|  |
| --- |
| **Создание таблицы** |

Для создания таблицы используется инструкция CREATE TABLE. Рассмотрим следующий шаблон:

|  |
| --- |
| **Шаблон:** |
| CREATE TABLE названиеТаблицы (названиеСтолбца\_1 типДанныхСтолбца\_1, названиеСтолбца\_2 типДанныхСтолбца\_2, …)  где:  названиеТаблицы/Столбца - любое название, которое захочет пользователь (если название состоит из нескольких фраз, то писать его слитно и не использовать пробел);  типДанныхСтолбца - определяется, какого типа данных будет храниться информация в данном столбце. |

|  |  |
| --- | --- |
| **Пример:** | |
| Создать таблицу с названием test и двумя столбцами: column\_1 с целочисленным типом данных и column\_2 со строковым типом данных с ограничением в 50 символов. | |
| **Решение:** | CREATE TABLE test(column\_1 int, column\_2 varchar(50)) |

Создание таблицы невозможно без представления типов данных. Рассмотрим их, разделив на группы.

|  |
| --- |
| **Типы данных полей** |

* Точные числа

bigint, int, smallint, tinyint - хранение любых целых значений, данные типы данных различаются предельным хранимым значением, более подробно в таблице ниже:

|  |  |  |
| --- | --- | --- |
| **Тип данных** | **Диапазон** | **Хранилище** |
| bigint | от -2^63 (-9 223 372 036 854 775 808)  до 2^63-1(9 223 372 036 854 775 807) | 8 байт |
| int | от -2^31 (-2 147 483 648) до 2^31-1(2 147 483 647) | 4 байта |
| smallint | от -2^15 (-32 768) до 2^15-1 (32 767) | 2 байта |
| tinyint | от 0 до 255 | 1 байт |

|  |  |
| --- | --- |
|  | Если не отслеживать перевод типа данных из одного в другой, то СУБД самостоятельно переводит число больше 2 147 483 647 не в тип данных bigint, а в decimal. |

|  |
| --- |
| **Задание 1** |
| Инструкция выборки может применяться для выражений без использования конкретных таблиц. К этому мы вернемся в дальнейшем, а пока необходимо переписать и выполнить запрос, приведенный ниже:  SELECT 2147483647/2 , 2147483651/2 |
| **Результат выполнения запроса записать в лист опроса** |

|  |
| --- |
| **Задание 2** |
| Написать инструкцию создания таблицы, состоящей из 2-х полей. В первом поле может храниться любое целочисленное число не больше 33 000. Во втором поле возможно хранение целочисленных данных, чье значение не больше 255. Имена полей выбирать самостоятельно. |
| **Инструкцию проверить в среде Management Studio и записать в лист опроса** |

bit – хранит значение или 1, или 0, или NULL

decimal, numeric - хранение целых чисел с фиксированной точностью и масштабом. Пример decimal(5,2) – 5 цифр будет всего и из них 2 цифры после запятой. Важно отметить, что существует ограничение точности, что можно увидеть на таблице ниже (там же приведены количественные показатели занимаемой памяти при увеличении точности):

|  |  |
| --- | --- |
| **Точность** | **Хранилище** |
| 1-9 | 5 |
| 10-19 | 9 |
| 20-28 | 13 |
| 29-38 | 17 |

|  |
| --- |
| **Задание 3** |
| 1. Составить инструкцию создания таблицы с 2-мя полями: одно с типом данных decimal(5,2), другое поле – numeric(10,5). |
| **Инструкцию создания проверить в среде Management Studio и записать в лист опроса** |
| 2. В данном задании мы воспользуемся инструкцией ввода - INSERT, которую более подробно рассмотрим далее по курсу. Добавим с помощью инструкции ввода, приведенной ниже, данные в созданную таблицу:  INSERT INTO названиеВашейСозданнойТаблицы VALUES(123, 12345.12)  3. Вывести все данные из вашей таблицы с помощью инструкции выборки, приведенной ниже:  SELECT названиеВашегоПервогоСтолбца, названиеВашегоВторогоСтолбца FROM  названиеВашейСозданнойТаблицы |
| **Результат выполнения запроса выбора всех данных из таблицы записать в лист опроса** |

|  |
| --- |
| **Задание 4** |
| Написать инструкцию создания таблицы, состоящей из 2-х полей. В первом поле может храниться любое целочисленное число не больше 255. Во втором поле возможно хранение целого числа с фиксированной точностью: всего 7 цифр, из них 6 после запятой. Имена полей выбирать самостоятельно. |
| **Инструкцию создания проверить в среде Management Studio и записать в лист опроса** |

money, smallmoney - поля с данными типами данных хранят в себе денежные (валютные) значения. Диапазон хранения представлен в таблице ниже:

|  |  |  |
| --- | --- | --- |
| **Тип данных** | **Диапазон** | **Хранилище** |
| money | от -922 337 203 685 477, 5808  до 922 337 203 685 477,5807 | 8 байт |
| smallmoney | от -214 748,3648 до 214 748,3647 | 4 байта |

* Приблизительные числа

float, real - хранение числовых данных с плавающей запятой. Написание – float(n), где n – некоторое значение в пределах, что указаны в таблице ниже:

|  |  |  |
| --- | --- | --- |
| **Значение n** | **Точность** | **Хранилище** |
| 1-24 | 7 знаков | 4 байта |
| 25-53 | 15 знаков | 8 байт |

|  |  |
| --- | --- |
|  | Вещественные данные пишутся с разделением **ТОЧКОЙ**, пример: 100.50 |

* Дата и время

time, date, smalldatetime, datetime, datetime2, datetimeoffset - определяют дату/время. Важно отметить, что дата по умолчанию задается в формате: гггг-мм-дд (пример: 2016-01-01).

Для выполнения следующего задания необходимо рассмотреть полезную функцию преобразования выражения из одного типа данных в другой – функция CAST.

|  |
| --- |
| **Шаблон:** |
| CAST (выражение AS приводимыйТипДанных) |

Для наглядного представления разницы между типами данных, отвечающими за дату/время, выполним следующее задание.

|  |
| --- |
| **Задание 5** |
| Выполнить запрос, приведенный ниже, который преобразует выражения в разные типы данных даты/время:  SELECT CAST(‘2016-01-01 12:00:00.1234567 +12:15’ AS time(7)),  CAST(‘2016-01-01 12:00:00.1234567 +12:15’ AS date),  CAST(‘2016-01-01 12:00:00.123’ AS smalldatetime),  CAST(‘2016-01-01 12:00:00.123’ AS datetime),  CAST(‘2016-01-01 12:00:00.1234567 +12:15’ AS datetime2(7)),  CAST(‘2016-01-01 12:00:00.1234567 +12:15’ AS datetimeoffset(7)); |
| **Результат выполнения запроса записать в лист опроса** |

* Символьные строки

char, varchar - с помощью данного типа возможно хранение строковых данных фиксированной длины. Написание – char(n), varchar(n), varchar(max), где n – длина строки от 1 до 8000, значение max подразумевает хранение максимально большой строки в 2 Гб. Если размеры записей одинаковы – то используется char, если размер записей будет постоянно меняться – varchar.

|  |  |
| --- | --- |
|  | Строковые данные пишутся в запросах с использованием одиночных кавычек (апострофов), пример: ‘Строка’. |

|  |
| --- |
| **Задание 6** |
| Написать инструкцию создания таблицы, состоящей из 2-х полей. В первом поле может храниться фамилия клиента. Во втором поле возможно хранение счета депозита человека, чья фамилия указана в первом поле таблицы. Имена полей выбирать самостоятельно. |
| **Инструкцию создания проверить в среде Management Studio и записать в лист опроса** |

* Двоичные данные

binary, varbinary - типы двоичных данных с фиксированной или переменной длиной. Написание – binary(n), varbinary(n), varbinary(max), где n – значение от 1 до 8000 (байт), max: 232-1 байт.

|  |
| --- |
| **Задание 7** |
| Выполнить запросы, которые преобразуют целочисленное выражение в бинарное, приведенные ниже:  SELECT CAST(123456 AS BINARY(2));  SELECT CAST(123456 AS BINARY(4)); |
| **Результат выполнения запросов записать в лист опроса** |

|  |
| --- |
| **Задание 8** |
| 1. Создать таблицу с 4-мя полями с типами данных: tinyint, float, date, char. Размерность (n) для типов и имена полей выбирать самостоятельно. |
| **Инструкцию создания проверить в среде Management Studio и записать в лист опроса** |
| 2. С помощью инструкций ввода, приведенных ниже, попробовать заполнить созданную таблицу:  INSERT INTO названиеВашейТаблицы VALUES(256,100.49,’2016-01-01’,’test’);  INSERT INTO названиеВашейТаблицы VALUES(256,100.9999999999,’2016-01-01’,’test’);  INSERT INTO названиеВашейТаблицы VALUES(255,100.49,’2016-01-01’,’test’);  INSERT INTO названиеВашейТаблицы VALUES(100,100.99999999999999999,’2016-01-01’,’test’); |
| **Заполнить лист опроса инструкциями заполнения таблицы, которые выполнились и которые НЕ выполнились. Описать, почему не выполнились инструкции.** |

Важным параметром при создании таблицы является выбор и задание первичного ключа – уникального поля.

|  |
| --- |
| **Установка первичного ключа** |

Установка первичного ключа для таблицы позволяет нам ускорить поиск необходимых данных, а также служит важным элементом для связи между таблицами.

Для того чтобы выделить один (или несколько) столбцов как первичный ключ, необходимо воспользоваться параметром **PRIMARY KEY(<название столбца>,…)**.

|  |
| --- |
| **Шаблон:** |
| CREATE TABLE названиеТаблицы (названиеСтолбца\_1 типДанныхСтолбца\_1 NOT NULL PRIMARY KEY(названиеСтолбца\_1), названиеСтолбца\_2 типДанныхСтолбца\_2, …) |

Особое внимание стоит обратить на параметр NOT NULL – устанавливает свойство поля, которое не позволяет хранить в нем нулевое значение (пустоту). Для установки первичного ключа данный параметр является обязательным.

|  |  |
| --- | --- |
| **Пример:** | |
| Создать таблицу test, состоящую из двух полей: первое поле с именем id целочисленного типа данных и с параметром первичного ключа, второе поле – name строкового типа данных и с ограничением в 10 символов. | |
| **Решение:** | CREATE TABLE test(id int NOT NULL PRIMARY KEY(id), name varchar(10)) |

Для последующего обращения к первичному ключу (изменение, удаление) используется специальное ограничение – CONSTRAINT – которое позволяет проименовать данный ключ и обращаться к нему уже по имени. Обычно, имя задается с аббревиатурой «PK\_» и именем таблицы, для которой выставляется ключ. В данном примере создается таблица с ключом, к которому возможно обратиться в любой момент времени по имени PK\_test.

|  |
| --- |
| **Пример:** |
| CREATE TABLE test(id int NOT NULL CONSTRAINT PK\_test PRIMARY KEY(id)) |

|  |
| --- |
| **Задание 9** |
| Составить инструкцию, которая создаст таблицу из 3-х полей. Одно из полей должно быть первичным ключом и данный ключ должен быть проименован. Типы данных этих полей должны быть такими, чтобы следующие инструкции добавления записи выполнились успешно:  INSERT INTO test VALUES (10,’proverka’,’2016-01-01’);  INSERT INTO test VALUES (250,’neproverka’,’2016-01-01’);  INSERT INTO test VALUES (40000,’samayaproverka’,’2016-01-01’) |
| **Инструкцию создания записать в лист опроса** |

Первичный ключ может состоять не только из одного поля, возможно использование составного первичного ключа – когда параметр PRIMARY KEY присваивается сразу нескольким полям.

|  |
| --- |
| **Шаблон:** |
| CREATE TABLE названиеТаблицы (названиеСтолбца\_1 типДанныхСтолбца\_1 NOT NULL, названиеСтолбца\_2 типДанныхСтолбца\_2 NOT NULL PRIMARY KEY(названиеСтолбца\_1, названиеСтолбца\_2)) |

|  |  |
| --- | --- |
|  | Параметр первичного ключа – PRIMARY KEY, как и ограничение CONSTRAINT, описывается для таблицы **ТОЛЬКО** один раз, если у нас предполагается, что ключ будет составной, то параметр PRIMARY KEY задаем после описания последнего поля, входящего в данный составной ключ. |

|  |
| --- |
| **Задание 10** |
| Составить инструкцию, которая создает таблицу из трех полей: номера, названия и даты. Выбрать типы данных для этих полей самостоятельно. Все три поля должны быть первичным ключом и выставлено ограничение CONSTRAINT для данного составного ключа. |
| **Инструкцию создания записать в лист опроса** |

Другим параметром, использующимся для создания таблиц, является IDENTITY(n,m) – автоикрементация, СУБД автоматически заполняет данные, начиная со значения n и с шагом в m.

|  |  |
| --- | --- |
| **Пример:** | |
| Создать таблицу test, состоящую из двух полей: первое поле с именем id целочисленного типа данных и с параметром первичного ключа с автоинкрементацией (шаг равен 1), второе поле – name строкового типа данных с ограничением в 10 символов. | |
| **Решение:** | CREATE TABLE test(id int IDENTITY(1,1) NOT NULL CONSTRAINT PK\_test PRIMARY KEY(id), name varchar(10)) |

При добавлении записи в поле name таблицы, созданной в примере выше, поле id будет автоматически заполняться СУБД, начиная со значения 1. Следующая запись в таблице будет иметь значение поля id равное 2 и так далее.

|  |  |
| --- | --- |
|  | При наличии автоинкрементного поля в таблице, во время заполнения его указывать и заполнять своими данными **НЕ** надо. |

|  |
| --- |
| **Задание 11** |
| 1. Составить инструкцию, которая создает таблицу из 2-х полей: id и name. Тип данных полей выбрать самостоятельно. Выставить для поля id значение первичного ключа (и указать ограничение) с автоинкрементацией, начинающегося с 1 и шагом в 100. |
| **Инструкцию создания записать в лист опроса** |
| 2. С помощью инструкций ввода, приведенных ниже, добавить в созданную таблицу 3 записи:  INSERT INTO названиеВашейТаблицы(name) VALUES(‘zapis\_1’);  INSERT INTO названиеВашейТаблицы(name) VALUES(‘zapis\_2’);  INSERT INTO названиеВашейТаблицы(name) VALUES(‘zapis\_3’);  3. Осуществить выборку всех данных из таблицы с помощью инструкции выборки, представленной ниже:  SELECT id, name FROM названиеВашейТаблицы |
| **Результат выполнения выборки записать в лист опроса** |

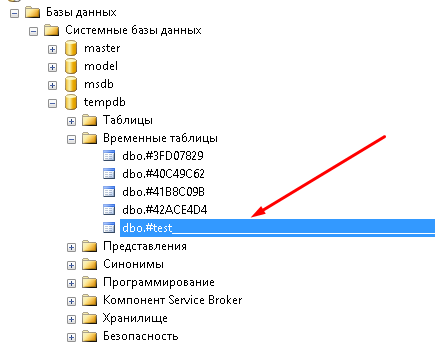
|  |
| --- |
| **Временные таблицы** |

Временные таблицы полезны, когда Вам необходимо получить некий набор данных и произвести с ним какие-то действия, но при этом без постоянного хранения на сервере.

Существует два вида временных таблиц: локальные и глобальные. Они отличаются друг от друга именами, видимостью и доступностью. Имена локальных временных таблиц начинаются с одного символа (#); они видны только текущему соединению пользователя и удаляются, когда пользователь отключается от экземпляра SQL Server. Имена глобальных таблиц начинаются с двух символов (##); они видны любому пользователю и удаляются, когда все пользователи, которые на них ссылаются, отключаются от экземпляра SQL Server.

|  |  |
| --- | --- |
| **Пример:** | |
| Создать локальную временную таблицу из двух полей: целочисленного и строкового с ограничением в 10 символов. | |
| **Решение:** | CREATE TABLE #test(id int, text varchar(10)) |

После выполнения инструкции, проверьте, что таблица появилась НЕ в вашей базе данных, а в базе данных «Системные базы данных» раздел временных сущностей - «tempdb» в папке «Временные таблицы».



Пока вы не закроете приложение SQL Management Studio, данная таблица будет вам доступна, и вы с ней можете работать как с обычной таблицей. После закрытия приложения – таблица удаляется автоматически.

|  |
| --- |
| **Задание 12** |
| 1. Составить инструкцию создания глобальной временной таблицы с двумя полями: id и comment. Типы данных для полей выбираете на свое усмотрение. Должен быть выставлен первичный ключ с автоинкрементацией и ограничением (CONSTRAINT). |
| **Инструкцию создания проверить в среде Management Studio и записать в лист опроса** |
|  |

**ЛИСТ ОПРОСА //БЛОК №1**

**Слушатель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Задание 1: результат выполнения инструкции**

SELECT 2147483647/2, 2147483651/2

|  |
| --- |
| 1073741823; 1073741825.500000 |

**Задание 2:**

Инструкция создания таблицы:

|  |
| --- |
| USE test\_createdb; |
| CREATE TABLE testtable(first\_1 smallint,second\_2 tinyint); |

**Задание 3:**

Инструкция создания таблицы:

|  |
| --- |
| USE test\_createdb;  CREATE TABLE task3 (stolb1 decimal, stolb2 numeric(10,5));  SELECT stolb1, stolb2 from k3tas;  INSERT INTO task3 VALUES(123,12345.12);  SELECT stolb1, stolb2 FROM task3; |
|  |

Результат выполнения запроса:

|  |
| --- |
|  |

**Задание 4:**

Инструкция создания таблицы:

|  |
| --- |
|  |
| USE test\_createdb;  CREATE TABLE task4 (stolb1 tinyint, stolb2 decimal(7,6)) |

**Задание 5: написать результат преобразований строкового типа в date/time**

|  |  |
| --- | --- |
| **Преобразование** | **Результат** |
| 2007-05-08 12:35:29. 1234567 +12:15 as time(7) | 12:35:29.1234567 |
| 2007-05-08 12:35:29. 1234567 +12:15 as date | 2007-05-08 |
| 2007-05-08 12:35:29.123 as smalldatetime | 2007-08-05 12:35:00 |
| 2007-05-08 12:35:29.123 as datetime | 2007-08-05 12:35:29.123 |
| 2007-05-08 12:35:29. 1234567 +12:15 as datetime2(7) | 2007-05-08 12:35:29.1230000 |
| 2007-05-08 12:35:29. 1234567 +12:15 as datetimeoffset(7) | 2007-05-08 12:35:29.1230000 +00:00 |

**Задание 6:**

Инструкция создания таблицы:

|  |
| --- |
| USE test\_createdb;  CREATE TABLE task5 (surname varchar(20), number char(12)) |
|  |

**Задание 7: написать результат выполнения инструкции преобразования в BINARY**

|  |  |
| --- | --- |
| **Преобразование** | **Результат** |
| 123456 as BINARY(2) | 0xE240 |
| 123456 as BINARY(4) | 0x0001E240 |

**Задание 8:**

|  |
| --- |
| Инструкция создания таблицы с 4-мя столбцами  USE test\_createdb;  CREATE TABLE task8(stolb1 tinyint, stolb2 float, stolb3 date, stolb4 char(5)); |
|  |

Выполнились инструкции:

|  |
| --- |
| INSERT INTO task8 VALUES(255,100.49,'2016-01-01','test'); |
| INSERT INTO task8 VALUES(100,100.99999999999999999,'2016-01-01','test'); |
|  |
|  |
|  |

Не выполнились инструкции, почему:

|  |
| --- |
| INSERT INTO task8 VALUES(256,100.49,'2016-01-01','test'); //256 > tinyint(max=255) |
| INSERT INTO task8 VALUES(256,100.9999999999,'2016-01-01','test'); //256 > tinyint(max=255) |
|  |
|  |
|  |
|  |
|  |

**Задание 9:**

Инструкция создания:

|  |
| --- |
| USE test\_createdb;  CREATE TABLE task9(first1 int NOT NULL PRIMARY KEY(first1), first2 char(20), first3 date) |
|  |

**Задание 10:**

Инструкция создания:

|  |
| --- |
| USE test\_createdb;  CREATE TABLE task10(first1 int NOT NULL , first2 char(20) NOT NULL, first3 date NOT NULL CONSTRAINT test\_createdb PRIMARY KEY (first1,first2,first3)) |
|  |

**Задание 11:**

Инструкция создания:

|  |
| --- |
| USE test\_createdb;  CREATE TABLE task11(id int IDENTITY(1,100) NOT NULL CONSTRAINT test\_createdb PRIMARY KEY (id), name1 char(20)) |
|  |

Результат выполнения выборки:

|  |
| --- |
| 1 zapis\_1 |
| 101 zapis\_2 |
| 201 zapis\_3 |

**Задание 12:**

Инструкция создания:

|  |
| --- |
| UsE test\_createdb;  cREaTe TAblE ##task12 (id int IdEnTiTy(1,1) NoT nuLl, comment char(30) nOT NulL CoNsTrAINt test\_createdsb PrImARy keY(id,comment)); |
|  |