**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: «Триггеры, вызовы процедур»

**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 и триггеров.

# 2. Программа работы

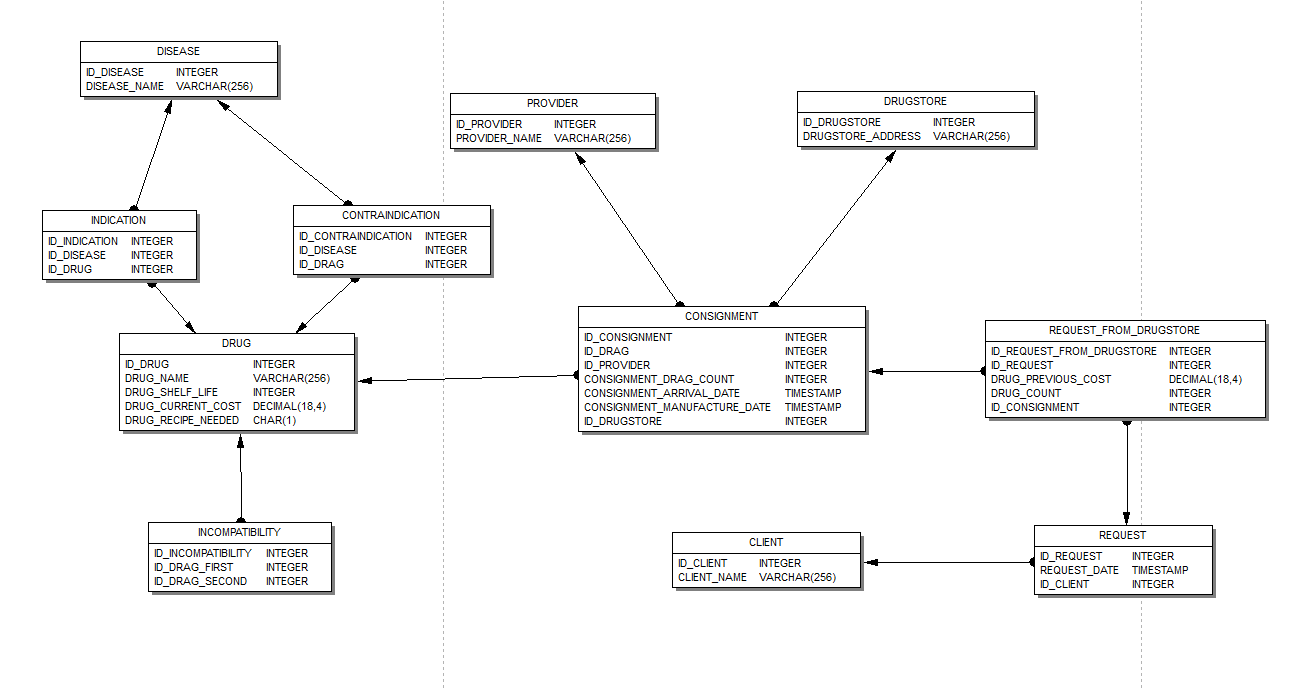
1. Создать два триггера: один триггер для автоматического заполнения ключевого поля, второй триггер для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице
2. Создать триггер в соответствии с индивидуальным заданием, полученным у преподавателя
3. Создать триггер в соответствии с индивидуальным заданием, вызывающий хранимую процедуру
4. Выложить скрипт с созданными сущностями в github
5. Продемонстрировать результаты преподавателю

# 3. Индивидуальное задание

1. При добавлении данных о несовместимости лекарств проверять дублирование данных. При обнаружении дубля - выбрасывать исключение.
2. При добавлении лекарства для диагноза проверять на несовместимость с другими лекарствами диагноза. Если есть несовместимость - не добавлять.

# 4. Ход работы

SQL-диаграмма базы данных:



**Триггер для автоматического заполнения ключевого поля**

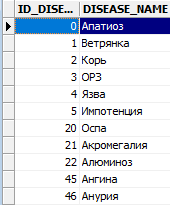
Для решения задачи автоматического заполнения ключевого поля таблицы болезней, был создан генератор DISEASE\_GENERATOR. Генератор определяет следующее значение для ключевого поля ID\_DISEASE:

|  |
| --- |
| **CREATE** **SEQUENCE** DISEASE\_GENERATOR;  **ALTER** **SEQUENCE** DISEASE\_GENERATOR RESTART **WITH** 5;    **CREATE** **TRIGGER** DISEASE\_AUTOINCREMENT **FOR** DISEASE      ACTIVE **BEFORE** **INSERT** **POSITION** 0  **AS**  **DECLARE** VARIABLE TEMP **BIGINT** **NOT** **NULL**;  **BEGIN**  **IF**(**NEW**.ID\_DISEASE **IS** **NULL**) **THEN**  **NEW**.ID\_DISEASE = GEN\_ID(DISEASE\_GENERATOR, 1);  **ELSE**  **BEGIN**          TEMP = GEN\_ID(DISEASE\_GENERATOR, 0);  **IF**(TEMP < **NEW**.ID\_DISEASE) **THEN**              TEMP = GEN\_ID(DISEASE\_GENERATOR, **NEW**.ID\_DISEASE - TEMP);  **END**    **END** |

Проверим корректность работы триггера:

|  |
| --- |
| **INSERT** **INTO** DISEASE(ID\_DISEASE, DISEASE\_NAME)  **VALUES**(20, 'Оспа');    COMMIT;    **INSERT** **INTO** DISEASE(DISEASE\_NAME)  **VALUES**('Акромегалия');    COMMIT;    **INSERT** **INTO** DISEASE(DISEASE\_NAME)  **VALUES**('Алюминоз');    COMMIT;    **INSERT** **INTO** DISEASE(ID\_DISEASE, DISEASE\_NAME)  **VALUES**(45, 'Ангина');    COMMIT;    **INSERT** **INTO** DISEASE(DISEASE\_NAME)  **VALUES**('Анурия');    COMMIT; |

Данные таблицы болезней до и после INSERT:



**Триггер для контроля целостности данных в подчиненной таблице**

При изменении данных о клиенте, проверяется не содержится ли он в таблице заказов, и если содержится, то выбрасывается соответствующее исключение:

|  |
| --- |
| **CREATE** EXCEPTION CLIENT\_CONTAINS\_IN\_OTHER\_TABLE 'It is impossible to modify client, besause he is contains in other table.';      **CREATE** **TRIGGER** CLIENT\_MODIFY **FOR** CLIENT  **BEFORE** **DELETE** **OR** **UPDATE**  **AS**  **BEGIN**  **IF**(**OLD**.ID\_CLIENT **IN** (**SELECT** REQUEST.ID\_CLIENT **FROM** REQUEST)) **THEN**          EXCEPTION CLIENT\_CONTAINS\_IN\_OTHER\_TABLE;  **END** |

Проверим корректность работы триггера:

|  |
| --- |
| **DELETE** **FROM** CLIENT  **WHERE** CLIENT.ID\_CLIENT = 3;    COMMIT; |

Было выведено исключение, потому что третий клиент содержится в таблице запросов:



**Проверка дублирования данных для таблицы несовместимости лекарств**

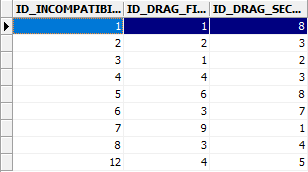
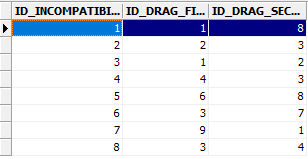
Если добавляемая пара лекарств уже содержится в таблице несовместимости, то выбрасывается соответствующее исключение, если в паре оба лекарства одинаковые также выбрасывается исключение:

|  |
| --- |
| **CREATE** EXCEPTION INCOMPATIBILITY\_ALREADY\_EXISTS 'Incompatibility is already exists.';  **CREATE** EXCEPTION INCOMPATIBILITY\_DRUGS\_THE\_SAME 'Drugs the same.';      **CREATE** **TRIGGER** INCOMPATIBILITY\_CHECK\_DOUBLE **FOR** INCOMPATIBILITY      ACTIVE **BEFORE** **INSERT**  **AS**  **DECLARE** VARIABLE CHECK\_NULL **INTEGER**;  **BEGIN**      CHECK\_NULL = **NULL**;      CHECK\_NULL = ( **SELECT** INCOMPATIBILITY.ID\_INCOMPATIBILITY **FROM** INCOMPATIBILITY  **WHERE** INCOMPATIBILITY.ID\_DRAG\_FIRST = **NEW**.ID\_DRAG\_FIRST **AND** INCOMPATIBILITY.ID\_DRAG\_SECOND = **NEW**.ID\_DRAG\_SECOND );    **IF**(**NOT** (CHECK\_NULL **IS** **NULL**)) **THEN**          EXCEPTION INCOMPATIBILITY\_ALREADY\_EXISTS;        CHECK\_NULL = **NULL**;      CHECK\_NULL = ( **SELECT** INCOMPATIBILITY.ID\_INCOMPATIBILITY **FROM** INCOMPATIBILITY  **WHERE** INCOMPATIBILITY.ID\_DRAG\_FIRST = **NEW**.ID\_DRAG\_SECOND **AND** INCOMPATIBILITY.ID\_DRAG\_SECOND = **NEW**.ID\_DRAG\_FIRST );    **IF**(**NOT** (CHECK\_NULL **IS** **NULL**)) **THEN**          EXCEPTION INCOMPATIBILITY\_ALREADY\_EXISTS;    **IF**(**NEW**.ID\_DRAG\_FIRST = **NEW**.ID\_DRAG\_SECOND) **THEN**          EXCEPTION INCOMPATIBILITY\_DRUGS\_THE\_SAME;  **END** |

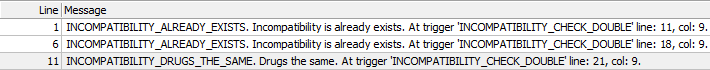
Проверим корректность работы триггера:

|  |
| --- |
| **INSERT** **INTO** INCOMPATIBILITY(ID\_INCOMPATIBILITY, ID\_DRAG\_FIRST, ID\_DRAG\_SECOND)  **VALUES** (9, 1, 8);    COMMIT;    **INSERT** **INTO** INCOMPATIBILITY(ID\_INCOMPATIBILITY, ID\_DRAG\_FIRST, ID\_DRAG\_SECOND)  **VALUES** (10, 8, 1);    COMMIT;    **INSERT** **INTO** INCOMPATIBILITY(ID\_INCOMPATIBILITY, ID\_DRAG\_FIRST, ID\_DRAG\_SECOND)  **VALUES** (11, 1, 1);    COMMIT;    **INSERT** **INTO** INCOMPATIBILITY(ID\_INCOMPATIBILITY, ID\_DRAG\_FIRST, ID\_DRAG\_SECOND)  **VALUES** (12, 4, 5);    COMMIT; |

Состояние таблицы несовместимости до и после INSERT:



Для первых трех операций INSERT были выброшены соответствующие исключения:



**Проверка на несовместимость с другими лекарствами**

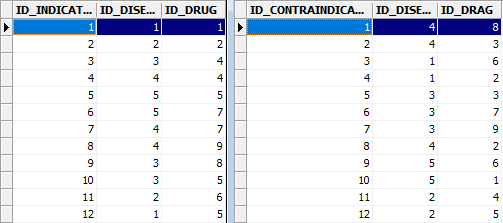
Была реализована проверка при попытке добавления в таблицу показаний лекарств для болезней. Очевидно, что одна и та же пара лекарство + болезнь не может содержаться одновременно и в таблице показаний, и в таблице противопоказаний. Также была реализована проверка на дублирование. Были созданы и вызваны две хранимые процедуры, которые выводят ключ таблиц показаний и противопоказаний по данным. Если данные в таблицах не найдены, то выводится NULL:

|  |
| --- |
| **CREATE** EXCEPTION INDICATION\_IS\_CONTRAINDICATION 'Indicalion contains into contraindication table.';  **CREATE** EXCEPTION INDICATION\_ALREADY\_EXISTS 'Indicalion is already exists.';        **CREATE** **PROCEDURE** GET\_INDICATION\_ID(ID\_DISEASE **INTEGER** **NOT** **NULL**, ID\_DRUG **INTEGER** **NOT** **NULL**)  **RETURNS**(INDICATION\_ID **INTEGER**)  **AS**  **BEGIN**      INDICATION\_ID = **NULL**;      INDICATION\_ID = ( **SELECT** INDICATION.ID\_INDICATION **FROM** INDICATION  **WHERE** INDICATION.ID\_DISEASE = :ID\_DISEASE **AND** INDICATION.ID\_DRUG = :ID\_DRUG );  **END**;        **CREATE** **PROCEDURE** GET\_CONTRAINDICATION\_ID(ID\_DISEASE **INTEGER** **NOT** **NULL**, ID\_DRUG **INTEGER** **NOT** **NULL**)  **RETURNS**(CONTRAINDICATION\_ID **INTEGER**)  **AS**  **BEGIN**      CONTRAINDICATION\_ID = **NULL**;      CONTRAINDICATION\_ID = ( **SELECT** CONTRAINDICATION.ID\_CONTRAINDICATION **FROM** CONTRAINDICATION  **WHERE** CONTRAINDICATION.ID\_DISEASE = :ID\_DISEASE **AND** CONTRAINDICATION.ID\_DRAG = :ID\_DRUG );  **END**;        **CREATE** **TRIGGER** INDICATION\_CHECK\_CONFLICT **FOR** INDICATION      ACTIVE **BEFORE** **INSERT**  **AS**  **DECLARE** VARIABLE CHECK\_NULL **INTEGER**;  **BEGIN**  **EXECUTE** **PROCEDURE** GET\_INDICATION\_ID  **NEW**.ID\_DISEASE, **NEW**.ID\_DRUG          RETURNING\_VALUES :CHECK\_NULL;    **IF**(**NOT** (CHECK\_NULL **IS** **NULL**)) **THEN**          EXCEPTION INDICATION\_ALREADY\_EXISTS;    **EXECUTE** **PROCEDURE** GET\_CONTRAINDICATION\_ID  **NEW**.ID\_DISEASE, **NEW**.ID\_DRUG          RETURNING\_VALUES :CHECK\_NULL;    **IF**(**NOT** (CHECK\_NULL **IS** **NULL**)) **THEN**          EXCEPTION INDICATION\_IS\_CONTRAINDICATION;  **END**; |

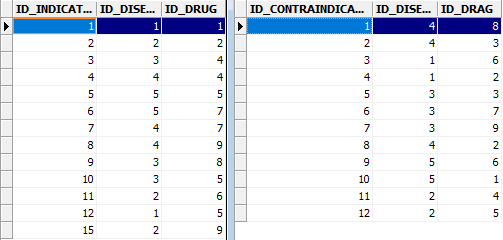
Проверим корректность работы триггера:

|  |
| --- |
| **INSERT** **INTO** INDICATION(ID\_INDICATION, ID\_DISEASE, ID\_DRUG)  **VALUES** (13, 4, 9);    COMMIT;    **INSERT** **INTO** INDICATION(ID\_INDICATION, ID\_DISEASE, ID\_DRUG)  **VALUES** (14, 3, 9);    COMMIT;    **INSERT** **INTO** INDICATION(ID\_INDICATION, ID\_DISEASE, ID\_DRUG)  **VALUES** (15, 2, 9);    COMMIT; |

Таблицы показаний и противопоказаний до INSERT:



Таблицы показаний и противопоказаний после INSERT:



Было добавлено только одно значение. Для всех остальных были выведены исключения:



# 5. Вывод

Триггер можно считать автоматической процедурой, срабатывающей на серверной стороне в результате некоторого события (insert, update, delete). Триггер может сработать до наступления события или после.

Главным преимуществом триггеров является контроль целостности базы данных любой сложности. Также упрощается логика приложения, так как часть логики выполняется на сервере.

Из недостатков триггеров можно выделить: уменьшение производительности системы при большом количестве триггеров, а также рекурсивную модификацию таблиц при неаккуратной реализации.