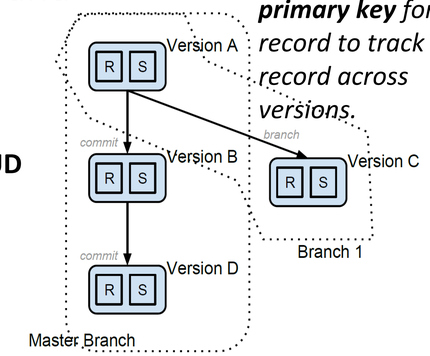
Decibel: The Relation Dataset Branching System

Michael Maddox (MIT CSAIL), David Goehring (MIT CSAIL), Aaron J. Elmore (University of Chicago), Samuel Madden (MIT CSAIL),
Aditya Parameswaran (University of Illinois), Amol Deshpande (University of Maryland)

- **Motivations:**
  - Need for data management systems that natively support ***versioning*** or ***branching*** of datasets to enable concurrent analysis, integration, manipulation, or curation of data across teams
  - Storage efficiency (reduce redundancy)
  - Lineage tracking of modifications
  - Versioned Queries (with runtime efficiency)
- **Overview**
  - *git* versioning semantics applied to a dataset, collection of tables
  - Append only data store
- **Contributions**
  - 3 storage models that implement dataset versioning
  - Versioned benchmark for evaluating performance of these storage models

Basic Operations and Workflow

- **Branch**
- **Commit/Checkout**
- **Data Modification (CRUD**
- **Single Branch Scan**
- **Multi-Branch Scan**
- **Diff Branches**
- **Merge Branches**



*Assumes a **unique primary key** for each record to track that record across versions.*

- *Theses basic primitives can be used to implement a wide variety of versioned queries.*

Versioned Queries

#	*Query Type*	*VQuel Query*	*SQL Equivalent*
-	Basic versioning commands	`branch, merge`	–
1	*Single version scan*: find all tuples in relation R in version v01	`range of E1 is Version("v01").Relations("R")` `retrieve E1.all`	`SELECT * FROM R` `WHERE R.Version = "v01"`

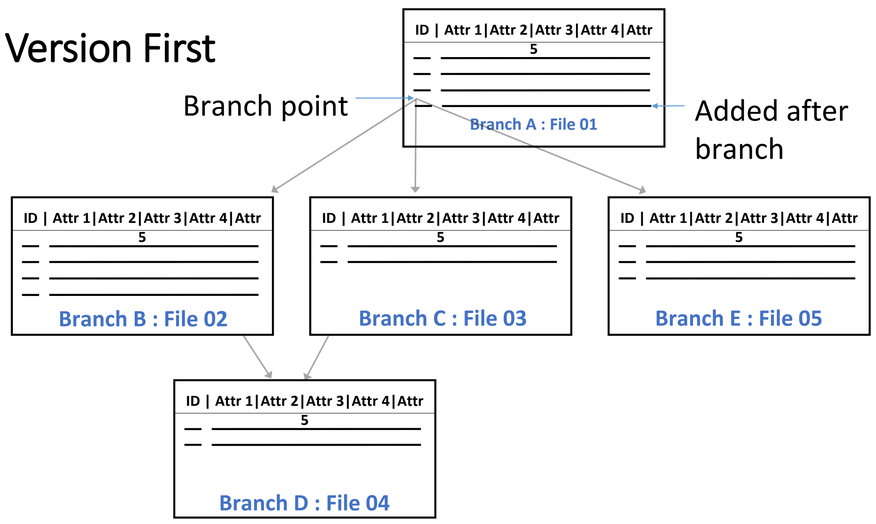
2	*Multiple version positive diff*: positive diff relation R between versions v01 and v02	`range of E1 is Version("v01").Relations("R")`
		`retrieve E1.all where E1.id not in`
		`(range of E2 is Version("v02").Relations("R")`
		`retrieve E2.id)`
			`SELECT * FROM R`
			`WHERE R.Version = "v01" AND R.id NOT IN`
			`(SELECT id from R`
			`WHERE R.Version = "v02")`

3	*Multiple version join*: join tuples in R in versions v01 and v02 satisfying Name = Sam	`range of E1 is Version("v01").Relations("R")`
		`range of E2 is Version("v02").Relations("R")`
		`retrieve E1.all, E2.all`
		`where E1.id = E2.id and E1.Name = "Sam"`
			`SELECT * FROM R as R1, R as R2`
			`WHERE R1.Version = "v01" AND R2.Version = "v02"`
			`AND R1.id = R2.id AND R1.Name = "Sam"`

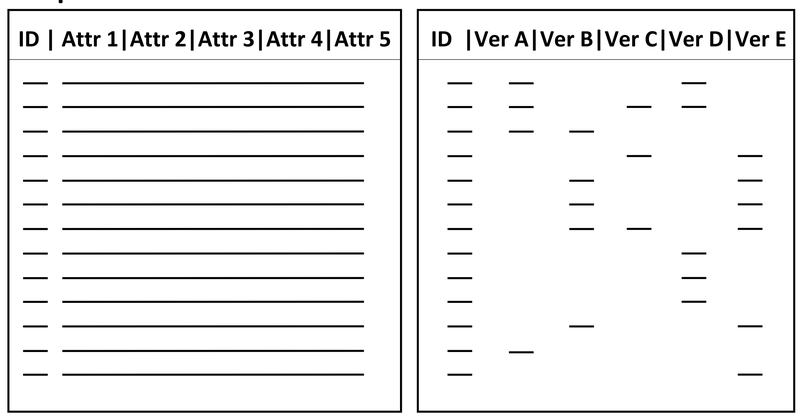
4	*Several version scan*: find all head versions of relation R	`range of E1 is Version`
		`range of E2 is E1.Relations("R")`
		`retrieve E2.all`
		`where HEAD(E1.id) = true`
			`SELECT * FROM R WHERE HEAD(R.Version) = true`

Table 1: VQuel Examples

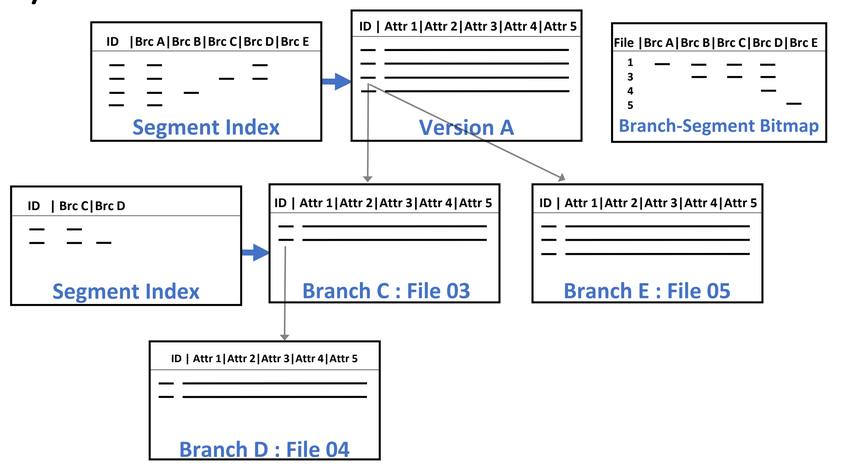
Version First



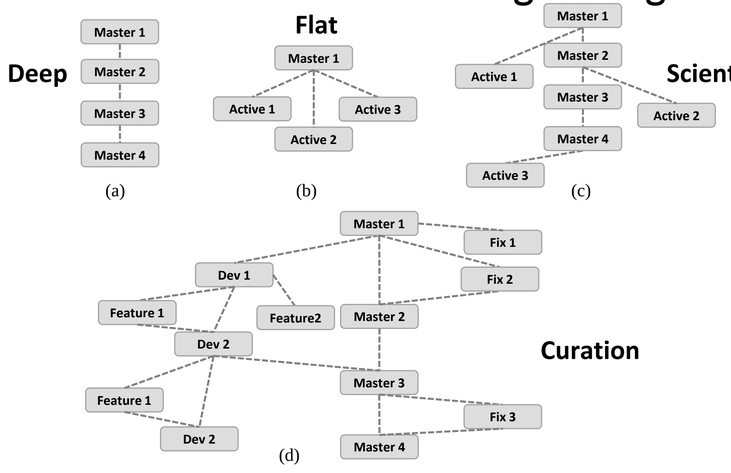
Tuple First



Hybrid

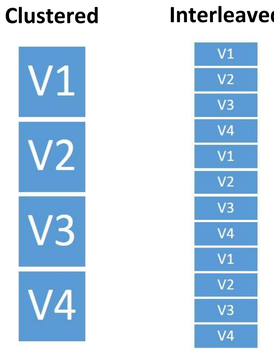


Versioned Benchmark: Branching Strategies

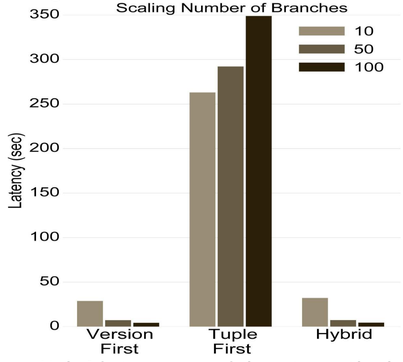


Versioned Benchmark: Data Generation and Loading

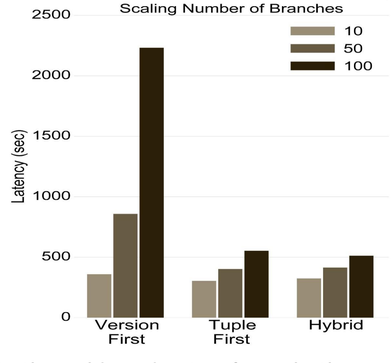
- Vary number of branches and data set size
- Stagger branch points
- Mixed workload (e.g. 20% updates, 80% inserts)
- Commits at regular intervals
- Insert skew
- Follow branching strategy for evolving version graph
- **Loading Mode**:
  - Insert/Update into branches to achieve **Physical layout**, e.g. clustering level in clustered mode
  - Or follow branching strategy in interleaved mode
- Load older branches



Single Version Scan on Flat (100 GB DB)



Multi Version Scan on Deep (100 GB DB)



Hybrid storage model captures the best of both worlds and outperforms both extreme storage models in almost every experiment.

	Version First	Tuple First	Hybrid
Single Branch Scan	✓	×	✓
Multi Branch Scan	×	✓	✓
Branch Membership	×	✓	✓
Bitmap Storage Efficiency	-	×	✓