WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Deutsch
Edit links

Algorithms for Recovery and Isolation Exploiting Semantics

From Wikipedia, the free encyclopedia

This article includes a list of references, related reading or external links, **but its sources remain unclear because it lacks inline citations.** Please improve this article by introducing more precise citations. *(March 2013)*

In computer science, **Algorithms for Recovery and Isolation Exploiting Semantics**, or **ARIES** is a recovery algorithm designed to work with a no-force, steal database approach; it is used by IBM DB2, Microsoft SQL Server and many other database systems.

Three main principles lie behind ARIES

- Write ahead logging: Any change to an object is first recorded in the log, and the log must be written to stable storage before changes to the object are written to disk.
- Repeating history during Redo: On restart after a crash, ARIES retraces the actions of a database before the crash and brings the system back to the exact state that it was in before the crash. Then it undoes the transactions still active at crash time.
- Logging changes during Undo: Changes made to the database while undoing transactions are logged to ensure such an action isn't repeated in the event of repeated restarts.

Contents [hide]

Logging [edit]

For the ARIES algorithm to work a number of log records have to be created during the operation of the database. Log entries are sequentially ordered with Sequence Numbers.

Usually the resulting logfile is stored on so-called "stable storage", that is a storage medium that is assumed to survive crashes and hardware failures. To gather the necessary information for the logging two data structures have to be maintained: the dirty page table (DPT) and the transaction table (TT).

The dirty page table keeps record of all the pages that have been modified and not yet written back to disc and the first Sequence Number that caused that page to become dirty. The transaction table contains all transactions that are currently running and the Sequence Number of the last log entry they caused.

We create log records of the form (Sequence Number, Transaction ID, Page ID, Redo, Undo, Previous Sequence Number). The Redo and Undo fields keep information about the changes this log record saves and how to undo them. The Previous Sequence Number is a reference to the previous log record that was created for this transaction. In the case of an aborted transaction, it's possible to traverse the log file in reverse order using the Previous Sequence Numbers, undoing all actions taken within the specific transaction.

Every time a transaction begins or commits we write a "Begin Transaction" entry or an "End Of Log" entry for that transaction respectively.

During a recovery or while undoing the actions of an aborted transaction a special kind of log record is written, the Compensation Log Record (CLR), to record that the action has already been undone. CLRs are of the form (Sequence Number, Transaction ID, Page ID, Redo, Previous Sequence Number, Next Undo Sequence Number). The Undo field is omitted because that information is already stored in the original log record for those actions.

Recovery [edit]

The recovery works in three phases. The first phase, Analysis, computes all the necessary information from the logfile. The Redo phase restores the database to the exact state at the crash, including all the changes of uncommited transactions that were running at that point in time. The Undo phase then undoes all uncommited changes, leaving the database in a consistent state.

Analysis [edit]

During the Analysis phase we restore the DPT and the TT as they were at the time of the crash.

We run through the logfile (from the beginning or the last checkpoint) and add all transactions for which we encounter Begin Transaction entries to the TT. Whenever an End Log entry is found, the corresponding transaction is removed. The last Sequence Number for each transaction is of course also maintained.

During the same run we also fill the dirty page table by adding a new entry whenever we encounter a page that is modified and not yet in the DPT. This however only computes a superset of all dirty pages at the time of the crash, since we don't check the actual database file whether the page was written back to the storage.

Redo [edit]

From the DPT we can compute the minimal Sequence Number of a dirty page. From there we have to start redoing the actions until the crash, in case they weren't persisted already.

Running through the log file, we check for each entry, whether the modified page P on the entry exists in the DPT table. If it doesn't, then we do not have to worry about redoing this entry since the data persists on the disk. If page P exists in the DPT table, then we see whether the Sequence Number in the DPT is smaller than the Sequence Number of the log record (i.e. whether the change in the log is newer than the last version that was persisted). If it isn't, then we don't redo the entry since the change is already there. If it is, we fetch the page from the database storage and check the Sequence Number stored on the page to the Sequence Number on the log record. If the former is smaller than the latter, the page does not need to be written to the disk. That check is necessary because the recovered DPT is only a conservative superset of the pages that really need changes to be reapplied. Lastly, when all the above checks are finished and failed, we reapply the redo action and store the new Sequence Number on the page. It is also important for recovery from a crash during the Redo phase, as the redo isn't applied twice to the same page.

Undo [edit]

After the Redo phase the database reflects the exact state at the crash. However the changes of uncommited transactions have to be undone to restore the database to a consistent state.

For that we run backwards through the log for each transaction in the TT table (those runs can of course be combined into one) using the Previous Sequence Number fields in the records. For each record we undo the changes (using the information in the Undo field) and write a compensation log record to the log file. If we encounter a Begin Transaction record we write an End Log record for that transaction.

The compensation log records make it possible to recover during a crash that occurs during the recovery phase. That isn't as uncommon as one might think, as it is possible for the recovery phase to take quite long. CLRs are read during the Analysis phase and redone during the Redo phase.

Checkpoints [edit]

To avoid rescanning the whole logfile during the analysis phase it is advisable to save the DPT and the TT regularly to the logfile, forming a checkpoint. Instead of having to run through the whole file it is just necessary to run backwards until a checkpoint is found. From that point it is possible to restore the DPT and the TT as they were at the time of the crash by reading the logfile forward again. Then it is possible to proceed as usual with Redo and Undo.

The naive way for checkpointing involves locking the whole database to avoid changes to the DPT and the TT during the creation of the checkpoint. Fuzzy logging circumvents that by writing two log records. One Fuzzy Log Starts Here record and, after preparing the checkpoint data, the actual checkpoint. Between the two records other logrecords can be created. During recovery it is necessary to find both records to obtain a valid checkpoint.

External links [edit]

- C. Mohan, ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging, ACM Transactions on Database Systems, Vol. 17, No. 1, March 1992, pp. 94–162
- C. Mohan, Repeating History Beyond ARIES, Proceedings of 25th International Conference on Very Large Data Bases, 1999
- *Impact of ARIES Family of Locking and Recovery Algorithms - C. Mohan*, archived from the original on 2012-08-19, retrieved 2013-09-18

Categories: Database algorithms