|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ | | | | | | | | |
|  | | |  | | |  | | |
| ФГБОУ ВО «Пермский государственный  национальный исследовательский университет» | | | | | | | | |
|  | | |  | | |  | | |
|  | | ОТЧЕТ  по лабораторной работе № 9 «Аудит» по дисциплине «Базы данных и СУБД» | | | | |  | |
|  | | |  | | |  | | |
|  | Работу выполнил  студент гр. \_\_\_\_\_\_\_\_\_  Зимин И. В.\_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2018 | | |  | Проверил  \_\_\_\_\_\_ кафедры МОВС  (доц., ст. преп., асс.)  Постаногов И. С.\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_ 2018 | | |  |
|  |  | | |  |  | | |  |
| Пермь 2018 | | | | | | | | |

1 Постановка задачи

Создать хранимую процедуру, которая для всех таблиц в базе данных создаёт таблицы-двойники. Таблица-двойник имеет те же самые поля с теми же самыми типами + поле 'дата и время модификации' + поле 'пользователь, внёсший изменения' + поле 'тип модификации (вставка, удаление или изменение)'. Для каждой исходной таблицы добавить триггер, который после вставки, удаления или изменения данных добавляет соответствующую запись с новыми значениями в таблицу-двойник. Для итерирования по таблицам требуется использовать курсоры.

Провести тестирование корректности работы программы.

Написать отчёт о проделанной работе.

P.S. Рассмотрите возможность использования значений по умолчанию для таблиц-двойников.

P.P.S. Структуру скрипта для сдачи работы на занятии разработайте самостоятельно.

P.P.P.S. Логин исполняющего запрос пользователя возвращается функцией SUSER\_SNAME()

2 Описание алгоритма работы программы

Как уже ранее было отмечено, **хранимая процедура** — объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

Мы имеем БД с некоторым количеством таблиц. Решая выше поставленную задачу, я придерживался следующего алгоритма:

1. Возьмем из БД первую таблицу и скопируем ее схему вместе с содержимым в новую таблицу с именем, равным имени текущей таблицы плюс ‘twin’.
2. Добавим в новую таблицу-дубликат следующие поля:
   1. Пользователь, внесший изменения. Значением по умолчанию будет являться текущий пользователь, работающий с БД.
   2. Дата внесения изменения. Значением по умолчанию будет являться текущая дата.
   3. Тип модификации. Не будет иметь значение по умолчанию

Схема таблицы дубликата примет следующий вид:

|  |  |  |
| --- | --- | --- |
| ... Поля текущей таблицы … | | |
| User | nvarchar(50) | NOT NULL |
| Date | datetime | NOT NULL |
| Type | nvarchar(50) | NOT NULL |

1. Создадим триггер, который после вставки в текущую таблицу, добавляет новую строчку в таблицу-дубликат, которая будет точной копией вставленной строки + поля, указанные в пункте 2. Причем, поля, а и b будут заполнены значениями по умолчанию, а поле c будет содержать выражение «INSERT», что будет означать что была совершена вставка.
2. Аналогичным образом создадим триггеры, который после обновления или удаления строки в текущей таблице, добавляет новую строчку в таблицу-дубликат. Типы модификации в них будут «UPDATE» и «DELETE» соответственно.
3. Перейдем к рассмотрению следующей таблицы и повторим все вышеперечисленные шаги.

3 Особенности реализации на SQL

Для выборки данных о таблицах БД воспользуемся представлением INFORMATION\_SCHEMA.TABLES, о котором ранее уже говорилось.

В данном представлении нас интересуют имена таблиц, их мы поместим в курсор. Но что же представляет из себя курсор?

Курсор — объект [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), который позволяет приложениям работать с записями «по одной», а не сразу с множеством, как это делается в обычных SQL командах.

Синтаксис работы курсора выглядит следующим образом:

Объявление курсора:

DECLARE @имя\_курсора CURSOR

FOR SELECT элементы\_выобрки FROM откуда\_происходит\_выборка

WHERE условие\_на\_выборку

Открытие курсора:

OPEN @имя\_курсора

Осуществить выбокру из курсора:

FETCH NEXT FROM @имя\_курсора INTO @куда\_выбрать

Пока что то в курсоре есть:

WHILE @@FETCH\_STATUS = 0

BEGIN

…

FETCH NEXT FROM @имя\_курсора INTO @куда\_выбрать

END

Закрытие курсора:

CLOSE @имя\_курсора

В даннам случае нам понадобится статический курсор:

SET @cursorForTable = CURSOR STATIC FOR SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES

Это связано с тем, что внутри данные в INFORMATION\_SCHEMA.TABLES будут меняться и поэтому нам нужно чтобы в курсоре хранилась копия этих данных. STATIC помогает нам это обеспечить.

Копию текущей таблицы будем создавать при помощи инструкции SELECT \* INTO, которая создает новую таблицу и заполняет ее результирующим набором инструкций.

SET @query = 'SELECT \* INTO [' + @nameNewTable + '] FROM ' +@nameCurTable

EXEC sp\_executesql @query

Запрос сделан динамическим в силу различных названий таблиц, содержащихся в предложении FROM. sp\_executesql – ХП, которая выполняет запрос из строки

Добавление новых полей в созданную таблицу будем осуществлять так же при помощи динамического запроса.

SET @query = 'ALTER TABLE [' + @nameNewTable + '] ADD [user] nvarchar(50) NOT NULL CONSTRAINT [DF\_'+@nameNewTable+'\_USER] DEFAULT (SUSER\_NAME())'

EXEC sp\_executesql @query

Переводя на русский язык, получится что мы после ALTER TABLE указываем имя таблицы, которую хотим изменить, затем после ключевого слова ADD пишем название нового столбца его тип и возможность принимать значение NULL. Дальше при помощи ключевого слова CONSTRAINT мы «навешиваем» на столбец значение по умолчанию DEFAULT, оно является результатом выполнения функции SUSER\_NAME (), которая возвращает идентификационное имя учетной записи текущего пользователя.

Аналогичным образом происходит добавление в таблицу еще двух полей: «Дата внесения изменения», где так же используется значение по умолчанию, получаемое при помощи функции GETDATE (), которая возвращает текущую дату.

SET @query = 'ALTER TABLE [' + @nameNewTable + '] ADD [date] datetime NOT NULL CONSTRAINT [DF\_'+@nameNewTable+'\_DATE] DEFAULT (getdate())'

EXEC sp\_executesql @query

И поля «Тип модификации», которое не имеет значение по умолчанию.

SET @query = 'ALTER TABLE [' + @nameNewTable + '] ADD [type] nvarchar(50) NOT NULL '

EXEC sp\_executesql @query

Триггеры после вставки/обновления/удаления будем создавать так же при помощи динамического запроса:

SET @query = 'CREATE TRIGGER [INSERTtrig'+ @nameNewTable +'] ON '+ QUOTENAME(@nameCurTable) +'

AFTER INSERT AS

BEGIN

INSERT INTO ['+@nameNewTable+']('+@columnsStr + 'type) SELECT ' +@columnsStr + '''INSERT'' FROM inserted

END'

EXEC sp\_executesql @query

Это триггер для текущей рассматриваемой таблицы, отрабатывающий после вставки. Здесь используется таблица inserted для получения вставленных данных. А в переменной @columnsStr содержится весь список столбцов текущей таблицы, разделенных запятой. Его мы получили до копирования, так же при помощи курсора (но уже не статического) и представления INFORMATION\_SCHEMA.COLUMNS из которого были выбраны все имена столбцов текущей таблицы. Остальные триггеры для обновления/удаления создаются аналогичным образом.

Тестирование программы

Для тестирования нам потребуется.

* Созданная БД
* Две таблицы в БД

Books(id\_book\*, Author’, name)

Readers’(id\_reader\*, FIO, Address)

* Созданная ХП, которая создает таблицы-дубликаты

**Тест №1. Вставка одной строки в первую таблицу**

* Суть теста: проверить, что таблицы-дубликаты были созданы и добавился триггер, верно отрабатывающий после вставки
* Последовательность шагов тестировании:
  + Вставить в таблицу Books запись
  + Вывести все исходные таблицы и таблицы-дубликаты на экран
* Входные данные: (1, “Пушкин”, “Онегин”)
* Ожидаемый результат:

|  |  |  |
| --- | --- | --- |
| id\_book | Author’ | name |
| 1 | Пушкин | Онегин |

|  |  |  |
| --- | --- | --- |
| id\_reader | FIO | Address |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id\_book | Author’ | name | User | Date | Type |
| 1 | Пушкин | Онегин | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.537 | INSERT |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id\_reader | FIO | Address | User | Date | Type |

* Фактический результат: в точности повторяет ожидаемый

**Тест №2. Вставка нескольких строк во вторую таблицу**

* Суть теста: проверить, что триггеры, отрабатывающие после вставки, были созданы не только для первой таблицы и отрабатывают верно при вставке сразу нескольких строк
* Последовательность шагов тестировании:
  + Вставить в таблицу Readers’ две записи
  + Вывести все исходные таблицы и таблицы-дубликаты на экран
* Входные данные: (1, “Коняев”, “ ул. Домовая, 15”) и

(2, “Цой”, “ ул. Попова, 15”)

* Ожидаемый результат:

|  |  |  |
| --- | --- | --- |
| id\_book | Author’ | name |
| 1 | Пушкин | Онегин |

|  |  |  |
| --- | --- | --- |
| id\_reader | FIO | Address |
| 1 | Коняев | ул. Домовая, 15 |
| 2 | Цой | ул. Попова, 15 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id\_book | Author’ | name | User | Date | Type |
| 1 | Пушкин | Онегин | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.537 | INSERT |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id\_reader | FIO | Address | User | Date | Type |
| 1 | Коняев | ул. Домовая, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.670 | INSERT |
| 2 | Цой | ул. Попова, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.670 | INSERT |

* Фактический результат: в точности повторяет ожидаемый

**Тест №3. Обновление одной записи**

* Суть теста: проверить, что триггеры, отрабатывающие после обновления, были созданы и отрабатывают верно при обновлении одной строки
* Последовательность шагов тестировании:
  + Обновить в таблице Books одну запись
  + Вывести все исходные таблицы и таблицы-дубликаты на экран
* Входные данные: у книги с id = 1 изменим название на «Евгений»
* Ожидаемый результат:

|  |  |  |
| --- | --- | --- |
| id\_book | Author’ | name |
| 1 | Пушкин | Евгений |

|  |  |  |
| --- | --- | --- |
| id\_reader | FIO | Address |
| 1 | Коняев | ул. Домовая, 15 |
| 2 | Цой | ул. Попова, 15 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id\_book | Author’ | name | User | Date | Type |
| 1 | Пушкин | Онегин | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.537 | INSERT |
| 1 | Пушкин | Евгений | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.693 | UPDATE |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id\_reader | FIO | Address | User | Date | Type |
| 1 | Коняев | ул. Домовая, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.670 | INSERT |
| 2 | Цой | ул. Попова, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.670 | INSERT |

* Фактический результат: в точности повторяет ожидаемый

**Тест №4. Обновление нескольких записей**

* Суть теста: проверить, что триггеры, отрабатывающие после обновления, были созданы не только для первой таблицы и отрабатывают верно при обновлении сразу нескольких записей.
* Последовательность шагов тестировании:
  + Обновить в таблице Readers’ все записи
  + Вывести все исходные таблицы и таблицы-дубликаты на экран
* Входные данные: у всем читателям сделать одинаковые ФИО «Читатель»
* Ожидаемый результат:

|  |  |  |
| --- | --- | --- |
| id\_book | Author’ | name |
| 1 | Пушкин | Евгений |

|  |  |  |
| --- | --- | --- |
| id\_reader | FIO | Address |
| 1 | Читатель | ул. Домовая, 15 |
| 2 | Читатель | ул. Попова, 15 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id\_book | Author’ | name | User | Date | Type |
| 1 | Пушкин | Онегин | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.537 | INSERT |
| 1 | Пушкин | Евгений | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.693 | UPDATE |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id\_reader | FIO | Address | User | Date | Type |
| 1 | Коняев | ул. Домовая, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.670 | INSERT |
| 2 | Цой | ул. Попова, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.670 | INSERT |
| 1 | Читатель | ул. Домовая, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.700 | UPDATE |
| 2 | Читатель | ул. Попова, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.700 | UPDATE |

* Фактический результат: в точности повторяет ожидаемый

**Тест №5. Удаление одной строки**

* Суть теста: проверить, что триггеры, отрабатывающие после удаления, были созданы и отрабатывают верно при удалении одной записи.
* Последовательность шагов тестировании:
  + Удалить из таблицы Books одну строку
  + Вывести все исходные таблицы и таблицы-дубликаты на экран
* Входные данные: удалим книгу с id = 1
* Ожидаемый результат:

|  |  |  |
| --- | --- | --- |
| id\_book | Author’ | name |

|  |  |  |
| --- | --- | --- |
| id\_reader | FIO | Address |
| 1 | Читатель | ул. Домовая, 15 |
| 2 | Читатель | ул. Попова, 15 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id\_book | Author’ | name | User | Date | Type |
| 1 | Пушкин | Онегин | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.537 | INSERT |
| 1 | Пушкин | Евгений | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.693 | UPDATE |
| 1 | Пушкин | Евгений | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.710 | DELETE |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id\_reader | FIO | Address | User | Date | Type |
| 1 | Коняев | ул. Домовая, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.670 | INSERT |
| 2 | Цой | ул. Попова, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.670 | INSERT |
| 1 | Читатель | ул. Домовая, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.700 | UPDATE |
| 2 | Читатель | ул. Попова, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.700 | UPDATE |

* Фактический результат: в точности повторяет ожидаемый

**Тест №6. Удаление нескольких записей**

* Суть теста: проверить, что триггер, отрабатывающий после удаления, был создан не только для первой таблицы и отрабатывают верно при удалении сразу нескольких записей.
* Последовательность шагов тестировании:
  + Удалить из таблицы Readers’ все записи
  + Вывести все исходные таблицы и таблицы-дубликаты на экран
* Входные данные: Удалим данные о всех читателях
* Ожидаемый результат:

|  |  |  |
| --- | --- | --- |
| id\_book | Author’ | name |

|  |  |  |
| --- | --- | --- |
| id\_reader | FIO | Address |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id\_book | Author’ | name | User | Date | Type |
| 1 | Пушкин | Онегин | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.537 | INSERT |
| 1 | Пушкин | Евгений | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.693 | UPDATE |
| 1 | Пушкин | Евгений | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.710 | DELETE |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id\_reader | FIO | Address | User | Date | Type |
| 1 | Коняев | ул. Домовая, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.670 | INSERT |
| 2 | Цой | ул. Попова, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.670 | INSERT |
| 1 | Читатель | ул. Домовая, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.700 | UPDATE |
| 2 | Читатель | ул. Попова, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.700 | UPDATE |
| 1 | Читатель | ул. Домовая, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.713 | DELETE |
| 2 | Читатель | ул. Попова, 15 | LAPTOP-6I6G07J5\User | 2018-04-10 08:28:45.713 | DELETE |

* Фактический результат: в точности повторяет ожидаемый