Inability to undo recovery - By enforcing that all integrity should
insiders recover their may be done purely

1. Give them is no explanation on when the

3. For instance add "end" to log

do something on your own. The device

2. Edit scan: perform the walks for

I. Construct less (the hit of poorly

Recovery procedure

2. cell - student will arrive - immediately

1. Penalty democracies, indeed will survive

At time of event

Db in Vio 4.1 + 42999
Summary:

1. Problem: Obtained by some process some application of the
   result is not correct if every result is
   correct.

2. Correction: Requirement for correctness of execution of code.

3. Correction 2: Requirement for correctness of output.

4. Checkpoint:
   - Initial: T
   - Intermediate: T
   - Final: T
   - Next:

---

Addendum:

- Log can be viewed.
- Checkpoint:
  - By writing each instruction.
  - After applying the log.

- Log redo recovery:
  - Allows full roll-forward.
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>E1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Quite resourceful.

Return id

cure until previous id - outcome - record size.

Procedure BEGIN-TRANSACTION(C1)

RECORD (PENDING)

i, r -> NEW-OUTCOME-RECORD (PENDING)

Deal with i in sequence until previous.id - outcome - record size.

If not PENDING, then continue.
Mark point discipline is "no transaction can begin reading its input until the preceding transaction has reached its mark point or is no longer pending."

procedure READ_CURRENT_VALUE (data_id, tid)
    starting at end of data-id repeat until beginning
    get previous version of data_id
    last-modifier := this-then-skip := v. action_id
    if last-modifier >= tid then skip v
    wait until (last-modifier, outcome_record.state != PENDING)
    if last-modifier, outcome_record.state = COMMITTED
        then return v.value
    else skip v
proc Make

sign ("error")

if u.action.id = bid

prev versions 2 oth. vid

shunt & send 2 other vid r & b.

proc write value (vid. deh. id, new val).

end proc.

if u.action.id = bid

clear value to other id.

append new version 2 the other id.

then sign ("error")

if (tid.outcome = reced. mark = share = variance)

proc new version (ref data id, tid)
forall 

procedure NEW_OUTCOME RECORD (STATE) 
allocate id. outcome - record. state 

id -> TICKET (outcome - record - segment) 
Acquire (outcome - record - lock) 
Procedure NEW - OUTCOME - RECORD (STATE) 

end if 

return id 

( pending = pending + 1 ) 
get pending 

procedure BEGIN TRANSACTION 

procedure END TRANSACTION
Mac's Point - announce (may 14)

Mark Debit & Credit & Accrual
Create new User(s)

Procedure: Transfer

End User

Cfr. User - State & Marked

Procedure: Mark - Point - Announce (file)

End