TR85-004

A Taxonomy of Time in Databases

Richard Snodgrass & Ilsoo Ahn

Department of Computer Science
University of North Carolina
Chapel Hill, North Carolina 27514

March, 1985

This paper will appear in Proceedings of the ACM-SIGMOD International Conference on Management of Data, Austin, Texas, May 1985.

A Taxonomy of Time in Databases

March, 1985

Richard Snodgrass†
Ilsoo Ahn

Department of Computer Science
University of North Carolina
Chapel Hill, NC 27514

Abstract

The need for supporting time varying information in databases has been recognized for quite some time. Many authors have proposed numerous schemes to satisfy this need by incorporating one or two time attributes in the database. Unfortunately, there has been confusion concerning the terminology and definition of these time attributes. This paper proposes a new taxonomy of three times for use in databases, one that is more cleanly defined, that may be conceptualized in a pictorial fashion, and that defines several kinds of databases differentiated by their ability to represent temporal information. The paper argues that future database management systems should support all three times to fully capture time varying behavior.

1. Introduction

The need for recording time varying information in databases has been recognized for quite some time [Bubenko 1976]. There have been significant research activities in formulating a semantics of time at the conceptual level [Anderson 1982, Breutmann et al. 1979, Bubenko 1977, Hammer & McLeod 1981, Klopprogge 1981], developing a model for time varying databases analogous to the relational model for static databases [Clifford & Warren 1983, Codd 1979, Sernadas 1980], and the design of temporal query languages [Ariav & Morgan 1981, Ben-Zvi 1982, Jones & Mason 1980, Snodgrass 1982]. Recently, it has been argued that a single time attribute is insufficient, and that two time attributes are necessary to fully capture time-varying information. Unfortunately, there has been some confusion concerning terminology and the definition of these time attributes.

The next section will discuss the various characteristics attributed to the two times; the third section will illustrate the difficulties posed by the vague definition of these times. The fourth section will present a new taxonomy of time in databases to replace the two previous times. The new taxonomy consists of three distinct time concepts and four distinct kinds of database management systems (DBMS), differing in their support of the new time concepts. The final section will compare the new taxonomy with the old one.

2. Previous Characterizations

In this paper, we will use the terms *physical time* and *logical time* [Lum et al. 1984] to discuss the concepts as they appear in the literature. Physical time has also been called transaction time [Copeland & Maier 1984], registration time [Ben-Zvi 1982], data-valid-time-from/to [Mueller & Steinbauer 1983], and start/end time [Reed 1978]. Logical time has also been called event time [Copeland & Maier 1984], effective time [Ben-Zvi 1982], state [Clifford & Warren 1983], valid time [Snodgrass 1984], and start/end time [Jones et al.

† The work of this author was supported by NSF grant DCR-8402339 and by an IBM Faculty Development Award.

1979, Jones & Mason 1980]. Each paper has defined the terms in slightly different ways. There is general agreement on the definitions, but little consensus concerning the details. The differences identified by previous authors between physical and logical time may be characterized in terms of three related attributes. The purpose of this section is to discuss these attributes and to examine their contributions to the concepts of logical and physical time. We will proceed by stating the view presented in the literature, then follow in the next section with an analysis of this view. This summary is drawn primarily from the works of Copeland and Maier [Copeland & Maier 1984], Dadam et al. [Dadam et al. 1984], and Lum et al. [Lum et al. 1984], although others have also noticed that a single time stamp or a pair of time stamps is inadequate.

2.1. Reality versus Representation

The correspondence of the model stored in the database with reality is one aspect that is used to distinguish between logical and physical time. Logical time is characterized as the time that an event occurs in reality; physical time is characterized as the time when the data concerning the event was stored in the database. Examples include retroactive salary changes, release dates of engineering versions, scheduled events that have not yet occurred, and scheduled events that were suppose to occur, yet did not.

2.2. Update Flexibility

The types of update permitted to time values is another way that logical and physical time have been differentiated in the literature. A physical time value may be added to the database, yet once it has been added, it may not be changed. The concept of a non-stop running clock is evoked to indicate how the time values are generated. Logical time values, on the other hand, are always subject to change, since discrepancies between the history (a sequence of events or time intervals) as it actually occurred and the representation of the history as stored in the database will often be detected after the fact. The distinction then is between permitting only appends and permitting arbitrary modifications.

2.3. Application Dependency

The third attribute used to distinguish between physical and logical time is that of application dependency. Logical time is generally characterized in the literature as being application-dependent, while physical time is considered to be application-independent. While this attribute is the hardest to define precisely, it is usually equated with the control the user of the DBMS has over the value of a temporal domain in the database. If the value can be computed automatically by the DBMS, the value must necessarily be independent of any particular application and must have a simple semantics. An application-dependent time value, on the other hand, must have been defined explicitly by the user. Its value must also be specified by the user, and may thus be quite complex. The integrity of this data must be maintained by the user; the value must be modifiable by users when a discrepancy is discovered between the real world and the database model. Hence, the DBMS cannot guarantee the integrity of logical time values. The relationship between the types of time identified in the literature and their attributes is shown in Figure 1.

3. Comparison

Two of the attributes differentiating physical and logical time, those of reality versus representation and update flexibility, are reasonably precise concepts. They are also strongly related to each other, in that a time value that records when the data was stored cannot later be changed. The third attribute, that of application dependence, is unfortunately fraught with difficulties. It makes certain assumptions of which the most crucial is that all actions performed by the DBMS are application-independent. This assumption is not valid, at least to a certain degree. The database schema, which directs most actions by the DBMS, is certainly application-dependent. Many DBMS's allow the specification of integrity constraints, which are application-dependent, yet are interpreted automatically by the DBMS without user intervention. Application-dependent values can be handled by the DBMS if their semantics can be defined in terms the DBMS can interpret.

An example often cited of the distinction between application-independent and application-dependent time is a retroactive salary raise, where the time at which the raise was recorded (say, 12/1/83) is considered application-independent, as it is not under the user's control, whereas the time at which the raise was to take effect (say, 8/1/83) is considered application-dependent, as it is in some sense arbitrary and under the user's control.

Examining this situation more closely, however, can result in precisely the reverse semantics. In many commercial settings, salary updates are batched together and executed against the database only once or twice a month, whereas payments might be made at the last possible date to minimize cashflow problems, and hence may occur at arbitrary times during the month. That a salary update was performed by the DBMS on 12/1/83 may simply be an artifact of when salary updates are entered, which is application dependent. On the other hand, the user has no control over when the salary was changed, and hence the effective date is in this sense application-independent.

The point to be made is that characterizing a time value as being dependent or independent of an application involves fairly subtle issues of the semantics of that value, both as interpreted within the DBMS and as applied to the situation being modeled. Given these difficulties, this attribute appears to be less than ideal in differentiating physical and logical time.

Reference	Terminology	Append -Only	Application Independent	Representation vs. Reality
[Ariav & Morgan 1982]	Time	Yes	Yes	Representation
[Ben-Zvi 1982]	Registration	Yes	Yes	Representation
	Effective	No	Yes	Reality
[Clifford & Warren 1983]	State	No	Yes	
[Copeland & Maier 1984]	Transaction	Yes	Yes	Representation
	Event (1)	No	No	Reality
[Dadam et al. 1984] & [Lum et al. 1984]	Physical	(2)	Yes	Representation
	Logical (1)	No	No	Reality
[Jones et al. 1979] & [Jones & Mason 1980]	Start/End	(2)	Yes	Reality
	User Defined	No	No	Reality
[Mueller & Steinbauer 1983]	Data-Valid-Time-From/To	(3)	Yes	Representation (4)
[Reed 1978]	Start/End	Yes	Yes	Representation
[Snodgrass 1984]	Valid Time	No	Yes	Reality

Notes:
(1) Not actually supported by the system
(2) Can make corrections only
(3) Can make changes only in the future
(4) Reality is indicated only in the future

Figure 1 : Types of Time

4. A New Characterization

The previous section argued that physical and logical time are not well defined, and that application time is particularly problematic. In this section we introduce a new taxonomy of time for use in databases. This taxonomy is more clearly defined, being based on reality versus representation, may be conceptualized in a

pictorial fashion which aids understanding, and defines several kinds of databases differentiated by their ability to represent temporal information. Though the following discussion is based on the relational model, similar arguments also apply to hierarchical or network models. We will first discuss static databases, focusing on their representational inadequacies. We then define three new time concepts to replace the vaguely defined physical and logical time. We introduce each time concept by discussing the features associated with a particular kind of DBMS supporting that time concept.

4.1. Static Databases

Conventional databases model the real world, as it changes dynamically, by a snapshot at a particular point in time. A *state* or an *instance* of a database is its current contents, which does not necessarily reflect the current status of the real world.

Updating the state of a database is performed using data manipulation operations such as insertion, deletion or replacement, taking effect as soon as it is committed. In this process, past states of the database, and those of the real world, are discarded and forgotten completely. We term this type of database a *static database*.

In the relational model, a database is a collection of *relations*. Each relation consists of a set of *tuples* with the same set of *attributes*, and is usually represented as a 2-dimensional table (see Figure 2). As changes occur in the real world, changes are made in this table.

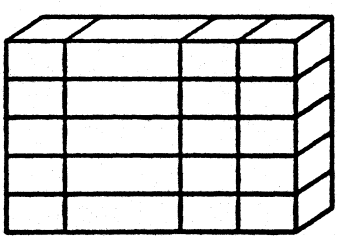

Figure 2 : A Static Relation

For example, an instance of a relation 'faculty' at a certain moment may be

name	rank
Merrie	full
Tom	associate

and a query in Quel, a tuple calculus based language for the INGRES database management system [Held et al. 1975], requesting Merrie's rank,

```
range of f is faculty
retrieve (f.rank)
        where f.name = "Merrie"
```

yields

rank
full

There are many situations where this static database relying on snapshots is inadequate. For example, it cannot answer queries such as

What was Merrie's rank 2 years ago?
(historical query)
How did the number of faculty change
over the last 5 years? (trend analysis)

nor record facts like

Merrie was promoted to a full professor
starting last month. (retroactive change)
James is joining the faculty next month.
(postactive change)

Without system support in this respect, many applications have had to maintain and handle temporal information in an ad-hoc manner.

4.2. Static Rollback Databases

One approach to resolve the above deficiencies is to store all past states, indexed by time, of the static database as it evolves. Such an approach requires a representation of *transaction time*, the time the information was stored in the database. A relation under this approach can be illustrated conceptually in three dimensions (Figure 3) with transaction time serving as the third axis. The relation can be regarded as a sequence of static relations indexed by time. By moving along the time axis and taking a vertical slice of the cube, it is possible to get a snapshot of the relation as of some time in the past (a static relation) and make queries upon it. The operation of taking a vertical slice is termed *rollback*, and a database supporting it is termed a *static rollback database*. Changes to a static rollback database may only be made to the most recent static state. The relation illustrated in Figure 3 had three transactions applied to it, starting from the null relation: (1) the addition of three tuples, (2) the addition of a tuple, and (3) the deletion of one tuple (entered in the first transaction) and the addition of another tuple. Each transaction results in a new static relation being appended to the front of the cube; once a transaction has completed, the static relations in the static rollback relation may not be altered.

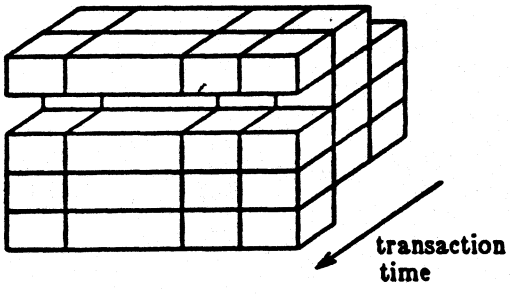

Figure 3 : A Static Rollback Relation

One limitation of supporting transaction time is that the history of database activities, rather than the history of the real world, is recorded. A tuple becomes valid as soon as it is entered into the database as in a static database. There is no way to record retroactive/postactive changes, nor to correct errors in past tuples. Errors can sometimes be overridden (if they are in the current state) but they cannot be forgotten.

Implementing a static rollback relation in this way is impractical, due to excessive duplication: the tuples that don't change between states must be duplicated in the new state. Another approach that partially addresses this difficulty appends the start and end points of the transaction time to each tuple, indicating the points in time when the tuple was in the database. A typical relation in this approach looks like Figure 4. The double vertical bars separate the non-temporal domains from the DBMS-maintained temporal domains. The latter domains do not appear in the schema for the relation, but may rather be considered part of the overheads associated with each tuple. Note the fact that Merrie was previously an associate professor, a fact which could not be expressed in the example for a static database.

name	rank	transaction time (start)	(end)
Merrie	associate	08/25/77	12/15/82
Merrie	full	12/15/82	∞
Tom	associate	12/07/82	∞
Mike	assistant	01/10/83	02/25/84

Figure 4 : A Static Rollback Relation

Any query language may be converted to one which may query a static rollback database by adding a clause effecting the rollback. TQuel (*T*emporal *QUE*ry *L*anguage) [Snodgrass 1984, Snodgrass 1985], an extension of Quel for temporal databases, augments the retrieve statement with an *as of* clause to specify the relevant transaction time. The TQuel query

```
range of f is faculty
retrieve (f.rank)
        where f.name = "Merrie"
        as of "12/10/82"
```

on a 'faculty' relation shown in Figure 4 will find the rank of Merrie as of 12/10/82:

rank
associate

Note that the result of a query on a static rollback database is a pure static relation.

The concept of transaction time has appeared in several systems, including *GemStone* [Copeland & Maier 1984], *MDM/DB* (Model Data Management/Database) [Ariav & Morgan 1982], and the *SWALLOW* object store [Reed 1978, Svobodova 1981].

4.3. Historical Databases

While static rollback databases record a sequence of static states, *historical databases* record a single *historical state* per relation, storing the history as it is best known. As errors are discovered, they are corrected by modifying the database. Previous states are *not* retained, so it is not possible to view the database as it was in the past. There is no record kept of the errors that have been corrected. Historical databases are similar to static databases in this respect. Historical databases must represent *valid time*, the time that the stored information models reality.

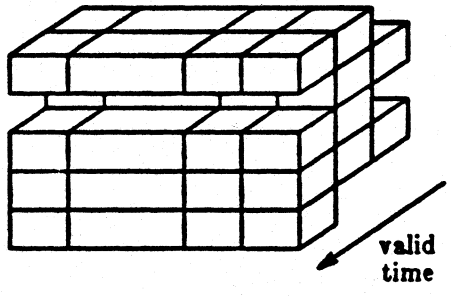

Figure 5: An Historical Relation

Historical databases may also be illustrated in three dimensions (see Figure 5). Though its

illustration looks similar to one for the static rollback database (in fact, for many transaction sequences, it will be identical), the label of the time axis has been changed to valid time and the semantics are more closely related to reality, instead of update history. Therefore more sophisticated operations are necessary to manipulate the complex semantics of valid time adequately, compared to the simple rollback operation.

A second distinction between historical and static rollback databases is that historical DBMS's support arbitrary modification, whereas static rollback DBMS's only allow static states to be appended. The same sequence of transactions which resulted in the static rollback relation in Figure 3 also results in the historical relation in Figure 5. However, a later transaction (not possible on a static rollback relation) has removed an erroneous tuple inserted on the first transaction (compare Figures 3 and 5 closely). Static rollback DBMS's can rollback to an incorrect previous static relation; historical DBMS's can record the current knowledge about the past.

Historical databases also incorporate user-defined time, which will be discussed in the context of temporal databases. Both valid time and user-defined time concern modeling of reality, and so it is appropriate that they should appear together.

Historical databases require more sophisticated query languages. There have been two such languages developed: LEGOL 2.0 [Jones et al. 1979], based on the relational algebra, and TQuel [Snodgrass 1984], based on Quel [Held et al. 1975], a relational calculus query language. *LEGOL 2.0* [Jones & Mason 1980] was developed for writing complex rules such as those in legislation or high level system specification where the correct handling of time is important. It also attaches to each tuple two time attributes which delimit the period of existence for the associated member of the entity set.

TQuel supports the expression of historical queries by augmenting the *retrieve* statement with a *valid* clause to specify how the implicit time domain is computed, and a *when* predicate to specify the temporal relationship of tuples participating in a derivation. These added constructs handle complex temporal relationships such as *start of, precede,* and *overlap.*

As with static rollback databases, implementing a historical relation directly as above is impractical. Figure 6 illustrates an alternative: appending the endpoints of the valid time to each tuple, indicating the points in time when the tuple accurately modeled reality. Like the transaction time in static rollback databases, the valid time is not included in the relation schema.

name	rank	valid time (from)	(to)
Merrie	associate	09/01/77	12/01/82
Merrie	full	12/01/82	∞
Tom	associate	12/05/82	∞
Mike	assistant	01/01/83	03/01/84

Figure 6 : A Historical Relation

The TQuel query requesting Merrie's rank when Tom arrived,

```
range of f1 is faculty
range of f2 is faculty

retrieve (f1.rank)
        where f1.name = "Merrie"
        and f2.name = "Tom"
        when f1 overlap start of f2
```

on the historical relation 'faculty' in Figure 6 yields

rank	valid time (from)	(to)
full	12/01/82	∞

Note that the derived relation is also an historical relation, which may be used in further historical queries. While both this query and the example given for a static rollback relation seem to query Merrie's rank on 12/05/82, the answers are different. The reason is that Merrie was promoted on 12/01/82, but this information was recorded in the database two weeks later. Hence, the database was inconsistent with reality for that period of time. In the historical database, the error was corrected, but it is not possible to determine that, at least for a while, the database was inconsistent.

Historical databases have been the subject of several research efforts, including *CSL* (Conceptual Schema Language) [Breutmann et al. 1979], *TERM* (Time-extended Entity Relationship Model) [Klopprogge 1981], the intensional logic IL_s [Clifford & Warren 1983], and *AMPPL-II* (Associative Memory Parallel Language II) [Findler & Chen 1971].

4.4. Temporal Databases

Benefits of both approaches can be combined by supporting both transaction time and valid time. While a static rollback database views tuples valid at some time as of that time, and a historical database always views tuples valid at some moment as of *now*, a temporal DBMS makes it possible to view tuples valid at some moment seen as of some other moment, completely capturing the history of retroactive/postactive changes.

We use the term *temporal database* to emphasize the need for both valid time and transaction time in handling temporal information. Since there are two time axes involved now, it should be illustrated in four dimensions (Figure 7 shows a *single* temporal relation). A temporal relation may be thought of as a sequence of historical states, each of which is a complete historical relation. The rollback operation on a temporal relation selects a particular historical state, on which an historical query may be performed. Each transaction causes a new historical state to be created; hence, temporal relations are append-only. The temporal relation in Figure 7 is the result of four transactions, starting from a null relation: (1) three tuples were added, (2) one tuple was added, (3) one tuple was added and an existing one deleted, and (4) a previous tuple was deleted (presumably it should not have been there in the first place).

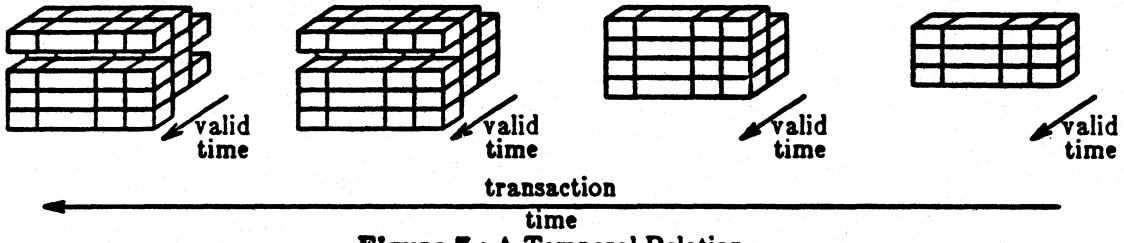

Figure 7 : A Temporal Relation

name	rank	valid time (from)	valid time (to)	transaction time (start)	transaction time (end)
Merrie	associate	09/01/77	∞	08/25/77	12/15/82
Merrie	associate	09/01/77	12/01/82	12/15/82	∞
Merrie	full	12/01/82	∞	12/15/82	∞
Tom	full	12/05/82	∞	12/01/82	12/07/82
Tom	associate	12/05/82	∞	12/07/82	∞
Mike	assistant	01/01/83	∞	01/10/83	02/25/84
Mike	assistant	01/01/83	03/01/84	02/25/84	∞

Figure 8 : A Temporal Relation

For example, the relation in Figure 6 will look like Figure 8 after adding transaction time. It shows that Merrie started working on 09/01/77, information that was entered into the database on 08/25/77 as a postactive data. Then she was promoted on 12/01/82, but the fact was recorded on 12/15/82 retroactively. Tom was entered into the database on 12/01/82 as joining the faculty as a full professor on 12/05/82; the fact that he was actually an associate professor was noted on 12/07/82. Mike left the faculty effective on 03/01/84, which was recorded on 02/25/84. Note all the details of history captured here, which were not expressible in other more restrictive databases. The TQuel query

```
range of f1 is faculty
range of f2 is faculty

retrieve (f1.rank)
        where f1.name = "Merrie"
        and f2.name = "Tom"
        when f1 overlap start of f2
        as of "12/10/82"
```

on this relation determines Merrie's rank when Tom arrived, according to the state of the database as of 12/10/82. The result is

rank	valid time (from)	(to)	transaction time (start)	(end)
associate	09/01/77	∞	08/25/77	12/15/82

This derived relation is a temporal relation, so further temporal relations can be derived from it. If a similar query is made as of 12/20/82, the answer would be *full* because the fact was recorded retroactively by that time.

TRM (Time Relational Model) is another example of a temporal database [Ben-Zvi 1982]. However, the query language defined for TRM is not a temporal query language, because it can derive only static relations.

4.5. User-defined time

User-defined time [Jones & Mason 1980] is necessary when additional temporal information, not handled by transaction or valid time, is stored in the database. As an example, consider the 'promotion' relation shown in Figure 9. Since it is an event relation, only one valid time is necessary. The effective date is the date shown on the promotion letter that the promotion was to take effect; the valid date is the date the promotion letter was signed, i.e., the date the promotion was validated; and the transaction date is the date the information concerning the promotion was stored in the database. Merrie's retroactive promotion to full was signed four days before it was recorded in the database. The effective date is application-specific; it is merely a date which appears on the promotion letter. The values of user-defined temporal domains are not interpreted by the DBMS, and are thus the easiest to support; all that is needed is an internal representation and input and output functions. Such domains will then be present in the relation schema. Conventional DBMS's supporting application time include the ENFORM DBMS [Tandem 1983], Query-by-Example [Bontempo 1983], an experimental version of INGRES [Overmyer & Stonebraker 1982], and MicroINGRES [Relational 1984].

name	rank	effective date	valid time (at)	transaction time (start)	(end)
Merrie	associate	09/01/77	08/25/77	08/25/77	∞
Merrie	full	12/01/82	12/11/82	12/15/82	∞
Tom	full	12/05/82	12/05/82	12/01/82	12/07/82
Tom	associate	12/05/82	12/07/82	12/07/82	∞
Mike	assistant	01/01/83	01/01/83	01/10/83	∞
Mike	left	03/01/84	02/25/84	02/25/84	∞

Figure 9 : A Temporal Event Relation

5. Conclusions

Three kinds of time, transaction time, valid time, and user-defined time, were introduced to replace the vague formulation of physical and logical time found in the literature. Database management systems may be categorized in terms of their support for handling temporal information. As shown in Figure 10, two orthogonal criteria are capabilities for rollback and historical queries. These criteria differentiate four types of databases: static, static rollback, historical and temporal. Support of the rollback capability requires the incorporation of transaction time, which concerns the representation; support of historical queries requires the incorporation of valid time, which is associated with reality (see Figure 11). DBMS's supporting rollback are append-only, whereas those not supporting rollback allow updates of arbitrary information. The attributes associated with the three kinds of time are illustrated in Figure 12, which should be compared to Figure 1.

The new time concepts may be loosely compared with those appearing previously in the literature. Transaction time is most closely associated with physical time, and valid and user-defined time with logical time. However, as we have shown in an earlier section, logical and physical time have not been precisely defined, whereas the new terms have been carefully defined by examining the aspects they model and the limitations they impose on the DBMS. Figure 13 classifies the time supported in existing or proposed systems according to the new taxanomy.

While fifteen years of research has focused on formalizing and implementing static databases, only a few researchers have recently studied the formalization of historical databases (e.g., [Clifford & Warren 1983]) and the implementation of static rollback databases (e.g., [Lum et al. 1984]). To the authors' knowledge, there has been nothing published on formalizing static rollback or temporal databases, nor implementing historical or temporal databases. The special opportunities promised by temporal databases are, at this time, matched by the challenges in supporting them.

	No Rollback	Rollback
Static Queries	Static	Static Rollback
Historical Queries	Historical	Temporal

Figure 10 : Types of Databases

	Transaction	Valid	User-defined
Static			
Static Rollback	√		
Historical		√	√
Temporal	√	√	√

Figure 11 : Attributes of the New Kinds of Databases

Terminology	Append-Only	Application Independent	Representation vs. Reality
Transaction	Yes	Yes	Representation
Valid	No	Yes	Reality
User-defined	No	No	Reality

Figure 12 : Attributes of the New Kinds of Time

Reference	System or Language	Transaction Time	Valid Time	User-defined Time
[Ariav & Morgan 1982]	MDM/DB	√		
[Ben-Zvi 1982]	TRM	√	√	
[Bontempo 1983]	QBE			√
[Breutmann et al. 1979]	CSL		√	
[Clifford & Warren 1983]	IL_s		√	
[Copeland & Maier 1984]	GemStone	√		
[Findler & Chen 1971]	AMPPL-II		√	
[Jones & Mason 1980]	LEGOL 2.0		√	√
[Klopprogge 1981]	TERM		√	
[Lum et al. 1984]	AIM	√		
[Relational 1984]	MicroINGRES			√
[Mueller & Steinbauer 1983]	—	√		
[Overmyer & Stonebraker 1982]	INGRES			√
[Reed 1978]	SWALLOW	√		
[Snodgrass 1985]	TQuel	√	√	√
[Tandem 1983]	ENFORM			√
[Wiederhold et al. 1975]	TODS		√	

Figure 13 : Time Support in Existing or Proposed Systems

Bibliography

[Anderson 1982] Anderson, T. L. *Modeling Time at the Conceptual Level.* In *Improving Database Usability and Responsiveness,* Ed. P. Scheuermann. Jerusalem, Israel: Academic Press, 1982 pp. 273-297.

[Ariav & Morgan 1981] Ariav, G. and H. L. Morgan. *MDM: Handling the time dimension in generalized DBMS.* Technical Report. The Wharton School, University of Pennsylvania. May 1981.

[Ariav & Morgan 1982] Ariav, G. and H. L. Morgan. *MDM: Embedding the Time Dimension in Information Systems.* Technical Report 82-03-01. Department of Decision Sciences, The Wharton School, University of Pennsylvania. 1982.

[Ben-Zvi 1982] Ben-Zvi, J. *The Time Relational Model.* PhD. Diss. UCLA, 1982.

[Bontempo 1983] Bontempo, C. J. *Feature Analysis of Query-By-Example,* in Relational Database Systems. New York: Springer-Verlag, 1983. pp. 409-433.

[Breutmann et al. 1979] Breutmann, B., E. F. Falkenberg and R. Mauer. *CSL: A language of defining conceptual schemas,* in Data Base Architecture. Amsterdam: North Holland, Inc., 1979.

[Bubenko 1976] Bubenko, J. A., Jr. *The temporal dimension in information modeling.* Technical Report RC 6187 #26479. IBM Thomal J. Watson Research Center. Nov. 1976.

[Bubenko 1977] Bubenko, J. A., Jr. *The Temporal Dimension in Information Modeling,* in Architecture and Models in Data Base Management Systems. North-Holland Pub. Co., 1977.

[Clifford & Warren 1983] Clifford, J. and D. S. Warren. *Formal Semantics for Time in Databases. ACM Transactions on Database Systems,* 8, No. 2, June 1983, pp. 214-254.

[Codd 1979] Codd, E.F. *Extending the Database Relational Model to Capture More Meaning. ACM Transactions on Database Systems,* 4, No. 4, Dec. 1979, pp. 397-434.

[Copeland & Maier 1984] Copeland, G. and D. Maier. *Making Smalltalk a Database System.* In *Proceedings of the Sigmod '84 Conference,* June 1984 pp. 316-325.

[Dadam et al. 1984] Dadem, P., V. Lum and H.-D. Werner. *Integration of Time Versions into a Relational Database System.* In *Proceedings of the Conference on Very Large Data Bases,* 1984.

[Findler & Chen 1971] Findler, N. and D. Chen. *On the problems of time retrieval, temporal relations, causality, and coexistence.* In *Proceedings of the International Joint Conference on Artificial Intelligence,* Imperial College: Sep. 1971.

[Hammer & McLeod 1981] Hammer, M. and D. McLeod. *Database Description with SDM: A Semantic Database Model. ACM Transactions on Database Systems,* 6, No. 3, Sep. 1981, pp. 351-386.

[Held et al. 1975] Held, G.D., M. Stonebraker and E. Wong. *INGRES--A relational data base management system. Proceedings of the 1975 National Computer Conference,* 44 (1975) pp. 409-416.

[Jones et al. 1979] Jones, S., P. Mason and R. Stamper. *LEGOL 2.0: a relational specification language for complex rules. Information Systems,* 4, No. 4, Nov. 1979, pp. 293-305.

[Jones & Mason 1980] Jones, S. and P. J. Mason. *Handling the Time Dimension in a Data Base.* In *Proceedings of the International Conference on Data Bases,* Ed. S. M. Deen and P. Hammersley. British Computer Society. University of Aberdeen: Heyden, July 1980 pp. 65-83.

[Klopprogge 1981] Klopprogge, M. R. *TERM: An approach to include the time dimension in the entity-relationship model.* In *Proceedings of the Second International Conference on the Entity Relationship Approach,* Oct. 1981.

[Lum et al. 1984] Lum, V., P. Dadam, R. Erbe, J. Guenauer, P. Pistor, G. Walch, H. Werner and J. Woodfill. *Designing DBMS Support for the Temporal Dimension.* In *Proceedings of the Sigmod '84 Conference,* June 1984 pp. 115-130.

[Mueller & Steinbauer 1983] Mueller, T. and D. Steinbauer. *Eine Sprachschnittstele zur Versionenkontrolle in CAM-Datanbanken,* in Informatik-Fachberichte. Berlin: Springer, 1983. pp. 76-95.

[Overmyer & Stonebraker 1982] Overmyer, R. and M. Stonebraker. *Implementation of a Time Expert in a Database System. SIGMod Record,* 12, No. 3, Apr. 1982, pp. 51-59.

[Reed 1978] Reed, D. *Naming and Synchronization in a Decentralized Computer System.* PhD. Diss. M.I.T., Sep. 1978.

[Relational 1984] Relational Technologies, Inc. *MicroINGRES Reference Manual.* 1984.

[Sernadas 1980] Sernadas, A. *Temporal Aspects of Logical Procedure Definition. Information Systems,* 5 (1980) pp. 167-187.

[Snodgrass 1982] Snodgrass, R. *Monitoring Distributed Systems: A Relational Approach.* PhD. Diss. Computer Science Department, Carnegie-Mellon University, Dec. 1982.

[Snodgrass 1984] Snodgrass, R. *The Temporal Query Language TQuel.* In *Proceedings of the Third ACM SIGAct-SIGMOD Symposium on Principles of Database Systems,* Waterloo, Ontario, Canada: Apr. 1984 pp. 204-212.

[Snodgrass 1985] Snodgrass, R. *A Temporal Query Language.* 1985. (Submitted to *Transactions on Database Systems.*)

[Svobodova 1981] Svobodova, L. *A reliable object-oriented data depository for a distributed computer.* In *Proceedings of 8th Symposium on Operating Systems Principles,* Dec. 1981 pp. 47-58.

[Tandem 1983] Tandem Computers, Inc. *ENFORM Reference Manual.* Cupertino, CA, 1983.

[Wiederhold et al. 1975] Wiederhold, G., J. F. Fries and S. Weyl. *Structured organization of clinical data bases.* In *Proceedings of the National Computer Conference,* AFIPS. 1975.