Министерство Образования и Науки Республики Казахстан

Карагандинский Технический Университет имени Абылкаса Сагинова

Кафедра ИВС

***Лабораторная работа №2***

**Дисциплина:** Основы баз данных

# Тема: Создание таблиц базы данных

**Принял:**

*Нұртай М.Д.*

*(оценка)*

*(фамилия, инициалы)*

*(подпись) (дата)*

**Выполнил:**

*Новиков Д.Д.*

*(фамилия, инициалы)*

*гр. Ис-20-2*

Караганда 2022

**Цель работы:** усвоить способы создания таблиц, пользовательских типов данных, умолчаний, ограничений и индексов базы данных средствами СУБД MS SQL Server 2012.

**Задачи:**

1. Создать пользовательский тип данных
2. Определить таблицы базы данных в соответствии с требованиями индивидуального варианта задания на разработку курсового проекта (при создании таблиц определите в них первичные и внешние ключи и при необходимости используйте значения по умолчанию, ограничения и правила и созданный вами в п. 1 пользовательский тип данных). ***Ввод данных в таблицы не осуществлять***!
3. Создать индексы для таблиц базы данных

**Скрипты на создание пользовательского типа данных, таблиц БД, индексов в соответствии с индивидуальным заданием**

use stud;

go

create type custInt from int;

create type custText from text;

go

use stud\_Novikov;

create table tempTable

(id int identity(1,1) not null,

Tname char(20) not null,

descript custText null,

Ttime time not null,

quantity custInt not null);

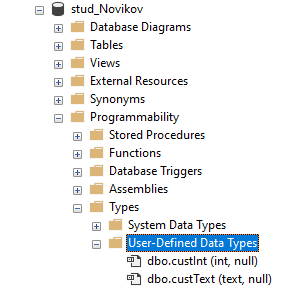
create index tempIndex

on tempTable (Tname,Ttime,quantity)

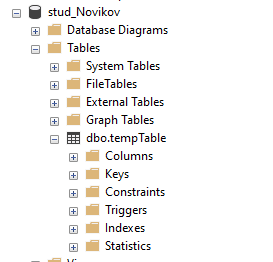
go

**Результаты выполнения индивидуального задания**

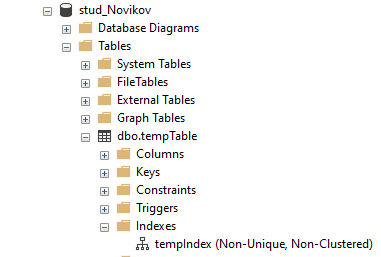
1. Были созданны 2 типа данных в базе данных



1. Была созданна таблица в базе данных



1. Был создан индекс в базе данных



**Контрольные вопросы**

1. Перечень поддерживаемых типов данных?

* точные числа, или числа с фиксированной точностью (Exact Numerics). Сюда включены типы данных:
* BIT;
* TINYINT;
* SMALLINT;
* INT;
* BIGINT;
* NUMERIC;
* DECIMAL;
* SMALLMONEY;
* MONEY.
* приближенные числа, или числа с плавающей точкой (Approximate Numerics, Float), включают типы данных:
* FLOAT;
* REAL.
* Символьные данные (Character). Включает две подгруппы:
* обычные символьные строки (Character Strings). Сюда включены следующие типы данных:
* CHAR;
* VARCHAR;
* TEXT.
* символьные строки в Юникоде (Unicode Character Strings). Типы данных:
* NCHAR;
* NVARCHAR;
* NTEXT.
* Дата и время (Date and Time). Содержит следующие типы данных:
* DATETIME;
* SMALLDATETIME;
* DATE;
* TIME;
* DATETIMEOFFSET;
* DATETIME2.
* Двоичные строки (Binary Strings). В эту группу включены следующие типы данных:
* BINARY;
* VARBINARY;
* IMAGE.
* Пространственные типы данных (Spatial Data Types). Относятся следующие типы данных:
* GEOMETRY;
* GEOGRAPHY.
* Другие типы данных (Other Data Types). Сюда включены типы данных, не очень подходящие для других категорий. В эту группу включены такие типы данных:
* SQL\_VARIANT;
* TIMESTAMP;
* UNIQUEIDENTIFIER;
* HIERARCHYID;
* CURSOR;
* TABLE;
* XML.

1. Пользовательский тип данных. Назначение и способы создания?

В SQL Server существует возможность создавать пользовательские типы данных (или псевдонимы), основываясь на базовых типах данных, а также "новые" типы данных при помощи классов сборки в среде Microsoft .NET Framework CLR. Пользовательские типы данных (типы данных задаваемые пользователями) являются индивидуально настроенными системными типами данных. Такая настройка полезна, когда вы имеете несколько таблиц, в колонках которых должны храниться данные одинаковых типов, а вы хотите генерировать точное соответствие колонок каждой из таблиц по типу, длине и возможности применения null-значений. Создав тип данных с осмысленным именем, вы упростите себе программирование и согласованность данных в ваших таблицах

Для создания пользовательского типа данных в Management Studio в окне «Обозреватель объектов» щелкните правой кнопкой мыши по элементу «Определяемые пользователем типы данных» (который находится в папке «Программирование» | «Типы») и в контекстном меню выберите элемент «Создать определяемый пользователем тип данных…». Появится диалоговое окно.

1. В поле «Имя» введите имя нового типа данных, затем задайте системный тип и длину вашего пользовательского типа данных. Если ваш тип данных позволяет использовать null-значения, то установите флажок «Разрешить значения Null».
2. Если ваш тип данных должен использовать какие-либо предопределенные правила и значения, то выберите их в соответствующих полях со списками.
3. Чтобы сохранить ваш тип данных, нажмите «Ok»

В качестве примера создадим тип данных phone, который будет использоваться для хранения телефонного номера клиента. В текстовом поле «Имя» введите phone, в раскрывающемся списке «Тип данных» выберите nchar. В качестве длины введите 10. Отметьте параметр «Разрешить значения null», чтобы иметь возможность не указывать телефонный номер при добавлении нового клиента.

После щелчка по кнопке OK в вашей базе данных будет создан новый пользовательский тип данных.

Применение операторов Transact-SQL

Для создания пользовательского типа данных в языке Transact-SQL используется оператор CREATE TYPE. При помощи запуска этой команды, когда вы работаете с модельной базой данных (model), можно организовать применение нового типа данных во всех вновь создаваемых вами БД. Запуск этой команды при работе с пользовательской БД, означает, что новый тип можно будет использовать только в данной БД.

1. Способы определения таблиц?

Таблицу можно создать как при использовании оператора CREATE TABLE языка Transact-SQL, так и с помощью диалоговых средств Management Studio.

1. В окне «Обозреватель объектов» в созданной вами базе данных в контекстном меню узла «Таблицы» выберите команду «Создать таблицу…». В рабочей области должна появиться вкладка с конструктором таблиц.
2. В первую строку в столбце «Имя столбца» введите IdCust, в столбце «Тип данных» выберите int. Убедитесь что параметр «Разрешить значения null» отключен.
3. В нижней половине экрана в разделе «Свойства столбцов» введите описание поля и измените значение параметра «Спецификация идентификатора / (Идентификатор)» на «Да» для того чтобы значения номера клиента формировались автоматически. Свойство «Идентифицирующий столбец» (Identity), обычно используемое совместно с типом данных int, предназначено для автоматического приращения значения на единицу при добавлении каждой новой записи. К примеру, клиент, добавленный в таблицу первым, будет иметь значение идентификатора 1, вторым – 2, третьим – 3, и т.д.
4. Аналогичным образом введите описания всех остальных полей и закройте окно конструктора таблиц. Введите в качестве имени таблицы Customer. Вновь созданная таблица должна появиться в дереве обозревателя объектов в папке «Таблицы».
5. Назначение ограничений?

В таблице могут описываться ограничения (constraint), определяющие некоторые дополнительные характеристики одного или группы столбцов таблицы

1. Перечень поддерживаемых умолчаний, ограничений, правил?

*ограничение на значения, помещаемые в столбцы таблицы.* Это ограничение CHECK, благодаря которому в таблицу не может быть помещена новая строка или выполнено изменение данных уже существующей в таблице строки, если будет нарушено указанное ограничение;

*значение по умолчанию*. Это ограничение DEFAULT. Если при добавлении в таблицу новой строки не было задано значение какого-то столбца, то ему будет присвоено значение по умолчанию. Если при описании столбца не было явно указано значение по умолчанию (ограничение DEFAULT), то этим значением является NULL. Значение по умолчанию используется только при добавлении новой строки в таблицу, но не при изменении значений данных существующей строки;

*допустимость для столбца значения NULL*. Предложение NOT NULL в описании столбца запрещает помещать в этот столбец значение NULL;

*ограничение первичного ключа.* Это ограничение PRIMARY KEY. Первичный ключ – это столбец или группа столбцов, значение которых однозначно определяет конкретную строку таблицы. Таблица может иметь один, и только один первичный ключ;

*ограничение уникального ключа.* Это ограничение UNIQUE, предназначенное для устранения дублирования значений. Каждая таблица может содержать произвольное количество уникальных ключей;

*ограничение внешнего ключа.* Это ограничение FOREIGN KEY. Внешний ключ – это столбец или группа столбцов таблицы, которые ссылаются на первичный или уникальный ключ другой или этой же самой таблицы.

1. Способы создания ограничений?

В контекстном меню папки «Ограничения» таблицы Customer выберите команду «Создать ограничение».

В открывшемся окне «Проверочные ограничения» заполните следующие поля:

Имя: CK\_Zip

Выражение: ([zip] like '[0-9][0-9][0-9][0-9][0-9][0-9]'). Данное выражение описывает ограничение, принимающее пять символов, которыми могут быть только цифры от 0 до 9.

Описание: Ограничение на значения почтового индекса

Щелкните на кнопке «Закрыть» и закройте конструктор таблиц (он был открыт, когда вы начали создавать ограничение) с сохранением изменений.

Применение операторов Transact-SQL

Например, если бы нам нужно было ограничить диапазон возможных значений, допустимых для ко­лонки Price (цена) таблицы Product\_Info, величинами от 1 до 50000, то мы использо­вали бы следующий оператор:

CONSTRAINT CK\_Price CHECK (price>=1 AND price <=50000)

Приведенный ниже оператор создает таблицу Customer1 с ограничением первичного ключа.

CONSTRAINT PK\_Customer1 PRIMARY KEY (IdCust)

Синтаксис задания ограничения уникального ключа отличается от синтаксиса первичного ключа только лишь названием вида ключа

CONSTRAINT UK\_Phone2 UNIQUE (Phone)

Приведенный ниже оператор создает таблицу Customer1 с внешним ключом

FOREIGN KEY (IdCity) REFERENCES City (IdCity)

ON DELETE CASCADE

ON UPDATE CASCADE

1. Назначение и классификация индексов?

Индекс – это вспомогательная структура данных, используемая системой SQL Server для доступа к данным. В системах без индексов весь поиск данных должен выполняться путем скани­рования таблиц. При сканировании таблиц приходится читать все данные и срав­нивать их с запрашиваемыми данными. Используя индекс, вы можете кардинально снизить количество операций ввода-вывода, ускорив дос­туп к данным и освободив системные ресурсы для других операций.

Существует два типа индексов: *кластеризованные (кластерные) индексы* и *некластеризованные (некластерные) индексы.* Кластеризованный индекс хранит в своих самых нижних узлах реальные строки данных, и поэтому дан­ные становятся доступны, как только найден определенный узел, что может сокращать количество операций ввода-вывода. Еще одним преимуществом кластеризованных индексов является то, что считы­ваемые данные получаются в отсортированном по индексу виде. Недостатком использования кластеризованного индекса является то, что доступ к таблице всегда происходит через индекс, что может приводить к дополнительной нагрузке на систему. Не имеющая кластерного индекса таблица называется *кучей* (heap).

Некластеризованный индекс является вспомогательной структурой и не содержит реальных данных таблицы в своих узлах. В некластеризованном индексе его самый нижний узел содержит значение ключа, а также идентификатор строки (Row ID), указывающий нужную строку в таблице. Это значение обеспе­чивает быстрый доступ к реальным данным, указывая точное местоположение этих данных. На практике используется несколько некластеризованных индексов по различным колонкам таблицам

1. Способы создания индексов?
2. В контекстном меню папки «Индексы» таблицы PEOPLE выберите команду «Создать индекс…».
3. В открывшемся окне «Создание индекса» заполните следующие поля:

* Имя индекса:
* Тип индекса:

1. Чтобы указать, какие столбцы войдут в состав индекса, щелкните по кнопке «Добавить…». Появится окно, в котором будут перечислены имена столбцов таблицы. Отметьте столбцы NAME1, NAME2, NAME3 и щелкните по кнопке OK.
2. Опять появится окно создания индекса с выбранными столбцами. Чтобы указать, что индекс уникальный, нужно отметить флажок в поле «Уникальный». Щелкните по кнопке OK, и созданный индекс появится в папке «Индексы».

Применение операторов Transact-SQL

Используя T-SQL для создания индекса, вы можете генерировать сценарий для соответствующей команды и запускать его многократно. Кроме того, этот метод создания индекса дает вам больше гибкости, поскольку вы имеете доступ к большему числу параметров.

Для создания индекса с помощью T-SQL вы должны использовать оператор CREATE INDEX. Эта команда имеет следующий синтаксис:

CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX <имя индекса>

ON [[<имя базы данных>.]<имя схемы>.]

{<имя таблицы> | <имя представления>}

(<имя столбца> [ ASC | DESC ] [, <имя столбца> [ ASC | DESC ]]...)

[ INCLUDE (<имя столбца> [, <имя столбца>]...) ]

[ WHERE <условие фильтра> ]

[ WITH (<параметр индекса> [, <параметр индекса>]...) ]

[ ON { <схема секционирования> (<разделяющий ключ>)

| <файловая группа>

| "default"

} ]

[ FILESTREAM\_ON { <файловая группа потока>

| <схема секционирования>

| "NULL"

}

];

1. Что представляют собой Null-значение?

Значение не имеет в себе хранимой информации

1. Типы целостности данных?

**Доменная целостность.** Ограничение диапазона данных, вводимых пользователем в поле. Основными инструментами обеспечения доменной целостности являются ограничения вводимых значений и значения по умолчанию.

**Сущностная целостность**. Обеспечение гарантии уникальности записей в таблицах и предотвращение их дублирования. Основными инструментами обеспечения целостности сущностей являются первичные ключи и ограничения уникальности.

**Целостность ссылок.** Суть обеспечения целостности ссылок очевидна из названия: данные в одной таблице, ссылающиеся на данные из другой таблицы, защищены от некорректного обновления. В терминологии SQL Server это называется *декларативной ссылочной целостностью* и достигается путем связывания первичного ключа одной из таблиц с внешним ключом другой таблицы (создается так называемое ограничение внешнего ключа).

**Каскадная ссылочная целостность.** При наличии ограничения внешнего ключа с параметрами по умолчанию вы не можете удалить запись или изменить значение первичного ключа главной таблицы в случае наличия связанных записей в подчиненной таблице (в которой организовано ограничение внешнего ключа). Однако это поведение можно изменить, используя каскадную ссылочную целостность.