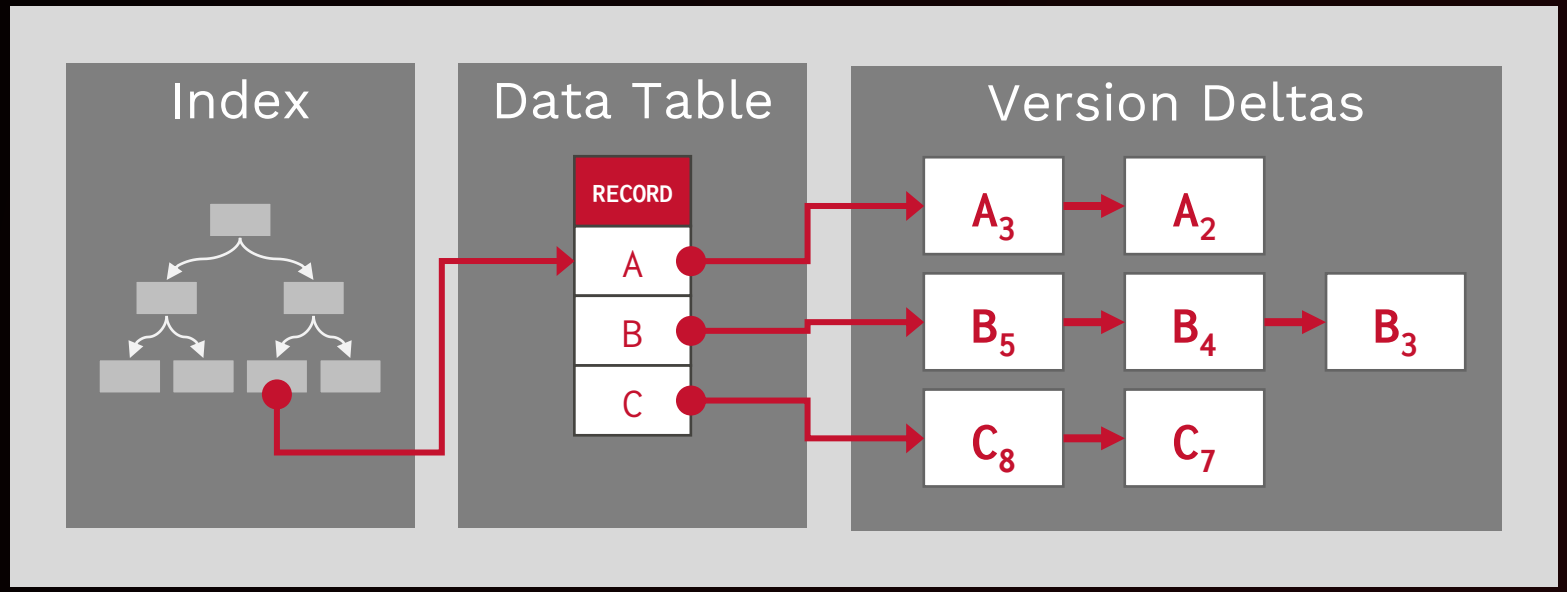
Everything is a Transaction

Unifying Logical Concurrency Control and Physical Data Structure Maintenance in Database Management Systems – CIDR 2021

@andy_pavlo

Motivation



Arrays

Linked Lists

Hash Tables

Trees

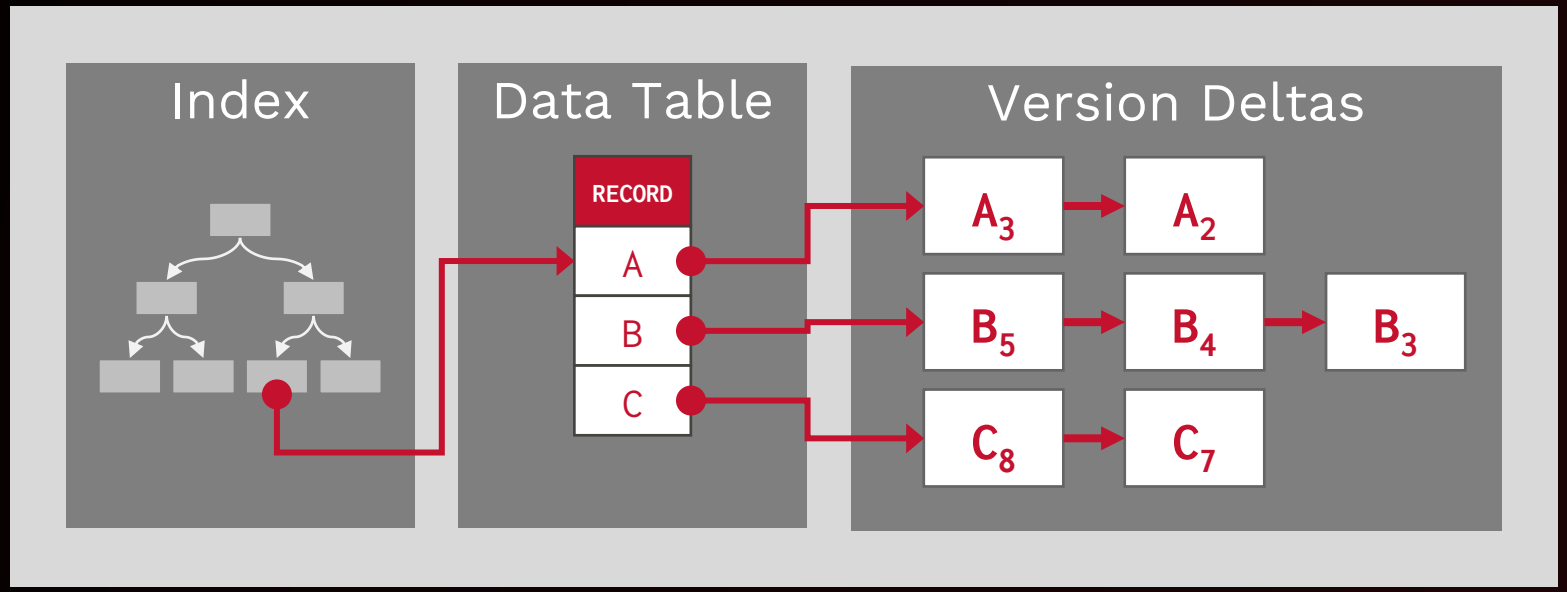
Motivation

Transaction

```
BEGIN;
 DELETE FROM table
  WHERE key = a;
COMMIT;
```

Transaction

```
BEGIN;
 SELECT FROM table
  WHERE key = a;
COMMIT;
```



- Arrays
- Linked Lists
- Hash Tables
- Trees

Motivation

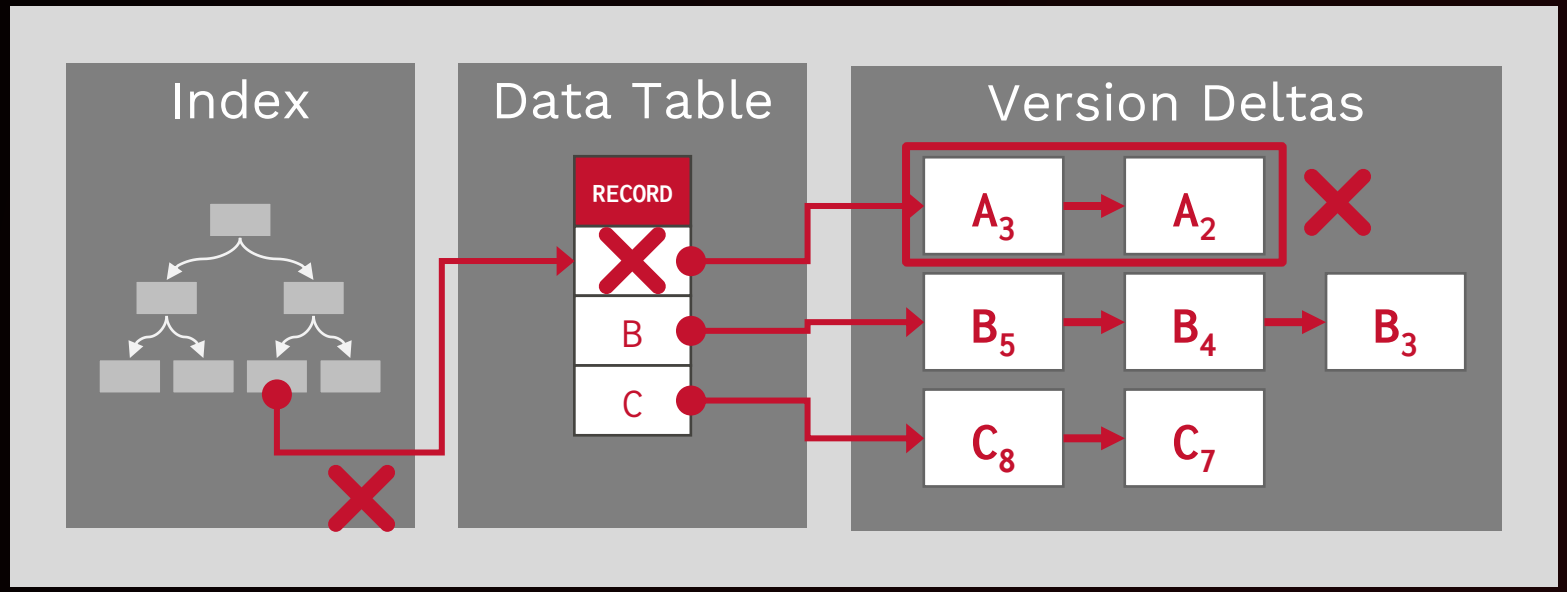
Transaction

```
BEGIN;
 DELETE FROM table
  WHERE key = a;
COMMIT;
```

Transaction

```
BEGIN;
 SELECT FROM table
  WHERE key = a;
COMMIT;
```

Must **defer** the removal of physical data until no active txn able to see it.



- Arrays
- Linked Lists
- Hash Tables
- Trees

Deferred Action Framework (DAF)

Asynchronous execution framework for internal maintenance actions in a multi-versioned DBMS.

Key Idea: DAF executes actions as transactions with the same visibility mechanisms.

- Single API call: `DEFER(action)`
- Each action is tagged with txn's commit timestamp.
- Invoke an action when there are no transactions with a start timestamp smaller than tagged timestamp.

Deferred Action Framework (DAF)

Transaction

```
BEGIN;
  DELETE FROM table
   WHERE key = a;
COMMIT;
```

Action Queue

OLDEST  = t0

Deferred Action Framework (DAF)

Transaction

```
BEGIN;   = t1
  DELETE FROM table
    WHERE key = a;
COMMIT;
```

Action Queue

OLDEST = t0

Deferred Action Framework (DAF)

Transaction

```
BEGIN;   = t1
  DELETE FROM table
    WHERE key = a;
COMMIT;
```

Action Queue

OLDEST = t1

Deferred Action Framework (DAF)

Transaction

```
BEGIN;   = t1
 DELETE FROM table
  WHERE key = a;
COMMIT;
```

DEFER(Unlink Version Chain)

DEFER(Delete Index Key=a)

Generated Actions

Action Queue

OLDEST = t1

Deferred Action Framework (DAF)

Transaction

```
BEGIN;    = t1
 DELETE FROM table
  WHERE key = a;
COMMIT;
```

Transaction

```
BEGIN;    = t2
 SELECT FROM table
  WHERE key = a;
COMMIT;
```

DEFER(Unlink Version Chain)

DEFER(Delete Index Key=a)

Generated Actions

Action Queue

OLDEST = t2

Deferred Action Framework (DAF)

Transaction

```
BEGIN;    = t1
 DELETE FROM table
  WHERE key = a;
COMMIT;
```

Transaction

```
BEGIN;    = t2
 SELECT FROM table
  WHERE key = a;
COMMIT;
```

DEFER(Unlink Version Chain)

DEFER(Delete Index Key=a)

Generated Actions

Action Queue

OLDEST = t2

Deferred Action Framework (DAF)

Transaction

```
BEGIN;    = t1
  DELETE FROM table
    WHERE key = a;
COMMIT;   = t3
```

Transaction

```
BEGIN;    = t2
  SELECT FROM table
    WHERE key = a;
COMMIT;
```

DEFER(Unlink Version Chain)

DEFER(Delete Index Key=a)

Generated Actions

Action Queue

OLDEST = t3

Deferred Action Framework (DAF)

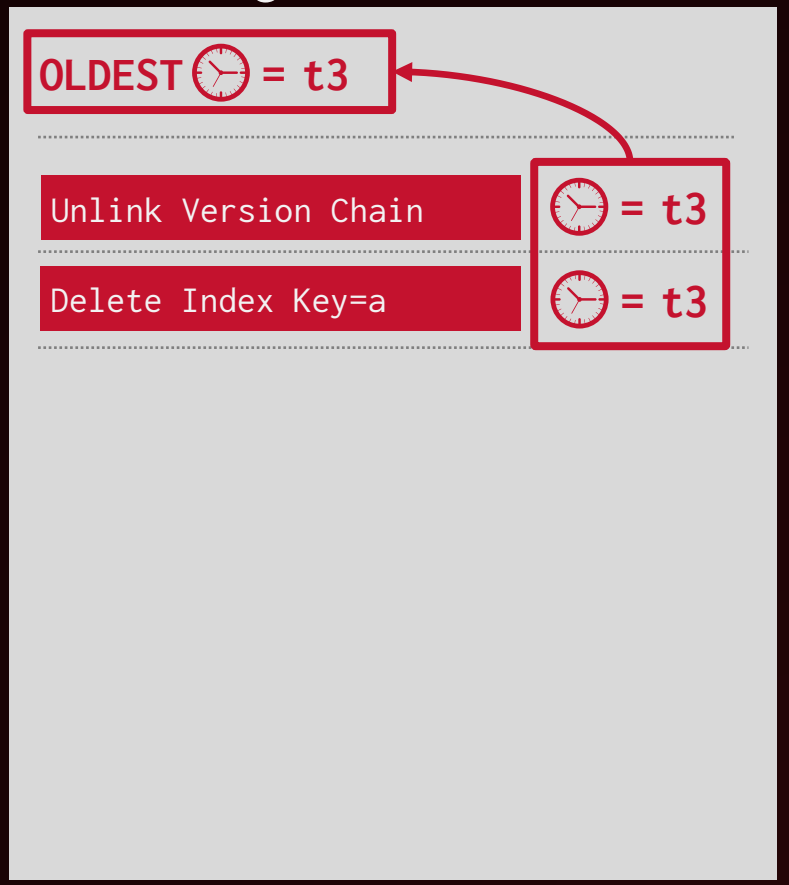
Transaction

```
BEGIN;    = t1
  DELETE FROM table
    WHERE key = a;
COMMIT;   = t3
```

Transaction

```
BEGIN;    = t2
  SELECT FROM table
    WHERE key = a;
COMMIT;
```

Action Queue



Deferred Action Framework (DAF)

Transaction

```
BEGIN;    = t1
  DELETE FROM table
    WHERE key = a;
COMMIT;   = t3
```

Transaction

```
BEGIN;    = t2
  SELECT FROM table
    WHERE key = a;
COMMIT;   = t4
```

Action Queue

OLDEST = t4

Unlink Version Chain	= t3
Delete Index Key=a	= t3

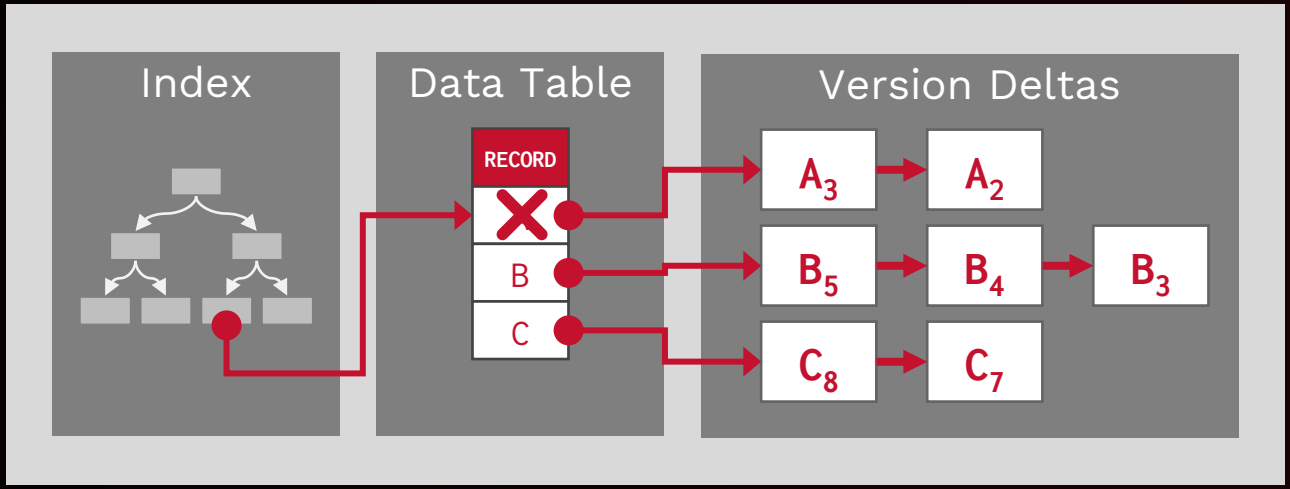
Deferred Action Framework (DAF)

Transaction

```
BEGIN;    = t1
  DELETE FROM table
   WHERE key = a;
COMMIT;   = t3
```

Transaction

```
BEGIN;    = t2
  SELECT FROM table
   WHERE key = a;
COMMIT;   = t4
```



Action Queue

OLDEST = t4

Unlink Version Chain	= t3
Delete Index Key=a	= t3

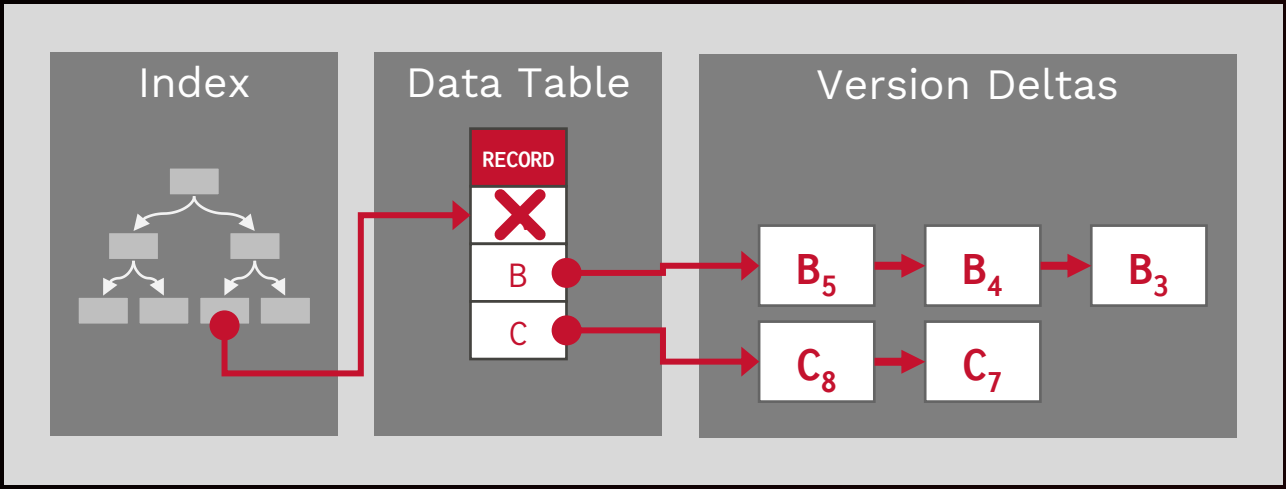
Deferred Action Framework (DAF)

Transaction

```
BEGIN;    = t1
  DELETE FROM table
   WHERE key = a;
COMMIT;   = t3
```

Transaction

```
BEGIN;    = t2
  SELECT FROM table
   WHERE key = a;
COMMIT;   = t4
```



Action Queue

OLDEST = t4

Unlink Version Chain	= t3
Delete Index Key=a	= t3

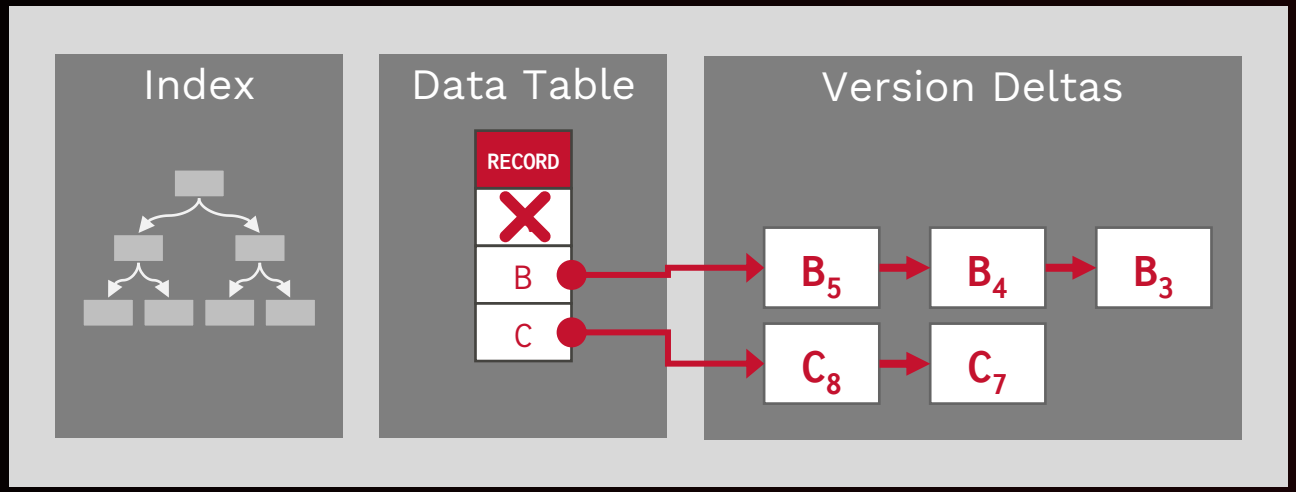
Deferred Action Framework (DAF)

Transaction

```
BEGIN;    = t1
  DELETE FROM table
   WHERE key = a;
COMMIT;   = t3
```

Transaction

```
BEGIN;    = t2
  SELECT FROM table
   WHERE key = a;
COMMIT;   = t4
```



Action Queue

OLDEST = t4

Delete Index Key=a = t3

Multi-Deferrals

Explicit ordering of concurrent actions.

- Example: action A drops a table; action B deletes a tuple in the table.
- Solution: Chained deferral of actions.

```
DEFER(DEFER(… DEFER(action)…))
```

Unwrap and reinsert multi-deferred action to queue with a later timestamp.

- Separate its execution with other concurrent actions.

Multi-Deferrals

Transaction

```
BEGIN;   = t1
  DROP TABLE table;
COMMIT;
```

Action Queue

OLDEST = t1

Multi-Deferrals

Transaction

```
BEGIN;   = t1
 DROP TABLE table;
COMMIT;
```

DEFER(DEFER(Delete Table + Index))

Action Queue

OLDEST = t1

Multi-Deferrals

Transaction

```
BEGIN;    = t1
 DROP TABLE table;
COMMIT;
```

Transaction

```
BEGIN;    = t2
 DELETE FROM table
  WHERE key = a;
COMMIT;
```

DEFER(DEFER(Delete Table + Index))

Action Queue

OLDEST = t2

Multi-Deferrals

Transaction

```
BEGIN;    = t1
 DROP TABLE table;
COMMIT;
```

Transaction

```
BEGIN;    = t2
 DELETE FROM table
  WHERE key = a;
COMMIT;
```

DEFER(DEFER(Delete Table + Index))

DEFER(Unlink Version Chain)

DEFER(Delete Index Key=a)

Action Queue

OLDEST = t2

Multi-Deferrals

Transaction

```
BEGIN;    = t1
  DROP TABLE table;
COMMIT;   = t3
```

Transaction

```
BEGIN;    = t2
  DELETE FROM table
    WHERE key = a;
COMMIT;
```

Action Queue

OLDEST = t3

DEFER(Delete Table + Index)

= t3

DEFER(Unlink Version Chain)

DEFER(Delete Index Key=a)

Multi-Deferrals

Transaction

```
BEGIN;          = t1
  DROP TABLE table;
COMMIT;         = t3
```

Transaction

```
BEGIN;          = t2
  DELETE FROM table
    WHERE key = a;
COMMIT;         = t4
```

Action Queue

OLDEST = t4

DEFER(Delete Table + Index) = t3

Unlink Version Chain = t4

Delete Index Key=a = t4

Multi-Deferrals

Transaction

```
BEGIN;      = t1
  DROP TABLE table;
COMMIT;     = t3
```

Transaction

```
BEGIN;      = t2
  DELETE FROM table
    WHERE key = a;
COMMIT;     = t4
```

Action Queue

OLDEST = t5

DEFER(Delete Table + Index) = t3

Unlink Version Chain = t4

Delete Index Key=a = t4

Multi-Deferrals

Transaction

```
BEGIN;    = t1
  DROP TABLE table;
COMMIT;   = t3
```

Transaction

```
BEGIN;    = t2
  DELETE FROM table
    WHERE key = a;
COMMIT;   = t4
```

Action Queue

OLDEST = t5

Unlink Version Chain	= t4
Delete Index Key=a	= t4
Delete Table + Index	= t5

Multi-Deferrals

Transaction

```
BEGIN;    = t1
  DROP TABLE table;
COMMIT;   = t3
```

Transaction

```
BEGIN;    = t2
  DELETE FROM table
    WHERE key = a;
COMMIT;   = t4
```

Action Queue

OLDEST = t5

Delete Index Key=a = t4

Delete Table + Index = t5

Multi-Deferrals

Transaction

```
BEGIN;      = t1
  DROP TABLE table;
COMMIT;     = t3
```

Transaction

```
BEGIN;      = t2
  DELETE FROM table
    WHERE key = a;
COMMIT;     = t4
```

At this time the table is not visible to any active txn and is safe to drop.

Action Queue

OLDEST = t6

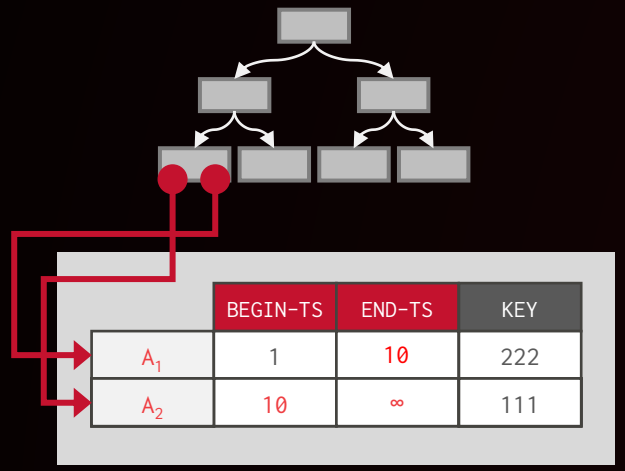
Delete Table + Index = t5

Applications

Index Cleaning

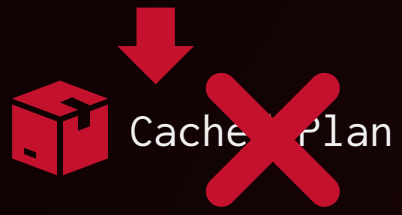
```
BEGIN;
 UPDATE table SET key=111
  WHERE key=222;
COMMIT;
```

Delete Index Key=222;



Cache Invalidation

```
PREPARE stmt1 AS
  SELECT a,b FROM table;
```

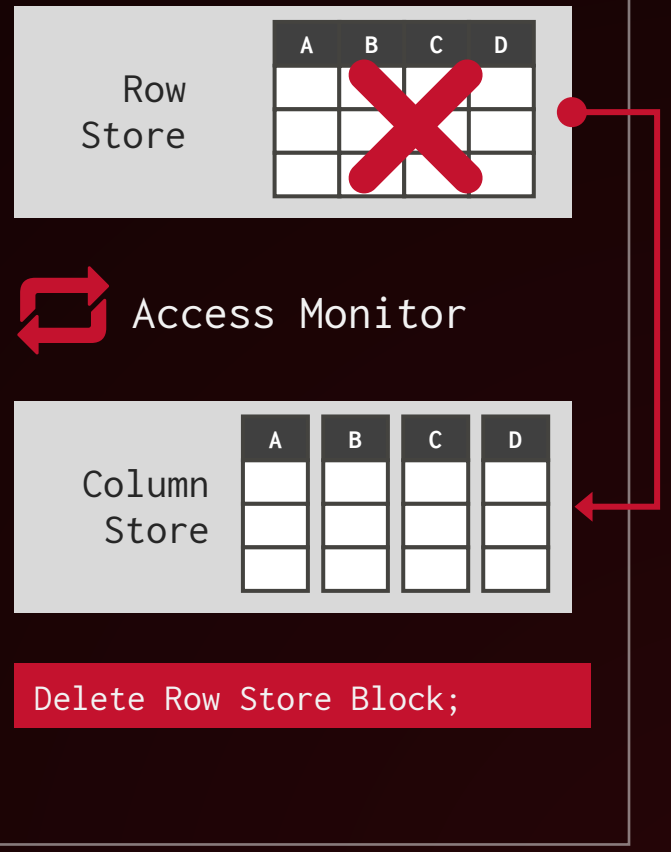



```
BEGIN;
 ALTER TABLE table
  DROP COLUMN b;
COMMIT;
```

Remove Column table.b;

Invalidate Plan Cache;

Data Transformation



Preliminary Results

Integrated DAF with NoisePage DBMS.

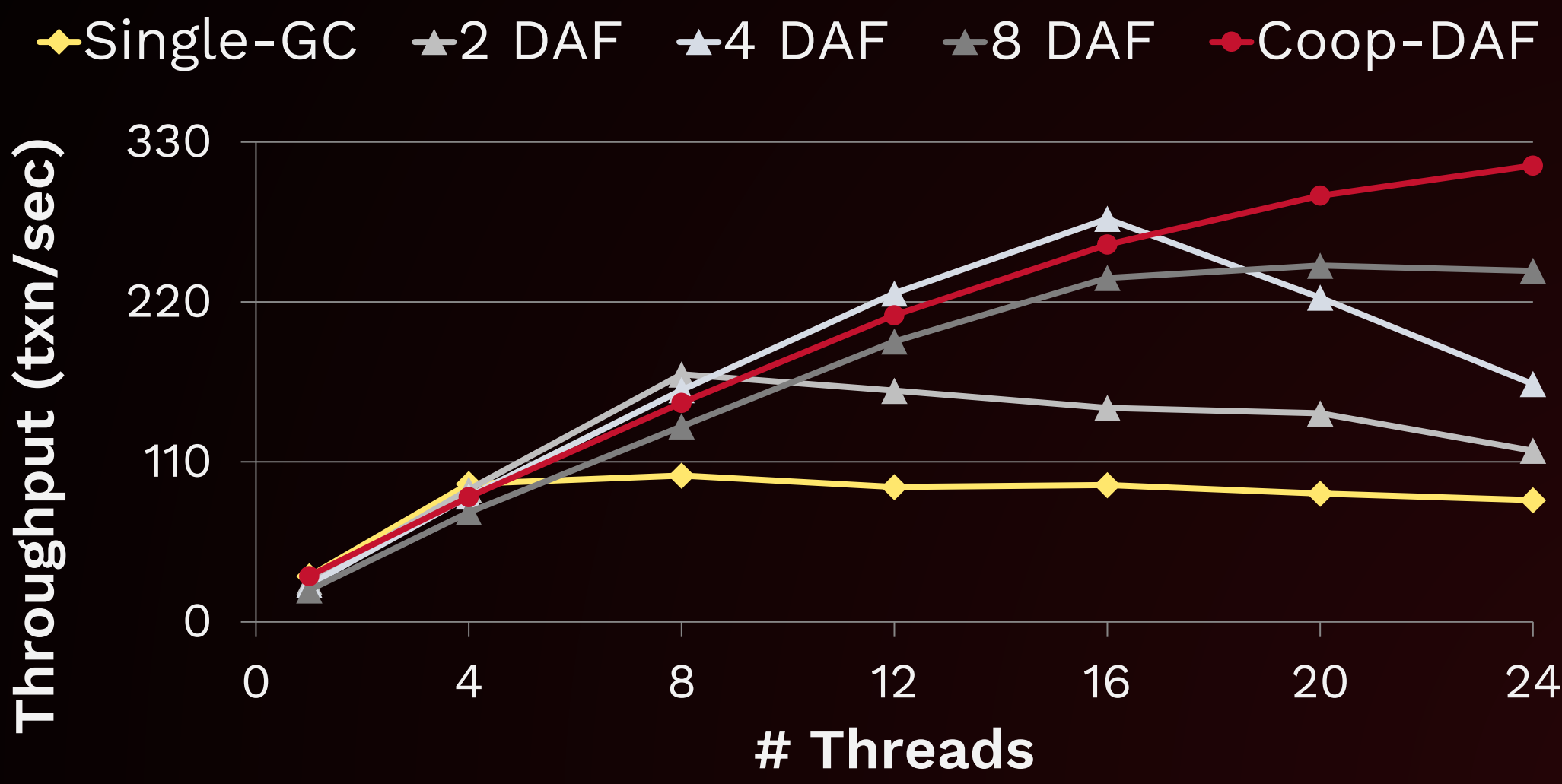
Measure MVCC GC scalability using TPC-C:

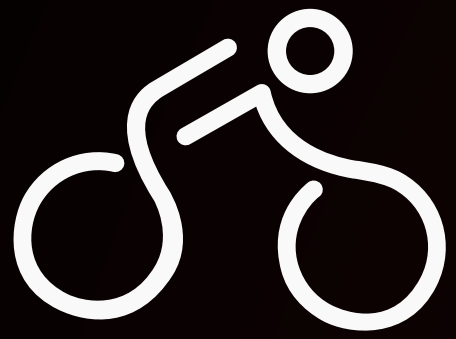
- One warehouse per worker thread.
- Stored Procedure API.
- Write-Ahead Logging Enabled.

Compare three system configurations:

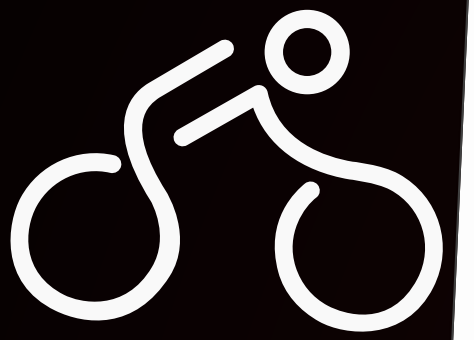
- Single GC thread
- Dedicated DAF threads
- Cooperative DAF threads

MVCC GC Scalability (TPC-C)





Peloton



222

cmu-db / **peloton** Archived

Unwatch 192 | Unstar 1.8k | Fork 591

Code | Issues 159 | Pull requests 31 | Actions | Projects | Wiki | Security | Insights | Settings

Added notice that the project is dead.

master

Browse files

apavlo committed on Mar 17, 2019 Verified

1 parent a96b376 commit 484d76df9344cb5c153a2c361c5d5018912d4cf4

Showing **1 changed file** with **8 additions** and **0 deletions**.

8 README.md

Unified | Split

```
@@ -6,6 +6,14 @@
6  6   [![Jenkins Status](http://jenkins.db.cs.cmu.edu:8080/job/peloton/job/master/badge/icon)](http://jenkins.db.cs.cmu.edu:8080
       /job/peloton/)
7  7   [![Coverage Status](https://coveralls.io/repos/github/cmu-db/peloton/badge.svg?branch=master)](https://coveralls.io/github/cmu-
       db/peloton?branch=master)
8  8
9    + ## UPDATE 2019-03-17
10   +
11   + The Peloton project is **dead**. We have abandoned this repository and moved on to build a new DBMS. There are a several engineering
       techniques and designs that we learned from this first system on how to support autonomous operations that we are doing a much better
       job at implementing in the second system.
12   +
13   + We will not accept pull requests for this repository. We will also not respond to questions or problems that you may have with
       running with this software.
```

noise
page

NoisePage Project

In-Memory HTAP DBMS

Postgres compatible (wire, SQL, catalog)

Apache Arrow compatible columnar storage

HyPer-style MVCC (snapshot isolation)

Hybrid Vectorization + Pipeline Query Codegen

JIT Query Compilation (DSL→OpCodes→LLVM)

Integrated self-driving components

NoisePage Projec

In-Memory HTAP DBMS

Postgres compatible (wir

Apache Arrow compatibl

HyPer-style MVCC (snap

Hybrid Vectorization + P

JIT Query Compilation (

Integrated self-driving c

Summary

DAF shows how to leverage logical data concurrency protocols for physical data structures.

Unifying the notion of visibility makes it easier to integrate new data structures and extend transactions to support maintenance operations.

Acknowledgements

END

https://noise.page