Санкт-Петербургский Политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчёт о лабораторной работе №4**

**Дисциплина**: Базы данных

**Тема**: Язык SQL-DML

Выполнил студент гр. 43501/1 М.А. Иконников

(подпись)

Руководитель А.В. Мяснов

(подпись)

“ ” 2015 г.

Санкт-Петербург

2015

1. **Цель работы**

Познакомить студентов с языком создания запросов управления данными SQL-DML.

1. **Программа работы**
2. Изучите SQL-DML
3. Выполните все запросы из списка стандартных запросов. Продемонстрируйте результаты преподавателю.
4. Получите у преподавателя и реализуйте SQL-запросы в соответствии с **индивидуальным** заданием. Продемонстрируйте результаты преподавателю.
5. Выполненные запросы SELECT сохраните в БД в виде представлений, запросы INSERT, UPDATE или DELETE -- в виде ХП. Выложите скрипт в Subversion.
6. **Язык SQL**

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

1. **Выполнение работы**
2. Выборка всех данных из каждой таблицы

connect 'C:\SCLUB' user 'SYSDBA' password 'masterkey';

create view AccountingSel as select \* from ACCOUNTING;

create view AccServicesSel as select \* from ACCOUNTINGSERVICES;

create view AddServicesSel as select \* from ADDSERVICES;

create view ClientsSel as select \* from CLIENTS;

create view DiscountsSel as select \* from DISCOUNTS;

create view RoomsSel as select \* from ROOMS;

create view SeasonTicketsSel as select \* from SEASONTICKETS;

create view SectionsSel as select \* from SECTIONS;

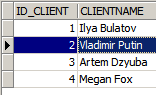
create view TicketTypesSel as select \* from TICKETTYPES;

create view TicketTypesToSectSel as select \* from TICKETTYPESTOSECTIONS;

create view TrainersSel as select \* from TRAINERS;

create view VisitsSel as select \* from VISITS;

Выборка данных из таблицы CLIENTS:



1. Выборка данных из одной таблицы при нескольких условиях, с использованием логических операций, LIKE, BETWEEN, IN

create view selectPutin as select \* from CLIENTS where CLIENTNAME like '% Putin %';

create view selectExpTrainers as select \* from TRAINERS where TRAINEREXP

between 3 and 10;

create view selectGymTennis as select \* from SECTIONS where "ID\_ROOM" in (2,3);

Были созданы следующие выборки:

* Из таблицы CLIENTS показать клиентов с фамилией Putin
* Из таблицы TRAINERS показать тренеров с опытом работы от 3 до 10 лет
* Из таблицы SECTIONS показать секции, которые расположены на площадках с id = 2 или 3

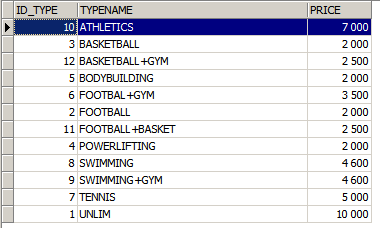
Пример выборки с тренерами с опытом работы от 3 до 10 лет:



1. Выборка всех данных с сортировкой по нескольким полям

create view selectSortTicketTypes as select \* from TICKETTYPES order by TYPENAME asc, PRICE asc;

Типы абонементов, отсортированные по имени и цене:



1. Выборка данных из связанных таблиц

create view selectRoomsSections as select ROOMS.roomname as Room,

SECTIONS.sectionname as Section from ROOMS, SECTIONS where SECTIONS.id\_room = ROOMS.id\_room;

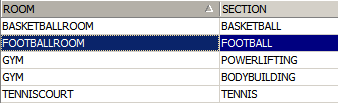
create view selectTrainersSectionsExp as select TRAINERS.trainername as TrainerName,

SECTIONS.sectionname as Section, TRAINERS.trainerexp as Exp from TRAINERS,

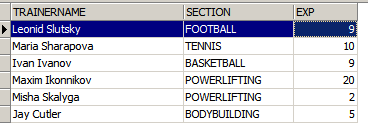
SECTIONS where TRAINERS.id\_section = SECTIONS.id\_section;

Выборки, отображающие:

* Площадки с видами спорта:



* Тренеров, привязанных к определенным секциям и их стаж работы:

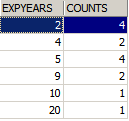


1. Запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки

create view selectCountExpTrainers as select TRAINERS.TRAINEREXP as ExpYears,

COUNT(TRAINERS.TRAINEREXP) as Counts from TRAINERS group by TRAINERS.TRAINEREXP;

Определение количества тренеров по стажу работы:

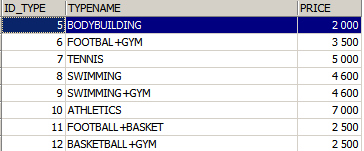


1. Использование вложенного запроса

create view selectUnusedTicketTypes as select \* from TICKETTYPES where ID\_TYPE

not in (select ID\_TYPE from SEASONTICKETS);

Отображение типов абонементов, которые не были куплены:



1. Использование оператора INSERT для добавления по одной записи в каждую таблицу:

connect 'C:\SCLUB' user 'SYSDBA' password 'masterkey';

create procedure insert\_acc ( i int, d int, p int, t int)

as begin

insert into ACCOUNTING (id\_client, id\_discount, paid, id\_ticket)

values (:i, :d, :p, :t);

end;

create procedure insert\_accs ( i int, d int)

as begin

insert into ACCOUNTINGSERVICES (id\_client, id\_service)

values (:i, :d);

end;

create procedure insert\_addserv ( i int, t int, n varchar(255), s int)

as begin

insert into ADDSERVICES (id\_service, id\_type, servicename, servprice)

values (:i, :t, :n, :s);

end;

create procedure insert\_clients ( i int, n varchar(255), t int, c int, d int)

as begin

insert into CLIENTS (id\_client, clientname, telnumber, numofclasses, id\_discount)

values (:i, :n, :t, :c, :d);

end;

create procedure insert\_disc ( i int, n varchar(255), v int)

as begin

insert into DISCOUNTS (id\_discount, discountname, discountvalue)

values (:i, :n, :v);

end;

create procedure insert\_room ( i int, n varchar(255))

as begin

insert into ROOMS (id\_room, roomname)

values (:i, :n);

end;

create procedure insert\_ticket ( i int, d int, p date)

as begin

insert into SEASONTICKETS (id\_ticket, id\_type, period)

values (:i, :d, :p);

end;

create procedure insert\_section ( i int, n varchar(255), r int)

as begin

insert into SECTIONS (id\_section, sectionname, id\_room)

values (:i, :n, :r);

end;

create procedure insert\_type ( i int, n varchar(255), p int)

as begin

insert into TICKETTYPES (id\_type, typename, price)

values (:i, :n, :p);

end;

create procedure insert\_ttts( i int, s int)

as begin

insert into TICKETTYPESTOSECTIONS (id\_type, id\_section)

values (:i, :s);

end;

create procedure insert\_trainer ( i int, n varchar(255), e int, s int)

as begin

insert into TRAINERS (id\_trainer, trainername, trainerexp, id\_section)

values (:i, :n, :e, :s);

end;

create procedure insert\_visit ( i int, v int)

as begin

insert into VISITS (id\_client, numberofvisits)

values (:i, :v);

end;

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

create procedure update\_disc (disc int) as

begin

update CLIENTS set ID\_DISCOUNT = :disc where CLIENTS.NUMOFCLASSES > 10;

end;

Добавление или изменение скидки для клиентов, у которых куплено больше 10 посещений.

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

create procedure del\_exptrainer (i int) as

begin

delete from TRAINERS where ID\_SECTION = :i and

TRAINEREXP = (select MIN(TRAINEREXP) from TRAINERS where ID\_SECTION = :i);

end;

Данная процедура удаляет тренера заданной секции с наименьшим стажем работы.

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

create procedure del\_types as

begin

delete from TICKETTYPES where ID\_TYPE not in (select ID\_TYPE from SEASONTICKETS);

end;

Удаление типов абонементов, которые никем не куплены.

Выполнение индивидуального задания:

Выполнить следующие запросы:

1. Отобразить 5 наиболее популярных типов абонементов.
2. Вывести 10 самых непопулярных секицй.
3. Удалить неиспользуемые типы абонементов.
4. Отобразить 5 наиболее популярных типов абонементов

create view top5types as select first 5 TICKETTYPES.typename, TICKETTYPES.typename as Name,

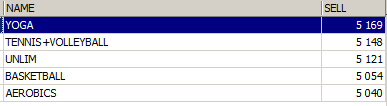
COUNT(SEASONTICKETS.ID\_TYPE) as Sell from TICKETTYPES, SEASONTICKETS

where SEASONTICKETS.ID\_TYPE = tickettypes.ID\_TYPE group by

TICKETTYPES.TYPENAME order by Sell desc;

Для тестирования данного скрипта было сгенирировано 100 000 записей в таблицу SEASONTICKETS.

Результат работы скрипта. 5 самых популярных типов абонементов:



Время выполнения: 530ms

1. Вывести 10 самых непопулярных секций

connect 'C:/SCLUB' user 'SYSDBA' password 'masterkey';

create view unpopular10sectionsver2 as

select

first 10 sectionname as Name,

count (id\_ticket) as Tickets

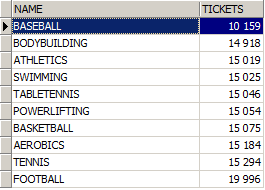
from SECTIONS s

join tickettypestosections t on s.id\_section = t.id\_section

join tickettypes tt on tt.id\_type = t.id\_type

join seasontickets ss on ss.id\_type = tt.id\_type

group by sectionname order by Tickets asc



Время выполнения: 1s 404ms

1. Удалить неиспользуемые типы абонементов

create procedure del\_types as

begin

delete from TICKETTYPES where ID\_TYPE not in (select ID\_TYPE from SEASONTICKETS);

end;

1. **Вывод**

В результате выполнения работы был изучен язык управления данными SQL-DML. Были выполнены стандартные запросы извлечения данных. Также были выполнены запросы в соответствии с индивидуальным заданием. Были изучены представления и хранимые процедуры, с помощью которых можно спокойно добавлять данные в БД. При выполнении работы проблем не было.

Использовались такие команды языка DML: insert (добавить), update (обновить), delete (удалить), select (выборка данных). Данный язык удобен для написания запросов разной сложности. При обращении к нескольким таблицам в запросе следует выбирать подходящую связь между таблицами для более быстрого выполнения SQL-запроса.