Санкт-Петербургский Государственный Политехнический Университет

Институт Информационных Технологий и Управления

Кафедра Компьютерных Систем и Программных Технологий

**Отчет**

**о лабораторной работе №4**

**Дисциплина:** Базы данных

**Тема:** Хранимые процедуры

**Выполнил**: гр.43501/1 Саитов Илья

**Преподаватель:** Мяснов А.В.

Санкт-Петербург 2015

1. **Цель работы:**

Ознакомиться с возможностями реализации более сложной обработки данных на стороне сервера с помощью хранимых процедур.

1. **Программа работы:**

* Изучить возможности языка PSQL.
* Создать две хранимые процедуры в соответствии с индивидуальным заданием, полученным у преподавателя.
* Продемонстрировать результаты преподавателю.

1. **Выполнение работы:**

**а) Для всех призывников, имеющих последнее положительное заключение медкомиссии создадим призыв на службу по призыву в заданной в/ч на заданной должности. Повторных призывов быть не должно:**

Алгоритм процедуры следующий:

В цикле, в таблице MedicalData ищем по первичному ключу запись, которая соответствует группе здоровья (Group\_ID < 7) A1, A2, Б1, Б2, Б3 или Б4. Затем в таблице RecState по тому же первичному ключу, ищем запись, которая содержит значение «Отсрочка» в поле StateRS\_ID. При совпадении обоих условий мы идем в таблицу MilitaryService и добавляем запись для человека, соответствующего первичному ключу, о том, в какой военной части он будет служить, в каком звании, должности, военной учетной специальности и на какой срок. Также, включена возможность задания количества людей, необходимых для призыва – как только в таблице MilitaryService будет сделано заданное количество заприсей, происходит выход из цикла. Последние названные модификации, которые дополняют исходное задание введены для гибкости и удобства пользования процедурой.

Листинг ХП приведен в **приложении №2**.

Протестируем написанную процедуру на нашей БД, предварительно сгенерировав в нее 100 000 записей (количество людей). Для примера, призовем сначала 5 солдат в одну часть, 10 солдат в другую сроком на 1 год. Вызов процедуры осуществляется командой:

EXECUTE PROCEDURE REC\_APPEAL

QUANT\_R, // Количество призываемых людей

RANK\_R INT, // Идентификатор звания

POST\_R INT, // Идентификатор должности

SPECIAL\_R INT, // Идентификатор специальности

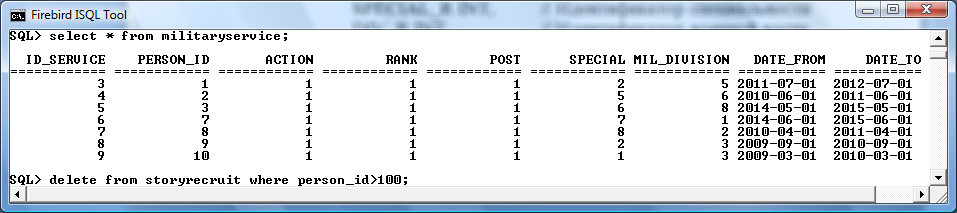
DIV\_R INT, // Идентификатор военной части

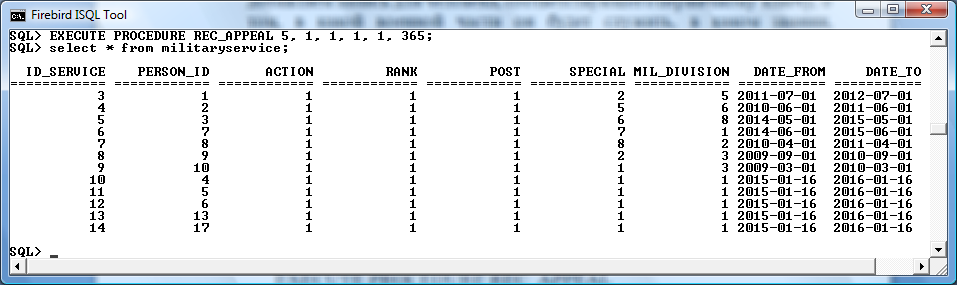
PERIOD\_DAY INT; // На какое количество дней

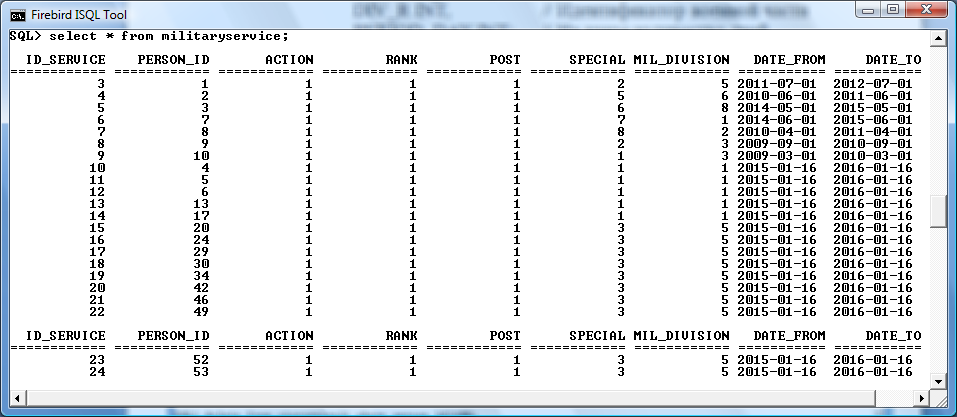
Пример вызова процедуры:

EXECUTE PROCEDURE REC\_APPEAL 5, 1, 1, 1, 1, 365;

Результаты выполнения команд:







**б) Для всех призывников, имеющих более заданного количества отсрочек, удалим все (кроме заданного количества) последние по хронологии отсрочки:**

Алгоритм процедуры следующий:

В цикле, в таблице StoryRecruit ищем по первичному ключу все записи отсрочек по определенному человеку. Данный запрос сортируется по убыванию «свежести» (т.е. по поля DateFrom и DateTo) и с помощью команды SKIP пропускается заданное количество записей, которые мы хотим оставить. В итоге этот запрос служит подзапросом к оператору DELETE, который передает ему идентификаторы записей, которые надо удалить.

Листинг ХП приведен в **приложении №3**.

Протестируем написанную процедуру на нашей БД, предварительно сгенерировав в таблицу StoryRecruit 1000 записей (по 100 людям). Берем так мало, только лишь для демонстрации работоспособности. Для 100000 и более записей процедура выполнения занимает более получаса. Для примера, удалим все отсрочки, кроме трех последних «по свежести». Вызов процедуры осуществляется командой:

EXECUTE PROCEDURE DEL\_DELAY

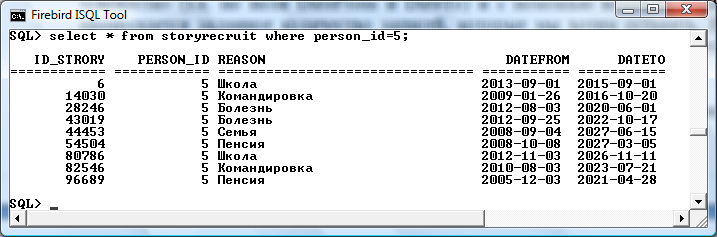
QUANT\_SAVE; // Количество записей, которое хотим // оставить.

Пример вызова процедуры:

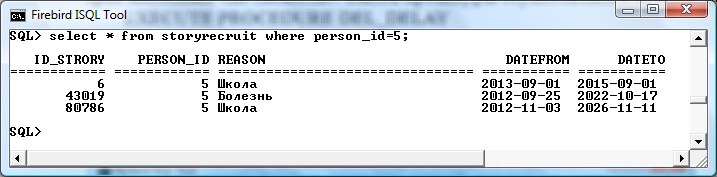
EXECUTE PROCEDURE DEL\_DELAY 3;

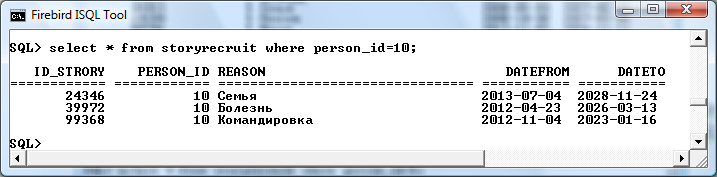
Результаты выполнения команд:

Например, посмотрим сколько отсрочек у любого человека в БД:



Видим, что он имеет 9 отсрочек. Выполим процедуру и посмотрим на результат (команда для такого количества записей выполнялась около 15 секунд:





**в) Для всех, кто проходил службу в заданной в/ч в заданное время создать пожизненную отсрочку:**

Алгоритм процедуры следующий:

Алгоритм подобен процедуре пункта а). В цикле, в таблице MilitaryService ищем по первичному ключу запись, которая соответствует заданной военной части. Извлекаем id человека из этой записи и потом в таблице RecState по этому ключу проверяем запись на наличие статуса «Служил». Затем в таблице MilitaryService проверяем эту запись на попадание в заданный временной интервал. При совпадении всех условий, мы идем в таблицу RecState и переводим статус данного человека на «Отсрочка» с текущего момента на 100 лет вперед. Затем идем в таблицу StoryRecruit и добавляем отсрочку данному человеку с текущего момента и на 100 лет вперед по причине «SERVICE».

Листинг ХП приведен в **приложении №4**.

Протестируем написанную процедуру на нашей БД, предварительно сгенерировав в таблицу ServiceMilitary 1000 записей (так мало для удобства представления результатов). Допустим, нам нужно дать пожизненную отсрочку всем, кто служим в части с ID=1 с момента 12.12.2000 по 12.12.2020. Вызов процедуры осуществляется командой:

EXECUTE PROCEDURE INF\_APPEAL

DIV\_R, // Идентификатор заданной военной части.

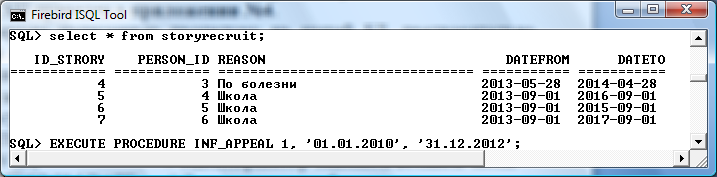
DATEFROM DATE, // Дата начала службы

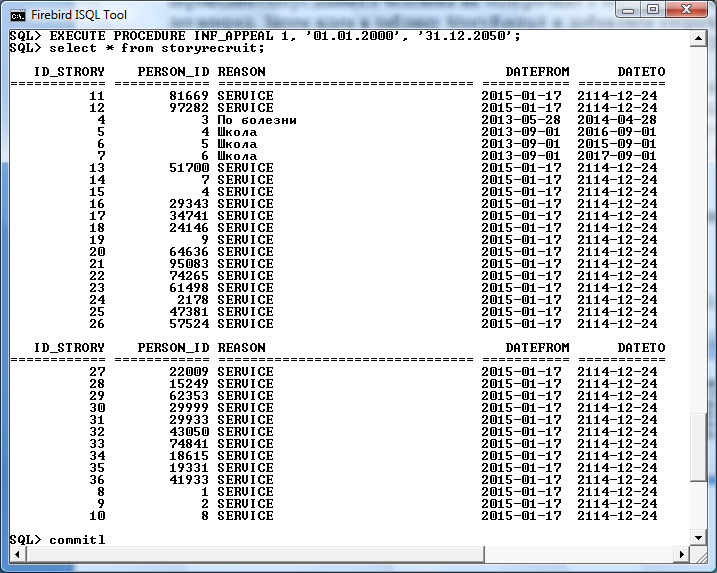
DATETO DATE, // Дата окончания службы

Пример вызова процедуры:

EXECUTE PROCEDURE INF\_APPEAL 1, ’01.12.2000’, ’31.12.2050’;

Результаты выполнения команд:



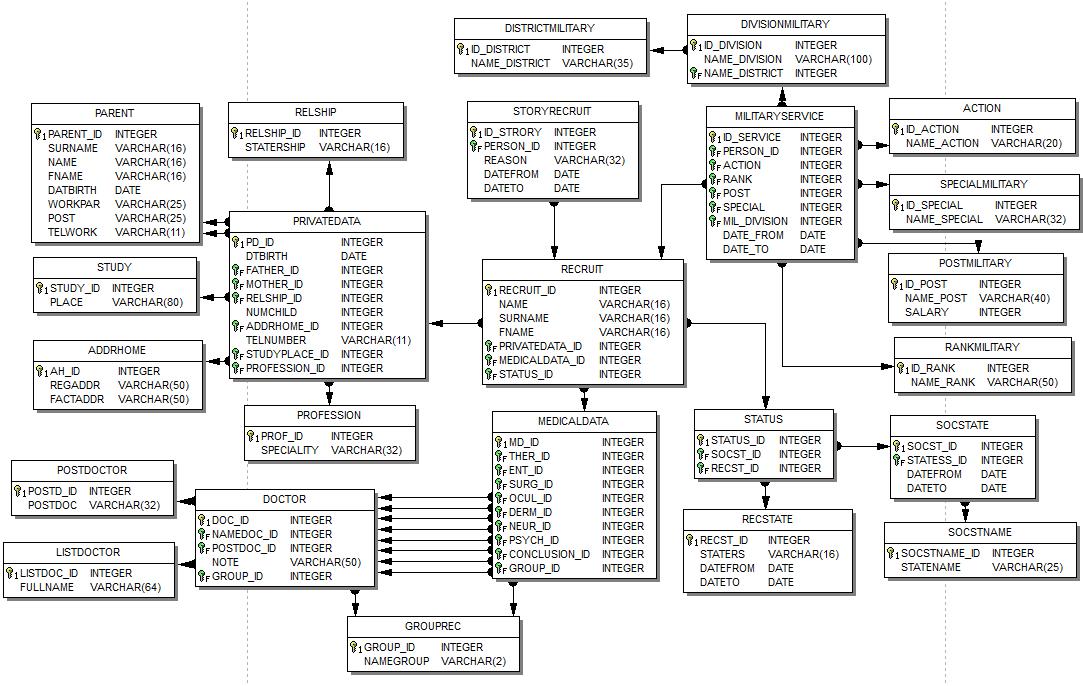


По сути, мы дали всем пожизненную отсрочку кто служил в военной части с ID=1.

1. **Вывод:**

Выполнив лабораторную работу №4, мы ознакомитлись с возможностями реализации более сложной обработки данных на стороне сервера с помощью хранимых процедур. Были изучены основы языка PSQL и созданы в соответствии с заданием необходимые процедуры. В ходе проверки на работоспобность написанных программ, испытания дали положительные предсказуемые результаты. Преимущество использования хранимых процедур заключается в том, что они позволяют установить те ограничения целостности, которые невозможно установить через описательные ограничения языка SQL. Кроме того, выполнение процедур на стороне сервера БД позволяет разгрузить канал между сервером приложений и сервером БД, что очень хорошо в случае, когда идет большой поток сложных запросов от пользователей.

**Приложение №1.** ER-диаграмма, полученная в IBExpert.



**Приложение №2**. **Хранимая процедура для создания призыва:**

SET TERM ^ ;

CREATE or ALTER PROCEDURE REC\_APPEAL (QUANT\_R INT, RANK\_R INT, POST\_R INT, SPECIAL\_R INT, DIV\_R INT, PERIOD\_DAY INT)

AS

declare variable TMP\_GROUP INT;

declare variable i INT;

declare variable i\_max INT;

declare variable id INT;

declare variable TMP\_STATERS INT;

declare variable TMP\_DATEFROM DATE;

declare variable TMP\_DATETO DATE;

declare variable CHECK\_REC INT;

BEGIN

i=0;

id=0;

SELECT MAX(medicaldata.md\_id) FROM medicaldata

INTO :i\_max;

SELECT MAX(militaryservice.id\_service) FROM militaryservice

INTO :id;

SELECT current\_date FROM rdb$database

INTO :TMP\_DATEFROM;

TMP\_DATETO=TMP\_DATEFROM+:PERIOD\_DAY;

WHILE(i\_max>=i)

DO

BEGIN

IF(QUANT\_R=0) THEN LEAVE;

i=i+1;

SELECT recruit.recruit\_id FROM recruit

WHERE recruit.recruit\_id=:i

INTO :CHECK\_REC;

IF(CHECK\_REC is not NULL) THEN

BEGIN

CHECK\_REC=NULL;

SELECT medicaldata.group\_id FROM medicaldata

WHERE medicaldata.md\_id =:i

INTO :TMP\_GROUP;

IF(TMP\_GROUP<7) THEN

BEGIN

SELECT recstate.staters\_id FROM recstate

WHERE recstate.recst\_id=:i

INTO :TMP\_STATERS;

IF(TMP\_STATERS=3) THEN

BEGIN

UPDATE recstate SET staters\_id=2 WHERE recst\_id= :i;

UPDATE recstate SET datefrom = :TMP\_DATEFROM where recst\_id= :i;

UPDATE recstate SET dateto = :TMP\_DATETO where recst\_id= :i;

id=id+1;

INSERT INTO militaryservice(id\_service, person\_id, action, rank, post, special, mil\_division, date\_from, date\_to)

VALUES (:id, :i, 1, :RANK\_R, :POST\_R, :SPECIAL\_R, :DIV\_R, :TMP\_DATEFROM, :TMP\_DATETO);

QUANT\_R=QUANT\_R-1;

END

END

END

END

END^

SET TERM ; ^

**Приложение №3. Хранимая процедура для удаления лишних отсрочек:**

SET TERM ^ ;

CREATE or ALTER PROCEDURE DEL\_DELAY (

QUANT\_SAVE INT)

AS

declare variable i INT;

declare variable i\_max INT;

begin

/\* Procedure Text \*/

i=0;

SELECT MAX(storyrecruit.person\_id) FROM storyrecruit

INTO :i\_max;

WHILE(i\_max>=i)

DO

BEGIN

i=i+1;

DELETE FROM storyrecruit WHERE id\_strory IN

( SELECT SKIP (:QUANT\_SAVE) id\_strory FROM storyrecruit WHERE person\_id = :i ORDER BY datefrom DESC, dateto DESC);

END

END^

SET TERM ; ^

**Приложение №4.** **Хранимая процедура для создания пожизненных отсрочек:**

SET TERM ^ ;

CREATE or ALTER PROCEDURE INF\_APPEAL (

DIV\_R INT,

DATEFROM DATE,

DATETO DATE

)

AS

declare variable i INT;

declare variable id\_max INT;

declare variable id INT;

declare variable TMP\_REC INT;

declare variable TMP\_STATERS INT;

declare variable TMP\_DATEFROM DATE;

declare variable TMP\_DATETO DATE;

declare variable TMP\_FROM DATE;

declare variable TMP\_TO DATE;

declare variable TMP\_DIVISION INT;

begin

/\* Procedure Text \*/

TMP\_STATERS=0;

i=0;

id=0;

SELECT MAX(storyrecruit.id\_strory) FROM storyrecruit

INTO :id;

SELECT MAX(militaryservice.id\_service) FROM militaryservice

INTO :id\_max;

SELECT current\_date FROM rdb$database

INTO :TMP\_DATEFROM;

TMP\_DATETO=TMP\_DATEFROM+36500;

WHILE(id\_max>=i)

DO

BEGIN

i=i+1;

SELECT militaryservice.mil\_division FROM militaryservice

WHERE militaryservice.id\_service =:i

INTO :TMP\_DIVISION;

SELECT militaryservice.person\_id FROM militaryservice

WHERE militaryservice.id\_service =:i

INTO :TMP\_REC;

IF(TMP\_DIVISION=DIV\_R) THEN

BEGIN

SELECT recstate.staters\_id FROM recstate

WHERE recstate.recst\_id=:TMP\_REC

INTO :TMP\_STATERS;

IF(TMP\_STATERS=1) THEN

BEGIN

SELECT militaryservice.date\_from FROM militaryservice

WHERE militaryservice.id\_service=:i

INTO :TMP\_FROM;

IF(DATEFROM<TMP\_FROM) THEN

BEGIN

SELECT militaryservice.date\_to FROM militaryservice

WHERE militaryservice.id\_service=:i

INTO :TMP\_TO;

IF(DATETO>TMP\_TO) THEN

BEGIN

UPDATE recstate SET staters\_id=3 WHERE recst\_id= :TMP\_REC;

UPDATE recstate SET datefrom = :TMP\_DATEFROM where recst\_id= :TMP\_REC;

UPDATE recstate SET dateto = :TMP\_DATETO where recst\_id= :TMP\_REC;

id=id+1;

INSERT INTO storyrecruit(id\_strory, person\_id, reason, datefrom, dateto)

VALUES (:id, :TMP\_REC, 'SERVICE', :TMP\_DATEFROM, :TMP\_DATETO);

END

END

END

END

END

END^

SET TERM ; ^