Modern techniques for transaction-oriented database recovery

Caetano Sauer
SOME HARVARD STUDENTS SAY AMERICA IS A BIGGER THREAT TO WORLD PEACE THAN ISIS
My pleas

1. Demand on-demand recovery
2. Make the log great again
Demand on-demand recovery
2. **REDO** scan

3. **UNDO** scan

**ARIES**

**Recovery info**

- Pages requiring redo
- Transactions requiring undo

**Buffer pool**

**Database**

1. **Log analysis**

**LOG**

**system mostly unavailable during three phases**
ARIES restart

Recovery info

- **Pages** requiring redo
- **Transactions** requiring undo

1. **Log analysis**
2. **REDO scan**
3. **UNDO scan**

System mostly unavailable during three phases
Instant recovery

Recovery info

Running txn.

fix(B)

Buffer pool

Buffer pool

B

Transactions requiring undo + locks

Pages requiring redo

LOG

per-page log chains

redo and undo on demand, without waiting for log scans

same recovery actions, different schedule

Aborting txn. (pre-failure)

lock(f)

lock conflict!

(lock(f))

(post-failure)
## Failure classes

<table>
<thead>
<tr>
<th>Failure class</th>
<th>Loss</th>
<th>Typical cause</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction</td>
<td>Single-transaction progress</td>
<td>Deadlock, constraint violation</td>
<td>Rollback</td>
</tr>
<tr>
<td>System</td>
<td>Server process (in-memory state)</td>
<td>Software fault, power loss</td>
<td>Restart</td>
</tr>
<tr>
<td>Media</td>
<td>Persistent database contents</td>
<td>Hardware fault</td>
<td>Restore</td>
</tr>
<tr>
<td>Single page</td>
<td>Local integrity</td>
<td>Partial writes, wear-out</td>
<td>Repair</td>
</tr>
</tbody>
</table>
Make the log great again
“... a DBMS is really two DBMSs, one managing the database as we know it and a second one managing the log.”

Michael Stonebraker
The log as a partitioned B-tree

like an LSM, but with page identifiers and their log records
The log as a partitioned B-tree (implementation)
Merging

Before:

After:
FineLine

Write-ahead logging:

Data structure → read → Database → write → Log records

volatile  persistent

append

FineLine:

Data structure

Database

fetch

Log index

append

volatile  persistent

reorg.
FineLine recovery

Data structure

Log records

Indexed log

volatile persistent
FineLine recovery
FineLine recovery
FineLine recovery

Indexed log

fetch

volatile
persistent
FineLine

In-memory database

- nodes recovered automatically during fetch
- volatile structures = in-memory database
- no undo, no dirty pages, no checkpoints, no offline log scans
In-memory databases
Log-structured storage
DBMS Buffer mgmt.
Physiological logging

FineLine
My pleas

1. Demand on-demand recovery
2. Make the log great again
Standing on the shoulders of Giants
5.8.6.2. How to Get Perfectly Reliable Data Management

watch this space (a coming attraction)

Thank you!

csauer@tableau.com