This month, I'll discuss the TIMESTAMP data type, or, to use its SQL Server name, the DATETIME data type. My objective is to compare the functionality of IBM DB2 v8.1, Microsoft SQL Server 2000, and Oracle9i (the Big Three) with respect to the data type's range and precision, as well as time zones. Hopefully you'll get a clear idea along the way about why TIMESTAMP is useful, and why using it can be dangerous.

**Format**

There are many ways to write a date and time, and it would be tedious to list all possible formats and the ways to convert between them. Without trying to pick sides, I'll just note that the ANSI/ISO SQL Standard format is:

```
TIMESTAMP 'YYYY-MM-DD HH:MM:SS [.fractional-seconds]'
```

For example, the literal `TIMESTAMP '2003-07-15 12:00:00.5'` refers to half a second past noon on July 15 2003 AD.

It's necessary to warn that this format should not be called simply the "ISO format" because the ISO 8601 and the ISO SQL timestamp format can be slightly different. (Note: ISO-8601: 1988(E) "Data elements and interchange formats - Information interchange - Representation of dates and times" is the International Standard that specifies numeric representations of date and time.)

It's also necessary to warn that this format is not the default for any of the three DBMSs under discussion. But, in an attempt to use a common language, I'll always use the ISO SQL format for timestamps – except for the date **January 1 4713 BC**. I'll explain the exception in a moment.

**Range**

The range of a data type is its minimum and maximum values. Table 1 shows the TIMESTAMP range offered by the Big Three.

<table>
<thead>
<tr>
<th>DBMS</th>
<th>MINIMUM</th>
<th>MAXIMUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>0001-01-01 00:00:00</td>
<td>9999-01-01 23:59:59</td>
</tr>
<tr>
<td>Oracle</td>
<td>January 1 4713BC 00:00:00</td>
<td>9999-01-01 23:59:59</td>
</tr>
<tr>
<td>SQL Server</td>
<td>1753-01-01 00:00:00</td>
<td>9999-01-01 23:59:59</td>
</tr>
</tbody>
</table>
Table 1: TIMESTAMP range.

The maximum timestamp is the same for all, and the year 9999 is far in the future. So the only point worth remarking on is that some old Oracle books report a maximum year value of 4712 rather than 9999.

The minimum year is where DBMSs differ. DB2's initial date is January 1 of the year 1, which is the required SQL Standard minimum. SQL Server is much more cautious; its calendar begins at the start of the year that followed the switch from the Julian to the Gregorian calendar in England and England's colonies, including what's now the USA. But Oracle's minimum date is the interesting one:

- The date January 1 4713BC is Oracle's actual minimum, but Oracle doesn't know it. In fact, Oracle falsely claims that the minimum date supported is January 1 4712BC. This is one of the three "4713BC bugs" or "Y-4K". You'll find a full description of these bugs in the appendix to this article. For anybody who wants to panic about them, I am available for consulting at the usual Y2K rates.

- Oracle date-arithmetic calculations start to differ from DB2 calculations for any dates earlier than October 15 1582, because for dates prior to that, Oracle uses the Julian calendar while DB2 uses the Gregorian calendar. This can be a fascinating topic, but it's not the immediate problem. If you want to follow it further, check the references about DBMS calendars at the end of this article.

Oracle's date range is the widest, but DB2's consistent use of the Gregorian calendar is more SQL Standard-conformant.

The Maximum Second

In Table 1, the largest second in a minute is given as 59, since the seconds in a minute are generally numbered from 0 to 59. Although that might seem obvious and uncontroversial, it's not 100% true. It's always the exceptions that are interesting, so let's look at them.

- First exception: DB2 happens to allow a time of '24:00:00' for midnight. ('24:00:01' is not legal.) This is an extension that is probably best avoided, because it's not compatible with other DBMSs, and because it's hard to distinguish '1999-01-01 24:00:00' from '1999-01-02 00:00:00'. Nevertheless, it might be necessary to correspond to legal documents that refer to "midnight on January 1" because in such documents midnight means the end of the day, not the beginning.

- Second exception: At irregular intervals, but always as the last second at the end of a calendar quarter, the International Earth Rotation Service (IERS) declares a leap second. (June 30th and December 31st are the preferred dates; since the system was introduced in 1972, only dates in June and December have been used.) That means the day has (24 x 60 x 60 + 1) 86,401 seconds instead of the usual mere 86,400 – which is why counting seconds between two wide-apart timestamps is almost certainly unreliable.

The SQL Standard actually allows for two leap seconds, so the last second on June 30 1972, for example, could have been '1972-06-30 23:59:61'. But in fact, it wasn't — the IERS has never declared a double leap second.
UTC

The leap second is a feature of the system known as UTC, which are the initials for "Coordinated Universal Time". Well, obviously they aren't really, but "CUT" would be the English initials and "TCU" would be the French initials, so perhaps "UTC" is a compromise.

"Do not confuse UTC with the old standard 'Greenwich Mean Time' (GMT) – GMT was a variant of UT1 that used a different method to correct for fluctuations. Beware of two prevalent but false opinions: that years are not getting longer (they are), or that UTC is a renaming of GMT (it is not). The distinctions are tiny, but any program which uses leap seconds or fractional seconds is getting into magnitudes which are smaller than those tiny distinctions."

— SQL-99 Complete, Really, by Peter Gulutzan and Trudy Pelzer

It's unfortunate that the manuals of all three DBMSs — DB2, Oracle, and SQL Server — refer to GMT, which is 22 seconds different from UTC at time of writing (because of leap seconds). The fact is, the time you're getting from any official source is the UTC time. And although you can set the initial time to UTC, your computer probably does calculations ignoring leap seconds, meaning it really uses what's called "floating TAI" rather than "UTC" for arithmetic.

Time Zones

Organizations that operate in multiple states or countries can have a problem: The sun rises at different times in different longitudes. This fact was known to the ancient Greeks, but didn't become a worry until the coming of the railway and the telegraph in the nineteenth century. The generally accepted solution is the "time zone," which is often represented by a displacement value from UTC.

The standard way to show a displacement is:

[+| -]HH:MM

Here, the optional +| - means add or subtract from the local clock, and HH:MM represents a number of hours and minutes (leading zeros are often omitted). The 00:00 displacement is UTC and happens to be the non-summer, or "standard" time in England, Portugal, Tunisia, and Namibia, among others. If you are west of the zero meridian, you must add a displacement — for example, the US Pacific coast is UTC +08:00. If you are east of the zero meridian, you must subtract a displacement — thus Moscow standard time is UTC -03:00.

It's common to see time-plus-displacement in everyday work now, for example it's shown on email headers. However, the DBMS support for time zones varies, from almost nil to thorough. Table 2 shows the support provided by the Big Three.

<table>
<thead>
<tr>
<th>DBMS</th>
<th>TIME ZONE SUPPORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>CURRENT TIMEZONE function returns the difference between UTC and local time at the application server. Subtracting CURRENT TIMEZONE from a local time converts that local time to UTC.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Allows SET TIME_ZONE = {local</td>
</tr>
</tbody>
</table>
Table 2: Support for time zones.

Question: Should a TIMESTAMP column contain:

(a) the UTC date and time, or
(b) the local date and time according to the server's location, or
(c) the date and time plus the displacement?

DB2 is at one extreme on this issue. The DB2 online documentation says:
"The timestamp string is converted to GMT on the server, so the time represents the server's time zone, not the client's. If the client is in a different time zone from the server, the server's local time should be used."

Ignoring the obsolete mention of GMT, what this means is that DB2 assumes you choose (a). There's not much flexibility here.

SQL Server has slightly more flexibility because it allows for getting the current timestamp as a UTC value - so we can say it's in the middle on this issue. Microsoft recommends that you store time values in UTC.

At the other extreme, we have Oracle. Oracle's description is rather lengthy because, as usual, Oracle allows multiple formats and shorthands, and different ways to do the same thing. But it's possible to sum it up this way:

"The timestamp is stored in the database according to a database time zone, which administrators can set. Each client can declare its own zone with ALTER SESSION, or can pass a full (timestamp and displacement) value to the server in timestamp literals. The DBMS converts to the database time zone before storing. When a client retrieves, the timestamp is automatically converted back from the database time zone to the session time zone."

Thus with Oracle you can use either (a) or (b). But nobody really allows for (c) ...

timestamps with TIMESTAMP data type are stored in a normalized form for space-saving and easy-sorting reasons.

Oracle also allows TIMESTAMP WITH TIME ZONE as a data type specification, which aligns it nicely with the standard SQL non-core requirements. DB2 and SQL Server have no equivalent. Indeed, except for some small open-source products, few DBMSs have as many time-zone-related features as Oracle. On the down side, there have been security problems related to time zones in Oracle's dark past (see the references at the end of this article).

Daylight Savings Time
If you store timestamps in UTC or standard time, then adding and subtracting timestamps will be reliable to the nearest minute (though not better than that, because of the leap-second complication). On the other hand, if you use local clock time, then arithmetic is thrown off because:

(a) the number of hours is not equal to the number of days times 24, if a switch to/from daylight savings occurred in the interval;

(b) on the day that the clocks are set back, the same timestamp, for example '2003-10-26 04:00:00' can appear twice;

(c) not everybody uses daylight savings time.

In the USA, an extreme example of the last point is Arizona. In most parts of the country, a switch to daylight savings time normally occurs the first Sunday in April. But in Arizona that doesn't happen — Arizona stays on standard time. However, in the Navajo Indian Reservation in northeast Arizona, the switch does happen because the reservation extends into New Mexico, and the Navajo prefer to use the same time throughout the reservation.

Can you rely on your DBMS to adjust calculations according to time zone? Definitely not. Sometimes this causes consternation. For example here is a Usenet complaint regarding Sybase:

"We found our database server (Sybase SQL Server 11) doesn't understand DaylightSavingTime; that is, it couldn't distinguish between 1997/10/26 2:30 a.m. EDT and 1997/10/26 2:30 a.m. EST, and thought 1997/10/26 2:59 a.m. EDT was after 1997/10/26 2:01 a.m. EST (instead of being a couple of minutes before). I imagine Microsoft SQL Server has the same problem."
— Seen on the Internet

**Precision**

Timestamps can include fractional seconds, as shown in Table 3. The number of digits after the decimal point is sometimes called the "fractional precision."

<table>
<thead>
<tr>
<th>DBMS</th>
<th>MINIMUM</th>
<th>MAXIMUM</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>0</td>
<td>6 (to nearest microsecond)</td>
<td>6</td>
</tr>
<tr>
<td>Oracle</td>
<td>0</td>
<td>9 (to nearest nanosecond)</td>
<td>6</td>
</tr>
<tr>
<td>SQL Server</td>
<td>0</td>
<td>3 (to nearest millisecond)</td>
<td>3</td>
</tr>
</tbody>
</table>

*Table 3: Fractional seconds precision support.*

Oracle allows for greater fractional precision than either DB2 or SQL Server, but it's notorious that precision and accuracy are not the same thing. Sometimes a computer clock time is based on a ticker that gets incremented every 1/60th of a second, which means that storing nanoseconds based on CURRENT_TIME's return is pointless. So, although any DBMS should allow for at least six digits of fractional precision because that's the SQL Standard requirement, the average shop can get by with less, and should explicitly say that the precision is less than the default.
(NOTE: Although SQL Server allows for 3 digits of fractional precision, Microsoft's documentation for its datetime (i.e., timestamp) data type says: "Date and time data from January 1, 1753, through December 31, 9999, with an accuracy of three-hundredths of a second, or 3.33 milliseconds.")

Verdicts

- The idea of using {CURRENT_TIMESTAMP combined with node identifier} as a unique transaction moniker is possibly dangerous because of the daylight saving time switch, and possibly non-portable because fractional precision can be as small as three.

- Oracle's TIMESTAMP data type has a wider range, more flexible zone support, and greater fractional precision than either DB2 or SQL Server can offer.

- Because of leap seconds and daylight savings time, and because precision is not the same as accuracy, arithmetic with timestamps can occasionally return incorrect results.

References

Celko, Joe. "Killing Time"

- An earlier DBazine.com mention of time-specific troubles.

Feuerstein, Steven. "New Datatypes, New Possibilities."

U.S. Naval Observatory's Time Service Department: Leap Seconds

- The most-often-referenced site for leap seconds.

Oracle TO_TIMESTAMP_TZ Remote System Buffer Overrun; Multiple Vulnerabilities in Oracle Database and Application Servers -- 24 February 2003

- Notes about security breaches related to time zones on Oracle.

Oracle Corporation. "New Datetime Data Types in Oracle9i."

- A slightly more detailed look at Oracle syntax than the one in this article.

Note about the SQL Standard:

References in this article to the SQL Standard can be checked against the SQL:2003 "Sneak Peek" draft documents, especially the SQL/Foundation, at ftp://sqlstandards.org/SC32/WG3/Progression_Documents/FDIS/4FDIS1SP1-02-Foundation-2003-04-25.pdf

This site may disappear soon, so get your free .PDF download of the next SQL Standard while you can.

Appendix
"The Oracle Calendar"

☞ How Oracle handles dates — in an "Oracle FAQ" site.

---

Trudy Pelzer has worked with SQL since 1986. She has co-authored four books, including one thick book about the SQL Standard (SQL-99 Complete, Really) and one thin book about portability and optimization for DB2, Oracle, SQL Server, and five other DBMSs (SQL Performance Tuning). This is Trudy's third article for DBAzone.
Studytonight Online DBMS Tests have the best questions to make you understand the topic well. After each section you study, give the related Practice test. DBMS MCQs Test 1. This Test will cover complete DBMS with very important questions, starting off from basics to advanced level. Q. Consider Join of a relation R with a relation S. If R has m tuples and S has n tuples, then maximum and minimum sizes of the Join respectively are.

5.) dbms systems commonly has a special language called sql to modify and reach the data easily. 6.) dbms systems has indexes to reach the data, not read it line by line. 7.) there are much more control mechanisms that approves the correctness of data. 

CARDINALITY OF RELATIONSHIP: The minimum and maximum values of this connectivity is called the cardinality of the relationship e.g., one to one, one to many, many to many, weak entity. 2.1.1 What is Data, Database System, DBMS? 2.1.2 Single and Multi-User Systems 2.1.3 Advantages and Drawbacks of DBMS 2.1.4 Architecture of DBMS 2.1.5 Users of DBMS 2.1.6 Roll of Database Administrator 2.2 Components of DBMS 2.3 Types of DBMS Hierarchical Network Relational

2.4 Why RDBMS? The structure of a Database Management System and its components are presented. After studying this chapter you will be able to explain DBMS and discuss advantages and drawbacks of DBMS. Summary of DBMS_STATS Subprograms. 138 DBMS_STATS. With the DBMS_STATS package you can view and modify optimizer statistics gathered for database objects. See Also: Oracle Database Performance Tuning Guide. Converts user-specified minimum, maximum, and histogram endpoint datatype-specific values into Oracle's internal representation for future storage using the SET_COLUMN_STATS Procedures.

PREPARE_COLUMN_VALUES_NVARCHAR2 Procedure.