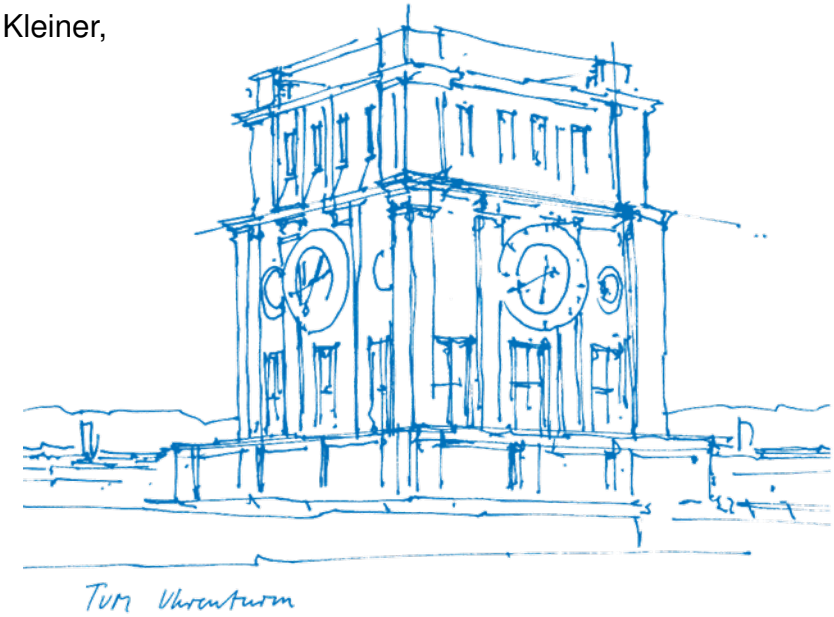
Chair III: Database Systems
Chair XXV: Data Science and Engineering
Department of Informatics
Technical University of Munich

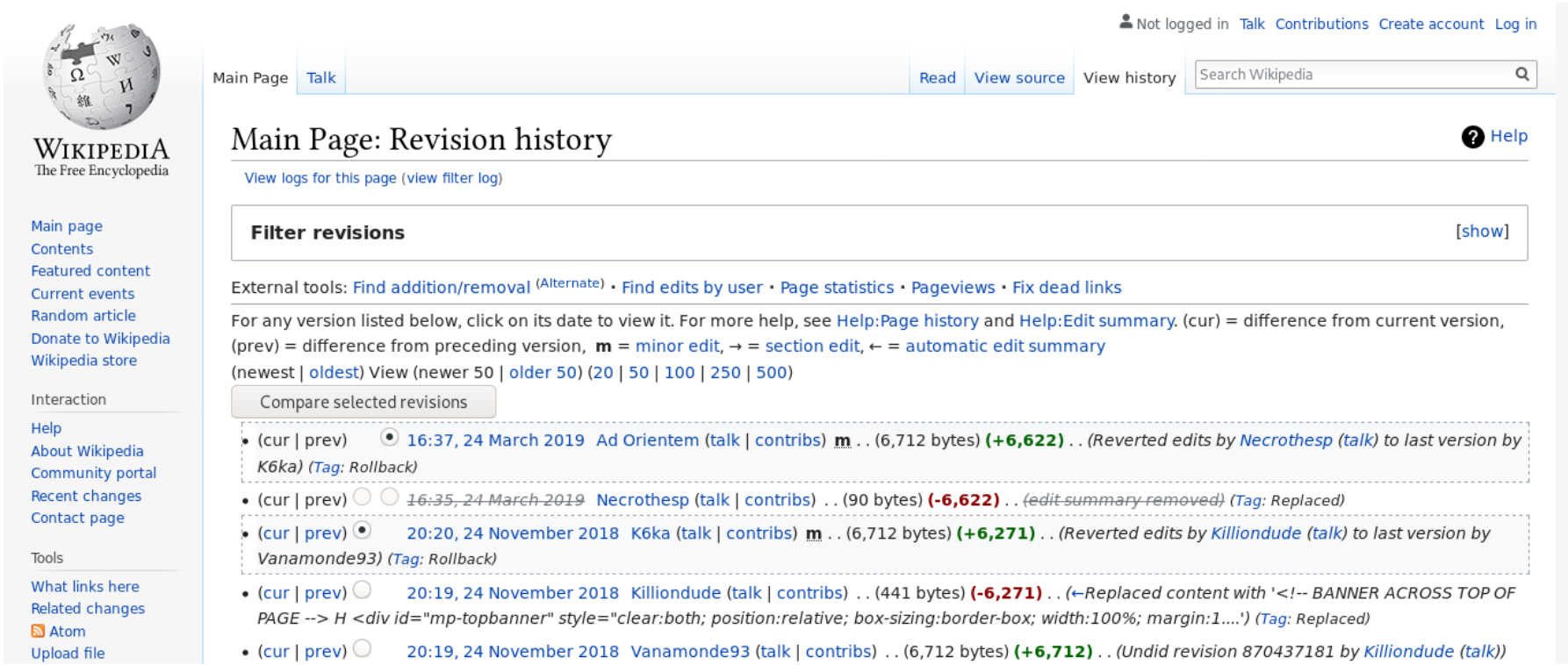
Versioning in Main-Memory Database Systems: From MusaeusDB to TardisDB

Maximilian E. Schüle, Lukas Karnowski, Josef Schmeißer, Benedikt Kleiner,
Alfons Kemper, Thomas Neumann

Santa Cruz, USA, July 22, 2019



Wikipedia: Version Control with Meta Tables



Not logged in Talk Contributions Create account Log in

Main Page | Talk | Read | View source | View history | Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Atom
Upload file

Main Page: Revision history

Help

View logs for this page (view filter log)

Filter revisions [show]

External tools: Find addition/removal (Alternate) · Find edits by user · Page statistics · Pageviews · Fix dead links

For any version listed below, click on its date to view it. For more help, see Help:Page history and Help:Edit summary. (cur) = difference from current version, (prev) = difference from preceding version, **m** = minor edit, → = section edit, ← = automatic edit summary

(newest | oldest) View (newer 50 | older 50) (20 | 50 | 100 | 250 | 500)

Compare selected revisions

- (cur | prev) 16:37, 24 March 2019 Ad Orientem (talk | contribs) **m** . . (6,712 bytes) **(+6,622)** . . *(Reverted edits by Necrothesp (talk) to last version by K6ka)* *(Tag: Rollback)*
- (cur | prev) ~~16:35, 24 March 2019~~ Necrothesp (talk | contribs) . . (90 bytes) **(-6,622)** . . ~~*(edit summary removed)*~~ *(Tag: Replaced)*
- (cur | prev) 20:20, 24 November 2018 K6ka (talk | contribs) **m** . . (6,712 bytes) **(+6,271)** . . *(Reverted edits by Killiondude (talk) to last version by Vanamonde93)* *(Tag: Rollback)*
- (cur | prev) 20:19, 24 November 2018 Killiondude (talk | contribs) . . (441 bytes) **(-6,271)** . . *(←Replaced content with '<!-- BANNER ACROSS TOP OF PAGE --> H <div id="mp-topbanner" style="clear:both; position:relative; box-sizing:border-box; width:100%; margin:1....')* *(Tag: Replaced)*
- (cur | prev) 20:19, 24 November 2018 Vanamonde93 (talk | contribs) . . (6,712 bytes) **(+6,712)** . . *(Undid revision 870437181 by Killiondude (talk))*

Wikipedia: Version Control with Meta Tables


```
CREATE TABLE page (
  page_id INT PRIMARY KEY,
  page_title TEXT,
  page_latest INT REFERENCES pagecontent (old_id)
);
CREATE TABLE revision (
  rev_id INT PRIMARY KEY,
  rev_page INT REFERENCES page (page_id),
  rev_text_id INT REFERENCES pagecontent (old_id),
  rev_parent_id INT,
  rev_timestamp TIMESTAMP
);
CREATE TABLE pagecontent (
  old_id INT PRIMARY KEY,
  old_text TEXT
);
```

	Size	Compression
Full Page Edit History	35.0 GiB	-
Current Version Only	1.1 GiB	-
History as File Diffs	14.0 GiB	59.77 %
History as Edit Diffs	9.4 GiB	72.71 %

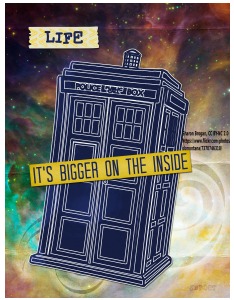
Table: Estimation of saved storage when using compression techniques based on the *Simple English* Wikipedia page edit history dump of October 1, 2018.

Challenges and Approaches

- version control including multiple tables and respecting referential integrity
- compressing articles by avoiding redundancy
- implicit version control **inside** or **on top** of database systems

MusaeusDB
works on top of existing database systems

TardisDB
integrated in a main-memory DB



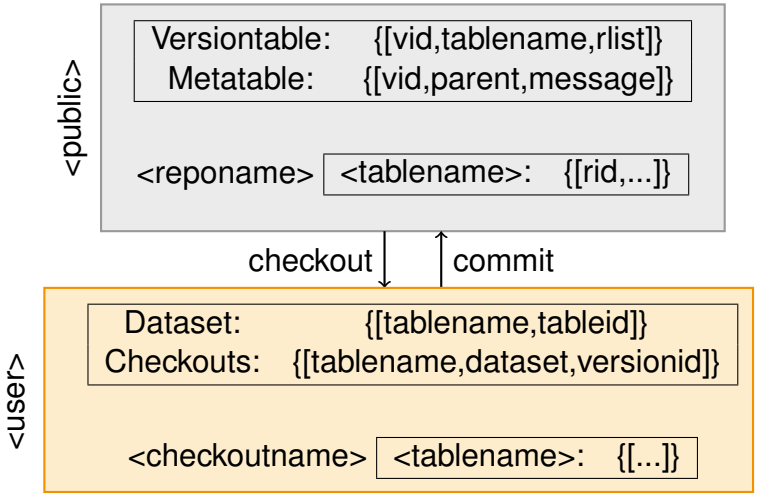
TardisBenchmark
Benchmark using text compression

MusaeusDB

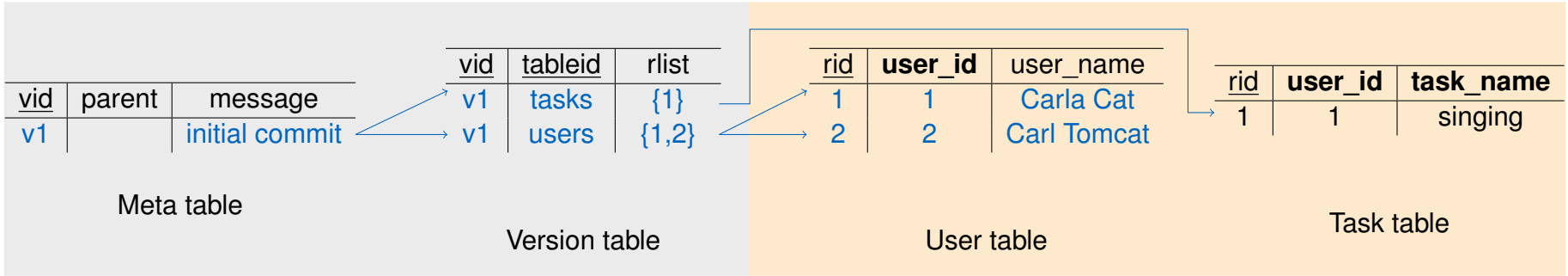
MusaeusDB: Concept

- extends OrpheusDB (VLDB 2017) to support multiple tables
- named after poet Musaeus of Athens, contemporary of Orpheus
- supports referential integrity

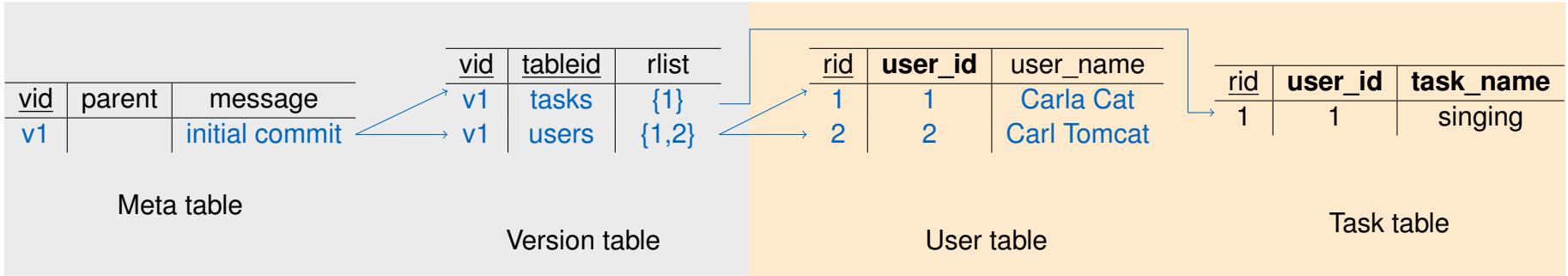
- checks out tables of a global repository into a local namespace
- table updates happen locally to be pushed afterwards
- every new entry gets a new and unique key (`rid`) assigned
- a version (identified by `vid`) consists of multiple `rids`



MusaeusDB: Example

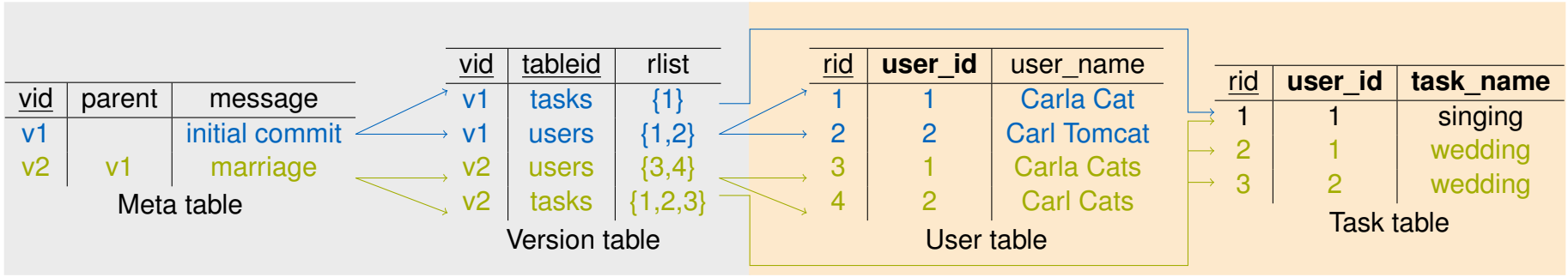


MusaeusDB: Example



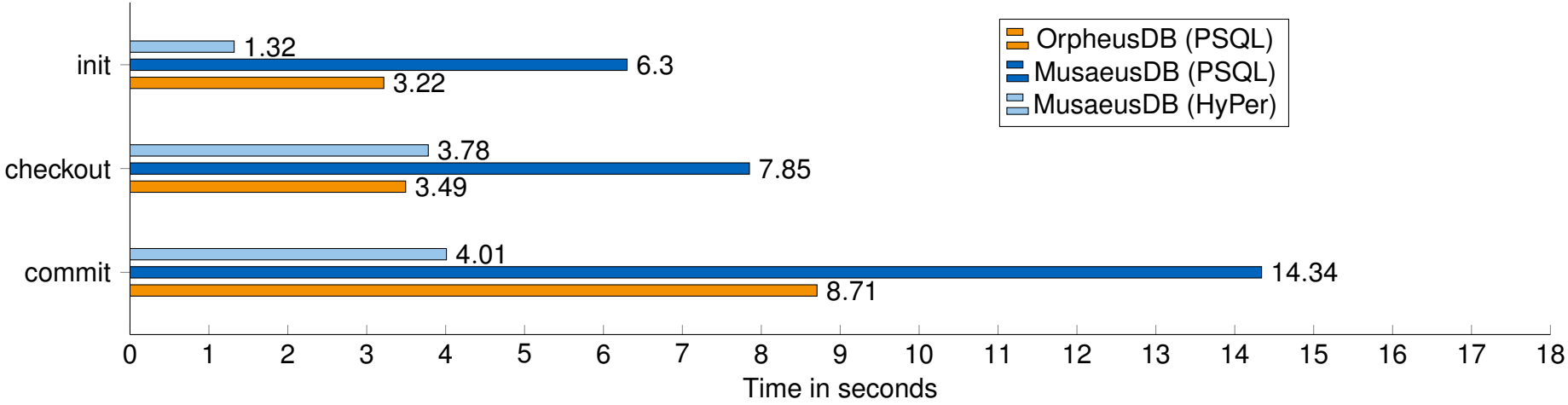

```
$ ./musaeus checkout public.cats schuele.cats
$ psql schuele cats "update user set user_name = 'Carla Cats' where user_id=1;"
$ psql schuele cats "update user set user_name = 'Carl Cats' where user_id=2;"
$ psql schuele cats "insert into tasks values (1,'wedding'),(2,'wedding');"
$ ./musaeus commit schuele.cats "marriage"
```

MusaeusDB: Example




```
$ ./musaeus checkout public.cats schuele.cats
$ psql schuele cats "update user set user_name = 'Carla Cats' where user_id=1;"
$ psql schuele cats "update user set user_name = 'Carl Cats' where user_id=2;"
$ psql schuele cats "insert into tasks values (1,'wedding'),(2,'wedding');"
$ ./musaeus commit schuele.cats "marriage"
```

MusaeusDB: Evaluation

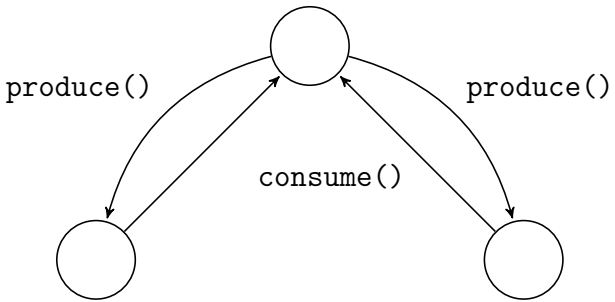


- Debian 9, 4 cores of Intel i7-7700HQ CPU, 2.80 GHz, 16 GiB RAM
- test data: 10^6 tuples of fictional users and corresponding tasks
- runtime of all operations doubled as two tables instead of one are processed

TardisDB

TardisDB: Bitmap Approach

- based on a LLVM code generating main memory database system prototype
- for each branch, one bitmap indicates every included tuple
- modified table scan operator checks bitmap for every tuple
- will lead to sparse bitmaps for all branches

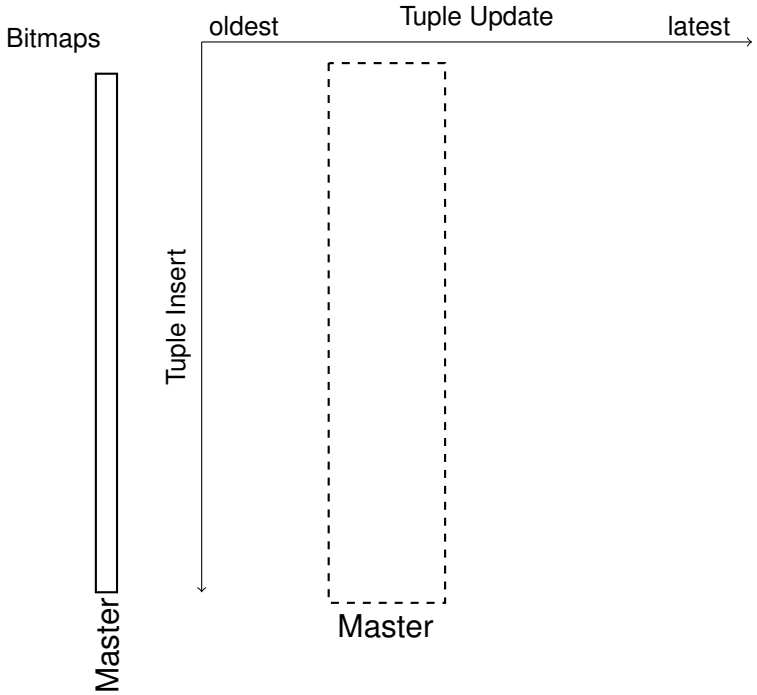



```
LoopGen scanLoop(funcGen,{{"index",cg_size_t(0ul)}});
cg_size_t tid(scanLoop.getLoopVar(0)); {
  LoopBodyGen bodyGen(scanLoop);
  auto branchId = _context.executionContext.branchId;
  IfGen visibilityCheck(isVisible(tid,branchId)); {
    produce(tid);
  }
}
cg_size_t nextIndex = tid+1ul;
scanLoop.loopDone(nextIndex<tableSize,{nextIndex});
```

TardisDB: Improved Concept

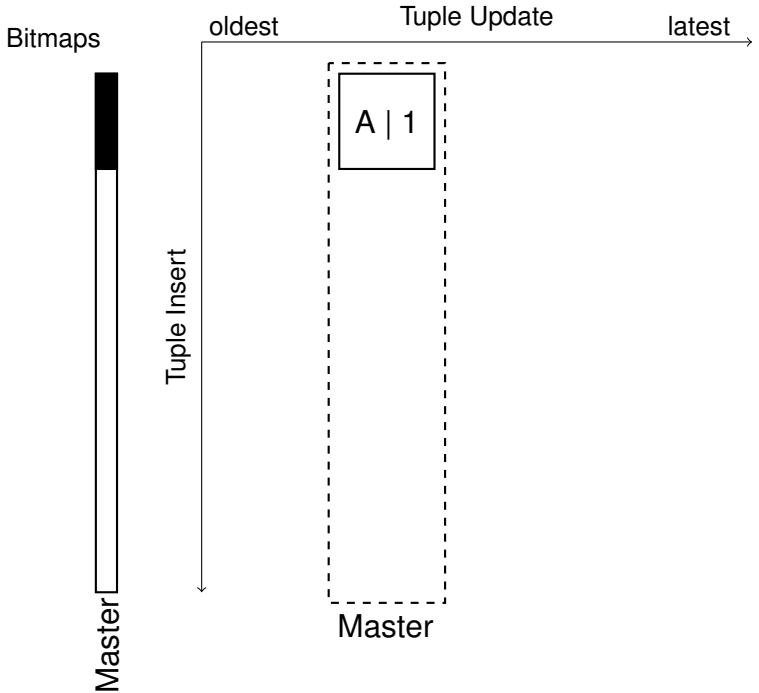
- **T**ime **a**nd **R**elative **D**imension**s** in **D**ata**b**ases
  - two dimensions: inserts and updates
- use bitmaps for insertions and deletions
  - bitmaps for every branch indicate included tuples
- reuses multi-version concurrency control for updates
  - every tuple is marked by the creator branch for table scans
  - prioritised master branch: no further operation needed
  - updates happen in place, previous versions are chained in buffers

TardisDB: Concept



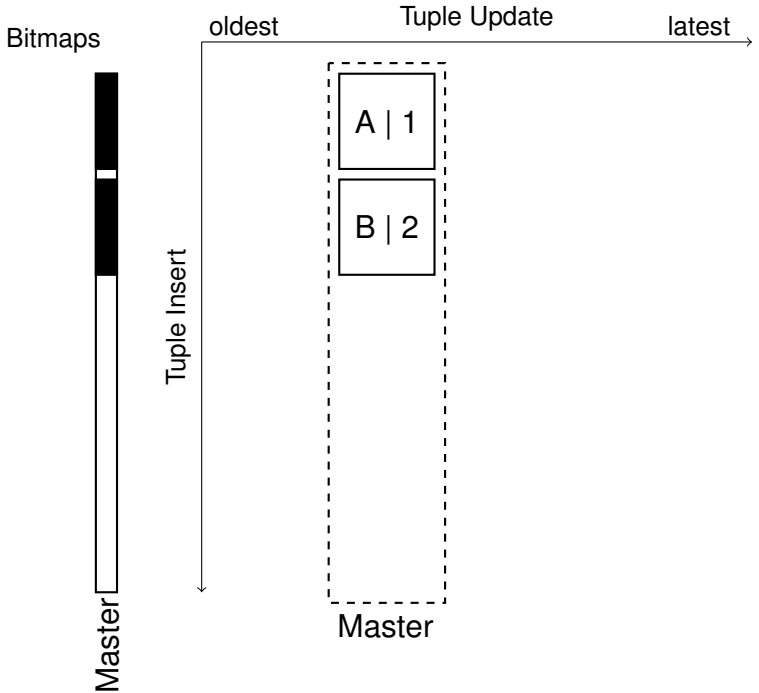
Time	Master	Branch 1	Branch 2	Branch 3

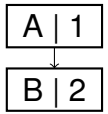
TardisDB: Concept



Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			

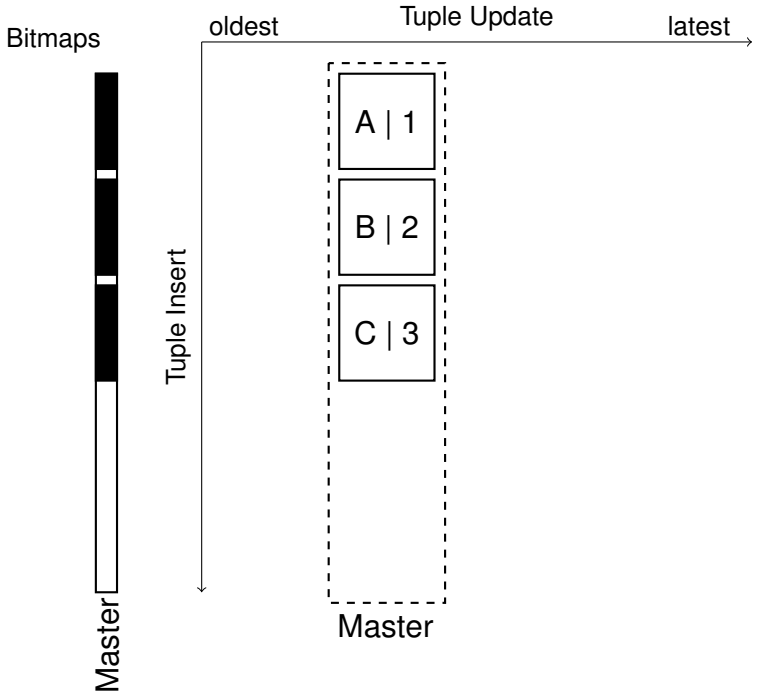
TardisDB: Concept

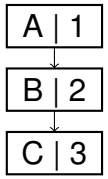




Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			

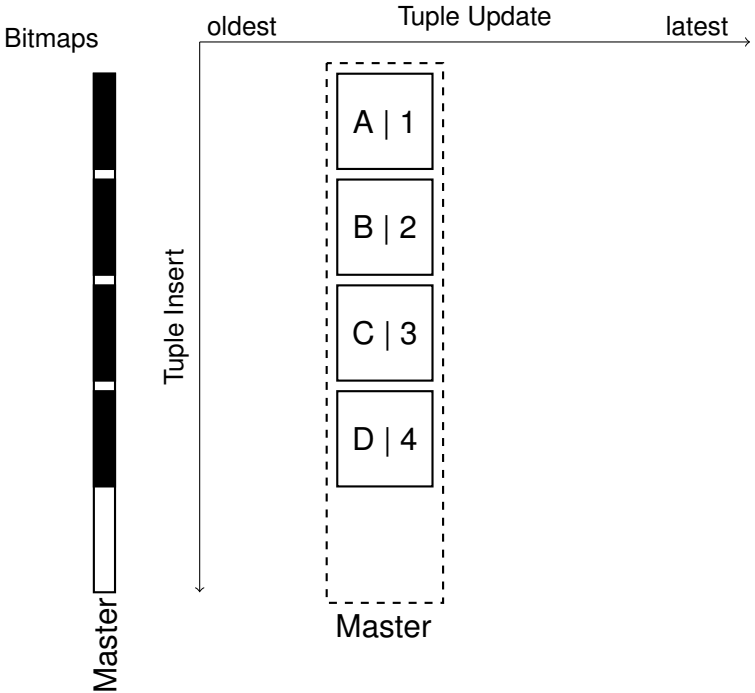
TardisDB: Concept

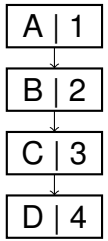




Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			

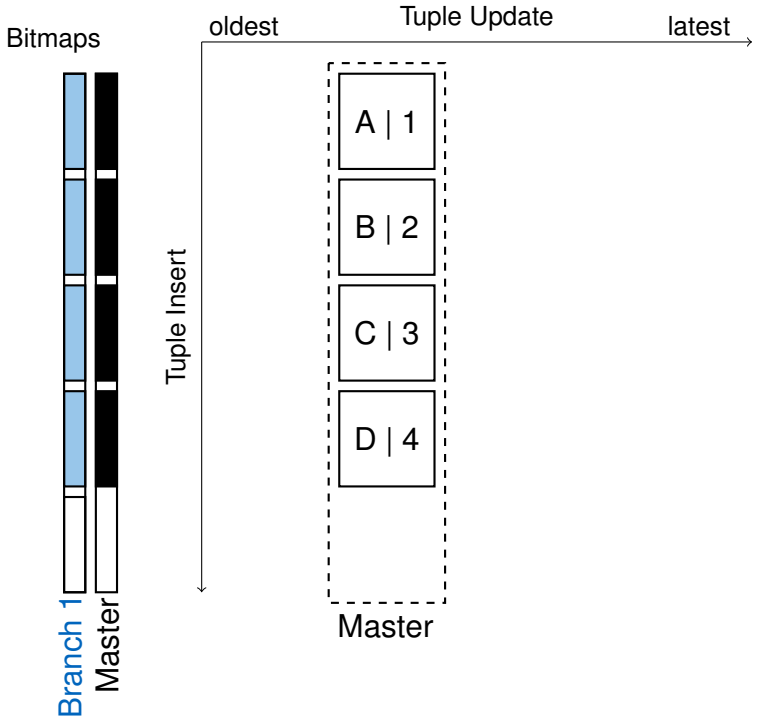
TardisDB: Concept

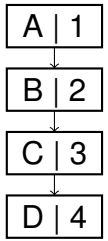




Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			

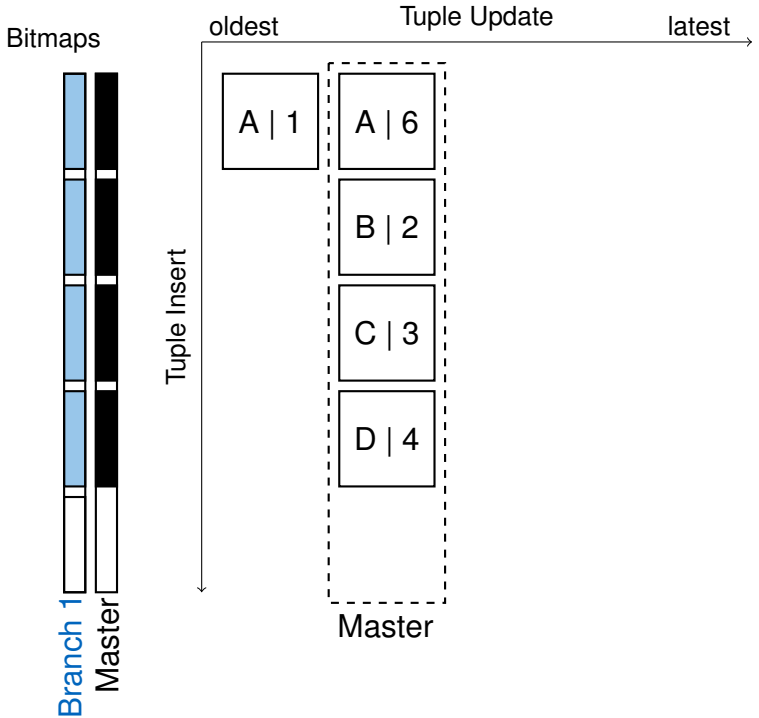
TardisDB: Concept

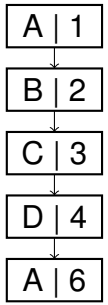




Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			

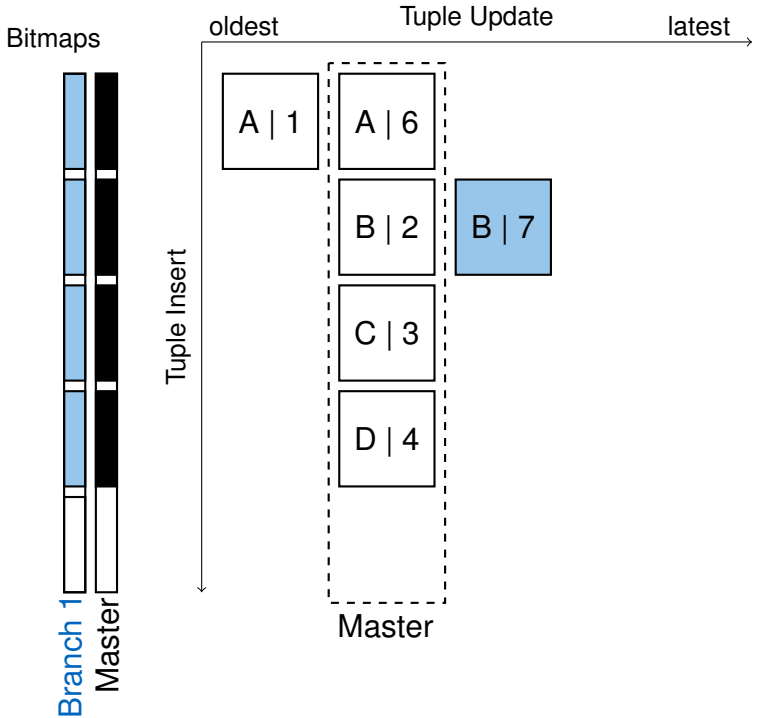
TardisDB: Concept

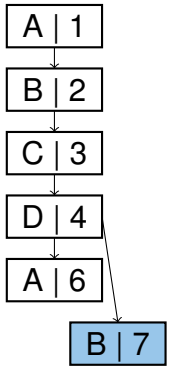




Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			
6	`update A`			

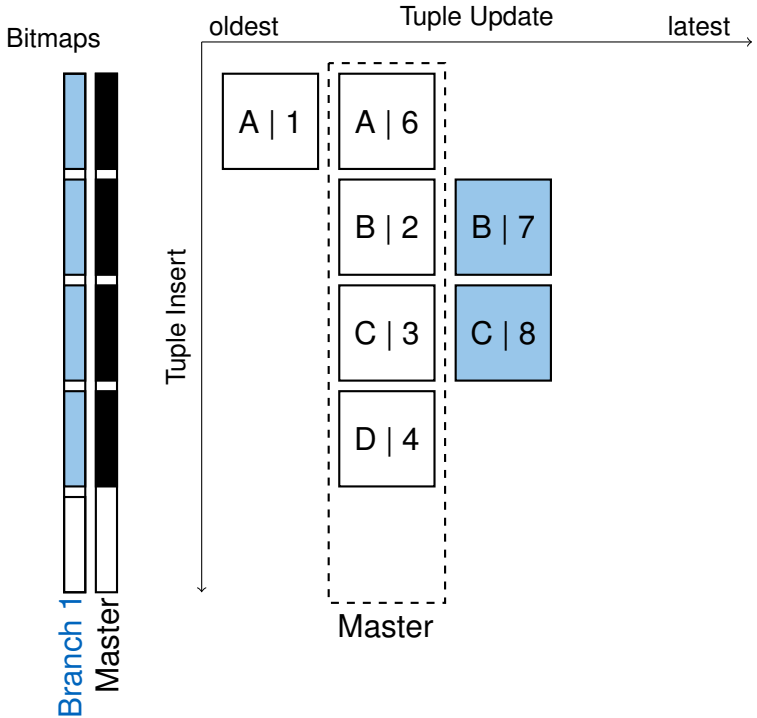
TardisDB: Concept

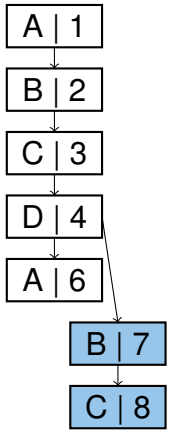




Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			
6	`update A`			
7		`update B`		

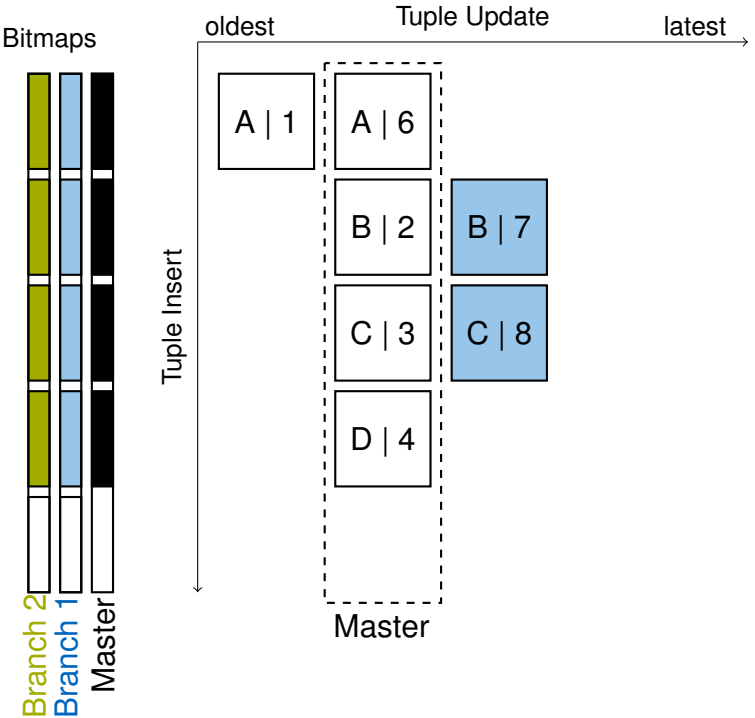
TardisDB: Concept

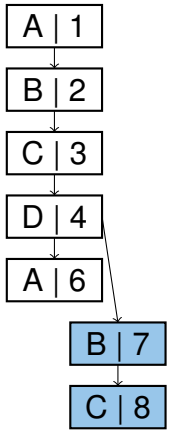




Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			
6	`update A`			
7		`update B`		
8		`update C`		

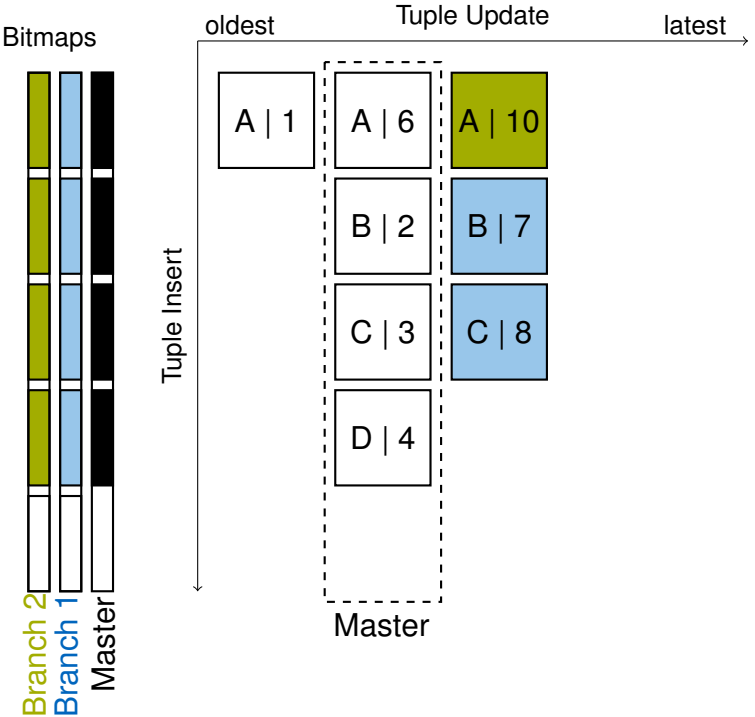
TardisDB: Concept

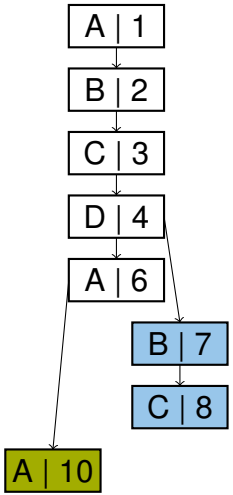




Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			
6	`update A`			
7		`update B`		
8		`update C`		
9	`branch 2`			

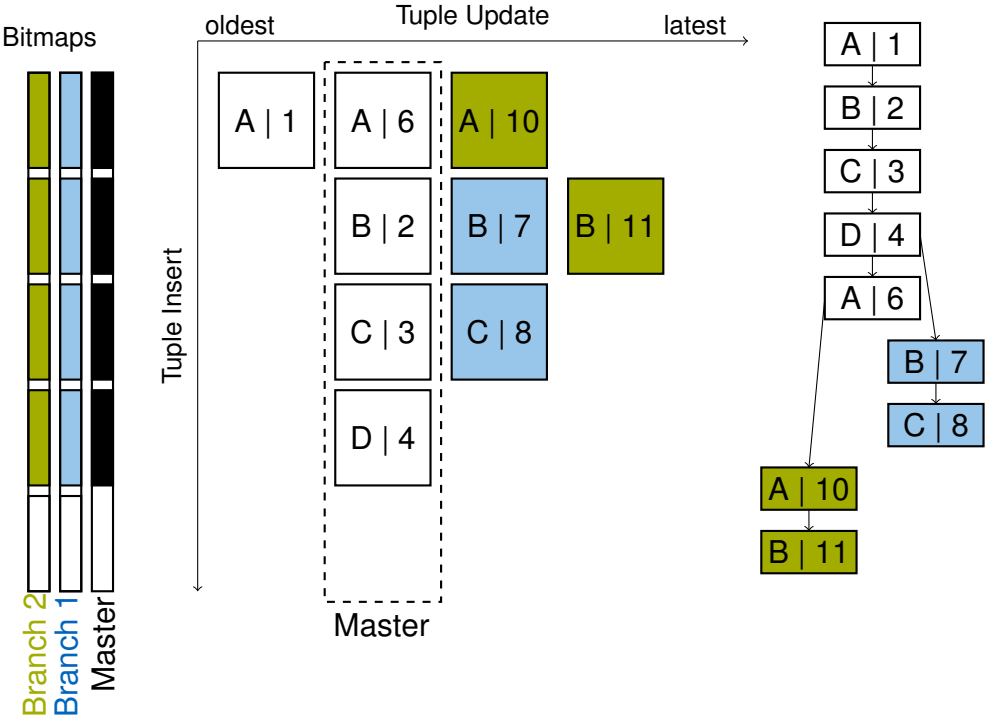
TardisDB: Concept





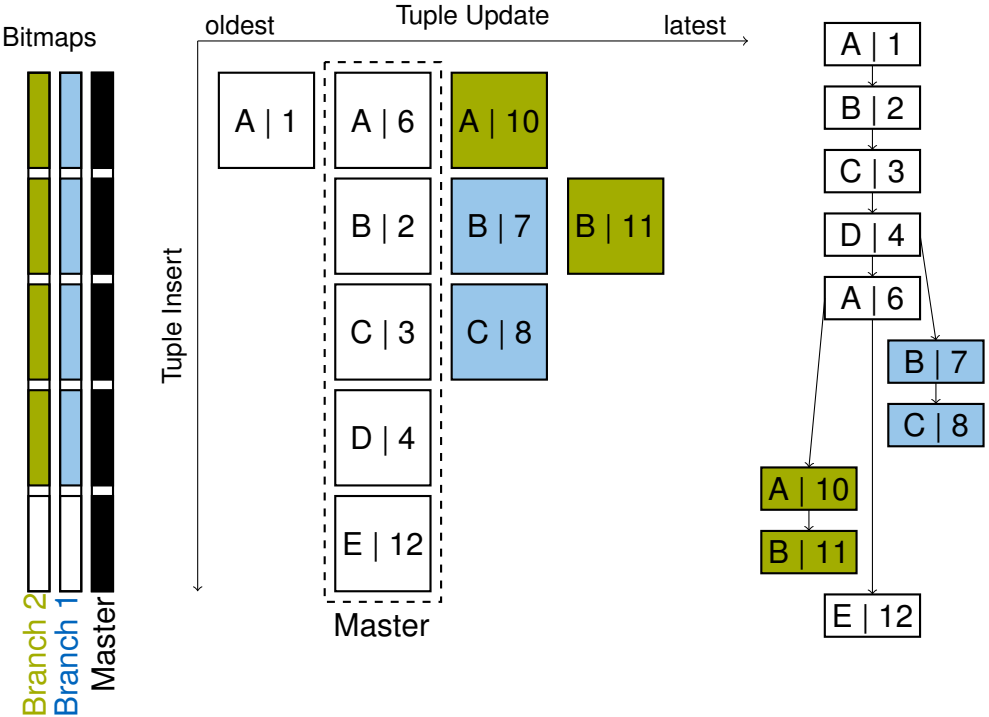
Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			
6	`update A`			
7		`update B`		
8		`update C`		
9	`branch 2`			
10			`update A`	

TardisDB: Concept



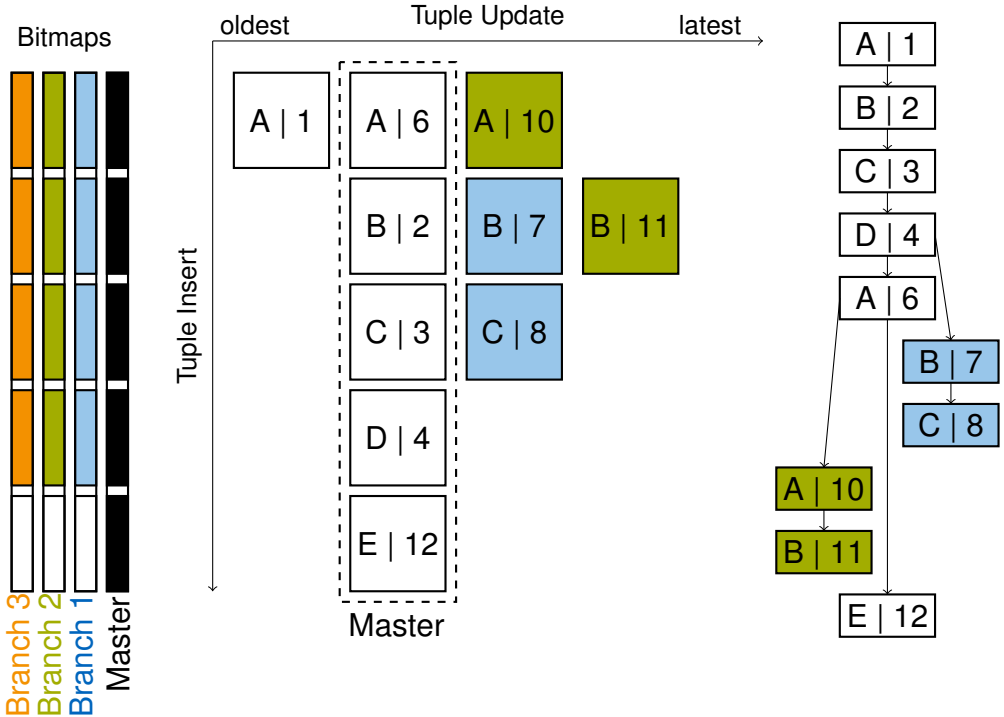
Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			
6	`update A`			
7		`update B`		
8		`update C`		
9	`branch 2`			
10			`update A`	
11			`update B`	

TardisDB: Concept



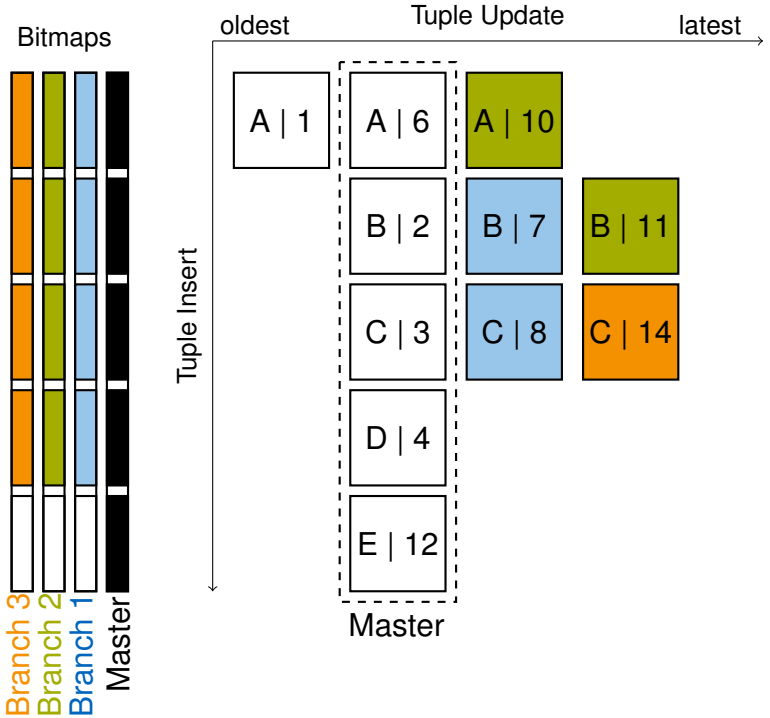
Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			
6	`update A`			
7		`update B`		
8		`update C`		
9	`branch 2`			
10			`update A`	
11			`update B`	
12	`insert E`			

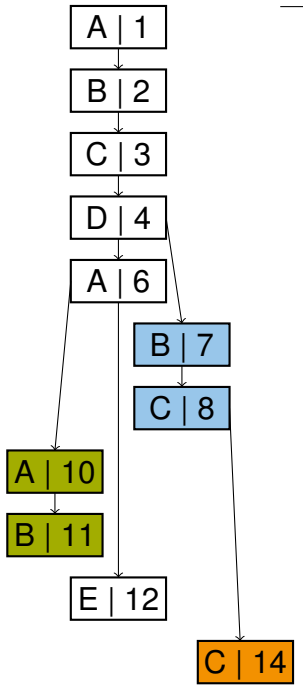
TardisDB: Concept



Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			
6	`update A`			
7		`update B`		
8		`update C`		
9	`branch 2`			
10			`update A`	
11			`update B`	
12	`insert E`			
13		`branch 3`		

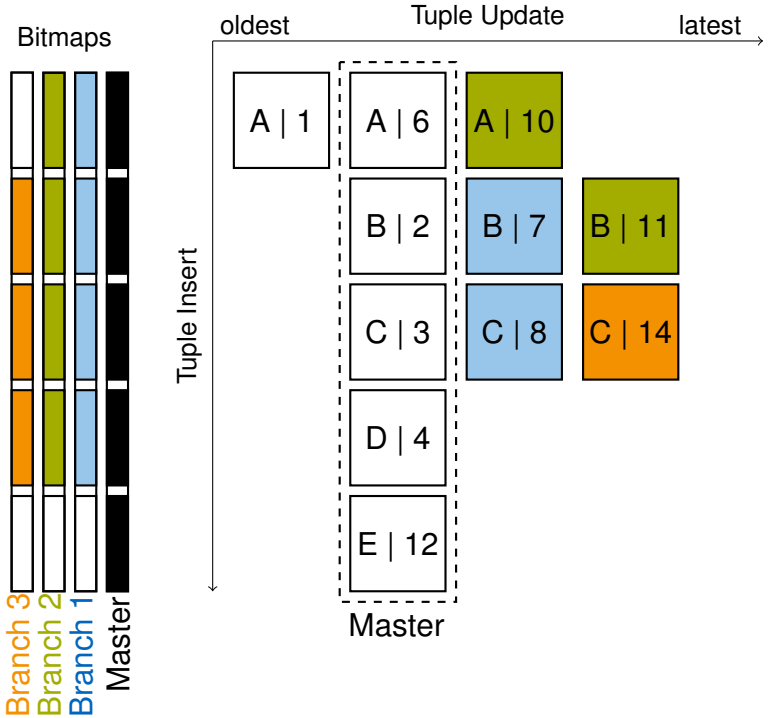
TardisDB: Concept

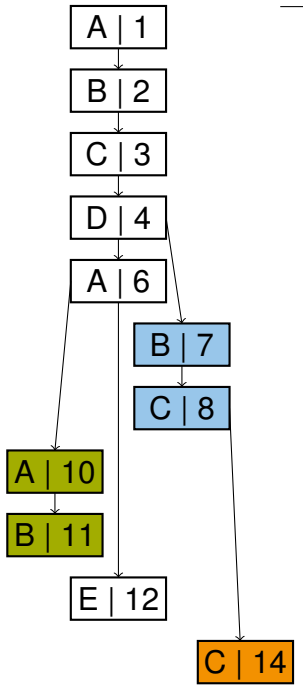




Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			
6	`update A`			
7		`update B`		
8		`update C`		
9	`branch 2`			
10			`update A`	
11			`update B`	
12	`insert E`			
13		`branch 3`		
14				`insert C`

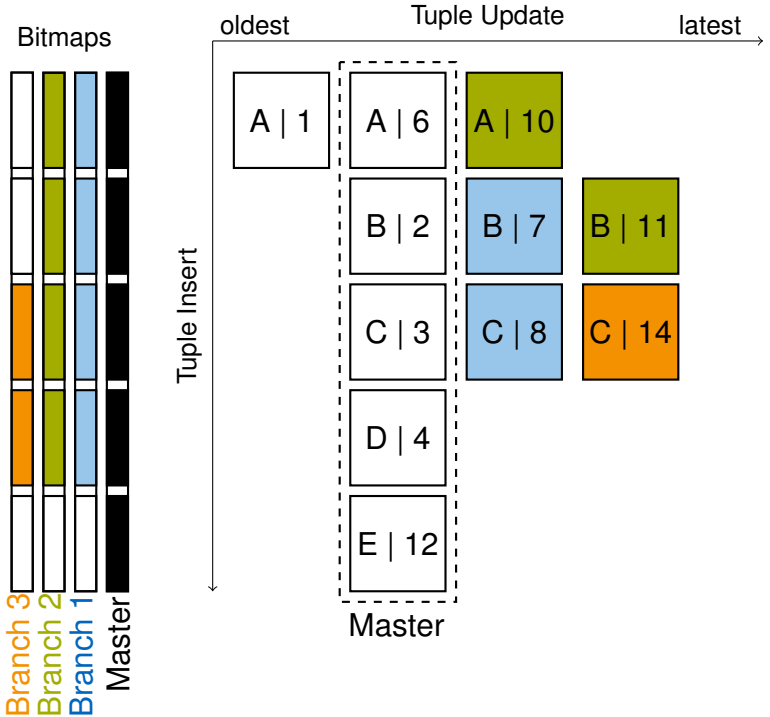
TardisDB: Concept

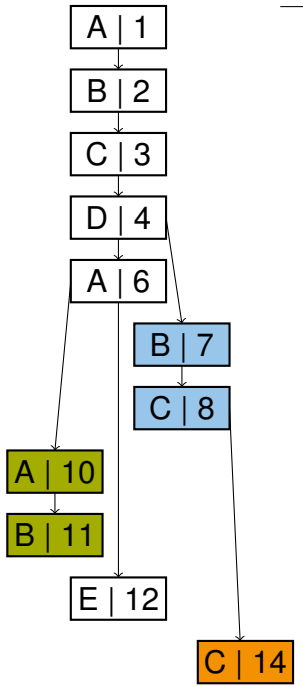




Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			
6	`update A`			
7		`update B`		
8		`update C`		
9	`branch 2`			
10			`update A`	
11			`update B`	
12	`insert E`			
13		`branch 3`		
14				`insert C`
15				`delete A`

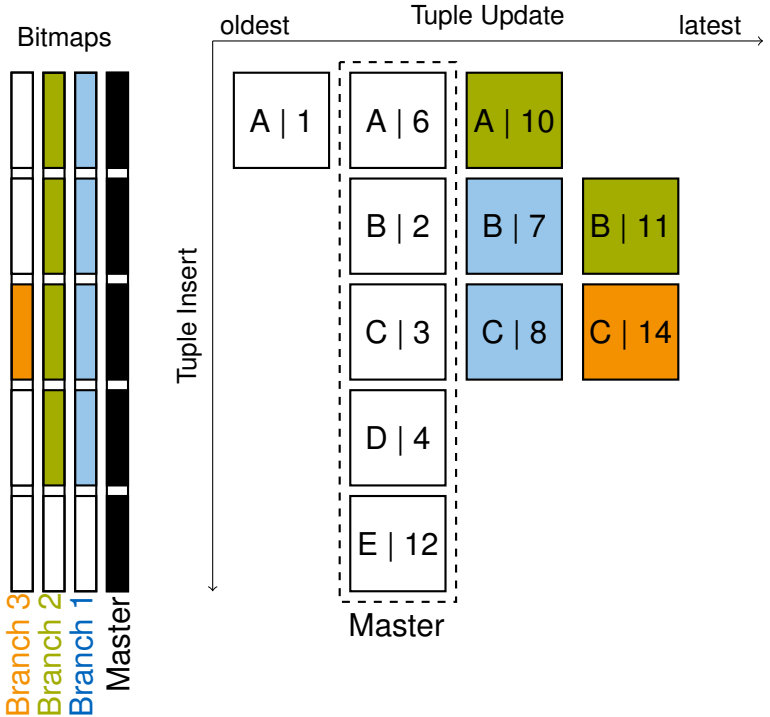
TardisDB: Concept

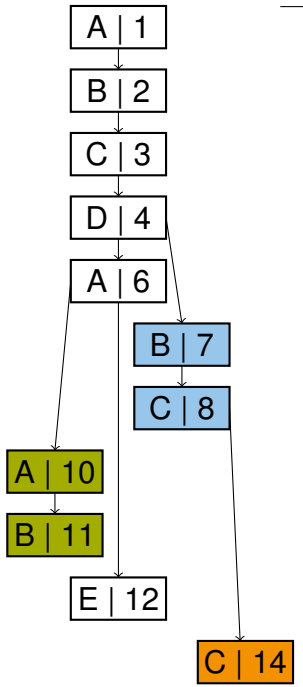




Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			
6	`update A`			
7		`update B`		
8		`update C`		
9	`branch 2`			
10			`update A`	
11			`update B`	
12	`insert E`			
13		`branch 3`		
14				`insert C`
15				`delete A`
16				`delete B`

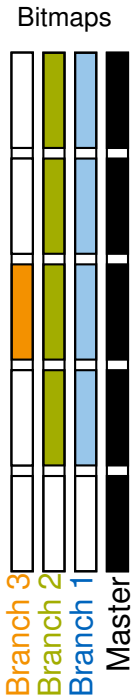
TardisDB: Concept

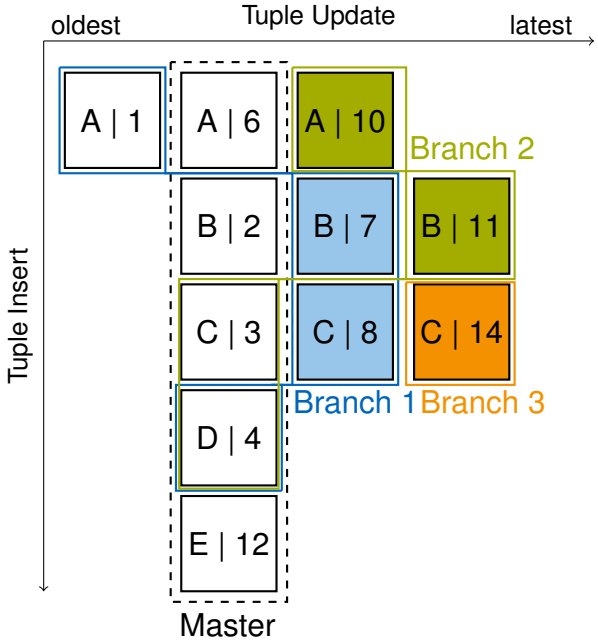


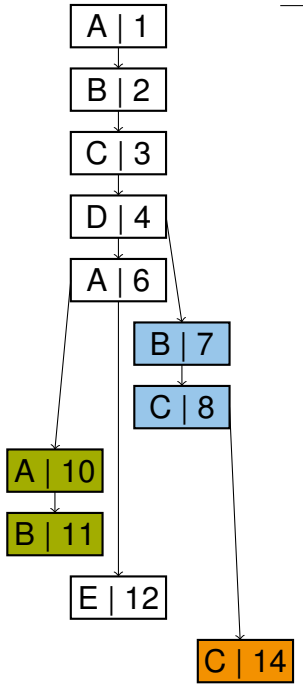


Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			
6	`update A`			
7		`update B`		
8		`update C`		
9	`branch 2`			
10			`update A`	
11			`update B`	
12	`insert E`			
13		`branch 3`		
14				`insert C`
15				`delete A`
16				`delete B`
17				`delete D`

TardisDB: Concept

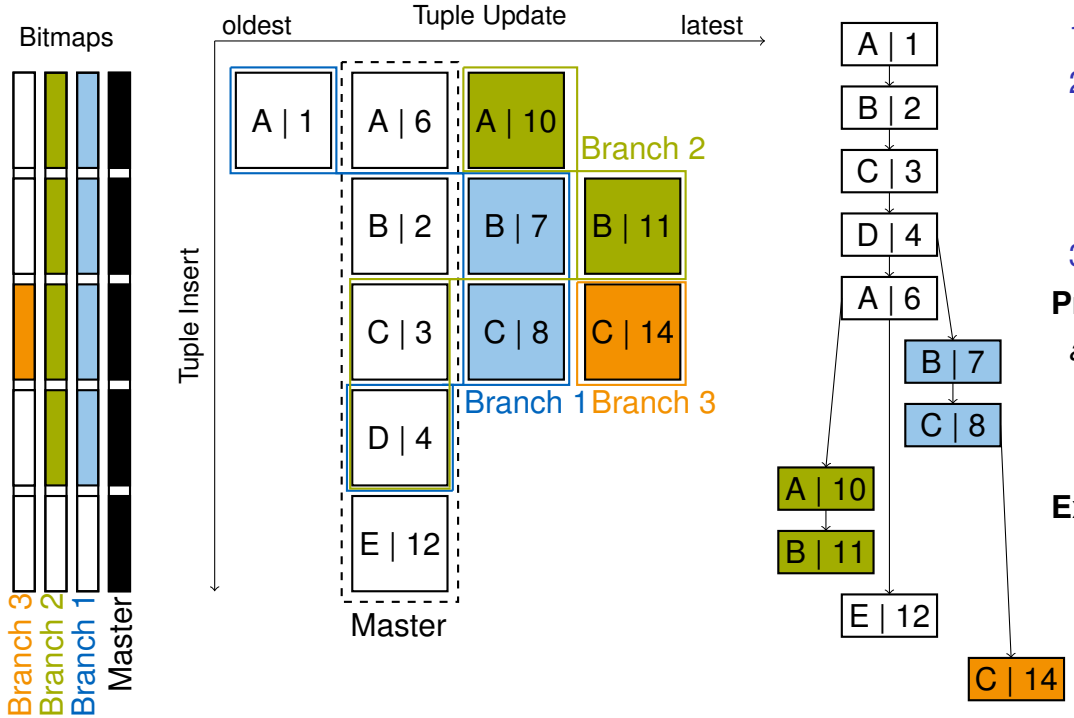






Time	Master	Branch 1	Branch 2	Branch 3
1	`insert A`			
2	`insert B`			
3	`insert C`			
4	`insert D`			
5	`branch 1`			
6	`update A`			
7		`update B`		
8		`update C`		
9	`branch 2`			
10			`update A`	
11			`update B`	
12	`insert E`			
13		`branch 3`		
14				`insert C`
15				`delete A`
16				`delete B`
17				`delete D`

TardisDB: Concept



Produce Tuple t in Branch b

timestamp $ts()$ for every branch and every tuple

1. Check bit in bitmap
2. evaluate predicate $active(t, b)$
   - created by the branch itself ($created(b, t)$)
   - or active in an ancestor branch before branching took place
3. take latest entry (with highest timestamp)

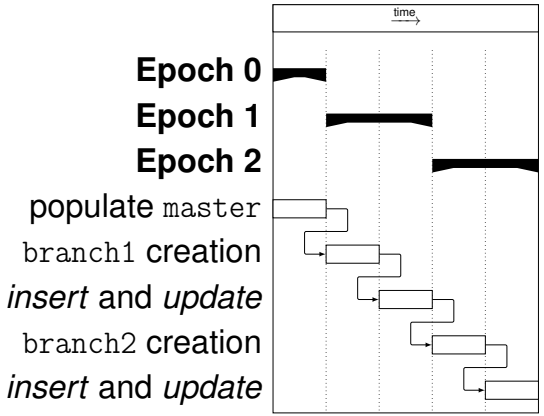
Predicate $active()$

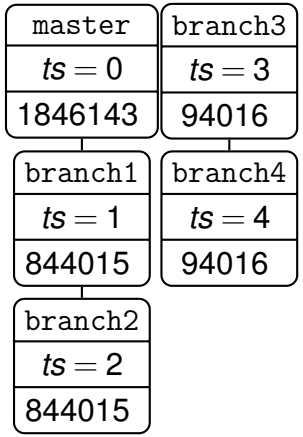
$$active(t,b) \Leftrightarrow created(b,t) \vee \bigvee_{p \in parent(b)} active(t,p) \wedge ts(t) < ts(b)$$

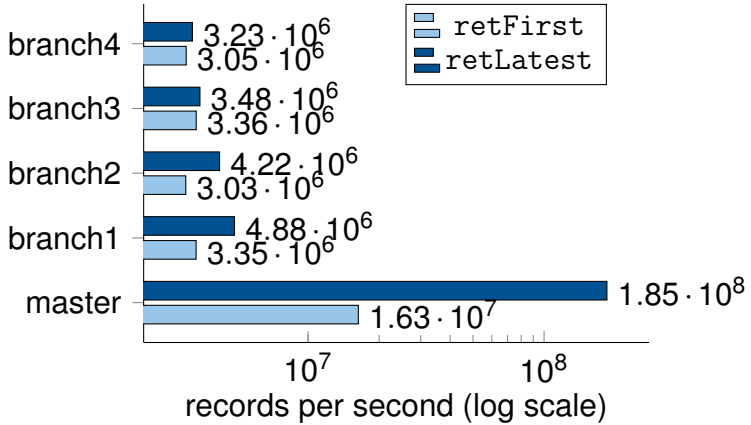
Examples

- `A` in Branch 3: bit not set, not visible
- `C` in Branch 3: visible, latest tuple is `C|14`
- `A` in Branch 1 ($ts(b) = 5$): visible, `A|6` not active

TardisDB: Versioning Benchmark





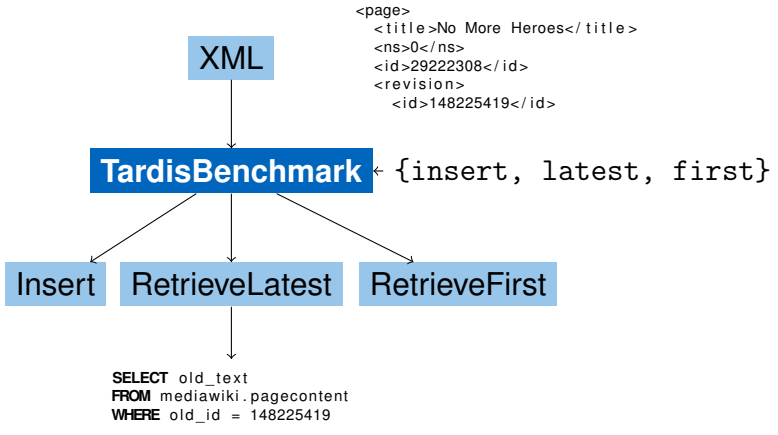


- Ubuntu 18.04 LTS, Intel Xeon CPU E5-2660 v2 processor, 2.20 GHz (20 cores), 256 GiB DDR4 RAM
- three epochs with $5 * 10^5$ inserted and $> 3 * 10^4$ updated tuples each
- reads of tuples in the master branch faster than of tuples in other branches (as optimised for this workload)

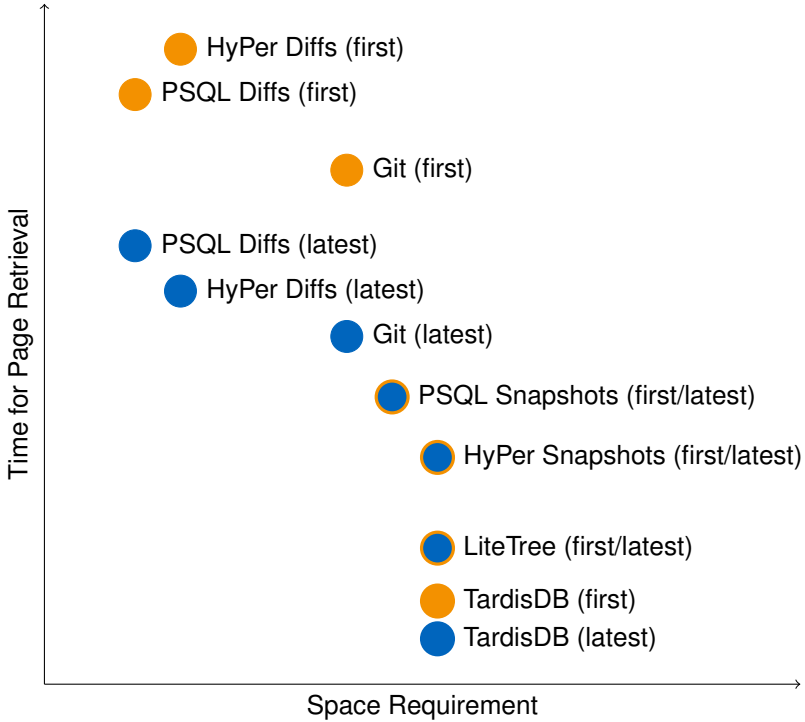
TardisBenchmark

TardisBenchmark: Concept

- benchmark based on Wikipedia
  - MediaWiki schema with tables for page, content and revisions
  - XML page dumps as input allows flexible workload (up to 13 TB)
- text compression methods
  - `Snapshot`: default in MediaWiki, stores every change as a whole
  - `Diff`: stores changes to the latest version as a deltas of differences
- operations
  - `insert`: inserts pages, compute deltas
  - `retrieveLatest`: retrieve latest page version
  - `retrieveFirst`: apply all deltas to retrieve first version

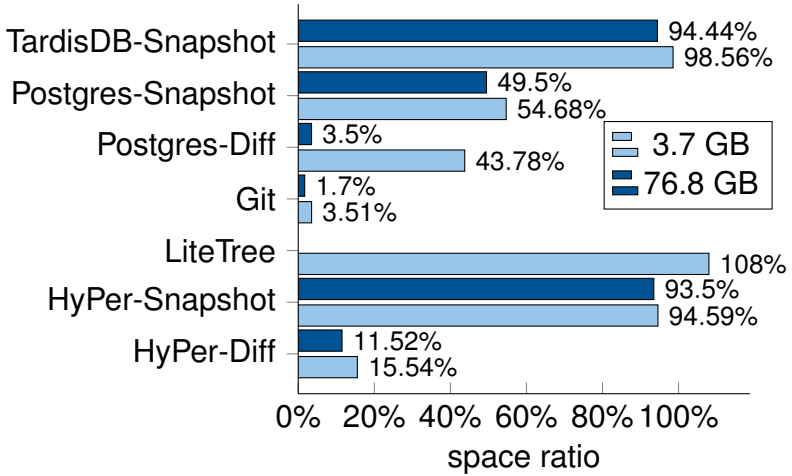


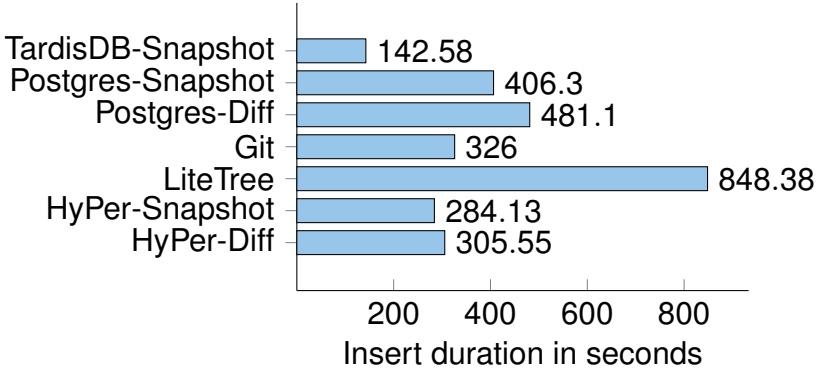
TardisBenchmark: Expected Trade-Off

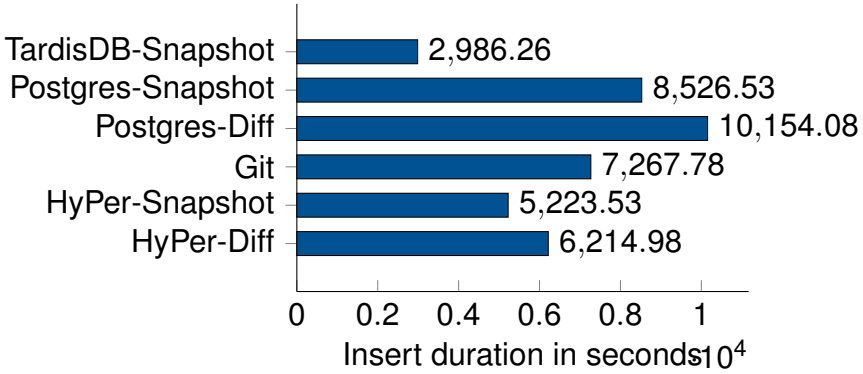


- Ubuntu 18.04 LTS, Intel Xeon CPU E5-2660 v2 processor, 2.20 GHz (20 cores), 256 GiB DDR4 RAM
- full page edit history from 1 August, 2018: pages 10 up to 2,087 (76.8 GB) and 30,227 up to 30,303 (3.7 GB)
- storage approaches:
  - `Snapshot` (HyPer, PSQL, TardisDB)
  - `Diff` (HyPer, PSQL)
  - `Git`
  - `LiteTree`
- operations: `insert`, `retrieveLatest`, `retrieveFirst`

TardisBenchmark: Insert Performance and Space Requirement

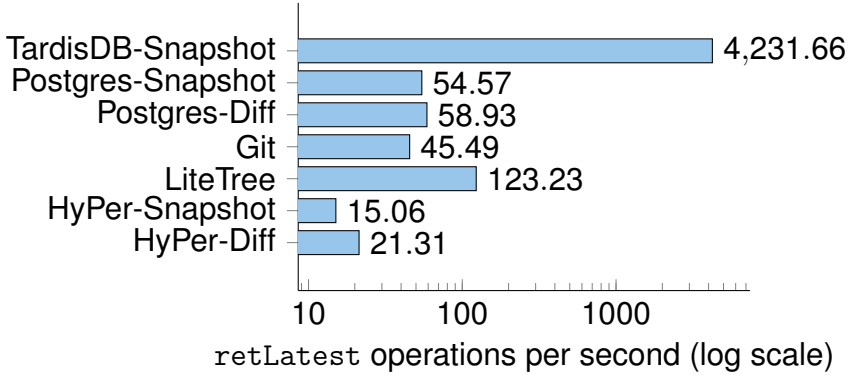


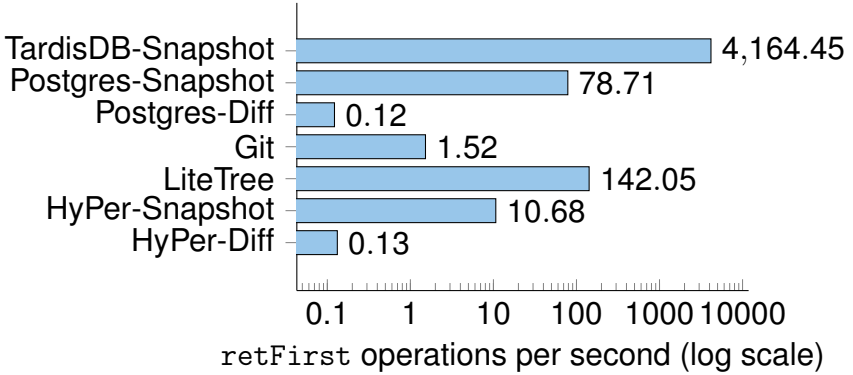


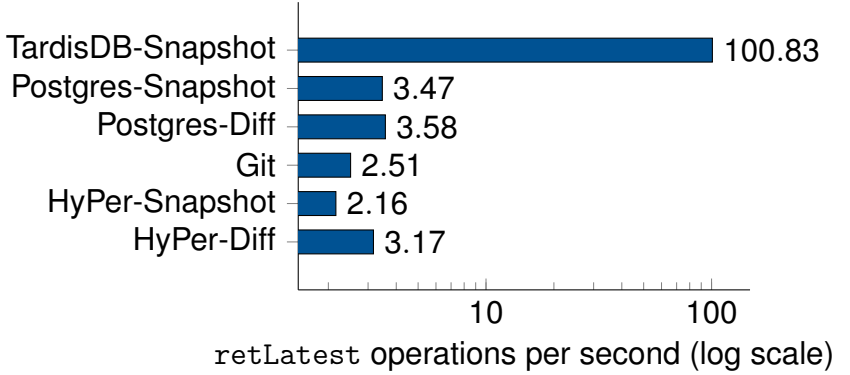


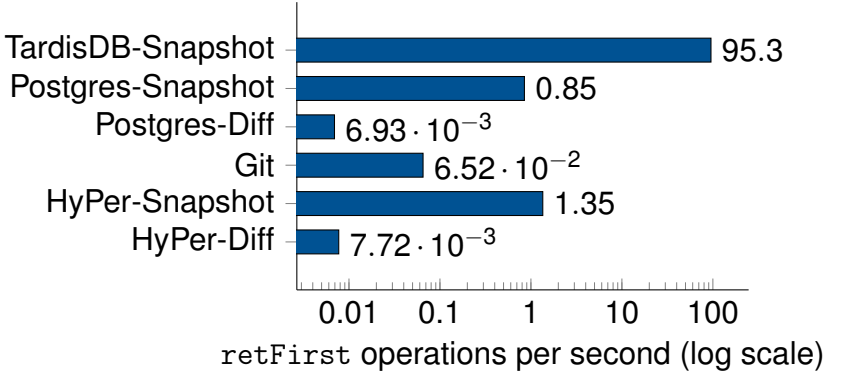
- `Diff`: slower, therefor less space needed
- `Git`: space efficient

TardisBenchmark: Retrieve Performance

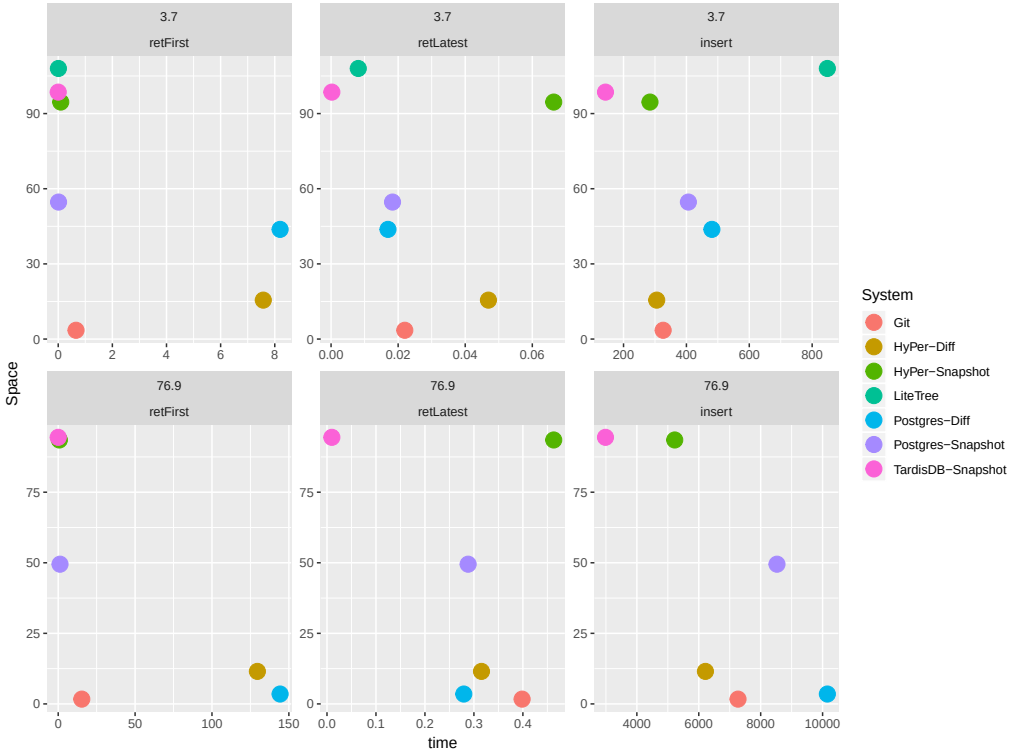








TardisBenchmark: Conclusion



- TardisDB showed best performance (as optimised for this workload)
- `Git`: best compression, good retrieval times
- `LiteTree` fast, but only allows up to 1024 branches; high space consumption
- PSQL and HyPer: `Diff` consumed only about 90 % less space than `Snapshot`

Conclusion and Future Work

- TardisBenchmark: reproducible results to support future research
- MusaeusDB: versioning on top of database systems
- TardisDB: integration in a MMDB (bitmaps/MVCC)

Improvements

- `VERSIONED` for text-like datatypes
- improvements for `Diff`:
  - store snapshot of every *N*th version for constant retrieval times
  - apply multiple deltas at once
- SQL integration

```
CREATE TABLE page (
  page_id INT PRIMARY KEY,
  page_title TEXT, page_latest INT
  page_content TEXT VERSIONED
);

SELECT *
FROM users VERSION <versionid>
```

Requirements for a Versioning System

$\rightarrow \mid \leftarrow$	Δ	$\mathcal{O}(1)$	ACID/MVCC	SQL
compressing full articles	Enable delta compression	Constant retrieval times (focus on latest version)	Database system guarantees	SQL as the declarative programming language

Thank you for your attention!

`https://gitlab.db.in.tum.de/tardisDB`