Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

**ОТЧЕТ по лабораторной работе №**15

по дисциплине «Базы данных»

**Выполнил:**

студент группы 5130902/20201 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. И. Сафонов

подпись

**Проверил:**

Кандидат тех. наук, доцент \_\_\_\_\_\_\_\_\_\_\_\_ С. А. Нестеров

подпись

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025г.

Санкт-Петербург, 2025

# ЛАБОРАТОРНАЯ РАБОТА № 1 (SQL SERVER) ШЕСТНАДЦАТОЙ НЕДЕЛИ КУРСА «УПРАВЛЕНИЕ ДАННЫМИ».

В Management Studio напишите код, проверяющий наличие временной таблицы #T и удаляющий ее, если она есть в базе данных.

Создайте временную таблицу #T со столбцами аналогичными столбцам Book и дополнительным целочисленным столбцом BookNum, допускающим значение NULL. Первичный ключ BookId – целочисленный, без автоматической генерации.

Заполните таблицу #T данными из таблицы Book.

Напишите функцию, принимающую в качестве входного параметра значение идентификатора BookID и возвращающую число экземпляров данной книги, упомянутых в таблице BookInLib. Обратите внимание, что если соответствующих записей в BookInLib не обнаружено, функция должна вернуть 0, а не NULL.

Используя написанную функцию, заполните значениями столбец BookNum в таблице #T. Создайте целочисленную переменную @MaxBookNum и присвойте ей максимальное значение, встречающееся в столбце BookNum таблицы #T. Командой Print выведите значение этой переменной.

Чтобы освежить в памяти материал предыдущей лекции, предложите и реализуйте пример, иллюстрирующий различия в области видимости локальной временной таблицы и табличной переменной.

Создайте хранимую процедуру, которая получает на вход название статуса книги и:

- если входной параметр (название статуса) имеет значение NULL, процедура заканчивает работу с кодом возврата -1;

- если такой статус есть в таблице BookStatus, процедура считает количество экземпляров книг с таким статусом в таблице BookInLib и возвращает его через выходной параметр; процедура завершается с кодом возврата 0;

- если такого статуса нет в таблице BookStatus, процедура добавляет такой статус в таблицу, возвращает 0 через выходной параметр и завершается с кодом возврата 1.

Проверьте работы процедуры при разных значениях входного параметра.

Для таблицы BookInLib создайте триггер на добавление и изменение данных, не позволяющий вносить в таблицу информацию о более чем 5 экземплярах одной книги. В случае попытки добавить 6-й и более экземпляр триггер, должен генерировать ошибку и откатывать транзакцию. Протестируйте работу триггера на разных комбинациях вводимых данных. Проверьте правильность работы триггера в случае, когда одним оператором insert вносится несколько записей.

В конспекте ко второму занятию этой недели рассмотрены DDLтриггеры. Создайте DDL-триггер на базу данных, не позволяющий удалять и изменять существующие таблицы и представления базы данных. Проверьте его работу. Убедитесь, что триггер блокирует только указанные выше действия. Попытайтесь удалить временную таблицу #T. Объясните полученный результат.

# ЛАБОРАТОРНАЯ РАБОТА (ORACLE) ШЕСТНАДЦАТОЙ НЕДЕЛИ КУРСА «УПРАВЛЕНИЕ ДАННЫМИ»

В данной лабораторной работе вы потренируетесь создавать функции и триггеры.

1. Подключитесь к базе как пользователь HR.
2. Создайте функцию get\_corr\_salary для корректировки переданного в качестве параметра значения зарплаты в зависимости от попадания в диапазон зарплат для некоторого идентификатора должности (job\_id).
   1. В качестве параметров функции должны быть переданы два входных (p\_sal и p\_jobid) и один выходной (p\_msg) – символьного типа.
   2. Объявите переменные v\_minsal и v\_maxsal того же типа, что и столбцы min\_salary и max\_salary таблицы jobs.
   3. В исполняемой секции присвойте значения переменным, получив информацию из таблицы jobs для переданного в качестве параметра идентификатора должности.
   4. Сравните переданное в качестве параметра значение зарплаты с v\_minsal и v\_maxsal:
      1. если оно попадает в заданный диапазон – верните его без изменений;
      2. если больше верхней границы – присвойте параметру p\_msg значение 'Зарплата была снижена до верхней границы диапазона зарплат должности !' и верните v\_maxsal;
      3. если меньше нижней – присвойте параметру p\_msg значение 'Зарплата была повышена до нижней границы диапазона зарплат должности !' и верните v\_minsal.
   5. Предусмотрите обработку ошибки NO\_DATA\_FOUND.
   6. Проверьте работу функции на разных значениях в анонимном PL/SQLблоке.
3. Создайте триггер audit\_emp\_trig, который при любой попытке изменения данных таблицы employees\_temp (вставка, обновление или удаление), будет вносить сведения об этом в специальную таблицу аудита.
   1. Создайте таблицу audit\_emp, содержащую три столбца:
      1. when – столбец типа DATE, в который будут записаны дата и время выполнения команды;
      2. what – символьный столбец, в который будет записано сообщение о том, какое действие (вставка, обновление или удаление было произведено).
   2. Напишите заголовок триггера, ответив на вопросы:
      1. в ответ на какую команду должен срабатывать триггер?
      2. когда он должен срабатывать: до или после выполнения команды?
      3. сколько раз он должен срабатывать: один или несколько – для каждой строки?
   3. В секции объявлений триггера объявите символьную переменную v\_comment и присвойте ей начальное значение 'обновление таблицы employees\_temp';
   4. В исполняемой секции триггера измените значение переменной v\_comment при необходимости (в зависимости от действия, которое вызвало срабатывание триггера). Используйте конструкцию вида:   
      IF { INSERTING | UPDATING | DELETING } THEN…   
      ELSIF { INSERTING | UPDATING | DELETING } THEN… …   
      Вставьте строку в таблицу audit\_emp, содержащую информацию о произошедшем событии.
4. Измените формат вывода даты в сессии с помощью команды: ALTER SESSION SET nls\_date\_format = 'dd.mm.yyyy hh24:mi:ss';
5. Проверьте работу триггера, выполнив любое изменение таблицы employees\_temp: убедитесь, что было выполнено как изменение таблицы employees\_temp, так и вставка строки в таблицу audit\_emp. Попробуйте откатить транзакцию и убедитесь в том, что были отменены изменения обеих таблиц. Проверьте, что при возникновении ошибки во время выполнения команды (например, при попытке задать слишком большое значение комиссионных или отрицательное значение зарплаты), вставка строк в таблицу аудита также не будет выполнена.
6. Создайте триггер corr\_sal\_trig, который будет проверять новое значение зарплаты при попытке её обновления в таблице employees\_temp и, при необходимости, корректировать:
   1. Напишите заголовок триггера, ответив на вопросы:
      1. в ответ на какую команду должен срабатывать триггер?
      2. когда он должен срабатывать: до или после выполнения команды?
      3. сколько раз он должен срабатывать: один или несколько – для каждой строки?
   2. В секции объявлений триггера объявите символьную переменную v\_msg, а также переменную v\_corr\_sal такого же типа, как столбец salary таблицы employees\_temp. Инициализируйте обе переменные с помощью функции get\_corr\_salary. Какое значение job\_id следует передать в качестве фактического значения параметра функции (старое или новое)?
   3. В исполняемой секции триггера при необходимости измените новое значение зарплаты и выведите сообщение v\_msg.
   4. Проверьте работу триггера:
      1. обновив зарплату какого-либо сотрудника, оставив её при этом в соответствующих его должности рамках;
      2. обновив должность и зарплату какого-либо сотрудника и указав зарплату, не попадающую в рамки, соответствующие новой должности.