

DB pro sem - isolation levels

Jeroen-Niclas Trzaska
TU Dresden
Dresden, Saxony, Germany
jeroen@trzaska.xyz

ABSTRACT

This paper is going to look at the strengthened ANSI definition of isolation levels, comparing them in terms of strength. We will also take a quick look at non lock based implementations and the problems the ANSI standard has in regards to such implementations as shown in [Berenson et al. 1995].

ACM Reference Format:

Jeroen-Niclas Trzaska. 2020. DB pro sem - isolation levels. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 3 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 PHENOMENA DEFINITIONS

We're now going to introduce the strengthened ANSI phenomena definitions as described in [Adya et al. 2000] which are based on the broad interpretation of the ANSI standard. These definitions are inherently stronger than their ANSI counterparts, as they rule out other phenomena not defined in the following paragraph.

One might assume that ANSI had these strengthened definitions in mind when specifying them as lock based implementations are by far the most common systems.

1.1 P0 - Dirty write

An example for the above mentioned problem with ANSI strength is that while a locking implementation of read uncommitted prevents the following phenomenon, a non locking system would not do so. So let's start by defining the first phenomenon, describing two consecutive writes by two different transactions leading to an unclear value for x when the first transaction aborts.

Example 1.1. Alice deposits 10€ into a bank account X. Bob then deposits a further 20€ into the same account. To do so he reads the value of X and then adds 20 and writes it back to the account. Alice then redacts her transaction. Thus the amount of money in account X is unclear.

1.2 P1 - Dirty read

For the second phenomenon we're going to look at reading inconsistent data. This can be a problem when retrieving information while another transaction that writes data is being executed.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, July 2017, Washington, DC, USA

© 2020 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

Example 1.2. Alice transfers 40€ from account x to account y. To do so she reads the value of x subtracts the 40€. Afterwards, Bob reads the value of x (10€) and y (15€) resulting in a sum of 25€. Alice then reads the value of y (15€) and then adds the 40€ to it resulting in a 55€ sum.

1.3 P2 - Fuzzy read

The next phenomenon is similar to P1 as it has to do with read inconsistencies. Though this time performing the two reads of the reading transaction without interruption would lead to the correct data, so the read is non repeatable.

Example 1.3. Bob starts to read the sum by reading x and gets interrupted by Alice who then transfers 40€ from x to y after this is completed Bob reads the value of y. Thus the sum is 100 for Alice and 140 for Bob. In contrast to P1 a second read of x by Bob would have returned the correct sum.

1.4 P3 - Phantom

Once again when looking at the ANSI specification one can see that it only disallows inserts on the serializable isolation level while the following definition of P3 prohibits deletions, inserts, and updates. This phenomenon is only applicable to transactions using predicates and applies to deletions, inserts, and updates.

Example 1.4. Alice reads the predicate blue cars. Bob then adds a new blue car to the database and increases the count of blue cars. Alice then reads the count of blue cars. This leads to a phantom item (Alice can't see Bob's car but knows of its existence due to the count).

2 ANSI ISOLATION LEVELS

We define the isolation levels by the phenomena prohibited by it and we define Degree 0 isolation to be the isolation level that doesn't prohibit any abnormal behaviour. Furthermore, we define short duration locks as locks that are held while a transaction reads or writes an item, in contrast to long-duration locks which are held until a transaction commits or aborts.

2.1 Read uncommitted

We define read uncommitted to be the isolation level that prohibits the phenomenon P0, dirty write, this can be accomplished by placing long-duration write locks when modifying a data item. Thus only allowing one transaction at a time to modify a given value.

2.2 Read committed

Read committed is defined as the isolation level where P1, dirty read, is prohibited. To accomplish this the use of short duration read

locks, as well as long write locks, is necessary. Thus disallowing a write on data that was just read by another transaction.

2.3 Repeatable read

For Repeatable read, the phenomenon P2, fuzzy read, has to be prohibited. This requires long duration write locks as well as long duration read locks on items and short ones for Predicates.

The difference to read committed is that read committed only guarantees that the data read was committed at the time of reading (no dirty reads), whereas repeatable read also guarantees that the data will not change before the transaction finishes either with a commit or abort.

2.4 Serializable

For serializability P3 needs to be prohibited, which can be done by placing long locks on reads and writes.

3 MOTIVATION

The motivation behind these theoretical analyses is to better understand the behaviour of different systems when presented with multi-item dependencies. Such dependencies include but aren't limited to constraints between two items.

Let's say that Alice has two accounts at a bank. She has 20€ on each of them now she withdraws 30€ from account A and another 20€ from account B. The looking at each of the transactions we can see that the constraint $A + B \geq 0$ is met. But when executed simultaneously the resulting sum can still be negative.

Example 3.1. Alice reads 20€ to be in both accounts and Bob does the same. Alice then withdraws 30€ from account A and writes -10€ into it and commits, the constraint is still met. Bob then withdraws 20€ from account and , as his transaction read both accounts to have 20€, he writes 0€ to B. This now violates the constraint $A + B \geq 0$.

4 CURSOR STABILITY

For the cursor stability, it is useful to define a fourth phenomenon P4 (lost update).

4.1 P4 - lost update

This phenomenon describes the loss of an update by a transaction due to it being immediately overwritten by a second transaction.

Example 4.1. Alice reads x to be 100€. Bob then reads the same, adds 20€ and writes 120 to x and commits. Alice then subtracts 40€ from x and writes 60€ to x and commits. Resulting in the loss of Bob's update.

4.2 Implications

The phenomena defined above can be prohibited when a read lock is placed on x (position of the cursor) when read and placing a long write lock on the items row when modified. The read lock is removed when the cursor moves (e.g. reading a new cell), while the write lock only gets removed after the transaction commits or aborts. As Cursor stability is stronger than read committed while being weaker than repeatable read it is used instead of the weaker repeatable read by some database systems. This behaviour is allowed by the ANSI standard as it requires a minimum set of guarantees.

4.3 Multi cursor

The system described above could also be extended with the use of multiple cursors thus parleying the effect of repeatable read isolation as long as the transactions don't access more items than the system has cursors. But obviously this isn't a general, or even practical solution to the phenomenon P2.

5 SNAPSHOT ISOLATION

All of the above-mentioned systems suggest lock-based implementations. We will now look at snapshot isolation which in contrast uses multi-versioning. In snapshot isolation, each transaction gets its own snapshot of committed data at the start. Thus reads are never blocked, while writes are performed on the given snapshot as to not disturb other transactions and allow reads of the updated data only by the same transaction (unless committed). For the committing procedure, a second (commit) timestamp is assigned, said timestamp is larger than every other assigned time stamp in the system. If another transaction A already modified data the commit of

5.1 Strength

Snapshot isolation is comparable in strength to repeatable read as both prohibit P0, P1, and P2. transaction B fails, leading to an abort if said data was modified by transaction B.

5.2 P5B - Write skew

To take a closer look at the difference to repeatable read it is useful to define a new phenomenon.

Example 5.1. (first commit wins):

Alice reads x and y to be 50€. Bob then reads the same data, while getting his own snapshot to operate on (newer timestamp than Alice). Alice then sets y to be -40€ and commits, resulting in a sum of 10€. Bob then sets y to be 30€ and tries to commit. As Alice already modified x Bob's transaction will abort.

5.3 Comparison with repeatable read

5.4 Comparison to ANSI

When looking at the weak interpretation of the ANSI specification as in [Adya et al. 2000] one can see that Snapshot isolation prohibits all three anomalies A1, A2, and A3 making it ANOMALY SERIALIZABLE. To show this we will now take a look at A3 as we have already shown above that Snapshot isolation prohibits P1 and P2.

5.5 A3 - Anomaly Phantom

This is the strict interpretation of the Phantom anomaly. The difference to P3 is the position of the transactions commit is not indifferent.

Example 5.2. Alice reads the predicate blue cars. Bob then adds a new blue car to the database and increases the count of blue cars. Bob commits. Alice then reads the count of blue cars. This leads to a phantom item (Alice can't see Bob's car but knows of its existence due to the count).

Thus A3 only operates on committed data while P3 allows for transactions to still be in progress.

6 CONCLUSION

REFERENCES

Atul Adya, Barbara Liskov, and Patrick O'Neil. Generalized isolation level definitions. 2000. URL <http://pmg.csail.mit.edu/papers/icde00.pdf>.

Hal Berenson, Phil Bernstein, Jim Gray, Jim Melton, Elizabeth O'Neil, and Patrick O'Neil. A critique of ansi sql isolation levels. *ACM SIGMOD Record*, 24(2):1–10, May 1995. doi: 10.1145/568271.223785. URL <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-95-51.pdf>.