A New Relationship Type: The Identity Connection

Gio Wiederhold and XiaoLei Qian
Computer Science Department
Stanford University
Stanford, CA 94305, USA

Abstract
A new relationship type is proposed which permits describing the issues raised when information is distributed over autonomous workstations. The proposed identity connection identifies data which should eventually match, but which can be in differing states for some time intervals. The concept permits modeling of the systems which are now becoming common in data processing.

1. Introduction

The environment of data processing has been changing, and will continue to change [Wiederhold 85]. As the systems become more complex we have to deal with them in an abstract sense. Database models serve this function. The abstractions have to illuminate the issues facing the designers of new systems.

Current models do not deal with issues of distribution. As long as distributed systems attempted to behave as if they were centralized systems, conceptual modeling of distribution was not needed. Issues of distribution were dealt with at a lower level of abstraction, and the issues were concurrency control and recovery after partitioning of systems.

Modern data processing is moving to cooperative use of autonomous workstations. Opinions about the desirability of movement in this direction are divided, since it diminshes centralized control of corporate information resources. We will not argue here for one approach versus the other. We will present a modeling concept to improve the quality of systems which include autonomous computing nodes. Autonomy implies that no strict requirement can be made in terms of accessibility and availability, and even less of a guarantee can be given about performance of transactions at the workstation.

This paper only deals with specification and not execution issues. Concepts of reliability and performance in a distributed network are not now of our concern. We assume implicitly that requests to a remote workstation will be performed eventually. When using the concept of an identity connection to link workstations some further assumptions will be needed. A typical assumption is that requests will be performed in the order received.

2. The Structural Model

The proposed identity connection adds a fourth connection type to the structural model [Wiederhold 80]*, [Wiederhold 83]. The structural model can be seen as a formalization of

* The term identity was used differently in earlier reports

the Entity-Relationship Model [Chen 76], although the mapping is not one-to-one.

The primitives of the structural model are relations and connections. Normalized relations are used to model entities, requiring one or more relations for each entity. Connections model the relationships between entities, and any relationships created due to the normalization process. Connections do not model $m : n$ relationships, so that such relationships are modeled by two connections and intermediary relation. The intermediate relation also contains any attributes associated with the relationship.

Connections are described by the quadruple:

{`C-NAME: C-TYPE, SOURCE` $\rightarrow$ `DESTINATION, EXTRA-CONSTRAINTS`}.

The `C-NAME` identifies the relationship, the `C-TYPE` designates one of the four types, and the `SOURCE` and `DESTINATION` identify the two relations and their attributes. The attributes must have matching domains. The attribute's fields for connections are typically key fields, the sole exception is the source field of a reference connection. For every instance in the database the attributes values are equal on both sides of the connection. The `EXTRA-CONSTRAINTS` permit the attachment of quantitative limits, such as $m < 40$ for the student tuples per class.

The type of the connection not only defines the permissible cardinality of the relationship, it also defines the dynamics of the relationship, that is, how to handle insertions and deletions of tuples participating in the relationship. The three established connection types are:

Connection type	Cardinality	Symbol	Update Rule Deletion Rule
Ownership	$1 : m$	—*	Source (Owner) tuple must exist Source deletion implies Destination deletion
Reference	$n : 1$	≻—	Destination (Referent) tuple must exist Source existence prohibits Destination deletion
Subset	$1 : 1\,partial$	—∋	Source (Super) tuple must exist Source deletion implies Destination deletion

The three connections permit modeling of all concepts seen in common models [ElMasri 80]. Complex relationships may require combinations of those connections. Not represented are relationships permitting arbitrary functional relationships [Buneman 81].

However, as indicated earlier, all these kinds of connections assume that integrity constraints, as implied by those rules, are always instantaneously maintained, or at least outside of active transactions. A new connection, the *Identity Connection* will now be developed to model issues where, as expected in distributed systems, integrity maintenance is not instantaneous:

Identity	$1 : 1\,delayed$	—=	Source (Primary) tuple must be updated first Source deletion causes Destination deletion

Multiple replications require multiple connections, since their destinations may be widely distributed. Before defining the identity connection further we will give some background.

3. Technology in Distribution

In this section we will identify several technologies which enable new ways to deal with problems arising in distributed processing. One is increased parallel access provided by versions, and then we consider fragmentation, replication, and finally dynamic identification of primary copies.

Versions

Maintenance of past versions of data permits READ transactions to proceed, gathering consistent data from multiple nodes, while potentially interfering WRITE transactions are being processed [Papadimitriou 84]. While consistency of read operations is maintained, strict timelines constraints of responses is relaxed [Garcia 82]. Since in autonomous distributed systems no concept of strict global time is feasible, this relaxation has deleterious effects.

Data adequate to create past views have to be maintained. The time window is such that no inconsistencies remain in the database. In the previous definitions this time was limited by the commitment of write transactions. We will extend this limit to deal with distribution delays. This means that more past data must be retained on the node computers of the distributed network. The reduction in storage cost seen recently permits consideration of such approaches. The introduction of optical disks, which inherently keep past versions available, further increases the feasibility of version access [Rathmann 84].

Fragmentation

The logical concept of an entity or a relation does not always conform to the best physical allocation of the database. Relations are hence fragmented prior to distribution. Such a distribution is not incognizant of the user semantics, it is the expected use patterns which determine fragmentation and, subsequently, allocation [Ceri 83].

We distinguish horizontal fragmentation, based on predicates, where the fragments of a relation only share the the schema; and vertical fragmentation, where tuples share the key portion [Navathe 84]. The distinction leads to differences in update semantics as well.

Replication

Replication of data is essential to the performance of distributed systems [Katz 83]. Replication permits access to data without needing to invoke communication links. Replication can also enhance availability, permitting operation when communication links to remote nodes are broken. The existence of replication is well recognized in distributed systems, as is implied by the extensive literature on consistency management (without replication databases would always be consistent) [Bernstein 81]. Fragments as well as entire relations may be replicated.

Primary Copies

An alternative to atomic updating of all replicated copies are mechanisms using primary copy concepts. Only the primary copy is updated by the user transaction. All other copies are updated asynchronously under system control. Alternate strategies are feasible for the secondary updates. They may be updated as soon as feasible or when a read request arrives at the secondary node.

When a secondary copy is up-to-date, it can become the primary copy. Once a potential primary copy receives an update, all other copies must become secondary. A primary copy token is used to identify the current primary copy [Minoura 82].

Relaxing Synchronization

There may also be applications where adequate results can be produced without strict synchronization. Numerical analysis techniques have led the way there [Chazan 79], but there may be other transactions which can be classified to require only approximate answers [Apers 84].

4. The IDENTITY Connection

An identity connection states that all tuples participate in the connection, and that for each tuple the destination fields will eventually become equal to the source fields.

We can now identify the parameters which describe the constraints associated with the identity connections. The general format is the same as the format of any connection, as described in Section 2. The specific and extra constraints are more complex.

Symbol

The graphic symbol we propose to depict an identity connection is a line terminated with an equal sign: —= . If the direction is not determined then the terminator appears on both sides: =—= .

Semantics

The identity connection indicates two relations having replicated attributes. For any instance the connected attributes must eventually be the same. If a tuple is missing it must eventually be created.

The term *eventually* has the meaning formally defined in temporal logic, as, for instance, used in [Hailpern 82]. It is intended that formal proofs about the correctness of algorithms which support the semantics of the identity connection can be made.

Direction

The connection is directed from the source or primary relation to the destination or secondary relations. If the direction is not predetermined in the model then the connection is symmetric, although for any instance the direction should be known. This convention permits recognition of schemes as the primary copy token method.

Cardinality

The cardinality of the identity connection is 1 : 1 although temporarily, when consistency is not established, the relationship may be partial. There may be multiple secondary relations for a primary source, but for the sake of consistency with the general relationship and connection concepts we find it more convenient to model multiple replications using multiple identity connections rather than a single, complex one.

Time Constraint

A time constraint more specific than *eventually* may be specified with the identity connection. It may be given as a time limit, i.e., less than $\Delta = 1$ hour, a time point, i.e., @5pm, or as an event trigger, i.e. prior to the `write-pay-check` transaction. The event trigger should receive a name related to the name of the connection: !pay. Arbitrary transactions can then be bound to that trigger.

Attribute Constraints

The attributes participating in the identity connection always include the key attributes, but all attributes may, and frequently will, participate. More complex relationship are built by combining other structural model primitives. For instance, a partial replication requires a definition of the replicated fragment at the source, using a subset connection, and then an identity connection from the source to the destination.

Derived data can participate in the identity connection, as shown in Example 2 below.

5. Examples

Some examples will help clarify the notions.

1. In a manufacturing environment a local computer in the factory warehouse is used to maintain the inventory. For the sales force the inventory information has to be available on a daily basis. Since the inventory at the factory increases throughout the day, it is not reasonable to keep the sales computer fully up-to-date.

   A connection:

```
factory-stock:  identity,
  factory-inv.{p-no, made}——= sales-inv.{p-id, avail}, @7am;
```

indicates that the sales inventory is updated daily at 7 am.

2. The hours worked during a week are needed for generating the paychecks.

```
pay-hours:  identity,
 work.{ssn, SUM(hours | group by ssn) ——= pay.{ssn, payhours}, @Fri.5pm;
```

   Note that identity connections with aggregation on the source side must have only one direction.

3. In a distributed banking environment accounts are maintained at the local branch when active, but updated centrally at night with interbank transactions.

```
accounts:  identity,
  branch.{ac-no, amount}=—= central.{ac-no, amount}, !any(tx);
```

   A primary copy token scheme can be used to implement the constraint. The detailed logs of the transactions are maintained locally.

6. Identity Connections and Distribution

In the design of data distribution, we make clear distinction between fragmentation and replication. With the help of the identity connection, both can be easily represented within our connection framework and the concept of maintaining consistency with data distribution is explicitly incorporated in the model.

In modeling fragmented relations, we distinguish fragmentation based on predicates from fragmentation based on attributes. We can safely assume that with horizontal fragmentation based on predicates, all fragments partition the conceptual relation and there is no overlap among the predicates, because any overlap is taken care of by replication. The only information the fragments share is the schema. Stored schemas, represented by relations, can of course be related using an indentity connection.

In vertical fragmentation based on attributes, tuples in different fragments of the same relation must share the key portion. The semantics lead to the definition of one identity connection between each pair of vertical fragments on the key attributes. The connection is 1 : 1, with direction determined by the application. Arbitrary time constraints can be specified on the enforcement of the connection.

Replication can be applied to either type of fragments as well as to the whole relation. The identity connections for replication are 1 : 1 symmetric, and are defined on all attributes. If all replicated copies have to be updated as atomic actions, we can define an identity connection between each pair of the replicates and specify necessary time constraints on the connection to enforce the atomicity. If, on the other hand, a primary copy is always identified and atomicity is not required, the number of identity connections can be reduced to those between primary copy and secondary ones with weaker time constraints. The connections may occasionally change over time because the primary copy token may move to another copy.

7. Conclusion

We have provided a concept and a notation to model the replicated distribution of data at a high conceptual level. We have felt the lack of modeling concepts for distribution for some time, but could use the centralized concepts for distributed databases which were designed to satisfy centralized constraints.

Acknowledgement

Support for this research was provided in part by DARPA contract N00039-84-C-0211 for Knowledge Based Management Systems. Concepts of data distribution have been developed over many years, largely through interaction of the folk cited in the references. The immediate trigger for this paper is due to a student in my class picking up the term 'identity connection' in the textbook and wanting to know where he could read more about it.

References

[Apers 84] Peter M.G. Apers, and Gio Wiederhold: "Transaction Classification to Survive a Network Partition"; Stanford CS report STAN-CS-85-1053, August 1984.

[Buneman 81] Buneman,P. and Nikhil,R.: "A Practical Functional Programming System for Databases"; *Proc.1981 Conf.on Functional Programming Languages and Computer Architecture*, pp.195-201.

[Bernstein 81] Bernstein,P. and Goodman,N.: "Concurrency Control in Distributed Database Systems"; *ACM C. Surveys*, vol.13 no.2, Jun.1981, pp.185–221.

[Chazan 79] Chazan,D. and Miranker,W.: "Chaotic Relaxation"; in *Linear Algebra and Its Applications*, Amer. Elsevier, 1969.

[Chen 76] Chen,P.P.: "The Entity-Relationship Model - Toward a Unified View of Data"; *ACM TODS*, vol.1 No.1, Mar.1976, pp.9–36.

[Ceri 83] Ceri,S., Navathe,S. and Wiederhold,G.: "Distribution Design of Logical Database Schemas"; *IEEE TSE*, vol.SE-9 no.4, Jul.1983, pp.487–563.

[ElMasri 80] ElMasri,R. and Wiederhold,G.: "Properties of Relationships and Their Representation"; *Proc. 1980 NCC*, AFIPS vol.49, May 1980, pp.319–326.

[Garcia 82] GarciaMolina,H. and Wiederhold,G.: "Read-Only Transactions in a Distributed Database"; *ACM TODS*, vol.7 no.2, Jun.1982, pp.209–234.

[ElMasri 81] ElMasri,R. and Wiederhold,G: "Formal Specifications of GORDAS: A High-Level Query Language for Graph Databases"; *Proceedings of the Second International Conference on the Entity-Relationship Approach to System Analysis and Design*, Washington DC, Oct.12–14, 1981.

[Hailpern 82] Hailpern, Brent T.: *Verifying Concurrent Processes Using Temporal Logic*; Lecture notes in CS no.129, Springer Verlag, 1982.

[Katz 86] Katz,R.H. and Wong,E.: "Resolving Conflicts in Global Storage Design through Replication"; ACM TODS, Vol.8 No.1, Mar.1983, pp.110–135.

[Minoura 82] Minoura,T. and Wiederhold,G.: "Resilient Extended True-Copy Token Scheme for a Distributed Database"; *IEEE TSE*, vol.SE-8 no.3, May.1982, pp.173–189.

[Navathe 84] Navathe,S., Ceri,S., Wiederhold,G. and Dou,J-L.: "Vertical Partitioning for Physical and Distribution Design of Databases"; *ACM TODS*, vol.9 no.4, Dec.1984, pp.680–710.

[Papadimitriou 84] Papadimitriou,C.H. and Kanallakis,Paris C.: "On Concurrency Control by Multiple Versions"; *ACM TODS*, vol.9 no.1, Mar.1984, pp.89–99.

[Rathmann 84] Peter Rathmann: "Dynamic Data Structures on Optical Disks"; *IEEE Data Engineering Conf.*, Los Angeles CA, Apr.24–27, 1984.

[Wiederhold 80] Wiederhold, Gio and Ramez ElMasri: "The Structural Model for Database Design"; *Entity-Relationship Approach to Systems Analysis and Design*, Chen(ed), North-Holland, 1980, pp.237-257.

[Wiederhold 83] Wiederhold,G.: *Database Design*; McGraw-Hill, Computer Science Series, 2nd ed., 1983.

[Wiederhold 84] Wiederhold,Gio: "Databases"; *IEEE Computer*, vol.17 no.10, Oct.1984, pp.211-223.