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

Отчет по лабораторной работе
Курс: «Базы данных»
Тема: «Хранимые процедуры»

Выполнил:
Бояркин Н.С. группа 43501/3
Проверил:
Мяснов А.В.

Санкт – Петербург
2017

1. Цель работы

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

2. Программа работы

1. Изучить возможности языка PSQL
2. Создать две хранимые процедуры в соответствии с индивидуальным заданием, полученным у преподавателя
3. Выложить скрипт с созданными сущностями в svn
4. Продемонстрировать результаты преподавателю

3. Индивидуальное задание

1. Сформировать скидки по клиентам (по некоторому произвольному алгоритму) на основе объема заказов клиентов за некоторое время.
2. На основе данных об остатках на складах и потока заказов сформировать набор новых поставок.

4. Ход работы

В первую очередь была изменена структура базы данных. Сделано это было по следующим причинам:

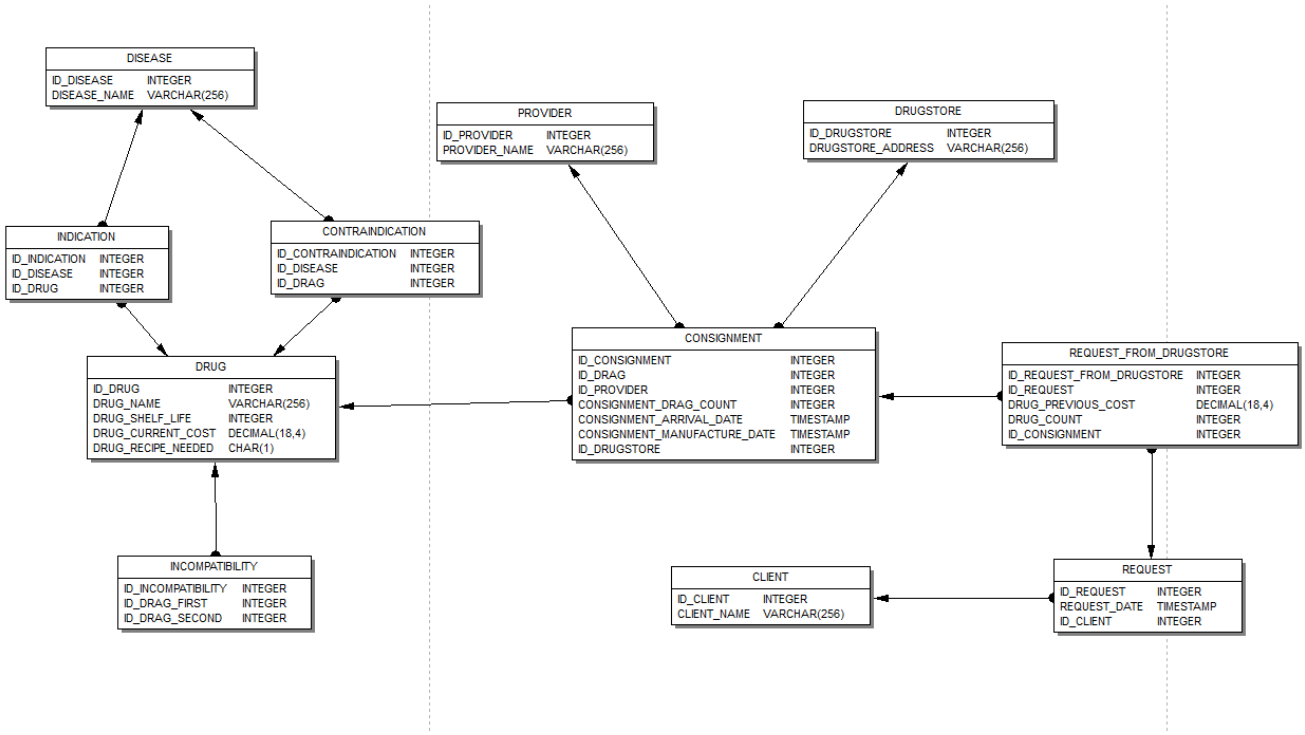
1. Не было никакой информации о клиенте.
2. Таблица REQUEST_FROM_DRUGSTORE должна быть связана с таблицей CONSIGNMENT, потому что каждая часть заказа должна быть взята из конкретной поставки.
3. Не нужна связь REQUEST_FROM_DRUGSTORE с DRUGSTORE, потому что эта связь уже учитывается (CONSIGNMENT + DRUGSTORE).

Был написан скрипт, который изменяет базу данных:

```
ALTER TABLE REQUEST_FROM_DRUGSTORE
DROP ID_DRUGSTORE;

COMMIT;

ALTER TABLE REQUEST_FROM_DRUGSTORE
ADD ID_CONSIGNMENT INTEGER NOT NULL;

COMMIT;

ALTER TABLE REQUEST_FROM_DRUGSTORE
ADD CONSTRAINT FK_REQUEST_FROM_DRUGSTORE
FOREIGN KEY (ID_CONSIGNMENT) REFERENCES CONSIGNMENT

COMMIT;

CREATE TABLE CLIENT
(
    ID_CLIENT       INTEGER         NOT NULL,
    CLIENT_NAME     VARCHAR(256)    NOT NULL,
   CONSTRAINT PK_CLIENT PRIMARY KEY (ID_CLIENT)
);

COMMIT;

ALTER TABLE REQUEST
ADD ID_CLIENT INTEGER NOT NULL;
```

```
COMMIT;

ALTER TABLE REQUEST
ADD CONSTRAINT FK_REQUEST
FOREIGN KEY (ID_CLIENT) REFERENCES CLIENT;

COMMIT;
```

Результирующая схема:



Скидки по клиентам на основе объема заказов

Процедура принимает идентификатор покупаемого лекарства и клиента. Если эти идентификаторы не содержатся в базе данных, то выбрасываются исключения. Далее, в зависимости от накопленной суммы всех заказов клиента ему предоставляется скидка (до 10000 – нет скидки, от 10000 до 50000 – 5%, от 50000 до 90000 – 10%, от 90000 – 15%). В результате возвращается значение цены лекарства для конкретного клиента с учетом скидки.

```
/* Exceptions which used by procedure. */

CREATE EXCEPTION CLIENT_IS_NOT_EXIST 'Client is not exist.';
CREATE EXCEPTION DRUG_IS_NOT_EXIST 'Drug is not exist.';


CREATE PROCEDURE GET_DRUG_COST_BY_CLIENT(ID_DRUG INTEGER NOT NULL, ID_CLIENT INTEGER NOT NULL)
    RETURNS(RESULT_DRUG_COST DECIMAL(18,4) NOT NULL)
AS
    DECLARE VARIABLE CHECK_NULL INTEGER;
    DECLARE VARIABLE COEFF DECIMAL(18,4) NOT NULL;
    DECLARE VARIABLE CLIENT_SUMM DECIMAL(18,4);
    DECLARE VARIABLE DRUG_COST DECIMAL(18,4);
BEGIN

    /* Throw exception if client is not exist. */

    CHECK_NULL = NULL;
    CHECK_NULL = ( SELECT CLIENT.ID_CLIENT FROM CLIENT
```

```
                    WHERE CLIENT.ID_CLIENT = :ID_CLIENT );

    IF (CHECK_NULL IS NULL) THEN
        EXCEPTION CLIENT_IS_NOT_EXIST;


    /* Throw exception if drug is not exist. */

    CHECK_NULL = NULL;
    CHECK_NULL = ( SELECT DRUG.ID_DRUG FROM DRUG
                   WHERE DRUG.ID_DRUG = :ID_DRUG );

    IF (CHECK_NULL IS NULL) THEN
        EXCEPTION DRUG_IS_NOT_EXIST;


    /* Getting all money, that client spent. */

    CLIENT_SUMM = NULL;

CLIENT_SUMM = ( SELECT SUM(REQUEST_FROM_DRUGSTORE.DRUG_PREVIOUS_COST * REQUEST_FROM_DRUGSTORE.DRU
G_COUNT) FROM REQUEST_FROM_DRUGSTORE
                    JOIN REQUEST ON REQUEST_FROM_DRUGSTORE.ID_REQUEST = REQUEST.ID_REQUEST
                    JOIN CLIENT ON REQUEST.ID_CLIENT = CLIENT.ID_CLIENT
                    WHERE CLIENT.ID_CLIENT = :ID_CLIENT );


    /* Calculate sale coefficient.  */

    IF (CLIENT_SUMM IS NULL) THEN
        COEFF = 1;
    ELSE IF(CLIENT_SUMM < 10000) THEN
        COEFF = 1;
    ELSE IF(CLIENT_SUMM < 50000) THEN
        COEFF = 0.95;
    ELSE IF(CLIENT_SUMM < 90000) THEN
        COEFF = 0.9;
    ELSE
        COEFF = 0.85;


    /* Calculate result drug cost.  */

    RESULT_DRUG_COST = ( SELECT (DRUG.DRUG_CURRENT_COST * :COEFF) FROM DRUG
                         WHERE DRUG.ID_DRUG = :ID_DRUG );

END
```

Формирование набора новых поставок

Процедура анализирует количество лекарств, поставленных на склад и количество лекарств, проданных заказчику. Если разница меньше 1000, то формируются новые заказы, которые дополняют результат до 1000. Поставщик и аптека по умолчанию выбираются с идентификатором 1, дата изготовления выбирается текущая, дата прибытия через 10 суток после изготовления. Эти параметры можно задать с помощью аргументов, как в предыдущей процедуре, но для разнообразия они вычисляются сами.

```
/* Exceptions which used by procedure. */

CREATE EXCEPTION PROVIDER_IS_NOT_EXIST 'Provider is not exist.';
CREATE EXCEPTION DRUGSTORE_IS_NOT_EXIST 'Drugstore is not exist.';


CREATE PROCEDURE CREATE_NEW_CONSIGMENTS()
AS
    DECLARE VARIABLE CURRENT_DRUG_ID INTEGER NOT NULL;
    DECLARE VARIABLE CONSIGNMENT_COUNT INTEGER;
```

```
    DECLARE VARIABLE REQUEST_COUNT INTEGER;

    DECLARE VARIABLE RESULT_COUNT INTEGER NOT NULL;
    DECLARE VARIABLE RESULT_MANUFACTURE_DATE TIMESTAMP NOT NULL;
    DECLARE VARIABLE RESULT_ARRIVAL_DATE TIMESTAMP NOT NULL;
    DECLARE VARIABLE PROVIDER_ID INTEGER;
    DECLARE VARIABLE DRUGSTORE_ID INTEGER;
BEGIN

     /* Throw exception if provider is not exist. */

    PROVIDER_ID = NULL;
    PROVIDER_ID = ( SELECT FIRST 1 PROVIDER.ID_PROVIDER FROM PROVIDER );

    IF (PROVIDER_ID IS NULL) THEN
        EXCEPTION PROVIDER_IS_NOT_EXIST;


    /* Throw exception if drugstore is not exist. */

    DRUGSTORE_ID = NULL;
    DRUGSTORE_ID = ( SELECT FIRST 1 DRUGSTORE.ID_DRUGSTORE FROM DRUGSTORE );

    IF (DRUGSTORE_ID IS NULL) THEN
        EXCEPTION DRUGSTORE_IS_NOT_EXIST;


    FOR
        /*
            Special table for loop:
                First column - drug_id.
                Second column - count of drugs in all consignments.
                Third column - count of drugs in all requests.
        */
        SELECT
            DRUG.ID_DRUG,
            SUM(BUFFER_CONSIGNMENT.CONSIGNMENT_COUNT) AS CONSIGNMENT_COUNT,
            SUM(BUFFER_REQUEST.REQUEST_COUNT) AS REQUEST_COUNT FROM DRUG
        LEFT JOIN (

  SELECT CONSIGNMENT.ID_DRAG AS ID_DRUG, CONSIGNMENT.ID_CONSIGNMENT AS ID_CONSIGNMENT, SUM(CONSIG
NMENT.CONSIGNMENT_DRAG_COUNT) AS CONSIGNMENT_COUNT
            FROM CONSIGNMENT
            GROUP BY CONSIGNMENT.ID_DRAG, CONSIGNMENT.ID_CONSIGNMENT
        ) BUFFER_CONSIGNMENT ON BUFFER_CONSIGNMENT.ID_DRUG = DRUG.ID_DRUG
        LEFT JOIN (

  SELECT REQUEST_FROM_DRUGSTORE.ID_CONSIGNMENT AS ID_CONSIGNMENT, SUM(REQUEST_FROM_DRUGSTORE.DRUG
_COUNT) AS REQUEST_COUNT
            FROM REQUEST_FROM_DRUGSTORE
            GROUP BY REQUEST_FROM_DRUGSTORE.ID_CONSIGNMENT
        ) AS BUFFER_REQUEST ON BUFFER_REQUEST.ID_CONSIGNMENT = BUFFER_CONSIGNMENT.ID_CONSIGNMENT
        GROUP BY DRUG.ID_DRUG
        ORDER BY DRUG.ID_DRUG
        INTO :CURRENT_DRUG_ID, :CONSIGNMENT_COUNT, :REQUEST_COUNT
    DO
    BEGIN

        /* Calculate result count of drugs (1000 is minimum). */

        IF(CONSIGNMENT_COUNT IS NULL) THEN
          RESULT_COUNT = 1000;
      ELSE IF(CONSIGNMENT_COUNT - REQUEST_COUNT < 1000) THEN
          RESULT_COUNT = 1000 - (CONSIGNMENT_COUNT - REQUEST_COUNT);
      ELSE
          CONTINUE;

      /* Get current time. */
```

```
      RESULT_MANUFACTURE_DATE = ( SELECT CURRENT_TIMESTAMP FROM RDB$DATABASE );
      RESULT_ARRIVAL_DATE = DATEADD(10 DAY TO RESULT_MANUFACTURE_DATE);

      /* Insert consignments into table. */


 INSERT INTO CONSIGNMENT (ID_CONSIGNMENT,  ID_DRAG, ID_PROVIDER, CONSIGNMENT_DRAG_COUNT, CONSIGNME
T_ARRIVAL_DATE, CONSIGNMENT_MANUFACTURE_DATE, ID_DRUGSTORE)
      VALUES (
          (SELECT (MAX(CONSIGNMENT.ID_CONSIGNMENT) + 1) FROM CONSIGNMENT),
          :CURRENT_DRUG_ID,
          :PROVIDER_ID,
          :RESULT_COUNT,
          :RESULT_ARRIVAL_DATE,
          :RESULT_MANUFACTURE_DATE,
          :DRUGSTORE_ID
      );

  END

END
```

5. Вывод

Хранимые процедуры позволяют сохранить часто используемые однотипные операции сложной выборки из базы данных.

Использование хранимых процедур позволяет повысить безопасность путем предоставления пользователю доступа только к ним, а не непосредственно к таблицам базы данных. Также стоит отметить уменьшение запросов к серверу, что позволяет экономить сетевой трафик и тратить дополнительное время на передачу по сети.

Из недостатков хранимых процедур стоит выделить отсутствие контроля целостности кода хранимой процедуры при изменении схемы базы данных.