jizfkyw                     5        10585
         366 gpcxnuwa                    5        10699
```

… и т.д.

5. Создайте запрос, вычисляющий несколько совокупных характеристик таблиц

```
SQL> select count(ID_DETAIL), max(AMOUNT), min(AMOUNT) from DETAILS_CATALOG;

        COUNT          MAX          MIN
============ ============ ============
       10100           10            0
```

6. Сделайте выборку данных из связанных таблиц (не менее двух примеров)

Пример 1

```
SQL> select DETAILS_CATALOG.ID_DETAIL, DETAILS_CATALOG.NAME, DETAILS.DETAIL_NAME, DETAILS_CATALOG.COST
CON> from DETAILS_CATALOG, DETAILS
CON> where DETAILS_CATALOG.ID_DETAIL_NAME = DETAILS.ID_NAME
CON> and DETAILS_CATALOG.COST <= 800;

   ID_DETAIL NAME            DETAIL_NAME                 COST
============ ============== ================== ============
         567 MIRROW         left mirrow                  132
         232 BAK            bak                          500
         451 ohtzgmyobdmc   Фильтры                      577
         506 xxabmqptpof    Тюнинг                       741
         609 xsofrlzwgcuxpbh Автохимия                   763
         662 snjxugssld     Водяные насосы               701
        1495 tcjrotuj       Коробка передач              599
        1728 qysisksxr      rear spring                  691
        1730 cyfeakg        window left                  684
        2101 ghjedgiqjp     Автозапчасти                 561
        2384 lidgjmmyneebdcf Электрооборудование         552
        2480 cdqgrquut      Фаркоп                       776
        3947 lwkjrxclv      Водяные насосы               638
        4714 wedmrcxfxn     Свечи зажигания              769
        4788 iicqhhnphbjvu  Фаркоп                       616
        4985 lpdptusedr     Бензонасосы                  764
        5117 stpyckq        left mirrow                  637
        5589 dnkhndp        Расходные материалы          552
        5671 sgzlxkng       Водяные насосы               561
        6133 wpormcvgivj    Автохимия                    748
```

… и т.д.

Пример 2

```
SQL> select REPAIR_ORDERS.ID_REPAIR_ORDER, WORKERS.FULL_NAME
CON> from REPAIR_ORDERS, WORKERS
CON> where REPAIR_ORDERS.ID_WORKER = WORKERS.ID_WORKER
CON> and REPAIR_ORDERS.ID_REPAIR between 1 and 4;

ID_REPAIR_ORDER FULL_NAME
=============== ==================================================
              1 Vladimir
              2 Ivan
              3 Konstantin
              4 Sergey
              7 Sergey
              8 Vladimir
             10 Vladimir
```

7. Создайте запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки

Предложение **HAVING** применяется после группировки для определения аналогичного предиката, фильтрующего группы пok значениям агрегатных функций. Это предложение необходимо для проверки значений, которые получены с помощью агрегатной функции не из отдельных строк источника записей, определенного в предложении **FROM**, а из групп таких строк.

```
SQL> select ID_DETAIL_NAME, count(ID_DETAIL_NAME) from DETAILS_CATALOG
CON> group by ID_DETAIL_NAME having count(ID_DETAIL_NAME) < 210;

ID_DETAIL_NAME        COUNT
============== ============
             3          202
            10          208
            15          209
            18          200
            19          196
            24          208
            31          201
            36          205
            40          195
            44          201
            49          201
            57          208
            58          205
```

8. Придумайте и реализуйте пример использования вложенного запроса

Сделаем выборку наименований деталей из каталога, которые стоят более 99 500 и которые являются оригинальными, и их поставщиком является WAG или BP.

```
SQL> select DETAILS_CATALOG.NAME, DETAILS_CATALOG.COST, DETAILS_CATALOG.ID_MANUFACTURER
CON> from DETAILS_CATALOG
CON> where DETAILS_CATALOG.ID_MANUFACTURER in (select ID_MANUFACTURER from MANUFACTURERS
CON> where ID_MANUFACTURER in (32,123))
CON> and DETAILS_CATALOG.ORIGINAL like 'y%'
CON> and DETAILS_CATALOG.COST > 99500;
```

NAME	COST	ID_MANUFACTURER
czfpiy	99883	32
chqjad	99940	123
naokklinqth	99772	123
vzoolsl	99739	123
iwgsvyckvbtp	99830	32
xvoabhoumpwsngb	99736	123
evjlegbvwkhmzx	99836	123
dojvglqh	99897	32
chygsum	99870	123
umbmwvcceqg	99670	123
eixmdxx	99567	32
hkvdhfeybp	99722	32
maqelzkhzh	99741	32
gzvqbokz	99611	123
gttmhqdi	99634	32
ycdeqogalx	99785	123

9. С помощью оператора INSERT добавьте в каждую таблицу по одной записи

```
   insert into CARS values ('er342rw', '23','Toyota', 'corola', 'Dark grey', '2000',
'HFJKDNFRTUSOAKLDH');
   insert into CLIENTS values (10011, 'Rerix', '8943267382');
   insert into DETAILS values (62, 'Прожектора');
   insert into DETAILS_CATALOG values (10100, 'left progh', 3853, 'Y', 2, 123,
'HFJKDNFRTUSOAKLDH', 62, 'y');
   insert into MANUFACTURER values (4, 'AUTORAN', 'www.autoran.ru', '9483748593');
   insert into ORDERS values (11, 'er342rw', '25.09.2012', '13.10.2012');
   insert into ORDERS_SOA values (6, 10034, 4);
   insert into PURCHASE values (7, 10034, 3000);
   insert into REPAIR values (7, 3, 'PLUS', 6);
   insert into REPAIR_ORDERS values (11, 7, 11, 6);
   insert into TYPES_OF_REPAIR values (3, 'INPUT DETAILS', 2500, '3 week');
   insert into WARRANTYS values (4, '12.04.2014', 2);
   insert into WORKERS values (6, 'Ura', 'mech3', '8965435318', '86543256785');
```

Результат:

ID_CLIENT	FULL_NAME	PHONE
10011	Rerix	8943267382

ID_CAR	ID_CLIENT	MARK	MODEL	COLOR	YEAR_OF_ISSUE	VIN
er342rw	23	Toyota	corola	Dark grey	2000	HFJKDNFRTUSOAKLDH

ID_ORDER	ID_CAR	BEGIN_AT	END_AT	ID_WARRANTY
11	er342rw	2012-09-25	2012-10-13	<null>

ID_REPAIR_ORDER	ID_REPAIR	ID_ORDER	ID_WORKER
11	7	11	6

ID_WORKER	FULL_NAME	SPECIALTY	PHONE	PASSPORT
1	Ivan	mech1	5830495864	34534634341
2	Petr	mech2	3468945764	34543903943
3	Konstantin	paint	3845769080	23787563453
4	Sergey	gear	8908837495	23489734895
5	Vladimir	assistent	8349572146	23849573984
6	Ura	mech3	8965435318	86543256785

ID_REPAIR	ID_TYPE	NAME	ID_ORDER_SOA
1	1	complex	1
2	2	paint	2
3	2	tuning	3
4	2	engine	4
5	2	suspension	5
6	1	cleaning	5
7	3	PLUS	6

ID_REPAIR	ID_TYPE	NAME	ID_ORDER_SOA
1	1	complex	1
2	2	paint	2
3	2	tuning	3
4	2	engine	4
5	2	suspension	5
6	1	cleaning	5
7	3	PLUS	6

ID_TYPE	NAME	PRICE	WARRANTY
1	work	1500	<null>
2	work2	300	2 weeks
3	INPUT DETAILS	2500	3 week

ID_MANUFACTURER	NAME	SITE	PHONE
345	EUROAUTO	www.euroauto.ru	435456
123	BP	www.bp.com	56794536
32	WAG	www.wag.de	3475897345
4	AUTORAN	www.autoran.ru	9483748593

ID_NAME	DETAIL_NAME
62	Прожектора

ID_PURCHASE	ID_DETAIL	PURCHASE_PRICE
1	456	1300
2	231	1000
3	228	34000
4	322	32000
5	567	100
6	232	400
7	10034	3000

ID_WARRANTY	DATE_APP	ID_ORDER
1	2013-03-25	9
2	2016-06-12	5
3	2016-06-30	10
4	2014-04-12	2

10. С помощью оператора UPDATE измените значения нескольких полей у всех записей, отвечающих заданному условию

```
SQL> select ID_DETAIL, EXISTENCE_Y_OR_N, AMOUNT from DETAILS_CATALOG
CON> where EXISTENCE_Y_OR_N = 'N' and AMOUNT > 0 and ID_DETAIL <= 10;

   ID_DETAIL EXISTENCE_Y_OR_N       AMOUNT
============ ================ ============
           0 N                           7
           1 N                           5
           4 N                           8
           6 N                           2
           8 N                           6

SQL> update DETAILS_CATALOG set EXISTENCE_Y_OR_N = 'Y' where EXISTENCE_Y_OR_N = 'N' and AMOUNT > 0;
SQL> select ID_DETAIL, EXISTENCE_Y_OR_N, AMOUNT from DETAILS_CATALOG
CON> where EXISTENCE_Y_OR_N = 'N' AND AMOUNT >= 10;
```

Ничего не выводится.

11. С помощью оператора DELETE удалите запись, имеющую максимальное (минимальное) значение некоторой совокупной характеристики

```
SQL> select count(ID_CAR) from CARS where YEAR_OF_ISSUE < 1990;

       COUNT
============
        1530

SQL> delete from CARS where YEAR_OF_ISSUE < 1990;
SQL> select count(ID_CAR) from CARS where YEAR_OF_ISSUE < 1990;

       COUNT
============
           0
```

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

```
SQL> select count(ID_CLIENT) from CLIENTS;

       COUNT
============
       10011

SQL> delete from CLIENTS where ID_CLIENT not in
CON> (select ID_CLIENT from CARS);
SQL> select count(ID_CLIENT) from CLIENTS;

       COUNT
============
        5651
```

Выполнение индивидуальных запросов

1. **Вывести 5 заказов с максимальной общей стоимостью.**

Скрипт:

```
connect 'D:/DB/SERVICE.FDB' user 'SYSDBA' password 'masterkey';
commit;

-- 1 --
create view MAX_COST as
    select first 5 REPAIR_ORDERS.ID_REPAIR_ORDER, max(REPAIR_ORDERS.PRICE) as PRICE
    from REPAIR_ORDERS
    group by REPAIR_ORDERS.ID_REPAIR_ORDER
    order by PRICE desc;
select * from MAX_COST;
```

Результат:

ID_REPAIR_ORDER	PRICE
2	34643
3	34631
8	8533
7	6321
4	4632

2. **Вывести 5 работников, на работу которых чаще всего поступают обращения по гарантии.**

Скрипт:

```
create view WORK as
    select REPAIR_ORDERS.ID_WORKER
    from REPAIR_ORDERS
    where ID_WARRANTY <> 0;
create view BAD_WORKERS as
    select first 5 WORK.ID_WORKER, count(WORK.ID_WORKER) as AMOUNT
    from WORK
    group by WORK.ID_WORKER
    order by AMOUNT desc;
select * from BAD_WORKERS;
```

Результат:

ID_WORKER	AMOUNT
1	2
2	2
5	2
3	1
4	1

3. Вывести 10 деталей, вместо которых чаще всего заказывают аналоги.

Скрипт:

```
connect 'D:/DB/SERVICE.FDB' user 'SYSDBA' password 'masterkey';
commit;

-- 3 --
create view DET_ALL_S as
    select ORDERS_SOA.ID_DETAIL, sum(ORDERS_SOA.AMOUNT) as AMOUNT
    from ORDERS_SOA
    group by ORDERS_SOA.ID_DETAIL;

create view DET_ALL_Y as
    select DET_ALL_S.ID_DETAIL, DET_ALL_S.AMOUNT,
           DETAILS_CATALOG.VIN, DETAILS_CATALOG.ID_DETAIL_NAME
    from DET_ALL_S, DETAILS_CATALOG
    where DET_ALL_S.ID_DETAIL = DETAILS_CATALOG.ID_DETAIL
           and DETAILS_CATALOG.ORIGINAL =  'y';

create view DET_ALL_N_1 as
    select DET_ALL_S.ID_DETAIL, DET_ALL_S.AMOUNT,
           DETAILS_CATALOG.VIN, DETAILS_CATALOG.ID_DETAIL_NAME
    from DET_ALL_S, DETAILS_CATALOG
    where DET_ALL_S.ID_DETAIL = DETAILS_CATALOG.ID_DETAIL
           and DETAILS_CATALOG.ORIGINAL =  'n';

create view DET_ALL_N as
    select DET_ALL_N_1.VIN, DET_ALL_N_1.ID_DETAIL_NAME, sum(DET_ALL_N_1.AMOUNT) as AMOUNT
    from DET_ALL_N_1
    group by DET_ALL_N_1.VIN, DET_ALL_N_1.ID_DETAIL_NAME;

create view TOP_10_DET as
    select first 10 DET_ALL_Y.ID_DETAIL, (DET_ALL_N.AMOUNT - DET_ALL_Y.AMOUNT) as AMT
    from DET_ALL_Y, DET_ALL_N
    where DET_ALL_Y.VIN = DET_ALL_N.VIN and DET_ALL_Y.ID_DETAIL_NAME =
DET_ALL_N.ID_DETAIL_NAME
    order by AMT desc;

select * from DET_ALL_S where ID_DETAIL <= 50;
select * from DET_ALL_Y;
select first 5 * from DET_ALL_N;
select * from TOP_10_DET;
```

Результат:

ID_DETAIL	AMT
511	2129
1520	1225
9310	918
8510	910
10103	889
9808	527
8323	481
7901	400
5989	349
8760	243

Сохранение запросов в виде хранимых процедур

1. INSERT

Скрипт:

```
connect 'D:/DB/SERVICE.FDB' user 'SYSDBA' password 'masterkey';
commit;

set term ! ;
create procedure INS_CLIENT(id int, name varchar(50), ph varchar(10))
as
begin
    insert into CLIENTS values (:id, :name, :ph);
end
!
set term ; !
execute procedure INS_CLIENT(10012, 'Feryn', '7564738748');
select * from CLIENTS where ID_CLIENT >= 10010;
```

Результат:

ID_CLIENT	FULL_NAME	PHONE
10010	Кузнецов	9103722303
10011	Rerix	8943267382
10012	Feryn	7564738748

2. UPDATE

Скрипт:

```
set term ! ;
create procedure UP_EX_Y(exi char(1))
as begin
    update DETAILS_CATALOG
    set EXISTENCE_Y_OR_N = :exi
    where AMOUNT > 0;
end
!
set term ; !

select * from DETAILS_CATALOG where EXISTENCE_Y_OR_N = 'N' and AMOUNT > 0;
execute procedure UP_EX_Y('Y');
select * from DETAILS_CATALOG where EXISTENCE_Y_OR_N = 'N' and AMOUNT > 0;
```

Результат:

ID_DETAIL	NAME	COST	EXISTENCE_Y_OR_N	AMOUNT	ID_MANUFACTURER	ID_DETAIL_NAME	ORIGINAL	VIN
2	lxcmuureqgakuz	73170	N	4	123	16	n	SXDERIXAZWKORHORP
7	gkkmgjji	94268	N	7	345	19	n	DDTBZWFVIOVXYTOJG
17	iaeucuv	29301	N	4	345	35	n	XHDAMMELZVBRJSEAD

```
SQL> select * from DETAILS_CATALOG where ID_DETAIL in (2, 7, 17);
```

ID_DETAIL	NAME	COST	EXISTENCE_Y_OR_N	AMOUNT	ID_MANUFACTURER	ID_DETAIL_NAME	ORIGINAL	VIN
2	lxcmuureqgakuz	73170	Y	4	123	16	n	SXDERIXAZWKORHORP
7	gkkmgjji	94268	Y	7	345	19	n	DDTBZWFVIOVXYTOJG
17	iaeucuv	29301	Y	4	345	35	n	XHDAMMELZVBRJSEAD

3. DELETE

Скрипт:

```
set term ! ;
create procedure DEL_DET(id_m int)
as begin
    delete from ORDERS_SOA
    where ID_DETAIL = (select ID_DETAIL from DETAILS_CATALOG where ID_MANUFACTURER = :id_m);
end
!
set term ; !

select count(ORDERS_SOA.ID_DETAIL) from ORDERS_SOA
    where ORDERS_SOA.ID_DETAIL = (select ID_DETAIL from DETAILS_CATALOG where ID_MANUFACTURER
= 4);
execute procedure DEL_DET(4);
select count(ORDERS_SOA.ID_DETAIL) from ORDERS_SOA
    where ORDERS_SOA.ID_DETAIL = (select ID_DETAIL from DETAILS_CATALOG where ID_MANUFACTURER
= 4);
```

Результат:

```
      COUNT
============
           2

      COUNT
============
           0
```

Выводы

В ходе работы был изучен язык создания запросов в базах данных SQL-DML.

Были изучены различные стандартные запросы, которые могут потребоваться в любой базе данных, а также были созданы представления и процедуры основанные на индивидуальном задании. Выполнение последних было намного сложнее, так как составление такого рода запросов требует особого внимания.

Чтобы работать с базой данных было проще, существуют хранимые процедуры. В нашем случае мы реализовали процедуры, с помощью которых можно добавлять, менять или удалять данные.