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

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.

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 an-

other 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 [11].

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 is the first paper to formalize mathematically the concurrency control problem. It also defines "conflict serializability", which is termed DSR in [19].

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 [27].

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 (1SR), 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.

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

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.

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 (Perfix-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.

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.

3 Theory

4 Robustness (and Dependency Graphs)

Fekete, A., Liarokapis, 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.

5 Concurrency Control Protocols

5 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 analysts 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 m 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.

5 Locking

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.

6 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.

7 Systems

8 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.