DB2 Basics: Fun with Dates and Times

Paul Yip
Database Consultant
IBM Toronto Lab

28 August 2003

Just updated with custom date/time formatting! This short article explains how to manipulate dates, times and timestamps using SQL on DB2 for Linux, UNIX, and Windows platforms.

Important: Read the disclaimer before reading this article.
This article is written for IBM® DB2® for Linux, UNIX®, and Windows®.

Introduction

This short article is intended for those who are new to DB2 and wish to understand how to manipulate dates and times. Most people who have worked with other databases are pleasantly surprised by how easy it is in DB2.

The basics

To get the current date, time, and timestamp using SQL, reference the appropriate DB2 registers:

```
SELECT current date FROM sysibm.sysdummy1
SELECT current time FROM sysibm.sysdummy1
SELECT current timestamp FROM sysibm.sysdummy1
```

The *sysibm.sysdummy1* table is a special in-memory table that can be used to discover the value of DB2 registers as illustrated above. You can also use the VALUES keyword to evaluate the register or expression. For example, from the DB2 Command Line Processor (CLP), the following SQL statements reveal similar information:

```
VALUES current date
VALUES current time
VALUES current timestamp
```

For the remaining examples, I will simply provide the function or expression without repeating `SELECT ... FROM sysibm.sysdummy1` or using the VALUES clause.

To get the current time or current timestamp adjusted to GMT/CUT, subtract the current timezone register from the current time or timestamp:

```
current time - current timezone
current timestamp - current timezone
```

Given a date, time, or timestamp, you can extract (where applicable) the year, month, day, hour, minutes, seconds, and microseconds portions independently using the appropriate function:

```
YEAR (current timestamp)
MONTH (current timestamp)
DAY (current timestamp)
HOUR (current timestamp)
MINUTE (current timestamp)
SECOND (current timestamp)
MICROSECOND (current timestamp)
```

Extracting the date and time independently from a timestamp is also very easy:

```
DATE (current timestamp)
TIME (current timestamp)
```

You can also perform date and time calculations using, for lack of a better term, English:

```
current date + 1 YEAR
current date + 3 YEARS + 2 MONTHS + 15 DAYS
current time + 5 HOURS - 3 MINUTES + 10 SECONDS
```

To calculate how many days there are between two dates, you can subtract dates as in the following:

```
days (current date) - days (date('1999-10-22'))
```

And here is an example of how to get the current timestamp with the microseconds portion reset to zero:

```
CURRENT TIMESTAMP - MICROSECOND (current timestamp) MICROSECONDS
```

If you want to concatenate date or time values with other text, you need to convert the value into a character string first. To do this, you can simply use the CHAR() function:

```
char(current date)
char(current time)
char(current date + 12 hours)
```

To convert a character string to a date or time value, you can use:

```
TIMESTAMP ('2002-10-20-12.00.00.000000')
TIMESTAMP ('2002-10-20 12:00:00')
 DATE ('2002-10-20')
 DATE ('10/20/2002')
 TIME ('12:00:00')
 TIME ('12.00.00')
```

The TIMESTAMP(), DATE() and TIME() functions accept several more formats. The above formats are examples only and I'll leave it as an exercise for the reader to discover them.

Warning: What happens if you accidentally leave out the quotes in the DATE function? The function still works, but the result is not correct:

```
SELECT DATE(2001-09-22) FROM SYSIBM.SYSDUMMY1;
```

Answer:

```
======
05/24/0006
```

Why the 2,000 year difference in the above results? When the DATE function gets a character string as input, it assumes that it is valid character representation of a DB2 date, and converts it accordingly. By contrast, when the input is numeric, the function assumes that it represents the number of days minus one from the start of the current era (that is, 0001-01-01). In the above query the input was 2001-09-22, which equals (2001-9)-22, which equals 1970 days.

Date functions

Sometimes, you need to know how the difference between two timestamps. For this, DB2 provides a built in function called TIMESTAMPDIFF(). The value returned is an approximation, however, because it does not account for leap years and assumes only 30 days per month. Here is an example of how to find the approximate difference in time between two dates:

```
timestampdiff (<n>, char(
 timestamp('2002-11-30-00.00.00')-
 timestamp('2002-11-08-00.00.00')))
```

In place of <n>, use one of the following values to indicate the unit of time for the result:

- 1 = Fractions of a second
- 2 = Seconds
- 4 = Minutes
- 8 = Hours
- 16 = Days
- 32 = Weeks
- 64 = Months
- 128 = Quarters
- 256 = Years

Using timestampdiff() is more accurate when the dates are close together than when they are far apart. If you need a more precise calculation, you can use the following to determine the difference in time (in seconds):

```
(DAYS(t1) - DAYS(t2)) * 86400 +
(MIDNIGHT_SECONDS(t1) - MIDNIGHT_SECONDS(t2))
```

For convenience, you can also create an SQL user-defined function of the above:

```
CREATE FUNCTION secondsdiff(t1 TIMESTAMP, t2 TIMESTAMP)
RETURNS INT
RETURN (
(DAYS(t1) - DAYS(t2)) * 86400 +
(MIDNIGHT_SECONDS(t1) - MIDNIGHT_SECONDS(t2))
)
@
```

If you need to determine if a given year is a leap year, here is a useful SQL function you can create to determine the number of days in a given year:

```
CREATE FUNCTION daysinyear(yr INT)
RETURNS INT
RETURN (CASE (mod(yr, 400)) WHEN 0 THEN 366 ELSE
        CASE (mod(yr, 4))   WHEN 0 THEN
        CASE (mod(yr, 100)) WHEN 0 THEN 365 ELSE 366 END
        ELSE 365 END
 END)@
```

Finally, here is a chart of built-in functions for date manipulation. The intent is to help you quickly identify a function that might fit your needs, not to provide a full reference. Consult the SQL Reference for more information on these functions.

SQL Date and Time functions are as follows:

- **DAYNAME**: Returns a mixed case character string containing the name of the day (e.g., Friday) for the day portion of the argument.
- **DAYOFWEEK**: Returns the day of the week in the argument as an integer value in the range 1-7, where 1 represents Sunday.
- **DAYOFWEEK_ISO**: Returns the day of the week in the argument as an integer value in the range 1-7, where 1 represents Monday.
- **DAYOFYEAR**: Returns the day of the year in the argument as an integer value in the range 1-366.
- **DAYS**: Returns an integer representation of a date.
- **JULIAN_DAY**: Returns an integer value representing the number of days from January 1, 4712 B.C. (the start of Julian date calendar) to the date value specified in the argument.
- **MIDNIGHT_SECONDS**: Returns an integer value in the range 0 to 86 400 representing the number of seconds between midnight and the time value specified in the argument.
- **MONTHNAME**: Returns a mixed case character string containing the name of month (e.g., January) for the month portion of the argument.
- **TIMESTAMP_ISO**: eturns a timestamp value based on date, time or timestamp argument.
- **TIMESTAMP_FORMAT**: Returns a timestamp from a character string that has been interpreted using a character template.
- **TIMESTAMPDIFF**: Returns an estimated number of intervals of the type defined by the first argument, based on the difference between two timestamps.
- **TO_CHAR**: Returns a character representation of a timestamp that has been formatted using a character template. TO_CHAR is a synonym for VARCHAR_FORMAT.

- **TO_DATE**: Returns a timestamp from a character string that has been inter-preted using a character template. TO_DATE is a synonym for TIMESTAMP_FORMAT.
- **WEEK**: Returns the week of the year of the argument as an integer value in range 1-54. The week starts with Sunday.
- **WEEK_ISO**: Returns the week of the year of the argument as an integer value in the range 1-53.

Changing the date format

A common question I get often relates to the presentation of dates. The default format used for dates is determined by the territory code of the database (which can be specified at database creation time). For example, my database was created using `territory=US`. Therefore the date format looks like the following:

```
values current date
1
----------
05/30/2003

1 record(s) selected.
```

That is, the format is MM/DD/YYYY. If you want to change the format, you can bind the collection of db2 utility packages to use a different date format. The formats supported are:

- **DEF**: Use a date and time format associated with the territory code.
- **EUR**: Use the IBM standard for Europe date and time format.
- **ISO**: Use the date and time format of the International Standards Organization.
- **JIS**: Use the date and time format of the Japanese Industrial Standard.
- **LOC**: Use the date and time format in local form associated with the territory code of the database.
- **USA**: Use the IBM standard for U.S. date and time format.

To change the default format to ISO on windows (YYYY-MM-DD), do the following steps:

1. On the command line, change your current directory to `sqllib\bnd`.
   For example:
   On Windows: `c:\program files\IBM\sqllib\bnd`
   On UNIX:`/home/db2inst1/sqllib/bnd`
2. Connect to the database from the operating system shell as a user with SYSADM authority:
   ```
   db2 connect to DBNAME
   db2 bind @db2ubind.lst datetime ISO blocking all grant public
   ```

   (In your case, substitute your database name and desired date format for DBNAME and ISO, respectively.)

Now, you can see that the database uses ISO date format:

```
values current date
1
----------
2003-05-30

  1 record(s) selected.
```

Custom Date/Time Formatting

In the last example, we demonstrated how to change the way DB2 presents dates in some localized formats. But what if you wish to have a custom format such as 'yyyymmdd'? The best way to do this is by writing your own custom formatting function.

Here is the UDF:

```
create function ts_fmt(TS timestamp, fmt varchar(20))
returns varchar(50)
return
with tmp (dd,mm,yyyy,hh,mi,ss,nnnnnn) as
(
    select
    substr( digits (day(TS)),9),
    substr( digits (month(TS)),9) ,
    rtrim(char(year(TS))) ,
    substr( digits (hour(TS)),9),
    substr( digits (minute(TS)),9),
    substr( digits (second(TS)),9),
    rtrim(char(microsecond(TS)))
    from sysibm.sysdummy1
    )
select
case fmt
    when 'yyyymmdd'
        then yyyy || mm || dd
    when 'mm/dd/yyyy'
        then mm || '/' || dd || '/' || yyyy
    when 'yyyy/dd/mm hh:mi:ss'
        then yyyy || '/' || mm || '/' || dd || ' ' ||
               hh || ':' || mi || ':' || ss
    when 'nnnnnn'
        then nnnnnn
    else
        'date format ' || coalesce(fmt,' <null> ') ||
        ' not recognized.'
    end
from tmp
</null>
```

The function code may appear complex at first, but upon closer examination, you'll see that it is actually quite simple and elegant. First, we use a common table expression (CTE) to strip apart a timestamp (the first input parameter) into its individual components. From there, we check the format provided (the second input parameter) and reassemble the timestamp using the requested format and parts.

The function is also very flexible. To add another pattern simply append another WHEN clause with the expected format. When an unexpected pattern is encountered, an error message is returned.

Usage examples:

```
values ts_fmt(current timestamp,'yyyymmdd')
 '20030818'
values ts_fmt(current timestamp,'asa')
 'date format asa not recognized.'
```

Summary

These examples answer the most common questions I've encountered on dates and times. I'll update this article with more examples if feedback suggests that I should. (In fact, I've updated this three times already... thanks to readers).

Acknowledgements

Bill Wilkins, DB2 Partner Enablement
Randy Talsma

Disclaimer

Resources

Learn

- Visit the developerWorks resource page for DB2 for Linux, UNIX, and Windows to read articles and tutorials and connect to other resources to expand your DB2 skills.
- Visit the developerWorks Information Management zone to find more resources for DB2 developers and administrators.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Attend a free developerWorks Live! briefing to get up-to-speed quickly on IBM products and tools as well as IT industry trends.
- Follow developerWorks on Twitter.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.

Get products and technologies

- Now you can use DB2 for free. Download DB2 Express-C, a no-charge version of DB2 Express Edition for the community that offers the same core data features as DB2 Express Edtion and provides a solid base to build and deploy applications.
- Build your next development project with IBM trial software, available for download directly from developerWorks.
- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement Service Oriented Architecture efficiently.

Discuss

- Get involved in the My developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the author

Paul Yip

Paul Yip is a database consultant from the IBM Toronto Lab where DB2 for distributed platforms is developed. His primary work involves helping companies migrate applications from other databases to DB2 and educating experienced DBAs on how to map their existing skills to the DB2 world. He is the author of several DB2 articles and white papers and likes to write reactively to customer needs. Paul can be reached at ypaul@ca.ibm.com