PROG8080
SQL Scalar Functions: Date Arithmetic and Formatting

Glenn Paulley

Fall 2015

Dates, times, and timestamps

- The ISO SQL standard defines:
  - The DATE type
  - The TIME type
  - The TIMESTAMP type
  - The INTERVAL type
- Relational database products commonly implement DATE, TIME, and TIMESTAMP and are mostly compatible with the SQL standard; any issues are largely ones of precision
- Few products implement the INTERVAL type, and instead permit implementation-defined semantics on arithmetic expressions involving DATE, TIME and TIMESTAMP values

CONESTOGA
Connect Life and Learning

DATE, TIME and TIMESTAMP

- The semantics and function support for date and times remain a significant difference between various relational database products, despite the existence of the SQL standard
  - Support for these type predated updates to the SQL standard, and due to legacy considerations much of the original support remains unchanged
- CAST of one type to another is fairly portable
- Date and time arithmetic is not portable
  - Microsoft SQL Server differs substantially from Oracle, which again differs from IBM DB2
  - DB2 is arguably the closest to the ISO SQL Standard

CONESTOGA
Connect Life and Learning

Date/Time datatypes in SQL Server 2012

- DATE
- TIME
- DATETIME
- SMALLDATETIME
- DATETIME2
- DATETIMEOFFSET

- Note: No TIMESTAMP data type in this list
  - In SQL Server, the TIMESTAMP type is for something else!

CONESTOGA
Connect Life and Learning

DATE ARITHMETIC

CONESTOGA
Connect Life and Learning

Date Arithmetic

- A number of days can be added to a DATETIME column with the “+” operator

```
SELECT dueDate, dueDate + 7 AS 'one week later'
FROM  dbo.Invoice
ORDER BY dueDate
```

CONESTOGA
Connect Life and Learning

Date Arithmetic

- A number of days can be subtracted from a DATETIME column with the “–” operator

```
SELECT dueDate, dueDate - 7 AS 'one week earlier'
FROM  dbo.Invoice
ORDER BY dueDate
```

CONESTOGA
Connect Life and Learning

DATE, TIME, AND TIMESTAMP FUNCTIONS

CONESTOGA
Connect Life and Learning

Date/time functions

DATE and TIME Functions in SQL Server
DATEADD
DATEDIFF
DATENAME
DATEPART
DAY
GETDATE
GETUTCDATE
MONTH
YEAR

Date Functions – DAY(), MONTH(), YEAR( )

- Much of SQL Server's DATE and TIME semantics are legacy behaviour that were preexisting before standardization
- GETDATE( ) returns the current date
  - Returns the date as a DATETIME type (up to 1/300 second accuracy)
- There are specific functions to extract day, month and year from a date:

```
SELECT DAY( expression )
SELECT MONTH( expression )
SELECT YEAR( expression )
```

CONESTOGA
Connect Life and Learning

Date part

- Other SQL Server date functions use “date part” abbreviations – see next slide
- “Date part” abbreviations include:
    - Day (dd or d)
    - Month (mm or m)
    - Year (yyyy or yy)
- Additional “date part” abbreviations cover financial quarters, Julian dates, and time

Datepart Abbreviations

Datepart	Abbreviation
Year	yy, yyyy
Quarter	qq, q
Month	mm, m
Dayofyear,	dy, y
Day	dd, d
Week	wk,ww
Weekday	dw
Hour	hh
Minute	mi, n
second	ss, s
millisecond	ms

CONESTOGA
Connect Life and Learning

Date Functions – DATEPART( )

- Use DATEPART( ) to extract “part” of a date:
  - SELECT DATEPART(date portion, column)
- Where date portion is
  - A datepart code (year, quarter, month, day, week, and so on), or
  - A datepart abbreviation

CONESTOGA
Connect Life and Learning

Date Functions – DATEPART( )

- These DATEPART( ) calls return the same results as DAY( ), MONTH( ) and YEAR( ):
  - SELECT DATEPART(DD, column)
  - SELECT DATEPART(MM, column)
  - SELECT DATEPART(YYYY, column)
- Note that the date part parameter is **not** a string literal enclosed in quotation marks

CONESTOGA
Connect Life and Learning

Date Functions – DATEPART( )

- This function returns the financial quarter (1-4):
  - SELECT DATEPART(Q, column)
- This function returns the Julian date:
  - SELECT DATEPART(DY, column)
- This function returns the weekday (Sunday = 1):
  - SELECT DATEPART(DW, column)

CONESTOGA
Connect Life and Learning

Date Functions – DATEPART( )

- These calls return the time (hours – 24 hour clock, minutes, seconds, milliseconds):
  - SELECT DATEPART(HH, column)
  - SELECT DATEPART(MI, column)
  - SELECT DATEPART(SS, column)
  - SELECT DATEPART(MS, column)

DATEADD and DATEDIFF

- DATEADD – Adds a number of units to a given date
- DATEDIFF – Determines the difference in days/times between two DATETIME types
- DATEADD:
  - SELECT DATEADD(YEAR, 2, GETDATE())
- DATEDIFF:
  - SELECT DATEDIFF(YEAR, DATEADD(YEAR, 2, GETDATE()), GETDATE())

CONESTOGA
Connect Life and Learning

Date Functions – DATEADD( )

- You can also use DATEADD( ) to do “date arithmetic”:
  - SELECT DATEADD(datepart, number, column)
- datepart is a datepart code (e.g. dd)
- number is the number to add (e.g. 1)
- To subtract (go backwards in time), use a negative number (e.g. -1)

Date Functions – DATEADD( )

- Example of using DATEADD( ) to do “date arithmetic”:

```
SELECT "date",
   DATEADD( day, 1, "date" ) AS "tomorrow",
   DATEADD( ww, 1, "date" ) AS "next week",
   DATEADD( mm, 1, "date" ) AS "next month",
   DATEADD( yy, -1, "date" ) AS "last year",
   DATEADD( yy, 1, "date" ) AS "next year"
FROM StudentOffence;
```

Date Functions – DATEDIFF( )

- Use DATEDIFF( ) to calculate the number of datepart units between two dates
  - DATEDIFF(datepart, column1, column2)
- Datepart is a datepart code
- column2 is subtracted from column1
- Note that the value returned is an integer (whole number), not a date

Date Functions – DATEDIFF( )

- Example of using DATEDIFF(datepart, column1, column2) to calculate the difference between a transaction date and a due date:

```
SELECT dueDate, transactionDate,
      DATEDIFF( DAY, dueDate, transactionDate ) as "billing grant"
 FROM [SIS].[dbo].[Invoice];
```

GETDATE() function

- The GETDATE( ) function is used in a SQL SELECT statement to return the current date and time:
  - SELECT GETDATE( )
- By default, SQL Server returns the date and time as a DATETIME type in this format:
  - 2003-09-08 12:01:48.217
- Interpretation:
  - Year-Month-Day Hour:Minutes:Seconds:Milliseconds
- The DATETIME type has a precision of approximately 1/300 second
  - Millisecond precision is possible only with the DATETIME2 type

CASTing Dates

- GETDATE() returns “now” as a DATETIME type
  - Only SAP Adaptive Server Enterprise has similar behaviour
  - In other DBMS systems, DATETIME2 is called a TIMESTAMP
  - Other DBMS systems use the CURRENT TIMESTAMP register for this
- Because GETDATE() returns a DATETIME type, the result of an expression using date arithmetic can be misleading or confusing because of implicit type conversions:

```
SELECT GETDATE() - '2000-01-01'
    AS 'Days since the millenium'
```

- Result (as of the afternoon of 27 August 2015):

```
1915-08-28 16:39:52.633 // not what we desired
```

CASTing Dates

- We can try an explicit CAST:

  SELECT GETDATE() - CAST('2000-01-01' as DATETIME)

  AS 'Days since the millenium'

  - Same result
  - Approximately equivalent to SELECT CAST(5718 AS DATETIME), which is not what we want
    - SQL Server interprets this as the number of days since 1 Jan 1900
- But if we CAST the result to INTEGER:

  SELECT CAST(GETDATE() - '2000-01-01' AS INTEGER)

  AS 'Days since the millenium'

- Result:

  5718 // as of August 27, 2015

CONESTOGA
Connect Life and Learning

DATE FORMATTING

Formatting Dates

- You can use the CONVERT( ) function to format the date differently, such as this:

  2003.09.08

- Example:

```
SELECT CONVERT( CHAR(10), GETDATE( ), 102 )
```

CONESTOGA
Connect Life and Learning

Formatting Dates

- Why CHAR(10)?
  - This defines a 10 character string, big enough for YYYY.MM.DD
- Why 102?
  - This is a style code. For a list of style codes see the Microsoft SQL Server 2014 documentation:
    - https://msdn.microsoft.com/en-us/library/ms187928.aspx
  - Experiment with other style codes to see what the output looks like

CONESTOGA
Connect Life and Learning

Formatting Dates

- Here is an example of formatting dates read from the database

```
SELECT CONVERT( CHAR(10), transactionDate, 102 )
FROM SIS.dbo.Invoice
```