Лабораторная работа №2. Создание таблиц базы данных и использование умолчаний, правил, ограничений.

Цель работы: усвоить способы создания вода, удаления, редактирования данных в таблицах БД MS SQL Server.

Оборудование, технические и инструментальные средства: ПК, MS SQL Server, пакет программ MS Office.

Краткие теоретические сведения

Добавление данных при помощи SQL-команды INSERT

Первый метод вставки данных заключается в использовании SQL-команды INSERT в Query Editor.

1. Запустите Query Editor.

2. Найдите в окне Object Explorer узел нужной вам таблицы и выделите его.

3. Щелкните правой кнопкой мыши и выберите в контекстном меню команду Script Table As, за которой скрываете» вложенное меню, в котором следует выбрать Insert to, а затем New Query Editor Windows.

4. В результате вы увидите фрагмент кода, который не помещается в окне (обратите внимание, в окне Query Editor отсутствует перенос по словам). Вот готовый к правке код, который сгенерирован в панели запросов окна Query Editor:

INSERT INTO [имя_базы_данных].[dbo].[имя_таблицы] ([имя_атр1], [имя_атр2], [имя_атр3], [имя_атр4], . . .)
VALUES (знач_атр1, знач_атр2, знач_атр3, знач_атр4, . . .)

5. Отредактировав код в панели запросов Query Editor, вы сможете убедиться, что следующая секция кода действительно добавляет данные в таблицу.

6. Выполните код, нажав *F5 или Ctrl+E* или щелкните на кнопке Execute (ВЫПОЛНИТЬ) на панели инструментов.

В операторе вставки указывается каждый столбец таблицы.

Один из способов избежать необходимости в том, чтобы вводить при вставке записи данные в каждый столбец, заключается в разрешении значений NULL для столбцов. Для этого требуется лишь установить флажок Allow Nulls (Разрешить NULL-значение). Существует главное правило, касающееся значений NULL, и заключается оно в том, что первичный ключ не может содержать в себе значений NULL.

Другой способ избежать необходимости по вводе данных в каждый столбец таблицы заключается и использовании значений по умолчанию.

Сущность этого механизма состоит в том, что для столбца определяется значение, которое он получит по умолчанию при добавлении новой строки в таблицу, причем в запросе на вставку данных это никак не упоминается. Значения по умолчанию используют в случаях, когда требуется выполнить множество операций INSERT для столбца, значения в котором при этом не меняются.

Если вы попытаетесь добавить в таблицу данные, которые нарушают наложенные условия, в этом случае запись не будет вставлена в таблицу.

Если команда INSERT попытается ввести данные в identity-столбец, также произойдет ошибка.

Добавление данных при помощи Management Studio

1. Откройте окно Management Studio.

2. Разверните узел нужной базы данных, затем узел с таблицами таблицей и выделите элемент нужной таблицы. Щелкните на нем правой кнопкой мыши и выберите в меню команду Edit Top 200 rows

3. В результате откроется таблица. Сетка этой таблицы должна отображать строки данных, содержащиеся в таблице, но, поскольку данных еще нет, сетка пуста и готова к добавлению записей (строк). Обратите внимание на черную стрелку-треугольник слева. Это маркер текущей записи, который обозначает запись, на которую направлены действия пользователя. В данном случае, этот маркер не уместен, но он станет полезным, когда в таблице будет содержаться хотя бы несколько записей.

4. Эта операция сводится к тому, чтобы ввести информацию в необходимые столбцы. Однако, если вы осуществите ввод не во все столбцы и оставите пустым столбец, ввод данных в который обязателен, то получите сообщение об ошибке. Нажатие клавиши "стрелка вниз" приводит к тому, что позиция ввода (и указатель текущей записи) перемещается вниз. В этот момент SQL-сервер попытается обновить в таблице запись, которую вы покидаете.

5. Щелкнув на ОК, мы вернемся назад и сможем доввести недостающие данные. Завершив ввод, мы снова нажмем клавишу стрелка вниз, при этом база данных обновится. Обратите внимание на следующий факт: несмотря на то, что введена лишь первая запись в таблице, в столбце видим значение 2. Очередное значение идентификатора генерируется при каждой попытке вставить запись, вне зависимости от успеха этой попытки. Следствием этого факта могут стать пробелы в нумерации строк. Эта проблема легко решается при помощи Query Editor. При необходимости, можно сбросить текущее значение счетчика таким образом, что Query Editor начнет отсчет с более желательного для вас числа. Синтаксис этой команды несложен:

DBCC CHEKIDENT ("имя таблицы"[, {NORESEED|RESEED [новое_значение_счетчика]}])

где, *имя таблицы*, в которой вы хотите сбросить счетчик автоматических значений, необходимо заключить в апострофы (одинарные кавычки).

Параметр NORESEED можно использовать для того, чтобы получить от SQL-сервера предполагаемое значение счетчика, то есть иными словами, максимальное из имеющихся в столбце значений.

Счетчик для таблицы можно сбросить автоматически, просто указав *параметр RESEED* без значения. При этом будет просмотрена вся таблица и счетчик примет наибольшее из существующих уже значений. Альтернативная возможность заключается в том, чтобы явным образом указать желаемое значение счетчика, отделив его запятой от зарезервированного слова RESEED.

Удаление данных

Удаление данных из таблицы производится командой DELETE. Ее синтаксис:

```
DELETE
FROM имя_таблицы
Where условие_отбора
```

Одной командой можно удалять записи только из одной таблицы.

Удалять записи из таблицы можно также при помощи Management Studio.

1. Откройте Management Studio, откройте базу данных и найдите нужную таблицу. Щелкните на ее значке правой кнопкой мыши, выберите в контекстном меню команду Edit Table 200 rows

1. В результате будут выбраны строки таблицы. Выделите произвольную строку щелчком на серой вертикальной панели слева. В примере выделена верхняя строка.
2. Теперь нажмите клавишу *Delete*. В ответ на экране появится окно сообщения, запрашивающее ваше подтверждение. Чтобы удалить запись, щелкните на Yes (Да).
3. 4. В результате вы увидите, что запись из таблицы удалена.
4. Другой способ удалить запись состоит в том, чтобы щелкнуть на соответствующей строке правой кнопкой мыши и выбрать в контекстном меню команду Delete.

Усечение таблицы

Все операции удаления строк посредством оператора DELETE записываются в журнал транзакций. Каждый раз, когда удаляется запись, этот факт регистрируется в журнале транзакций. Если вы удаляете очень много записей из таблицы до того, как закрыть транзакцию, ваш журнал транзакций

будет расти очень быстро. В этой ситуации, чтобы не расходовать ресурсы, используют команду TRUNCATE TABLE.

Оператор TRUNCATE TABLE удаляет все записи и таблице без обработки транзакций и без ведения журнала. Синтаксис команды усечения таблицы:

TRUNCATE TABLE имя_таблицы

При использовании команды TRUNCATE TABLE после закрытия транзакции никакого способа восстановить удаленные таким образом данные не существует. Удаляются все записи, невозможно удалить записи выборочно. Одно из ограничений этой команды заключается в том, что ее нельзя применять к таблицам, содержащим внешние ключи.

Удаление таблицы

Другой способ быстро удалить данные из таблицы состоит в том, чтобы удалить саму таблицу, а затем заново ее создать. Чтобы удалить таблицу, используйте следующий код:

DROP TABLE имя_таблицы

Команда DROP TABLE не может удалить таблицу, содержащую внешние ключи. В такой ситуации необходимо удалить вначале ключи, а затем уже удалять таблицу.

Транзакции

Транзакция представляет собой единицу работы, которая должна быть подвергнута тесту ACID (Atomicity, Consistency, Isolation и Durability - Атомарность, Целостность, Изоляция и Окончательность), прежде чем быть классифицированной, как транзакция.

В своей простевшей форме атомарность означает: все изменения данных, входящие в транзакцию, должны быть успешно завершены, или же отменены все вместе, то есть изменения, которые к моменту сбоя уже успели произойти, должны быть отменены.

Вне зависимости от того, принята транзакция или отменена, целостность данных должна сохраниться.

Любые изменения данных, проводимые в рамках транзакции, должны быть изолированы от изменений, идущих в рамках любой другой транзакции. Каждая транзакция должна видеть данные в том состоянии, которое они либо имели до начала другой транзакции, либо пробрели после ее завершения.

Невозможно увидеть промежуточное состояние данных, которое они имеют в процессе выполнения транзакции.

После того как транзакция завершена, все данные приобрели свое окончательное состояние. и изменить их можно только при помощи другой транзакции. Любой системный сбой (как программный, так и аппаратный) не должен повлиять на состояние данных после завершения транзакции.

В транзакцию можно заключить любые манипуляции с данными, будь то обновление, вставка или удаление. Речь может идти о манипуляциях с единственной записью или с набором записей. Транзакции нужны только в тех случаях, когда данные в результате манипуляций изменяются, и может возникнуть необходимость отменить внесенные изменения. При работе с транзакциями нужно помнить, что, начав транзакцию, вы блокируете часть таблицы или даже всю таблицу и, при определенных настройках базы данных, можете сделать невозможной работу других пользователей. Более того, в результате выполнении транзакции может возникнуть тупиковая ситуация.

По этой причине рекомендуется делать транзакции небольшими, короткими и ни при каких обстоятельствах не удерживать блокировку более чем на несколько секунд. Чтобы достигнуть этого, необходимо помещать в транзакцию минимальное число строк кода и принимать в последующем коде решение об отмене транзакции или ее принятии настолько быстро, насколько это вообще возможно.

Транзакция состоит из двух основных элементов. Это начало (старт) транзакции и окончание транзакции, когда мы решаем, принять транзакцию или отменить ее. Команда BEGIN TRAN объявляет начало транзакции. Собственно транзакция стартует с момента выполнения этой команды. Начиная с итого момента и оканчивая моментом выполнения одной из двух команд - COMMIT TRAN или ROLLBACK TRAN - любые операции, вносящие изменения в данные, относятся к данной транзакции.

Команду BEGIN TRAN можно дополнить именем транзакции длиной до 32 символов. Если вам потребуется вложенная транзакция, то есть транзакция, стартующая внутри другой транзакции, то именем можно снабдить только "внешнюю" транзакцию.

Команда COMMIT TRAN завершает транзакцию и записывает все изменения в базу данных. После выполнения этой команды отмена изменений становится невозможной. Это команда должна выполняться только тогда, когда все изменения уже готовы для того, чтобы стать окончательными.

Если нам потребуется отменить все изменения, которые успели внести операции, относящиеся к текущей транзакции (например, по причине возникших ошибок), то мы можем использовать команду ROLLBACK TRAN. Представьте себе следующую ситуацию: мы начали транзакцию командой BEGIN TRAN, затем успешно вставили строку в таблицу командой INSERT, а потом команда UPDATE завершилась *с* ошибкой. Мы можем "откатить" таблицу к предыдущему состоянию, то есть состоянию еще до выполнения команды INSERT при помощи команды ROLLBACK TRAN. Таким образом,

команда INSERT также будет отменена, несмотря на то, что она была выполнена успешно и без ошибок.

Основное назначение блокировок состоит в том, чтобы начавшаяся транзакция, которая вносит изменения в данные и "знает", что их, возможно, придется затем отменить, могла быть твердо "уверенной" в том, что никакая другая транзакция к этому моменту не изменит данные еще раз. Однако с таким подходом связана одна проблема, заключающаяся в следующем: SQL-сервер не может просто блокировать данные, уже подвергшиеся изменению в транзакции. Блокируются данные, которые изменены не завершившейся транзакцией (блокировка на уровне строки или записи), но сервер может заблокировать и всю базу данных (блокировка на уровне базы данных). Между этими двумя крайними случаями есть целый ряд промежуточных уровней, поэтому можно заблокировать большие или меньшие ресурсы, в зависимости от выполняемых операций.

Блокировками SQL-сервер управляет автоматически, но понимание происходящих при этом процессов позволит вам эффективно использовать блокировки в ваших транзакциях.

Обновление данных на основе механизма транзакций

1. Запустите Query Editor. Первый пример демонстрирует в действии использование команды COMMIT TRAM.

2. Введите следующий код и выполните его.

```
SET QUOTED_IDENTIFIER OFF
GO
BEGIN TRAN Restore_Value
DECLARE @ValueToUpdate  varchar (30)
SET @ValueToUpdate = " Значение_1"
UPDATE  имя_таблицы
SET имя_столбца = @ValueToUpdate
Where условие_отбора
COMMIT TRAN
```

Обратите внимание, что COMMIT TRAN не использует имя транзакции, заданное в BEGIN TRAN.

3. Теперь мы продемонстрируем другой исход транзакции - отмену иди "откат" транзакции при помощи команды ROLLBACK TRAN.

```
SET QUOTED_IDENTIFIER OFF
GO
BEGIN TRAN
UPDATE  имя_таблицы
SET имя_столбца = " Значение_1"
```

Where условие_отбора

4.После выполнения этого кода, мы увидим, что произошла ошибка: весь столбец принял одно и тоже значение " Значение_1" .

5. Чтобы исправить ошибку, нужно воспользоваться командой ROLLBACK TRAN. Выполните эту команду. Это можно сделать в панели запросов.

Порядок выполнения работы

1. Изучение теоретических сведений по теме лабораторной работы.
2. Выполнение задания: ввести в свои таблицы тестовые наборы данных (в родительские таблицы – не менее 10 записей, в дочерние – не менее 20). При необходимости откорректировать созданные в лабораторной работе № 1 ограничения на вводимые данные.
3. Оформление отчета по лабораторной работе.
4. Защита лабораторной работы.

Правила отчетности студента

Лабораторная работа рассчитана на 1 часа аудиторных занятий и включает в себя изучение кратких теоретических сведений, выполнение задания к лабораторной работе, оформление отчета. Сдача лабораторной работы заключается в ответах на контрольные вопросы и демонстрации индивидуального задания.

Содержание отчета:

1. наименование, цель и задание к лабораторной работе;
2. результаты выполнения задания: PrintScrin экранной формы Edit Top 200 rows для каждой из таблиц БД с тестовыми наборами данных;
3. ответы на контрольные вопросы.

Контрольные вопросы

1. Способы ввода данных в таблицы. Ограничения целостности при вводе данных.
2. Понятие транзакции. Свойства транзакций.
3. Удаление данных, усечение таблиц, удаление таблиц.

Список рекомендуемой литературы

1. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учеб. - М.: Бином-пресс, 2007.

2. Ульман Д., Уидом Дж. Введение в системы баз данных. - М.: Издательство «Лори», 2000.

3. Дейт К. Введение в системы баз данных. - М.: Издательский дом «Вильямс», 2001.

4. Карпова Т.С. Базы данных: модели, разработка, реализация. - СПб.: Питер, 2001.

5. Ицик Бен-Ган. Microsoft SQL Server. Основы T-SQL: пер. с англ. / Ицик Бен-Ган. - СПб.: БХВ-Петербург, 2009.

6. Астахова И.Ф., Мельников В.М., Толстобров А.П., Фертиков В.В. СУБД: язык SQL в примерах и задачах: учеб. пос. для вузов. - М.: Физматлит, 2009.