DB pro sem - isolation levels

Jeroen-Niclas Trzaska

ABSTRACT

abstract

ACM Reference Format:
Jeroen-Niclas Trzaska. 2020. DB pro sem - isolation levels. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRO

2 PHEMOMENA DEFINITIONS

We're now going to introduce the strengthened ANSI phenomena definitions taken from (cite here) which are based on the broad interpretation of the ANSI standard (cite).

2.1 P0 - Dirty write

Lets start by defining the first phenomena, describing two consecutive writes by two differnt transactions leading to an unclear value for x when the first transaction aborts.

Example 2.1. Alice deposits 10€ into a bank account X. Bob than 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. Allice than redacts her transaction. Thus the amount of monney in account X is unclear.

2.2 P1 - Dirty read

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

Example 2.2. Alice transfers 40€ form 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.

2.3 P2 - Fuzzy read

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

Example 2.3. 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.

2.4 P3 - Phantom

This phonomenon is only applicable to transactions using predicates and aplies to deletions, insets, and upadates.

Example 2.4. 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).

3 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.

3.1 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.

3.2 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.

3.3 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.

3.4 Serializable

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

4 MOTIVATION

The motivation behind these theoretical analyses is to better understand the behaviour of different systems when presented with multi-item dependencies.

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

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

5 CURSOR STABILITY

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

5.1 P4 - lost update

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.

5.2 Implications

The phenomena define 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 then read commited while being weaker then 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 reqiures a minimum set of guarantees.

5.3 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 dont acces more items then the system has cursors. But obiously this isn't a general, or practical solution to the phenomenon P2.

6 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 transaction B fails, leading to an abort if said data was modified by transaction B.

Example 6.1. (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.

6.1 Strength

Snapshot isolation is comparable in strength to repeatable read