13.5.1 Time Representation, Calendars, and Time Dimensions

For temporal databases, time is considered to be an *ordered sequence* of **points** in some **granularity** tha determined by the application. For example, suppose that some temporal application never requires time u that are less than one second. Then, each time point represents one second in time using this granularity. reality, each second is a (short) *time duration*, not a point, since it may be further divided into millisecon microseconds, and so on. Temporal database researchers have used the term **chronon** instead of point describe this minimal granularity for a particular application. The main consequence of choosing a minim granularity—say, one second—is that events occurring within the same second will be considered to *simultaneous events*, even though in reality they may not be.

Because there is no known beginning or ending of time, one needs a reference point from which measure specific time points. Various calendars are used by various cultures (such as Gregorian (Wester Chinese, Islamic, Hindu, Jewish, Coptic, etc.) with different reference points. A **calendar** organizes time in different time units for convenience. Most calendars group 60 seconds into a minute, 60 minutes into an ho 24 hours into a day (based on the physical time of earth's rotation around its axis), and 7 days into a wee Further grouping of days into months and months into years either follow solar or lunar natural phenomen and are generally irregular. In the Gregorian calendar, which is used in most Western countries, days a grouped into months that are either 28, 29, 30, or 31 days, and 12 months are grouped into a year. Compl formulas are used to map the different time units to one another.

In SQL2, the temporal data types (see Chapter 7) include DATE (specifying Day, Month, and Year DD-MM-YYYY), TIME (specifying Hour, Minute, and Second as HH:MM:SS), TIMESTAMP (specifying Date/Time combination, with options for including sub-second divisions if they are needed), INTERVAL relative time duration, such as 10 days or 250 minutes), and PERIOD (an *anchored* time duration with a fixe starting point, such as the 10-day period from January 1, 1999, to January 10, 1999, inclusive).[83]

Event Information Versus Duration (or State) Information. A temporal database will store informatio concerning when certain events occur, or when certain facts are considered to be true. There are sever different types of temporal information. **Point events** or **facts** are typically associated in the database wit a **single time point** in some granularity. For example, a bank deposit event may be associated with th timestamp when the deposit was made, or the total monthly sales of a product (fact) may be associated wit a particular month (say, February 1999). Note that even though such events or facts may have differen granularities, each is still associated with a *single time value* in the database. This type of information is ofte represented as **time series data** as we shall discuss in Section 13.4.5. **Duration events** or **facts**, on the othe hand, are associated with a specific **time period** in the database.[84] For example, an employee may have worke in a company from August 15, 1993, till November 20, 1998.

A **time period** is represented by its **start** and **end time points** [START-TIME, END-TIME]. For example the above period is represented as [15-08-1993, 20-11-1998]. Such a time period is often interpreted to mea the *set of all time points* from start-time to end-time, inclusive, in the specified granularity. Hence, assumin day granularity, the period [15-08-1993, 20-11-1998] represents the set of all days from August 15, 1993, unti November 20, 1998, inclusive.[85]

83. Unfortunately, the terminology has not been used consistently. For example, the term *interval* is often used to denote an anchore duration. For consistency, we shall use the SQL terminology.

84. This is the same as an anchored duration. It has also been frequently called a **time interval**, but to avoid confusion we will use perio to be consistent with SQL terminology.

85. The representation [15-08-1993, 20-11-1998] is called a *closed interval* representation. One can also use an *open interval*, denoted [15-08-1993, 21-11-1998), where the set of points *does not include* the end point. Although the latter representation is sometimes more convenient, we shall use closed intervals throughout to avoid confusion.

lid Time and Transaction Time Dimensions. Given a particular event or fact that is associated with rticular time point or time period in the database, the association may be interpreted to mean different ngs. The most natural interpretation is that the associated time is the time that the event occurred, or period during which the fact was considered to be true *in the real world.* If this interpretation is used, the ciated time is often referred to as the **valid time**. A temporal database using this interpretation is called alid time database.**

However, a different interpretation can be used, where the associated time refers to the time when the rmation was actually stored in the database; that is, it is the value of the system time clock when the rmation is valid *in the system*.[86] In this case, the associated time is called the **transaction time**. A temporal abase using this interpretation is called a **transaction time database.**

Other interpretations can also be intended, but these two are considered to be the most common ones, d they are referred to as **time dimensions.** In some applications, only one of the dimensions is needed and other cases both time dimensions are required, in which case the temporal database is called a **bitemporal** abase. If other interpretations are intended for time, the user can define the semantics and program the plications appropriately, and it is called a **user-defined time.**

The next section shows with examples how these concepts can be incorporated into relational databases, d Section 13.4.3 shows an approach to incorporate temporal concepts into object databases.

13.5.2 Incorporating Time in Relational Databases Using Tuple Versioning

id Time Relations. Let us now see how the different types of temporal databases may be represented the relational model. First, suppose that we would like to include the history of changes as they occur the real world. Consider again the database in Figure 13.13, and let us assume that, for this application, granularity is day. Then, we could convert the two relations EMPLOYEE and DEPARTMENT into **valid time** tions by adding the attributes VST (Valid Start Time) and VET (Valid End Time), whose data type is DATE der to provide day granularity. This is shown in Figure 13.19a, where the relations have been renamed VT and DEPT_VT, respectively.

Consider how the EMP_VT relation differs from the nontemporal EMPLOYEE relation (Figure 13.13).[87] In VT, each tuple V represents a version of an employee's information that is valid (in the real world) only g the time period [V.VST, V.VET], whereas in EMPLOYEE each tuple represents only the current state or ent version of each employee. In EMP_VT, the **current version** of each employee typically has a special now as its valid end time. This special value, ***now***, is a **temporal variable** that implicitly represents the ent time as time progresses. The nontemporal EMPLOYEE relation would only include those tuples from the VT relation whose VET is *now*.

Figure 13.20 shows a few tuple versions in the valid-time relations EMP_VT and DEPT_VT. There are two ons of Kumar, three versions of Chandra, one version of Agarwal, and one version of Narayan. We can ee how a valid time relation should behave when information is changed. Whenever one or more utes of an employee are **updated,** rather than actually overwriting the old values, as would happen in a mporal relation, the system should create a new version and **close** the current version by changing its o the end time. Hence, when the user issued the command to update the salary of Kumar effective on 1, 2003, to Rs. 30000, the second version of Kumar was created (see Figure 13.20). At the time of this the first version of Kumar was the current version, with *now* as its VET, but after the update *now* was

25000 Rs. 31, 7. now

---

xplanation is more involved, as we shall see in Section 13.4.3.

mporal relation is also called a **snapshot relation** as it shows only the *current snapshot* or *current state* of the database.

changed to May 31, 2003 (one less than June 1, 2003, in day granularity), to indicate that the version has become a closed or history version and that the new (second) version of Kumar is now the current one.

(a) EMP_VT

NAME	ENO	SALARY	DNO	SUPERVISOR_ENO	VST	VET

DEPT_VT

DNAME	DNO	TOTAL_SAL	MANAGER_ENO	VST	VET

(b) EMP_TT

NAME	ENO	SALARY	DNO	SUPERVISOR_ENO	TST	TET

DEPT_TT

DNAME	DNO	TOTAL_SAL	MANAGER_ENO	TST	TET

(c) EMP_BT

NAME	ENO	SALARY	DNO	SUPER VISOR_ENO	VST	VET	TST	TET

DEPT_BT

DNAME	DNO	TOTAL_SAL	MANAGER_ENO	VST	VET	TST	TET

FIGURE 13.19 Different types of temporal relational databases. (a) Valid time database schema. (b) Transaction time database schema. (c) Bitemporal database schema.

EMP_VT

NAME	ENO	SALARY	DNO	SUPER VISOR_ENO	VST	VET
Kumar	123456789	25000	5	333445555	15-06-2002	31-05-2003
Kumar	123456789	30000	5	333445555	01-06-2003	now
Chandra	333445555	25000	4	999887777	20-08-1999	31-01-2001
Chandra	333445555	30000	5	999887777	01-02-2001	31-03-2002
Chandra	333445555	40000	5	888665555	01-04-2002	now
Agarwal	222447777	28000	4	999887777	01-05-2001	10-08-2002
Narayan	666884444	38000	5	333445555	01-08-2003	now

. . .

DEPT_VT

DNAME	DNO	MANAGER_ENO	VST	VET
Research	5	888665555	20-09-2001	31-03-2002
Research	5	333445555	01-04-2002	now

. . .

FIGURE 13.20 Some tuple versions in the valid time relations EMP_VT and DEPT_VT.