# The Double Life of the Transaction Abstraction: Fundamental Principle and Evolving System Concept

Henry F. Korth

AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974
hfk@cs.utexas.edu

## Abstract

The transaction concept in computing goes back to the early days of computerized data processing. It has developed and evolved over the years both in terms of formal theory and practical application. This evolutionary process has been driven in large part by applications that require transaction-like properties. Newly emerging applications include several that involve people in a time-dependent role. The new forms of human involvement in transaction processing required by these applications are generating new systems-level challenges. Likewise, these needs present challenges and opportunities from a theoretical standpoint. This talk reviews the history of synergy between theory and practice in the area of transaction processing, and considers currently emerging needs from that perspective.

## 1 Introduction

The transaction concept in the business world predates the computer science version of the concept by thousands of years. In human interactions, concepts, such as that of a transaction, are formalized by the legal and legislative processes and enforced by a judiciary process. In the world of computing, concepts are formalized by the theory community (or theoretically-inclined systems people), and enforced by the systems built by systems engineers and programmers. Just as there is some real-world flexibility in law enforcement, so there is some flexibility in the way that real systems implement formal concepts, and, conversely, some simplification that occurs when implemented ideas are abstracted into formal concepts.

The transaction concept has an extraordinarily interesting history of synergy between its formalization and implementation. This history is interesting in its own right as a lesson in the role of theory and practice – a lesson of great importance as pressures grow for research to be "relevant," that is, justifiable from a business perspective. Furthermore, this history is of interest as the point from which the transaction concept continues to evolve – perhaps to the point where transactions are no longer *really* transactions and another term may be appropriate.

## 2 Historical Perspective

The transaction concept goes back to the dawn of civilization, and its presence in the computer science literature goes back to the early 1970s [Bjo73, Dav73]. The classic papers by members of the System R group at the IBM San Jose Research Lab ([EGLT76, GLPT75, Gra78], among others) served both to identify a simple, yet powerful, specification of the transaction concept and to describe a systems-level approach to its implementation.

The fundamental idea of a transaction is based on the four "ACID" properties: atomicity, consistency, isolation, and durability [BHG87, GR93]. Within the framework of this concept, research advanced rapidly along several fronts. Theoreticians explored the limits of the model in terms of provably correct concurrent transaction execution [Pap79, Yan82, Pap86, LMWF94]. Alternatives to the approach taken by System R for managing transactions were explored, including timestamp-based protocols [BG80, Ree83], validation techniques [KR81], and locking techniques derived from a graph-based organization of data items (such as the tree protocol of [SK80]). Concurrent with the above-cited (and other related) activities, several



**Proceedings of the 21st VLDB Conference**
**Zürich, Switzerland, 1995**

landmark database systems were being built, among them System R [CAB+81], Ingres [Sto86], and SDD-1 [BSJ80]. Extensions to transaction-management techniques for distributed computer systems were widely explored as well (see [ÖzsuV91] for details and citations).

Several things are noteworthy about this flurry of activity in transaction processing. First is that there was concurrent activity among theoreticians, implementors, and systems researchers who fit somewhere between theory and implementation. Second is that the work was done both by academia and industry, with the former well-supported by government, and the latter producing a large volume of published work. As a result, transaction processing rapidly became a well-understood field – so much so that a panel session at the 1983 VLDB conference was entitled "Concurrency Control: Are We Done With Algorithms?" [Pap83]

The death of transaction processing as a field never did occur. The success of the transaction concept in data-processing applications led to efforts to apply it elsewhere, including computer-aided design [KLMP84] and other long-duration activities, which were called "sagas" by Hector Garcia-Molina and Ken Salem [GMS87]. These issues and their relation to earlier work in transaction processing are discussed succinctly in [Gra81]. A basic theme that runs through these three papers as well as others appearing around that time is the following: The consequences of rigid adherence to the ACID properties are too draconian for these then-new applications, yet there is an appeal to the conceptual framework of the transaction abstraction that should not be entirely abandoned.

Early work kept the basic principles of the model intact, while adding nesting [Mos87, BBG89] or exploiting commutativity properties of a general set of database operations (as opposed to simply *read* and *write*) [Kor83, BR92, Wei88].

The key ingredient in applications for which the ACID properties are too strict is *interaction*. Whereas in data processing, people prepare transactions and submit them for execution, in interactive transactions, people provide input while the transaction is running. This alone is sufficient to make any transaction long-duration from a computer-system perspective. Furthermore, many of the examples (like the ubiquitous travel-reservation example) are of long-duration in human terms as well. As a result, waits imposed by concurrency control are long-duration waits, and aborts may undo large amounts of work. The waiting time is not simply that of a waiting process. It may be a person (say, a chip designer) waiting for needed data. Likewise, an abort may result in a person being told that several minutes (or hours) of work are being undone. This led to the idea that interactive transactions are really a set of nested *cooperative* activities [BKK85, KKB88].

Each instance of work inspired by the limits of the transaction model added semantics to the model and, using these added semantics, was able to avoid some of the ill effects of strict application of the ACID dogma [GM83]. Because the set of applications of extended transaction models is so broad, a huge variety of proposals have appeared in the literature. It would be too lengthy to summarize them here. A good starting point to read about several of these models (including Open Nested Transactions, the ConTract Model, the ACTA Model, and Split Transactions) is [Elm92], a collection of contributed papers. Other work includes the NT/PV Model [KS94], compensating transactions [LKS91], and ASSET [BDG+94]. The work in this area has not been only academic. As was the case earlier, there has been concurrent and synergistic development of concepts in industry and in universities. Among the noteworthy practical transaction systems are Camelot [EMS91] and Aries [MHL+92].

The ACID properties remain as a fundamental principle, but each has evolved in practice:

- Atomic $\longmapsto$ Multi-level
- Consistent $\longmapsto$ Adequate in a (application-specific) semantic sense
- Isolated $\longmapsto$ Cooperative, Negotiating
- Durable $\longmapsto$ Predictable, Explainable

Similar developments, though not necessarily involving interactivity, have arisen in multidatabase systems [BGMS92].

# 3 The Human Transaction Processor

The recent evolution of the transaction model is a shift from machine-oriented concepts to human-oriented concepts. The various applications of long-duration, interactive transactions studied in the late 1980s involved transaction-based assistance to human activities. Currently, we are witnessing a further evolution along this path, with the emergence and growing acceptance of workflow systems and groupware systems. These systems have many of the features of CAD, CASE, and similar applications, but there are several key differences.

Many of the steps in a workflow system are short, but require direct human intervention, as in approval of a form, dealing with an exception that has arisen, etc. It is thus important to present routine activities in a way that people can process them quickly and identify exceptions clearly so that they may be given due consideration. System action must be influenced by real-time considerations. The overall operation of the system may depend on the timely processing *by a person* of a request. As a result, information routing may be dependent on the time of day (e.g., routing information to an office in a time zone where it is currently normal working hours). Requests sent to a particular person may be withdrawn and rerouted if the response time is not adequate. Other requests, if not acted upon by a deadline, may become obsolete (leading either to their deletion or to some sort of compensation).

The main distinction between the kinds of activities discussed above and those studied previously is that performance terminology is applied to *people* rather than solely to machines. Indeed what is being optimized is not processor utilization or scare communication bandwidth, but rather human efficiency. In this environment, the metric of TPS, transactions per second, is not the issue. Rather the quantity to be maximized is *human information transfers per second* (HITS). "Per second" may seem wrong for human activity. Indeed for such matters as exception handling, minutes (or longer) may be more appropriate. But for routine approvals of forms, setting meeting times, and the like, the time frame is closer to a second than a minute.

The role of the computer system in such an environment is to automate where possible, present exceptions clearly, maintain audit trails of human responsibility (to allow recovery to occur at the human level as well as the machine level), and prevent tasks from becoming "lost" in cyberspace. This is not a simple matter of user interfaces. Fancy interfaces – icons, menus, use of handwriting, etc. – simplify the use of a system framework, but first, there is the issue of the framework itself. What is needed is a systems model for the human as a transaction processor – a model subject to performance study, algorithm development, correctness analysis, etc. The model would include aspects of revision-control systems and expert database systems, with significant enhancements.

An example from outside the transaction realm may illustrate more clearly why computer system concepts do not translate directly to maximization of HITS. Consider data in a database. It is painstakingly organized, indexed, checked against integrity constraints, etc. Now consider the paper databases in offices. Some are more organized than others, but even the best organization is imperfect. A person searching for information in an office will tolerate false starts and occasional failures. However, such behavior in a database system engenders dissatisfaction.

People are better at tolerating at least a moderate level of entropy, as compared with computers, but they are much worse at interrupt handling and task switching. People are more comfortable with "non-traditional" data (diagrams, handwritten data, data on paper, etc.) than with traditional database data. The state of these non-traditional data stores – including those on paper or in people's heads – is part of real-world consistency constraints. Consistency is difficult to define in this context. Even if consistency were defined perfectly, it remains necessary to test it on domains containing various non-traditional forms of data, where only approximations are possible[1].

Whereas disconnection is a mode of failure in distributed systems, people operate on "off-line" data routinely. This trend will only increase with the growth in mobile computing [IK95]. A significant challenge is providing people with a single information system environment despite the heterogeneous and sometimes-disconnected nature of the computing infrastructure. Porting old algorithms from the TPS-based world to the new problem of maximizing HITS is not likely to be a promising approach[2].

Let us consider some of the "features" of a human transaction processor and the relevance of the ACID properties. Generally speaking, all four ACID properties are valued, but they are not absolutes. Perhaps the next evolution of the ACID properties is something like the following[3]:

- Atomic $\longmapsto$ Structured, but flexible
- Consistent $\longmapsto$ Mostly consistent, with exceptions clearly noted
- Isolated $\longmapsto$ Cooperative, Negotiated
- Durable $\longmapsto$ Auditable trail of responsibility

[1] These observations suggest that the mathematical fields of statistics and continuous functions will grow in importance both for transaction processing, and for computing in general, relative to discrete mathematics.

[2] All distributed transactions on off-line data will be of long-duration and saga-like. Local transactions on off-line data must deal with such peculiarities as total failure (such as a lost or physically destroyed laptop computer), lack of processing power (paper-based data), etc.

[3] Arguably, a completely new set of properties, and not necessarily four of them, are required. The ACID framework appears to be a good starting point, though [Gra81] warns of a version of the Peter Principle: "every good idea is generalized to its level of inapplicability."

If workflow and groupware are to have their promised impact on productivity, it will come from making people more efficient at what, for people, is a natural mode of information processing. It most certainly will not come from an attempt to make people behave and think more like computers (we've tried that – it does not work).

# 4 Concluding Remarks

The history of transaction-processing research is one of great successes in theory and practice. Work in this field has had impact on huge markets and changed the way people conduct business. However, the "transaction action" is far from over. There continue to be new, financially significant applications that can benefit from a continued evolution of the transaction concept. To those actively doing research and/or development, this presents a great opportunity. For those managing or funding such activity, the historical lessons of the rapid progress that can occur from industry-university synergy are worth careful consideration.

Finally, despite the emphasis here on human-level transaction processing, it should be noted that the importance of traditional transaction processing is undiminished in this era of electronic commerce and the mass marketing of the Internet.

# References


[BBG89] C. Beeri, P. A. Bernstein, and N. Goodman. A model for concurrency in nested transaction systems. *Journal of the ACM*, 36(2):230–269, April 1989.

[BDG+94] A. Biliris, S. Dar, N. Gehani, H. V. Jagadish, and K. Ramamritham. ASSET: A system for supporting extended transactions. In *Proceedings of ACM-SIGMOD 1994 International Conference on Management of Data, Minneapolis, Minnesota*, 1994.

[BG80] P. A. Bernstein and N. Goodman. Timestamp-based algorithms for concurrency control in distributed database systems. In *Proceedings of the Sixth International Conference on Very Large Databases*, pages 285–300, 1980.

[BGMS92] Y. Breitbart, H. Garcia-Molina, and A. Silberschatz. Overview of multidatabase transaction management. *VLDB Journal*, 1(2), 1992.

[BHG87] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, Reading, MA, 1987.

[Bjo73] L. A. Bjork. Recovery scenario for a DB/DC system. In *Proceedings of the ACM Annual Conference, Atlanta*, pages 142–146, 1973.

[BKK85] F. Bancilhon, W. Kim, and H. F. Korth. A model of CAD transactions. In *Proceedings of the Eleventh International Conference on Very Large Databases, Stockholm*, pages 25–33, 1985.

[BR92] B. R. Badrinath and K. Ramamritham. Semantics-based concurrency control: Beyond commutativity. *ACM Transactions on Database Systems*, 17(1):163–199, March 1992.

[BSJ80] P. A. Bernstein, D. W. Shipman, and J. B. Rothnie Jr. Concurrency control in a system for distributed databases (SDD-1). *ACM Transactions on Database Systems*, 5(1):18–51, March 1980.

[CAB+81] D. D. Chamberlin, M. M. Astrahan, M. W. Blasgen, J. N. Gray, W. F. King, B. G. Lindsay, R. A. Lorie, J. W. Mehl, T. G. Price, P. G. Selinger, M. Schkolnick, D. R. Slutz, I. L. Traiger, B. W. Wade, and R. A. Yost. A history and evaluation of System R. *Communications of the ACM*, pages 632–646, October 1981.

[Dav73] C. T. Davies. Recovery semantics for a DB/DC system. In *Proceedings of the ACM Annual Conference, Atlanta*, pages 136– 141, 1973.

[EGLT76] K. Eswaran, J. Gray, R. Lorie, and I. Traiger. The notion of consistency and predicate locks in a database system. *Communications of the ACM*, pages 624–633, November 1976.

[Elm92] A. K. Elmagarmid, editor. *Database Transaction Models for Advanced Applications.* Morgan Kaufmann, 1992.

[EMS91] J. L. Eppinger, L. B. Mummert, and A. Z. Spector. *Camelot and Avalon: A Distributed Transaction Facility.* Morgan Kaufmann, 1991.

[GLPT75] J. N. Gray, R. A. Lorie, G. R. Putzolu, and I. L. Traiger. Granularity of locks and degrees of consistency in a shared data base. In *IFIP Working Conference on Modeling of Data Base Management Systems*, pages 1–29, 1975.

[GM83] H. Garcia-Molina. Using semantic knowledge for transaction processing in a distributed database. *ACM Transactions on Database Systems*, 8(2):186–213, June 1983.

[GMS87] H. Garcia-Molina and K. Salem. Sagas. In *Proceedings of ACM-SIGMOD 1987 International Conference on Management of Data, San Francisco*, pages 249–259, 1987.

[GR93] J. N. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann, 1993.

[Gra78] J. N. Gray. Notes on database operating systems. In *Lecture Notes in Computer Science, Operating Systems: An Advanced Course*, volume 60, pages 393–481. Springer-Verlag, Berlin, 1978.

[Gra81] J. N. Gray. The transaction concept: Virtues and limitations. In *Proceedings of the Seventh International Conference on Very Large Databases, Cannes*, pages 144–154, 1981.

[IK95] T. Imielinski and H. F. Korth, editors. *Mobile and Wireless Computing*. Kluwer, 1995. to appear.

[KKB88] H. F. Korth, W. Kim, and F. Bancilhon. On long duration CAD transactions. *Information Sciences*, 46:73–107, October 1988.

[KLMP84] W. Kim, R. Lorie, D. McNabb, and W. Plouffe. Nested transactions for engineering design databases. In *Proceedings of the Tenth International Conference on Very Large Databases, Singapore*, pages 355–362, 1984.

[Kor83] H. F. Korth. Locking primitives in a database system. *Journal of the ACM*, 30(1):55–79, January 1983.

[KR81] H. Kung and J. Robinson. On optimistic methods for concurrency control. *ACM Transactions on Database Systems*, pages 213–226, June 1981.

[KS94] H. F. Korth and G. Speegle. Formal aspects of concurrency control in long-duration transaction systems using the NT/PV model. *ACM Transactions on Database Systems*, September 1994.

[LKS91] E. Levy, H. F. Korth, and A. Silberschatz. A theory of relaxed atomicity. In *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, August 1991.

[LMWF94] N. Lynch, M. Merritt, W. Weihl, and A. Fekete. *Atomic Transactions*. Morgan Kaufmann, 1994.

[MHL+92] C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, and P. Schwarz. ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Transactions on Database Systems*, 17(1):94–162, March 1992.

[Mos87] J. E. B. Moss. Nested transactions: An introduction. In B. Bhargava, editor, *Concurrency Control and Reliability in Distributed Systems*, pages 395–425. Van Nostrand Reinhold, 1987.

[ÖzsuV91] M. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, 1991.

[Pap79] C. Papadimitriou. Serializability of concurrent database updates. *Journal of the ACM*, 26(4):631–653, October 1979.

[Pap83] C. H. Papadimitriou. Concurrency control: Are we done with algorithms? In *Proceedings of the Ninth International Conference on Very Large Databases, Florence*, 1983. Panel session.

[Pap86] C. Papadimitriou. *The Theory of Database Concurrency Control*. Computer Science Press, Rockville, Maryland, 1986.

[Ree83] D. Reed. Implementing atomic actions on decentralized data. *ACM Transactions on Computer Systems*, pages 3–23, February 1983.

[SK80] A. Silberschatz and Z. Kedem. Consistency in hierarchical database systems. *Journal of the ACM*, pages 72–80, January 1980.

[Sto86] M. Stonebraker, editor. *The Ingres Papers*. Addison Wesley, 1986.

[Wei88] W. E. Weihl. Commutativity-based concurrency control for abstract data types. *IEEE Transactions on Computers*, C-37(12):1488–1505, December 1988.

[Yan82] M. Yannakakis. Issues of correctness in database concurrency control by locking. *Journal of the ACM*, 29(3):718–740, July 1982.