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 А.Е. Балясников

(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

#Выборка всех данных из каждой таблицы

CREATE VIEW Countries as SELECT \* FROM country;

CREATE VIEW Clubs as SELECT \* FROM club;

CREATE VIEW Leagues as SELECT \* FROM league;

CREATE VIEW Cups as SELECT \* FROM cup;

CREATE VIEW Stages as SELECT \* FROM stage\_type;

CREATE VIEW Cup\_Participants as SELECT \* FROM club\_cup;

CREATE VIEW League\_Participants as SELECT \* FROM club\_league;

CREATE VIEW Transfers as SELECT \* FROM transfer;

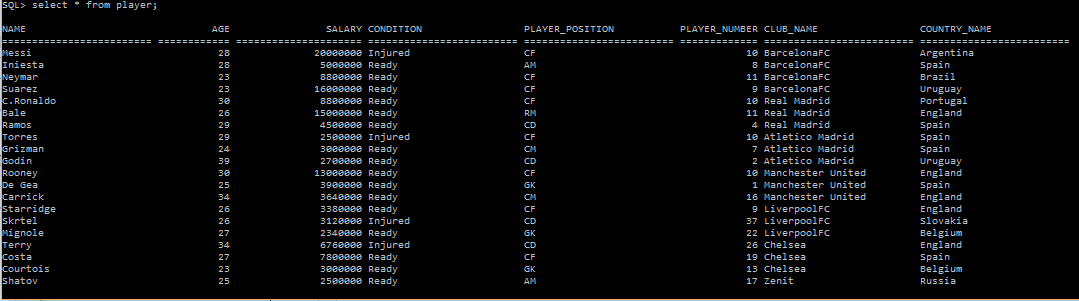
CREATE VIEW Games as SELECT \* FROM game;

CREATE VIEW Players as SELECT \* FROM player;

CREATE VIEW Player\_In\_Games as SELECT \* FROM player\_game;

CREATE VIEW Actions as SELECT \* FROM action;

Выборка данных из таблицы CLIENTS:



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

CREATE VIEW Barcelona\_Salaries AS SELECT name, salary

FROM player

WHERE club\_name = 'BarcelonaFC'

AND player.salary BETWEEN 0 AND 15000000;

CREATE VIEW MLL\_coaches AS SELECT name, coach

FROM club WHERE city IN ('Madrid', 'Liverpool', 'Lyon');

CREATE VIEW LiverpoolFC\_transfers AS SELECT player\_name, price

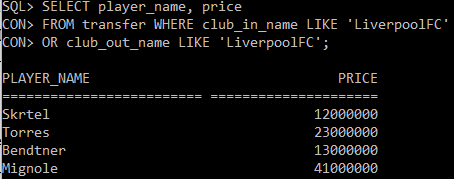
FROM transfer WHERE club\_in\_name LIKE 'LiverpoolFC'

OR club\_out\_name LIKE 'LiverpoolFC';

Были созданы следующие выборки:

1. Выбрать имя и заработную плату игрока из Барселоны, где зарплата между 0 и 15000000.
2. Выбрать название клуба и его тренера для клубов, из городов Мадрид, Ливерпуль, Лион.
3. Выбрать игроков и цену, за которую они перешли в LiverpoolFC или ушли из LiverpoolFC.

Приведем пример для третьего пункта:

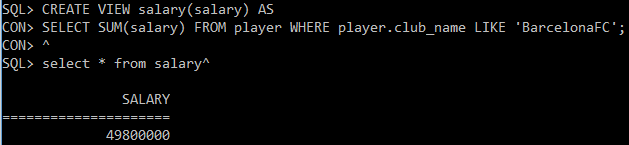


1. Вычисляемое поле в запросе.

#Вычисляемое поле в запросе

CREATE VIEW salary(salary) AS

SELECT SUM(salary) FROM player WHERE player.club\_name LIKE 'BarcelonaFC';



Здесь вычисляется суммарная зарплата всех игроков клуба Барселоны.

1. Пример сортировки по нескольким полям:

CREATE VIEW club\_cities(name, city) AS

SELECT name, city FROM club ORDER BY city,name;



Сортируем клубы по алфавиту их городов.

1. Выборка данных из связанных таблиц

CREATE VIEW England\_Players(name, salary, club) AS

SELECT player.name, player.salary, club.name

FROM player, club

WHERE player.country\_name

LIKE 'England'

AND player.club\_name = club.name;

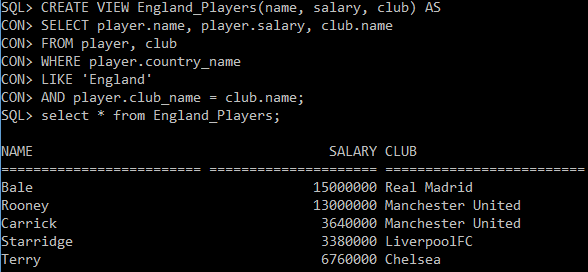
CREATE VIEW Madrid(name, salary, club) AS

SELECT club\_league.club\_name, club\_league.league\_name, club.coach

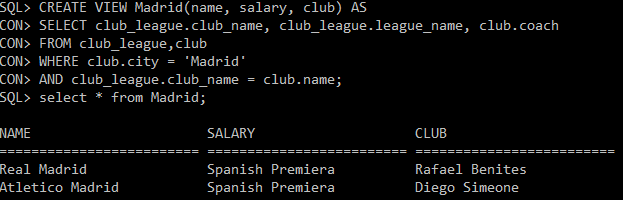
FROM club\_league,club

WHERE club.city = 'Madrid'

AND club\_league.club\_name = club.name;



На рисунке выше видно, что первый запрос выбирает имя, зарплату и клуб всех игроков из Англии.



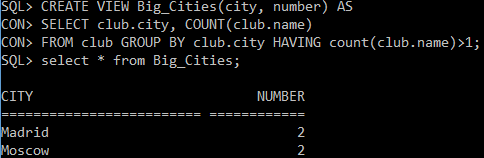
Как видно выше, выбираем клуб; лигу, где он играет и его тренера из разных таблиц, при условии, что этот клуб из города Мадрид.

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

CREATE VIEW Big\_Cities(city, number) AS

SELECT club.city, COUNT(club.name)

FROM club GROUP BY club.city HAVING count(club.name)>1;



Выше, мы выбрали города, в которых находятся больше одной команды.

Это Москва и Мадрид.

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

# DML INSERT UPDATE DELETE

set term ^;

CREATE PROCEDURE insert\_country ( n VARCHAR(25), rate INT)

AS BEGIN

INSERT INTO country(name, rating)

VALUES(:n, :rate);

END^

CREATE PROCEDURE insert\_club ( name VARCHAR(25), coach VARCHAR(25),

owner VARCHAR(25), site VARCHAR(25),

city VARCHAR(25), country\_name VARCHAR(25))

AS BEGIN

INSERT INTO club(name, coach, owner, site, city, country\_name)

VALUES(:name, :coach, :owner, :site, :city, :country\_name);

END^

CREATE PROCEDURE insert\_league ( name VARCHAR(25), country VARCHAR(25), div INT)

AS BEGIN

INSERT INTO league(name, country\_name, division)

VALUES(:name, :country, :div);

END^

CREATE PROCEDURE insert\_cup( name VARCHAR(25), country VARCHAR(25))

AS BEGIN

INSERT INTO cup(name, country\_name)

VALUES(:name, :country);

END^

CREATE PROCEDURE insert\_transfer( t\_id INT, t\_date DATE, club\_in VARCHAR(25),

club\_out VARCHAR(25), player VARCHAR(25), price BIGINT)

AS BEGIN

INSERT INTO transfer(transfer\_id, transfer\_date, club\_in\_name, club\_out\_name, player\_name, price)

VALUES(:t\_id, :t\_date, :club\_in, :club\_out, :player, :price);

END^

CREATE PROCEDURE insert\_game( id INT, club\_home VARCHAR(25), club\_guest VARCHAR(25),

goals\_home INT, goals\_guest INT, game\_date DATE, cup VARCHAR(25), league VARCHAR(25))

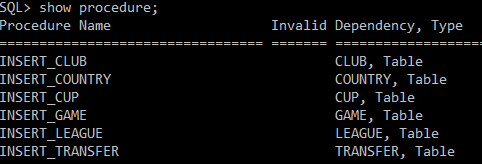
AS BEGIN

INSERT INTO game(game\_id, club\_home, club\_guest, goals\_home, goals\_guest, game\_date, cup\_name, league\_name)

VALUES(:id, :club\_home, :club\_guest, :goals\_home, :goals\_guest, :game\_date, :cup, :league);

END^

set term ;^



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

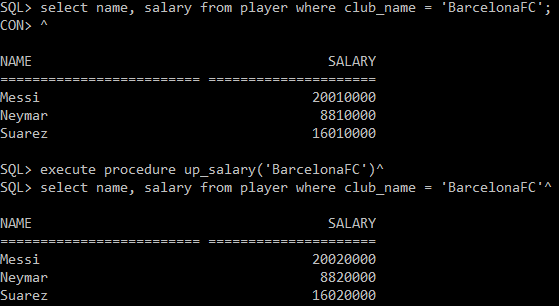
CREATE PROCEDURE up\_salary(club VARCHAR(25))

AS BEGIN

UPDATE player SET salary = salary+10000 WHERE player.club\_name = :club;

end^

Данная процедура повышает зарплату игроков клуба на 10000.



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

CREATE PROCEDURE rem\_player(club VARCHAR(25))

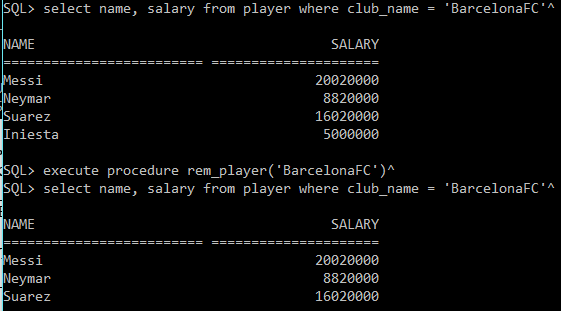
AS BEGIN

DELETE FROM player WHERE club\_name=:club

AND salary = (SELECT MIN(salary) FROM player WHERE club\_name=:club);

END^

Данная процедура удаляет игрока с самой маленькой зарплатой из клуба, являющегося аргументом процедуры.



Таким образом, был удалён мой любимый Iniesta.

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

CREATE PROCEDURE del\_unused\_country

AS BEGIN

DELETE FROM country WHERE name NOT IN

(SELECT country\_name FROM club UNION SELECT country\_name FROM player);

END^

Удаление всех стран, к которым не принадлежит ни один игрок и ни одна команда.

**Индивидуальное задание:**

1. Отобразить #10 клубов, которые заработали наибольшее количество денег на трансферах за заданный период времени.

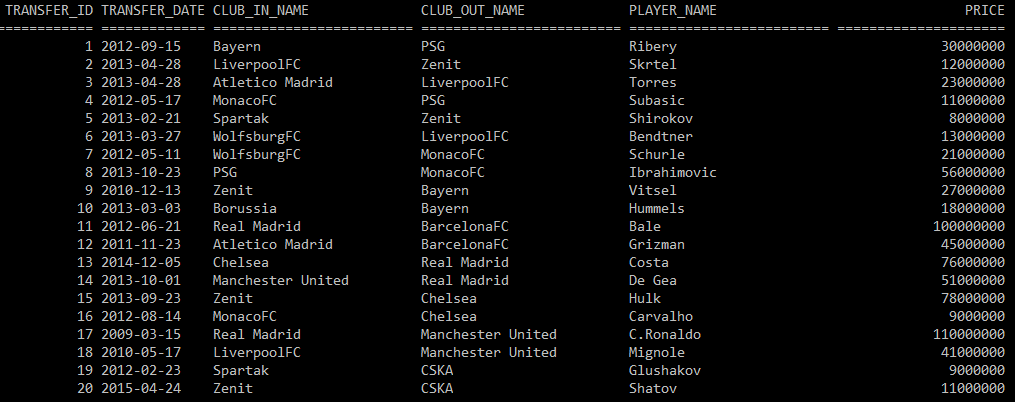
CREATE VIEW big10(Name, Summary) AS

SELECT FIRST 10 club\_out\_name, SUM(price)

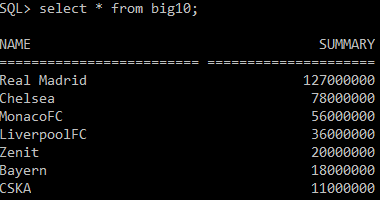
FROM transfer WHERE transfer\_date BETWEEN '2013-01-01' AND '2015-12-12'

GROUP BY club\_out\_name ORDER by SUM(price) desc;

Для проверки работы скрипта в таблицу transfer были добавлены ещё несколько записей, чтобы результат был нагляднее.



Тогда получаем следующие результаты:



1. Вывести 10 игроков, наиболее успешных по какому-либо показателю.

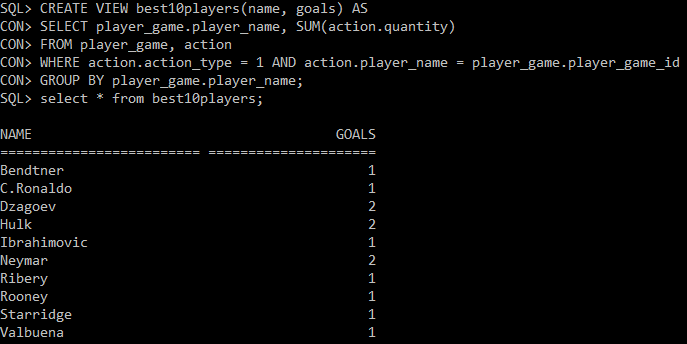
CREATE VIEW best10players(name, goals) AS

SELECT player\_game.player\_name, SUM(action.quantity)

FROM player\_game, action

WHERE action.action\_type = 1 AND action.player\_name = player\_game.player\_game\_id

GROUP BY player\_game.player\_name;



1. Удалить неиспользуемые типы действий футболистов:

CREATE PROCEDURE del\_unused\_types AS BEGIN

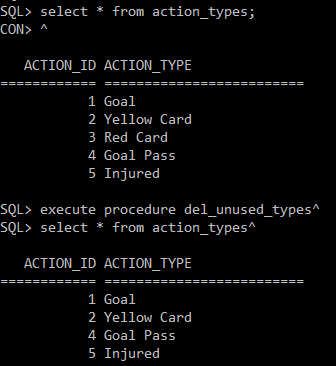
DELETE from action\_types WHERE action\_id NOT IN

(SELECT action\_types.action\_id FROM action, action\_types

WHERE action.action\_type = action\_types.action\_id GROUP BY action\_types.action\_id);

END^

Так, как никто у меня в базе данных не получал красную карточку, то именно это действие и должно удалиться:



Всё верно.

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.

Было очень неудобно создавать хранимые процедуры в isql, так как при их создании транзакция заканчивалась раньше нужного, оператором «;» в середине ХП. Поэтому пришлось использовать set term, который позволил заменить оператор конца транзакции. Было бы здорово, если бы компилятор(транслятор) языка распознавал бы хранимую процедуру и не заканчивал транзакцию, если оператор «;» встречался бы внутри неё.