A method and system for maintaining data integrity utilizes a log file organized into slots with the number slots never exceeding the largest number of concurrent transactions updating the database comprising: creating a log file (50) for storing a log record corresponding to transaction-in-progress; request locking of all database records involved in transaction (53); identifying a slot available after all database records involved in the slots have been locked (58); reusing the slots to store log records in order that they become available (90).
FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

<table>
<thead>
<tr>
<th>Code</th>
<th>Country</th>
<th>Code</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM</td>
<td>Armenia</td>
<td>GB</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>AT</td>
<td>Austria</td>
<td>GE</td>
<td>Georgia</td>
</tr>
<tr>
<td>AU</td>
<td>Australia</td>
<td>GN</td>
<td>Guinea</td>
</tr>
<tr>
<td>BB</td>
<td>Barbados</td>
<td>GR</td>
<td>Greece</td>
</tr>
<tr>
<td>BE</td>
<td>Belgium</td>
<td>HU</td>
<td>Hungary</td>
</tr>
<tr>
<td>BF</td>
<td>Burkina Faso</td>
<td>IE</td>
<td>Ireland</td>
</tr>
<tr>
<td>BG</td>
<td>Bulgaria</td>
<td>IT</td>
<td>Italy</td>
</tr>
<tr>
<td>BJ</td>
<td>Benin</td>
<td>JP</td>
<td>Japan</td>
</tr>
<tr>
<td>BR</td>
<td>Brazil</td>
<td>KE</td>
<td>Kenya</td>
</tr>
<tr>
<td>BY</td>
<td>Belarus</td>
<td>KG</td>
<td>Kyrgyzstan</td>
</tr>
<tr>
<td>CA</td>
<td>Canada</td>
<td>KP</td>
<td>Democratic People's Republic of Korea</td>
</tr>
<tr>
<td>CF</td>
<td>Central African Republic</td>
<td>KR</td>
<td>Republic of Korea</td>
</tr>
<tr>
<td>CG</td>
<td>Congo</td>
<td>KZ</td>
<td>Kazakhstan</td>
</tr>
<tr>
<td>CH</td>
<td>Switzerland</td>
<td>LIE</td>
<td>Liechtenstein</td>
</tr>
<tr>
<td>CI</td>
<td>Côte d'Ivoire</td>
<td>LK</td>
<td>Sri Lanka</td>
</tr>
<tr>
<td>CM</td>
<td>Cameroon</td>
<td>LR</td>
<td>Liberia</td>
</tr>
<tr>
<td>CN</td>
<td>China</td>
<td>LT</td>
<td>Lithuania</td>
</tr>
<tr>
<td>CS</td>
<td>Czechoslovakia</td>
<td>LU</td>
<td>Luxembourg</td>
</tr>
<tr>
<td>CZ</td>
<td>Czech Republic</td>
<td>LV</td>
<td>Latvia</td>
</tr>
<tr>
<td>DE</td>
<td>Germany</td>
<td>MC</td>
<td>Monaco</td>
</tr>
<tr>
<td>DK</td>
<td>Denmark</td>
<td>MD</td>
<td>Republic of Moldova</td>
</tr>
<tr>
<td>EE</td>
<td>Estonia</td>
<td>MG</td>
<td>Madagascar</td>
</tr>
<tr>
<td>ES</td>
<td>Spain</td>
<td>ML</td>
<td>Mali</td>
</tr>
<tr>
<td>FI</td>
<td>Finland</td>
<td>MN</td>
<td>Mongolia</td>
</tr>
<tr>
<td>FR</td>
<td>France</td>
<td>MR</td>
<td>Mauritania</td>
</tr>
<tr>
<td>GA</td>
<td>Gabon</td>
<td>MW</td>
<td>Malawi</td>
</tr>
<tr>
<td>MX</td>
<td>Mexico</td>
<td>NE</td>
<td>Niger</td>
</tr>
<tr>
<td>NL</td>
<td>Netherlands</td>
<td>NO</td>
<td>Norway</td>
</tr>
<tr>
<td>NZ</td>
<td>New Zealand</td>
<td>PL</td>
<td>Poland</td>
</tr>
<tr>
<td>PT</td>
<td>Portugal</td>
<td>RO</td>
<td>Romania</td>
</tr>
<tr>
<td>RU</td>
<td>Russian Federation</td>
<td>SD</td>
<td>Sudan</td>
</tr>
<tr>
<td>SE</td>
<td>Sweden</td>
<td>SG</td>
<td>Singapore</td>
</tr>
<tr>
<td>SI</td>
<td>Slovenia</td>
<td>SK</td>
<td>Slovakia</td>
</tr>
<tr>
<td>SN</td>
<td>Senegal</td>
<td>SZ</td>
<td>Swaziland</td>
</tr>
<tr>
<td>TD</td>
<td>Chad</td>
<td>TG</td>
<td>Togo</td>
</tr>
<tr>
<td>TJ</td>
<td>Tajikistan</td>
<td>TT</td>
<td>Trinidad and Tobago</td>
</tr>
<tr>
<td>UA</td>
<td>Ukraine</td>
<td>UG</td>
<td>Uganda</td>
</tr>
<tr>
<td>US</td>
<td>United States of America</td>
<td>UZ</td>
<td>Uzbekistan</td>
</tr>
<tr>
<td>VN</td>
<td>Viet Nam</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SYSTEM FOR MAINTENANCE OF DATABASE INTEGRITY

BACKGROUND OF THE INVENTION

The present invention relates generally to maintaining database integrity in the event of a system failure and more particularly relates to process for creating a system log to facilitate recovery processes.

There are many methods utilized to maintain database integrity in the event of a system failure. Fundamental to many methods is a programmatic construct called a transaction which can be an explicitly delimited operation or set of operations that changes the content of a database from one consistent state to another.

The database operations within a transaction are treated as single unit. Either all changes made by the transaction to the database are made permanent (the transaction is committed) on none of the changes to the database are made permanent (the transaction is aborted). If a failure occurs during the execution of a transaction, whatever partial changes were made to the database are undone automatically, thus leaving the database in a consistent state.

If the transaction affects files or database records in a database, then, before the transaction permanently commits its changes to the database, information about the database records affected by the transaction is written to an audit trail file.

To facilitate a complete recovery of a database in the event of a catastrophic failure a complete journal of all transactions is maintained. Complex disc management protocols such as handling overflows by wrap-around and archiving must be utilized.
SUMMARY OF THE INVENTION

A present invention is a system and process for the maintenance of database integrity in the presence of multiple failures that utilizes a LOG file organized into slots to store LOG records, identified by increasing log sequence numbers, corresponding to transactions performed on the database.

According to one aspect of the invention, prior to writing the details of a transaction to a slot all database records involved in the transaction are locked to assure independence of concurrent transactions. The database records involved in a transaction are unlocked when the transaction is complete. Slots are available when the corresponding transaction is complete and unavailable if the transaction is currently in progress and are reused in the order that they become available. Thus, the number of slots never exceeds the largest number of concurrent transactions updating the database and complex disc management processes such as wrap around and archiving are not required.

According to another aspect of the invention, during recovery LOG records are dropped from further consideration if there is a later LOG record that involves one of the same database records. The remaining LOG records are processed in LOG sequence number order. For each LOG record, if the corresponding transaction was only partially complete then it is backed out.

Other features and advantages of the invention will be apparent in view of the following detailed description and appended drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a data processing system;

Fig. 2 is a schematic diagram of a LOG file;

Fig. 3 is a symbolic diagram of a FIFO;

Fig. 4 is a schematic diagram of two transactions;

Fig. 5 is a flow chart of a process for creating a LOG file;
Fig. 6 is a flow chart of a recovery process; and Fig. 7 is a symbolic diagram of two transactions.

DESCRIPTION OF THE PREFERRED EMBODIMENT

A preferred embodiment of the invention is a fault tolerant Name Server process, operating in a UNIX environment, that recovers from failures to maintain database integrity. The Name Server controls the UNIX filename hierarchy, mapping UNIX filenames down to disc addresses. A database is maintained with records, called inodes in UNIX parlance, with other records pointing to files and directory names. The result of a lookup are the disc address of the file and the file information (size, dates, etc.).

In particular, Data Description Language (DDL) transactions may involve several database records. Thus, if a failure occurs during a DDL transaction, the recovery process must resolve the database to a consistent state by either backing out updates already implemented or completing partial updates.

Fig. 1 is a block diagram of a data processing system in which the recovery process can be implemented. The system includes first and second CPUs 12 and 14 coupled by an Interprocessor BUS (IPB) 16. First and second data busses 18 and 20 are coupled to the I/O ports of the CPUs 12 and 14 and to disc storage units 22 and 24 via disc controllers 26-32 and cables 34.

As depicted in Fig. 2, the LOG file 36 used in the present recovery process is an unstructured file organized into 2048-byte slots. Each slot is referred to by a slot index identifying the relative position of the slot in the file, e.g., the slot index for the first slot is 0, for second slot is 1, etc.

At any given time, a slot in the LOG file is either occupied or available. If a particular slot is occupied then there is a DDL transaction in progress, the details of which are recorded in the particular slot. If a particular slot is available, then the last transaction to use the slot has been completed. When the name server needs to use a LOG slot, it
first determines if a slot is available; if not, a new slot is added to the end of the LOG.

The occupied or available status of each slot is kept in name server memory. Accordingly, when a slot if freed there is no need to record the availability of the slot on disk.

As depicted symbolically in Fig. 3, as a slot becomes available its slot index is added to the bottom of a FIFO stack 38. In Fig. 3, when a slot becomes available its slot index is added to the bottom of the stack and when a slot is needed to record the details of a DDL transaction a slot index of a slot to be used is provided from the top of the stack.

There are several key requirement for recovery utilizing the LOG file. The first requirement is that it must be possible to unambiguously determine whether, prior to the name server failure, a given DDL transaction began and if it completed. The second requirement is that concurrent DDL transactions must be independent in the sense that the concurrent transactions do not involve the same database records. For example, as depicted in Fig. 4, suppose that DDL transaction A 40 is a long transaction involving inode X and the DDL transaction B 42 is a short transaction involving the same inode. Further, suppose that A starts before B and the B finishes before A. If a failure occurs between the time that B completes and the time the A completes, then backing out transaction A will likely restore X to its state prior to the start of B even though B is complete prior to the failure.

The process for writing LOG records to slots in the LOG file will now be described with reference to the flow chart depicted in Fig. 5. Each LOG record written to the LOG 38 has a Log Sequence Number (LSN) where the LSNs are initialized 50 to zero and increase monotonically as LOG records are written to the LOG 38.

A current DDL transaction is commenced 52 and the DDL operation requests 53 a lock of the database records involved in the current transaction. It is then tested 54 whether any of the involved database records are currently
locked. If any of the database records are currently locked by another active DDL transaction then the current transaction is suspended 56 until the lock is released by the other transaction. If none of the involved database records are locked then the request is granted 57. Thus, the second requirement that the concurrent DDL transactions be independent is satisfied.

Once the inodes involved in the transaction are locked it is tested 58 whether an available slot is in the FIFO. If no slot is available then the name server adds 60 another slot to the LOG 36 and adds the slot index of the added slot to the bottom of the FIFO stack 38. Once there is an available slot in the FIFO stack 38 then the details of the DDL transaction are written 62 to a LOG record in the slot and LSN is incremented 64.

When the DDL transaction is completed the inodes involved in the DDL transaction are unlocked 66, the slot is made available, and the slot index of the slot is added 68 to the bottom of FIFO stack 38. The unlocking of the slot upon completion of the DDL transaction assures satisfaction of the second requirement.

The recovery process will now be described with reference to the flow chart depicted in Fig. 6. First, the LOG is sorted 70 in forward order of increasing LSNs. The LOG is the read sequentially 74; each LOG record is dropped from further consideration 78 if there is a later LOG record (one with a higher LSN) involved with one of the same inodes 76. Thus, irrelevant LOG records are not processed during recovery processing. The relevant LOG records are then scanned in forward order and partially completed DDL transactions are backed out.

The effect of the recovery process on the DDL transactions A and B depicted in Fig. 4 will now be described. Those transactions are not independent because the transactions are concurrently operating on the same inode X. In the procedure described with reference to Fig. 4, the inodes for DDL A are locked prior to writing the details of DDL A to the LOG with an LSN having a value of N. Thus, DDL B
can not begin until inode X is unlocked at the completion of DDL A. When DDL B begins inode X is again locked prior to writing the details of DDL B to a LOG record having an LSN greater than N. These transactions are depicted in Fig. 7.

In this case, during a recovery process for a failure occurring during the transaction DDL B, the LOG record having an LSN equal to N will be ignored because there is a later LOG record, the LOG record for DDL B, involving inode X. Thus, the irrelevant LOG record for DDL A is ignored because that transaction had been completed and a backout of the partially completed DDL B is performed.

Additionally, the reuse of available slots on a FIFO basis assures that LOG records for all active DDL transactions are not overwritten and also assures that the number of slots in log file never exceeds the number of LOG records equal to the largest number of concurrent DDL transactions.

The invention has now been described with reference to the preferred embodiments. However, alternatives and substitutions will now be apparent to persons of ordinary skill in the art. In particular, although the preferred embodiment is describe in a UNIX name server environment, the process described applies equally well to other environments to preserve a single applications's database integrity in the face of multiple failures of the system on which the application is running once that application is restarted. The UNIX specific inodes would be replaced with other keys or pointers identifying database records involved in a transaction. Accordingly, it is not intended to limit the application except as provided by the appended claims.
WHAT IS CLAIMED IS:

1. An improved process for preserving a database in the event of a failure occurring during a transaction involving database records, said process comprising the steps of:

   creating a log file, including a plurality of slots each identified by a unique slot index, for storing a LOG record corresponding to an transaction-in-progress, with each LOG record identified by a unique monotonically increasing log sequence number (LSN) and with each slot storing details of single transaction;

   locking all database records involved with a given transaction prior to writing a LOG record for the given transaction in the log file so that transactions concurrently active cannot operate on same database records thereby assuring independence of concurrent transactions;

   unlocking all database records involved with said given transaction when the transaction is completed;

   identifying a slot as unavailable if any database records involved with a corresponding transaction recorded in the slot have been locked;

   identifying a slot as an available slot after all database records involved with a corresponding transaction recorded in the slot have been unlocked;

   reusing slots to store LOG records of subsequently active transactions in the order that slots become available so that the slots are reused on a first-in-first-out basis thereby assuring that if there is a LOG record in the LOG file involving a given database record then the LOG file contains a LOG record corresponding to the last transaction involving the given database record; and

   during a recovery transaction, eliminating a given LOG record from further consideration if there is a LOG record with a higher LSN involving a same database record as involved with the transaction corresponding to the given LOG record.
2. The process of claim 1 wherein said step of
eliminating further comprises the steps of:
   sorting the log file in forward order of increasing
   LSNs'; and
   for a given LOG record corresponding to a
transaction involving a particular database record, scanning
forward from the given LOG record to determine whether a LOG
record having a higher LSN corresponds to an transaction
involving the particular database record.
2/3

A

INITIALIZE LSN

BEGIN CURRENT DDL TRANSACTION

REQUEST LOCKING OF ALL DATABASE RECORDS INVOLVED WITH CURRENT DDL TRANSACTION

ARE ANY REQUESTED DATABASE RECORDS LOCKED?

YES

NO

SUSPEND CURRENT DDL TRANSACTION

GRANT REQUESTED AND LOCK DATABASE RECORDS

IS THERE AN AVAILABLE SLOT IN THE FIFO?

YES

NO

CREATE A NEW SLOT AND ADD SLOT INDEX TO BOTTOM OF FIFO

WRITE DETAILS OF DDL TRANSACTION TO SLOT AT THE TOP OF THE FIFO

LSN ← LSN + 1

COMPLETE DDL AND UNLOCK INVOLVED DATABASE RECORDS

MAKE SLOT AVAILABLE AND ADD TO BOTTOM OF FIFO

A

FIG. 5.
SUBSTITUTE SHEET (RULE 26)
3/3

SORT LOG IN FORWARD ORDER OF INCREASING LSNs

SEND LOG SEQUENTIALLY

IS A LATER LOG RECORD INVOLVED WITH THE SAME NODES?

NO

YES

IGNORE LOG RECORD

FIG. 6.

FIG. 7.

LOCK X

UNLOCK X

DDL A

DDL B

LOCK X

UNLOCK X

SUBSTITUTE SHEET (RULE 26)
### INTERNATIONAL SEARCH REPORT

**INTERNATIONAL SEARCH REPORT**

A. **CLASSIFICATION OF SUBJECT MATTER**
- IPC(6): G06F 17/40, 11/34
- US CL: 395/600, 650, 700, 575; 364/285.1, 285.2
- According to International Patent Classification (IPC) or to both national classification and IPC

B. **FIELDS SEARCHED**
- Minimum documentation searched (classification system followed by classification symbols)
  - U.S.: 395/600, 650, 700, 575; 364/285.1, 285.2

- Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
- APS, DIALOG
  - search terms: log file, journal file, backup, restore, locking and unlocking database records

C. **DOCUMENTS CONSIDERED TO BE RELEVANT**

<table>
<thead>
<tr>
<th>Category*</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X, P</td>
<td>US, A, 5,455,944 (HADERLE ET AL.) 03 October 1995, col. 5, line 42 - col. 6, line 48 and col. 12, lines 33-46, fig 2.</td>
<td>1-2</td>
</tr>
</tbody>
</table>

See patent family annex.

Date of the actual completion of the international search: 20 MARCH 1996

Date of mailing of the international search report: 02 APR 1996

Name and mailing address of the ISA/US Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703) 305-3230

Authorized officer: [Signature]
CUAN PHAM
Telephone No. (703) 305-9600

Form PCT/ISA/210 (second sheet)(July 1992)