Revising 1-Copy Equivalence in Replicated Databases with Snapshot Isolation

F. D. Muñoz[1], J. M. Bernabé[1], R. de Juan[1], J. E. Armendáriz[2], J. R. González de Mendívil[2]

[1]Instituto Tecnológico de Informática
Univ. Politécnica de Valencia
Camino de Vera, s/n
46022 Valencia, Spain

[2]Depto. de Ing. Matemática e Informática
Univ. Pública de Navarra
Campus de Arrosadía, s/n
31006 Pamplona, Spain

{fmunyoz, jbgisber, rjuan}@iti.upv.es {enrique.armendariz, mendivil}@unavarra.es

Technical Report ITI-SIDI-2009/003

Revising 1-Copy Equivalence in Replicated Databases with Snapshot Isolation

F. D. Muñoz[1], J. M. Bernabé[1], R. de Juan[1], J. E. Armendáriz[2], J. R. González de Mendívil[2]

[1]Instituto Tecnológico de Informática
Univ. Politécnica de Valencia
Camino de Vera, s/n
46022 Valencia, Spain

[2]Depto. de Ing. Matemática e Informática
Univ. Pública de Navarra
Campus de Arrosadía, s/n
31006 Pamplona, Spain

Technical Report ITI-SIDI-2009/003

e-mail: {fmunyoz, jbgisber, rjuan}@iti.upv.es {enrique.armendariz, mendivil}@unavarra.es

June 5, 2009

Abstract

Multiple database replication protocols have used replicas supporting the snapshot isolation level. They have provided some kind of one-copy equivalence, but such concept was initially conceived for serializable databases. In the snapshot isolation case, due to its reliance on multi-versioned concurrency control that never blocks read accesses, such one-copy equivalence admits two different variants. The first one consists in relying on sequential replica consistency, but it does not guarantee that the snapshot used by each transaction holds the updates of the last committed transactions in the whole replicated system, but only those of the last locally committed transaction. Thus, a single user might see inconsistent results when two of her transactions have been served by different delegate replicas: the updates of the first one might not be in the snapshot of the second. The second variant avoids such problem, but demands atomic replica consistency, blocking the start (i.e., in many cases, read accesses) of new transactions. Several protocols of each kind exist nowadays, and most of them have given different names to their intended correctness criterion. We survey such previous works and propose uniform names to these criteria, justifying some of their properties.

1 Introduction

Consistency has been thoroughly studied in parallel and distributed systems, mainly in those with shared memory, generating a set of consistency models [22, 1]. A replicated database can be considered as an example of such kind of systems, since all replicas hold copies of the same data that should be kept consistent, building thus a specialized kind of logical shared memory, although commonly implemented in a shared-nothing set of nodes. *One-copy serializability* [7] (a.k.a.*1SR*) has been the commonly accepted correctness criterion for replicated databases, since it was enough relaxed for ensuring good performance and strict-enough for guaranteeing a comfortable replica consistency model for the application programmer. But this 1SR single concept encompasses two different issues. First, the *isolation level* being responsible for the isolation consistency among all concurrent transactions being executed in the system and, second, the *replica consistency*; i.e., the degree of admissible divergence among the states of all replicas.

In the last twenty years several things have changed that yield some opportunities for revising such issues. Regarding the first one (isolation), new levels have been defined [5, 2], being *Snapshot Isolation* (SI) one of the most important, since it is quite close to the serializable level and it does not need to block read accesses due to its reliance on multi-versioned concurrency control mechanisms. Multiple DBMSs

(e.g., Oracle, PostgreSQL, MS SQL Server,...) support the SI level, and even some of them label it as serializable although there are some isolation anomalies [5] that can not be avoided with SI. Despite this, with some care [12] a DBMS supporting SI is able to ensure serializable isolation. Considering the second issue (replica consistency), 1SR is considered [22] equivalent to a *sequential* [20] consistency model, but some applications might require either stricter models like the *linearizability* semantics proposed in [17](or *atomic* memory model; we will use this latter term in the sequel, since linearizability is a correctness condition for objects that enforces a non-blocking behavior and such characteristic might not be achieved in a database replication system. Some papers, e.g. [9], refer to both concepts as synonyms when they are applied to *distributed shared memory*, or DSM, systems.) –providing thus an easier programming model– or more relaxed models like the *cache* [14] one, improving thus system performance. So, it is interesting to analyze how such other replica consistency models could be combined with snapshot isolation in order to generate new *isolation+replica consistency* models for replicated databases.

Thus, the contributions of this paper consist in surveying different combinations between the snapshot isolation level and replica consistency models (adding cache and atomic semantics to the traditionally accepted sequential consistency). Note that such combinations have been already used in previous works (e.g., both the *Strong SI* level in [10] and the *Conventional SI* of [11] can be considered a combination between the SI isolation level and the atomic replica consistency model), but using different names in order to refer to the same things. As a result of this survey, we provide a new taxonomy of combined consistency models as our second contribution. As a final contribution we show that this new taxonomy is able to justify some protocol properties that are difficult to prove otherwise.

The rest of this paper is structured as follows. Section 2 describes the assumed system model, integrating transactions –and, as a result, isolation– in the traditional replica consistency models. Section 3 summarizes the isolation and replica consistency models commonly assumed in modern database replication protocols. Later, Section 4 presents a new taxonomy of models that combine both isolation and replica consistencies. Section 5 shows that such taxonomy easily justifies some unproven properties of modern database replication protocols. Finally, Section 6 concludes the paper.

2 System Model

We consider a distributed system composed by a set N of nodes, interconnected by a network. Each system node has a local DBMS able to manage the *snapshot isolation* (SI) level. Replication is being managed by a middleware layer using some *Read-One Write-All-Available* (ROWAA) [7] replication protocol that should take care of guaranteeing the intended global isolation level and of ensuring some replica consistency model. Modern replication protocols [31] are based on executing transactions in a single delegate replica and propagating later the transaction updates in FIFO total order to all other replicas in the commit procedure. We assume such behavior in this paper.

Replica consistency models could be any of the traditional DSM ones. It is worth noting that these latter models have usually been specified considering as relevant events both *read* and *write* operations applied to a given set of variables shared among multiple processes. Such specifications cannot be trivially migrated to a system where the reads and updates are made in the context of transactions, since such transactions encompass multiple individual operations of that traditional kind. So, in order to discuss such replica consistency models, we need to adapt the transaction concept to the equivalent sequence of read and write operations. This is feasible when a strict-enough isolation level is being assumed.

Thus, we will transform transactions into sequences of operations in the following way: each transaction read access will be logically advanced till the transaction start time, sharing all read operations the same logical time, whilst all update accesses will be logically put at the transaction commit time, as a single multi-variable write operation. Note that only database accesses are being considered here. So, once an update on a given item has been made, the application program will be able to know which will be that item new value, e.g., using local variables to this end, without requiring another physical read access on the database item. If any of such read accesses is being made in a given transaction, it will be eliminated in the resulting mapping. As a consequence, such a reordering of read and write events can be made on all transactions. Note also that only committed transactions need to be considered, since aborted ones do not have any effect on the database state once they have terminated.

3 Isolation and Replica Consistency

Although there are multiple isolation levels [5, 2] supported by modern DBMSs, only few of them have deserved attention in order to implement database replication protocols. Concretely, such isolation levels are the *serializable* [7] and the *snapshot* [5] ones. Other more relaxed levels have not been widely considered in research papers. In the best cases, they have been included [5, 2] in order to carefully specify such levels or for supporting them [6] in order to provide replication transparency.

A similar scenario can be found regarding replica-consistency models. Although there are many models that could be considered [22, 1], only a few of them (cache, sequential and atomic) have been assumed in database replication. For instance, strict consistency levels (e.g., the atomic one) ensure consistency in some kinds of applications where a single client is able to access different copies of a given item in a short interval, like in three-tiered web applications [24]. In such context, without atomic consistency, the last client request might be executed before the effects of previous requests were applied in its serving replica. So, clients could get inconsistent replies. On the other hand, new data management trends [13, 16] (e.g., in the *cloud computing* field) suggest that temporary inconsistencies should be afforded by modern applications, and that this should be considered a strong requirement when scalability is a must. So, they advocate for relaxed consistency models based on asynchronous update propagation. As a result, these two opposite trends show that multiple replica consistency models need to be considered.

Complete specifications of such isolation levels and replica-consistency models can be found in [2] and [9], respectively. We will summarize them in the following two sections.

3.1 Isolation Levels

In order to specify an isolation level, most works [7, 5, 2] have used *transaction histories* composed of a partial order of transactions' events and a total order on the committed item versions generated by such transactions. Taking such histories as a base, a *dependency* [5] or *serialization* [2] *graph* is built, using transactions as its nodes and transaction dependencies as its edges. Several isolation phenomena are specified (describing which set of dependencies must exist in the serialization graph for each kind of phenomena). Finally, an isolation level is respected when each considered transaction avoids a given subset of phenomena.

Thus, following [2]'s conventions, the possible transaction dependencies ($T_i \rightarrow T_j$) are:

- T_j *directly write-depends on* T_i ($T_i \xrightarrow{ww} T_j$) when T_i installs X_i and T_j installs X's next version.
- T_j *directly read-depends on* T_i ($T_i \xrightarrow{wr} T_j$) when T_i installs X_i, T_j reads X_i or T_j performs a predicate-based read, X_i changes the matches of T_j's read, and X_i is the same or an earlier version of X in T_j's read.
- T_j *directly anti-depends on* T_i ($T_i \xrightarrow{rw} T_j$) when T_i reads X_h and T_j installs X's next version or T_i performs a predicate-based read and T_j overwrites this read.
- T_j *start-depends on* T_i ($T_i \xrightarrow{s} T_j$) when $c_i < s_j$; i.e., when it starts after T_i commits. When start dependencies are considered, the resulting graph is named a *start serialization graph* (SSG(H)).

And the phenomena to be considered in order to specify SI are:

- *G1a: Aborted Reads.* A history H shows phenomenon G1a if it contains an aborted transaction T_1 and a committed transaction T_2 such that T_2 has read some object modified by T_1.
- *G1b: Intermediate Reads.* A history H shows phenomenon G1b if it contains a committed transaction T_2 that has read a version of object X written by transaction T_1 that was not T_1's final modification of X.
- *G1c: Circular Information Flow.* A history H exhibits phenomenon G1c if its serialization graph contains a directed cycle consisting entirely of dependency (i.e., write-dependencies or read-dependencies) edges.

- *G-SIa: Interference.* A history H exhibits phenomenon G-SIa if SSG(H) contains a read/write-dependency edge from T_i to T_j without there also being a start-dependency edge from T_i to T_j.
- *G-SIb: Missed Effects.* A history H exhibits phenomenon G-SIb if SSG(H) contains a directed cycle with exactly one anti-dependency edge.

A history respects the snapshot isolation level when G1a, G1b, G1c, G-SIa and G-SIb phenomena are proscribed. This has been ensured in stand-alone database systems using multi-versioned concurrency control, combined in some cases with write locks.

3.1.1 Transaction Validation Rules

In a replicated setting, a database replication protocol is needed. As already outlined above (in Section 2), most of these protocols execute initially the transactions in a single delegate node, collecting their writeset when commit is being requested (but before processing such request) and multicasting it to all replicas (usually, in FIFO total order) [3, 26]. Once such updates are delivered at their target nodes, a validation stage is executed. If transactions overcome such validation, they are applied in the underlying database. To this end, the concurrency control mechanisms commented for stand-alone databases could still be useful; i.e., they may provide information about conflicts between the transaction whose writeset is being applied (and that should be committed) and other in-course transactions that might block such writeset application. If so arises, such local in-course transactions are aborted.

Using this approach, the replication protocol needs only be concerned with the validation rules followed in order to guarantee an isolation level. Thus, the validation rules for snapshot isolation [19, 11, 21] consist in detecting write-write conflicts between the transaction being validated and other already-committed concurrent transactions (considering concurrent those pairs of transactions that do not have a start dependency). To this end, all replication protocol classes need to propagate transaction writesets and the logical transaction start timestamps. Thus, phenomena G1a, G1b and G-SIa are avoided by the local concurrency control mechanisms used in each replica, whilst the validation rule prevents phenomena G1c and G-SIb from appearing, since no cycle of dependencies is allowed by such rule.

3.2 Consistency Models

The three replica-consistency models that have been used in common database replication protocols have been informally specified [22] as follows:

- *Atomic consistency*: Operations take effect at some point in an operation interval. Such intervals divide time into non-overlapping consecutive slots. This implies that when a process has read a given value (or version) of a concrete item, no other process could read afterward any of such item's previous versions.

  We do not demand a strict compliance to the atomic consistency semantics in this paper since they are difficult to achieve in a practical deployment, but at least that the following property is guaranteed avoiding thus the problems mentioned in [24]: *If a single client forwards a transaction t_a to a replica r_i and gets the result of t_a, any other transaction t_b sent later by this same client to any other replica r_j should be able to read the updates caused by t_a,* assuming that no other transaction is submitted to the system between t_a and t_b.

- *Sequential consistency*: This model was defined in [20] as follows: *The result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program.* So, it can be implemented using FIFO total order for applying all write operations in system replicas.

  Note that this does not avoid the problem outlined in [24], since sequential consistency ensures that all updates will be applied following the same sequence in all replicas. However, if replica r_j is overloaded and holds a long queue of pending updates (to be applied in the database), it might serve the first read accesses of t_b before applying the updates of t_a and, of course, before locally committing t_a.

- *Cache consistency* [14]: This model only requires that accesses are sequentially consistent on a *per-item* basis.

  There are some replication protocols (as listed in Section 4.1) that are able to comply with the requirements of this model but provide a consistency slightly higher, but that does not correspond to any already specified model. Such protocols are based on total order update propagation, but they allow that writeset application breaks such total order when writesets do not conflict (i.e., there are no write-write conflicts) with any of the previously-delivered but not-yet-committed transactions. Note that this ensures a per-item sequential consistency (as requested in the *cache* model), but also a per-transaction-writeset consistency (i.e., we can not commit half of a writeset WS_A before writeset WS_B and the other half of WS_A afterward), although not a complete sequential consistency.

Note that some authors [15] only admit the *atomic* and *sequential* consistency models when talking about 1-copy-consistency. So, in such context, a *cache* system is not allowed. However, we include it in this paper in order to state that other models more relaxed than the sequential one could make sense if performance is a must.

4 One-Copy Equivalence

One-Copy Equivalence was introduced in [7], tailored for the serializable isolation level, as a correctness criterion for replicated databases. Its aim is to ensure that the interleaved execution of multiple transactions in a replicated database system were equivalent to a unique (and serial, for that isolation level) execution in a logical single computer.

Our proposal consists in the extension of such equivalence concept to other isolation levels, considering also different replica-consistency models. There have been many proposals of this kind, but they were focused in a single isolation+replica consistency combination. For instance, we can find multiple definitions [11, 21, 10] of what should be understood as 1C-SI, and none of them does exactly match the others. So, it seems appropriate to carefully state how such different kinds of consistency (isolation-related and replica-related) can be merged, giving neutral names to the resulting combinations, in order to promote their acceptance. Our proposal explicitly states which is the actual combination of replica-consistency model and isolation level when a *one-copy equivalence* is being provided. If we only consider the currently existing SI database replication proposals, the resulting models are summarized in Table 1. Note that modern database replication approaches are based on FIFO total order update propagation and application into the replicas. This naturally provides a sequential replica-consistency model. So, we do not consider mandatory to specify such replica-consistency model when stating a one-copy equivalence model, being it the default one.

Table 1: One-Copy SI Equivalence Models

Replica-Consistency Model		
Cache	Sequential	Atomic
1C-Cache-SI	1C-SI	1C-Atomic-SI

In Sections 4.1 through 4.3, we survey some of the existing database replication protocols that support each one of such isolation+consistency models, presenting the name they associate to such combination of isolation and consistency. The aim of these sections is to illustrate that, in some cases, a given combination has received different names in different papers. So, it seems convenient to propose and promote a standard name for such models.

4.1 1C-Cache-SI Model

Lin et al. [21] proposed a protocol named SRCA-Opt that implements the 1C-Cache-SI model. Note however that such protocol is not proposed as the main contribution of such paper. Indeed, its aim is to

propose a valid 1C-SI criterion (that follows also our assumptions and ensures 1C-Sequential-SI) and a protocol that implements such criterion (its SRCA-Rep one). In its performance evaluation section, they propose a variant of such SRCA-Rep protocol with relaxed replica consistency, and the authors are aware of that issue. One of the aims of such evaluation is to compare how these two kinds of replica consistency (sequential and cache) are able to cope with increasing workloads. Their results show that with relaxed replica consistency, and for the benchmark used in such paper with an update-intensive workload, SRCA-Opt (i.e., the protocol with cache consistency) always provides shorter transaction completion times than SRCA-Rep (i.e., the variant with sequential consistency), and such differences increase with the load.

Note, however, that in such paper no reference is given to the kind of replica consistency being guaranteed by such SRCA-Opt protocol. Indeed, a single comment is given referring to the relaxation provided:

> *However, in update intensive workloads, SRCA-Opt might be a better alternative even it does not provide full 1-copy-SI. This might be comparable with approaches in centralized systems where at high workloads lower levels of isolation are chosen (e.g., READ COMMITTED) to speed up performance.*

And, it simply compares such performance improvement with that achievable by relaxed isolation levels; i.e., it indirectly says that, among others, there are two ways for improving performance: to relax replica consistency and to relax isolation. This confirms that both kinds of consistency (isolation-related and replica-related) should be considered in order to specify a *one-copy equivalence* model.

4.2 1C-(Sequential-)SI Model

The first SI replication protocols [19, 21, 11] were based on sequential replica consistency, since such consistency model was already widely accepted for the serializable isolation level as part of its 1SR correctness criterion. Such protocols used the ROWAA approach and propagated transaction updates in a single total-order broadcast, eliminating the need of a traditional distributed commit protocol (either two-phase or three-phase), following the principles of the last protocol variant suggested by [3], that was proven correct for implementing 1SR in [26].

Lin et al. [21] have the merit of being the first providing a sequential specification for SI replication, and naming it as 1C-SI (as suggested in the current paper), based on the 1SR one given in [7]. Moreover, in that same paper they propose a 1C-SI protocol named SRCA-Rep and analyze its performance, as we have discussed above. However, they did not explicitly state in any part of their specification that the replica consistency being assumed was the sequential one.

On the other hand, [11] was the first paper that discussed the differences between the guarantees being provided by common SI replication protocols and those provided by stand-alone SI databases, since the notion of *latest snapshot* is different in those environments. Thus, they identify two SI variants:

- *Conventional Snapshot Isolation* (CSI) refers to stand-alone implementations, where it is trivial to guarantee that such *latest snapshot* being used by every starting transaction corresponds to the one generated by all previously committed transactions. Elnikety et al. [11] do not recommend CSI for replicated settings, since this might require blocking the start of new transactions in order to guarantee that their delegate replicas receive the updates of all previously committed transactions.

- *Generalized Snapshot Isolation* (GSI) refers to the common replicated scenario guaranteeing sequential replica consistency. In it, the delegate replica where a transaction is executed does not need to maintain all the updates of all previously committed transactions. This may happen when such updates are still in transit or buffered in such receiving replica but not yet applied. So, the *latest snapshot* for a given transaction will be that of its delegate serving replica, but such latest snapshot does not hold all the updates generated by all transactions committed in the system.

Thus, [11] proposed GSI as a new concept, distinguishing it from CSI, but such paper did not state that their differences were derived from different consistency models (sequential for GSI and atomic for CSI).

Similar differences were identified by [10]. However, such paper proposed other names in order to refer to similar concepts. Thus, its *Global Weak Snapshot Isolation* refers to our 1C-SI model, whilst

Global Strong Snapshot Isolation refers to our 1C-Atomic-SI model. Its authors carefully identify the main problems of both models: usage of past snapshots in 1C-SI and expensive/blocking protocols in 1C-Atomic-SI, agreeing with the proposal of [11]. In order to solve such problems, [10] proposes an intermediate model: *Strong Session Snapshot Isolation*, that does only require *strong SI* semantics among the transactions of a given session, whilst transactions belonging to different sessions do only require *weak SI* semantics. A session holds the transactions being generated by a given client.

Wu and Kemme [32] provide a thorough description of the database concurrency control needed in order to implement SI in PostgreSQL. They propose a database replication protocol embedded into the DBMS core, guaranteeing thus a very good performance since transaction validation in the replication protocol can delegate many of its functions to the original DBMS concurrency control. The paper also provides a detailed justification about how replica consistency is being guaranteed by such replication protocol, although it does not mention that the resulting consistency is sequential.

A similar principle was used in [23]; i.e., to delegate conflict evaluation in the certification step to the underlying DBMS, but implementing the protocol in a middleware layer, enhancing thus the portability and maintainability of the resulting protocol, without severely compromising its performance.

In [31], a classification of modern database replication protocols based on total order was given. There are four main classes: active, passive, certification-based, and weak voting. In the *active* one, all transaction operations need to be delivered to all replicas before such transaction execution is started. Once this is done, each replica is able to directly execute such sequence of operations and to commit or abort locally such transaction without further interaction with other replicas. To this end, the transaction logic should be deterministic. In the *passive* variant, all transactions are completely executed in a single primary replica that propagates the transactions' updates at commit time to all other backup replicas. In the *certification-based* class, a transaction is executed in a single delegate replica (but different transactions may select different delegates). Once the transaction requests its commit, its writeset is collected and multicast to all replicas in total order. At delivery time, such writeset is certified against all other delivered and concurrent writesets (i.e., those belonging to transactions that committed while the transaction being certified was executed). If a write-write conflict is found, the transaction being certified is aborted. Otherwise, it is committed. Note that all replicas hold the same historic list of previously delivered writesets. So, they are able to certify each transaction without exchanging more messages with other replicas. Finally, in the *weak voting* variant, transactions are served by delegate replicas like in the *certification-based* class, and their writesets are multicast to all other replicas at commit time, but the conflict evaluation is only done in the delegate replica. In this case, no historic list of previously delivered writesets is needed. This protocol family is able to check for conflicts when each one of the remote transactions accepted by the protocol is being committed in each replica. If such commit is blocked by a local transaction that maintains a write lock on any of the updated items, such local transaction is immediately aborted. So, when the writeset of a transaction t_i is delivered in its delegate replica, if t_i has not been aborted by any previously committed remote transaction, its delegate replica will reliably broadcast a $commit(t_i)$ message to all replicas; otherwise, it broadcasts an $abort(t_i)$ one. All replicas act according to such final message in order to determine the fate of t_i.

Most of the protocols cited up to now belong to the *certification-based* class [19, 21, 11, 32, 23], since it does not demand a second broadcast in order to certify a transaction. According to [31], both *certification-based* and *weak voting* classes are able to provide the best transaction completion time. The *weak voting* class has the advantage of removing the need of a historic list of delivered writesets for certifying transactions. Note that such list could demand a lot of memory in case of dealing with long transactions, although this seldom arises. However, it introduces the problem of needing two separate broadcasts for managing each transaction. Despite this, its usage has been considered in some papers. Thus, the voting protocol of [28] is a sample of this class and it ensures 1C-SI, since the *uniform data store* assumed in such paper was implemented using PostgreSQL, whilst its non-voting protocol belongs to the *certification-based* class. Another example of weak protocol can be found in [18], where three different correctness criteria are supported: 1C-sequential-SI, 1C-atomic-SI and 1SR [7]. To this end, 1C-sequential-SI is implemented with the solution presented in the previous paper, and 1C-atomic-SI is supported extending such protocol with the pessimistic approach of [24] (detailed in Section 4.3). On the other hand, the third criterion is the traditional 1SR (i.e., 1C-sequential-serializable using our naming conventions), that can be implemented extending the 1C-sequential-SI variant with the mechanisms suggested in [12].

Finally, there have been other papers following the *passive* replication protocol class. The Ganymed

middleware [27] is one of such systems. Its scheduler demands each transaction to be tagged with a *read-only* or *update* label. Depending on its label, transactions should be forwarded to the primary replica (the update ones) or can be directly served by any secondary replica (read-only transactions). This approach simplifies concurrency management, since write-write conflicts may only arise in the primary replica and can be dealt with the underlying DBMS concurrency control mechanisms without requiring any inter-replica interaction. On the other hand, since read-only transactions are served by secondary replicas, they do not interfere with the update load service and thus scalability is greatly enhanced. This same architecture was assumed in [10].

4.3 1C-Atomic-SI Model

As we have seen in Section 4.2, at least two papers have referenced this model: [11] for its CSI, and [10] for its Strong SI. The former provided an analytical evaluation based on an abstract protocol that added a blocking starting step to the concrete protocol presented for supporting GSI, but none of these papers presented a complete algorithm that implements 1C-Atomic-SI.

An implementation of this model needs to re-force a 1C-SI protocol in order to support atomic replica consistency instead of sequential replica consistency, but this is not easy. There have been some general protocols (independent of the isolation level being supported) able to provide such kind of support. The first approaches can be found in [24], where two different protocols based on DBSM [25] were described. Note that DBSM was able to support both *snapshot* and *serializable* isolations, as described in [34]. Thus, the first protocol of [24] uses an optimistic evaluation: read accesses are considered in the commit-time evaluation steps of the protocol, so both read-only and read-update transactions might abort if they have accessed past versions of their items. In its second protocol a pessimistic approach is used. It is based on multicasting in total order a transaction START message that ensures that each transaction is able to get its intended last snapshot; i.e., that all previously finished transactions have delivered their writesets in the delegate replica of the starting transaction. Let us assume that r_i is such delegate replica for a starting transaction t_i. This approach is able to approximate atomic semantics, since it ensures that when t_i is allowed to start, all transactions that were in their committing step have been able to deliver their writesets in r_i; i.e., there will not be any transaction being terminated that had their writeset "*in transit*" and that will not be known in the snapshot taken by t_i. Note that such "*in transit*" transactions might have terminated in some of the replicas and could be read by other starting transactions. So, their effects need to be present in the snapshot of t_i in order to follow the atomic consistency semantics. However, to precisely implement this solution we need to delay again the start of t_i until all such delivered writesets have been positively certified and applied in r_i, and this might imply a long interval in overloaded replicas.

A second paper that ensures 1C-Atomic consistency for replicated databases is [29]. Its *Write-Consensus Read-Quorum* (WCRQ) protocol is based on read-write quorums, minimizing thus the communication costs for guaranteeing such kind of replica consistency. To this end, writesets are broadcast to all database replicas, but in order to accept an update transaction, only a write-quorum of positive acknowledgments is needed. On the other hand, read-only transactions need to be checked in a read-quorum of replicas, and they are accepted when all such replicas return a positive acknowledgment. Let us note that such positive acknowledgments require that the versions read or written respect all the constraints imposed by the atomic consistency model; i.e., that the read values correspond to the latest existing item versions, and that the write order is the same in all replicas.

Another algorithm supporting snapshot isolation and atomic consistency was presented in [30], where the 1C-Atomic-SI model was named *Strict Snapshot Isolation* (SSI). However, instead of using a blocking step as suggested in [11], this last solution uses an optimistic approach: all transactions are allowed to start without blocking, although they need to multicast in total order a message in order to find out whether such starting phase complies with SSI; i.e., when such START message is delivered, the logical transaction starting point is set. When the transaction requests commitment, its readset is compared against those writesets delivered before its START message but not included in its snapshot. If a non-empty intersection is found, the transaction is aborted. This approach eliminates all the delays presented in our description of the pessimistic protocol of [24], although at the price of aborting all transactions that need to read any of the items being updated in such hypothetical blocking interval.

Finally, [33] presents an interesting set of results discussing the overhead implied by supporting several

correctness criteria in a single protocol, showing that stricter criteria do not always imply any noticeable overhead. Such work is focused on the *serializable* isolation level, so it does not explicitly cover the target of this section (1C-Atomic-SI protocols), but it includes three different correctness criteria for *serializable* isolation: 1SR (i.e., 1C-sequential-serializable, following our naming recommendations), 1C-session-serializable (based on the specifications given in [10]), and *strong serializable* [8] (that would correspond to 1C-atomic-serializable with our recommendations). The mechanisms needed for assuring the 1C-atomic-serializable model are similar to those already described above: to multicast in total order a message that allows the transaction start, even in read-only transactions.

5 Applicability

One of the results of this paper consists in finding two different and complementary issues in the consistency being ensured by database replication protocols: isolation consistency and replica consistency. Different papers have given different names to the correctness criteria assumed in their protocols. A first aim of this paper is to propose a uniform naming for those criteria. We also consider that traditional database replication protocols ensure sequential consistency and that such fact should be proved. On the other hand, this also allows to revisit DSM systems, looking for theoretical results that justify some of the properties exhibited by some database replication variants. These issues are detailed in Sections 5.1 to 5.3.

5.1 Naming

Table 2 summarizes the names given to their assumed correctness criteria in all surveyed papers where some SI-related database replication protocol is described. Note that in some papers more than one protocol is proposed or more than one isolation level is being supported by a single protocol. We only consider sequential (assumed as default) and atomic (abbreviated as "at") replica consistency in that table, but we also add the serializable (abbreviated as SER) isolation level for completeness, since some protocols or papers also discuss it.

Paper		1C-SI	1C-at-SI	1C-SER	1C-at-SER
Bernstein et al., 1987	[7]	–	–	1SR	–
Daudjee & Salem, 2006	[10]	weak SI	strong SI	–	–
Elnikety et al., 2005	[11]	GSI	CSI	–	–
Juárez et al., 2007	[18]	GSI	SI	SER	–
Kemme & Alonso, 2000	[19]	SI	–	SER	–
Lin et al., 2005	[21]	1C-SI	–	–	–
Muñoz-Escoí et al., 2006	[23]	GSI	–	–	–
Plattner & Alonso, 2004	[27]	SI	–	–	–
Salinas et al., 2008	[30]	GSI	strict SI	–	–
Wu & Kemme, 2005	[32]	SI	–	–	–
Zuikevičiūtė & Pedone, 2005	[34]	SI	–	1SR	–
Zuikevičiūtė & Pedone, 2008	[33]	–	–	1SR	strong SER

Table 2: Names given to the correctness criteria.

In such table, GSI stands for *Generalized Snapshot Isolation* and CSI for *Conventional Snapshot Isolation*, whilst the well-known 1SR acronym means 1C-serializability.

Note that for the *serializable* isolation level there is no possible ambiguity since the concept of 1SR [7] (1C-sequential-serializable) has been widely accepted. On the other hand, when *snapshot* isolation is considered in a replicated context, multiple names have been used and referring only to SI is ambiguous. The aim of our paper is to avoid such ambiguity, promoting a naming that refers to both kinds of consistency when a correctness criterion is used.

5.2 Justification of Some Protocol Properties

An example of such properties is one of the propositions presented (but not proved) in [11] regarding *Conventional Snapshot Isolation*(CSI) (i.e., 1C-Atomic-SI). Note that such paper assumed a pessimistic protocol behavior, and its *Proposition 3* says: *There is no non-blocking implementation of CSI in an asynchronous system, even if database sites never fail.* On the other hand, GSI (i.e, 1C-Sequential-SI) does not demand such blocking implementations. In such scope, "*blocking*" means that transactions are prevented from starting for some time. That proposition can be directly proven using some of the results generated in the *atomic* replica consistency model. For instance, [4] proves that in a *linearizable* (i.e., atomic) replica consistency model, the minimum worst-case time for a read operation is at least $u/4$ whilst the worst-case time for a write operation is at least $u/2$, being u the uncertainty in the message transmission delay ($u > 0$). So, both kinds of operations have a blocking interval in such model. On the other hand, the same paper proves that in a sequentially consistent system either read or write operations can be immediately completed, but not both. In practice, in the database replication field, read operations can be immediately served by the local replica, whilst write operations demand a blocking interval to deliver the updates being propagated by a total order broadcast. So, such results are able to directly justify the blocking differences between the GSI and CSI concepts presented in [11].

Note that there may be database replication protocols that overcome such blocking constraint; for instance, those based on optimistic management [30, 24]. However, this is achieved by transactions that in their certification phase will be sanctioned to abort if they have violated the replica consistency model properties; i.e., they do not block at their start, but if they read data from an obsolete snapshot, they will abort. The blocking behavior of a pessimistic management prevents such kinds of abort from happening, ensuring always that the adequate snapshot is being read.

5.3 Sequential Consistency

Mosberger [22] states the following referring to the sequential consistency model and one-copy serializability:

> In a sequentially consistent system, all processors must agree on the order of observed effects.
>
> This is equivalent to the one-copy serializability concept found in work on concurrency control for database systems [7].

However, such paper does not prove the second sentence, although such proof is almost immediate (and have been widely assumed as such) when the 1SR definition is consulted.

Such definition says (Theorem 8.3 in [7, Page 275]):

> Let H be an RD history. If H has the same reads-from relationships as a serial 1C history H_{1C}, where the order of transactions in H_{1C} is consistent with SG(H), then H is 1SR.

Given that:

1. An RD history H is a *complete replicated data history* [7, Pages 271-272] that includes all the operations executed in every system replica and that maintains the execution order in every transaction and replica.
2. The RD history definition also compels to maintain the order between conflicting operations being executed by different transactions. A pair of operations conflict if at least one of them is a write.
3. The 1SR definition uses a logical H_{1C} history whose transaction order is consistent with that of H, with the same aim as the "*...in some sequential order...*" clause of the sequential consistency definition; i.e., to define a logical global order that matches the program order of each system process.

...such three elements set a clear correspondence between the consistency issues of 1SR and those of the sequential consistency model, justifying the statements given in [22].

6 Conclusions

Sequential consistency was assumed in the first one-copy equivalence targeted for replicated databases: one-copy serializability. Snapshot isolation is another isolation level widely used in modern applications since it does not need to block read operations and is also able to ensure serializability with some care. Thus, multiple database replication protocols have supported snapshot isolation in a replicated environment, but there is no consensus on what should be understood as one-copy equivalence when such isolation level is used. Almost each paper has given a different name to its assumed correctness criterion.

We have surveyed multiple database replication papers that provide such isolation level, and we have proposed a taxonomy in order to refer to one-copy equivalence. To this end, multiple variants have been distinguished depending on the assumed replica consistency model. This permits an easy justification of some protocol properties, since they depend on both the isolation level and the consistency model. Existent properties in such two fields are able to justify the behavior of such protocols.

Acknowledgments

This work has been partially supported by EU FEDER and the Spanish MEC under grants TIN2006-14738-C02 and by IMPIVA under grant IMIDIC/2007/68.

References

[1] Sarita V. Adve and Kourosh Gharachorloo. Shared memory consistency models: A tutorial. *IEEE Computer*, 29(12):66–76, 1996.

[2] A. Adya. *Weak Consistency: A Generalized Theory and Optimistic Implementations of Distributed Transactions*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, March 1999.

[3] Divyakant Agrawal, Gustavo Alonso, Amr El Abbadi, and Ioana Stanoi. Exploiting atomic broadcast in replicated databases. In *3rd International Euro-Par Conference*, pages 496–503, Passau, Germany, August 1997. Springer.

[4] Hagit Attiya and Jennifer L. Welch. Sequential consistency versus linearizability. *ACM Trans. Comput. Syst.*, 12(2):91–122, 1994.

[5] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil. A critique of ANSI SQL isolation levels. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 1–10, San José, CA, USA, May 1995. ACM Press.

[6] Josep M. Bernabé-Gisbert, Raúl Salinas-Monteagudo, Luis Irún-Briz, and Francesc D. Muñoz-Escoí. Managing multiple isolation levels in middleware database replication protocols. *Lecture Notes in Computer Science*, 4330:511–523, December 2006.

[7] Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems*. Addison Wesley, Reading, MA, USA, 1987.

[8] Yuri Breitbart, Hector Garcia-Molina, and Abraham Silberschatz. Overview of multidatabase transaction management. *VLDB J.*, 1(2):181–239, 1992.

[9] Vicent Cholvi and Josep Bernabéu. Relationships between memory models. *Information Processing Letters*, 90:53–58, 2004.

[10] Khuzaima Daudjee and Kenneth Salem. Lazy database replication with snapshot isolation. In *32nd International Conference on Very Large Data Bases*, pages 715–726, Seoul, Korea, September 2006. ACM.

[11] S. Elnikety, W. Zwaenepoel, and F. Pedone. Database replication using generalized snapshot isolation. In *Symposium on Reliable Distributed Systems*, pages 73–84, Orlando, FL, USA, October 2005. IEEE-CS Press.

[12] Alan Fekete, Dimitrios Liarokapis, Elizabeth J. O'Neil, Patrick E. O'Neil, and Dennis Shasha. Making snapshot isolation serializable. *ACM Trans. Database Syst.*, 30(2):492–528, 2005.

[13] Shel Finkelstein, Rainer Brendle, and Dean Jacobs. Principles for inconsistency. In *4th Biennial Conf. on Innovative Data Systems Research (CIDR)*, Asilomar, CA, USA, January 2009.

[14] J. Goodman. Cache consistency and sequential consistency. Technical Report 61, IEEE Scalable Coherence Interface Working Group, 1989.

[15] Rachid Guerraoui, Benoît Garbinato, and Karim Mazouni. The GARF library of DSM consistency models. In *ACM SIGOPS European Workshop*, pages 51–56, 1994.

[16] Pat Helland and Dave Campbell. Building on quicksand. In *4th Biennial Conf. on Innovative Data Systems Research (CIDR)*, Asilomar, CA, USA, January 2009.

[17] Maurice Herlihy and Jeannette M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Trans. Program. Lang. Syst.*, 12(3):463–492, 1990.

[18] J. R. Juárez-Rodríguez, J. E. Armendáriz-Iñigo, J. R. González de Mendívil, F. D. Muñoz-Escoí, and J. R. Garitagoitia. A weak voting database replication protocol providing different isolation levels. In *7th Intnl. Conf. on New Technologies of Distr. Syst.*, pages 261–268, Marrakesh, Morocco, June 2007.

[19] B. Kemme and G. Alonso. A new approach to developing and implementing eager database replication protocols. *ACM Transactions on Database Systems*, 25(3):333–379, September 2000.

[20] Leslie Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE Trans. Computers*, 28(9):690–691, 1979.

[21] Y. Lin, B. Kemme, M. Patiño-Martínez, and R. Jiménez-Peris. Middleware based data replication providing snapshot isolation. In *SIGMOD Conf. on Management of Data*, pages 419–430, Baltimore, MD, USA, 2005. ACM Press.

[22] David Mosberger. Memory consistency models. *Operating Systems Review*, 27(1):18–26, 1993.

[23] Francesc D. Muñoz-Escoí, Jerónimo Pla-Civera, María Idoia Ruiz-Fuertes, Luis Irún-Briz, Hendrik Decker, José Enrique Armendáriz-Iñigo, and José Ramón González de Mendívil. Managing transaction conflicts in middleware-based database replication architectures. In *Symposium on Reliable Distributed Systems*, pages 401–410, 2006.

[24] Rui Carlos Oliveira, José Pereira, Alfrânio Correia Jr., and Edward Archibald. Revisiting 1-copy equivalence in clustered databases. In *Symposium on Applied Computing*, pages 728–732, Dijon, France, April 2006. ACM Press.

[25] F. Pedone. *The Database State Machine and Group Communication Issues*. PhD thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 1999.

[26] Fernando Pedone, Rachid Guerraoui, and André Schiper. Exploiting atomic broadcast in replicated databases. In *4th International Euro-Par Conference*, pages 513–520, Southampton, UK, September 1998. Springer.

[27] Christian Plattner and Gustavo Alonso. Ganymed: Scalable and flexible replication. *IEEE Data Eng. Bull.*, 27(2):27–34, 2004.

[28] L. Rodrigues, H. Miranda, R. Almeida, J. Martins, and P. Vicente. The GlobData fault-tolerant replicated distributed object database. In *1st EurAsian Conf. on Information and Communication Technology*, pages 426–433, Shiraz, Iran, 2002. Springer.

[29] Luís Rodrigues, Nuno Carvalho, and Emili Miedes. Supporting linearizable semantics in replicated databases. In *7th IEEE International Symposium on Networking Computing and Applications*, pages 263–266, Cambridge, MA, USA, July 2008. IEEE-CS Press.

[30] Raúl Salinas, Francesc D. Muñoz-Escoí, J. Enrique Armendáriz-Íñigo, and José Ramón González de Mendívil. A performance analysis of g-bound, a consistency protocol supporting multiple isolation levels. *Lecture Notes in Computer Science*, 2008.

[31] Matthias Wiesmann and André Schiper. Comparison of database replication techniques based on total order broadcast. *IEEE Trans. on Knowledge and Data Engineering*, 17(4):551–566, April 2005.

[32] Shuqing Wu and Bettina Kemme. Postgres-R(SI): Combining replica control with concurrency control based on snapshot isolation. In *21st Intl. Conf. on Data Eng. (ICDE)*, pages 422–433, Tokyo, Japan, April 2005. IEEE-CS Press.

[33] V. Zuikevičiūtė and F. Pedone. Correctness criteria for database replication: Theoretical and practical aspects. In *10th International Symposium on Distributed Objects, Middleware and Applications*, pages 639–656, Mexico, November 2008. Springer.

[34] Vaide Zuikevičiūtė and Fernando Pedone. Revisiting the database state machine. In *VLDB Workshop on Design, Implementation and Deployment of Database Replications*, Trondheim, Norway, August 2005.