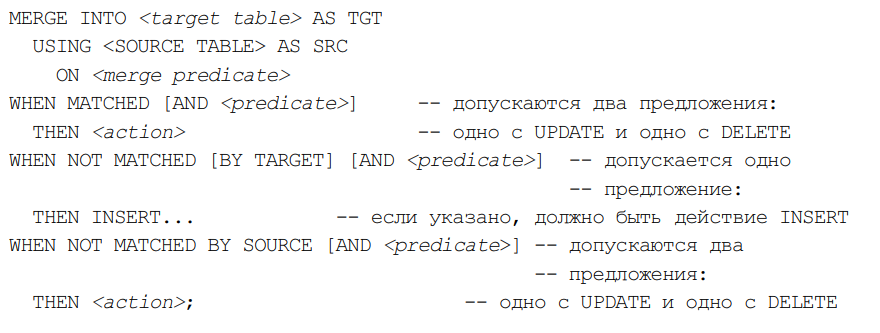
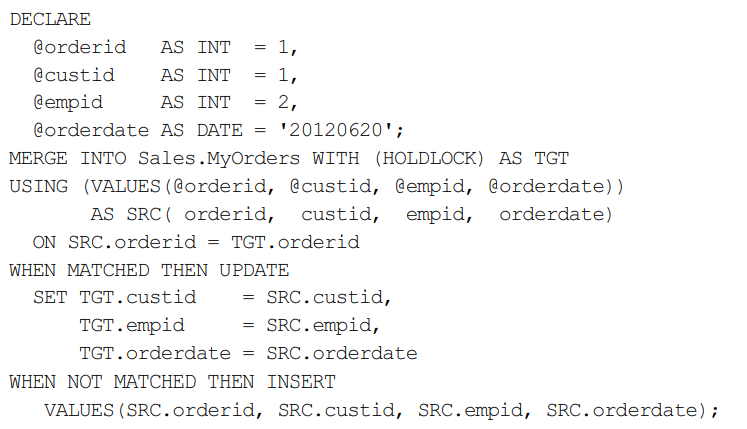
# Merge

С помощью инструкции MERGE выполняется слияние данных из исходной таблицы или табличного выражения и целевой таблицы с помещением их в целевую таблицу

1. у вас есть таблица, которая не обновляется напрямую вашим приложением; вместо этого, вы периодически получаете диапазон изменений от внешней системы. Сначала вы загружаете этот диапазон изменений в промежуточную таблицу и затем используете эту промежуточную таблицу в качестве источника для операции слияния с целевой таблицей.
2. Как пример сценария работы с хранилищем данных, предположим, что вы поддерживаете агрегированные представления данных в своем хранилище данных. Используя инструкцию MERGE, вы можете применять изменения, которые происходят в строках детализации, в агрегированный формат.



->



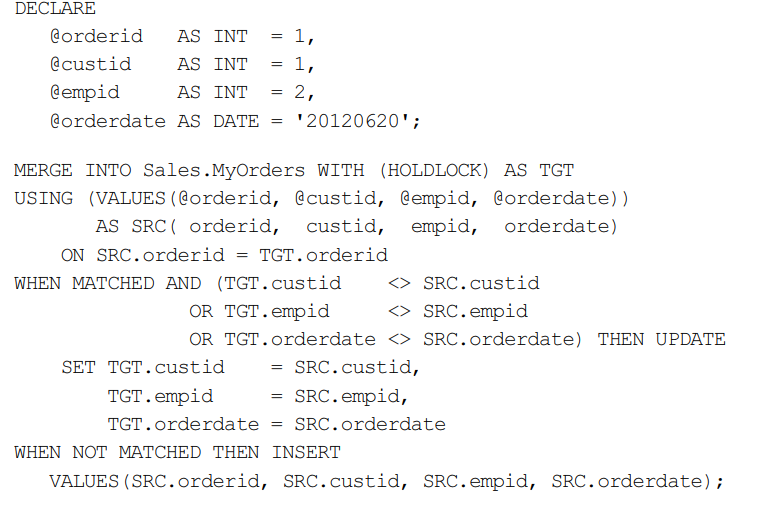
***When Matched*** есть в src и в tgt. разрешенными являются операции UPDATE и DELETE

***When not Matched*** есть в src и нет в tgt. Можно insert

***WHEN NOT MATCHED BY SOURCE*** оставшиеся строки в tgt, которых нет в src. Можно UPDATE, либо и DELETE, но не INSERT.

## дополнительный предикат

WHEN в инструкции MERGE разрешает дополнительный предикат, который должен принимать значение "истина" для применения соответствующего действия

******

***<> это неравенство***

# SEQUENCE

## Свойства

* INCREMENT BY — значение приращения, по умолчанию равно 1;
* MINVALUE — минимальное поддерживаемое значение. Значением по умолчанию является минимальное значение, установленное для указанного типа данных. Например, для типа INT это будет значение –2 147 483 648;
* MAXVALUE — максимальное поддерживаемое значение; значением по умолчанию является максимальное значение, установленное для указанного типа данных; Другие виды модификации данных
* CYCLE | NO CYCLE — определяет, разрешено ли для последовательности циклическое повторение; значение по умолчанию — NO CYCLE;
* START WITH — начальное значение последовательности; по умолчанию устанавливается в MINVALUE для возрастающей последовательности (положительное приращение) и MAXVALUE для убывающей последовательности.
* CACHE N/ NO CACHE - Если установлен параметр NO CACHE, SQL Server должен делать запись на диск для каждого запроса нового значения последовательности. Но с кэшированием производительность намного выше. По умолчанию в момент написания этого курса значение кэша устанавливалось равным 50 - ALTER SEQUENCE Sales.SeqOrderIDs CACHE 100;
* Oбъект последовательности не гарантирует отсутствие разрывов в данных. происходит сбой транзакции, изменение значения последовательности не отменяется
* Можно обновлять столбцы с помощью инструкции UPDATE, используя результаты функции NEXT VALUE FOR.
* Инструкция TRUNCATE не сбрасывает текущее значение объекта последовательности,

## usage

CREATE SEQUENCE Sales.SeqOrderIDs AS INT

MINVALUE 1

CYCLE;

….

SELECT NEXT VALUE FOR Sales.SeqOrderIDs;

…

ALTER SEQUENCE Sales.SeqOrderIDs RESTART WITH 1;

…

INSERT INTO Sales.MyOrders(orderid, custid, empid, orderdate) VALUES (NEXT VALUE FOR Sales.SeqOrderIDs, 1, 2, '20120620'), (NEXT VALUE FOR Sales.SeqOrderIDs, 1, 3, '20120620'), (NEXT VALUE FOR Sales.SeqOrderIDs, 2, 2, '20120620');

## Default seq

ALTER TABLE Sales.MyOrders

ADD CONSTRAINT DFT\_MyOrders\_orderid

DEFAULT(NEXT VALUE FOR Sales.SeqOrderIDs) FOR orderid

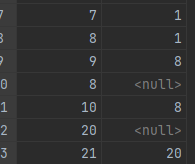
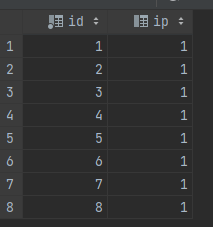
# IDENTITY

* Только один столбец в таблице может иметь свойство IDENTITY, как и PK
* IDENTITY не может гарантировать уникальность изза IDENTITY\_INSERT, нид к PRIMARY KEY
* Свойство IDENTITY привязано к конкретному столбцу в определенной таблице.
* Вы не можете удалить свойство, уже существующее у столбца, а также добавить это свойство существующему столбцу.
* Столбец должен быть определен с этим свойством. Иногда нужно, чтобы ключи не конфликтовали в разных таблицах, но свойство IDENTITY определяется на уровне таблицы.
* Иногда требуется сгенерировать значение до его использования. Это невозможно при использовании свойства IDENTITY. Необходимо вставить строку и только потом получить новое значение с помощью функции.
* Нельзя обновить столбец IDENTITY. Свойство IDENTITY не поддерживает циклическое повторение.
* Инструкция TRUNCATE сбрасывает свойство идентификатора.

## последнее сгенерированное значение

* SCOPE\_IDENTITY возвращает последнее значение идентификатора, сгенерированное в вашем сеансе в данной области;
* функция @@IDENTITY возвращает последнее значение идентификатора, сгенерированное в вашем сеансе независимо от области;
* IDENT\_CURRENT принимает в качестве входа таблицу и возвращает последнее значение идентификатора, сгенерированное во входной таблице независимо от сессии.

## Identity после identity insert будет продолжать с самого большого числа в столбце



# TRUNCATE vs delete

* Truncate меньше записывает в журнал транзакций, В случае инструкции DELETE, SQL Server записывает в журнал реальные данные, которые были удалены.
* Инструкция DELETE не пытается сбросить свойство identity к начальному, если оно установлено для столбца в целевой таблице. Инструкция TRUNCATE делает это.
* Если есть внешний ключ на таблицу где удаляем, delete- allowed, trunc – not allowed
* Инструкция DELETE применяется к таблице, являющейся частью индексированного представления. Инструкция TRUNCATE в таком случае является недопустимой.
* Инструкция DELETE требует разрешений DELETE на целевую таблицу.
* Инструкция TRUNCATE требует разрешений ALTER на целевую таблицу
* Trunc is much faster than delete

# Delete

DELETE FROM Sales.MyOrderDetails

WHERE productid = 11;

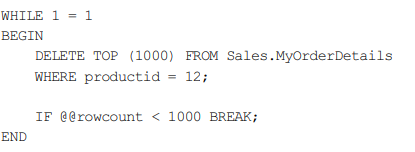
## протоколирование

Инструкция DELETE выполняется с полным протоколированием, и, следовательно, удаление большого количества данных может требовать много времени для своего завершения. Такие объемные удаления могут приводить к значительному увеличению журнала транзакций.

## Блокировки

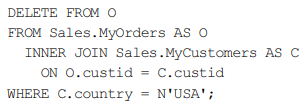
Удаление большого колва строк также могут приводить к укрупнению блокировок, а это означает, что SQL Server укрупняет блокировки мелких фрагментов данных, такие как блокировки строк, до полной блокировки таблицы. Такое укрупнение блокировок может приводить к блокированию доступа ко всем данным таблицы для других процессов.

Для предотвращения названных выше проблем следует разбить большой процесс удаления на меньшие блоки. Этого можно достичь с помощью инструкции DELETE, использующей параметр TOP, который ограничивает количество задействованных строк в цикле. Далее приведен пример реализации такого решения.

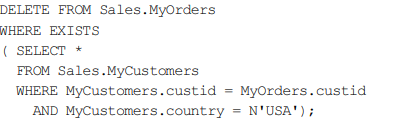


## tricky delete

### where exists



=



## Delete CTE

WITH OldestOrders AS  
( SELECT TOP (100) \*  
 FROM Sales.Orders  
 ORDER BY orderdate )  
DELETE FROM OldestOrders;

# Update

## Simple update

update dbo.TestTable  
set name = 'asda'  
where id %2=1

## Update from () subTable

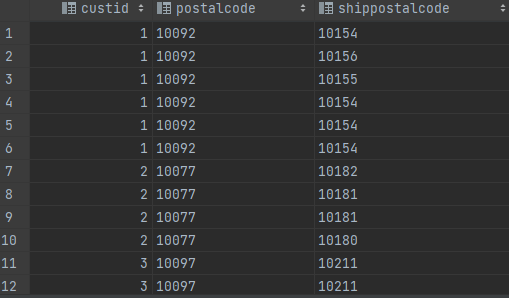
UPDATE OD  
 SET OD.discount += 0.05  
FROM Sales.Customers AS C  
 INNER JOIN Sales.Orders AS O  
 ON C.custid = O.custid  
 INNER JOIN Sales.OrderDetails AS OD  
 ON O.orderid = OD.orderid  
WHERE C.country = N'Norway';

## Недетерминированная UPDATE

Короче я хз че не так там, спросить

когда несколько исходных строк соответствуют одной целевой строке, наапример:

SELECT C.custid, C.postalcode, O.shippostalcode  
FROM Sales.Customers AS C  
 INNER JOIN Sales.Orders AS O  
 ON C.custid = O.custid  
ORDER BY C.custid;



Типо если мы захотим из этого списка обновить кастомера

## UPDATE и принцип единовременности

принцип означает, что все выражения, которые появляются в той же фазе логической обработки запроса, концептуально оцениваются в один и тот же момент времени.

# Inserts

## INSERT VALUES

С помощью инструкции INSERT VALUES можно вставить одну или более строк в целевую таблицу, опираясь на выражения значений

INSERT INTO Sales.MyOrders(custid, empid, orderdate, shipcountry, freight)  
 VALUES(2, 19, '20120620', N'USA', 30.00);

Указание имен целевых столбцов после имени таблицы является необязательным, но считается наиболее правильным

insert into dbo.TestTable  
values ('sda',1,1);

### identity insert

set identity\_insert dbo.TestTable on  
insert into dbo.TestTable (id, name, level, fulllevel)  
values (12313, 'sda',1,1);  
set identity\_insert dbo.TestTable off

то что pk не влияет, главное чтобы не было повторений значений

## INSERT SELECT

Инструкция INSERT SELECT вставляет результирующий набор, возвращенный запросом, в указанную целевую таблицу

SET IDENTITY\_INSERT Sales.Orders ON;  
INSERT INTO Sales.Orders(orderid, custid, empid, orderdate, shipcountry,freight)  
 SELECT orderid, custid, empid, orderdate, shipcountry, freight  
 FROM Sales.Orders  
 WHERE shipcountry = N'Norway';  
SET IDENTITY\_INSERT Sales.Orders OFF;

## INSERT EXEC

С помощью инструкции INSERT EXEC можно вставлять результирующий набор (или наборы), возвращенный динамическим пакетом или хранимой процедурой в указанную целевую таблицу

Процедура:

IF OBJECT\_ID('Sales.OrdersForCountry', 'P') IS NOT NULL  
 DROP PROC Sales.*OrdersForCountry*;  
GO  
CREATE PROC Sales.*OrdersForCountry* @country AS NVARCHAR(15)  
AS  
SELECT orderid, custid, empid, orderdate, shipcountry, freight  
FROM Sales.Orders  
WHERE shipcountry = @country;  
GO  
  
  
SET IDENTITY\_INSERT Sales.Orders ON;  
INSERT INTO Sales.Orders(orderid, custid, empid, orderdate, shipcountry,  
freight)  
 EXEC Sales.*OrdersForCountry* @country = N'Portugal';  
SET IDENTITY\_INSERT Sales.Orders OFF;

## SELECT INTO

Эта инструкция создает целевую таблицу, опираясь на определение источника, и вставляет результирующие строки из запроса в эту таблицу.

IF OBJECT\_ID('Sales.MyOrders', 'U') IS NOT NULL DROP TABLE Sales.MyOrders;  
SELECT orderid, custid, orderdate, shipcountry, freight  
INTO Sales.MyOrders  
FROM Sales.Orders  
WHERE shipcountry = N'Norway';

### Что копируется

+

**NULL**

1. Если необходимо, чтобы целевой столбец был определен как недопускающий значения NULL, требуется использовать функцию ISNULL: ISNULL(orderid + 0, -1) AS orderid.
2. Аналогично, исходный столбец custid определен в источнике как разрешающий значения NULL. Чтобы целевой столбец был определен как NOT NULL, следует использовать выражение ISNULL(custid, -1) AS custid.

**IDENTITY**

Если вы хотите, чтобы Вставка, обновление и удаление данных 323 у целевого столбца не было этого свойства, вам следует выполнить некоторую обработку, например, orderid + 0 AS orderid. Обратите внимание, после применения этой обработки целевой столбец будет определен как допускающий значения NULL.

-

**defaults**

**Индексы**

**Ограничения**

**Триггеры**

**Разрешения**

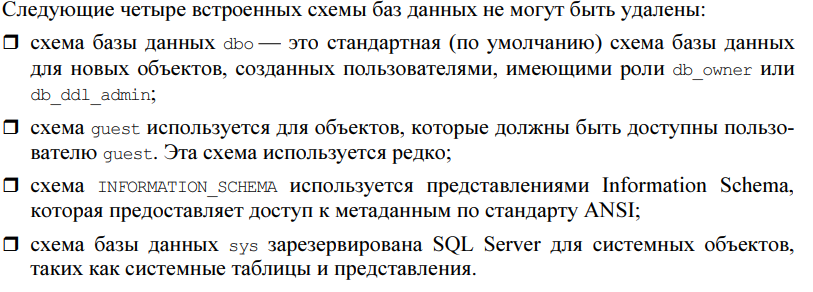
### Warnings

У инструкции SELECT INTO имеются и недостатки.

невозможно контролировать — например, файловую группу целевой таблицы.

Также следует помнить, что инструкция SELECT INTO затрагивает как создание таблицы, так и заполнение ее данными. Это означает, что и относящиеся к целевой таблице метаданные, и **данные** **монопольно блокируются** до тех пор, пока не будет закончена транзакция SELECT INTO. В результате можно столкнуться c блокировками, вызванными конфликтами, связанными с доступом как к данным, так и к метаданным.

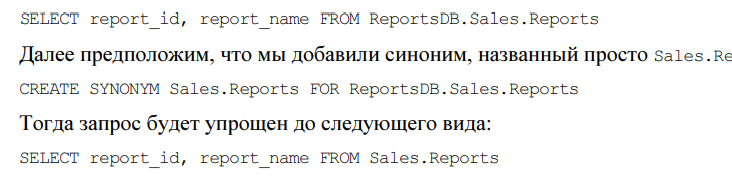
# Sys/dbo/guest/information\_schema



# Синонимы

CREATE SYNONYM dbo.Categories FOR Production.Categories;

Затем конечный пользователь может делать выборку из Categories без необходимости указывать схему. SELECT categoryid, categoryname, description FROM Categories;

* Чтобы изменить синоним, его надо сначала удалить, а затем снова создать.
* Можно модифицировать данные с помощью синонима, но нельзя изменить базовый объект
* Синонимы не предоставляют никаких метаданных объектов, на которые они ссылаются.
* Синоним не хранит никакого кода T-SQL и никаких данных. Он хранит только объект, на который выполняется ссылка.
* Цепочки синонимов не разрешены; синоним не может ссылаться на другой
* Синонимы могут ссылаться на объекты в других базах данных, а также на объекты, на которые ссылаются связанные серверы 

## Синонимы и ссылки на несуществующие объекты

Можно создать синоним, даже если объект, на который выполняется ссылка, не существует

Недостатком этого свойства является то, что не существует такого компонента, как WITH SCHEMABINDING: если вы удаляете объект в базе данных, он будет удален, независимо от того, ссылается на него синоним или нет. Любые синонимы, ссылающиеся на объект, фактически являются потерянными объектами; они прекращают работать, когда кто-то пытается их использовать.

## Разрешение

Для создания синонима необходимо иметь разрешение CREATE SYNONYM, которое наследуется из разрешения CONTROL SERVER.

# Functions

* CREATE FUNCTION должна быть первой инструкцией в пакетном запросе

CREATE FUNCTION Production.*GetTopProducts*(@supplierid AS INT, @n AS BIGINT)  
RETURNS TABLE  
AS  
RETURN  
 SELECT productid, productname, unitprice  
 FROM Production.Products  
 WHERE supplierid = @supplierid  
 ORDER BY unitprice, productid  
 OFFSET 0 ROWS FETCH FIRST @n ROWS ONLY;  
GO

# View

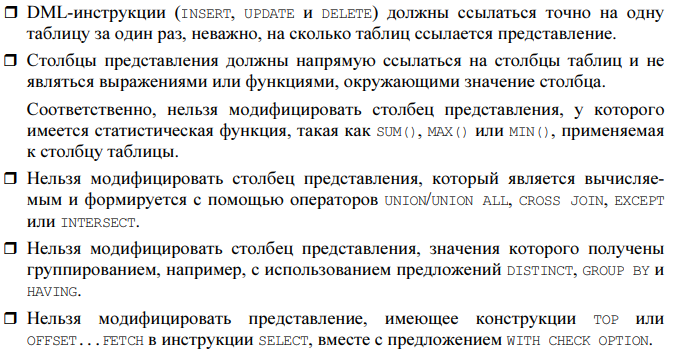
* в базе данных сохраняется не результирующий набор представления, а только лишь его определение
* не принимают параметры

CREATE VIEW Sales.RankedProducts  
AS  
SELECT categoryid, productid, productname, unitprice  
FROM Production.Products;

CREATE VIEW Sales.OrderTotalsByYear WITH SCHEMABINDING

## Модификация данных с помощью представления

Можно обновлять, вставлять или удалять данные с помощью представления вместо того, чтобы напрямую ссылаться на базовые таблицы, но при этом следует учитывать множество ограничений, приведенных далее.



## View хранящее дату (Un clust idx)

о можно создать уникальный кластеризованный индекс на представлении и таким образом материализовать данные. В этом случае хранится не только определение представления. На диске сохраняются фактические результаты запроса, в структуре кластеризованного индекса. Для того чтобы быть проиндексированным, представление должно отвечать нескольким важным ограничениям.

## Delete default column

Если есть default на колонке то не удалить тк это ограничение

Надо удалить ограничение и тогда толькоо колонкку

Тогда если во вьюхе было эта колонка то она тоже делетнется

## Ограничения view

Нельзя order by при объявлении

Представление не может создать таблицу, как постоянную, так и временную. Иными словами, в представлении нельзя использовать синтаксис SELECT/INTO.

Представление может ссылаться только на постоянные таблицы; оно не может ссылаться на временную таблицу()

## WITH SCHEMABINDING

SCHEMABINDING, который гарантирует, что структуры базовых таблиц не могут быть изменены без удаления представления (не имеется ввиду insert)

Не получится дропнуть колоку если привязана или вычисляемая

Но независимые от этого изи

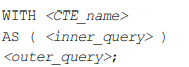
## WITH VIEW\_METADATA

Предложение WITH VIEW\_METADATA, когда оно указано, возвращает метаданные представления, а не базовой таблицы.

## With check option

. Если вы определяете представление с ограничением фильтра в предложении WHERE инструкции SELECT, а затем измените строки таблицы с помощью представления, то можете изменить некоторое значение так, что задействованная строка уже не будет удовлетворять фильтру предложения WHERE. Возможно даже обновление строк, которые выходят за пределы области фильтра. Предложение WITH CHECK OPTION препятствует подобному исчезновению строк при обновлении через представление, а также ограничивает модификации только строками, которые удовлетворяют критериям фильтра

# CTE



with top10prod as (  
select top 10 Products.productid, Products.productname, Products.categoryid, c.categoryname, c.description  
from Production.Products inner join Production.Categories C on C.categoryid = Products.categoryid  
order by Products.productid)

# DDL

## Cant alter identity

## Default

unitprice MONEY NOT NULL CONSTRAINT DFT\_Products\_unitprice DEFAULT(0)

### alter

alter table dbo.TestTable  
add constraint def\_name  
default 'bic' for name

## Имя с пробеламы

SET QUOTED\_IDENTIFIER on

>

CREATE TABLE Production.[Yesterday's News] / CREATE TABLE Production."Tomorrow's Schedule"

## Вычисляемые столбцы

create table TestTable  
(  
 id int identity,  
 fulllevel221131 as [TestTable].[fulllevel] \* 12  
)

## сжатие таблиц

Существуют два уровня сжатия данных:

ROW — при сжатии на уровне строк SQL Server применяет более компактный формат хранения к каждой строке в таблице;

PAGE — сжатие на уровне страниц включает в себя сжатие на уровне строк плюс дополнительные алгоритмы сжатия, которые могут выполняться на уровне страницы.

Следующая команда добавляет сжатие на уровне строк для таблицы Production.OrderDetails как часть инструкции

CREATE TABLE Sales.OrderDetails ( orderid INT NOT NULL, ... )

WITH (DATA\_COMPRESSION = ROW);

ALTER TABLE Sales.OrderDetails

REBUILD WITH (DATA\_COMPRESSION = PAGE);

# XML

## FOR XML RAW

Будет фрагмент кода xml, то есть не будет открывающего и закрывающего тэга

<row orderid="10835" custid="1" empid="1" />

<row orderid="10952" custid="1" empid="1" />

<row orderid="11011" custid="1" empid="3" />

<row orderid="10692" custid="1" empid="4" />

<row orderid="10702" custid="1" empid="4" />

<row orderid="10643" custid="1" empid="6" />

### Изменить тэг raw(‘cust\_name’)

select Orders.orderid, Orders.custid, Orders.empid

from Sales.Orders

order by Orders.custid, Orders.empid

for xml RAW('custname')

\_\_\_\_\_\_

<cust\_name orderid="10835" custid="1" empid="1" />

<cust\_name orderid="10952" custid="1" empid="1" />

<cust\_name orderid="11011" custid="1" empid="3" />

<cust\_name orderid="10692" custid="1" empid="4" />

<cust\_name orderid="10702" custid="1" empid="4" />

<cust\_name orderid="10643" custid="1" empid="6" />

### Rootname

select Orders.orderid, Orders.custid, Orders.empid

from Sales.Orders

order by Orders.custid, Orders.empid

for xml RAW, root('root\_name')

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

<root\_name>

<row orderid="10835" custid="1" empid="1" />

<row orderid="10952" custid="1" empid="1" />

<row orderid="11011" custid="1" empid="3" />

<row orderid="10692" custid="1" empid="4" />

<row orderid="10702" custid="1" empid="4" />

</root\_name>

### Elements- Сделать root для каждой строки

select Orders.orderid, Orders.custid, Orders.empid

from Sales.Orders

order by Orders.custid, Orders.empid

for xml RAW('custname'), root('root\_name') ,elements

\_\_\_\_\_\_\_\_\_\_\_\_

<root\_name>

<custname>

<orderid>10835</orderid>

<custid>1</custid>

<empid>1</empid>

</custname>

<custname>

<orderid>10952</orderid>

<custid>1</custid>

<empid>1</empid>

</custname>

<custname>

<orderid>11011</orderid>

<custid>1</custid>

<empid>3</empid>

</custname>

## FOR XML auto

select Orders.orderid, Orders.custid, Orders.empid

from Sales.Orders

order by Orders.custid, Orders.empid

for xml auto

\_\_\_\_\_\_\_\_

<Sales.Orders orderid="10835" custid="1" empid="1" />

<Sales.Orders orderid="10952" custid="1" empid="1" />

<Sales.Orders orderid="11011" custid="1" empid="3" />

<Sales.Orders orderid="10692" custid="1" empid="4" />

### Все названия можно менять из запроса

select orderid, custid, empid as ei

from Sales.Orders as SO

order by custid, empid

for xml auto

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

<SO orderid="10835" custid="1" ei="1" />

<SO orderid="10952" custid="1" ei="1" />

<SO orderid="11011" custid="1" ei="3" />

### Elements – same

select orderid, custid, empid as ei

from Sales.Orders as SO

order by custid, empid

for xml auto, elements

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

<SO>

<orderid>10835</orderid>

<custid>1</custid>

<ei>1</ei>

</SO>

<SO>

<orderid>10952</orderid>

<custid>1</custid>

<ei>1</ei>

</SO>

<SO>

### Root – same

## FOR XML PATH

### FOR XML PATH

<row>

<custid>1</custid>

<companyname>Customer NRZBB</companyname>

<orderid>10692</orderid>

<orderdate>2007-10-03T00:00:00</orderdate>

</row>

### FOR XML PATH('customer')

<customer>

<custid>1</custid>

<companyname>Customer NRZBB</companyname>

<orderid>10692</orderid>

<orderdate>2007-10-03T00:00:00</orderdate>

</customer>

## Xml explicit

Не пробовал, но вообще это когда path делает фигню

## XMLNAMESPACE

WITH XMLNAMESPACES('TK461-CustomersOrders' AS co)

SELECT [co:Customer].custid AS [co:custid],

[co:Customer].companyname AS [co:companyname],

[co:Order].orderid AS [co:orderid],

[co:Order].orderdate AS [co:orderdate]

FROM Sales.Customers AS [co:Customer]

INNER JOIN Sales.Orders AS [co:Order]

ON [co:Customer].custid = [co:Order].custid

ORDER BY [co:Customer].custid, [co:Order].orderid

FOR XML AUTO, ELEMENTS, ROOT('CustomersOrders');

<CustomersOrders xmlns:co="TK461-CustomersOrders">

<co:Customer>

<co:custid>1</co:custid>

<co:companyname>Customer NRZBB</co:companyname>

<co:Order>

<co:orderid>10692</co:orderid>

<co:orderdate>2007-10-03T00:00:00</co:orderdate>

</co:Order>

<co:Order>

<co:orderid>10702</co:orderid>

<co:orderdate>2007-10-13T00:00:00</co:orderdate>

</co:Order>

<co:Order>

<co:orderid>10952</co:orderid>

<co:orderdate>2008-03-16T00:00:00</co:orderdate>

</co:Order>

</co:Customer>

<co:Customer>

<co:custid>2</co:custid>

<co:companyname>Customer MLTDN</co:companyname>

<co:Order>

<co:orderid>10308</co:orderid>

<co:orderdate>2006-09-18T00:00:00</co:orderdate>

</co:Order>

<co:Order>

<co:orderid>10926</co:orderid>

<co:orderdate>2008-03-04T00:00:00</co:orderdate>

</co:Order>

</co:Customer>

</CustomersOrders>

## Xquery

Это штука для работы с xml текстом

# Full text search

Глава 6 в бен гане

## Isfullsearch installed

SELECT SERVERPROPERTY('IsFullTextInstalled');

## Как посмотреть Фильтры

Фильтры, которые в терминологии полнотекстового поиска называются iFilters, извлекают текстовую информацию и удаляют в **документе** форматирование. Можно проверить, какие фильтры установлены в вашем экземпляре SQL Server с помощью следующего запроса:

***EXEC sys.sp\_help\_fulltext\_system\_components 'filter';***

Кроме системной хранимой процедуры, для проверки, **какой фильтр установлен на вашем** **экземпляре SQL Server**, можно использовать представление каталога sys.fulltext\_document\_types следующим образом:

**SELECT document\_type, path FROM sys.fulltext\_document\_types**

## Скачать фильтры

Загрузить пакет фильтров Microsoft Office 2010 можно по этому адресу: <http://www.microsoft.com/en-us/download/details.aspx?id=17062>.

## После установки

После установки пакета фильтров на компьютер, необходимо зарегистрировать эти фильтры в SQL Server с помощью следующей команды: EXEC sys.sp\_fulltext\_service 'load\_os\_resources', 1;

## Чек языки фильтров

SELECT lcid, name FROM sys.fulltext\_languages ORDER BY name;

Для каждого полнотекстового индексированного столбца можно указать свой язык. Изменить язык по умолчанию можно с помощью системной процедуры sys.sp\_configure

## Поиск и изменение синонимов

SQL Server ищет синонимы в файле тезауруса. Каждый язык имеет свой XML-файл тезауруса. Файлы тезауруса для экземпляра по умолчанию находятся по адресу SQL\_Server\_install\_ path\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\FTDATA\.

Можно вручную редактировать каждый файл тезауруса и конфигурировать следующие элементы:

* diacritics\_sensitive (с учетом диакритических знаков). Установите этот элемент в значение 0, если язык не использует диакритические знаки, или в значение 1, если использует;
* expansion (расширение). Этот элемент используется для добавления расширяющих слов к искомому слову. Например, можно добавить расширяющее слово "автор" к слову "писатель", чтобы искать также и слово "автор", когда конечный пользователь ищет слово "писатель";
* replacement (замена). Этот элемент следует использовать для определения замещающих слов или терминов для определенного слова или термина. Например, "Windows 2008" может быть замещением для "Win 2k8". В таком случае SQL Server будет искать "Windows 2008", хотя в качестве элемента поиска будет использоваться "Win 2k8".

После редактирования файла тезауруса для определенного языка его нужно загрузить с помощью вызова следующей системной процедуры: EXEC sys.sp\_fulltext\_load\_thesaurus\_file 1033;

## Создание и управление полнотекстовыми каталогами и индексами

### создать полнотекстовой каталог

Полнотекстовые индексы хранятся в полнотекстовых каталогах. Полнотекстовый каталог — это виртуальный объект, контейнер для полнотекстовых индексов. Как виртуальный объект, он не принадлежит ни к одной файловой группе. Далее приведен синтаксис для создания полнотекстового каталога.

CREATE FULLTEXT CATALOG catalog\_name  
 [ON FILEGROUP filegroup ]  
 [IN PATH 'rootpath']  
 [WITH <catalog\_option>]  
 [AS DEFAULT]  
 [AUTHORIZATION owner\_name ]  
<catalog\_option> ::= ACCENT\_SENSITIVITY = {ON|OFF}

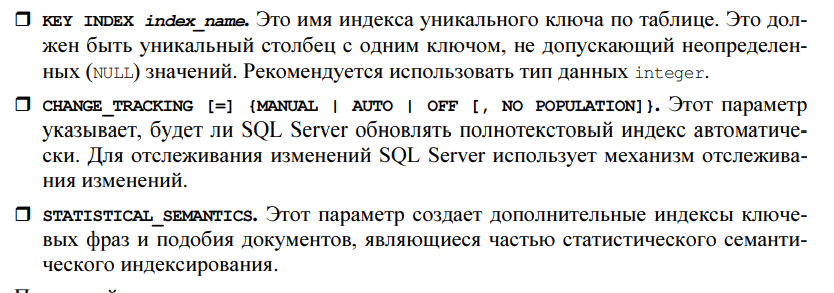
Параметры **ON FILEGROUP** и **IN PATH** предназначены для обеспечения обратной совместимости с SQL Server 2008 и более ранними версиями и не влияют на SQL Server 2012

о. Параметр ACCENT\_SENSITIVITY определяет, учитывают ли полнотекстовые индексы в этом каталоге диакритические знаки или нет. Если вы позже измените этот параметр, вам нужно будет перестроить все полнотекстовые индексы в данном каталоге.

### создать полнотекстовой индекс

После того как полнотекстовый каталог создан, можно создать соответствующие полнотекстовые индексы. Далее приведен синтаксис для создания полнотекстового индекса.

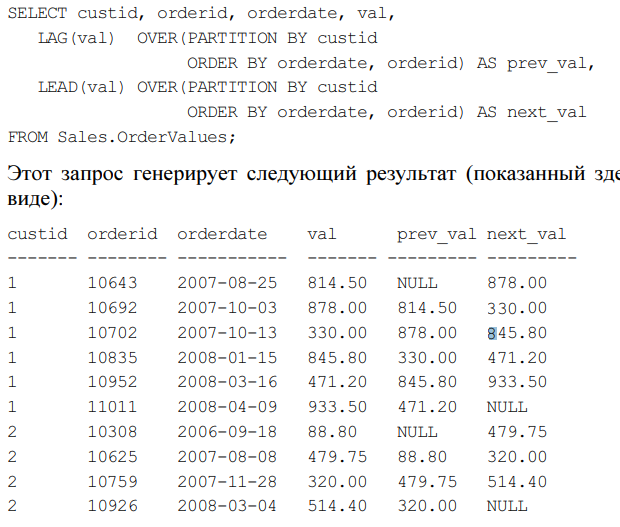
CREATE FULLTEXT INDEX ON table\_name  
 [ ( { column\_name  
 [ TYPE COLUMN type\_column\_name ]  
 [ LANGUAGE language\_term ]  
 [ STATISTICAL\_SEMANTICS ]  
 } [,...n] )]   
 KEY INDEX index\_name  
 [ ON <catalog\_filegroup\_option> ]  
 [ WITH [ ( ] <with\_option> [,...n] [ ) ] ]  
[;]  
<catalog\_filegroup\_option>::=  
 { fulltext\_catalog\_name  
 | ( fulltext\_catalog\_name, FILEGROUP filegroup\_name)  
 | ( FILEGROUP filegroup\_name, fulltext\_catalog\_name)  
 | ( FILEGROUP filegroup\_name )  
 }  
<with\_option>::=  
 { CHANGE\_TRACKING [ = ] { MANUAL | AUTO | OFF [, NO POPULATION ] }  
 | STOPLIST [ = ] { OFF | SYSTEM | stoplist\_name }  
 | SEARCH PROPERTY LIST [ = ] property\_list\_name  
 }



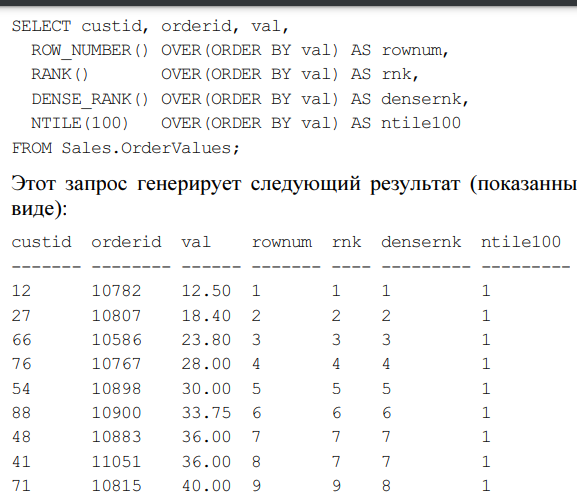
STATISTICAL\_SEMANTICS дает возможность глубже анализировать документы путем извлечения и индексирования статистически релевантных ключевых фраз. Полнотекстовый поиск использует эти ключевые фразы для идентификации и индексирования подобных или связанных документов

# Lead/lag

Возращают prev/post val



# Ранжирующие функции – rank/rownumber/dense/ntile

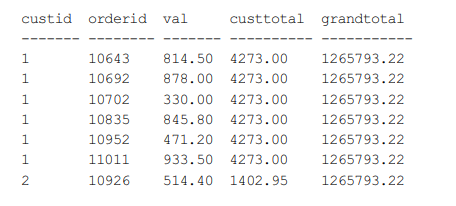


# Aggregate+over partition

SELECT custid, orderid, val,  
 SUM(val) OVER(PARTITION BY custid) AS custtotal,  
 SUM(OrderValues.custid) OVER() AS custidSum  
FROM Sales.OrderValues;

SUM(val) OVER(PARTITION BY custid) AS custtotal – для каждого custid вывести сумму val

SUM(OrderValues.custid) OVER() AS custidSum – сумма всех custid



CСари в конце custTotal другой

## BETWEEN UNBOUNDED PRECEDING / CURRENT ROW

Это чтобы брать не все custid, а только определенные

SELECT custid, orderid, orderdate, val,  
 SUM(val) OVER(PARTITION BY custid  
 ORDER BY orderdate, orderid  
 ROWS BETWEEN 2 PRECEDING  
 AND CURRENT ROW) AS runningtotal  
FROM Sales.OrderValues;

То есть делаем rows between Потом промежуток

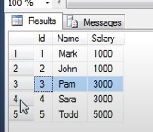
Мб current row / 2 preceding/ 2 following / UNBOUNDED(до конца секции custid)

rows between UNBOUNDED PRECEDING and current row

=

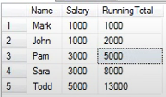
rows UNBOUNDED PRECEDING

## range vs raws

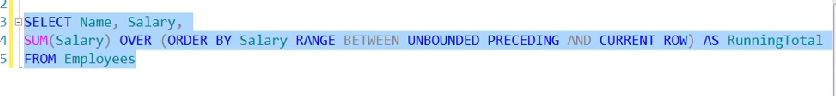


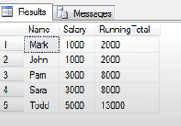
Rows:





Range делает для одинаковых всю их сумму:

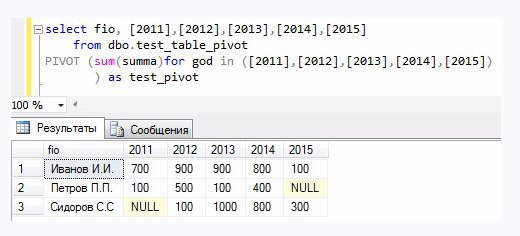
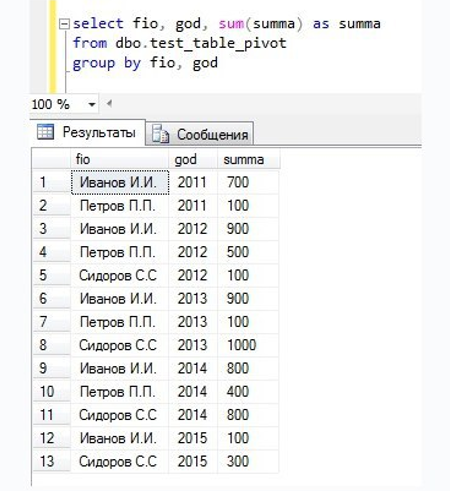




# Pivot/unpivot

## Pivot

Pivot = повышение = Из значений колонки сделать название



Pivot(AGGREGATE(у нас сумма ‘summa’) for СТОЛБЕЦ(у нас god) in (конкретные значения god) ) **P**

Название Р в конце обязательно!!!

unPIVOT(агрегатная функция - where code = (select max(code)

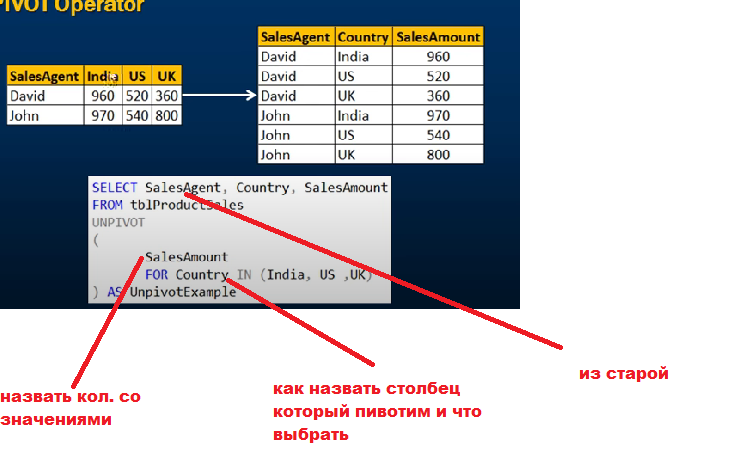
for cтолбец, содержащий значения, которые станут именами столбцов - code

IN ([значения по горизонтали],…) - model speed ram hd cd price

)AS псевдоним таблицы (обязательно) - wnsdjnajda

## Unpivot

unPivot = unПовышение = Сделать из названий колонок сделать колонку со значениями названий



# Groping sets

## Groping sets

SELECT shipperid, YEAR(shippeddate) AS shipyear, COUNT(\*) AS numorders  
FROM Sales.Orders  
WHERE shippeddate IS NOT NULL  
GROUP BY GROUPING SETS  
 ( (shipperid, YEAR(shippeddate)),  
 (shipperid)  
 )  
order by Orders.shipperid, YEAR(Orders.shippeddate), numorders

Этот код эквивалентен

SELECT shipperid, null, COUNT(\*) AS numorders  
FROM Sales.Orders  
WHERE shippeddate IS NOT NULL  
GROUP BY shipperid  
union  
SELECT shipperid, year(Orders.shippeddate), COUNT(\*) AS numorders  
FROM Sales.Orders  
WHERE shippeddate IS NOT NULL  
GROUP BY shipperid, year(Orders.shippeddate)  
order by 1, 2

### Если пустые скобки

То это будет значить

SELECT shipperid, YEAR(shippeddate) AS shipyear, COUNT(\*) AS numorders  
FROM Sales.Orders  
WHERE shippeddate IS NOT NULL  
GROUP BY GROUPING SETS  
 ( (shipperid, YEAR(shippeddate)),  
 ()  
 )  
order by Orders.shipperid, YEAR(Orders.shippeddate), numorders

Плюс еще простой

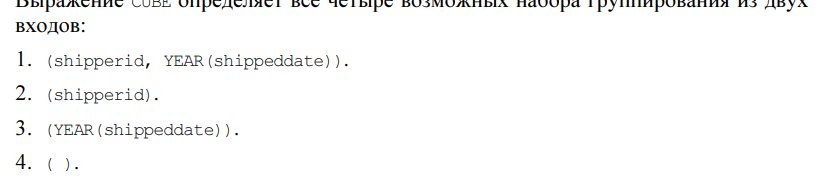
SELECT count(\*)  
FROM Sales.Orders  
WHERE shippeddate IS NOT NULL

Вместо 1 и 2 будет null соответственно

## Cube

SELECT shipperid, YEAR(shippeddate) AS shipyear, COUNT(\*) AS numorders  
FROM Sales.Orders  
GROUP BY CUBE(shipperid, YEAR(shippeddate));

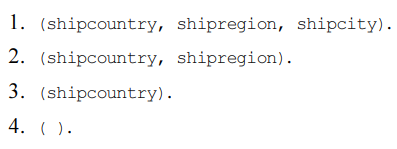
Eq



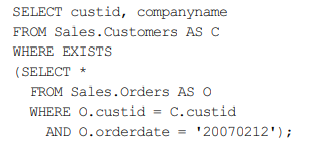
## Rollup

SELECT shipcountry, shipregion, shipcity, COUNT(\*) AS numorders  
FROM Sales.Orders  
GROUP BY ROLLUP(shipcountry, shipregion, shipcity)

Eq



# Exists

Предикат EXISTS принимает подзапрос на вход и возвращает значение "истина", когда подзапрос возвращает хотя бы одну строку, в противном случае — "ложь". 

Или вот еще пример, можно забить

with top10prod as (  
select top 10 Products.productid, Products.productname, Products.categoryid, c.categoryname, c.description  
from Production.Products inner join Production.Categories C on C.categoryid = Products.categoryid  
order by Products.productid)  
--12768  
  
select \*  
from Production.Categories c  
where exists(select \*  
from top10prod p  
 where p.categoryid = c.categoryid)

# TOP/OFFSET-FETCH

## top

TOP позволяет фильтровать запрашиваемое **количество или процент** строк в результате запроса на основе указанной сортировки

Top percent округляет в большую часть, 25 процентов от 10 записей выведет 3

Чтобы показать что мы не хотим отсортировать и не ошиблись можно написать

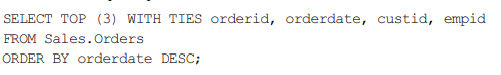
order by (select null)

order by (null)не прокатит

## With ties

включить все связи с последней строкой

Но что если существуют другие строки в результирующем наборе без параметра TOP, которые имеют такую же дату заказа, что и последняя строка в нашем примере?



## OFFSET-FETCH

OFFSET 50 ROWS FETCH NEXT 25 ROWS ONLY;

пропускает 50 строк и выбирает следующие 25 строк.

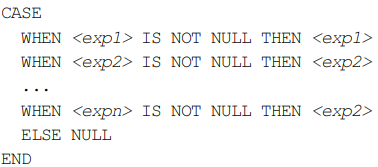
### требует присутствия предложения ORDER BY.

параметр OFFSET...FETCH требует присутствия предложения ORDER BY.

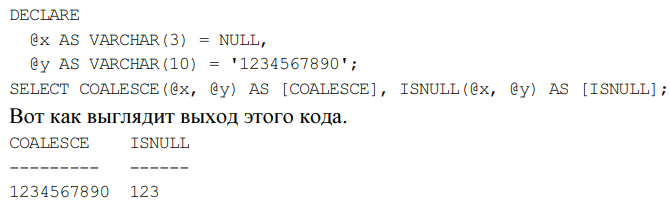
**!OFFSET...FETCH не поддерживает имеющиеся у TOP возможности, такие как PERCENT и WITH TIES.**

# COALESCE/IS\_Null

Функция COALESCE принимает список выражений на вход и возвращает первое выражение, не равное NULL, или NULL, если все выражения имеют значение NULL. Например, функция COALESCE(NULL, 'x', 'y') возвращает 'x'. В более общем смысле, функция COALESCE(, , ..., ) аналогична следующему:



---



**(!) тип выражения COALESCE определяется возвращаемым элементом, тогда как тип выражения ISNULL определяется первыми выходными данными, здесь varchar(3), поэтмоу и 3выводится**

# типы данных для ключей

## Свойство столбца идентификаторов

## Объект последовательности

## Непоследовательные GUID

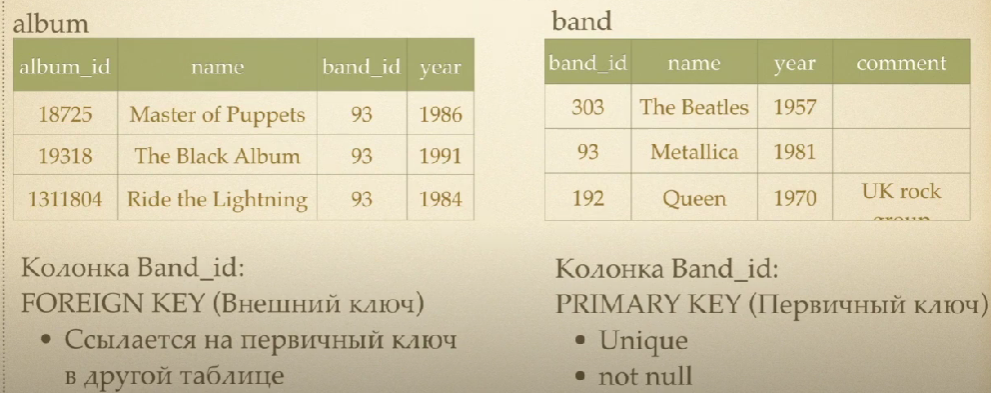
## Последовательные GUID

## функции NEWID и NEWSEQUENTIALID

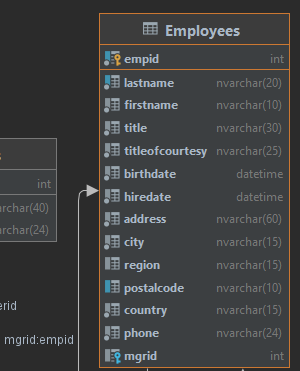
## Пользовательские решения

# Foreign key

Есть табла альбом и банда, в альбоме id банды, и нам надо чтобы среди банд обязательно был id из альбома, чтобы таблица не ссылалась на не существующую банду, это и делается через внешний ключ, он говорит, что такое значение обязательно будет в band’e.



Он может ссылаться на ту же таблицу где он и находится, например есть таблица работников и ктото из них управляющий над другим, тогда можно сделать employee id + manager id, и manager id будет foreign key on empId



# CAST, CONVERT, PARSE

SELECT CAST('abc' AS INT);

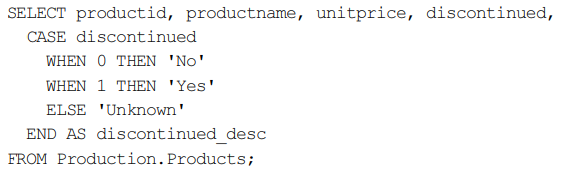
CONVERT(DATE, '1/2/2012', 101), также можно просто convert(datetime2 ,'1947-09-19')

PARSE('1/2/2012' AS DATE USING 'en-US')

## Try

Возращает null если ошибка

# case



# Функции со строками

## Left/right

LEFT('abcde', 3) возвращает подстроку 'abc', а функция RIGHT('abcde', 3) возвратит 'cde'.

## CHARINDEX

возвращает позицию первого вхождения строки, представленной первым аргументом, в строке, представленной вторым аргументом. Например, функция CHARINDEX(' ','Itzik Ben-Gan') ищет первое вхождение пробела во втором входном аргументе и возвращает 5 в данном случае. Заметьте, можно указать третий аргумент, обозначающий позицию начала поиска.

## PATINDEX

если с помощью функции CHARINDEX вы ищете постоянную строку, то функция PATINDEX выполняет поиск шаблона. Шаблон очень похож на шаблон LIKE, который вам, вероятно, знаком, где можно использовать подстановочные символы, такие как % для любой строки, \_ для одного символа и квадратные скобки ([]), представляющие один символ из определенного списка или диапазона

Например, выражение PATINDEX('%[0-9]%', 'abcd123efgh') ищет первое вхождение цифры (символ в диапазоне 0—9) во втором выражении и возвращает позицию 6.

## LEN и DATALENGTH

**LEN** возвращает длину входной строки в виде определенного количества символов . Если имеются хвостовые пробелы, функция LEN их удаляет.

**DATALENGTH** возвращает длину входного значения в виде числа байтов. DATALENGTH не удаляет хвостовые пробелы.

## REPLACE

REPLACE('.1.2.3.', '.', '/') заменяет все вхождения точки (.) слешем (/), возвращая строку '/1/2/3/'.

## REPLICATE

REPLICATE('0', 10) повторяет строку '0' десять раз, возвращая '0000000000'.

## STUFF

Данная строка 1 арг

Позиция с которой удаляются элементы 2 агр

Сколько - 3

4 аргумент –строка в конце которой вставится все

Например, функция STUFF(',x,y,z', 1, 1, 'aaa') удаляет первый символ во входной строке и возвращает 'aaax,y,z'.

## UPPER, LOWER, LTRIM, RTRIM

Ясно

отсечение ведущих пробелов

## FORMAT

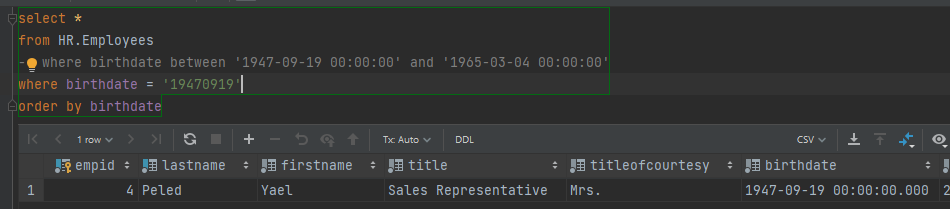
## parse

## cast

# Функции даты и времени

## Поиск по дате

Формат должен быть определенный:



Потому что если написать

where birthdate = '1947-09-19'

то будет ошибка, он не скастит

## GETDATE

Функция GETDATE — это собственная функция языка T-SQL, возвращающая текущую дату и время на экземпляре SQL Server, к которому вы подключены, с типом данных DATETIME

## CURRENT\_TIMESTAMP

Функция CURRENT\_TIMESTAMP делает то же самое

## GETUTCDATE

Функция GETUTCDATE возвращает текущую дату и время в терминах UTC как тип данных DATETIME;

## SYSUTCDATETIME

функция SYSUTCDATETIME делает то же самое, но только возвращает результат с более точным типом данных DATETIME2.

## SYSDATETIME

Функции SYSDATETIME и SYSDATETIMEOFFSET подобны, но они возвращают значения с более точными типами данных — DATETIME2 и DATETIMEOFFSET (включая смещение) соответственно

## SYSDATETIMEOFFSET

Возвращает значение типа **datetimeoffset(7)**, которое содержит дату и время компьютера, на котором запущен экземпляр SQL Server. Смещение часового пояса включается.

SYSDATETIME() - 2007-04-30 13:10:02.0474381

SYSDATETIMEOFFSET() - 2007-04-30 13:10:02.0474381 -07:00

## CURRENT\_TIMESTAMP

returns the current database system timestamp as a **datetime** value, without the database time zone offset. CURRENT\_TIMESTAMP derives this value from the operating system of the computer on which the instance of SQL Server runs.

## DATEPART

DATEPART(month, '20120212') возвращает 2

## YEAR, MONTH и DAY

как сокращения функции DATEPART

## DATENAME

подобна функции DATEPART, но она возвращает имя части выражения в виде символьной строки. Обратите внимание, эта функция зависима от языка. Это означает, что если действующий язык сессии us\_english, выражение DATENAME(month, '20120212') возвращает 'February', а для итальянского языка она возвратит 'febbraio'.

## DATEFROMPARTS

## DATETIME2FROMPARTS

## DATETIMEFROMPARTS

## DATETIMEOFFSETFROMPARTS

## SMALLDATETIMEFROMPARTS

## TIMEFROMPARTS.

## EOMONTH

вычисляет соответствующую дату конца месяца для входной величины даты и времени. Например, предположим, что сегодня 12 февраля 2012 г. Выражение EOMONTH(SYSDATETIME()) тогда возвращает дату '2012-02-29'.

## DATEADD

DATEADD(year, 1, '20120212')

## DATEDIFF

DATEDIFF(day, '20110212', '20120212')

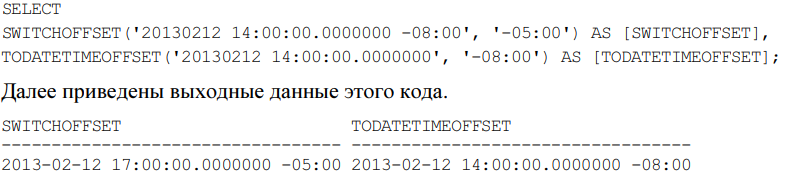
## SWITCHOFFSET

С помощью функции SWITCHOFFSET можно возвратить входное значение DATETIMEOFFSET с требуемым смещением. Например, рассмотрим выражение SWITCHOFFSET(SYSDATETIMEOFFSET(), '-08:00'). Независимо от смещения экземпляра системы, к которому вы подключены, вы хотите получить значение текущей даты и времени со смещением '-08:00'. Если системное смещение составляет, скажем, '-05:00', функция сделает поправку на эту величину и вычтет 3 часа из входного значения.

## TODATETIMEOFFSET

Ее применяют для построения значения DATETIMEOFFSET из двух входных величин

TODATETIMEOFFSET(dt, theoffset)



# Types

## Sources

<https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql-server-data-type-mappings?redirectedfrom=MSDN>

## Bit

**Values:** 1, 0, or NULL

>1=true, <0=true, 0 = false

## Tinyint

**Values:** 0 to 255 (2^8-1) – **1 byte**

## Smallint

**Values:** -2^15 (-32,768) to 2^15-1 (32,767) – **2 bytes**

## Int

**Values:** -2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647) – **4 bytes**

## Bigint

**Values:** -2^63 (-9,223,372,036,854,775,808) to 2^63-1 (97,223,372,036,854,775,807) **8 bytes**

## Smallmoney

**Values:** – 214,748.3648 to 214,748.3647 - **4 bytes like INT, but .XXXX**

Ref to Decimal

## Money

Values: -922,337,203,685,477.5808 to 922,337,203,685,477.5807 - **Same as bigint, 8b, but with .XXXX**

## numeric[ (p[ ,s] )] / decimal [ (p[ ,s] )]

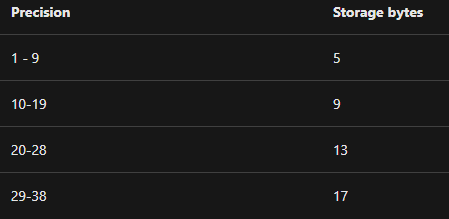
Numeric=decimal(5 to 17 bytes)

*money(8 byte) and smallmoney(4 byte) are also Exact Data Type and will map to Decimal*

Fixed **precision** and **scale(after .)** numbers

**precision** - maximum precision of 38. The default precision is 18.

**Scale** - Количество десятичных цифр, которые хранятся справа от десятичной точки



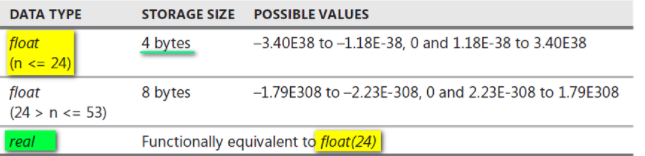
## Difference

NUMERIC должен быть точно таким же точным, как он определен — поэтому, если вы определяете 4 decimal места слева от точки decimal и 4 decimal места справа от нее, DB всегда должен хранить 4 + 4 decimal места, не больше и не меньше.

DECIMAL может свободно разрешать более высокие числа, если это проще реализовать. Это означает, что база данных на самом деле может хранить больше цифр, чем указано (из-за того, что в хранилище behind-the-scenes есть место для дополнительных цифр). Это означает, что база данных может разрешить хранение 12345.0000 в приведенном выше примере 4 + 4 decimal мест, но хранение 1.00005 по-прежнему запрещено, если это может повлиять на любые будущие вычисления.

## Float /real

* real (4 byte)
  + will map to Single in .NET
  + The ISO synonym for real is float(24)
* float (8 byte)
  + will map to Double in .NET



Параметр, передаваемый для типа данных с плавающей запятой, определяет количество битов, которые используются для хранения мантиссы числа с плавающей запятой.

## char [ ( N ) ]

Fixed-size string data N = 1 to 8000 non-Unicode characters bytes

## nvarchar [ ( N | max ) ]

N or 2^31-1 N = 1 to 8000 non-Unicode characters bytes

Max = 2^31-1 bytes (2 GB) non-Unicode characters bytes

## Text

Fixed and variable-length data types for storing large non-Unicode and Unicode character and binary data.

2^31-1 1 to 2^31-1 (2,147,483,647) non-Unicode characters bytes

## nchar [ ( N ) ]

N = 1 to 4000 UNICODE UCS-2 bytes

## nvarchar [ ( N | max ) ]

N or 2^31-1 N = 1 to 4000 UNICODE UCS-2 bytes

1 to 2^31-1 (2,147,483,647) UNICODE UCS-2 bytes

## Ntext

Fixed and variable-length data types for storing large non-Unicode and Unicode character and binary data.

2^30-1 Maximum size 2^30 – 1 (1,073,741,823) bytes

## binary [ ( N ) ]

N = 1 to 8000 bytes

## varbinary [ ( N | max) ]

N or 2^31-1 N = 1 to 8000 bytes

Max = 0 to 2^31-1 bytes

## Date

3bytes, 0001-01-01 through 9999-12-31

## time [ (fractional second precision) ]

3 to 5 00:00:00.0000000 through 23:59:59.9999999

## Smalldatetime

4 bytes Date: 1900-01-01 through 2079-06-06

Time: 00:00:00 through 23:59:59

## Datetime

8 bytes

Diapason: January 1, 1753, through December 31, 9999

Time: 00:00:00 through 23:59:59.997

map to System.DateTime in .NET - no difference there.

**Precision:** DATETIME is limited to 3 1/3 milliseconds, while DATETIME2 can be accurate down to 100ns.

*recommend using DATETIME2 whenever possible*

## datetime2 [ (fractional seconds precision) ]

6 to 8 bytes

Diapason: 0001-01-01 through 9999-12-31

Time: 00:00:00 through 23:59:59.9999999

map to System.DateTime in .NET - no difference there.

**Precision:** DATETIME is limited to 3 1/3 milliseconds, while DATETIME2 can be accurate down to 100ns.

## datetimeoffset [ (fractional seconds precision) ]

8 to 10 bytes Date: 0001-01-01 through 9999-12-31

## Time:

Map to TimeSpan

00:00:00 through 23:59:59.9999999

Time zone offset:

-14:00 through +14:00