Best Practices in SQL Programming

Madhivanan

- Do not use irrelevant datatype
  - VARCHAR instead of DATETIME
  - CHAR(N) instead of VARCHAR(N)
  - etc

Do not use VARCHAR instead of DATETIME

```
create table #employee_master(emp_id int,
   emp_name varchar(100), dob varchar(12))
    insert into  #employee_master
    select 1,'Sankar','09/19/1976' union all
    select 2,'Kamal','02/11/1968' union all
    select 3,'Rajesh','22/29/1200'

select emp_name, dob from #employee_master
   order by dob
```

Result is

emp_name	dob
Kamal	02/11/1968
Sankar	09/19/1976
Rajesh	22/29/1200

```
select emp_name, month(dob) as month_name from
    #employee_master
```

Result is

```
emp_name                                                    month_name
------------------------------------------------------------------------------
    -
Sankar                                                      9
Kamal                                                       2
Msg 241, Level 16, State 1, Line 1
Conversion failed when converting date and/or time from
    character string.
```

Forces implicit datatype convertion (from execution plan)

[Expr1004] = Scalar Operator(datepart(month**,CONVERT_IMPLICIT**(datetimeoffset(7),[tempdb].[dbo].[#employee_master].[dob],0)))

Wrong datatypes

- Force implicit convertion
- Store invalid data

Always specify size for character datatypes

```
declare @name varchar
set @name='Welcome to Chennai SQL Server
   User Group'
select @name
```

Result

w

```
select cast('Welcome to Chennai SQL Server
    User Group' as varchar)
```

Result

Welcome to Chennai SQL Server

```
declare @table table (name varchar)
insert into @table(name)
select 'Welcome to Chennai SQL Server User
   Group'

select name from @table
```

Msg 8152, Level 16, State 14, Line 3
String or binary data would be truncated.
The statement has been terminated.

(0 row(s) affected)

```
declare @v varchar
set @v='124'
select @v
```

Result

1

What is the result?

```
declare @v varchar
set @v=124
select @v
```

```
declare @v varchar
set @v=124
select @v
```

Result

*

By specifying size, you can avoid

- data truncation
- Error
- Unexpected result

CHAR(1) vs VARCHAR(1)

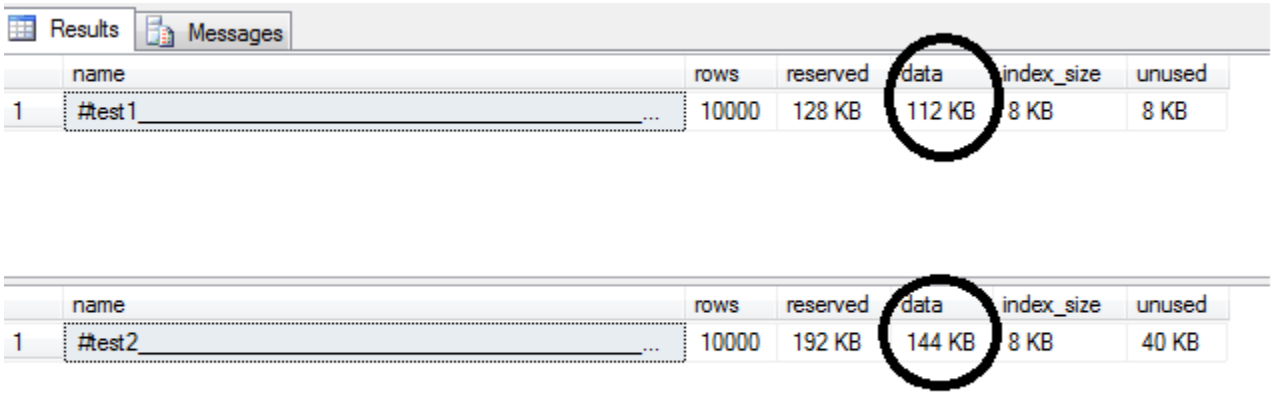
To store Gender values like M for Male, F for Female

T for True or F for False or 1 for True or 0 for False

Do not use VARHCAR(1)

```
create table #test1(name char(1))

insert into #test1(name)
select top 10000 'M' from sys.objects as s1 cross join
    sys.columns as s2



create table #test2(name varchar(1))

insert into #test2(name)
select top 10000 'M' from sys.objects as s1 cross join
    sys.columns as s2
```

```
exec tempdb..sp_spaceused '#test1'
exec tempdb..sp_spaceused '#test2'
```



Results | Messages

	name	rows	reserved	data	index_size	unused
1	#test1_____...	10000	128 KB	112 KB	8 KB	8 KB

	name	rows	reserved	data	index_size	unused
1	#test2_____...	10000	192 KB	144 KB	8 KB	40 KB

CHAR(N) will store N bytes of data

VARCHAR(N) will store actual data in bytes + extra byte to store the length of data

Do not use TEXT datatype

- It is deprecated
- String functions like left, right, etc cannot be used
- No direct support of updating this column
- Order by is not allowed
- Union operator is not allowed

Do not use regional date format

- The regional date formats are
  - DD/MM/YYYY
  - MM/DD/YYYY
  - DD-MON-YYYY
  - etc

```
Declare @test table(dates datetime)
insert into @test
select '12/03/2010'

select dates,month(dates) as month_value from @test
```

Result is

```
dates                          month_value
-----------------------        -----------
2010-12-03 00:00:00.000    12
```

```
Declare @test table(dates datetime)
insert into @test
select '27/03/2009'

select dates,month(dates) as month_value from @test
```

Result is

```
Msg 242, Level 16, State 3, Line 4
The conversion of a varchar data type to a datetime data
    type resulted in an out-of-range value.
```

Dbcc useroptions

Result is

Set options Value

---

language us_english

dateformat mdy

.

.

Forcing the regional date format

```
set dateformat dmy

Declare @test table(dates datetime)
insert into @test
select '12/03/2010'union all
select '27/03/2009‘

select dates,month(dates) as month_value from @test
```

Result is

dates	month_value
2010-03-12 00:00:00.000	3
2009-03-27 00:00:00.000	3

- DD-MON-YYYY format is good only if the default language of the server/session is English

- -- Works good

```
Set dateformat dmy
Declare @test table(dates datetime)
insert into @test
select '12-Mar-2010'union all
select '27-Mar-2009‘
select dates from @test
```

```
--Works good

Set dateformat mdy

Declare @test table(dates datetime)
insert into @test
select '12-Mar-2010'union all
select '27-Mar-2009’
select dates from @test
```

```
set language german
Declare @test table(dates datetime)
insert into @test
select '12-Mar-2010'union all
select '27-Mar-2009‘
select dates from @test
Result is

Die Spracheneinstellung wurde auf Deutsch geändert.
Msg 241, Level 16, State 1, Line 3
Fehler beim Konvertieren einer Zeichenfolge in ein Datum
    und/oder eine Uhrzeit.
```

- **Unambiguous formats**

  YYYYMMDD
  YYYYMMDD HH:MM:SS
  YYYY-MM-DDTHH:MM:SS

```
set language german
Declare @test table(dates datetime)
insert into @test
select '20100312'union all
select '20090327‘
select dates from @test
```

Result is

```
Dates
----------------------
2010-03-12 00:00:00.000
2009-03-27 00:00:00.000
```

```
set language german
Declare @test table(dates datetime)
insert into @test
select '2010-03-12T00:00:00'union all
select '2009-03-27T00:00:00‘
select dates from @test
```

Result is

```
Dates
----------------------
2010-03-12 00:00:00.000
2009-03-27 00:00:00.000
```

```
Set dateformat dmy

Declare @test table(dates datetime)
insert into @test
select '20100312'union all
select '20090327‘
select dates from @test
```

Result is

```
Dates
----------------------
2010-03-12 00:00:00.000
2009-03-27 00:00:00.000
```

```
Set dateformat  mdy

Declare @test table(dates datetime)
insert into @test
select '20100312'union all
select '20090327‘
select dates from @test

Result is

Dates
----------------------
2010-03-12 00:00:00.000
2009-03-27 00:00:00.000
```

```
Set dateformat  ymd

Declare @test table(dates datetime)
insert into @test
select '20100312'union all
select '20090327‘
select dates from @test

Result is

Dates
----------------------
2010-03-12 00:00:00.000
2009-03-27 00:00:00.000
```

Best practices when using datetime values

- Always use proper DATETIME datatype to store dates and times
- Do date formation at your front end application
- Beware of ISDATE() function that it is not fully reliable
- Always express datetime values in unambiguous formats

Do date formation at your front end application

```
Declare @date datetime
Set @date='20121019'
Select convert(varchar(10),@date,101)
```

Result

10/19/2012

The result is of character datatype and not useful for any date calculcations until converted back to datetime datatype

Understand how ISDATE function works

```
select isdate('2007-10-12'),isdate('12-12-2006'),isdate('March 12, 2007')
```

Result is 1

```
select isdate(2007),isdate('2007')
```

Result is 1

select cast(2007 as datetime)

Result is 1905-07-01 00:00:00.000

select cast('2007' as datetime)

Result is 2007-01-01 00:00:00.000

```
declare @dates table (dates varchar(8))
insert into @dates(dates)
Select '20071201' union all Select '20071111' union all
Select '2007' union all Select 2007 union all
Select '1800‘
select dates from @dateswhere ISDATE(dates) =1
```

Result

```
dates
--------
20071201
20071111
2007
2007
1800
```

```
select dates from @dates
where ISDATE(dates) =1 and LEN(dates)=8
```

Result

```
dates
--------
20071201
20071111
```

Don’t format dates using sql

Reasons

- Dates become Varchars and wont allow date related caluculations (dateadd, datediff,etc)
- Wont allow to make use of index (if defined) if formatted at where clause
- Web page, reports, etc treat it as varchars (calculations, Ordering,etc wont work properly)

Proper usage of format function (date format in SQL)

- Export results to text file with specific date format
- Import data from other sources where dates are in different formats
- Front end application can't be changed but it needs specific date formats for display

  etc

Efficient Querying on dates

Internal Storage

- Two 4-byte integers
    - First 4-bytes represents number of days from the base date (January 01, 1900)
    - Second 4-bytes represents time which is number of milliseconds after midnight

Rounding
Datetime value is rounded to
0.000,0.003 or 0.007 milliseconds

Example
select cast('2010-01-01 12:45:34.755' as datetime)
Result

2010-01-01 12:45:34.757

Rounding numbers

Last digit of the millisecond	Rounded value
0 or 1	0
2,3 or 4	3
5,6,7 or 8	7
9	0 (increase previous digit)

SmallDatetime

Rounding
SmallDatetime value is rounded to 1 minute

Values of 29.998 seconds or less are rounded down to the nearest minute; values of 29.999 seconds or more are rounded up to the nearest minute.

Example

```
select
    cast('2010-01-01 12:45:24.755' as smalldatetime),
    cast('2010-01-01 12:45:34.755' as smalldatetime)
```

Result

```
------------------------------------------------------------
2010-01-01 12:45:00         2010-01-01 12:46:00
```

Millisecond expression

```
select
        cast('2010-01-01 12:45:34.79' as datetime),
        cast('2010-01-01 12:45:34:79' as datetime)
```

Result

```
-----------------------------------------------------------------------
2010-01-01 12:45:34.790  2010-01-01 12:45:34.080
```

Make sure millisecond part is always three digits

```
select
        cast('2010-01-01 12:45:34.790' as datetime),
        cast('2010-01-01 12:45:34:790' as datetime)
```

Result

```
-----------------------------------------------------------------------
2010-01-01 12:45:34.790  2010-01-01 12:45:34.790
```

Date '1900-01-01 00:00:00.000‘ is equal To number 0

SELECT CAST(0 as DATETIME)
Result

1900-01-01 00:00:00.000

SELECT CAST(1 as DATETIME)
Result

1900-01-02 00:00:00.000

Remove time part

Usual method

```
declare @date datetime
set @date='2010-01-01T12:45:34.79‘
--Convert to style MM/dd/yyyy
select convert(varchar(10),@date,101)

Result
----------
01/01/2010

--Convert  back to datetime
Select convert(datetime,convert(varchar(10),@date,101))
Result
------------------------------
2010-01-01 00:00:00.000
```

Efficient method

Select dateadd(day,datediff(day,0,@date),0)

Result

2010-01-01 00:00:00.000

Find first day of the month

Method 1
(String handling)

```
declare @date datetime
set @date='2010-03-12 12:36:45'
select cast(cast(year(@date) as char(4))+'-
    '+right('00'+cast(month(@date) as varchar(2)),2)+'-01' as datetime)
```

Result

```
----------------------
2010-03-01 00:00:00.000
```

Method 2

(Date handling)

```
select dateadd(month,datediff(month,0,@date),0)
```

Result

```
----------------------
2010-03-01 00:00:00.000
```

Find first day of the year

Method 1
(String handling)

```
declare @date datetime
set @date='2010-03-12 12:36:45'
select cast(cast(year(@date) as char(4))+‘-01-01' as datetime)
```

Result

```
-----------------------
2010-01-01 00:00:00.000
```

Method 2
(Date handling)

```
select dateadd(year,datediff(year,0,@date),0)
```

Result

```
----------------------
2010-01-01 00:00:00.000
```

Do not use Between operator to check datetime values

```
declare @t table(date_col datetime)
insert into @t (date_col)
select '2001-01-01T12:14:20' union all
select '2001-01-17T19:20:55' union all
select '2001-01-21T00:00:00' union all
select '2001-01-31T16:22:56' union all
select '2001-02-01T00:00:00'
```

```
select date_col from @t
where date_col between '20010101' and
    '20010131'
```

Result

2001-01-01 12:14:20.000

2001-01-17 19:20:55.000

2001-01-21 00:00:00.000

```
select date_col from @t
where date_col between '20010101' and '20010131
    23:59:59:999'
```

Result

2001-01-01 12:14:20.000

2001-01-17 19:20:55.000

2001-01-21 00:00:00.000

2001-01-31 16:22:56.000

2001-02-01 00:00:00.000

```
select date_col from @t
where date_col >='20010101' and
    date_col<'20010201'
```

Result

2001-01-01 12:14:20.000

2001-01-17 19:20:55.000

2001-01-21 00:00:00.000

2001-01-31 16:22:56.000

Sample Queries

Find data added on yesterday

Assumption : Table has a datetime column with default value getdate()

Wrong method

Select columns from table

Where date_col=getdate()-1

Correct Methods

Method 1 (will not make use of index)

```
Select columns from table
Where datediff(day,date_col,getdate())=1
```

Method 2

```
Select columns from table
Where
date_col>=dateadd(day,datediff(day,0,getdate()),-1) and
date_col<dateadd(day,datediff(day,0,getdate()),0)
```

```
declare @orders table(order_id int identity(1,1) primary key, order_date datetime,
    order_value decimal(14,2))
insert into @orders(order_date,order_value)
select '2001-07-16T00:52:26.913',332.38 union all
select '2001-02-10T11:00:39.003',111.61 union all
select '1999-01-12T13:03:42.147',265.83 union all
select '1995-12-15T13:04:42.667',141.34 union all
select '2000-09-10T16:09:58.817',251.30 union all
select '1996-10-15T17:52:51.607',258.17 union all
select '2004-11-22T10:33:05.947',222.98 union all
select '2005-07-15T03:29:47.653',3148.33 union all
select '2006-03-16T19:08:30.423',213.97 union all
select '2006-02-21T15:38:42.770',381.91 union all
select '2007-12-23T00:28:14.490',1140.51 union all
select '2007-10-24T510:19:59.220',33.25
```

Find orders placed on December 15,1995

```
select * from @orders
where
order_date>='19951215' and
 order_date<'19951216'
```

- **Find orders placed from April 1,2001 to July 31, 2001**

```
select * from @orders
where
  order_date>='20010401' and
  order_date<'20010801'
```

- **Find orders placed on July month**

```
select * from @orders
where
order_id>0 and month(order_date)=7
```

Note : order_id>0 is used to make use of index (seek and not scan)

- **Find orders placed on or after 12 PM**

```
select * from @orders
where order_id>0 and
 datepart(hour,order_date) >=12
```

- **Find orders placed in Year 2006**

```
select * from @orders
where
order_date>='20060101' and
order_date<'20070101‘
```

Do not use the following

```
select * from @orders
Where year(orderdate)=2006
```

Do not use deprecated features/functions

- TEXT, NTEXT datatypes
  - Use VARCHAR(MAX) or NVARCHAR(MAX)
- TIMESTAMP
  - Use ROWVERSION
- Set Rowcount option
  - Use TOP Operator

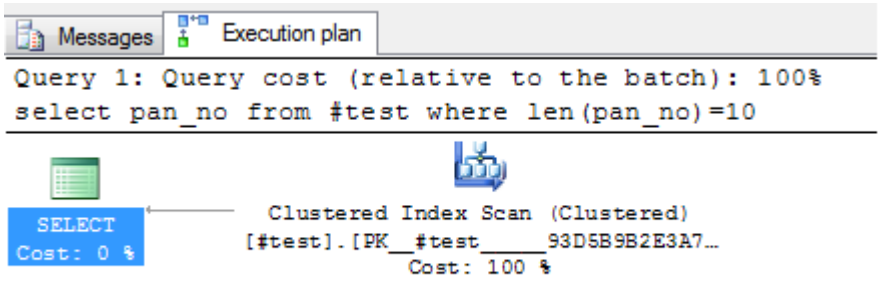
Do not use functions on indexed column

```
create table #test(pan_no char(10) primary key)
insert into #test (pan_no)
select 'ALUJM8765H' union all
select 'GHOKL7653K' union all
select 'IMNK68765H' union all
select 'LOMRF0897U' union all
select 'LPIAC778J' union all
select 'MZXUI1296E'
```

Find out pan no with length =10

```
select pan_no from #test
where len(pan_no)=10
```

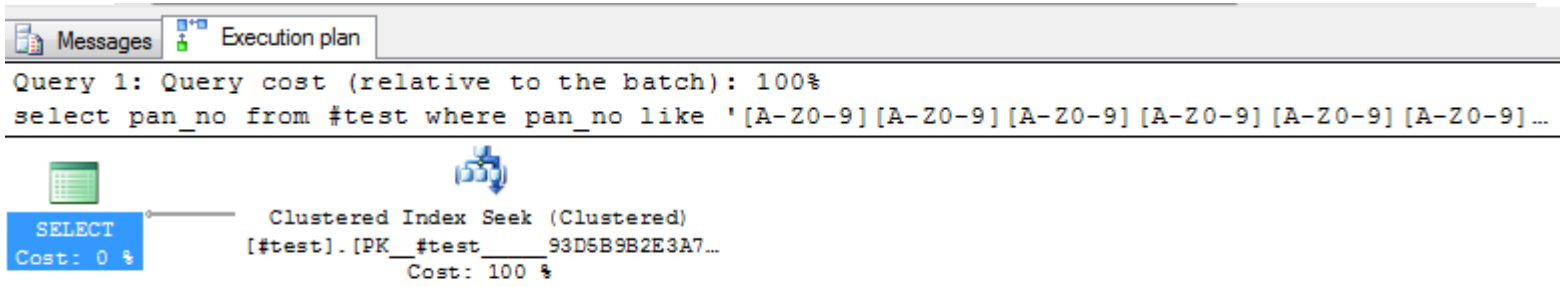
- Execution plan (Index scan)



Use Regular expression whenever needed

```
select pan_no from #test
where pan_no like '[A-Z0-9][A-Z0-9][A-Z0-9][A-Z0-9][A-Z0-9][A-Z0-9][A-Z0-9][A-Z0-9][A-Z0-9][A-Z0-9]'
```

- Execution plan (Index seek)



- Find out Pan no with first five characters are alphabets,next four are numbers and last is an alphabet

```
select pan_no from #test
Where
substring(pan_no,1,1) in (‘A’,’B’,…’Z’)
substring(pan_no,2,1) in (‘A’,’B’,…’Z’)
substring(pan_no,3,1) in (‘A’,’B’,…’Z’)
.
.
right(pan_no,1) in (0,1,2,3,4,5,6,7,8,9)
```

```
select pan_no from #test
where pan_no like '[A-Z][A-Z][A-Z][A-Z][A-Z][0-9][0-9][0-9][0-
    9][A-Z]'
```

Result is

Pan no

ALUJM8765H

GHOKL7653K

LOMRF0897U

MZXUI1296E

Truncate Vs Delete

- Truncate is minimally logged (only page de-allocation is logged)
- Delete is fully logged

- If you want to empty the table, use always TRUNCATE

Beware of implicit convertion

- Select 5/2 (Result 2)
- Select 5+'2‘ (Result 7)
- Select 10+3.5 (Result 13.5)

Understand how ISNUMERIC function works

Select isnumeric(12)

Select isnumeric(87.4)

Select isnumeric('12e2')

Select isnumeric(',')

Result is 1

Select cast(‘12e2’ as float)

Result is 1200 (12*power(10,2))

Select cast(‘,’ as money)

Result is 0.00

Isnumeric works for any expression that can be converted to any number datatypes

Missing comma and Column alias

```
create table #employee_master(emp_id int, emp_name
    varchar(100), dob varchar(12))
insert into  #employee_master
select 1,'Sankar','09/19/1976' union all
select 2,'Kamal','02/11/1968' union all
select 3,'Rajesh','22/29/1200'
```

What is the result of this query?

```
Select emp_id, emp_name dob from #employee_master
```

emp_id	dob
1	Sankar
2	Kamal
3	Rajesh

Check for missing comma in the SELECT statement

Use ORDER BY Clause only when needed

```
create table #numbers(number int)
insert into #numbers
select top 100000 row_number() over (order by (select 0)) as sno from
    sys.columns  as c1 cross join sys.columns as s2

set statistics time on
select * from #numbers
set statistics time off

set statistics time on
select * from #numbers order by number
set statistics time off
```

(100000 row(s) affected)

SQL Server Execution Times:
CPU time = 15 ms, elapsed time = 371 ms.

(100000 row(s) affected)

SQL Server Execution Times:
CPU time = 125 ms, elapsed time = 405 ms.

- **Use ORDER BY Clause for the following cases**

- Export the result in specific order to files (txt, csv, xls, etc)
- Generating the serial numbers, ranking,(also resetting in each group),etc It may be useful for versions prior to 2005.
- A CURSOR that should have data in specific ordering
- Creating a new table (temporarily used for an analysis) with identity column from another table where identity values should depend on some ordering of some columns
- Whenever TOP clause is used

- **Avoid using ORDER BY Clause for the following cases**

- If the result is shown in a front end application (web, reporting tool, etc), you do not need to use it in a query as it can be easily sorted in the application layer.
- When adding data from one table to another. Because you still need ORDER BY clause if want results in specific order from the target table
- When creating a view

Thank You

Contact Me

madhivanan2001@gmail.com

Blog

http://beyondrelational.com/blogs/madhivanan