# Manage transactions

## Trancount

update [dbo].[tblEmployee] set EmployeeNumber = 123 where EmployeeNumber = 122

select \* from [dbo].[tblEmployee]

select @@TRANCOUNT --0

begin tran

 select @@TRANCOUNT --1

 begin tran

 update [dbo].[tblEmployee] set EmployeeNumber = 122 where EmployeeNumber = 123

 select @@TRANCOUNT --2

 commit tran

 select @@TRANCOUNT --1

if @@TRANCOUNT > 0 --Yes

commit tran

select @@TRANCOUNT --0

select \* from [dbo].[tblEmployee]

## isolation levels;

Transaction 1

begin transaction

update [dbo].[tblEmployee] set EmployeeNumber = 122 where EmployeeNumber = 123

commit tran

update [dbo].[tblEmployee] set EmployeeNumber = 123 where EmployeeNumber = 122

insert into [dbo].[tblEmployee]([EmployeeNumber]

 ,[EmployeeFirstName]

 ,[EmployeeMiddleName]

 ,[EmployeeLastName]

 ,[EmployeeGovernmentID]

 ,[DateOfBirth]

 ,[Department])

values (122,'H','I','T','H','2010-01-01','H')

delete from [dbo].[tblEmployee]

where EmployeeNumber = 122

Transaction 2

set transaction isolation level read committed

begin tran

select \* from [dbo].[tblEmployee]

waitfor delay '00:00:20'

select \* from [dbo].[tblEmployee]

commit tran

## Clustered Index

create clustered index idx\_tblEmployee on [dbo].[tblEmployee]([EmployeeNumber])

drop index idx\_tblEmployee on [dbo].[tblEmployee]

select \* from [dbo].[tblEmployee2] where [EmployeeNumber] = 127

select \* from [dbo].[tblEmployee2]

select \*

into [dbo].[tblEmployee2]

from [dbo].[tblEmployee]

where EmployeeNumber <> 131

--seek = few number of rows based on the index

--scan = going through the entire table

alter table [dbo].[tblEmployee2]

add constraint pk\_tblEmployee2 PRIMARY KEY(EmployeeNumber)

create table myTable (Field1 int primary key)

## Non-clustered Index

create nonclustered index idx\_tblEmployee\_DateOfBirth on [dbo].[tblEmployee]([DateOfBirth])

create nonclustered index idx\_tblEmployee\_DateOfBirth\_Department on [dbo].[tblEmployee]([DateOfBirth],Department)

drop index idx\_tblEmployee on [dbo].[tblEmployee]

select \* from [dbo].[tblEmployee2] where [EmployeeNumber] = 127

select \* from [dbo].[tblEmployee2]

select DateOfBirth, Department

from [dbo].[tblEmployee]

where DateOfBirth>='1992-01-01' and DateOfBirth<'1993-01-01'

--seek = few number of rows based on the index

--scan = going through the entire table

alter table [dbo].[tblDepartment]

add constraint unq\_tblDepartment UNIQUE(Department)

### Filtered indices

CREATE NONCLUSTERED INDEX idx\_tblEmployee\_Employee

 ON dbo.tblEmployee(EmployeeNumber) where EmployeeNumber<139;

### INCLUDE

CREATE NONCLUSTERED INDEX idx\_tblEmployee\_Employee

 ON dbo.tblEmployee(EmployeeNumber) include (EmployeeFirstName);

DROP INDEX idx\_tblEmployee\_Employee ON dbo.tblEmployee

## Include Client Statistics

select \*

from [dbo].[tblEmployee] as E

Table Scan

select \*

from [dbo].[tblEmployee] as E

where E.EmployeeNumber = 134

Still a Table Scan

## Hash match:

select \*

from [dbo].[tblDepartment] as D

left join [dbo].[tblEmployee] as E

on D.Department = E.Department

select D.Department, D.DepartmentHead, E.EmployeeNumber, E.EmployeeFirstName, E.EmployeeLastName

from [dbo].[tblDepartment] as D

left join [dbo].[tblEmployee] as E

on D.Department = E.Department

## Nested Loop

select D.Department, D.DepartmentHead, E.EmployeeNumber, E.EmployeeFirstName, E.EmployeeLastName

from [dbo].[tblDepartment] as D

left join [dbo].[tblEmployee] as E

on D.Department = E.Department

where D.Department = 'HR'

select \*

from [dbo].[tblEmployee] as E

left join [dbo].[tblTransaction] as T

on E.EmployeeNumber = T.EmployeeNumber

select E.EmployeeNumber, T.Amount

from [dbo].[tblEmployee] as E

left join [dbo].[tblTransaction] as T

on E.EmployeeNumber = T.EmployeeNumber

## Merge Joins

CREATE UNIQUE CLUSTERED INDEX [idx\_tblEmployee] ON [dbo].[tblEmployee]

([EmployeeNumber])

GO

CREATE UNIQUE CLUSTERED INDEX [idx\_tblTransaction] ON [dbo].[tblTransaction]

([EmployeeNumber],[DateOfTransaction],[Amount])

GO

select E.EmployeeNumber, T.Amount

from [dbo].[tblEmployee] as E

left join [dbo].[tblTransaction] as T

on E.EmployeeNumber = T.EmployeeNumber

select \*

into dbo.tblEmployeeNoIndex

from dbo.tblEmployee

select \*

into dbo.tblTransactionNoIndex

from dbo.tblTransaction

select E.EmployeeNumber, T.Amount

from [dbo].[tblEmployee] as E

left join [dbo].[tblTransaction] as T

on E.EmployeeNumber = T.EmployeeNumber

select E.EmployeeNumber, T.Amount

from [dbo].[tblEmployeeNoIndex] as E

left join [dbo].[tblTransactionNoIndex] as T

on E.EmployeeNumber = T.EmployeeNumber

## Even bigger savings of time when using a SARG

select E.EmployeeNumber, T.Amount

from [dbo].[tblEmployee] as E

left join [dbo].[tblTransaction] as T

on E.EmployeeNumber = T.EmployeeNumber

where E.EmployeeNumber = 134

select E.EmployeeNumber, T.Amount

from [dbo].[tblEmployeeNoIndex] as E

left join [dbo].[tblTransactionNoIndex] as T

on E.EmployeeNumber = T.EmployeeNumber

where E.EmployeeNumber = 134

select E.EmployeeNumber, T.Amount

from [dbo].[tblEmployee] as E

left join [dbo].[tblTransaction] as T

on E.EmployeeNumber = T.EmployeeNumber

where E.EmployeeNumber / 10 = 34 --Not SARG

select E.EmployeeNumber, T.Amount

from [dbo].[tblEmployee] as E

left join [dbo].[tblTransaction] as T

on E.EmployeeNumber = T.EmployeeNumber

where E.EmployeeNumber between 340 and 349 --SARG

## plan guides

select \*

into dbo.tblTransactionBig

from [dbo].[tblTransaction]

insert into dbo.tblTransactionBig ([Amount], [DateOfTransaction], [EmployeeNumber])

select T1.Amount, T2.DateOfTransaction, 1 as EmployeeNumber

from [dbo].[tblTransaction] as T1

cross join (select \* from [dbo].[tblTransaction] where EmployeeNumber<200) as T2

create nonclustered index idx\_tbltblTransactionBig on dbo.tblTransactionBig(EmployeeNumber)

create proc procTransactionBig(@EmployeeNumber as int) WITH RECOMPILE

as

select \*

from tblTransactionBig as T

left join tblEmployee as E

on T.EmployeeNumber = E.EmployeeNumber

where T.EmployeeNumber = @EmployeeNumber

exec procTransactionBig 1

exec procTransactionBig 132

## Hints

select D.Department, D.DepartmentHead, E.EmployeeNumber, E.EmployeeFirstName, E.EmployeeLastName

from [dbo].[tblDepartment] as D WITH (NOLOCK)

left join [dbo].[tblEmployee] as E

on D.Department = E.Department

where D.Department = 'HR'

select D.Department, D.DepartmentHead, E.EmployeeNumber, E.EmployeeFirstName, E.EmployeeLastName

from [dbo].[tblDepartment] as D WITH (REPEATABLEREAD)

left join [dbo].[tblEmployee] as E

on D.Department = E.Department

where D.Department = 'HR'

* dynamic vs. parameterised queries

DECLARE @param varchar(1000) = '127';

DECLARE @sql nvarchar(max) =

 N'

 SELECT \*

 FROM [dbo].[tblTransaction] AS T

 WHERE T.EmployeeNumber = ' + @param;

EXECUTE (@sql);

DECLARE @parameter varchar(1000) = '127' + char(10) + 'SELECT \* from dbo.tblTransaction';

DECLARE @sql nvarchar(max) =

 N'

 SELECT \*

 FROM [dbo].[tblTransaction] AS T

 WHERE T.EmployeeNumber = ' + @parameter;

EXECUTE (@sql);

DECLARE @param varchar(1000) = '127';

EXECUTE sys.sp\_executesql

 @statement =

 N'

 SELECT \*

 FROM [dbo].[tblTransaction] AS T

 WHERE T.EmployeeNumber = @EmployeeNumber;',

 @params = N'@EmployeeNumber varchar(1000)',

 @EmployeeNumber = @param;

# DMVs (Index Related Dynamic Management Views and Functions)

## dm\_db\_index\_usage\_stats

select db\_name(database\_id) as [Database Name]

, object\_name(ddius.object\_id) as [Table Name]

, i.name as [Index Name]

, ddius.\*

from sys.dm\_db\_index\_usage\_stats as ddius

join sys.indexes as i on ddius.object\_id = i.object\_id and ddius.index\_id = i.index\_id

where database\_id = db\_id()

## sys.dm\_db\_missing\_index\_details

select T.\*

into dbo.tblTransactionBigger

from [dbo].[tblTransaction] as T

cross join [dbo].[tblTransaction] as T2

select \* from dbo.tblTransactionBigger

where [EmployeeNumber] = 127

select \* from sys.dm\_db\_missing\_index\_details

select mig.\*, statement as table\_name, column\_id, column\_name, column\_usage

from sys.dm\_db\_missing\_index\_details as mid

cross apply sys.dm\_db\_missing\_index\_columns(mid.index\_handle)

inner join sys.dm\_db\_missing\_index\_groups as mig on mig.index\_handle = mid.index\_handle

where database\_id = db\_id()

order by column\_id

drop table dbo.tblTransactionBigger

## sys.dm\_db\_index\_physical\_stats

SELECT \* FROM sys.dm\_db\_index\_physical\_stats

 (DB\_ID(N'70-461'), OBJECT\_ID(N'dbo.tblEmployee'), NULL, NULL , 'DETAILED');

 database\_id object\_id index\_id/partition\_number/mode

# Evaluate the use of row-based operations vs. set-based operations

## When to use cursors

declare @EmployeeID int

declare csr CURSOR FOR

select EmployeeNumber

from [dbo].[tblEmployee]

where EmployeeNumber between 120 and 299

open csr

fetch next from csr into @EmployeeID

while @@FETCH\_STATUS = 0

begin

 select \* from [dbo].[tblTransaction] where EmployeeNumber = @EmployeeID

 fetch next from csr into @EmployeeID

end

close csr

deallocate csr

## Alternatives

select T.\*

from tblTransaction as T

right join tblEmployee as E

on T.EmployeeNumber = E.EmployeeNumber

where E.EmployeeNumber between 120 and 299
and T.EmployeeNumber is not null

## impact of scalar UDFs

--set statistics time on

CREATE FUNCTION fnc\_TransactionTotal (@intEmployee as int)

returns money

as

begin

declare @TotalAmount as money

select @TotalAmount = sum(Amount)

from [dbo].[tblTransaction]

where EmployeeNumber = @intEmployee

return @TotalAmount

end

set showplan\_all on

go

set showplan\_text on

go

select [EmployeeNumber], dbo.fnc\_TransactionTotal([EmployeeNumber])

from [dbo].[tblEmployee]

select E.[EmployeeNumber], sum(Amount) as TotalAmount

from [dbo].[tblEmployee] as E

left join [dbo].[tblTransaction] as T

on E.EmployeeNumber = T.EmployeeNumber

group by E.[EmployeeNumber]

set statistics time off

set showplan\_all off

select EmployeeNumber, dbo.fnc\_TransactionTotal(EmployeeNumber)

from dbo.tblEmployee

select E.EmployeeNumber, sum(T.Amount) as TotalAmount

from dbo.tblEmployee as E

left join dbo.tblTransaction as T

on E.EmployeeNumber = T.EmployeeNumber

group by E.EmployeeNumber

select E.EmployeeNumber, (select sum(Amount) from tblTransaction as T

 where T.EmployeeNumber = E.EmployeeNumber) as TotalAmount

from dbo.tblEmployee as E

create function fnc\_TransactionAll (@intEmployee as int)

returns @returntable table

(Amount smallmoney)

as

begin

 insert @returntable

 select amount

 from dbo.tblTransaction

 where EmployeeNumber = @intEmployee

 return

end

select \* from dbo.fnc\_TransactionAll (128)

select EmployeeNumber, sum(T.Amount) as TotalAmount

from dbo.tblEmployee as E

outer apply fnc\_TransactionAll(EmployeeNumber) as T

group by EmployeeNumber

select E.EmployeeNumber, sum(T.Amount) as TotalAmount

from dbo.tblEmployee as E

left join dbo.tblTransaction as T on E.EmployeeNumber = T.EmployeeNumber

group by E.EmployeeNumber

# Query and manage XML data

## RAW

select P.ProductID, P.Name, S.Name as SubcategoryName

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml raw

<row ProductID="706" Name="HL Road Frame - Red, 58" SubcategoryName="Road Frames" />

<row ProductID="707" Name="Sport-100 Helmet, Red" SubcategoryName="Helmets" />

<row ProductID="708" Name="Sport-100 Helmet, Black" SubcategoryName="Helmets" />

<row ProductID="709" Name="Mountain Bike Socks, M" SubcategoryName="Socks" />

select P.ProductID, P.Name, S.Name as SubcategoryName

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml raw('MyRow')

<MyRow ProductID="706" Name="HL Road Frame - Red, 58" SubcategoryName="Road Frames" />

<MyRow ProductID="707" Name="Sport-100 Helmet, Red" SubcategoryName="Helmets" />

<MyRow ProductID="708" Name="Sport-100 Helmet, Black" SubcategoryName="Helmets" />

<MyRow ProductID="709" Name="Mountain Bike Socks, M" SubcategoryName="Socks" />

select P.ProductID, P.Name, S.Name as SubcategoryName

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml raw('MyRow'), type

-- You can optionally specify the TYPE directive to retrieve the results as xml type. The TYPE directive does not change the content of the results. Only the data type of the results is affected. +

<MyRow ProductID="706" Name="HL Road Frame - Red, 58" SubcategoryName="Road Frames" />

<MyRow ProductID="707" Name="Sport-100 Helmet, Red" SubcategoryName="Helmets" />

<MyRow ProductID="708" Name="Sport-100 Helmet, Black" SubcategoryName="Helmets" />

<MyRow ProductID="709" Name="Mountain Bike Socks, M" SubcategoryName="Socks" />

select P.ProductID, P.Name, S.Name as SubcategoryName

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml raw, elements

<row>

 <ProductID>706</ProductID>

 <Name>HL Road Frame - Red, 58</Name>

 <SubcategoryName>Road Frames</SubcategoryName>

</row>

<row>

 <ProductID>707</ProductID>

 <Name>Sport-100 Helmet, Red</Name>

 <SubcategoryName>Helmets</SubcategoryName>

</row>

<row>

 <ProductID>708</ProductID>

 <Name>Sport-100 Helmet, Black</Name>

 <SubcategoryName>Helmets</SubcategoryName>

</row>

<row>

 <ProductID>709</ProductID>

 <Name>Mountain Bike Socks, M</Name>

 <SubcategoryName>Socks</SubcategoryName>

</row>

## AUTO

select P.ProductID, P.Name, S.Name as SubcategoryName

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml auto

<P ProductID="706" Name="HL Road Frame - Red, 58">

 <S SubcategoryName="Road Frames" />

</P>

<P ProductID="707" Name="Sport-100 Helmet, Red">

 <S SubcategoryName="Helmets" />

</P>

<P ProductID="708" Name="Sport-100 Helmet, Black">

 <S SubcategoryName="Helmets" />

</P>

<P ProductID="709" Name="Mountain Bike Socks, M">

 <S SubcategoryName="Socks" />

</P>

select P.ProductID, P.Name, S.Name as SubcategoryName

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml auto, elements

<P>

 <ProductID>706</ProductID>

 <Name>HL Road Frame - Red, 58</Name>

 <S>

 <SubcategoryName>Road Frames</SubcategoryName>

 </S>

</P>

<P>

 <ProductID>707</ProductID>

 <Name>Sport-100 Helmet, Red</Name>

 <S>

 <SubcategoryName>Helmets</SubcategoryName>

 </S>

</P>

<P>

 <ProductID>708</ProductID>

 <Name>Sport-100 Helmet, Black</Name>

 <S>

 <SubcategoryName>Helmets</SubcategoryName>

 </S>

</P>

<P>

 <ProductID>709</ProductID>

 <Name>Mountain Bike Socks, M</Name>

 <S>

 <SubcategoryName>Socks</SubcategoryName>

 </S>

</P>

## EXPLICIT

select 1 as Tag, NULL as Parent

 , P.ProductID as [Product!1!ProductID]

 , P.Name as [Product!1!ProductName]

 , S.Name as [Product!1!SubcategoryName]

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml explicit

<Product ProductID="706" ProductName="HL Road Frame - Red, 58" SubcategoryName="Road Frames" />

<Product ProductID="707" ProductName="Sport-100 Helmet, Red" SubcategoryName="Helmets" />

<Product ProductID="708" ProductName="Sport-100 Helmet, Black" SubcategoryName="Helmets" />

<Product ProductID="709" ProductName="Mountain Bike Socks, M" SubcategoryName="Socks" />

## PATH

select P.ProductID, P.Name, S.Name as SubcategoryName

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml path

<row>

 <ProductID>706</ProductID>

 <Name>HL Road Frame - Red, 58</Name>

 <SubcategoryName>Road Frames</SubcategoryName>

</row>

<row>

 <ProductID>707</ProductID>

 <Name>Sport-100 Helmet, Red</Name>

 <SubcategoryName>Helmets</SubcategoryName>

</row>

<row>

 <ProductID>708</ProductID>

 <Name>Sport-100 Helmet, Black</Name>

 <SubcategoryName>Helmets</SubcategoryName>

</row>

<row>

 <ProductID>709</ProductID>

 <Name>Mountain Bike Socks, M</Name>

 <SubcategoryName>Socks</SubcategoryName>

</row>

select P.ProductID, P.Name, S.Name as SubcategoryName

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml path('Products')

<Products>

 <ProductID>706</ProductID>

 <Name>HL Road Frame - Red, 58</Name>

 <SubcategoryName>Road Frames</SubcategoryName>

</Products>

<Products>

 <ProductID>707</ProductID>

 <Name>Sport-100 Helmet, Red</Name>

 <SubcategoryName>Helmets</SubcategoryName>

</Products>

<Products>

 <ProductID>708</ProductID>

 <Name>Sport-100 Helmet, Black</Name>

 <SubcategoryName>Helmets</SubcategoryName>

</Products>

<Products>

 <ProductID>709</ProductID>

 <Name>Mountain Bike Socks, M</Name>

 <SubcategoryName>Socks</SubcategoryName>

</Products>

select P.ProductID as '@ProductID', P.Name, S.Name as SubcategoryName

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml path('Products')

<Products ProductID="706">

 <Name>HL Road Frame - Red, 58</Name>

 <SubcategoryName>Road Frames</SubcategoryName>

</Products>

<Products ProductID="707">

 <Name>Sport-100 Helmet, Red</Name>

 <SubcategoryName>Helmets</SubcategoryName>

</Products>

<Products ProductID="708">

 <Name>Sport-100 Helmet, Black</Name>

 <SubcategoryName>Helmets</SubcategoryName>

</Products>

<Products ProductID="709">

 <Name>Mountain Bike Socks, M</Name>

 <SubcategoryName>Socks</SubcategoryName>

</Products>

select P.ProductID as '@ProductID', P.Name as '@ProductName', S.Name as SubcategoryName

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml path('Products')

--@ = attribute, otherwise it is an element.

<Products ProductID="706" ProductName="HL Road Frame - Red, 58">

 <SubcategoryName>Road Frames</SubcategoryName>

</Products>

<Products ProductID="707" ProductName="Sport-100 Helmet, Red">

 <SubcategoryName>Helmets</SubcategoryName>

</Products>

<Products ProductID="708" ProductName="Sport-100 Helmet, Black">

 <SubcategoryName>Helmets</SubcategoryName>

</Products>

<Products ProductID="709" ProductName="Mountain Bike Socks, M">

 <SubcategoryName>Socks</SubcategoryName>

</Products>

select P.ProductID as '@ProductID', P.Name as '@ProductName'

, S.Name as 'Subcategory/SubcategoryName'

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml path('Products')

<Products ProductID="706" ProductName="HL Road Frame - Red, 58">

 <Subcategory>

 <SubcategoryName>Road Frames</SubcategoryName>

 </Subcategory>

</Products>

<Products ProductID="707" ProductName="Sport-100 Helmet, Red">

 <Subcategory>

 <SubcategoryName>Helmets</SubcategoryName>

 </Subcategory>

</Products>

<Products ProductID="708" ProductName="Sport-100 Helmet, Black">

 <Subcategory>

 <SubcategoryName>Helmets</SubcategoryName>

 </Subcategory>

</Products>

<Products ProductID="709" ProductName="Mountain Bike Socks, M">

 <Subcategory>

 <SubcategoryName>Socks</SubcategoryName>

 </Subcategory>

</Products>

## Query and FLWOR

declare @x xml

set @x='<Shopping ShopperName="Phillip Burton" >

<ShoppingTrip ShoppingTripID="L1" >

 <Item Cost="5">Bananas</Item>

 <Item Cost="4">Apples</Item>

 <Item Cost="3">Cherries</Item>

</ShoppingTrip>

<ShoppingTrip ShoppingTripID="L2" >

 <Item>Emeralds</Item>

 <Item>Diamonds</Item>

 <Item>Furniture</Item>

</ShoppingTrip>

</Shopping>'

SELECT @x.query('

 for $Item in /Shopping/ShoppingTrip/Item

 return $Item

')

<Item Cost="5">Bananas</Item><Item Cost="4">Apples</Item><Item Cost="3">Cherries</Item><Item>Emeralds</Item><Item>Diamonds</Item><Item>Furniture</Item>

SELECT @x.query('

 for $Item in /Shopping/ShoppingTrip/Item

 return string($Item)

')

Bananas Apples Cherries Emeralds Diamonds Furniture

SELECT @x.query('

 for $Item in /Shopping/ShoppingTrip/Item

 return concat(string($Item),";")

')

Bananas; Apples; Cherries; Emeralds; Diamonds; Furniture;

SELECT @x.query('

 for $Item in /Shopping/ShoppingTrip[1]/Item

 order by $Item/@Cost

 return concat(string($Item),";")

')

Bananas; Cherries; Apples;

SELECT @x.query('

 for $Item in /Shopping/ShoppingTrip[1]/Item

 let $Cost := $Item/@Cost

 where $Cost = 4

 order by $Cost

 return concat(string($Item),";")

')

Apples;

## Modify

SET @x.modify('

 replace value of (/Shopping/ShoppingTrip[1]/Item[3]/@Cost)[1]

 with "5.0"

')

SELECT @x

<Shopping ShopperName="Phillip Burton">

 <ShoppingTrip ShoppingTripID="L1">

 <Item Cost="5.0">Apples</Item>

 <Item Cost="2">Bananas</Item>

 <Item Cost="3">Cherries</Item>

 </ShoppingTrip>

 <ShoppingTrip ShoppingTripID="L2">

 <Item>Diamonds</Item>

 <Item>Emeralds</Item>

 <Item>Furniture</Item>

 </ShoppingTrip>

</Shopping>

SET @x.modify('

 insert <Item Cost="5">Manu Item 5 at Loc 1</Item>

 into (/Shopping/ShoppingTrip)[1]

')

SELECT @x

<Shopping ShopperName="Phillip Burton">

 <ShoppingTrip ShoppingTripID="L1">

 <Item Cost="4">Apples</Item>

 <Item Cost="2">Bananas</Item>

 <Item Cost="3">Cherries</Item>

 <Item Cost="5">Manu Item 5 at Loc 1</Item>

 </ShoppingTrip>

 <ShoppingTrip ShoppingTripID="L2">

 <Item>Diamonds</Item>

 <Item>Emeralds</Item>

 <Item>Furniture</Item>

 </ShoppingTrip>

</Shopping>

SET @x.modify('

 delete (/Shopping/ShoppingTrip)[1]

')

SELECT @x

<Shopping ShopperName="Phillip Burton">

 <ShoppingTrip ShoppingTripID="L2">

 <Item>Diamonds</Item>

 <Item>Emeralds</Item>

 <Item>Furniture</Item>

 </ShoppingTrip>

</Shopping>

## Value

SELECT @x.value('(/Shopping/ShoppingTrip/Item)[1]','varchar(50)')

Apples

SELECT @x.value('(/Shopping/ShoppingTrip/Item/@Cost)[1]','varchar(50)')

4

## Nodes

select T2.Loc.query('.') from @x.nodes('/Shopping/ShoppingTrip') as T2(Loc) –Table(Column) –shreds xml into relational data

<ShoppingTrip ShoppingTripID="L1"><Item Cost="4">Apples</Item><Item Cost="2">Bananas</Item><Item Cost="3">Cherries</Item></ShoppingTrip>

<ShoppingTrip ShoppingTripID="L2"><Item>Diamonds</Item><Item>Emeralds</Item><Item>Furniture</Item></ShoppingTrip>

<https://docs.microsoft.com/en-us/sql/t-sql/xml/nodes-method-xml-data-type>

select T2.Loc.value('@Cost','varchar(50)')

from @x.nodes('/Shopping/ShoppingTrip/Item') as T2(Loc)

4

2

3

NULL

NULL

NULL

Create Table #tblXML

(pkXML INT PRIMARY KEY,

xmlCol XML)

declare @x xml

set @x='<Shopping ShopperName="Phillip Burton" >

<ShoppingTrip ShoppingTripID="L1" >

 <Item Cost="5">Bananas</Item>

 <Item Cost="4">Apples</Item>

 <Item Cost="3">Cherries</Item>

</ShoppingTrip>

<ShoppingTrip ShoppingTripID="L2" >

 <Item>Emeralds</Item>

 <Item>Diamonds</Item>

 <Item>Furniture</Item>

</ShoppingTrip>

</Shopping>'

INSERT INTO #tblXML(pkXML, xmlCol)

VALUES (1, @x)

SELECT MyTable.ColXML.query('.')

FROM #tblXML

CROSS APPLY xmlCol.nodes('Shopping/ShoppingTrip') as MyTable(ColXML)

drop table #tblXML

go

<ShoppingTrip ShoppingTripID="L1"><Item Cost="5">Bananas</Item><Item Cost="4">Apples</Item><Item Cost="3">Cherries</Item></ShoppingTrip>

<ShoppingTrip ShoppingTripID="L2"><Item>Emeralds</Item><Item>Diamonds</Item><Item>Furniture</Item></ShoppingTrip>

SELECT MyTable.ColXML.value('@Cost','varchar(50)')

FROM #tblXML

CROSS APPLY xmlCol.nodes('Shopping/ShoppingTrip/Item') as MyTable(ColXML)

5

4

3

NULL

NULL

NULL

## XML data: how to handle it in SQL Server and when and when not to use it, including XML namespaces

select P.ProductID, P.Name, S.Name as SubcategoryName

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml raw,xmldata --this is being depreciated

<Schema name="Schema2" xmlns="urn:schemas-microsoft-com:xml-data" xmlns:dt="urn:schemas-microsoft-com:datatypes">

 <ElementType name="row" content="empty" model="closed">

 <AttributeType name="ProductID" dt:type="i4" />

 <AttributeType name="Name" dt:type="string" />

 <AttributeType name="SubcategoryName" dt:type="string" />

 <attribute type="ProductID" />

 <attribute type="Name" />

 <attribute type="SubcategoryName" />

 </ElementType>

</Schema>

<row xmlns="x-schema:#Schema2" ProductID="706" Name="HL Road Frame - Red, 58" SubcategoryName="Road Frames" />

<row xmlns="x-schema:#Schema2" ProductID="707" Name="Sport-100 Helmet, Red" SubcategoryName="Helmets" />

<row xmlns="x-schema:#Schema2" ProductID="708" Name="Sport-100 Helmet, Black" SubcategoryName="Helmets" />

<row xmlns="x-schema:#Schema2" ProductID="709" Name="Mountain Bike Socks, M" SubcategoryName="Socks" />

select P.ProductID, P.Name, S.Name as SubcategoryName

from [Production].[Product] as P

left join [Production].[ProductSubcategory] as S

on P.ProductSubcategoryID = S.ProductSubcategoryID

where P.ProductID between 700 and 709

for xml raw,xmlschema

<xsd:schema targetNamespace="urn:schemas-microsoft-com:sql:SqlRowSet2" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:sqltypes="http://schemas.microsoft.com/sqlserver/2004/sqltypes" elementFormDefault="qualified">

 <xsd:import namespace="http://schemas.microsoft.com/sqlserver/2004/sqltypes" schemaLocation="http://schemas.microsoft.com/sqlserver/2004/sqltypes/sqltypes.xsd" />

 <xsd:element name="row">

 <xsd:complexType>

 <xsd:attribute name="ProductID" type="sqltypes:int" use="required" />

 <xsd:attribute name="Name" use="required">

 <xsd:simpleType sqltypes:sqlTypeAlias="[AdventureWorks2014].[dbo].[Name]">

 <xsd:restriction base="sqltypes:nvarchar" sqltypes:localeId="1033" sqltypes:sqlCompareOptions="IgnoreCase IgnoreKanaType IgnoreWidth" sqltypes:sqlSortId="52">

 <xsd:maxLength value="50" />

 </xsd:restriction>

 </xsd:simpleType>

 </xsd:attribute>

 <xsd:attribute name="SubcategoryName">

 <xsd:simpleType sqltypes:sqlTypeAlias="[AdventureWorks2014].[dbo].[Name]">

 <xsd:restriction base="sqltypes:nvarchar" sqltypes:localeId="1033" sqltypes:sqlCompareOptions="IgnoreCase IgnoreKanaType IgnoreWidth" sqltypes:sqlSortId="52">

 <xsd:maxLength value="50" />

 </xsd:restriction>

 </xsd:simpleType>

 </xsd:attribute>

 </xsd:complexType>

 </xsd:element>

</xsd:schema>

<row xmlns="urn:schemas-microsoft-com:sql:SqlRowSet2" ProductID="706" Name="HL Road Frame - Red, 58" SubcategoryName="Road Frames" />

<row xmlns="urn:schemas-microsoft-com:sql:SqlRowSet2" ProductID="707" Name="Sport-100 Helmet, Red" SubcategoryName="Helmets" />

<row xmlns="urn:schemas-microsoft-com:sql:SqlRowSet2" ProductID="708" Name="Sport-100 Helmet, Black" SubcategoryName="Helmets" />

<row xmlns="urn:schemas-microsoft-com:sql:SqlRowSet2" ProductID="709" Name="Mountain Bike Socks, M" SubcategoryName="Socks" />

## import and export XML

bcp [70-461S3].dbo.tblDepartment out a-wn.out -N -T

CREATE TABLE [dbo].[tblDepartment2](

 [Department] [varchar](19) NULL,

 [DepartmentHead] [varchar](19) NULL

)

GO

bcp [70-461S3].dbo.tblDepartment2 in a-wn.out -N -T

drop table [dbo].[tblDepartment2]

DROP TABLE #tblXML

GO

CREATE TABLE #tblXML (XmlCol xml);

GO

BULK INSERT #tblXML FROM 'c:\SampleFolder\SampleData4.txt'

select \* from #tblXML

* INSERT ... SELECT \* FROM OPENROWSET(BULK...)

CREATE TABLE #tblXML (IntCol int, XmlCol xml);

GO

INSERT INTO #tblXML(XmlCol)

SELECT \* FROM OPENROWSET(

 BULK 'c:\SampleFolder\SampleData3.txt',

 SINGLE\_BLOB) AS x; --Binary Large Object (BLOB)

select \* from #tblXML

## XML indexing

CREATE XML INDEX secpk\_tblXML\_Path on #tblXML(xmlCol)

USING XML INDEX pk\_tblXML FOR PATH;

CREATE XML INDEX secpk\_tblXML\_Value on #tblXML(xmlCol)

USING XML INDEX pk\_tblXML FOR VALUE;

CREATE XML INDEX secpk\_tblXML\_Property on #tblXML(xmlCol)

USING XML INDEX pk\_tblXML FOR PROPERTY;