& Claiborne 1966]. Overall Carbon-14 dating is accurate to within at best 50 to 1000 years of the age of the sample.

Carbon-14 radioactive dating works with organic materials, but many scientists, especially geochronologists, need to date inorganic materials. Geochronologists have developed a battery of radioactive dating techniques based on ratios of isotopes of lead and uranium, rubidium and argon, and potassium and strontium, among others [Hood 1969]. The ratios of these isotopes are fixed in inorganic materials when molten rock solidifies into igneous rock. Unequal decay rates change the ratio and can be used to date the igneous layers. The methods used to determine the ratio are similar to the technique of Carbon-14 dating, but the measurements yield dates that are much older, and consequently, less precise. The date that results from inorganic radioactive dating techniques and the error bars on the dating process are in terms of millions of years.

Another clock used by stratigraphers to date inorganic materials is the *sedimentation clock* [Silverberg 1971]. Sediments are deposited in layers (with perhaps intervening igneous layers). The relative age of sedimentary strata can be deduced from their position in the layering; usually the oldest layers are on the bottom while the youngest are on top, but tectonic activity may permute the ordering.

The polarity of the Earth's magnetic field offers yet another geologic clock. Samples drilled from basalt layers in the ocean floor indicate that the Earth's polarity switches, sometimes as rapidly as every few million years. The change in polarity is believed to be the result of a change in the direction of the swirling mass of molten magma at the core. Polarity reversals are detectable in ancient basalt outflowings on the ocean floor [Kirkaldy 1977].

This brief survey of the important clocks used by scientists and the general public is by no means complete. The details on clocks presented here only provide the necessary background to understanding the base-line clock. For a more in-depth treatment of an individual clock, the interested reader is directed to the reference(s) cited with each clock discussion.

tropical years in terms of days, while common to most calendars, is a very difficult problem.

Dendrochronology is the study of the *tree-ring record*. A tree trunk viewed in cross-section will often exhibit a series of concentric rings. The rings are a by-product of the uneven seasonal growth of the tree. In the northern hemisphere, a growth spurt occurs in the Spring and early Summer, with little or no growth during the Fall and Winter. The growth rings are of nonuniform width since the growth depends on nonconstant variables such as yearly climatic conditions. Also, the rings only occur in some tree species.

A tree-ring record is constructed by correlating rings from many different trees. Since tree-ring records depend on local climatic conditions, they are specific to a geographic region. Tree ring records in the Southwestern United States are different from those in the Northwest. However, tree-rings are a very accurate type of archaeological dating techniques. A tree-ring record spanning 770 years of English weather and accurate to the season was pieced together in 1838 by Charles Babbage, whom some claim to be the father of the modern computer. In the United States, a tree-ring record of the past 3000 years has been made from Sequoias in central California. An even more ambitious tree ring record dating back to 5112 B.C. was made from Bristlecone Pine cores at the University of Arizona's Laboratory of Tree-Ring Research [Silverberg 1971].

Another kind of archaeological dating is based on the decay rate of Carbon-14. Carbon-14 is a naturally occurring unstable isotope of Carbon. It has a half-life of somewhere between 5,700 and 5,770 years. New Carbon-14 is constantly being produced in the atmosphere as a by-product of the cosmic radiation bombardment of nitrogen. The Carbon-14 binds with oxygen to form carbon dioxide. It is assumed that the amount of Carbon-14 in the environment is uniform. But it is somewhat rarer than some other isotopes of Carbon. In particular, for every trillion Carbon-12 atoms there is approximately one Carbon-14 atom.

Plants take in both Carbon-12 and Carbon-14 atoms in carbon dioxide from the atmosphere during photosynthesis. From plants, Carbon-14 is passed up the food chain to other organisms. During the lifetime of an organism the ratio of Carbon-12 to Carbon-14 remains at a trillion to one. But after the organism dies no more Carbon-14 or Carbon-12 is accumulated. As the Carbon-14 decays, the ratio of Carbon-12 to Carbon-14 increases (Carbon-12 is a stable isotope). Determining the ratio is a complicated procedure. It involves counting the number of Carbon-14 disintegrations per hour per gram of carbon in the sample. Because of the probabilistic nature of radioactivity, the number of disintegrations per hour is not fixed, but is described by a known probability distribution. Intuitively, the distribution captures the notion that it is more likely that a recent sample produces more disintegrations per hour than does an ancient sample. The average rate can be probabilistically estimated after several hours of counting. Longer periods of observations will lead to increasingly more probable estimations of the ratio.

Once the ratio has been determined, it can be used to date the remains of a deceased organism. For instance, 5,700 to 5,770 years after the death of an organism approximately half of its original Carbon-14 has decayed. The accuracy of Carbon-14 dating depends on assumptions about the rate of Carbon-14 decay, the uniform presence and rate of renewal of Carbon-14 atoms in the absence of other sources of Carbon-14 contamination, and the precision of the instruments used to measure the amount of Carbon-14. On top of this, 45,000 to 70,000 years after the death of an organism, Carbon-14 dating is impossible since only an undetectable amount is left in a sample [Goudsmit

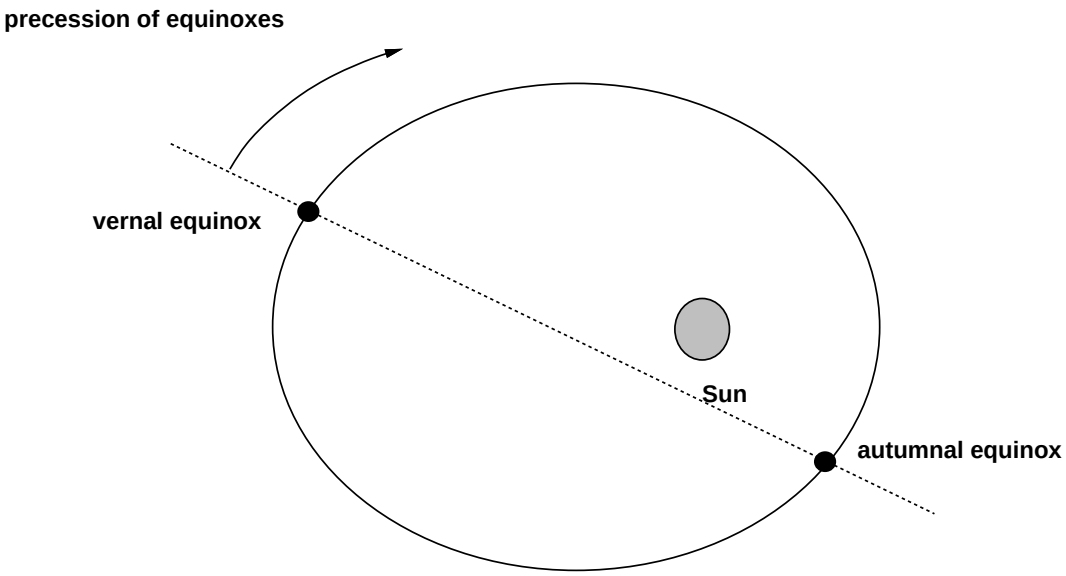

Figure 13: Precession of the equinoxes, a ≈26,000 year cycle [Schultz 1986]

365.242,191 ephemeris days, and the sidereal year was 365.256,363 ephemeris days [Fraser 1987]. The anomalistic year is slightly longer than the sidereal year because of the progression of the line of the apsides. And the sidereal year is slightly longer than the tropical year because of the precession of the equinoxes.

The Gregorian calendar, like most calendars, is designed to keep count of the tropical year. Recall that the tropical year is the year with respect to the seasons. Keeping track of the seasons is important for both biological and cultural reasons. But since the Gregorian calendar counts days and there are not an integral number of days in the tropical year, an *intercalation* must be made to keep the calendar year roughly equivalent to the tropical year. The intercalation used in the Gregorian calendar adds a leap day in years divisible by four, except for those years that happen to fall on a century mark and the century count is not evenly divisible by four. This rule is actually not quite good enough. Even with this intercalation rule the Gregorian year is approximately 25 seconds faster than the tropical year. Twenty-five seconds may not seem like very long, but over time, this difference accumulates. In 3,330 years, the Gregorian calendar will be one day out of sync with the tropical year. Other reforms have been proposed to help keep the Gregorian calendar in sync. In 1923, an Eastern Orthodox congress held in Constantinople agreed to legislation that stated that century years divisible by 900 will be leap years only if the remainder is 200 or 600. This keeps the Gregorian calendar accurate to 1 day in 40,000 tropical years [Aveni 1989].

In general, simple intercalation rules are not enough in the long term to keep calendar years, counted in terms of days, and tropical years in step. The number of days in the year is gradually decreasing due to the long-term slowdown in the Earth's rate of rotation. About 400 million years ago, there were approximately 400 solar days in a tropical year. Today there are between 365 and 366. Also the length of the tropical year in relation to other clocks is not a fixed period. In terms of ephemeris days, it is slowly decreasing at the rate of 0.530 seconds per century due to a complex gravitational interaction of the Earth, Sun, and Moon [Thomson 1978]. In summary, counting

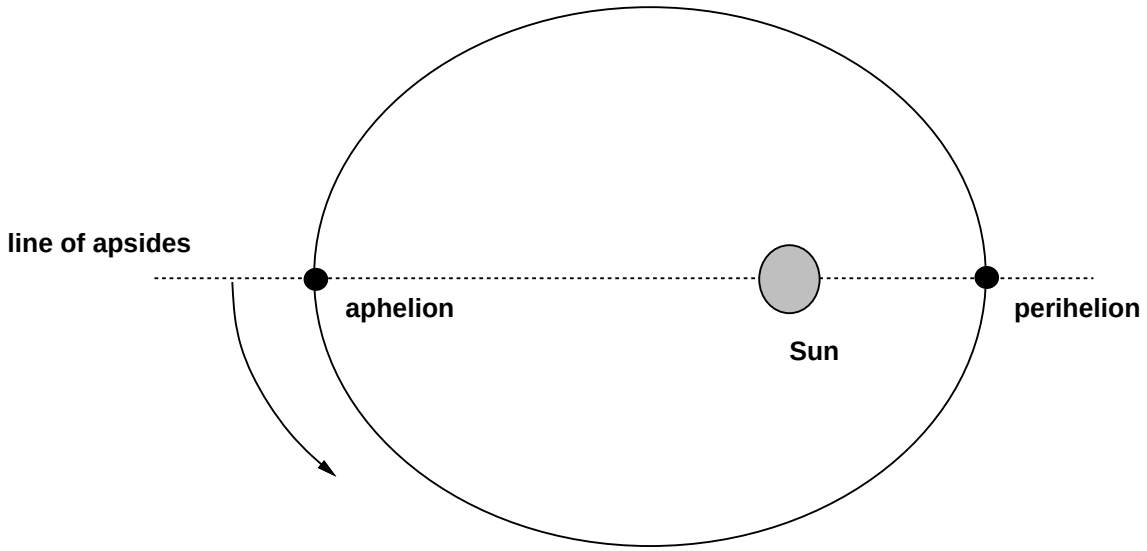

Figure 12: Progression of the line of apsides, a ≈110,000 year cycle [Schultz 1986]

takes takes approximately 110,000 years as illustrated in Figure 12 [Schultz 1986].

The unit of the second primary clock is called the *tropical year*. The tropical year is the time between successive passages of the Sun through the vernal equinox. An equinox occurs when the Sun's path, known as the ecliptic, intersects the celestial equator. A less precise definition of an equinox is the day when the duration of sunlight and darkness are the same. It happens twice a year; once in the spring, the vernal equinox, and once in the Fall, the autumnal equinox. The equinoxes are important events because they mark the beginning of the Fall and Spring seasons. The tropical year can be roughly thought of as the year with respect to the seasons. The equinoxes do not occur in the same place in the orbit each year. The equinoxes gradually migrate through the Earth's orbit in a clockwise direction in what is called the *precession of the equinoxes*. This migration is due to a slow, cyclic wobbling of the Earth around its axis. The wobbling moves the celestial equator a fraction each year. But this movement is enough to affect the measurement of the tropical year [O'Neil 1975]. One complete precessional cycle takes about 26,000 years. The precession of the equinoxes is depicted in Figure 13.

A *sidereal year* is another clock unit based on the Earth's orbit. But the sidereal year looks to the distant stars, rather than the Sun, to determine when a year has elapsed. Since the distant stars are a fixed frame of reference, the start of the sidereal year occurs in the same place in the Earth's orbit every year.

In general, *calendar years* are related to, yet independent of, celestial year clocks. There are a bewildering variety of calendars, too many to describe even briefly in this paper [Pavise 1982]. In order to simplify the present discussion we shall adopt the Gregorian calendar as our prototypical calendar. Calendar years are typically an integral number of days, yet there are not (currently) an integral number of days in any astronomically computed year. For example, for the Gregorian calendar year 1985, the anomalistic year was 365.259,641 ephemeris days, the tropical year was

in the northern hemisphere's Spring and fast in its Fall; and unpredictable variations, probably caused by differing rates of rotation between the core and the mantle. The gradual slowing of the rate of rotation adds about 1.5 milliseconds to the length of a day (in comparison to other clocks) during a century. The length of a day could fluctuate by 4 milliseconds over the course of a decade due to the unpredictable variations. Finally, seasonal variations can cause changes on the order of 1.2 milliseconds in the length of a day during a year [Howse 1980].

Another factor to consider in making precise sidereal time measurements is *polar wander*. Polar wander is a slight circular wobble of the Earth around the North/South pole on the order of 8 meters a year. Polar wander shifts an observer's meridian by a fraction each day (recall that the observer's meridian is used to determine when the sidereal and solar days start). The size of the shift depends on the observer's latitude.

The family of *universal times* attempts to correct for these variations. UT0 is the mean solar time for the prime meridian computed by direct astronomical observations. The prime meridian is the 0^o line of longitude that runs through Greenwich Observatory in England. Corrections for polar wander applied to UT0 yields a new measure of time called UT1. Further corrections for seasonal variations are used to compute UT2 from UT1 [Thomson 1978]. Since 1958, the UK time service has broadcast yet another time, *Coordinated Universal Time* (UTC). Prior to 1972, UTC measured time by comparing a quartz clock with UT2. The announced time was kept to within 0.1 seconds of UT2 by adding offsets. In 1972, UTC was redefined to be the relationship between TAI and UT1; UTC must be within 0.9 seconds of UT1. The International Earth Rotation Service (IERS) currently has the responsibility for computing UTC and making leap seconds adjustments. The difference between UTC and UT1, which is important for navigational purposes, is also broadcast [Howse 1980].

The clocks that are mostly of astronomical use are maintained by several observatories throughout the world. However, there are some distinguished observatories. The Royal Greenwich Observatory maintains Greenwich Mean Time (GMT), which is the mean solar day clock at the prime meridian. The BBC has broadcast GMT since 1924. GMT was the standard for universal time prior to UTC. Another important organization is the Bureau International de l'Heure (BIH) at the Paris Observatory. BIH used to maintain UT1 and determine polar wander, but in 1987 that responsibility was transferred to the International Earth Rotation Service [Quinn 1991].

A naturally occurring physical phenomenon with a stability approaching (or perhaps exceeding) that of the best atomic clocks is a *millisecond pulsar*. A millisecond pulsar is a rapidly rotating neutron star. A neutron star is the collapsed remnant of a supernova explosion. As it rotates, the star emits periodic broad-band radio pulses which can be detected here on Earth. Over 30 of these remarkable clocks have been identified to date [Taylor 1991].

There are three primary clocks based on the Earth's orbit around the Sun. As noted before, the Earth's orbit is slightly elliptical. One orbital measurement is based on the time needed for the Earth to travel from aphelion to perihelion back to aphelion. The period is known as the *anomalistic year*. The perihelion and aphelion do not occur in the same place in the orbit each year. There is an imaginary line, known as the *line of apsides*, connecting perihelion and aphelion. The line is not stationary, it gradually migrates in a counterclockwise direction through the Earth's orbit. This is known as *progression of the line of apsides*. A complete cycle in the progression

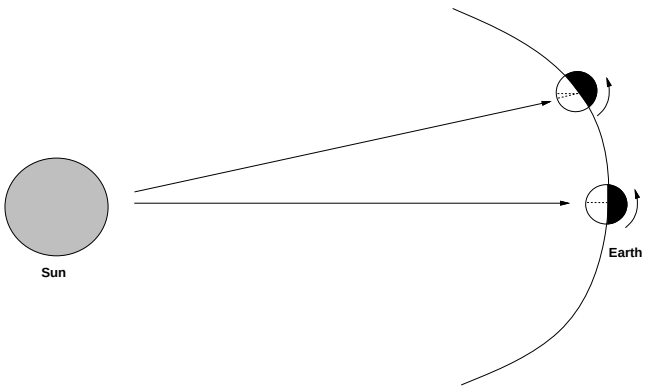

Figure 10: The orbital motion of the Earth is noticeable in an *apparent solar day* [O'Neil 1975]

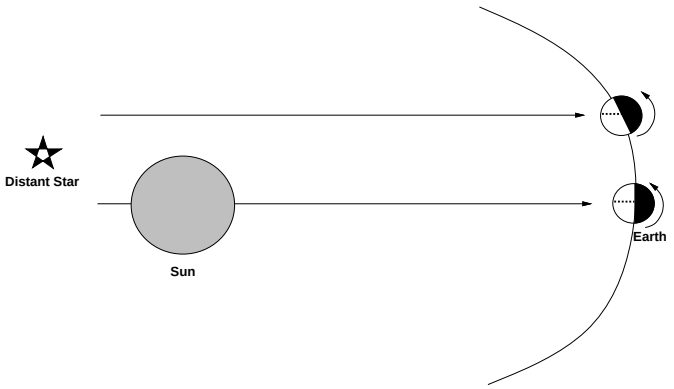

Figure 11: The effects of orbital motion are negligible in a *sidereal day* [O'Neil 1975]

to a distant star (not the Sun). It is the time between successive crossings of a earthly observer's meridian by a distant star. This can be measured using a telescope that can only move up and down along the meridian (called a meridian circle) [Schultz 1986]. Because the orbital motion of the Earth with respect to the distant stars is negligible, the Earth rotates exactly 360 degrees in one sidereal day. This is illustrated in Figure 11.

The sidereal day differs from the solar day in one important respect; a sidereal day is slightly shorter than a solar day. The Earth rotates a tiny fraction more during a solar day than it does during a sidereal day, because the effect of orbital motion is negligible when observing a distant star. Currently, in solar time, the solar day is 24^h and the sidereal day is $23^h50^m4.09055^s$ (although this relationship varies with changes in the Earth's orbit). In sidereal time, the sidereal day is 24^h and the solar day is $24^h03^m56.55537^s$ [Fraser 1987].

Accurate astronomical predictions of celestial bodies depend on an unvarying fundamental unit of time. Unfortunately, a sidereal day is not a fixed period, it also fluctuates in duration (in comparison to other clocks). The fluctuation is due to a combination of factors that affect the rate of the Earth's rotation. These factors include a gradual overall slowing of the rate of rotation thought to be caused by tidal friction in the shallow seas; seasonal variations, the Earth runs slow

Appendix: Clocks

In this appendix we provide a more detailed examination of the clocks discussed briefly in Section 3. Due to the diversity of physical processes underlying these clocks, and the variety of scientific disciplines represented, information on these clocks is widely scattered [Das 1990]. Hence, we go into some detail in our discussion to provide a comprehensive overview of some two dozen of the most important clocks. These details are also required to fully understand the motivation behind the base-line clock we propose in Section 3.2.

An *apparent solar day* is one rotation of the Earth with respect to the Sun. It is the time between successive "noons". Noon is when the Sun crosses the meridian of a stationary earthbound observer. A meridian is a line of longitude. An apparent solar day begins at noon and ends just before the next day's noon. During an apparent solar day, the Earth not only spins on its axis, it also moves in orbit around the Sun. Because of the orbital motion, the Earth must rotate slightly more than 360 degrees during one apparent solar day. This is illustrated in Figure 10.

As far back as Ptolemy, it was known that the length of each apparent solar day fluctuates in comparison to other clocks—Ptolemy could have used a water clock—by as much as 15 minutes during the course of a year (a year will be defined later). This fluctuation is due to two factors. First, the Earth's orbit is elliptical rather than circular. Because it is elliptical, the Earth speeds up from aphelion (the point in the orbit closest to the Sun) to perihelion (the point farthest from the Sun) and slows down from perihelion to aphelion. The change in orbital velocity changes how far the Earth must rotate between successive noons. Second, the Earth is tilted on its axis. An earthly observer placed on the equator can witness the Sun tracing a slightly different route each day. In December the Sun will be below the equator while in July it will be above the equator. The varying positions of the Sun also play a role in alterning the length of the apparent solar day [O'Neil 1975].

The daily fluctuations in the length of the apparent solar day can be "smoothed" using a different method of measurement; this is the basis of the *mean solar day*. Instead of measuring the apparent Sun, this clock measures an imaginary Sun called the "mean Sun". The Earth orbits the mean Sun at a constant rate, with the Sun always in the plane of the celestial equator. A mean solar day is roughly equivalent to an average apparent solar day. For any day of the year, the difference between the apparent solar day and the mean solar day is given by the *equation of time*. From here on, a solar day will refer to a mean solar day rather than an apparent solar day.

One curious aspect of a solar day is that it starts at noon while the day familiar to most of us starts at midnight. The day starting at midnight is called a *civil day*. A civil day is computed from a solar day by shifting the solar day by half a day. In many cultures, each civil day (and each solar day) is commonly subdivided into 24 units called hours (and each hour is subdivided into 60 minutes, etc.). But this is not the only possible division. Other cultures, such as the Chinese, (historically) subdivided the civil day differently. The Chinese split the day into 100 *k'o* (a 14 minute 24 second period) [Fraser 1987].

An altogether different method of measurement is used to determine a *sidereal day*, yet another clock based on the Earth's rotation. One sidereal day is one rotation of the Earth with respect

[Quinn 1991] Quinn, T.J. "The BIPM and the Accurate Measurement of Time." *Proceedings of the IEEE*, 79, No. 9, July 1991, pp. 894–906.

[Rohr 1965] Rohr, R. "Sundials: History, Theory, and Practice." Toronto, Canada: University of Toronto Press, 1965.

[Roy & Clarke 1988] Roy, A. and D. Clarke. "Astronomy: Principles and Practice, 3rd Edition." Bristol and Philadelphia: Adam Hilger, 1988.

[Schultz 1986] Schultz, J. "Movement and rhythms of the stars." Edinburgh: Floris Books, 1986.

[Silverberg 1971] Silberberg, R. "Clocks for the Ages: How Scientists Date the Past." New York: The Macmillian Company, 1971.

[Soo & Snodgrass 1992A] Soo, M. and R. Snodgrass. "Multiple Calendar Support for Conventional Database Management Systems." Technical Report 92-7. Computer Science Department, University of Arizona. Feb. 1992.

[Soo & Snodgrass 1992B] Soo, M. and R. Snodgrass. "Mixed Calendar Query Language Support for Temporal Constants." TempIS Technical Report 29. Computer Science Department, University of Arizona. Revised May 1992.

[Soo et al. 1992] Soo, M., R. Snodgrass, Dyreson, C., Jensen, C. S., and N. Kline. "Architectural Extensions to Support Multiple Calendars." TempIS Technical Report 32. Computer Science Department, University of Arizona. Revised May 1992.

[Tansel & Arkun 1986] Tansel, A.U. and M.E. Arkun. "HQUEL, A Query Language for Historical Relational Databases," in *Proceedings of the Third International Workshop on Statistical and Scientific Databases*. July 1986.

[Taylor 1991] Taylor, J.H., Jr. "Millisecond Pulsars: Nature's Most Stable Clocks." *Proceedings of the IEEE*, 79, No. 9, July 1991, pp. 1054–1063.

[Thomson 1978] Thomson, M. "The Beginning of the Long Dash: a history of timekeeping in Canada." Toronto, Canada: University of Toronto Press, 1978.

[Whitrow 1980] Whitrow, G.J. "The Natural Philosophy of Time." New York, NY: Oxford University Press, 1980.

[Wiederhold et al. 1991] Wiederhold, G., S. Jajodia and W. Litwin. "Dealing with Granularity of Time in Temporal Databases," in *Proc. 3rd Nordic Conf. on Advanced Information Systems Engineering*. Trondheim, Norway: May 1991.

[Wyatt 1971] Wyatt, S. "Principles of Astronomy." Boston, MA: Allyn and Bacon, Inc., 1971.

[Gadia 1986] Gadia, S.K. "Toward a Multihomogeneous Model for a Temporal Database," in *Proceedings of the International Conference on Data Engineering.* IEEE Computer Society. Los Angeles, CA: IEEE Computer Society Press, Feb. 1986, pp. 390–397.

[Goudsmit & Claiborne 1966] Goudsmit, S. and R. Claiborne. "Time." New York: Time Inc., 1966.

[Guinot & Seidelmann 1988] Guinot, B. and P.K. Seidelmann. "Time scales: their history, definition and interpretation." *Astronmy & Astrophysics*, 194 (1988), pp. 304–308.

[Hawking 1988] Hawking, S. "A Brief History of Time." New York: Bantam Books, 1988.

[Hood 1969] Hood, P. "How Time is Measured." Oxford, England: Oxford University Press, 1969.

[Howse 1980] Howse, D. "Greenwich Time and the discovery of the longitude." Oxford: Oxford University Press, 1980.

[Jesperson & Fitz–Randolph 1979] Jesperson, J. and J. Fitz–Randolph. "Time & Clocks for the Space Age." New York: Atheneum, 1979.

[Kirkaldy 1977] Kirkaldy, J.F. "Geological Time." Edinburgh, UK: Oliver and Boyd, 1977.

[Kudlek & Mickler 1971] Kudlek, M. and E. Mickler. "Solar and Lunar Eclipses of the Ancient Near East from 3000 B.C. to 0 with Maps." Neukirchen–Vluyn: Verlag Butzon & Bercker Kevelaer, 1971.

[McNally 1974] McNally, D. "Positional Astronomy." Bristol: Muller Educational, 1974.

[Melton 1990] Melton, J. (ed.) "Solicitation of Comments: Database Language SQL2." American National Standards Institute, Washington, DC, 1990.

[Navathe & Ahmed 1987] Navathe, S. B. and R. Ahmed. "TSQL-A Language Interface for History Databases," in *Proceedings of the Conference on Temporal Aspects in Information Systems.* AFCET. France: May 1987, pp. 113–128.

[O'Neil 1975] O'Neil, W.M. "Time and the Calendars." Sydney, Australia: Sydney University Press, 1975.

[USNO 1992] Observatory, U.S. Naval "Time Service Announcement." Series 14. Washington, D.C.. Feb. 1992.

[Pavise 1982] Pavise, F. "The Book of Calendars." New York: Facts on File, 1982.

[Petley 1991] Petley, B.W. "Time and Frequency in Fundamental Metrology." *Proceedings of the IEEE*, 79, No. 9, July 1991, pp. 1070–1077.

References

[Allen & Hayes 1985] Allen, J.F. and P.J. Hayes. "A Common-Sense Theory of Time," in *Proceedings of the International Joint Conference on Artificial Intelligence*. Los Angeles, CA: Aug. 1985, pp. 528–531.

[Anderson 1982] Anderson, T.L. "Modeling Time at the Conceptual Level," in *Proceedings of the International Conference on Databases: Improving Usability and Responsiveness*. Ed. P. Scheuermann. Jerusalem, Israel: Academic Press, June 1982, pp. 273–297.

[Ariav 1986] Ariav, G. "A Temporally Oriented Data Model." *ACM Transactions on Database Systems*, 11, No. 4, Dec. 1986, pp. 499–527.

[Aveni 1989] Aveni, A. F. "Empires of Time: Calendars, Clocks, and Cultures." New York: Basic Books, Inc., 1989.

[Bell & Bell 1963] Bell, T. and C. Bell. "The Riddle of Time." New York: The Viking Press, 1963.

[Clifford & Tansel 1985] Clifford, J. and A.U. Tansel. "On an Algebra for Historical Relational Databases: Two Views," in *Proceedings of ACM SIGMOD International Conference on Management of Data*. Ed. S. Navathe. Association for Computing Machinery. Austin, TX: May 1985, pp. 247–265.

[Clifford & Rao 1987] Clifford, J. and A. Rao. "A Simple, General Structure for Temporal Domains," in *Proceedings of the Conference on Temporal Aspects in Information Systems*. AFCET. France: May 1987, pp. 23–30.

[Das 1990] Das, T. K. "The Time Dimension: An Interdisciplinary Guide." New York: Praeger Books, 1990.

[Date & White 1990] Date, C. J. and C. J. White. "A Guide to DB2." Reading, MA: Addison-Wesley, 1990. Vol. 1, 3rd edition.

[Doane 1966] Doane, D. "Time Changes in the U.S.A.." Tempe, AZ: American Federation of Astrologers, Inc., 1966.

[Dyreson & Snodgrass 1992] Dyreson, C. E. and R. T. Snodgrass. "Historical Indeterminacy." Technical Report TR 91-30a. Computer Science Department, University of Arizona. Revised Feb. 1992.

[Fitzpatrick 1990] Fitzpatrick, G. "International Time Tables." London: The Scarecrow Press, 1990.

[Fraser 1987] Fraser, J. "Time the Familiar Stranger." Redmond, WA: Tempus Books, 1987.

For the physical realization, we proposed the high, low, and extended resolution time-stamp formats. Our formats blend OS-style and database-style time-stamp features. Like OS-style time-stamps, our formats are a simple count of chronons from the origin. They support efficient comparison and arithmetic operations, but are less efficient at input and output. Like database-style time-stamps, our formats have both extensive range and fine granularity. Some of our formats also have a limited internal structure (e.g., the high resolution time-stamp maintains separate fields for milliseconds and microseconds).

Our proposed representations met certain design goals. First, the range of our representation had to accommodate all of time. The low and extended resolution formats satisfy this goal. Second, we placed a 64 bit size limit on our basic time-stamp formats. The formats we expect to be used most often are the standard high resolution format and the chunked with uniform distribution high resolution format; both of these formats are 64 bits in size. Third, we did not want to unduly restrict the granularity of time-stamps that are close to "now". The standard high resolution format has a granularity of one microsecond while the extended resolution format has a granularity of a nanosecond (with accommodation for even more precision via extra words; just a single extra word narrows the granularity to 0.0001 femtoseconds; surely 10^{-18} seconds should be adequate for quite awhile!). Fourth, we wanted to be able to represent historical indeterminacy at minimal cost. This lead to the creation of the chunking scheme. Fifth, we have shown how the issues of granularity and historical indeterminacy can be unified, by utilizing chronons to model fixed-length periods of indeterminacy. Sixth, time and space efficiency were important concerns. Our lowest space efficiency, that of the high resolution format, is 93%. In contrast, the highest space efficiency of a database-style format, DB2's `timestamp` format, is only 73%. The timings given in Section 6 show that our formats are very efficient at frequent time-stamp operations. Finally, our time-stamps are calendar "independent". Unlike both DB2 and SQL2 time-stamps, our time-stamps can be easily and efficiently used to support multiple calendars [Soo et al. 1992].

In future, we will need to more thoroughly explore the issue of compact interval time-stamps, both determinate and historically indeterminate. Two type values, 1101 and 1110, have been reserved to accommodate the addition of new formats, if they are deemed appropriate.

Acknowledgements

This work was supported in part by NSF grant ISI-8902707 and in part by IBM contract #1124. We would also like to thank Alan Condit, who prototyped the basic event time-stamp formats, and Michael Soo, Nick Kline, Christian S. Jensen, Thomas W. Snodgrass, and Suchen Hsu, for their valuable comments on previous drafts of the paper.

probability distribution, we optimized representation of this distribution. It is represented in the type of the time-stamp, and thus requires slightly less than one bit to encode.

Even with a uniform probability distribution, we still need to encode two times in a single time-stamp. On the surface, it would appear that we would require more than 64 bits for an indeterminate event since a single time-stamp requires 64 bits. But we expect that arbitrary periods of indeterminacy will be rare. What will be more common are periods such as N hours, N days, or N years. We also expect that precise knowledge of starting and terminating times for large periods of indeterminacy will be rare. For example, it would be very uncommon for a user to know that an event occurred sometime between 6:23:43.003 A.M. July 23, 1985 and 3:00:57.23409 August 15, 1990. It is more likely that the user knows it occurred sometime between July 1985 and August 1990.

The chunking scheme was developed to meet the expectation that regular periods of indeterminacy will the norm. Chunking is a very efficient method of encoding a terminating time; the encoding only occupies eleven bits. But the space efficiency comes at the expense of some run-time computation since the terminating time must be computed on the fly. To ease the burden of the run-time computation, we plan to precompute every combination of chunk and chunk size and to pin this table in main memory (the table is 48K bytes in size). The computation of the terminating time will cost one table lookup (to get the duration specified by the chunk and chunk size) and one time-stamp addition to add the duration to the starting time. Another cost is that many periods of indeterminacy cannot be represented using the chunking scheme. For example, we cannot represent a period of 3 hours and 2 minutes using 1 minute chunk sizes (the maximum period in this case is 2 hours and 8 minutes).

8 Summary and Future Work

A time-stamp has a temporal interpretation and a physical realization. The temporal interpretation given in this paper is the base-line clock. The base-line clock tells us which time a pattern of bits in a time-stamp represents. We presented some twenty clocks which could have provided this interpretation, but we chose only a select few for the base-line clock, and we motivated these choices. Mappings exist between the base-line clock and other clocks.

The base-line clock spans all of time. The base-line clock ticks in ephemeris seconds during prehistorical times (from the "Big Bang" to B.C. 9,000). Because historical dates are more likely to be given in terms of the mean solar clock than the ephemeris clock, the base-line clock switches to the mean solar clock at the dawn of civilization. Within the last half-century, the need for precise time measurements spawned a host of clocks. From this pack, Universal Coordinated Time (UTC), which describes the relationship between International Atomic Time and the mean solar clock, has emerged as the new standard. The base-line clock follows UTC from the inception of its current definition (January 1, 1972 A.D.) to when it is no longer defined (July 1, 1992 A.D.). Beyond that time, the base-line clock switches to an idealized "atomic time", Terrestrial Dynamic Time (TDT).

Chunks and chunk size are either recorded explicitly or implicitly in the time-stamp format. Every time-stamp that does not explicitly encode a particular terminating time uses a chunking scheme to compute the terminating time. For example, the standard high resolution format has an implicit terminating time. Furthermore it has an implicit chunking scheme. It is assumed to have a single chunk with a chunk size of one microsecond.

7.2 Indeterminate Intervals and Spans

Indeterminate intervals are intervals which have indeterminate delimiting events. These formats are identified by the type fields in the underlying events. The semantics of operations on indeterminate intervals is given elsewhere [Dyreson & Snodgrass 1992]. Like determinate intervals, indeterminate intervals are represented using two event time-stamps. Because there are so many indeterminate event time-stamps, there are 153 different indeterminate interval formats (plus sixteen determinate formats). These formats are identified by the type fields in the underlying events. Some of these formats are larger than their determinate brethren; the largest indeterminate interval time-stamp is a whopping 448 bits (14 words). We anticipate that the 128 bit (4 word) indeterminate interval formats will be the most common; this is still smaller by 32 bits than SQL2's limited `datetime` event type without fractional seconds. As with determinate intervals, we are considering special formats for common indeterminate formats.

Many common spans are of indeterminate duration. For example the typical response as to when the garbage will be carried out is "in about five minutes," which actually represents a fixed span of between five minutes and several days. Fixed indeterminate spans can use the same representational formats as indeterminate events, but we have yet to define the semantics of operations using fixed indeterminate spans. Likewise, we have yet to define the semantics of operations using variable indeterminate spans. We expect that such spans will use the same (or a similar) representational format as variable determinate spans.

7.3 Design Decisions

The three basic parts of a time-stamp format are exactly what is needed to represent indeterminate events. Each of our event time-stamps represents a single indeterminate event. No other operating system or database time-stamp format has this feature.

Although it is essential, support for indeterminate events greatly compounds the complexity of the representation, because an event is no longer just a single time, rather it is two times and a probability distribution. In the worst case, the probability distribution information alone adds an extra word to the representation. The representation of a probability distribution has three parts, the name of a *normalized distribution* (including identifying parameters), a *left offset*, and a *right offset*, occupying somewhat less than 32 bits in toto [Dyreson & Snodgrass 1992]. The probability distribution of events that have the same normalized distribution are distinguished by how much of the distribution has been removed through the machinations of various operators. The left and right offsets are the percentage that has been removed from the "early" and the "late" portion of the period of indeterminacy. Since we expect that most indeterminate events will use the uniform

Chunk Size		*Number of Seconds*	*High*	*Low*	*Extended*
10	microseconds	10×10^{-6}	1	—	1
250	microseconds	250×10^{-6}	2	—	2
10	milliseconds	10×10^{-3}	3	—	3
250	milliseconds	250×10^{-3}	4	—	4
5	seconds	5	5	1	5
30	seconds	30	6	2	—
1	minute	60	7	3	6
5	minutes	300	8	4	—
1	hour	3,600	9	5	7
6	hours	21,600	10	6	—
1	day	86,400	11	7	8
7	days	604,800	12	8	9
30	days	2,592,000	13	9	10
1	year	31,536,000	14	10	11
25	years	788,400,000	15	11	—
1	century	3,153,000,000	16	12	12
1	millennium	3,153,000,000,000	—	13	13
100	millennium	315,300,000,000,000	—	14	14
10	million years	31,530,000,000,000,000	—	15	15
250	million years	788,250,000,000,000,000	—	16	16

Table 8: The default chunk sizes

distribution, which is assumed to be the uniform probability distribution. It is in this sense that every time-stamp stored in the database is indeterminate, and that granularity and historical indeterminacy are two sides of the same coin.

The terminating time is also sometimes encoded implicitly. If it is encoded implicitly then it it composed of a *chunk size* and a number of *chunks*. A chunk is a fixed duration. The length of the duration is given by the chunk size. The terminating time is computed by adding the number of chunks, each of size *chunk size*, to the starting time. For example, to represent a period of 7 hours using chunks, the time-stamp would record that there are seven chunks with a chunk size of an hour. The chunk sizes that can be used with the high resolution formats are different from those that can be used with the low resolution formats since low resolution cannot use chunks smaller than one second. One of the duties of the database implementor (whose other duties are described elsewhere [Soo & Snodgrass 1992A]) is to build three chunk size tables, one for each kind of resolution.

Table 8 lists the sixteen default chunk sizes we provide for each resolution, but the database implementor can override the default choices. We chose these chunk sizes for the default chunk sizes because we expect them to be the commonly used chunk sizes. The entries marked as "—" indicate that the chunk size is unavailable for the indicated resolution. Reading the table from top to bottom, the chunk sizes increase from smaller to larger. The jump from one chunk size to the next is by a factor of between 10 and 100. For instance, the 250 millisecond chunk size is 25 times bigger than the 10 millisecond chunk size. If the jump factor were smaller, little advantage would be gained by having different chunk sizes. If the jump factor were larger, there would be gaps in there range of available chunk sizes since there can be at most 128 chunks each of chunk size.

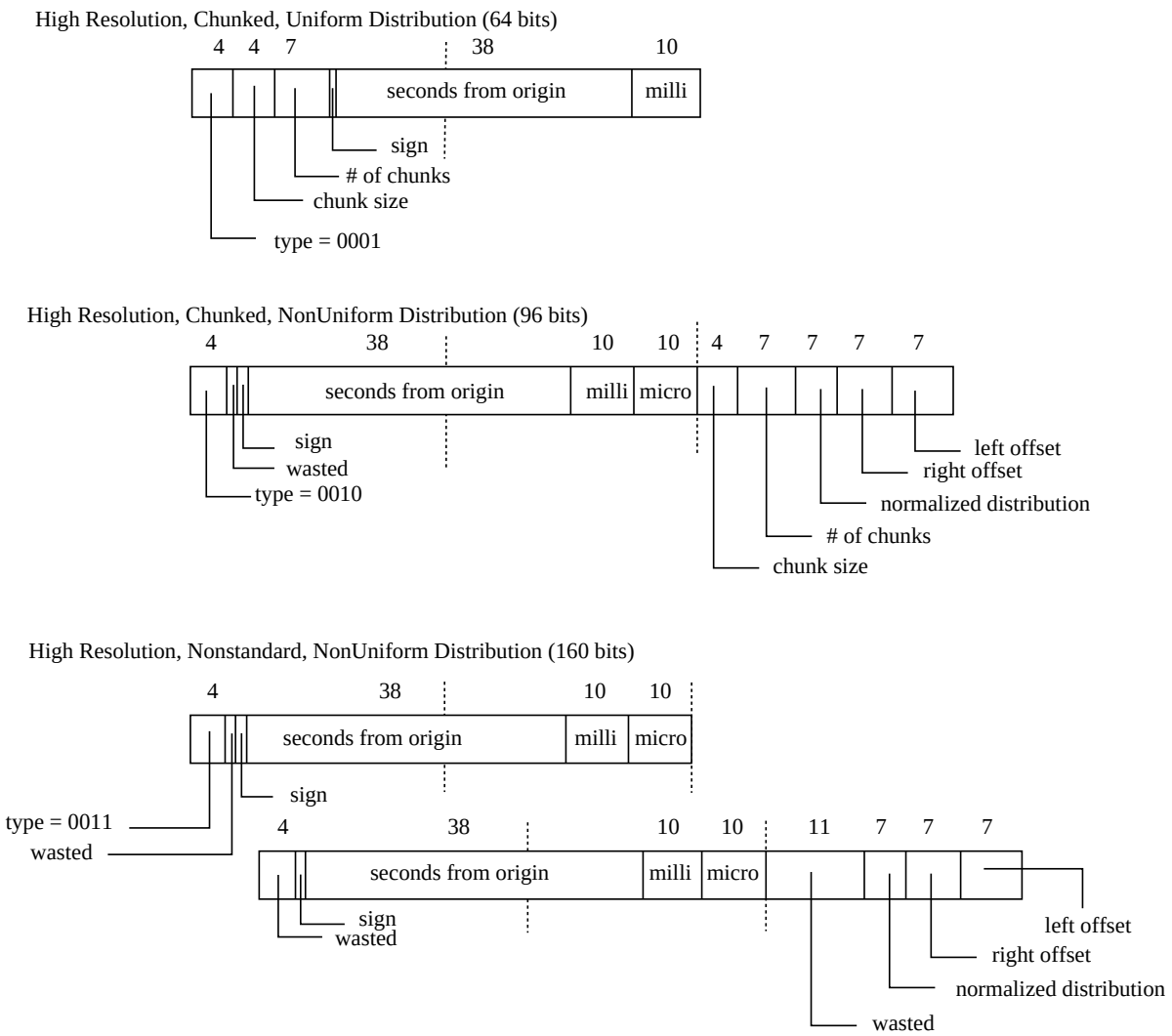

Figure 9: The indeterminate high resolution formats

FORMAT	*Starting Time*	*Terminating Time*	*Period of Indeterminacy*	*Probability Distribution*
Standard	explicit	implicit	implicit	implicit
Chunked, Uniform	explicit	implicit	explicit	implicit
Chunked, NonUniform	explicit	implicit	explicit	explicit
NonStandard	explicit	explicit	implicit	explicit

Table 7: Encoding of the high resolution formats

Supporting indeterminacy is important because our basic model of time incorporates indeterminacy at a fundamental level. We view granularity as an indeterminacy issue. For example, Unix time-stamps have a granularity of a second. In our model of time, a Unix time-stamp would have a resolution of a second with an implicit indeterminacy of a single second and a uniform distribution. Our standard low resolution format has exactly the same semantics, but with a much wider range. The advantage of viewing granularity as indeterminacy is that changing the granularity changes the amount of indeterminacy in the stored database but does not change the time-stamp or query semantics. Alternatively, both the time-stamp and query semantics could be modified by translating each to the coarsest granularity [Wiederhold et al. 1991]. But we feel that it is an advantage to make answers to queries independent of the granularity of the time-stamps in the underlying database to as great an extent as possible.

When indeterminacy is supported, the concept of "now" can have both a determinate and an indeterminate representation. The determinate representation is the current clock reading when the value is stored in the database. It is used to record that an event occurred now. Now, in this sense, is merely used as a synonym for the current clock value. Another common use of "now" is to indicate that an event will occur sometime in the future, but when is unknown. For example, an employee tuple could indicate that an employee has been working with the company from 1/1/80 until now. Now, in this sense, represents a currently unknown future time when the employee will stop working for the company. An indeterminate event can capture this use of "now".

7.1 Indeterminate Events

In this paper, we focus on the representation of historical indeterminacy. The semantics of the operations performed on such values is given elsewhere [Dyreson & Snodgrass 1992]. To represent indeterminate events, we added nine formats, three of each resolution. The proposed high resolution indeterminate formats are shown in Figure 9. There are three analogous formats for low resolution (64, 96, and 160 bits in length), and three for extended resolution (96, 128, and 192 bits in length).

The formats appear to be very different, but they are all fundamentally alike. Each time-stamp format has four parts: a *starting time*, a *terminating time*, a *period of indeterminacy*, and a *probability distribution*. The starting and terminating times are times on the base-line clock. They anchor the time-stamp to a particular interval on the base-line clock. The *period of indeterminacy* is the duration (in terms of chronons) between the starting and terminating times. The probability distribution function ranges over the chronons in the anchored duration on the base-line clock. The value of the function at a particular chronon during the anchored duration is the probability that the event occurred during that chronon.

Each time-stamp format contains some of these four parts only implicitly. Whether these parts are represented explicitly or implicitly for each format is indicated in Table 7. For example, the standard high resolution format (shown in Figure 4) can represent a time to the precision of a microsecond. It has an explicit starting time, but an implicit terminating time. The starting time is the start of the represented microsecond and the terminating time is the end of the represented microsecond. The period of indeterminacy, one microsecond, is implicit, as is the probability

query while input and output occur at most once per tuple. Consider a sequence of database operations, S, that determines the overlap of relation R_A and relation R_B. Overlap is a very common time-stamp operation and can be implemented with four precedes comparisons. R_A contains A tuples and R_B contains B tuples. We will also assume that each time-stamp is input (using the Gregorian calendar) and that $A+B$ tuples are output (using the Gregorian calendar) during the execution of S. The time needed to evaluate S is approximately $4 \times (A \times B) \times C_p + (A+B) \times C_i + (A+B) \times C_o$ where C_p is the cost of an individual precedes operation, C_i is the cost of the input operation, and C_o is the cost of the output operation. Overlap is an $O(A \times B)$ operation because it compares every tuple in R_A with every tuple in R_B. If there are 49 tuples in R_A and in R_B then the execution of S with SQL2 time-stamps costs 18.6 milliseconds while the execution of S using high resolution time-stamps costs 18.5 milliseconds (calculated from the timings given in Table 6). Greater than 49 tuples in R_A or R_B increases the time savings of our format over that of SQL2. For example, if there are 2000 tuples in each relation then S with SQL2 time-stamps takes 30 seconds while S with high resolution time-stamps takes just 13 seconds.

In summary, the high, low, and extended resolution time-stamps are competitive. They are outperformed only by the OS format on the frequent time-stamp operations. But the superior performance of the OS format is a product of its 32 bit size, and an adequate range and granularity cannot be represented using just 32 bits. The relatively slow Gregorian calendar input and output performance of the high, low, and extended resolution formats is offset by the observation that input and output are infrequent time-stamp operations in the evaluation of a query and input and output to non-Gregorian calendars is relatively quick.

7 Historical Indeterminacy

Often a user knows only approximately when an event happened. For instance, a user may know that it happened "between 2 PM and 4 PM", "on Friday", "sometime last week", or "around the middle of the month". These are examples of *historical indeterminacy*. Information that is historically indeterminate can be characterized as "don't know when" information, or more precisely, "don't know *exactly* when" information. A *null* time-stamp can be used to represent missing, but applicable, time-stamp information; but historical indeterminacy is more general since it concerns information that is missing or partially known.

Our proposal includes support for both determinate and indeterminate events [Dyreson & Snodgrass 1992]. An event is *determinate* if it is known during which chronon it occurred. A determinate event cannot overlap two chronons. If it is unknown when an event occurred, but known that it did occur, then the event is *indeterminate*. Two pieces of information completely describe an indeterminate event: a *set of possible chronons* and a *probability distribution*. The set of possible chronons describes when the event could have occurred (we assume that the chronons in the set are contiguous). The event happened during exactly one of the chronons in the set, but we do not know during which chronon it happened. For each chronon in the set, the probability distribution gives the probability that the event occurred during that chronon. A uniform probability distribution means that every possible chronon is equally likely.

SYSTEM	*Precedes*	*Addition*	*Division*	*Gregorian*		*Generic*	
				Output	*Input*	*Output*	*Input*
OS	0.20	1.0	3.3	76	21	0.79	0.29
DB2	1.3	150	240	4.3	44	26	190
SQL2	1.9	7.5	170	1.8	1.9	19	78
High	0.81	4.6	63	86	24	3.0	3.2
Low	0.81	2.5	15	82	23	2.1	2.0
Extended	1.3	5.2	86	86	24	3.3	3.3

Table 6: Results of benchmark tests (in microseconds)

less efficient are the arithmetic operations. There are five internal fields in both the SQL2 and DB2 formats, four in the extended resolution format, three in the high resolution format, and one in both the OS and low resolution formats. The OS and low resolution formats have the lowest arithmetic operation speeds, followed by the high resolution, extended resolution, SQL2, and DB2 formats, in that order. The arithmetic operation times for the DB2 format are much higher than the others because each operation must convert the DB2 time-stamp from a packed decimal to an integer representation and back (the conversion must also be done during translation, but only once, versus three times for arithmetic). This conversion imposes a debilitating overhead.

The speed of the comparison operations is ordered by time-stamp word size. The size of the OS format is one word, the low and high resolution formats are each two words, the extended resolution and DB2 (10 bytes) are three words, and the SQL2 format is five words in size.

The translation benchmarks highlight the differences between the SQL2 format and the other formats. The SQL2 format is efficient at input and output to the Gregorian calendar, but inefficient at input and output to generic calendars. The DB2 format is efficient at output to the Gregorian calendar, but inefficient at input because the conversion to a packed decimal representation is more expensive than the conversion from a packed decimal representation. Both the SQL2 and DB2 formats store the relevant Gregorian calendar information (the year, month, day, hour, minute, and second of a Gregorian date) but must compute the generic calendar information (the number of seconds from the origin). Conversely, the other formats store the relevant generic calendar information (the number of seconds from the origin) but must compute the Gregorian calendar information. Hence, while the high, low, and extended resolution formats outperform both the DB2 and SQL2 formats on some time-stamp operations, they are outperformed by SQL2 on Gregorian calendar input by a factor of 12 and on output by a factor of 47.

To determine which format is better overall, the the disparity in the execution speeds of the various time-stamp operations must be balanced against the frequency of those operations. Comparison operations will be the most frequent kind of time-stamp operation, followed by arithmetic operations, output, and then input (in that order). Input will be performed exactly once per time-stamp.

Efficient comparison and arithmetic operations outweigh efficient input and output since comparison and arithmetic operations are likely to be executed many times during a evaluation of a

```
function GREG_TO_SECONDS(year, month, day, hour, minute, second: integer) : integer;
    const
        leap_year: array [0..399] of boolean;
        year_to_seconds: array [0..399] of integer;
        month_day_to_seconds: array [boolean, 1..12, 1..31] of integer;
        seconds_to_1970 = 62167132800;
        seconds_in_400_years = 146097 × 86400;
    var
        temp, year_residue: integer;
    begin
        temp := -seconds_to_1970; { Only if needed }
        temp := (year div 400) × seconds_in_400_years + temp;
        year_residue := year mod 400;
        return temp +
            year_to_seconds[year_residue] +
            month_day_to_seconds[leap_year[year_residue], month, day] +
            hour × 3600 + minute × 60 + second
    end; { GREG_TO_SECONDS }
```

Figure 8: The conversion from the Gregorian to the generic calendar representation

SYSTEM	*Precedes*	*Addition*	*Division*
OS	$3^h14^m7^s$ 1/23/2038	$3^h14^m7^s$ 1/23/2038	$3^h14^m12^s$ 1/23/2038
DB2	$23^h59^m59^s$ 12/31/9999	$23^h59^m59^s$ 9/31/1970	$23^h59^m59^s$ 12/31/9999
SQL2	$23^h59^m59^s$ 12/31/9999	$23^h59^m59^s$ 9/31/1970	$23^h59^m59^s$ 12/31/9999
High	$22^h36^m47^s$ 11/25/8502	$1^h8^m15^s$ 1/1/1970	$6^h9^m3^s$ 7/14/10680
Low	$22^h36^m47^s$ 11/25/8502	$6^h28^m15^s$ 2/7/2106	$7^h45^m3^s$ 6/19/109626219
Extended	$22^h36^m47^s$ 11/25/8502	$6^h28^m15^s$ 2/7/2106	$7^h45^m3^s$ 6/19/109626219

Table 5: The dates used in the benchmark tests (in Gregorian format)

to compare every word in the SQL2 format. The worst-case performance for the comparison and arithmetic operations depended on the date used in the benchmark. The exact date we used is shown in Table 5. For addition and comparison, the same date is used for both operands. For division, the indicated date is the divisee; the divisor is the scalar value $2^{32} - 1$. Every translation benchmark used the same date: 3:14:07 A.M. January 19, 2038 A.D. We chose this date because it is within the range of every time-stamp format, yet it is distant enough from the origin to serve as an exotic date for translation purposes. We repeated each benchmark ten times in succession to exploit any advantageous cache effects. The results in Table 6 show the median execution time with a 0.12 microsecond correction for extraneous test loop overhead.

As we expected, the OS format is very efficient at comparison, addition, and division. This is due, in part, to its single word size. The implementation of the benchmark operations for the OS format did not have to make use of slow 64 bit division and addition operations; quicker 32 bit operations were adequate. Also, as we anticipated, the more fields there are in a format, the

fields for the year, month, day in month, hour, minute, second, millisecond, microsecond, and nanosecond. We experimented with the C library functions `timegm()` and `gmtime()` available in Unix to perform the translations between the calendar generic representation and the Gregorian calendar representation, but discovered that these functions have poor execution speeds. Hence, we wrote our own Gregorian calendar translation functions. For certain dates, a single call to `gmtime()` takes nine milliseconds, versus 20 *micro*seconds for our implementation.

The algorithm to convert the Gregorian calendar representation to the generic representation is shown in Figure 8; the reverse conversion is similar. Our algorithm is very efficient at translating Gregorian dates between 1 A.D. to 400 A.D. We precompute the cumulative number of seconds for each year in this period, yielding a table of 400 values. We also precompute the number of seconds in each month and day from the start of an arbitrary year (a table of 744 values). The tables in toto require 12352 bytes of main memory. To translate a Gregorian date between 1 A.D. and 400 A.D., we perform one table lookup to obtain the cumulative number of seconds to the start of the given year and a second table lookup to determine the cumulative number of seconds to the start of the day within the year. We then convert the hours and minutes in the date to seconds and sum all of the translated values, yielding the number of seconds from the Midnight January 1, 1 A.D. to that date.

For Gregorian dates that are not between 1 A.D. and 400 A.D., we observe that there are a fixed number of seconds in every 400 year period from the 1 A.D. For instance, the period from 1 A.D. to 400 A.D. has $(365 \times 400 + 97) \times 86400$ seconds (there are 97 leap days every 400 years and 86400 seconds in a day) as does the period from 401 A.D. to 800 A.D. Our algorithm to determine the number of seconds from 1 A.D. to a Gregorian date first calculates the number of 400 year periods to that date and then determines how many seconds lie between the nearest earlier 400 year boundary and the date using the techniques described above. For example, to determine how many seconds lie between 1 A.D. and Midnight July 2nd, 1970 A.D., we observe that 1970 is $400 + 400 + 400 + 400 + 370$. We add the number of seconds in a 400 year period four times to the number of seconds between Midnight January 1, 1 A.D. and Midnight July 2nd, 370 A.D. We optimized the calculation of the 400 year periods to avoid a 64 bit division and multiplication (this optimization is not shown in Figure 8). We iterate over the bits in the Gregorian year, determine via a table lookup how many 400 year periods an individual bit represents, and add the appropriate number of seconds to a running total.

We implemented the benchmarks for each time-stamp format in the C programming language. To avoid penalizing any operation with a spurious function call overhead each operation was coded in-line. We compiled the benchmarks using the GNU C compiler, version 2.0, with compiler optimization fully enabled. We chose this compiler primarily because it supports a 64 bit integer type (`long long`) which we utilized to implement the division and translation benchmarks. All the benchmarks were performed in a controlled environment on a dedicated Sun-4. For the DB2 format, we wrote code to simulate the packed decimal arithmetic operations; neither C nor the SPARC architecture directly supports arithmetic on packed decimal values.

Each arithmetic and comparison benchmark performed the worst-case amount of work for an operation between ten thousand and ten million times. For example, the precedes benchmark on the SQL2 format compared 11:59 PM, December 31, 9999 with itself. This comparison does the worst-case amount of work since the comparison of a date with itself forces the precedes operation

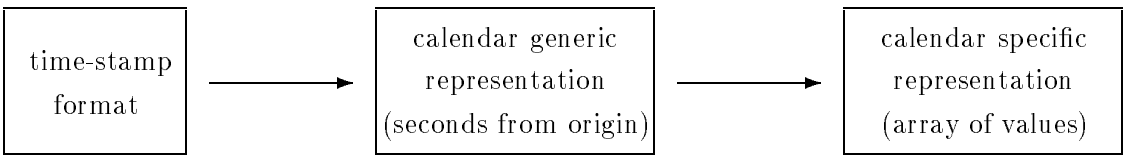

Figure 6: The two-step output process

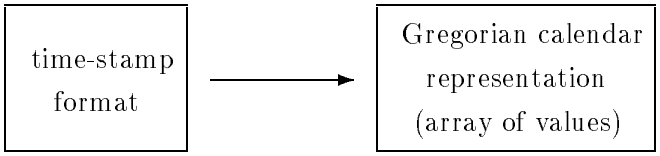

Figure 7: Output to the Gregorian calendar representation

Division also operates field by field with the necessary carries between the fields. Unlike addition, the time-stamp fields are divided from largest granularity to smallest (e.g., first seconds, then milliseconds). For example, to divide a one hour SQL2 fixed span by the scalar value 3 results in a fixed span of 20 minutes. In some cases (e.g., when dividing by a very large scalar value), the SQL2 and DB2 formats must be converted into the number of seconds represented and reconverted into years, months, days, hours, minutes, and seconds after the division. Like addition, division also checks to ensure that the resulting span is valid.

We implemented two arithmetic operations because we expect there to be significant variation in the execution speed of (binary) arithmetic operations. The two operations we implemented, addition and division, are at opposite ends of the execution speed spectrum; addition is one of the quickest arithmetic operations while division is one of the slowest.

We view output as a two-step process shown in Figure 6 [Soo & Snodgrass 1992A]. The first step converts a time-stamp to a calendar generic representation. The second step converts the calendar generic representation to a calendar specific representation. The calendar specific representation is an array of pertinent values. For example, the Gregorian calendar representation has array values for the year, month, day, hour, minute, second, millisecond, microsecond, and nanosecond. Input is the reverse process. The two-step process simplifies the task of input from and output to multiple calendars by utilizing a calendric *lingua franca*, the calendar generic representation.

We include benchmarks for translation to both the calendar generic representation and the Gregorian calendar representation because the DB2 and SQL2 formats are clearly designed to favor the Gregorian calendar. The translation to and from the Gregorian calendar representation skips the calendar generic representation as shown in Figure 7. Our benchmark for the calendar generic translation does not include the second translation step, from the calendar generic representation to the calendar specific representation, since the cost of this step is the same for every time-stamp.

The calendar generic representation is a 64 bit unsigned integer representing the number of seconds from the origin. The Gregorian calendar representation has separate unsigned integer

Translation to and from a generic calendar representation — These translation operations are used to quantify input and output to *multiple* calendar formats (e.g., Gregorian, Julian, Mayan, Islamic, etc.). The generic calendar representation is the number of seconds offset from a favorably placed origin. The origin for the SQL2 and DB2 time-stamps is Midnight January 1, 1 A.D. while the orgin for the rest is Midnight January 1, 1970 A.D.

We believe that these operations serve as an adequate benchmark. The benchmark operations were culled from a more complete set of time-stamp operations [Soo et al. 1992]. The other time-stamp operations in this set are either similar to one of the benchmark operations or a combination thereof. For example, the subtraction of a fixed span from an event is similar to the addition of an event with a fixed span, while the overlap comparison operation, which determines if two intervals intersect, is a combination of four precedes comparisons. Note that none of the time-stamp formats (except ours) have an explicit fixed span format. To remedy this we used the same format for both events and fixed spans.

There are important details connected with the implementation of each operation. The precedes operation is virtually the same for every format. Each format orders its fields by granularity, a field with a larger granularity is to the left of a field with a smaller granularity, This property allows our implementation of the precedes operation to ignore fields altogether and optimally compare the formats word by word from left to right.

The addition operation for the OS and low resolution time-stamps is a straightforward addition of seconds to seconds. Addition for the other formats requires adding each relevant field separately (e.g., microseconds to microseconds, milliseconds to milliseconds) and performing the necessary carries between the fields. The time-stamp fields are added from smallest granularity to largest (e.g., first milliseconds, then seconds).

Not every addition of an event and a fixed span results in a valid event. For example, adding two OS time-stamps could result in an overflow. Hence, we further constrain addition to indicate whether or not the operation produces a valid result. To satisfy this constraint, the SQL2 and DB2 formats must be checked to ensure that each field conforms to Gregorian calendar standards. The validity test makes sure that the number of days in a month does not exceed the allowed number, and takes into account monthly differences in the number of days in a month (including leap years). Any implementation which adheres to the SQL2 or DB2 standard must perform this test. Neither standard mentions "leap seconds" so we did not implement a check for leap seconds, although leap seconds are sometimes used with Gregorian dates after 1972. A leap second test would involve determining whether the number of seconds corresponds to a valid number of seconds for a given year, month, day, hour, and minute. We felt that the expense of doing a leap second check on the DB2 and SQL2 formats would make meaningless any benchmarking.

The OS, high, low, and extended resolution time-stamps perform a different kind of validity check for addition. The addition operation for these formats ensures that no field overflows during computation (e.g., no more than $2^{38} - 1$ seconds can be put into the seconds field of the high resolution time-stamp). These formats do not need a validity test for days in a month or for leap seconds since none of these formats records either minutes, hours, days, months, or years separately.

SYSTEM	*Size (bytes)*	*Range*	*Granularity*	*Bytes Needed*	*Space Efficiency*
High Resolution	8	$\approx$ 17400 years	microsecond	7.5	93%
Low Resolution	8	$\approx$ 36 billion years	second	7.6	95%
Extended Resolution	12	$\approx$ 36 billion years	nanosecond	11.3	94%

Table 4: A comparison of the proposed event formats

Output is a more complex operation. Each time-stamp must be unpacked into relevant fields during output. For example, assume that a low resolution time-stamp is to be printed using a Gregorian calendar format. In this case, the year, month, day, hour, and minute would have to be extracted from the seconds field. Calculating these values involves numerous expensive divisions. We could have decided to store each relevant field separately in the actual time-stamp (as is commonly done in database-style time-stamps) and thus avoid the expense of extracting these fields. This was not done for two reasons. First, such a scheme would complicate arithmetic operations, and to a lesser extent comparison operations, which we chose to optimize over output. Second, deciding which fields are relevant is very calendar specific. If a separate field is created inside the time-stamp to record minutes, as was done in DB2, then an extra level of translation must be provided to turn minutes back into seconds for the traditional Chinese calendar, which does not have minutes.

The standard event formats are summarized in Table 4, which may be compared with Table 2 on page 12. The space efficiency of the low resolution format is surprisingly only 93% rather than 100% because we count the type field (4 bits) as wasted space. But, even with this penalty, we have the highest space utilization of any database time-stamp, and get within 7% of the optimal space utilization of the generic OS time-stamp.

6 Execution Efficiency

To obtain quantitative time measures for the DB2 `timestamp`, SQL2 `datetime`, generic Operating System (OS), high resolution, low resolution, and extended resolution time-stamps, we benchmarked each time-stamp on the seven operations listed below.

Precedes comparison — This is a binary operation that determines if one event precedes another. The result of the comparison is a boolean value that indicates if the first operand is earlier than the second operand.

Addition — This operation displaces an event by the amount of time represented by a fixed span.

Division by a scalar value — The division operation divides the amount of time represented by a fixed span by a scalar value in the range of 1 to 2^{31}.

Translation to and from a Gregorian calendar representation — These translation operations are specific to the Gregorian calendar.

Date	*High*	*Low*	*Extended*
Midnight 1/1/1970	0000000000000000	4000000000000000	800000000000000000000000
12:04:29 AM 1/1/1970	0100000010D00000	410000000000010D	81000000000010D00000000
10:51:41 PM 12/31/1969	0000000010300000	4000000000000103	80000000000010300000000
Midnight 1/1/9000 B.C.	—	4000005098092C00	8000005098092C0000000000
Midnight 1/1/1972	01003C2670000000	4100000003C26700	8100000003C2670000000000
Midnight 7/1/1992	0102962561A00000	0100000002962561A	810000002962561A00000000
12:00:00.001 AM 12/4/1961	0000F2FD78000800	400000000F2FD780	800000000F2FD78000800000

Table 3: Some time values, in hexadecimal

time-stamp.) The origin need only be sometime within several thousand years of "now" so that the range of the high resolution formats overlaps the historical past and the present. We chose an origin close to now because this results in small time-stamps for times close to now. Since we fixed the origin to a relatively recent date, we need a sign bit to represent dates in the past. A sign bit is also useful because exactly the same formats are used to represent fixed spans (fixed spans can be either positive or negative values). Fixed spans were described in Section 5.3.

The operations on temporal objects available at the logical level are presented elsewhere [Soo & Snodgrass 1992A]. The physical level support that underpins these operations can be divided into four broad categories: comparison operations, arithmetic operations, output, and input. Of these categories, we anticipate that comparison operations will be the most frequent, arithmetic operations will be less frequent, output will be less frequent still, and input will occur exactly once per time-stamp. We also anticipate that the performance of output will not be critical, since the generated string will need to be printed eventually, and printing is a very slow operation compared with instruction execution.

Comparison operations compare two time-stamps using the standard comparison predicates. In our design of the formats we kept the goal of efficient comparison operations firmly in sight. The most obvious manifestation of this design decision is that the high resolution format has separate second, millisecond, and microsecond fields even though this increases the complexity of arithmetic operations. In a comparison operation with a low resolution format, only the millisecond and microsecond fields need to be masked out. Alternatively, the high resolution format could have been (and initially was) a simple count of microseconds. In this scenario, a comparison to the low resolution format would have entailed extracting the number of seconds, a costly operation involving a division by one million.

Although comparison operations were optimized at the expense of arithmetic operations, arithmetic operations are by no means inefficient. Performing arithmetic operations on the time-stamps is essentially the same as doing limited mixed radix arithmetic. The operations are done separately on the seconds, milliseconds, microseconds, and nanoseconds fields with the necessary carries between the fields. Note, however, that the maximum of four sub-fields in our formats is exceeded both by SQL2's datetime format (six sub-fields), and by DB2's time-stamp format (nine sub-fields), implying that arithmetic on those other formats will be less efficient.

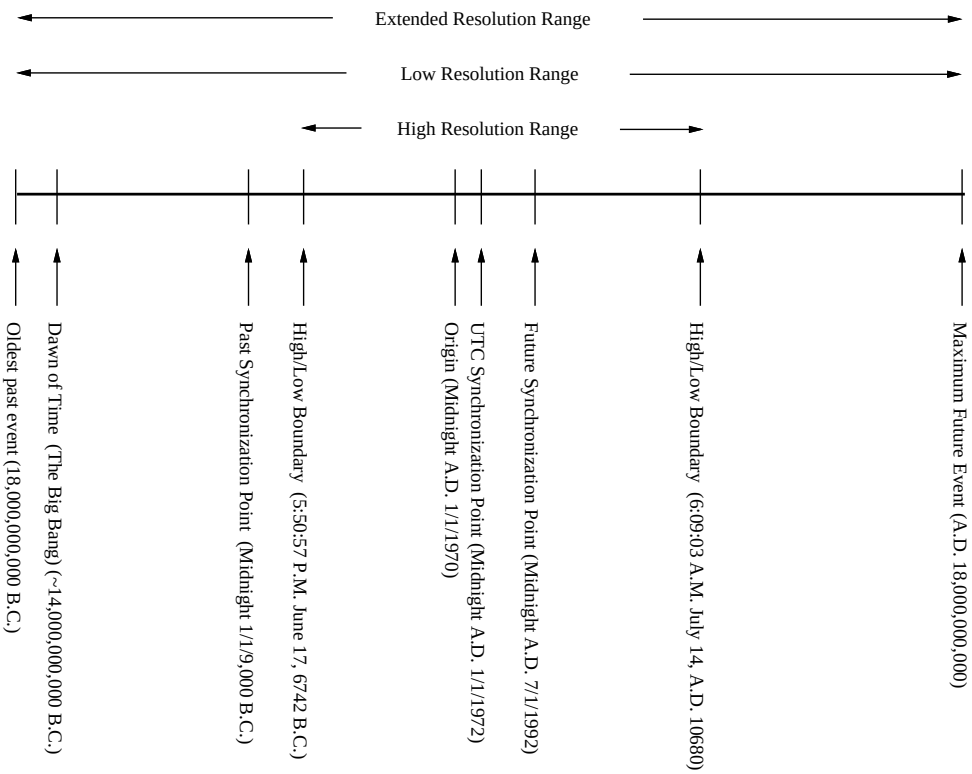

Figure 5: Ranges of three event formats (not to scale)

Using 64 bits we found that we could represent all of time (that is, a range of 34 billion years) to the granularity of a second, and less extensive ranges to granularities finer than a second. We also observed that users have a "telescoping" view of time. Events close to "now" should be capable of being represented very precisely, while events further in the past or the future can have much coarser granularities because users will not know such times precisely. Our design trades range for resolution. High resolution has limited range but extended precision while low resolution has extended range but limited precision. Extended resolution handles those uncommon cases where the user wants both an extended range and an extended precision, at the cost of an extra word of storage. The "extra flag" field in the extended resolution format signals whether the user wants even finer precision by adding a further word(s) of precision information. The range of the low resolution formats overlaps that of the high resolution formats and is coextensive with the extended resolution formats. The range of each format on the base-line clock is diagramed in Figure 5. Table 3 shows several Gregorian calendar dates, including the three synchronization points, and their corresponding time-stamp representation (in hexadecimal). One date is outside the range of the high resolution format.

We also included a sign bit in every format. The sign represents the direction from the *origin*. The origin of each time-stamp format is the same (so that high and low resolution time-stamps can be compared quickly). The "time" at the origin is represented by a time-stamp with the relevant fields all set to zero. We arbitrarily located the origin at Midnight January 1, 1970 GMT. (Those familiar with the Unix operating system will notice that this is also the origin of the Unix

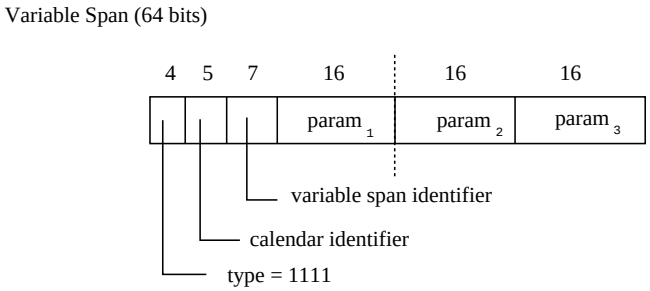

Figure 4: The variable span format

from the origin. The sign bit indicates whether the span is positive or negative rather than the direction from the origin.

Variable spans, however, use an altogether different format, shown in Figure 4. The variable span format is a 64 bit record with six fields: type, calendar identifier, variable span identifier, and three parameter fields. Using a smaller format would not result in a space savings, as variable spans must coexist with fixed spans, which are at least 64 bits in length. A variable span belongs to a specific *calendar* [Soo & Snodgrass 1992B]. Since many variable spans can belong to the same calendar, the format must also be able to identify the specific variable span (e.g., month). A variable span can have up to three *parameters*. Fields for these parameters are also included in the format (16 bits each). Finally, the variable span format has a type value that distinguishes variable from fixed spans.

For example, in the Gregorian calendar (calendar identifier `1`), the variable span "3 months and 4 days" (span identifier of `2` indicating the "month+day" span), would have parameter values: param_1 = `3`, param_2 = `4`, param_3 = `0`. The hexadecimal value of the span would be `F083000300040000`.

5.4 Design Decisions

The time-stamp representations that we developed had to simultaneously satisfy several design goals. One very important goal was that the entire range of the base-line clock, all of time, had to be represented. As shown in Table 2, most time-stamp formats have a limited range. For example, the Unix time-stamp can only represent times between January 1, 1970, 00:00 GMT and (approximately) January 1, 2091. There is a natural tradeoff between range and granularity in time-stamp development. Using the same number of bits, a time-stamp designer can make the granularity coarser to extend the range or she can limit the range to support finer granularities. Unfortunately, the Unix time-stamp suffers from both a limited range and a coarse granularity (only a second).

These observations imply that a format based on a single 32 bit word is inadequate for our purposes; there are simply not enough bits. Since we wanted to keep the time-stamp formats on 32 bit word boundaries we allocated the next word increment, or 64 bits, to our basic format.

To differentiate amongst the three time-stamp formats, each format has a type field (the type also distinguishes the indeterminate formats that are described in Section 7). The type field is stored in the high order portion rather than the low order portion because not every format is the same size. The extended resolution format is 32 bits bigger than the high or low resolution formats.

The type field is also used to distinguish special events, such as "beginning" and "forever", from other events. Beginning is represents the youngest possible time while forever represents the oldest. The hexadecimal representation for beginning is `C000000000000001` while the representation of forever is `C000000000000002`. Finally, a null time-stamp is represented as `C000000000000000`.

Each time-stamp is a distance, in base-line clock chronons, from the origin (recall that a chronon in the base-line clock is defined to be a second). The sign bit indicates the direction from the origin. The high resolution format has separate fields for milliseconds and microseconds which can be viewed as uniform-width offsets within the specified chronon. The extended resolution format has fields for milliseconds, microseconds, and nanoseconds, and can be extended to any desired accuracy. The "extra flag" field indicates whether an additional word or words of accuracy is desired. Each word adds nine decimal digits of accuracy and has its own flag field to indicate whether more words follow.

5.2 Intervals

Interval time-stamp formats are simply two event time-stamps, one for the starting interval event and one for the terminating interval event. We use this representation because all operations on intervals are actually operations on their delimiting events [Soo et al. 1992]. There are sixteen interval time-stamp formats in toto. The type fields in the delimiting event time-stamps distinguish each format.

Interval time-stamps are quite large; all intervals require at least 128 bits (16 bytes). Smaller interval time-stamps are possible. For instance, we could develop a 96 bit interval for common intervals. We are considering adding such time-stamp formats. Two type values, 1101 and 1110, are still available for this purpose.

5.3 Spans

There are two kinds of spans, fixed and variable [Soo & Snodgrass 1992A]. A *fixed span* is a count of chronons. It represents a fixed duration (in terms of chronons) on the base-line clock between two time values. A *variable span*, on the other hand, is a span whose duration varies in relation to an associated event. A common variable span is a *month*. The duration represented by a month varies depending upon whether that month is associated with an event in June (30 days) or in July (31 days), or even in February (28 or 29 days).

The fixed span formats use exactly the same layouts as the standard event formats, but are interpreted differently. The span chronon count is independent of the origin instead of a count

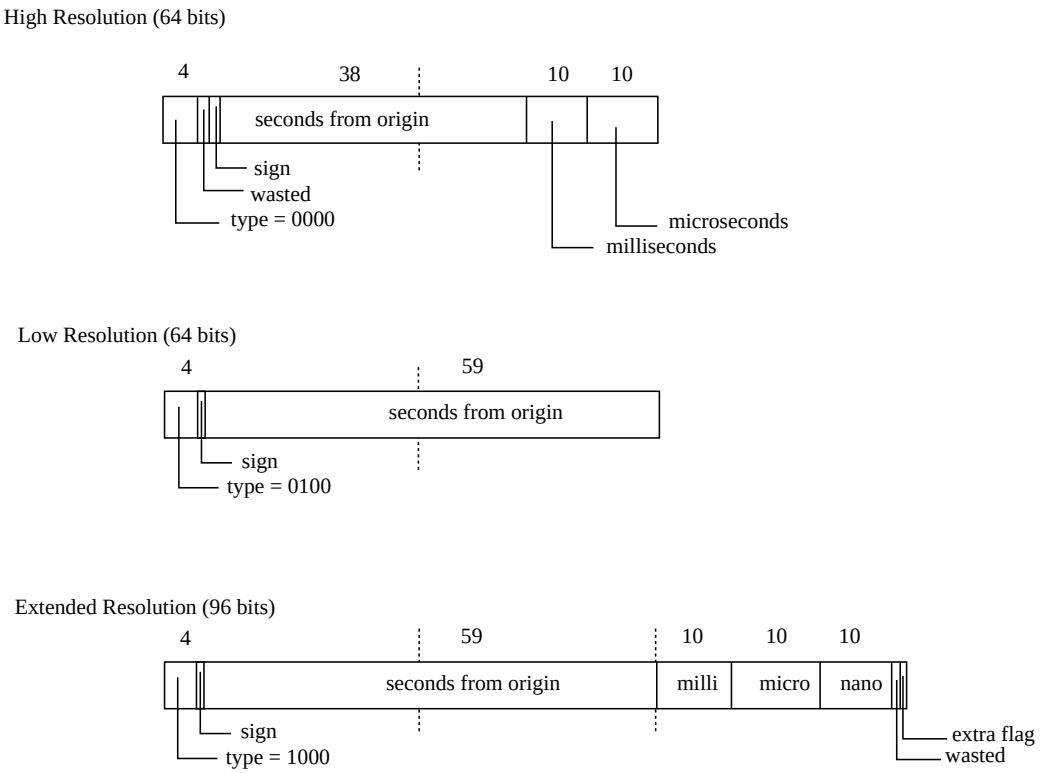

Figure 3: The standard event formats

5 Proposed Time-stamp Formats

There are three basic types of time-stamps: events, spans, and intervals [Soo & Snodgrass 1992A]. The next two sections cover event time-stamps in detail. The first section describes a simple general form behind each event time-stamp format that belies the format's complex facade. The next sections show that the span and interval time-stamps are natural extensions of the event time-stamps. The decisions we took in designing the event time-stamps are motivated in the subsequent subsection. In Section 7, we expand these formats to cover indeterminacy [Dyreson & Snodgrass 1992].

5.1 Event Time-stamp Formats

We developed three new event time-stamp formats, shown in Figure 3. The dashed lines mark word boundaries. The number above a subfield is the size of that subfield. The time-stamps have different *resolutions*: there are high, low, and extended resolution time-stamps. Resolution is a rough measure of a time-stamp precision. The low resolution format can represent times to the precision of a second. High resolution narrows the precision to a microsecond while extended resolution is even more precise; it can represent times to the precision of a nanosecond.

Space efficiency is a very important property of a time-stamp. Very roughly, it is a measure of how much of the representation is actually needed. It is computed as a percentage of the number of bits needed to represent every chronon in the temporal interpretation (DB2 and SQL2 both use the Gregorian calendar temporal interpretation) versus the number of bits devoted to the physical realization. All other considerations aside, that which is not utilized is wasted.

The minimum number of *bytes needed* to store the number of chronons dictated by a time-stamp's granularity and range is shown as a separate column. For instance, SQL2's `datetime` time-stamp uses 20 bytes, but only 4.8 bytes of space are needed to store a range of 10,000 years to the granularity of a second.

The evaluated time-stamps fall into two camps: OS-style time-stamps and database-style time-stamps. OS-style time-stamps have a limited range (about 121 years) and granularity (to the second). These limitations are dictated by the size of the time-stamp (32 bits). OS-style time-stamps are maximally space efficient. The time-stamp itself is merely a count of the number of chronons that have elapsed since the origin in the temporal interpretation. But optimal space efficiency is attained at the expense some *time efficiency*. Detailed time efficiency measures are presented in Section 6. To manipulate an OS-style time-stamp, it must sometimes be converted to an unpacked version. This translation imposes an overhead on many time-stamp operations. The cost of arithmetic and comparison operations, however, remains low because time-stamps can be manipulated as unsigned integers with no unpacking needed.

In contrast, database-style time-stamps, as exemplified by the DB2 `timestamp` format, are generally larger than OS-style time-stamps. Because database-style time-stamps are larger, they have a wider range and finer granularity than the OS-style time-stamps. But, as a group, they also have poorer space utilization. Database-style time-stamps, instead of being a simple count of chronons, have a record-like structure with separate fields for seconds, hours, days, etc. The advantage of representing these values separately is that they can be quickly accessed. Extracting the number of years from an OS-style time-stamp is more involved than performing a similar task on an DB2 `timestamp`. As we show in Section 6, the input and output performance of the database-style time-stamps is only partially satisfactory because they support but a single calendar. However, separate fields within a time-stamp makes arithmetic operations more complex. The addition of two database-style time-stamps requires adding the value at each field separately and performing carries between the fields.

Neither style of time-stamp is completely adequate for our purposes. Most of the existing time-stamps are incapable of realizing the base-line clock because of their limited range and granularity. The proposed SQL2 format has the widest range, but even with 192 bits, it is still incapable of representing all of time. Also, none of the time-stamps are able to represent *historical indeterminacy* (which we discuss in Section 7). Consequently, we developed our own time-stamp formats, incorporating features from both the OS-style and database-style time-stamps.

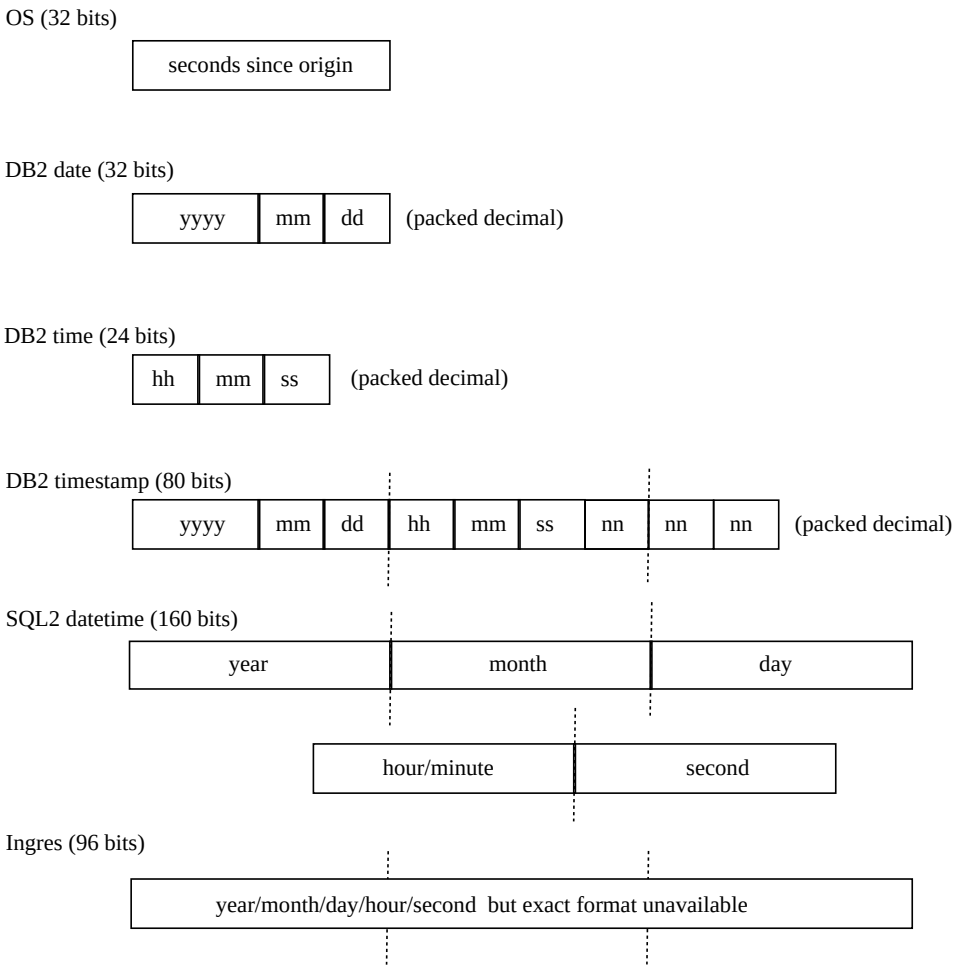

Figure 2: The bit layout of several time-stamps

SYSTEM	*Size (bytes)*	*Range*	*Granularity*	*Bytes Needed*	*Space Efficiency*
OS (several)	4	$\approx$ 136 years	second	4	100%
DB2 — `date`	4	10,000 years	day	2.9	71%
DB2 — `time`	3	24 hours	second	2.2	72%
DB2 — `timestamp`	10	10,000 years	microsecond	7.3	73%
SQL2 — `datetime`	20	10,000 years	second	4.8	24%
SQL2 — fractional `datetime`	27	10,000 years	microsecond	7.3	27%
Ingres	12	10,000 years(?)	second	4.8	40%(?)

Table 2: A comparison of some physical layouts

than one mean solar second in A.D. 1992. Another feature is that the number of chronons in a "day" is not the same among all the constituent clocks because of leap seconds. The mean solar clock counts $24 \times 60 \times 60$ chronons (if the granularity is one second) between Midnight February 23, 1925 and Midnight February 24, 1925. But, the UTC clock counts $24 \times 60 \times 60 + 1$ seconds between Midnight December 31, 1990 and Midnight January 1, 1991, because a leap second was added.

Having defined the base-line clock we now turn to describing the physical structure of a time-stamp.

4 Existing Time-stamp Representations

The bit-level representation of a temporal object has a significant impact on the performance of a temporal database system. A space inefficient representation will tax the resources of the database's storage system. In some cases, the traditional space/time tradeoff can be exploited to reduce space requirements. A common method of obtaining space efficiency is to encode the time-stamp, that is, to pack the representation into as few bits as possible. The benefits of compactness, however, need to be balanced against the extra burden placed on the time efficiency of the database system. Support for common time-stamp operations should not incur a drastic efficiency penalty as might happen if the encoded representation had to be decoded each time it was used.

With these considerations in mind, we examine several existing time-stamp formats. We scrutinize time-stamps from the MS Dos, Unix, and MacIntosh operating systems as well as the database systems DB2, Ingres, and SQL2. The bit layouts of the database time-stamps are depicted in Figure 2, drawn approximately to scale, along with a generic operating system time-stamp (all the operating system time-stamps share a common format). The dashed lines in the figure indicate word boundaries. The Ingres format is proprietary. We evaluate these time-stamp formats on five significant properties: *range*, *granularity*, *size*, *bytes needed*, and *space efficiency*. A table summarizing the candidate time-stamps is shown in Table 2. The SQL `datetime` time-stamp appears twice in the comparison, once with its optional fractional second precision field (not shown in the Figure 2) set to microseconds, and once without the optional field. In Section 6, we evaluate these formats in terms of execution efficiency.

Size is the number of bytes devoted to the representation, while *range* refers to the difference between the youngest and oldest time values that can be represented. We assume that the generic OS time-stamp does not have a sign bit, although the Unix time-stamp is defined as an `integer`. In general, a time-stamp with a narrow range is of limited usefulness since range is a restrictive property. The range could be increased at a later date, but such conversions, such as the ongoing zipcode conversion, from five to nine digits, in the United States, can be very costly.

The *granularity* of a time-stamp is the precision to which a time value can be represented. Granularity is also a restrictive feature. The choice of a particular granularity for a time-stamp eliminates the ability to represent time values with finer granularities. A granularity of a second means that time values cannot be represented more precisely than to the nearest second.

there are almost exactly the same number of days in a year today as there were in 9000 B.C (the rate of decrease is on the order of 0.0014 days per century).

The switch from ephemeris to mean solar time happens at the "past synchronization point". We define this point as the time when the two clocks have the exact same time. Thus, there is no difference between the time on the mean solar clock and the ephemeris clock at the past synchronization point. One year prior to Midnight January 1, 9000 B.C. is one ephemeris year prior to that date. One day after Midnight January 1, 9000 B.C. is one mean solar day after that date.

The mean solar clock carries the base-line clock up to Midnight January 1, 1972 after which the base-line clock follows UTC. Midnight January 1, 1972 is when UTC was synchronized with the atomic clock and the current system of leap seconds adopted. In particular, during the interval in which the base-line clock uses UTC, 26 leap seconds were added. The base-line clock runs on UTC until one second before Midnight, July 1, 1992. This is the next time at which a leap second may be added (a leap second will be added on this date according to the latest International Earth Rotation Service bulletin [USNO 1992]).

The synchronization point at July 1, 1992 is a "moving" synchronization point. When UTC becomes known beyond that time (by adding or subtracting leap seconds) the synchronization point will move. Each movement of the synchronization point engenders a new definition of the base-line clock because more of UTC is known. When the base-line clock is redefined, time-stamps in the database that represent times later than the "old" synchronization point will have to be modified to ensure consistency with the redefined base-line clock (e.g., a second will have to be added to each).

After Midnight July 1, 1992, until the "Big Crunch" or the end of our base-line clock, the base-line clock follows Terrestrial Dynamic Time (TDT) since both UTC and mean solar time are unknown and unpredictable. Also, since 1984, TDT has been favored over ephemeris time by the international standards community.

The base-line clock defines the meaning of each time-stamp bit pattern in the physical realization of a time-stamp. The chronons of the base-line clock are the chronons in its constituent clocks. We will assume that each chronon is one second in the underlying constituent clock. For example, a bit pattern corresponding to Midnight February 23, 1925 is interpreted using the mean solar clock but Midnight June 30, 1992 is interpreted using UTC. Assuming that Midnight February 23, 1925 is a date in the Gregorian calendar, it is $(10925 \times 365 + 54 + 2648) \times 24 \times 60 \times 60$ chronons after the past synchronization point (there have been 2648 leap days) while Midnight June 30, 1992 is $((20 \times 365 + 181 + 6) \times 60 \times 60) + 26$ chronons after the UTC synchronization point (there have been 6 leap days and 26 leap seconds).

The clocks that we chose for the base-line clock have two nonobvious features that are important to a database implementor. One feature is that the size of chronons in the base-line clock varies (with respect to ephemeris time) because chronons in the mean solar day clock are not of fixed size. Since the mean solar day 11,000 years ago was slightly longer (with respect to ephemeris time) than it is today (about 160 milliseconds longer, extrapolating backwards from the current rate of slow-down in the Earth's rotation), one mean solar second in 9000 B.C. was slightly longer

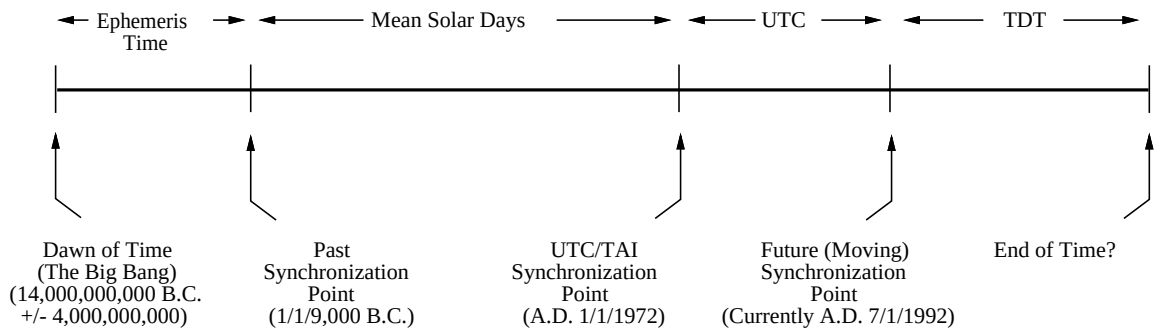

Figure 1: The base-line clock

because we expected the typical user to use tropical rather than anomalistic years.

The base-line clock is shown in Figure 1 (not to scale). It partitions the time line into a set of contiguous *periods*. Each period runs on a different clock. A *synchronization point*, where two clocks are correlated, delimits a period boundary. The synchronization points occur at Midnight on the specified date.

We have adopted the beginning and ending of time in the base-line clock from the scientific cosmology of the "Big Bang", although debates about the correctness of this cosmology continue (Hawking provides a readable introduction to this controversy [Hawking 1988]). The cosmology posits that time begins with the Big Bang, but there is much debate on when it will end. It depends on whether the universe is *open* or *closed*. If the universe is closed then time will have an end when the universe collapses back on itself in what is called the "Big Crunch". If it is open then the universe will go on forever. We will not decide on the question of an open versus a closed universe. Time in the base-line clock either does not end or ends at the "Big Crunch".

The base-line clock and its representation are independent of any calendar. We use Gregorian calendar dates in this discussion only to provide an informal indication of when the synchronization point occurred. From the Big Bang to Midnight January 1, 9000 B.C. the base-line clock runs on ephemeris time. More formally, the past synchronization point is $(10971 \times 365 + 2659) \times 24 \times 60 \times 60$ chronons (mean solar seconds) prior to the UTC/TAI synchronization point (we calculate 2659 leap days between these two Gregorian dates). This clock is preferable to the tropical year clock since ephemeris time is independent of the formation of the Earth and the Solar System. Also, we prefer using the ephemeris clock to the tropical year clock because an ephemeris year is a fixed duration, unlike the tropical year. Finally, when scientists date prehistoric events, if there is any difference between the year they use and the ephemeris year, it is inconsequential to their calculations (the difference between tropical and ephemeris years is on the order of 10^{-5} seconds per year, which is masked by the imprecision of their measurements).

For historic events, 9000 B.C. to January 1, 1972, the base-line clock follows the mean solar day clock. Historic events are usually dated with calendars. Calendar dates invariably count days and use an intercalation rule to relate the number of days to longer-term celestial clocks, e.g., the Gregorian calendar relates days to tropical years. Because the rate of Earth's rotation is slowly decreasing, there are more days in a tropical year now than there were 400 million years ago, though

Clock	*Kept by Whom*	*Users*
Apparent Solar Day	observatories	general
Mean Solar Day	observatories	general
Sidereal Day	observatories	astronomers
Tropical Year	observatories	general
Sidereal Year	observatories	astronomers
Anomalistic Year	observatories	astronomers
Ephemeris Time	observatories	scientists
UT0	observatories	astronomers
UT1	observatories	navigators
UT2	observatories	astronomers
UTC (pre 1972)	BIH	general
TAI	BIH/BIPM	general
UTC (post 1972)	BIH/IERS	general
TDT	observatories	scientists
Millisecond Pulsar	observatories	astronomers
Generic Calendar	cultural organizations	general
Tree-Ring Dating	tree ring labs	dendrochronologists
Carbon-14 Dating	labs	archaeologists
Radioactive Dating	labs	geochronologists
Sedimentation Clock	labs	stratigraphers
Magnetic Clock	labs	geochronologists

Table 1: Clocks (continued)

3.2 The Proposed Base-line Clock

The previous discussion of clocks contains several implicit points that are germane to the overall structure of the base-line clock. The first, perhaps obvious, comment is that events that occurred in the prehistoric past cannot be dated as precisely as events that occur in the present. We have an implicit "telescoping view" of time. Our dating of recent events can often be done to the millisecond while events that occurred 400 million years ago can be dated only to, at best, the nearest 100,000 years. Dating future events is also problematic. It is impossible to say how many seconds there will be between the UTC times Midnight January 1, 1990 A.D. and Midnight January 1, 2300 A.D. because we don't know how many leap seconds will be added to correct for changes in the rotational clock. We can currently guess at the number of seconds, but "leap shifts" to the current clock will invalidate our guess.

Second, many clocks are limited in their range of applicability. Tree-ring dating cannot be used to date future events nor can UTC be used to date prehistoric events.

Finally, certain clocks are of interest only to specialists. In most cases, times on these clocks can, within range limits, be approximately translated to times on clocks which are of more general interest, but this translation might be expensive and imprecise. Hence, we anticipated the needs of a typical user and favored those clocks that we believed to be of general interest for incorporation into the base-line clock. For instance, we favored the tropical year clock, based on the seasons, over the anomalistic year clock, based on the Earth's transit from perihelion to aphelion and back,

Clock	*Based On*	*Corrections Applied (if any)*
Apparent Solar Day	Sun crossing the observer's meridian	none
Mean Solar Day	Sun crossing the observer's meridian	variable orbital speed and ecliptic orbit
Sidereal Day	distant star crossing the observer's meridian	none
Tropical Year	Earth's orbit w.r.t. the equinoxes	none
Sidereal Year	Earth's orbit w.r.t. distant stars	none
Anomalistic Year	Earth's orbit w.r.t. apogee and perigee	none
Ephemeris Time	celestial observations	none
UT0	mean solar day	voting among observatories
UT1	UT0	polar wander
UT2	UT1	seasonal variations in rate of rotation
UTC (pre 1972)	UT2	within 0.1 seconds of UT2
TAI	hyperfine states of cesium atom	voting among atomic clocks
UTC (post 1972)	TAI	within 0.9 seconds of UT1
TDT	TAI and Ephemeris Time	none
Millisecond Pulsar	rotating neutron star	none
Generic Calendar	celestial day/month/year	intercalations (e.g. leap years)
Tree-Ring Dating	tree rings	none
Carbon-14 Dating	none	none
Radioactive Dating	radioactive decay	none
Sedimentation Clock	rock/fossil layers	tectonics
Magnetic Clock	Earth's magnetic polarity	none

Table 1: Clocks

Clock	*Unit of Measure*	*Range of Applicability*	*Precision*
Apparent Solar Day	day	lifetime of Earth's rotation	unknown
Mean Solar Day	day	lifetime of Earth's rotation	unknown
Sidereal Day	day	lifetime of Earth's rotation	0.005 seconds
Tropical Year	year	lifetime of Solar System	millisecond
Sidereal Year	year	lifetime of Solar System	millisecond
Anomalistic Year	year	lifetime of Solar System	millisecond
Ephemeris Time	second	lifetime of Universe	millisecond
UT0	day	1958 – present	nanosecond
UT1	second	1958 – present	nanosecond
UT2	second	1958 – present	millisecond
UTC (pre 1972)	second	1/1/1958 – 12/31/1971	second
TAI	second	1/1/1958 – present	10^{-14} seconds
UTC (post 1972)	second	1/1/1972 – present	second
TDT	second	1/1/1984 – end of time	millisecond
Millisecond Pulsar	millisecond	lifetime of neutron star	10^{-10} seconds
Generic Calendar	politically legislated	politically legislated	usually day
Tree Ring Dating	seasons	5112 B.C. – present	six months
Carbon-14 Dating	SI year	$\approx 70,000$ B.C. – A.D. 1500	centuries
Radioactive Dating	SI year	≈ 4.5 billion – 100,000 years ago	10^4–10^9 years
Sedimentation Clock	geologic epoch	4.5 billion years ago – present	10^4–10^9 years
Magnetic Clock	constant polarity intervals	≈ 170 million years ago – present	10^6–10^9 years

Table 1: Clocks (continued)

3.1.3 Dating the Prehistoric Past

Astronomical observations can be used to determine the current time and to predict the time of future events, but astronomical observations cannot be used to date past events unless the configuration of the heavens was recorded along with the event. Instead, clocks based on geological and biological processes are used to date the past.

Dendrochronology is the study of the tree-ring record. By correlating rings from many different trees, a *tree-ring clock* accurate to the season with a range of 5000–7000 years can be constructed [Silverberg 1971]. *Carbon-14 dating* determines the ratio of Carbon-14, an unstable isotope of Carbon with a half-life of between 5700 and 5770 years, to Carbon 12, a stable isotope [Goudsmit & Claiborne 1966]. Geochronologists have developed a battery of *radioactive dating* techniques based on ratios of isotopes of lead and uranium, rubidium and argon, and potassium and strontium, among others [Hood 1969]. The relative age of sedimentary strata can be deduced from their position in the layering, providing a *sedimentation clock* [Silverberg 1971]. Finally, flips in the polarity of the Earth's magnetic field provide yet another clock, the *magnetic clock* [Kirkaldy 1977].

3.1.4 Summary

A comparative summary of the clocks we have considered is shown in Table 1 on pages 7 through 8. The appendix briefly discusses a few additional clocks not included in this table. The various clocks are compared on the physical process they are based on, which corrections are applied in measuring that process, the unit of measure, their range of applicability, precision, who performs the measurements, and who uses the clock. The *unit of measure* is the common unit that the clock counts. The *precision* column is a rough estimate of the average error in the method of measurement. One of the most interesting comparative features is the *range of applicability*. Since the clocks are based on physical processes, these processes may, at one time or another, be nonexistent. For instance, 4.5 billion years ago, clocks that measure the Earth's rate of rotation did not exist since the Earth did not exist. The universe is thought to be about 14 billion years old (give or take 4 billion years), so there is a long period of time when rotational clocks simply are not applicable.

The other columns are informative rather than comparative. The first column indicates the physical process upon which each clock is ultimately based. Since some clocks are based on the same physical process, they differ not in the process but in the way that process is measured. This is shown in the *corrections applied* column. Some of the clocks discussed above are of interest only to specialists as indicated in the final column. For instance, Carbon-14 dating is used extensively by anthropologists, historians, and archaeologists, the anomalistic and sidereal years are used exclusively by astronomers to chart the movement of the heavens, and geochronologists use the magnetic and sedimentation clocks to glean knowledge about past events. In day-to-day life, the general public never encounters these clocks, except perhaps in a visit to a museum or an observatory. However, the general public is in some sense regulated with UTC, as the alternating current in their homes and offices is in synchronization with that clock.

atomic time (TAI) is the mean time of several, globally distributed atomic clocks.

Because of extraordinary accuracy of TAI, TAI was incorporated into the definition of UTC (Universal Coordinated Time) in 1972. UTC was synchronized to TAI + 10 seconds and is kept an integral number of seconds offset from TAI. Since TAI runs increasingly faster than the rotational clock, leap seconds must be sporadically added to UTC to keep UTC synchronized with the rotational clock. The number of seconds to offset is computed by deducing the rate of the Earth's rotation from astronomical observations. Preference is given to July 1 or January 1 to add (or subtract) leap seconds if needed.

In 1984, *Dynamic Time* (DT) replaced ephemeris time as the fundamental invariable time scale. Dynamic time is a relationship between TAI and ephemeris time and can be thought of as an "idealized atomic time" [Guinot & Seidelmann 1988]. The dynamic time on Earth, known as *Terrestrial Dynamic Time* (TDT), is for all practical purposes TAI + 32.184 seconds (in 1984, the difference between TAI and ET was determined to be 32.184 seconds). TDT does not suffer the computational lag inherent in ET; it can be computed immediately from TAI.

A clock that rivals the stability of the atomic clocks are the *millisecond pulsars*. A millisecond pulsar is a rapidly rotating neutron star that emits broad-band radio noise in periodic sequences. Recent studies have suggested that millisecond pulsars are extremely reliable and can be used as a diagnostic tool to check the long-term stability of individual atomic clocks [Taylor 1991].

3.1.2 Year Clocks

Clocks based on the Earth's rotation count off relatively short periods of time, but clocks that measure longer periods of time are also important. The long duration that is of interest is commonly called a *year*. Like day clocks, most year clocks are based on astronomical processes.

There are three primary clocks based on the Earth's orbit around the Sun. The time needed for the Earth to travel from aphelion to perihelion back to aphelion in its slightly elliptical orbit is known as an *anomalistic year*. A *tropical year* is the time between successive passages of the Sun through the vernal equinox (an equinox is the day when the duration of sunlight and darkness are the same). A *sidereal year* is one rotation of the Earth around the Sun with respect to the distant stars. The celestial years differ slightly in length [Schultz 1986].

Calendar years are typically related to, but independent of, celestial years. For example, the Gregorian calendar keeps in synchronization with the tropical year by counting mean solar days. This is a difficult task since there are not currently an integral number of solar days in the tropical year and the length of the solar day and of the tropical year are both changing. The *intercalation* or adjustment used by the Gregorian calendar to keep the count of solar days in step with the tropical year is to add a leap day every few years. Even with the current leap day system, the Gregorian year is approximately 25 seconds faster than the tropical year [Fraser 1987].

3.1 Previously Defined Clocks

Traditionally, clocks have measured celestial processes, such as the rhythm of day and night, the change of seasons, the phases of the Moon, and the patterns of stars at night. But nonastronomical processes, such as tree-rings, radioactive decay, and transitions between excitation states of atoms, can also make perfectly good clocks. The following sections develop a catalogue of existing clocks, each of which is a candidate for the base-line clock. One reason for presenting this brief review is to collect, from quite disparate sources, descriptions of most of the commonly used clocks. A second reason is to demonstrate that different clocks are used for different purposes, and that there is no "best clock".

3.1.1 Day Clocks

A day is an important biological and cultural unit. It is such a familiar unit that it may be surprising to learn that there are actually several different kinds of day. Each, however, is ultimately related to the rotation of the Earth.

An *apparent solar day* is one rotation of the Earth with respect to the Sun. The length of this day fluctuates by as much as 15 minutes during the course of a year, and continues to lengthen slightly as the years progress. The daily fluctuations in the length of the apparent solar day can be "smoothed", resulting in the *mean solar day* (by measuring an imaginary Sun called the "mean Sun") or a *sidereal day* (by measuring the rotation of the Earth with respect to a distant star). Unfortunately, a mean solar day also fluctuates in duration, due to various factors that affect the rate of the Earth's rotation. Correcting for combinations of these factors results in a family of *universal times*: UT0, UT1, UT2, and UTC [Howse 1980].

The family of universal times are ultimately related to the rotation of the Earth. But since the rate of rotation varies, scientists who need to make accurate time calculations need a unit of time that is independent of the Earth's rotation. In 1955, one such standard, called *Ephemeris Time* (ET) was adopted by the scientific community. Prior to 1955, the International System of Units (SI) defined the second in terms of the mean solar day. Ephemeris time is based on the tropical year (to be defined later) 1900 A.D.; an ephemeris second is defined to be $1/31,556,925.9747$ of that year. While this may seem an odd definition, the ephemeris second is actually the average value of a second calculated from astronomical observations over the eighteenth and nineteenth centuries [Quinn 1991]. Ephemeris time is a computed time. There is no mechanical device that counts off ephemeris seconds. Variations in the length of the day in terms of ephemeris seconds can be computed ex post facto from observed versus predicted positions of celestial bodies. But this computation is commonly done years after the observations are made. Ephemeris time is impractical for everyday use since it suffers from this computational lag [Howse 1980].

The advent of atomic clocks have provided another unit that is fixed for all practical purposes. In 1967, the *atomic second* was adopted as the fundamental unit of time (the SI second) by the international standards community. The atomic clock measures "hyperfine" states of the cesium atom. The accuracy of the measurement methods are extraordinary [Petley 1991]. Modern atomic clocks lose at most one second in $30,000$ years (with respect to ephemeris time). International

mentation or application dependent feature, from the semantics of the underlying temporal model. For example, assume that two events occur during the same hour-long chronon. In our model, the two events do not occur at the same time. Queries that ask for this information should respond that the events are not simultaneous. If the user changes the granularity to one second, the two events may or may not occur during the same second. Even if they do occur during the same second-long chronon, the events still cannot be said to occur simultaneously.

In addition to events, we are also interested in representing *intervals* and *spans*. An interval is a set of contiguous chronons, or in our model of time, a particular segment of the time-line. Our intervals are closed rather than open; an interval is denoted by the starting and terminating chronons. We will assume that an interval begins somewhere within the starting chronon, but exactly where is unknown, and ends somewhere within the terminating chronon.

Spans are unanchored durations of time, that is, a span is a number of chronons with no specific starting or terminating chronon. An example span is "60 seconds" which denotes a duration of one minute.

3 Clocks

Our representation of time is related to the physical world through a *base-line clock*. The base-line clock is an abstract clock. Unlike a Seiko wristwatch, the base-line clock exists only on paper. This section explores the nature of the base-line clock and how it relates to other clocks.

There are literally thousands of kinds of clocks. They vary greatly in their precision, accuracy, unit of measure, applicability to different times, and relevance. Before describing our proposal for a base-line clock, we examine briefly the major clocks that have been used to reckon time. A basic understanding of these clocks is necessary to justify our definition of the base-line clock. The appendix augments the discussion given here.

A clock is a physical process coupled with a method of measuring that process. A clock by itself does not measure time; it only measures the process. The units of measurement are the chronons of the clock. The size of the units is the granularity of the clock. The accuracy of the measurement method restricts the granularity.

For example, consider a clock based on the rotation of the Earth. A chronon in this clock might be the period from one dawn to the next. To observe when dawn occurs we can imagine a suitable method that involves being able to distinguish a white thread from a black thread by dawn's early light (and takes into account various factors such as clouds and solar eclipses). The best accuracy we could hope to achieve from this method is to distinguish one dawn from the next. The granularity of our clock is the unit of measure, the period from one dawn to the next (e.g., a "day").

One desirable clock feature is that chronons be of uniform duration (in comparison to other clocks). If the rotational clock described above were used at the North Pole, there would be a wide variation in the size of each chronon, compared to other clocks, between winter and summer.

clock. We begin with a comparative summary of existing time-stamps that are candidates for the physical realization. The inadequacies of these time-stamp layouts in realizing the base-line clock will become apparent. Consequently, we propose new layouts, describe those layouts, and briefly motivate our basic design goals and decisions. We test the utility of our design by making detailed time and space comparisons with existing time-stamps. We then propose additional layouts to support *indeterminate events*. An indeterminate event is an event that is known to have occurred, but precisely when is unknown. An appendix provides a more extensive discussion of the clocks introduced in the first part of this paper.

2 Model of the Time-Line

This paper is not a philosophical treatise on time; rather it defines the meaning of a time-stamp by showing a correspondence between existing clocks and time-stamp bit patterns. This correspondence is based on a particular model of time.

In the temporal database community, two basic time models have been proposed: the *continuous model*, in which time is viewed as being isomorphic to the real numbers, with each real number corresponding to a "point" in time, and the *discrete model*, in which time is viewed as being isomorphic to the natural numbers [Clifford & Tansel 1985]. We choose to use a discrete model. We propose that time is modeled by a closed interval on the real number line. Since it is a closed interval, time has both an origin and an endpoint. The time-line can be subdivided into a finite number of contiguous, pairwise disjoint subintervals. Each subinterval is called a *chronon* [Ariav 1986, Clifford & Rao 1987]. A chronon is the smallest duration of time that we can represent. Other terms, such as "instant" [Gadia 1986], "moment" [Allen & Hayes 1985], "time quantum" [Anderson 1982], and "time unit" [Navathe & Ahmed 1987, Tansel & Arkun 1986], have been used in the literature to describe a nondecomposable unit of time, but we favor the term chronon.

It is important to note that a chronon is not a point, but a line segment. We assume that the set of chronons is linearly ordered. A different model of time, where the chronons are partially ordered, is called *branching time*. Our model of time is linear rather than branching. The least and the greatest values in the linear ordering are specially designated, respectively, as *beginning* and *forever*. These two values are not chronons; rather, they are the extreme points on the time-line.

The size of each chronon is usually fixed by the *granularity* of the interpretation (e.g, second, day, year), but this need not be the case. We assume that every event occurs at a point in time; but, because we are using a discrete model, a chronon represents a line segment. We can only record that an event occurred *during* a particular chronon. We further assume that an event may occur anywhere within a chronon. We consider events to be instantaneous, and we do not model events that have duration [Anderson 1982], although we do model intervals (see below). Hence, the time that an event occurred is never precisely known. At best, only the chronon during which it happened is known. Two events that occur during the same chronon may still occur at different *times*.

Our motivation in selecting this model of time is to unhitch the granularity, often an imple-

1 Overview

Many database management systems and operating systems provide support for time values. This support is present at both the *logical* and *physical* levels. The logical level is the user's view of the time values and the query level operations permitted on those values, while the physical level concerns the bit layout of the time values and the bit level operations on those values. The physical level serves as a platform for the logical level but is inaccessible to the average user. This paper presents a proposal for the physical level of time values. A proposal for the logical level in the relational data model is presented elsewhere [Soo & Snodgrass 1992A].

At the physical level, time values are known as *time-stamps*. A time-stamp has both a *physical realization* and a *temporal interpretation*. The physical realization is an encoding as a pattern of bits while the temporal interpretation stipulates the meaning of each bit pattern, the *time* to which each pattern corresponds. Without such an interpretation, the meaning of a time-stamp stored in a database or by an operating system is unclear.

Various time-stamps are in use in commercial database management systems and operating systems. These time-stamp representations suffer from inadequate range, too coarse granularity, excessive space requirements, or a combination of these drawbacks. For example, the DB2 `date` type has a granularity of a day [Date & White 1990]. Most operating system formats reduce this coarse granularity to a second, but only cover about 120 years. DB2's `timestamp` type further reduces the granularity to a microsecond, but requires 10 bytes and still only covers 10,000 years, which is of little use to geologists. The SQL2 standard `datetime` type requires 27 bytes to accommodate an identical range and granularity [Melton 1990].

All previous time-stamps were defined in terms of seconds. However, as will be shown, there are many definitions of second, including the *ephemeris second*, the *sidereal second*, the *UT1* and *UT2 seconds*, the *SI second*, and the *TAI* and *UTC seconds*. When a range of less than 10,000 years is supported, the differences between these definitions are generally inconsequential. When ranges of several billion years are supported, however, these definitions differ significantly. Another consideration is that seconds based on the mean solar day, as in Universal Coordinated Time (UTC), are not defined more than 4.5 billion years ago, before the Earth existed.

In this paper, we propose a physical realization for time-stamps that encompasses the entire age of the universe to a granularity of a second, encompasses all of recorded history to a granularity of a microsecond, and requires only eight bytes. The proposed physical realization does not sacrifice the efficiency of run-time operations. We also provide a well-defined temporal interpretation for each possible bit pattern that avoids the ambiguity of previous proposals.

This paper is divided into two parts. The first part describes the *base-line clock*. The base-line clock provides the semantics for the physical representation. Our model of time, which underlies the base-line clock, is presented in the second section. The third section explores many kinds of clocks in detail. Some of these clocks are part of the base-line clock. The base-line clock, itself a collection of physical clocks, is then presented.

The second part of this paper describes the physical realization that is used for the base-line

Acknowledgements **31**

References **32**

Appendix **35**

Contents

1 Overview **1**

2 Model of the Time-Line **2**

3 Clocks **3**

3.1 Previously Defined Clocks 4

3.1.1 Day Clocks 4

3.1.2 Year Clocks 5

3.1.3 Dating the Prehistoric Past 6

3.1.4 Summary 6

3.2 The Proposed Base-line Clock 8

4 Existing Time-stamp Representations **11**

5 Proposed Time-stamp Formats **14**

5.1 Event Time-stamp Formats 14

5.2 Intervals 15

5.3 Spans 15

5.4 Design Decisions 16

6 Execution Efficiency **19**

7 Historical Indeterminacy **25**

7.1 Indeterminate Events 26

7.2 Indeterminate Intervals and Spans 29

7.3 Design Decisions 29

8 Summary and Future Work **30**

Copyright © Curtis E. Dyreson and Richard T. Snodgrass 1991

Time-stamp Semantics and Representation

Time-stamp Semantics and Representation

Curtis E. Dyreson[1]
Richard T. Snodgrass[2]

TR 92-16

Revised July 3, 1992

Abstract

Many database management systems and operating systems provide support for time values. At the physical level time values are known as *time-stamps*. A time-stamp has a *physical realization* and a *temporal interpretation*. The physical realization is a pattern of bits while the temporal interpretation is the meaning of each bit pattern, that is, the time each pattern represents. All previous proposals defined time-stamps in terms of seconds. However, as we show, there are at least seven definitions of this fundamental time unit. We propose a more precise temporal interpretation, the *base-line clock*, that constructs a time-line by using different well-defined clocks in different periods. We also propose time-stamp formats for events, intervals, and spans. These formats can represent all of time to the granularity of a second, and all of recorded history to a finer granularity of a microsecond. Our proposed formats were designed to be more space and time efficient than existing representations. We compare our formats with those used in common operating systems and database management systems.

[1]Department of Computer Science
University of Arizona
Tucson, AZ 85721
curtis@cs.arizona.edu

[2]Department of Computer Science
University of Arizona
Tucson, AZ 85721
rts@cs.arizona.edu