**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

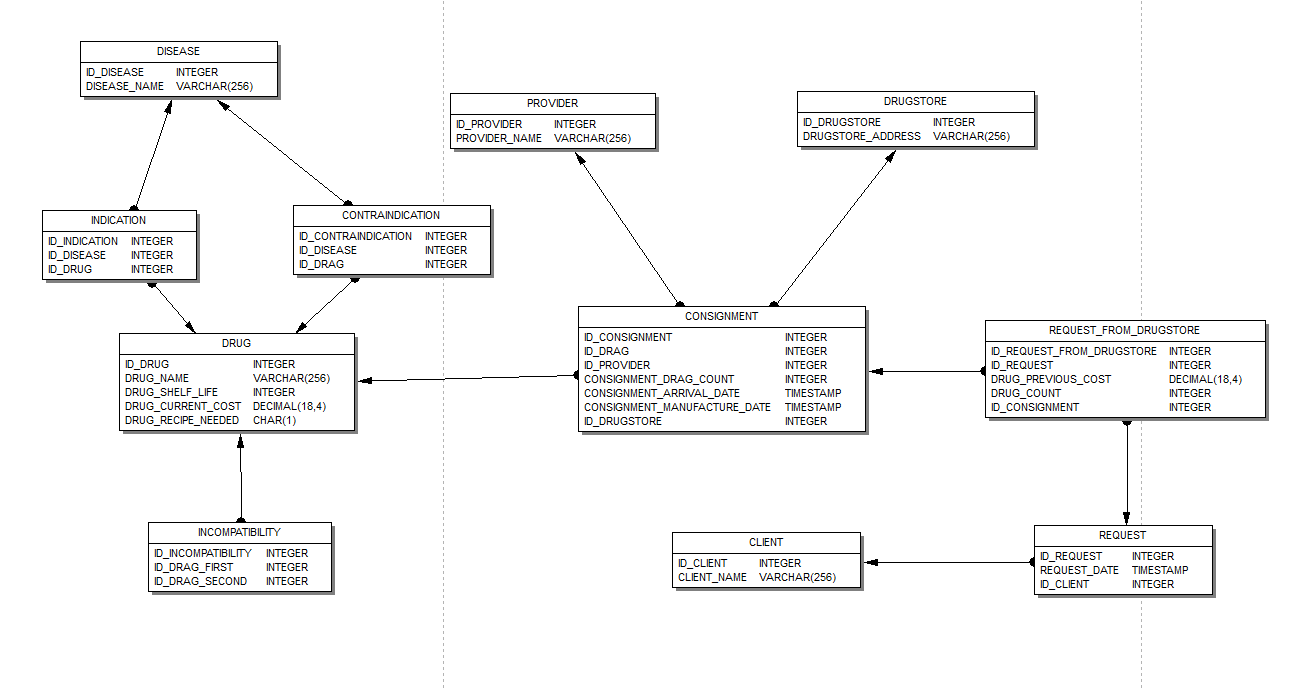
В первую очередь была изменена структура базы данных. Сделано это было по следующим причинам:

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.DRUG\_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**(CONSIGNMENT.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**   1. RESULT\_COUNT = 1000; 2. **ELSE** **IF**(CONSIGNMENT\_COUNT - REQUEST\_COUNT < 1000) **THEN** 3. RESULT\_COUNT = 1000 - (CONSIGNMENT\_COUNT - REQUEST\_COUNT); 4. **ELSE** 5. CONTINUE; 7. */\* Get current time. \*/* 9. RESULT\_MANUFACTURE\_DATE = ( **SELECT** **CURRENT\_TIMESTAMP** **FROM** RDB$DATABASE ); 10. RESULT\_ARRIVAL\_DATE = DATEADD(10 **DAY** **TO** RESULT\_MANUFACTURE\_DATE); 12. */\* Insert consignments into table. \*/* 14. **INSERT** **INTO** CONSIGNMENT (ID\_CONSIGNMENT,  ID\_DRAG, ID\_PROVIDER, CONSIGNMENT\_DRAG\_COUNT, CONSIGNMENT\_ARRIVAL\_DATE, CONSIGNMENT\_MANUFACTURE\_DATE, ID\_DRUGSTORE) 15. **VALUES** ( 16. (**SELECT** (**MAX**(CONSIGNMENT.ID\_CONSIGNMENT) + 1) **FROM** CONSIGNMENT), 17. :CURRENT\_DRUG\_ID, 18. :PROVIDER\_ID, 19. :RESULT\_COUNT, 20. :RESULT\_ARRIVAL\_DATE, 21. :RESULT\_MANUFACTURE\_DATE, 22. :DRUGSTORE\_ID 23. ); 25. **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.