**MSSQL SERVER**

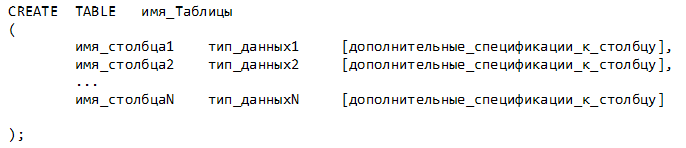
Адрес сервера 10.3.21.100

Имя БД : sp2822

Логин : sp2822user

Пароль : sp2822pswd

Синтаксис:



Типы данных :

int

tinyint – 1 байт

smallint – 2 байта

bigint – 8 байт

real()

float()

numeric()

smalldatetime

datetime

binary(кол-во требуемых байт) – для хранения байтов(картинка, музыка) – размер фиксированный, занимает на диске сколько указано

varbinary() – размер переменный, занимает сколько реальный размер

char(кол-во символов)

varchar(кол-во символов)

Запрос на добавление продуктов и изменение цены:

INSERT INTO product\_09549(id, name, price, weight) VALUES

(1, 'Snikers', 12.5, 45),

(2, 'Mars', 20, 75),

(3, 'Twix', 15.7, 35),

(4, 'Bounty', 18, 40),

(5, 'Lion', 20, 50),

(6, 'Nuts', 21, 60),

(7, 'KitKat', 14, 55),

(8, 'Aero', 11, 40),

(9, 'Wispa', 17, 30),

(10, 'Picnic', 15, 50);

SELECT \* FROM product\_09549;

UPDATE product\_09549 SET price = price \* 1.2 WHERE price < 15;

Дополнительные спецификации к столбцу:

null – в столбце допустимы значения null (стоит по умолчанию)

not null – в данном столбце не могут быть значения null (приведет к ошибке СУБД)

primary key – столбец является первичным ключом

identity(start, step) – аналог счетчика в MS Access, где start – начальное значение счетчика, step – шаг изменения (увеличения) следующего значения для первичного ключа

unique – в столбце должны быть только уникальные значения, попытка вставить повторяющееся значение приведет к ошибке

default ‘значение по умолчанию для столбца’ – позволяет задать значение по умолчанию для столбца, - если при добавлении строки значение для столбца не будет указано, ему будет присвоено не значение null, а значение по умолчанию

Синтаксис удаления таблицы:

DROP TABLE имя\_таблицы

CREATE TABLE product\_09549

(

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

prodname varchar(64) unique,

price numeric(8, 2),

prodweight int default '50'

);

INSERT INTO product\_09549(prodname, price, prodweight)

VALUES

('Snikers', 12.5, 45),

('Mars', 20, 75),

('Twix', 15.7, 35);

Создание таблицы с доп. Спецификаторами:

--CREATE TABLE product\_09549

--(

-- id int not null primary key identity(1, 1),

-- prodname varchar(64) unique,

-- price numeric(8, 2),

-- prodweight int default '50'

--);

--INSERT INTO product\_09549(prodname, price, prodweight)

--VALUES

--('Snikers', 12.5, 45),

--('Mars', 20, 75),

--('Twix', 15.7, 35);

--INSERT INTO product\_09549(prodname)

--VALUES ('Bounty');

--SELECT \* FROM product\_09549;

Запрос к базе Library (вывод книги по программированию с максимумом страниц)

SELECT books.name, books.npages, themes.name

FROM books, themes

WHERE books.npages =

(

SELECT MAX(npages)

FROM books, themes

WHERE books.id\_themes = themes.id

AND themes.name = 'Программирование'

)

AND books.id\_themes = themes.id

AND themes.name = 'Программирование';

**ВНЕШНИЕ ОБЪЕДИНЕНИЯ : OUTER JOIN**

Два внешних объединения:

- правое внешнее объединение RIGHT OUTER JOIN;

- левое внешнее объединение LEFT OUTER JOIN.

При правом внешнем объединении строки таблицы находящиеся справа от конструкции RIGHT OUTER JOIN включаются в результирующую таблицу целиком, а строки таблицы находящиеся слева от RIGHT OUTER JOIN включаются в результирующую таблицу только если имеют пару со строками таблицы, находящимися справа.

*Пример: вывод списка студентов и если студент был в библиотеке вывод идентификатора взятой книги*

SELECT Students.lastname + ' ' + Students.firstname AS student, s\_cards.id\_book AS book

FROM s\_cards RIGHT OUTER JOIN Students ON Students.id = s\_cards.id\_student;

*вывод списка студентов и если студент был в библиотеке вывод названия взятой книги*

SELECT Students.lastname + ' ' + Students.firstname AS student, books.name AS book

FROM books RIGHT OUTER JOIN

(s\_cards RIGHT OUTER JOIN Students ON Students.id = s\_cards.id\_student)

ON books.id = s\_cards.id\_book;

Внешние объединения очень полезны, когда нужно делать выборку из базы данных в которой по какой-то причине нарушена целостность базы данных (если не наложено условие целостности).

CREATE TABLE Genres\_09549

(

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

gname varchar(64)

);

INSERT INTO Genres\_09549 (gname) VALUES ('Фантастика');

INSERT INTO Genres\_09549 (gname) VALUES ('Драма');

INSERT INTO Genres\_09549 (gname) VALUES ('Боевик');

CREATE TABLE Films\_09549

(

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

fname varchar(64),

id\_genre int

);

INSERT INTO Films\_09549 (fname, id\_genre) VALUES ('Star Wars 8', 1);

INSERT INTO Films\_09549 (fname, id\_genre) VALUES ('Titanic', 2);

INSERT INTO Films\_09549 (fname, id\_genre) VALUES ('Крепкий орешек', 3);

SELECT Films\_09549.fname, Genres\_09549.gname

FROM Films\_09549, Genres\_09549

WHERE Films\_09549.id\_genre = Genres\_09549.id;

DELETE FROM Genres\_09549 WHERE id = 2;

SELECT Films\_09549.fname, Genres\_09549.gname

FROM Genres\_09549 RIGHT OUTER JOIN Films\_09549 ON Films\_09549.id\_genre = Genres\_09549.id;

При левом объединении строки находящиеся слева от LEFT OUTER JOIN включаются в результирующую таблицу целиком, а строки таблицы находящиеся справа от LEFT OUTER JOIN включаются в результирующую таблицу, если имеют пару со строками таблицы, находящимися слева.

*Пример: вывод списка студентов и если студент был в библиотеке вывод идентификатора взятой книги*

SELECT Students.lastname + ' ' + Students.firstname AS student, s\_cards.id\_book AS book

FROM Students LEFT OUTER JOIN s\_cards ON Students.id = s\_cards.id\_student;

*или:*

SELECT Students.lastname + ' ' + Students.firstname AS student, s\_cards.id\_book AS book

FROM Students LEFT OUTER JOIN

(s\_cards Left OUTER JOIN books ON s\_cards.id\_book = books.id)

ON Students.id = s\_cards.id\_student;

**ВИДЫ : VIEW**

Вид – это виртуальная таблица, которая создается в момент обращения к ней. В базе данных хранится всего лишь команда, на основе которой эта виртуальная таблица создается. Вид базируется на реальных таблицах, поэтому обращение к виду всегда будет давать актуальные данные. Любой вид базируется на запросе SELECT.

**Синтаксис создания вида:**

CREATE VIEW имя\_вида

AS

SELECT …

[WITH CHECK OPTION];

WITH CHECK OPTION – суть заключается в том, что при добавлении строк в обновляемый вид (6 правило Кодда) должно использоваться для проверки правильности добавляемых данных условие WHERE, которое указано в предложении SELECT запроса, на основе которого базируется вид. Если этой опции нет, то никакой проверки осуществляться не будет.

**Синтаксис удаления вида:**

DROP VIEW имя\_вида;

*Пример создания вида:*

CREATE VIEW BooksAuthors\_09549

AS

SELECT books.name AS book, authors.lastname + ' ' + authors.firstname AS author

FROM books, authors

WHERE books.id\_author = authors.id;

**GO** между запросами – сначала выполнить один запрос, потом переходим к следующему

*Выборка количества книг автора:*

SELECT author, COUNT(\*)

FROM BooksAuthors\_09549

GROUP BY author;

***ТРАНЗАКЦИИ***

Транзакция позволяет объединить несколько команд SQL в одну большую составную команду, которая или выполняется, или не выполняется целиком.

Любая транзакция обладает свойством ACID:

A – Atomicity (атомарность, неделимость транзакции – или выполняется, или не выполняется целиком)

C – Consistency (связность – как в случае успешного, так и в случае аварийного завершения транзакции целостность данных в базе данных не нарушается)

I – Isolation (изолированность – одна транзакция не может взаимодействовать с другими транзакциями)

D – Durability (надежность – транзакция никак не зависит от внешних факторов типа поломок компа, отключения света – в этом случае изменение в БД не произойдет).

Транзакции различают:

1. Явные – начинаются командой **Begin Tran** или **Begin Transaction**, заканчиваются при помощи команд (подтверждение транзакции) **Commit Tran** или **Commit Word**, отмена транзакции **Rollback Tran**, или **Rollback Transaction**, или **Rollback Word**.
2. Неявные – команда Set Implicit\_transaction ON, отмена неявной транзакции командой Set Implicit\_transaction OFF
3. Автоматические – это транзакции, относящиеся к единичной команде SQL (если при удалении нескольких строк где-то произойдет ошибка то ничего не удалится)

***ОБЪЯВЛЕНИЕ ПЕРЕМЕННЫХ В MS SQL SERVER. СИСТЕМНАЯ ПЕРЕМЕННАЯ @@error.***

**DECLARE @имя\_переменной тип\_данных;**

Например:

*DECLARE @a INT;*

*DECLARE @price NUMERIC(8, 2);*

*DECLARE @title VARCHAR(128);*

***Присвоение значения переменным:***

**SET @имя\_переменной = значение;**

Например:

*DECLARE @a INT;*

*SET @a = 10;*

*SELECT @a;*

*DECLARE @title VARCHAR(128);*

*SET @title = ‘Microsoft Windows Forever’;*

*SELECT @title;*

Системная переменная @@error содержит код последней операции SQL (если 0 то без ошибок, если 1 есть ошибка)

Точка сохранения устанавливается при помощи синтаксиса:

***SAVE TRNSACTION имя\_точки\_сохранения***

Для отката того, что было после этой точки сохранения

***ROLLBACK TRANSACTION имя\_точки\_сохранения***

***COMMIT TRANSACTION***

Если не использовать имя точки сохранения будет отменена вся транзакция.

***КОРРЕЛИРУЮЩИЕ ПОДЗАПРОСЫ***

Коррелирующий (связанный) подзапрос использует таблицу верхнего (объемлющего) запроса.