Лабораторный практикум 10

Вариант 2

Тема: Проектирование активных баз данных на основе триггеров с использованием PL/SQL

1 Описание тестовой базы данных

Для выполнения заданий используется стандартная демо-схема БД Oracle – HUMAN RESOURCES (HR) из лабораторной работы 1.

Задания

Задание 1. Журналирование DML-операций

Разработать механизм журнализации *DML*-операций, выполняемых над таблицей с подразделениями, учитывая следующие действия:

- создать таблицу с именем *LOG_DEPARTMENTS*. Структура таблицы должна включать: имя пользователя, тип операции, дата выполнения операции, атрибуты, содержащие старые и новые значения.
- создать триггер журналирования.

Проверить работу триггера журналирования для операции *INSERT, UPDATE, DELETE*.

Задание 2. Автоматическая генерация целочисленных значений PK-колонок

В предыдущих лабораторных работах необходимо было создавать генераторы последовательностей (*SEQUENCE*) для *PK*-атрибутов таблиц. Известно, что в *Oracle 11XE* отсутствует возможность:

1) создавать генераторы, автоматически проставляя начальные значения с учетом содержимого таблиц БД, что требует от администратора вручную выполнять запросы на получение максимальных значений PK-атрибутов;

2) включать созданные генераторы в секцию *DEFAULT* описания колонок таблиц для автоматического внесения сгенерированного значения в колонку, что требует от программиста включать вручную функцию генерации *NEXTVAL* в *INSERT*-команды.

Задание 2.1 Учитывая 2-й недостаток управления генераторами, разработать триггер автоматического внесения сгенерированного значения в PK-колонки целочисленного типа таблицы *EMPLOYEES* и *DEPARTMENTS*.

Проверить работу триггера для операций *INSERT* в таблицы *EMPLOYEES* и *DEPARTMENTS.*

Задание 2.2 Учитывая 1-й недостаток управления генераторами, а также используя разработанный в 4-м задании 7-й лабораторной работы анонимный *PL/SQL*-блок по автоматическому созданию генераторов целочисленных значений PK-колонок, создать хранимую процедуру с учетом свойств:

- название процедуры – *CREATE_SEQUENCE*;
- входные параметры процедуры: имя таблицы, имя *PK*-колонки;
- в коде процедуры используется динамический запрос, содержащий *PL/SQL*-код триггера, пример которого создан в решении задания 1.1

Проверить работу процедуры для таблиц *EMPLOYEES* и *DEPARTMENTS.*

Задание 3. Обеспечение сложных правил ограничения целостности данных

Разработать сложное ограничение целостности по запрету устанавливать зарплаты сотрудникам, которые противоречат данным таблицы с учетом страны расположения подразделения и должности, на которой работает сотрудник.

Ограничение разработать с учетом следующих действий:

- создать таблицу *SALARIES* с ограничениями по зарплатам сотрудников, включающая: *COUNTRY_NAME, JOB_TITLE, SALARY_MIN, SALARY_MAX*. Заполнить таблицу двумя тестовыми записями;
- создать триггер по контролю указанного ограничения целостности.

Проверить работу триггера для операции *INSERT, UPDATE*.

Задание 4. Материализация представлений (виртуальных таблиц)

Сократить временя расчета средней зарплаты сотрудников в каждом подразделении на основе материализации соответствующего запроса.

4.1 Создать запрос на получение максимальной зарплаты сотрудников в каждом подразделении

4.2 Создать таблицу *MAX_DEPART_SALARY* по созданному ранее запросу.

4.3 Создать триггер по автоматическому согласованию содержимого этой таблицы и содержимого таблицы сотрудников.

Проверить работу триггера для операции *INSERT, UPDATE, DELETE*.

Задание 5. Автоматическая генерация строковых значений PK-колонок

Некоторые PK-колонки таблиц БД из лабораторной работы содержат строковые значения, что не позволяет создавать генераторы. Примером такой таблицы является таблица *JOBS* и ее колонка *JOI_ID.*

Разработать функцию генерации значения колонки *JOB_ID* , предполагая следующее:

- название функции *GET_JOB_ID*;
- входной параметр функции – строка со значением колонки *JOB_TITLE*;
- значение колонки *JOB_ID* получать из значения *JOB_TITLE* как объединение первых букв слов из значения колонки с разделителем «_», например, *Shipping Clerk = S_C*
- для получения в цикле первых букв слов рекомендуется использовать функции:
    - *INSTR(string, substring, start_position)*;
    - *SUBSTR(string, start_position, length)*;
- после формирования значения проверять его на уникальность в БД;
- если полученное значение не является уникальным, добавлять к нему цифру.

Разработать триггер автоматического внесения или изменения сгенерированного значения в PK-колонку *JOB_ID* строкового типа таблицы *JOBS*, использующий созданную процедуру *GET_JOB_ID*.

Проверить работу триггера для операции *INSERT, UPDATE*.

Задание 6. Генерация *PL/SQL*-кода журналирующих триггеров

Если в БД существует множество таблиц, которые необходимо журналировать, то такой процесс выполнения 1-го заданий может оказаться трудоемким.

Разработать функцию автоматического создания PL/SQL-кода журналирующих триггеров для заданной таблицы, предполагая следующее:

- название функции - *GENERATE_LOGGING*;
- входной параметр функции – название журналируемой таблицы;
- возвращаемое значение – строка PL/SQL-кода журналирующего триггера;
- список колонок журналируемой таблицы формируется на основании динамического запроса к таблице *USER_TAB_COLUMNS*;
- в функции создается *PL/SQL*-код журналирующего триггера, который выполняется через динамический запрос.

Проверить работу функции на примере таблицы *DEPARTMENTS,* сравнив полученный код с кодом из решения 1-го задания.

Требования к оформления отчета решений по лабораторной работе

Все команды оформить в виде файла-скрипта Фамилия_9.sql

По каждому заданию включите в файл:

1) условие задания (включите в виде комментариев)

2) PL/*SQL*-код