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

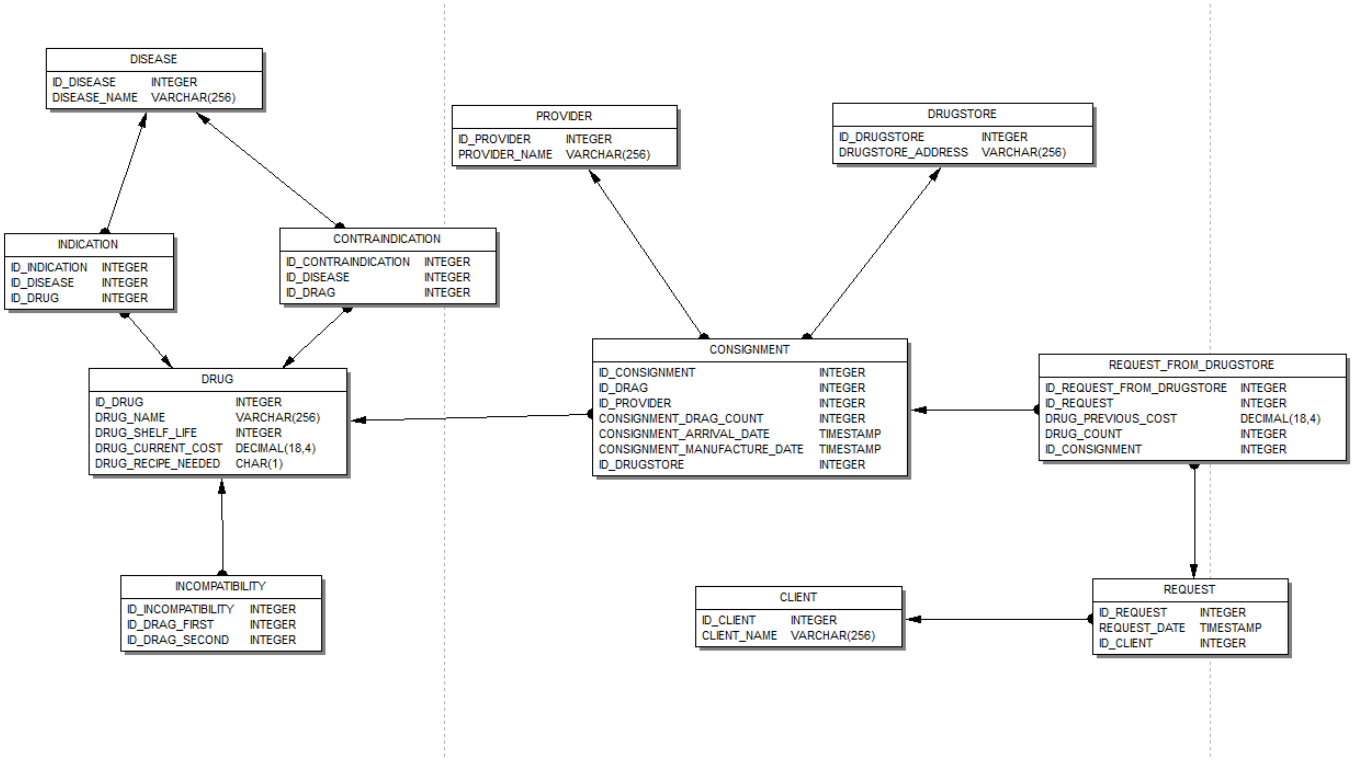
1. Создать два триггера: один триггер для автоматического заполнения ключевого поля, второй триггер для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице
2. Создать триггер в соответствии с индивидуальным заданием, полученным у преподавателя
3. Создать 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:

ID_DISE...	DISEASE_NAME
0	Апатиоз
1	Ветрянка
2	Корь
3	ОРЗ
4	Язва
5	Импотенция

ID_DISE...	DISEASE_NAME
0	Апатиоз
1	Ветрянка
2	Корь
3	ОРЗ
4	Язва
5	Импотенция
20	Оспа
21	Акромегалия
22	Алюминоз
45	Ангина
46	Анурия

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:

Line	Message
1	CLIENT_CONTAINS_IN_OTHER_TABLE. It is impossible to modify client, besause he is contains in other table. At trigger 'CLIENT_MODIFY' line: 6, col: 9.

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 = NE
W.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:

ID_INCOMPATIBI...	ID_DRAG_FI...	ID_DRAG_SEC...
1	1	8
2	2	3
3	1	2
4	4	3
5	6	8
6	3	7
7	9	1
8	3	4

ID_INCOMPATIBI...	ID_DRAG_FI...	ID_DRAG_SEC...
1	1	8
2	2	3
3	1	2
4	4	3
5	6	8
6	3	7
7	9	1
8	3	4
12	4	5

Для первых трех операций INSERT были выброшены соответствующие исключения:

Line	Message
1	INCOMPATIBILITY_ALREADY_EXISTS. Incompatibility is already exists. At trigger 'INCOMPATIBILITY_CHECK_DOUBLE' line: 11, col: 9.
6	INCOMPATIBILITY_ALREADY_EXISTS. Incompatibility is already exists. At trigger 'INCOMPATIBILITY_CHECK_DOUBLE' line: 18, col: 9.
11	INCOMPATIBILITY_DRUGS_THE_SAME. Drugs the same. At trigger 'INCOMPATIBILITY_CHECK_DOUBLE' line: 21, col: 9.

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:

ID_INDICAT...	ID_DISE...	ID_DRUG
1	1	1
2	2	2
3	3	4
4	4	4
5	5	5
6	5	7
7	4	7
8	4	9
9	3	8
10	3	5
11	2	6
12	1	5

ID_CONTRAINDICA...	ID_DISE...	ID_DRAG
1	4	8
2	4	3
3	1	6
4	1	2
5	3	3
6	3	7
7	3	9
8	4	2
9	5	6
10	5	1
11	2	4
12	2	5

Таблицы показаний и противопоказаний после INSERT:

ID_INDICAT...	ID_DISE...	ID_DRUG
1	1	1
2	2	2
3	3	4
4	4	4
5	5	5
6	5	7
7	4	7
8	4	9
9	3	8
10	3	5
11	2	6
12	1	5
15	2	9

ID_CONTRAINDICA...	ID_DISE...	ID_DRAG
1	4	8
2	4	3
3	1	6
4	1	2
5	3	3
6	3	7
7	3	9
8	4	2
9	5	6
10	5	1
11	2	4
12	2	5

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

Line	Message
1	INDICATION_ALREADY_EXISTS. Indicalion is already exists. At trigger 'INDICATION_CHECK_CONFLICT' line: 11, col: 9.
6	INDICATION_IS_CONTRAINDICATION. Indicalion contains into contraindication table. At trigger 'INDICATION_CHECK_CONFLICT' line: 18, col: 9.

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.