International Journal of Research in Advent Technology, Vol.6, No.6, June 2018
E-ISSN: 2321-9637
Available online at www.ijrat.org

Repairing Damaged Database Using Table Structure

Nadeem Beg[1], Dr. R. B. Ingle[2]
[1]*Department of Computer Engineering, Pune Institute of Computer Technology, Pune, India*
[2]*Department of Computer Engineering, Pune Institute of Computer Technology, Pune, India*
Email: nadeembeg007@gmail.com [1]

Abstract - Database is collection of related data that can easily be manageable. In relational database system, data is stored in the form of tables (i.e. rows and columns) and that data can be indexed so that one can efficiently perform search operation on data. DBMS (database management system) organizes data in such a way that it can easily be accessible. Like other computer systems, DBMS is also subject to failure due to several reasons such as disk failure, hardware failure or transaction failure. After a failure, database must be recovered to its previous consistent state as consistent data should be available whenever require. For example, during execution of transaction, database system must ensure that all operations are carried out (i.e. completed successfully) and changes made permanent into database or in case of failure, transaction had no effect on database. For performance, application or programme is being integrated with database framework code, this application or programme has rights to access database frameworks buffer, and because of this access the possibility of data being corrupted due to unintentional application writes is increased, to detect and fix these kinds of corruption we used an alternative approach that uses table header to fix corrupted region of data.

Index Terms - Database corruption, data integrity, detection, repair.

1. INTRODUCTION

Now a day, hardware is more reliable than ever before, therefore programming errors are often the biggest threat that affects availability of database system. For high-performance, application programme is integrated with database system programme, therefore users get rights to add new data types into database. Performance-critical software required high performance that can achieved by accessing data directly that stored in a main-memory [1]. In both cases, because of inter-process communication, which is complex and costlier, due to these it is difficult to meet performance required by applications. Therefore, availability of database can be affected by software errors as well as errors in application programme.
Recovery is process of restoring database data that has been corrupted, inadvertently deleted, made inaccessible for some reason. The basic unit of recovery in a database system is the transaction. It is a unit of program execution that accesses and possibly updates various data items. It is initiated by a user program written in a high-level programming language. A database system must ensure proper execution of transactions despite failures i.e. either the entire transaction executes, or none of it does. Furthermore, it must manage concurrent execution of transactions in a way that avoids the introduction of inconsistency. To ensure integrity of the data, database system must maintain the ACID properties of the transactions. A computer system is subject to failure due to several reasons like disk crash, power outage, software error, a fire in the machine room, even sabotage. Transactions may also fail for several reasons, such as violation of integrity constraints or deadlocks. In event of failure information concerning the database system is lost. Recovery scheme detects failures and restores the database to a state that existed before the occurrence of the failure.

- Failure in database can stop the normal operation (access to database) of a database. For some failures, automatic recovery is applicable and requires little or no action on the part of the database user or database administrator. Several types of failure need to be dealt with in a different manner. There are mainly three types of failures: transaction failures, system failures, and media failures.

- Transaction Failure: When a transaction is failed to execute or it reaches a point after which it cannot be completed successfully it should abort. This is called transaction failure. Logical error and System error may cause a transaction to fail.
    - Logical error: where a transaction cannot complete because of it has some code error or any internal error condition [2].
    - System error: where the database system itself terminates an active transaction because database is not able to execute it or it should stop because of some system condition. Deadlock is example of system error [2].
- System Crash: There are problems, which are external to the system, which may cause the system to stop abruptly and cause the system to crash. For example, interruption in power supplies, failure of underlying hardware or software failure. Examples may include operating system errors.

- Disk Failure: A disk failure occurs when any part of the stable storage is destroyed. Disk failures include formation of bad sectors, unreachability to the disk, disk head crash or any other failure, which destroys all or part of disk storage.
- Repair utility reads table file to detect errors. Table file contains table header, field header and record header followed by record. Header gives information about table such as string identifier, table encryption, auto encrypted value, memo files and so on. This information is used to check correctness of table as well as memo file. This header value cross checked against table record to identify corruption for example we can calculate file length using table header and check that value with actual file length.

2. REVIEW OF LITERATURE

Log Based Recovery [1], [2]: Log is a sequential arrangement of the records, which keeps up the records of actions performed on database by a transaction. Log should be written before updating the database physically and should be stored on non-volatile storage which is fail-safe. Log reports following information transaction id, item id, old value and new value. This information is useful to undo the transaction when transaction fails. There are mainly two methods are used to update database: Deferred update and Immediate update.

Deferred updates [2]: It update the database only when transaction reached to commit point. Before reaching commit, updates of all operations of a transaction are stored in local place. For all unsuccessful transaction, there is no need to use UNDO as database is updated only after transaction reached to commit point. Sometimes it requires to use REDO operation that are stored in local place, as there result may not yet updated in database, therefore generally deferred update termed as ‘no-undo’ algorithm. Below example of Deferred update method, which updates database only after the commit point.

DEFERRED UPDATE

Log	Database
<T_0 start >	
<T_0, x, 400 >	
<T_0, y, 600 >	
<T_0 commit >	
	x = 400
	y = 600

Immediate updates [2]: In this method database is updated immediately after performing single or several operations on database before to reach commit point. All these operations are reported into the log, before updating the database, so that after failure one can easily rolled back to recover database into previous known consistent state. Hence this method is known as ’undo/redo’ algorithm.

IMMEDIATE UPDATE

Log	Database
<T_0 start >	
<T_0, x, 400, 500 >	
<T_0, y, 600, 700>	
	x = 500
	y = 700
<T_0 commit>	

Shadow Paging [2], [15]: Shadow paging is an alternate approach to log based recovery. It required very less disk access than log based recovery, but it is difficult to extend paging to allow multiple transactions executing simultaneously. Shadow paging is like paging used in operating system memory management.

The logic behind this approach is to use two pages during a transaction: current page table and shadow page table. Initially, before transaction starts executing, both tables are exactly same. The shadow page table will not change until the transaction is completed. In contrast, the current page table is updated every time with each write operation on database. All database input and output operations uses current page table to locate position of database pages on disk. Each entry on table is pointer that points to a physical page on the disk. After completion of transaction, it will be committed, then all entries of current page table are copied into shadow page table and disk is updated with new data. Shadow page table is stored in non-volatile storage which is fail-safe. If system crash occurs during transaction, then make the current page entries identical to the shadow page entries by copying shadow entries into current page, as shadow page table is never changed so need of using undo operation.

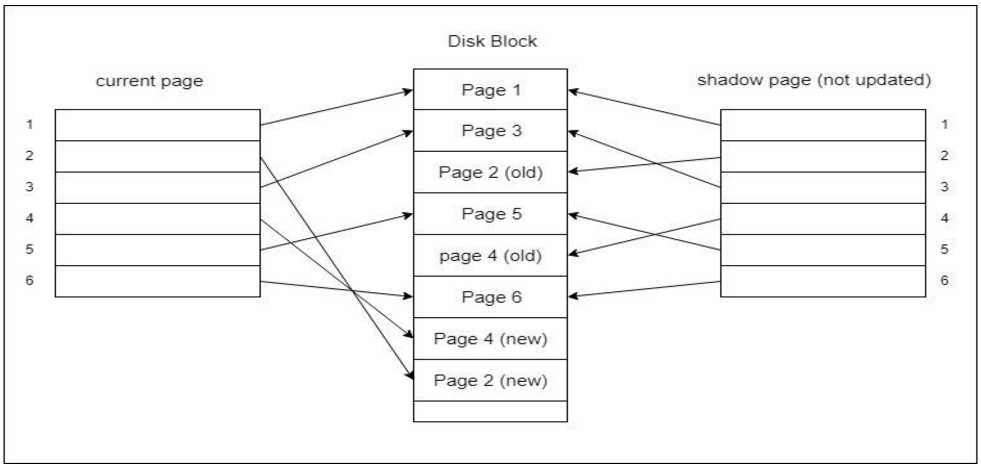


Fig. 1. Shadow paging

Recovery with Concurrent Transaction [3]: Concurrent transaction include number of users/clients that perform multiple operations on a single database which has single disk buffer and a log, which is shared by all the transactions. In this scenario database is immediately updated after a single write operation and buffer can have data items updated by multiple transactions. For concurrent transactions (i.e. multiple transactions executing simultaneously) logs are not in sequential order. So, it become difficult for recovery system to track all logs and then begin for recovery. To make it easy, most of database management system uses ’checkpoints’. When a system with concurrent transactions crashes and recovers, it behaves in the below manner-

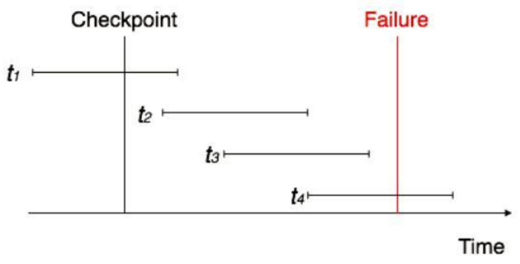


Fig. 2. Checkpoint

- The recovery system reads the logs backwards from the end to the last checkpoint.
- It maintains two lists, an undo-list and a redo-list.
- If the recovery system sees a log with <T_n, start > and <T_n, commit> or just <T_n, commit>, it puts the transaction in the redo-list.
- If the recovery system sees a log with <T_n, start> but no commit or abort log found, it puts the transaction in undo-list.

All the transactions in the undo-list are then undone and their logs are removed. All the transactions in the redo-list and their previous logs are removed and then redone before saving their logs.

Before Image Table (BI table) [4]: Before image is a image of date before it is updated or deleted. It is available in all DBMS with different form. BI table is tie with its base table and has same structure as its corresponding base table. Data which is deleted from base table is inserted automatically into BI table with the help of triggers. When data in base table is updated or deleted then after delete or after update trigger is automatically invoked and old value is inserted into corresponding BI table.

BI tables may introduce redundant data because there are chance to exist more than one before image in database system. Size of BI table is increases with increasing update or delete operation on database.

3. SYSTEM OVERVIEW

A. Software requirements

Operating System: Microsoft Windows 2003 or greater
IDE: Borland C++ Builder 6.x
Database Server

B. Hardware requirements

Intel(R) Core(TM) 2 Duo CPU @ 2.2GHz or later Memory: 2 GB DDR3 or more
Capacity: 1697MHz or more
Cores: 2 or more

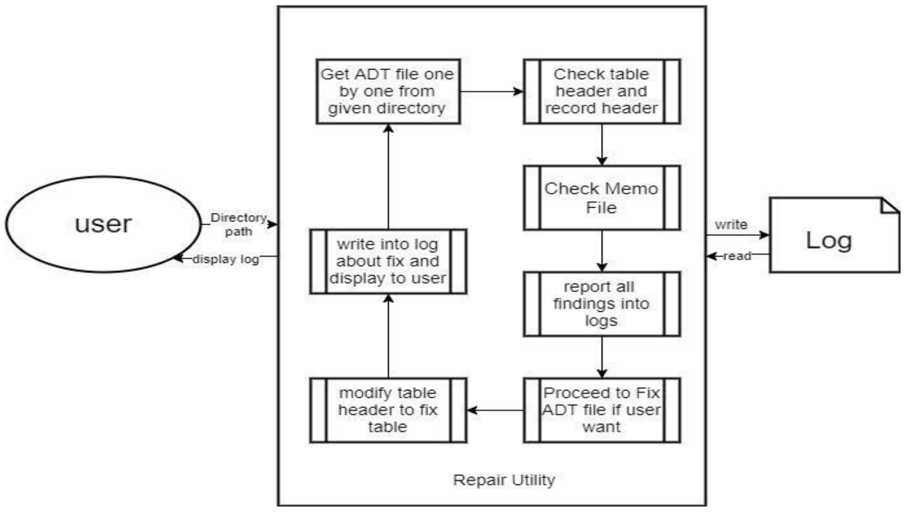

Get ADT file one by one from given directory
Check table header and record header
Check Memo File
report all findings into logs
Proceed to Fix ADT file if user want
modify table header to fix table
write into log about fix and display to user
user
Directory path
display log
write
read
Log
Repair Utility

Fig. 3. System architecture

C. Input

Directory path that contains table files and password for encrypted table.

D. Output

Display log that contains all findings information related to table as well memo files.

Create backup file for original table files before fixing the errors as it may lead to loss of data.

4. SYSTEM ARCHITECTURE

Repair utility scans a given directory for table files and performs several checks for corruption or bad data. It writes all findings to a log file and displays a 'VALID' tag for each correct file and 'INVALID' for each corrupted file in the directory. If client decide to fix the invalid files, it asks to client if he'd like to back up the original table files. It then performs the same checks and fixes problems whenever possible.

Encrypted tables require the table password to be analyzed by repair utility. If no password or an incorrect password is supplied, repair utility displays the table name along with 'ENCRYPTED' showing it was unable to analyze the table.

5. SYSTEM ANALYSIS

A. Table header

Table 1 shows structure of table header with some of its field. Table header is in the same physical file as record data.

B. Record header

Record Header is 5 bytes. It gives following record information

4 extra bytes

- Contains user ID if record involve in a transaction else contain 4-bytes long value of zero.

1 byte for other information

- Record Encrypted
- Record updated/appended within a transaction

TABLE 1: TABLE HEADER

Data Type	Description
STRING	Table identifier
UINT32	Table structure version.
UINT64	Number of records in the table
UINT32	First record position (IOW, the header length)
UINT32	Record width
UINT64	Record number of first deleted record in our deleted record linked list
UINT64	Number of records marked for deletion.
STRING	Encrypted encryption password associated with this table
VOID	Options information. Bits set for: 1) Contains memo file, 2) Has auto-increment field, 3) Table is encrypted

C. What and how to fix table file

- Incorrect file size: check physical file size with calculated file size using table header. Following is formula to calculate file size using table header

fsize = (rlength * no:of records)+headerlength (1)

where,
- fsize = filesize
rlength = record length

- Record is encrypted but table is not encrypted: Even though it is possible to encrypt some records and not others, it is usually corruption in the record header rather than intentional. With this option enabled, repair utility will clear the encryption bit in the record header.
- Records that have never been released from a transaction: Repair utility will reset transaction bits to a complete transaction state.
- Incorrect record count: Repair utility will reset the record count in the table header to reflect the current record number.
  Repair utility finds maximum auto increment value and row version value by scanning each record and compare that value with header auto increment and row version value. If mismatch found then repair utility update header value by actual max auto and max row version value which got from records.
- Corruption associated with memo files:

Memo block overlapping If repair utility finds multiple records in the table file that have memo fields pointing to the same memo data blocks in the memo file, it will make a copy of one of the memos to avoid the overlap.

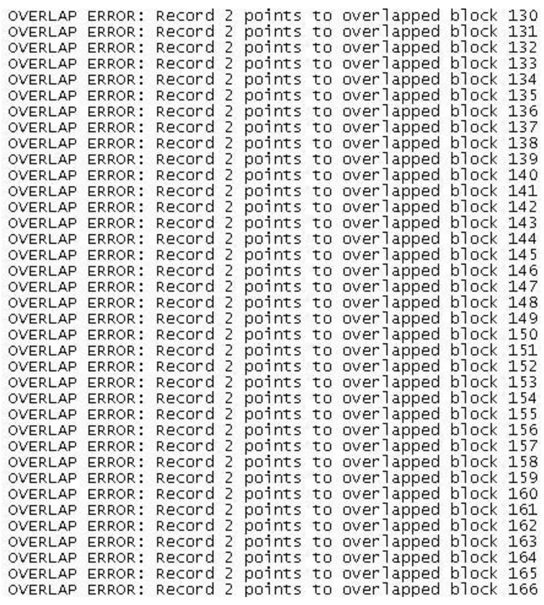
Corruption in the free block list memo files keep a list of allocated but unused blocks for memo data. If repair utility finds free list entries with bad or overlapping pointers it will clear them.

Invalid memo pointers (past EOF, before BOF, in memo header) table records with bad memo pointers will be reset to zero (no data). Orphaned memo blocks in the memo allocated but not referenced by any records will be reported to the log, but not modified.

6. RESULT

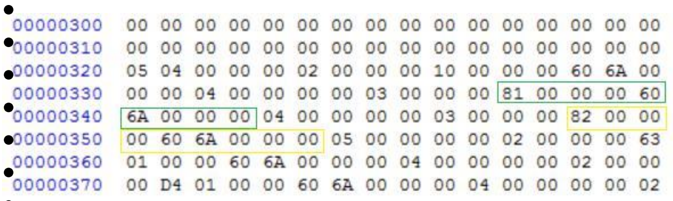
After providing corrupted table file repair utility checks table as well as memo files and report all the findings into log file. As shown in figure 4 it gives corruption information of table file.

Figure 5 shows, record points to page 81 00 00 00 with size of memo 60 6A 00 00 i.e 27232 bytes and next record points to 82 00 00 00 which is within the previous memo. This is corruption of overlapping blocks.



```
OVERLAP ERROR: Record 2 points to overlapped block 130
OVERLAP ERROR: Record 2 points to overlapped block 131
OVERLAP ERROR: Record 2 points to overlapped block 132
OVERLAP ERROR: Record 2 points to overlapped block 133
OVERLAP ERROR: Record 2 points to overlapped block 134
OVERLAP ERROR: Record 2 points to overlapped block 135
OVERLAP ERROR: Record 2 points to overlapped block 136
OVERLAP ERROR: Record 2 points to overlapped block 137
OVERLAP ERROR: Record 2 points to overlapped block 138
OVERLAP ERROR: Record 2 points to overlapped block 139
OVERLAP ERROR: Record 2 points to overlapped block 140
OVERLAP ERROR: Record 2 points to overlapped block 141
OVERLAP ERROR: Record 2 points to overlapped block 142
OVERLAP ERROR: Record 2 points to overlapped block 143
OVERLAP ERROR: Record 2 points to overlapped block 144
OVERLAP ERROR: Record 2 points to overlapped block 145
OVERLAP ERROR: Record 2 points to overlapped block 146
OVERLAP ERROR: Record 2 points to overlapped block 147
OVERLAP ERROR: Record 2 points to overlapped block 148
OVERLAP ERROR: Record 2 points to overlapped block 149
OVERLAP ERROR: Record 2 points to overlapped block 150
OVERLAP ERROR: Record 2 points to overlapped block 151
OVERLAP ERROR: Record 2 points to overlapped block 152
OVERLAP ERROR: Record 2 points to overlapped block 153
OVERLAP ERROR: Record 2 points to overlapped block 154
OVERLAP ERROR: Record 2 points to overlapped block 155
OVERLAP ERROR: Record 2 points to overlapped block 156
OVERLAP ERROR: Record 2 points to overlapped block 157
OVERLAP ERROR: Record 2 points to overlapped block 158
OVERLAP ERROR: Record 2 points to overlapped block 159
OVERLAP ERROR: Record 2 points to overlapped block 160
OVERLAP ERROR: Record 2 points to overlapped block 161
OVERLAP ERROR: Record 2 points to overlapped block 162
OVERLAP ERROR: Record 2 points to overlapped block 163
OVERLAP ERROR: Record 2 points to overlapped block 164
OVERLAP ERROR: Record 2 points to overlapped block 165
OVERLAP ERROR: Record 2 points to overlapped block 166
```

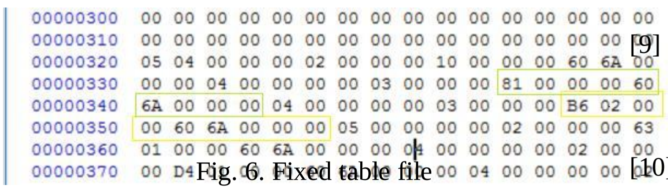
Fig. 4. Log file



```
00000300  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000310  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000320  05 04 00 00 00 02 00 00 00 10 00 00 00 60 6A 00
00000330  00 00 04 00 00 00 00 03 00 00 00 81 00 00 00 60
00000340  6A 00 00 00 04 00 00 00 00 03 00 00 00 82 00 00
00000350  00 60 6A 00 00 00 05 00 00 00 00 02 00 00 00 63
00000360  01 00 00 60 6A 00 00 00 04 00 00 00 00 02 00 00
00000370  00 D4 01 00 00 60 6A 00 00 00 04 00 00 00 00 02
```

Fig. 5. Corrupted table file

- To resolve this overlapping issue repair utility copies those overlapped blocks so that each record can have their own copy of memo and newer page location is updated in table file as shown in figure 6.



```
00000300  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000310  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000320  05 04 00 00 00 02 00 00 00 10 00 00 00 60 6A 00
00000330  00 00 04 00 00 00 00 03 00 00 00 81 00 00 00 60
00000340  6A 00 00 00 04 00 00 00 00 03 00 00 00 B6 02 00
00000350  00 60 6A 00 00 00 05 00 00 00 00 02 00 00 00 63
00000360  01 00 00 60 6A 00 00 00 04 00 00 00 00 02 00 00
00000370  00 D4 [illegible] 00 04 00 00 00 00 [illegible]
```

Fig. 6. Fixed table file

7. COCLUSION

Database is an important part of computer system that is used to store and retrieve the data. When failure occurs, then the information stored in database is lost and database becomes inconsistent. Recovery mechanism is an essential part of a database system which can restore the database to the consistent state that existed before the failure.

The system i.e. repair utility will be helpful to detect and fix that errors using table header and record header.

REFERENCES

[1] Phillip Bohannon, Rajeev Rastogi, S. Seshadri, Avi Silberschatz, “Detection and Recovery Techniques for Database Corruption”, vol.15, no.5, September/October 2003.

[2] P. R. Patel, “Database recovery Techniques: A review,” International Journal of Engineering and Computer Science, vol. 4, April 2015.

[3] P. Bohannon, J. Parker, R. Rastogi, S. Seshadri, A. Silberschatz, and S. Sudarshan, “Distributed multi-level recovery in a main memory database,” Proc. Fourth Intl Conf. Parallel and Distributed Information Systems, 1996.

[4] M. Xie, H. Zhu, Y. Feng, and G. Hu, “Tracking and repairing damaged databases using before image table,” IEEE Japan-China Joint Workshop on Frontier of Computer Science and Technology, 2008.

[5] M. Mahmoodi, A. Baraani, and M. R. Khayyambashi, “Recovery time improvement in the mobile database systems,” Institute of Electrical and Electronics Engineers (IEEE), 2009.

[6] B. Panda and K. A. Haque, “Extended data dependency approach: A robust way of rebuilding database,” Proc. ACM Symp. Applied Computing (SAC 02), pp. 446–452, 2002.

[7] G. Li and L. Shu, “Design and evaluation of a low-latency checkpointing scheme for mobile computing systems,” The Computer Journal, vol. 49, no. 5, 2006.

[8] R. Koo and S. Toueg, “Checkpointing and rollback recovery for distributed systems,” IEEE Trans. Softw. Eng., vol. 13, no. 1, pp. 23–31, 1987.

[9] R. Prakash and M. Singhal, “Low-cost checkpointing and failure recovery in mobile computing systems,” IEEE Transactions on Parallel and Distributed Systems, vol. 7.

[10] Y. Rajesh and B. Panda, “Transaction fusion: A model for data recovery from information attacks,” journal of Intelligent Information Systems Attacks, vol. 23, Nov. 2004.

[11] C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, and P. Schwarz, “Aries: A transaction recovery method supporting fine granularity locking and partial rollbacks using write-ahead logging,” ACM Trans. Database Systems, vol. 17, no. 1, pp. 94–162, 1992.

[12] T. Chiueh and D. Pilania, “Design, implementation, and evaluation of a repairable database management system,” Proc. Annual Computer Security Applications Conference (ACSAC 04), pp. 179–188, Dec. 2004.

[13] P. Broessler, G. Weikum, C. Hasse and P. Muth, “Multi-level recovery,” (A.1) Proc. ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems, pp. 109–123, June 1990.

[14] H. Jagadish, A. Silberschatz, and S. Sudarshan, “Recovering from main memory lapses,” Proc. Intl Conf. Very Large Databases, 1993.

[15] V. Kumar and A. Burger, “Performance measurement of main memory database recovery algorithms based on update-in-place and shadow approaches,” IEEE Transactions on Knowledge and Data Engineering, 1992.