Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

# ЛАБОРАТОРНА РОБОТА № 1

з дисципліни «Організація баз даних та знань»

Тема: «Збережена процедура та Тригер»

Виконав:

студент 3 курсу

групи КС-32

Безрук Ю.Р.

Перевірила:

Лазурик В.М.

Харків – 2020

Целью данной работы является создание сохраненной процедуры (СП) и триггера.

При вызове СП выводит данные о наличии товара в нашей предметной области. Триггер валидирует входные данные о списании товаров таким образом, что бы невозможно было списать количество товара, не имеющегося на складе и списать его до прибытия. Также триггер проверяет списание конкретного существующего товара (запрещает ввод недопустимых кодов товаров).

Список объектов базы данных представлен на следующем скриншоте:

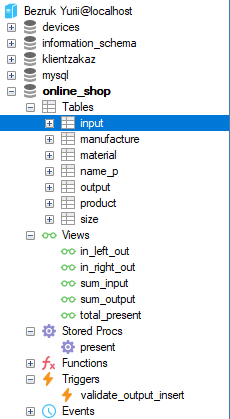


Рисунок - список объектов базы данных online\_shop

Таблицы input, manufacture, material, name\_p, output, product, size соответствуют данным в условии работы.

Для формирования общего наличия в сохраненной процедуре используются несколько вложенных представлений. Далее на скриншотах представлены их коды и результаты выполнения.

В запросах sum\_input и sum\_output считается сумма товара в наличии и проданного, та так же суммарная их стоимость и конкатенация наименования.



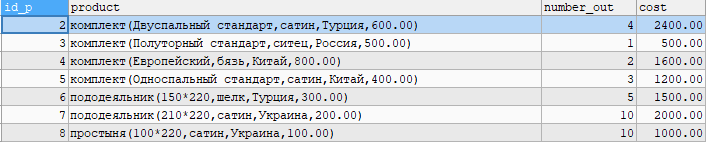
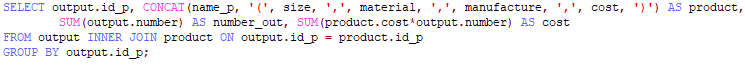
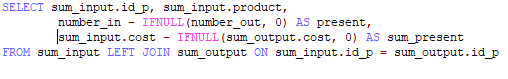
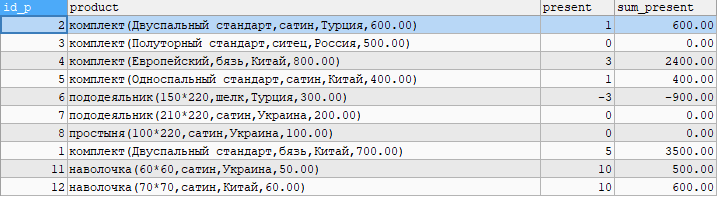


Рисунок - представления sum\_input и sum\_output и их результат

Далее по этим представлениям необходимо выполнить FULL JOIN для учёта всех возможных вариантов появления товаров. Для этого создаем два представления с левым и правым пересечением предыдущих представлений, где заодно вычисляем разность присутствующих и проданных приборов и их цен, in\_left\_out и in\_right\_out.





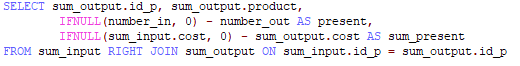
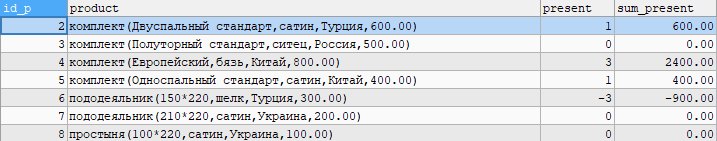
 

Рисунок 3 - представления sum\_in\_left\_sum\_out и sum\_in\_right\_sum\_out и их результат

После этого для выполнения полного внешнего пересечения делаем объединение предыдущих представлений в финальное представление total\_present.

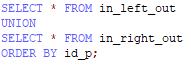


Рисунок - представление total\_present и его результат

Теперь из данного представления формируем сохраненную процедуру, которая и будет выводить текущее наличие товара.

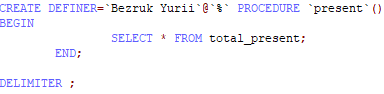
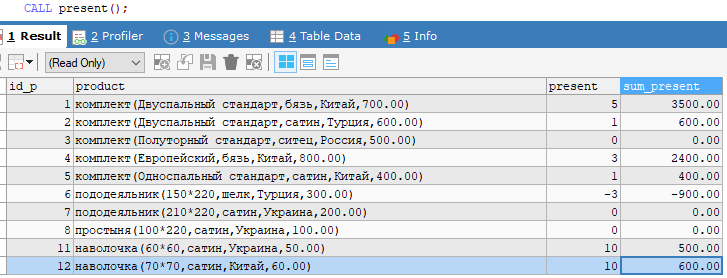
 

Рисунок - создание и вызов процедуры present()

Однако при таких входных данных вывод процедуры (как и данные в таблице) являются некорректными (присутствуют отрицательные числа, не все данные по датам в аутпут внесены правильно). Для того что бы решить эту проблему, создаем триггер, привязанный к вставке в таблицу output, который будет проверять валидность вводимых в нее данных. В триггере проверяем, что пользователь вводит существующий и неотрицательный код товара, что количество товара снимается не большее чем присутствует в магазине, и что дата снятия не предшествует дате поступления.

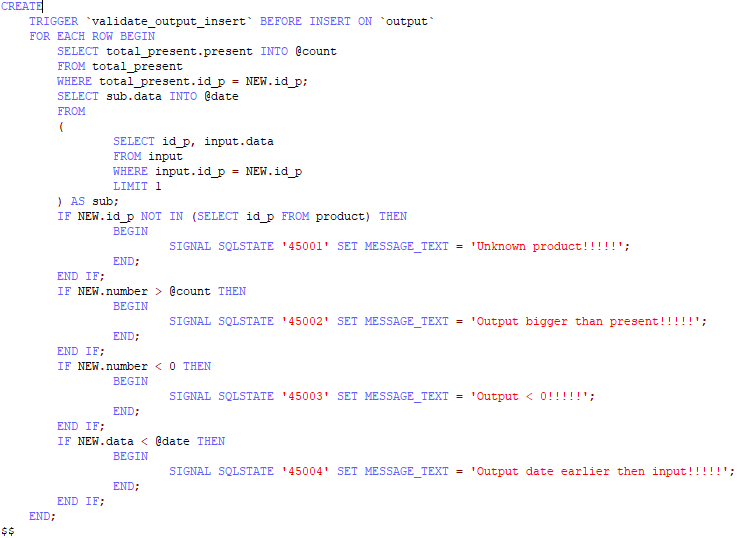


Рисунок - создание триггера validate\_output\_insert

Теперь выполняем TRUNCATE TABLE output, и пытаемся заново вставить исходные данные в таблицу output. Получаем ошибку:

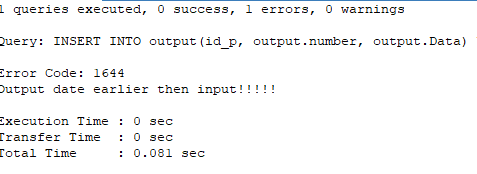


Рисунок - валидация в триггере не пропустила неправильные данные

С учётом валидации, правильные данные состояния указаны ниже:



Рисунок - правильные данные наличия товаров в магазине

Далее приведены коды представлений, процедуры и триггера в текстовом виде:

sum\_input:

SELECT input.id\_p, CONCAT(name\_p, '(', size, ',', material, ',', manufacture, ',', cost, ')') AS product,

SUM(input.number) AS number\_in, SUM(product.cost\*input.number) AS cost

FROM input INNER JOIN product ON input.id\_p = product.id\_p

GROUP BY input.id\_p;

sum\_output:

SELECT output.id\_p, CONCAT(name\_p, '(', size, ',', material, ',', manufacture, ',', cost, ')') AS product,

SUM(output.number) AS number\_out, SUM(product.cost\*output.number) AS cost

FROM output INNER JOIN product ON output.id\_p = product.id\_p

GROUP BY output.id\_p;

in\_left\_out:

SELECT sum\_input.id\_p, sum\_input.product,

number\_in - IFNULL(number\_out, 0) AS present,

sum\_input.cost - IFNULL(sum\_output.cost, 0) AS sum\_present

FROM sum\_input LEFT JOIN sum\_output ON sum\_input.id\_p = sum\_output.id\_p;

in\_right\_out:

SELECT sum\_output.id\_p, sum\_output.product,

IFNULL(number\_in, 0) - number\_out AS present,

IFNULL(sum\_input.cost, 0) - sum\_output.cost AS sum\_present

FROM sum\_input RIGHT JOIN sum\_output ON sum\_input.id\_p = sum\_output.id\_p;

total\_present:

SELECT \* FROM in\_left\_out

UNION

SELECT \* FROM in\_right\_out

ORDER BY id\_p;

процедура present():

CREATE DEFINER=`Bezruk Yurii`@`%` PROCEDURE `present`()

BEGIN

SELECT \* FROM total\_present;

END;

DELIMITER ;

Триггер validate\_output\_insert:

CREATE

TRIGGER `validate\_output\_insert` BEFORE INSERT ON `output`

FOR EACH ROW BEGIN

SELECT total\_present.present INTO @count

FROM total\_present

WHERE total\_present.id\_p = NEW.id\_p;

SELECT sub.data INTO @date

FROM

(

SELECT id\_p, input.data

FROM input

WHERE input.id\_p = NEW.id\_p

LIMIT 1

) AS sub;

IF NEW.id\_p NOT IN (SELECT id\_p FROM product) THEN

BEGIN

SIGNAL SQLSTATE '45001' SET MESSAGE\_TEXT = 'Unknown product!!!!!';

END;

END IF;

IF NEW.number > @count THEN

BEGIN

SIGNAL SQLSTATE '45002' SET MESSAGE\_TEXT = 'Output bigger than present!!!!!';

END;

END IF;

IF NEW.number < 0 THEN

BEGIN

SIGNAL SQLSTATE '45003' SET MESSAGE\_TEXT = 'Output < 0!!!!!';

END;

END IF;

IF NEW.data < @date THEN

BEGIN

SIGNAL SQLSTATE '45004' SET MESSAGE\_TEXT = 'Output date earlier then input!!!!!';

END;

END IF;

END;

$$

# ВЫВОДЫ

В процессе выполнения лабораторной работы была создана база данных online\_shop, созданы представления для получения данных о наличии товара, а так же предусмотрена сохраненная процедура для вывода текущего наличия товаров. Также был создан триггер, реализующий валидацию данных о наличии товаров на корректность ввода пользователем.