

# Annotated Bibliography on Transactions

Hengfeng Wei  
hfwei@nju.edu.cn

## 1 Books

Bernstein, P. A., Hadzilacos, V., and Goodman, N. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley Longman Publishing Co., Inc., USA, 1987

Weikum, G., and Vossen, G. *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001

*It is researchers-oriented. Highly recommended.*

## 2 Transactional Consistency Models

### 2 Frameworks

Gray, J. N., Lorie, R. A., Putzolu, G. R., and Traiger, I. L. *Granularity of Locks and Degrees of Consistency in a Shared Data Base*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994, 181–208

*This paper defines four degrees of consistency, attempting to show the equivalence of locking, dependency, and anomaly-based characterizations. The anomaly definitions were too vague. The authors continue to get criticism for that aspect of the definitions. Only the more mathematical definitions in terms of histories and dependency graphs or locking have stood the test of time.*

*This is a book chapter version. Originally published in 1977. See also [24].*

Berenson, H., Bernstein, P., Gray, J., Melton, J., O'Neil, E., and O'Neil, P. A critique of ansi sql isolation levels. *SIGMOD Rec.* 24, 2 (May 1995), 1–10

*Defines Isolation Levels in terms of phenomena; Introduces new phenomena; Define Snapshot Isolation.*

Atluri, V., Bertino, E., and Jajodia, S. A theoretical formulation for degrees of isolation in databases. *Information and Software Technology* 39, 1 (1997), 47 – 53

*This paper formulates these different degrees of isolation in terms of histories, as in the case of the usual serialization theory and proposes timestamp-based protocols for different degrees of isolation.*

Adya, A., and Liskov, B. H. *Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions*. PhD thesis, USA, 1999. AAI0800775

*This thesis presents the first implementation-independent specifications of existing ANSI isolation levels and a number of levels that are widely used in commercial systems, e.g., Cursor Stability, Snapshot Isolation.*

*We use a graph-based approach to define different isolation levels in a simple and intuitive manner.*

*The thesis describes new implementation techniques for supporting different weak consistency levels in distributed client-server environments.*

Generalized isolation level definitions. In *Proceedings of the 16th International Conference on Data Engineering, ICDE '00*, IEEE Computer Society (USA, 2000), 67

*Our specifications are portable; they apply not only to locking implementations, but also to optimistic and multi-version concurrency control schemes. Furthermore, unlike earlier definitions, our new specifications handle predicates in a correct and flexible manner at all levels.*

*It also discusses “Mixing of Isolation Levels”.*

Koskinen, E., and Parkinson, M. The push/pull model of transactions. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '15*, Association for Computing Machinery (New York, NY, USA, 2015), 186–195

*We present a general theory of serializability, unifying a wide range of transactional algorithms, including some that are yet to come. To this end, we provide a compact semantics in which concurrent transactions PUSH their effects into the shared view (or UNPUSH to recall effects) and PULL the effects of potentially uncommitted concurrent transactions into their local view (or UNPULL to detangle).*

Cerone, A., Bernardi, G., and Gotsman, A. A Framework for Transactional Consistency Models with Atomic Visibility. In *26th International Conference on Concurrency Theory (CONCUR 2015)*, L. Aceto and D. de Frutos Escrig, Eds., vol. 42 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (Dagstuhl, Germany, 2015), 58–71

*we propose a framework for specifying a variety of consistency models for transactions uniformly and declaratively. Our specifications are given in the style of weak memory models, using structures of events and relations on them. The specifications are particularly concise because they exploit the property of atomic visibility guaranteed by many consistency models: either all or none of the updates by a transaction can be visible to another one. This allows the specifications to abstract from individual events inside transactions.*

Crooks, N., Pu, Y., Alvisi, L., and Clement, A. Seeing is believing: A client-centric specification of database isolation. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC '17*, Association for Computing Machinery (New York, NY, USA, 2017), 73–82

*This paper introduces the first state-based formalization of isolation guarantees.*

Crooks, N. *A Client-Centric Approach to Transactional Datastores*. PhD thesis, The University of Texas at Austin, 2019

*The PhD Thesis version of [16].*

## 2 Serializability

Weikum, G., and Vossen, G. *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001

*The “TIS” book contains several variants of serializability: Final State Serializability, View Serializability, Conflict Serializability, Commit Serializability, Multiversion Serializability, Global Serializability, Quasi Serializability*

Eswaran, K. P., Gray, J. N., Lorie, R. A., and Traiger, I. L. The notions of consistency and predicate locks in a database system. *Commun. ACM* 19, 11 (Nov. 1976), 624–633

*This paper defines the concepts of transaction, consistency and schedule and shows that consistency requires that a transaction cannot request new locks after releasing a lock. Then it is argued that a transaction needs to lock a logical rather than a physical subset of the database. These subsets may be specified by predicates. An implementation of predicate locks which satisfies the consistency condition is suggested.*

*This is the first paper to formalize mathematically the concurrency control problem. It also defines “conflict serializability”, which is termed DSR in [35].*

Papadimitriou, C. H. The serializability of concurrent database updates. *J. ACM* 26, 4 (Oct. 1979), 631–653

*It is shown that recognizing the transaction histories that are serializable is an NP-complete problem. Several efficiently recognizable subclasses are introduced.*

Kanellakis, P. C., and Papadimitriou, C. H. Is distributed locking harder? In *Proceedings of the 1st ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, PODS '82, Association for Computing Machinery (New York, NY, USA, 1982), 98–107

*We examine the problem of determining whether a set of locked transactions, accessing a distributed database, is guaranteed to produce only serializable schedules. For a pair of transactions we prove that this concurrency control problem (which is polynomially solvable for centralized databases) is in general coNP-complete.*

Yannakakis, M. Serializability by locking. *J. ACM* 31, 2 (Mar. 1984), 227–244

*It is shown that locking cannot achieve the full power of serializability. An exact characterization of the schedules that can be produced if locking is used to control concurrency is given for two versions of serializability: state serializability and view serializability.*

*See also its STOC conference version [45].*

Attar, R., Bernstein, P. A., and Goodman, N. Site initialization, recovery, and backup in a distributed database system. *IEEE Trans. Softw. Eng.* 10, 6 (Nov. 1984), 645–650

*Introduce One Copy Serializability (ISR), as a distributed/replicated counterpart of Serializability in a single-server system.*

Kanellakis, P. C., and Papadimitriou, C. H. The complexity of distributed concurrency control. *SIAM J. Comput.* 14, 1 (Feb. 1985), 52–74

*We present a formal framework for distributed databases, and we study the complexity of the concurrency control problem in this framework. Our transactions are partially ordered sets of actions, as opposed to the straight-line programs of the centralized case. The concurrency control algorithm, or scheduler, is itself a distributed program.*

Hadzilacos, T., and Papadimitriou, C. H. Algorithmic aspects of multiversion concurrency control. In *Proceedings of the Fourth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, PODS '85, Association for Computing Machinery (New York, NY, USA, 1985), 96–104

*In this paper we introduce a new notion of multiversion serializability (MVSR) based on conflicts (MVCSR), and discuss its relation with the well known single version conflict serializability (CSR).*

Ports, D. R. K., and Grittner, K. Serializable snapshot isolation in postgresql. *Proc. VLDB Endow.* 5, 12 (Aug. 2012), 1850–1861

*This paper describes our experience implementing PostgreSQL's new serializable isolation level. It is based on the recently-developed Serializable Snapshot Isolation (SSI) technique. This is the first implementation of SSI in a production database release as well as the first in a database that did not previously have a lock-based serializable isolation level.*

Raynal, M., Thia-Kime, G., and Ahamad, M. From Serializable to Causal Transactions for Collaborative Applications. Research Report RR-2802, INRIA, 1996

*It defines Causal Serializability.*

## 2 Snapshot Isolation

Elnikety, S., Zwaenepoel, W., and Pedone, F. Database replication using generalized snapshot isolation. In *Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems*, SRDS '05, IEEE Computer Society (USA, 2005), 73–84

*It defines GSI (Generalized Snapshot Isolation) and PSI (Prefix-Consistent Snapshot Isolation).*

*While (conventional) snapshot isolation requires that transactions observe the “latest” snapshot of the database, generalized snapshot isolation allows the use of “older” snapshots, facilitating a replicated implementation.*

Yabandeh, M., and Gómez Ferro, D. A critique of snapshot isolation. In *Proceedings of the 7th ACM European Conference on Computer Systems, EuroSys '12*, Association for Computing Machinery (New York, NY, USA, 2012), 155–168

*We introduce write-snapshot isolation, a novel isolation level that has a performance comparable with that of snapshot isolation, and yet provides serializability. The main insight in write-snapshot isolation is to prevent read-write conflicts in contrast to write-write conflicts that are prevented by snapshot isolation.*

Du, J., Elnikety, S., and Zwaenepoel, W. Clock-si: Snapshot isolation for partitioned data stores using loosely synchronized clocks. In *Proceedings of the 2013 IEEE 32nd International Symposium on Reliable Distributed Systems, SRDS '13*, IEEE Computer Society (USA, 2013), 173–184

*Clock-SI is a fully distributed protocol that implements snapshot isolation (SI) for partitioned data stores. It derives snapshot and commit timestamps from loosely synchronized clocks, rather than from a centralized timestamp authority as used in current systems.*

Binnig, C., Hildenbrand, S., Färber, F., Kossmann, D., Lee, J., and May, N. Distributed snapshot isolation: Global transactions pay globally, local transactions pay locally. *The VLDB Journal* 23, 6 (Dec. 2014), 987–1011

*This paper revisits the problem of implementing Snapshot Isolation in a distributed database system and makes three important contributions. First, a complete definition of Distributed Snapshot Isolation is given, thereby extending existing definitions from the literature. Based on this definition, a set of criteria is proposed to efficiently implement Snapshot Isolation in a distributed system. Second, the design space of alternative methods to implement Distributed Snapshot Isolation is presented based on this set of criteria. Third, a new approach to implement Distributed Snapshot Isolation is devised; we refer to this approach as Incremental.*

Cerone, A., and Gotsman, A. Analysing snapshot isolation. *J. ACM* 65, 2 (Jan. 2018)

*We give an alternative specification to SI that characterises it in terms of transactional dependency graphs of Adya et al., generalising serialisation graphs.*

*We then exploit our specification to obtain two kinds of static analyses. The first one checks when a set of transactions running under SI can be chopped into smaller pieces without introducing new behaviours, to improve performance. The other analysis checks whether a set of transactions running under a weakening of SI behaves the same as when running under SI.*

## 2 Mixed Transactional Consistency Models

Fekete, A. Allocating isolation levels to transactions. In *Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '05, Association for Computing Machinery (New York, NY, USA, 2005), 206–215

*In this paper, we discuss the problem of taking a collection of transactions, and allocating each to run at an appropriate isolation level (and thus use a particular concurrency control mechanism), while still ensuring that every execution will be conflict serializable. When each transaction can use either S2PL, or snapshot isolation, we characterize exactly the acceptable allocations, and provide a simple graph-based algorithm which determines the weakest acceptable allocation.*



Milano, M., and Myers, A. C. Mixt: A language for mixing consistency in geodistributed transactions. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2018, Association for Computing Machinery (New York, NY, USA, 2018), 226–241

*To manipulate both weakly and strongly consistent data in a single transaction, we introduce a new abstraction: mixed-consistency transactions, embodied in a new embedded language, MixT. Programmers explicitly associate consistency models with remote storage sites; each atomic, isolated transaction can access a mixture of data with different consistency models.*

de Régil Basáñez, B. A. A fast implementation of parallel snapshot isolation. Master’s thesis, Universidad Complutense de Madrid, 2020

*This work shows an approach to bridge the gap between PSI, NMSI and strong consistency models like serialisability. It introduces and implements fastPSI, a consistency protocol that allows the user to selectively enforce serialisability for certain executions, while retaining the scalability properties of weaker consistency models like PSI and NMSI.*

### 3 Robustness (and Dependency Graphs)

Fekete, A., Liarakapis, D., O’Neil, E., O’Neil, P., and Shasha, D. Making snapshot isolation serializable. *ACM Trans. Database Syst.* 30, 2 (June 2005), 492–528

*This article develops a theory that characterizes when nonserializable executions of applications can occur under SI.*

Beillahi, S. M., Bouajjani, A., and Enea, C. Robustness Against Transactional Causal Consistency. In *30th International Conference on Concurrency Theory (CONCUR 2019)*, W. Fokkink and R. van Glabbeek, Eds., vol. 140 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (Dagstuhl, Germany, 2019), 30:1–30:18

*In this paper, we investigate application-specific relationships between several variations of causal consistency and we address the issue of verifying automatically if a given transactional program is robust against causal consistency, i.e., all its behaviors when executed over an arbitrary causally consistent database are serializable.*

## 4 Concurrency Control Protocols

### 4 Theory

Kung, H. T., and Papadimitriou, C. H. An optimality theory of concurrency control for databases. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, SIGMOD '79, Association for Computing Machinery (New York, NY, USA, 1979), 116–126

*There is a growing body of literature on various solutions to the concurrency control problem. This paper gives a uniform framework for evaluating these solutions, and, in many cases, for establishing their optimality. We point out a trade-off between the performance of a scheduler and the information that it uses. We show that most of the existing work on concurrency control is concerned with specific points of this fundamental trade-off. For example, our framework allows us to formally show that the popular approach of Serialization is the best one can hope for when only syntactic information is available.*

### 4 Overview

Bernstein, P. A., and Goodman, N. Concurrency control in distributed database systems. *ACM Comput. Surv.* 13, 2 (June 1981), 185–221

*In this paper we survey, consolidate, and present the state of the art in distributed database concurrency control. The heart of our analysis is a decomposition of the concurrency control problem into two major subproblems: read-write and write-write synchronization. We describe a series of synchronization techniques for solving each subproblem and show how to combine these techniques into algorithms for solving the entire concurrency control problem. Such algorithms are called “concurrency control methods.” We describe 48 principal methods, including all practical algorithms that have appeared in the literature*

*plus several new ones. We concentrate on the structure and correctness of concurrency control algorithms. Issues of performance are given only secondary treatment.*

## 4 Locking

Gray, J. N., Lorie, R. A., and Putzolu, G. R. Granularity of locks in a shared data base. In *Proceedings of the 1st International Conference on Very Large Data Bases, VLDB '75*, Association for Computing Machinery (New York, NY, USA, 1975), 428–451

*This paper proposes a locking protocol which associates locks with sets of resources. This protocol allows simultaneous locking at various granularities by different transactions. It is based on the introduction of additional lock modes besides the conventional share mode and exclusive mode. The protocol is generalized from simple hierarchies of locks to directed acyclic graphs of locks and to dynamic graphs of locks. The issues of scheduling and granting conflicting requests for the same resource are then discussed. Lastly, these ideas are compared with the lock mechanisms provided by existing data management systems.*

*It introduces intention locks.*

Yannakakis, M. A theory of safe locking policies in database systems. *J. ACM* 29, 3 (July 1982), 718–740

*Necessary and sufficient conditions are found for a locking policy to be safe, but it is shown that in general it is NP-complete to test for these conditions. However, when the database has a given structure, a simple set of rules which is sufficient for safety and, moreover, necessary for a wide class of natural locking policies is developed.*

Buckley, G. N., and Silberschatz, A. Beyond two-phase locking. *J. ACM* 32, 2 (Apr. 1985), 314–343

*Graph protocols.*

Thomasian, A. Concurrency control: Methods, performance, and analysis. *ACM Comput. Surv.* 30, 1 (Mar. 1998), 70–119

*This tutorial reviews CC methods based on standard locking, restart-oriented locking methods, two-phase processing methods including optimistic CC, and hybrid methods (combining optimistic CC and locking) in centralized systems.*

## 4 Atomic Commitment and Consensus

Rao, J., Shekita, E. J., and Tata, S. Using paxos to build a scalable, consistent, and highly available datastore. *Proc. VLDB Endow.* 4, 4 (Jan. 2011), 243–254

*Spinnaker is an experimental datastore that is designed to run on a large cluster of commodity servers in a single data center. It features key-based range partitioning, 3-way replication, and a transactional get-put API with the option to choose either strong or timeline consistency on reads. This paper describes Spinnaker’s Paxos-based replication protocol.*

Mahmoud, H., Nawab, F., Pucher, A., Agrawal, D., and El Abbadi, A. Low-latency multi-datacenter databases using replicated commit. *Proc. VLDB Endow.* 6, 9 (July 2013), 661–672

*Spanner uses Two-Phase Commit and Two-Phase Locking to provide atomicity and isolation for globally distributed data, running on top of Paxos to provide fault-tolerant log replication. We show in this paper that it is possible to provide the same ACID transactional guarantees for multi-datacenter databases with fewer cross-datacenter communication trips, compared to replicated logging. Instead of replicating the transactional log, we replicate the commit operation itself, by running Two-Phase Commit multiple times in different datacenters and using Paxos to reach consensus among datacenters as to whether the transaction should commit.*

Chockler, G., and Gotsman, A. Multi-Shot Distributed Transaction Commit. In *32nd International Symposium on Distributed Computing (DISC 2018)*, U. Schmid and J. Widder, Eds., vol. 121 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (Dagstuhl, Germany, 2018), 14:1–14:18

*We introduce Transaction Certification Service (TCS), a new formal problem that captures safety guarantees of multi-shot transaction commit protocols with integrated concurrency control. TCS is parameterized by a certification function that can be instantiated to support common isolation levels, such as serializability and snapshot isolation. We then derive a provably correct crash-resilient protocol for implementing TCS through successive refinement.*

Maiyya, S., Nawab, F., Agrawal, D., and Abbadi, A. E. Unifying consensus and atomic commitment for effective cloud data management. *Proc. VLDB Endow.* 12, 5 (Jan. 2019), 611–623 Data storage in the Cloud needs to be scalable and fault-tolerant. Atomic commitment protocols such as Two Phase Commit (2PC) provide ACID guarantees for transactional access to sharded data and help in achieving scalability. Whereas consensus protocols such as Paxos consistently replicate data across different servers and provide fault tolerance. Cloud based datacenters today typically treat the problems of scalability and fault-tolerance disjointedly. In this work, we propose a unification of these two different paradigms into one framework called Consensus and Commitment (C&C) framework. The C&C framework can model existing and well known data management protocols as well as propose new ones.

## 4 Implementations

Didona, D., Guerraoui, R., Wang, J., and Zwaenepoel, W. Causal consistency and latency optimality: Friend or foe? *Proc. VLDB Endow.* 11, 11 (July 2018), 1618–1632

*In this paper, we show that such “latency-optimal” ROTs induce an extra overhead on*

*writes that is so high that it actually jeopardizes performance even in read-dominated workloads. We show this result from a practical as well as from a theoretical angle.*

Wu, C., Sreekanti, V., and Hellerstein, J. M. Transactional causal consistency for serverless computing. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, Association for Computing Machinery (New York, NY, USA, 2020), 83–97

*This raises the challenge of Multisite Transactional Causal Consistency (MTCC): the ability to provide causal consistency for all I/Os within a given transaction even if it runs across multiple physical sites. We present protocols for MTCC implemented in a system called HYDROCACHE.*

## 5 Formal Methods

Pollak, D. H. Reasoning about two-phase locking concurrency control. Technical report, June 2017

*We present a program logic for serializable transactions that are able to manipulate a shared storage.*

*We show this by providing the first application of our logic in terms of the Two-phase locking (2pl) protocol which ensures serializability.*

Kaki, G., Nagar, K., Najafzadeh, M., and Jagannathan, S. Alone together: Compositional reasoning and inference for weak isolation. *Proc. ACM Program. Lang.* 2, POPL (Dec. 2017)

*Unfortunately, the semantics of weak isolation is poorly understood, and usually explained only informally in terms of low-level implementation artifacts. Consequently, verifying high-level correctness properties in such environments remains a challenging problem. To address this issue, we present a novel program logic that enables compositional reasoning about the behavior of concurrently executing weakly-isolated transactions.*

---

## 6 Systems

## 7 Testing

Rahmani, K., Nagar, K., Delaware, B., and Jagannathan, S. Clotho: Directed test generation for weakly consistent database systems. *Proc. ACM Program. Lang.* 3, OOPSLA (Oct. 2019)

*This paper presents a novel testing framework for detecting serializability violations in (SQL) database-backed Java applications executing on weakly-consistent storage systems. We manifest our approach in a tool, CLOTHO, that combines a static analyzer and model checker to generate abstract executions, discover serializability violations in these executions, and translate them back into concrete test inputs suitable for deployment in a test environment.*

Biswas, R., and Enea, C. On the complexity of checking transactional consistency. *Proc. ACM Program. Lang.* 3, OOPSLA (Oct. 2019)

*In this work, we investigate the problem of checking whether a given execution of a transactional database adheres to some consistency model. We show that consistency models like read committed, read atomic, and causal consistency are polynomial-time checkable while prefix consistency and snapshot isolation are NP-complete in general.*