Seminar Report

Extensibility and Data Sharing in Evolving Multi-Tenant Database

Reporter: Qiu Lei

About Author

- **Stefan Aulbach**
- Technische Universität MünchenInstitut für Informatik

- **Research Area**
- Multi-Tenancy Databases and Software-as-a-Service
- Next-Generation Databases
- AutoGlobe: An Adaptive Infrastructure for Distributed Services

- **Publication List**
- Extensibility and Data Sharing in Evolving Multi-Tenant Databases (ICDE 2011)
- A Comparison of Flexible Schemas for Software as a Service (SIGMOD 2009)
- Multi-Tenant Databases for Software as a Service (SIGMOD 2008)
- Ruminations on Multi-Tenant Databases (BTW 2007)

Background

- **Essential features of enterprise applications**
- **Data Sharing**
- (master data should be shared rather than replicated for each tenant)
- **Extensibility**
- (application modification and extension, which applies both to the database schema and master data it contains)
- **Evolution**
- (evolution of the schema and master data, which occurs as the application and its extensions are upgraded)
- **The typical SaaS environment**
- the latest available version of the applicaion is the most popular and a limited number of tenants lag one or more versions behind
- **Optimized for main-memory**
- allows to identify which versions of the instances are currently used by a given set of tenants, and which can be removed from main-memory

Contributions

- **FLEXSCHEME**
- a data management model for Multi-Tenant DBMSs

- **XOR Delta**
- a physical data organization approach of segment sequences
- (there is a definition of segment sequences in FLEXSCHEME)

Requirement

- SaaS applcations develop in at least two dimensions:Extensibility and Evolution

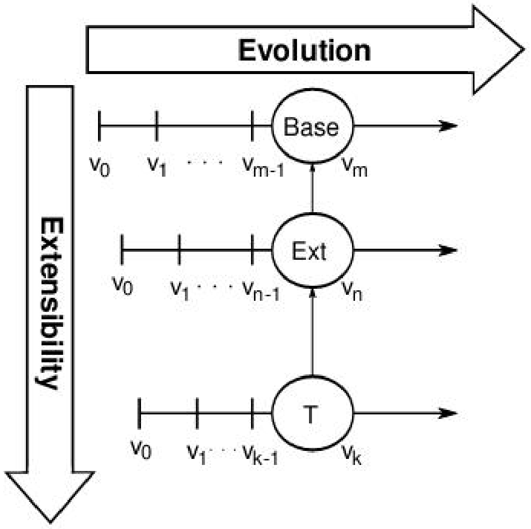

Evolution
Extensibility
Base
v_0 v_1 · · · v_{m-1} v_m
Ext
v_0 v_1 · · · v_{n-1} v_n
T
v_0 v_1 · · · v_{k-1} v_k

Fig. 1: Extensibility and Evolution from a Single Tenant's Perspective

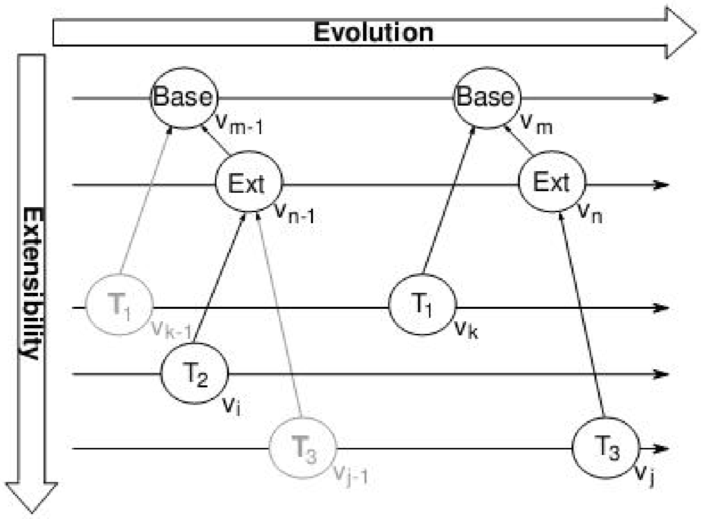


Fig. 2: Extensibility and Evolution for Multiple Tenants

Data Management Model

- The global view with three Tenants(*Fig.3*)
- The local view of Tenant T3(*Fig.4*)

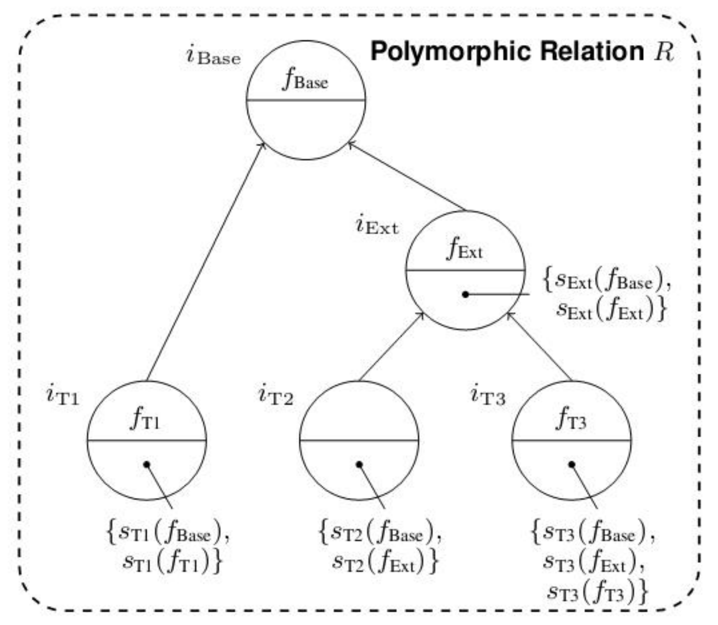
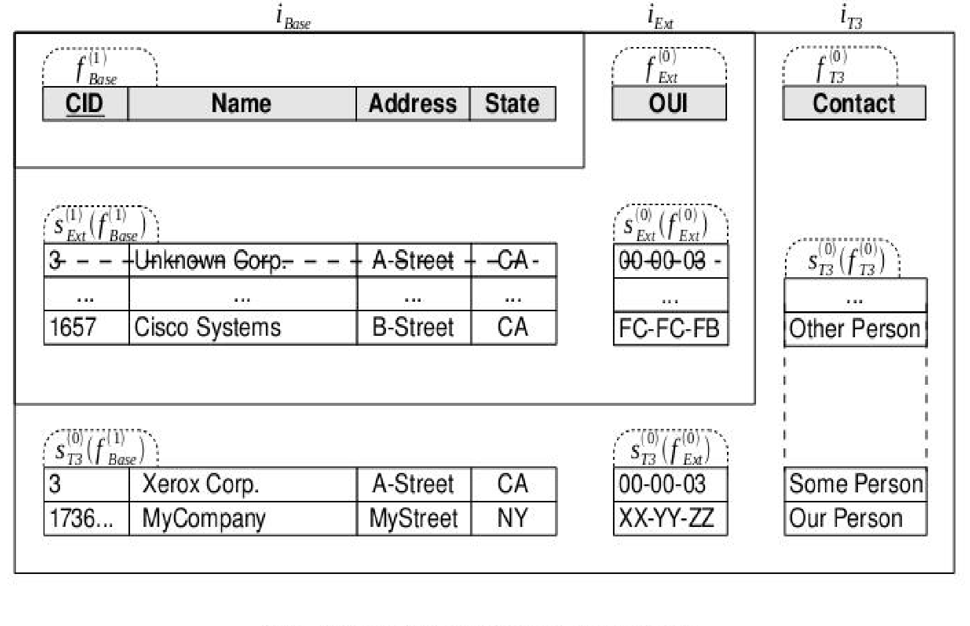

Polymorphic Relation R
i_{Base}
f_{Base}
i_{Ext}
f_{Ext}
$\{s_{\mathrm{Ext}}(f_{\mathrm{Base}}), s_{\mathrm{Ext}}(f_{\mathrm{Ext}})\}$
i_{T1}
f_{T1}
i_{T2}
i_{T3}
f_{T3}
$\{s_{\mathrm{T1}}(f_{\mathrm{Base}}), s_{\mathrm{T1}}(f_{\mathrm{T1}})\}$
$\{s_{\mathrm{T2}}(f_{\mathrm{Base}}), s_{\mathrm{T2}}(f_{\mathrm{Ext}})\}$
$\{s_{\mathrm{T3}}(f_{\mathrm{Base}}), s_{\mathrm{T3}}(f_{\mathrm{Ext}}), s_{\mathrm{T3}}(f_{\mathrm{T3}})\}$

Fig. 3: Example of a Polymorphic Relation R


i_{Base}
i_{Ext}
i_{T3}
$f^{(1)}_{Base}$
$f^{(0)}_{Ext}$
$f^{(0)}_{T3}$
$s^{(1)}_{Ext}(f^{(1)}_{Base})$
$s^{(0)}_{Ext}(f^{(0)}_{Ext})$
$s^{(0)}_{T3}(f^{(0)}_{T3})$
$s^{(0)}_{T3}(f^{(1)}_{Base})$
$s^{(0)}_{T3}(f^{(0)}_{Ext})$

CID	Name	Address	State	OUI	Contact
~~3~~	~~Unknown Corp.~~	~~A-Street~~	~~CA~~	~~00-00-03~~	
...	...	...	...	...	...
1657	Cisco Systems	B-Street	CA	FC-FC-FB	Other Person
3	Xerox Corp.	A-Street	CA	00-00-03	Some Person
1736...	MyCompany	MyStreet	NY	XX-YY-ZZ	Our Person

Fig. 4: Virtual Private Table for Tenant T_3

FlexScheme

- The relation across all tenants
- *(e.g.T1,T2,T3)*

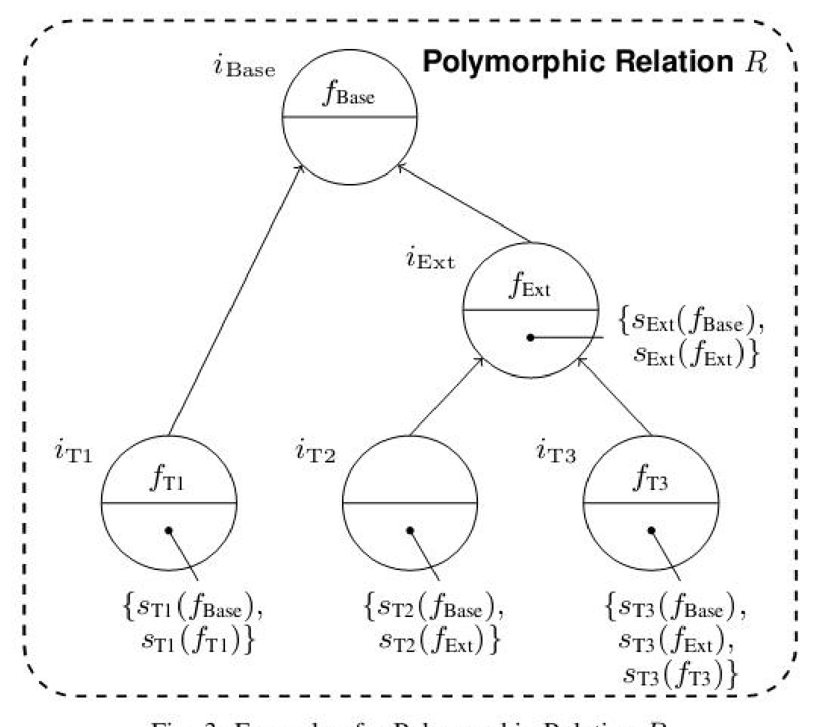
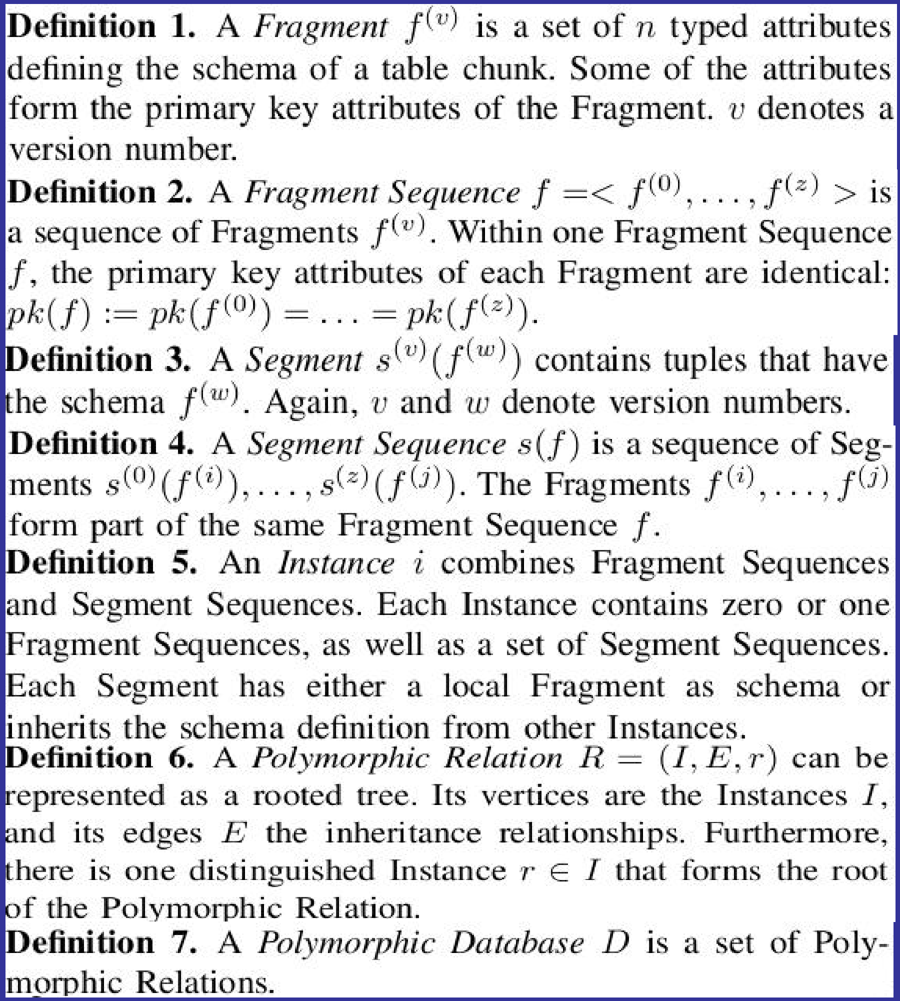

Polymorphic Relation R
i_{Base}
f_{Base}
i_{Ext}
f_{Ext}
$\{s_{\text{Ext}}(f_{\text{Base}}),$
$s_{\text{Ext}}(f_{\text{Ext}})\}$
i_{T1}
f_{T1}
i_{T2}
i_{T3}
f_{T3}
$\{s_{\text{T1}}(f_{\text{Base}}),$
$s_{\text{T1}}(f_{\text{T1}})\}$
$\{s_{\text{T2}}(f_{\text{Base}}),$
$s_{\text{T2}}(f_{\text{Ext}})\}$
$\{s_{\text{T3}}(f_{\text{Base}}),$
$s_{\text{T3}}(f_{\text{Ext}}),$
$s_{\text{T3}}(f_{\text{T3}})\}$

Fig. 3: Example of a Polymorphic Relation R



Definition 1. A *Fragment* $f^{(v)}$ is a set of n typed attributes defining the schema of a table chunk. Some of the attributes form the primary key attributes of the Fragment. v denotes a version number.

Definition 2. A *Fragment Sequence* $f = < f^{(0)}, \ldots, f^{(z)} >$ is a sequence of Fragments $f^{(v)}$. Within one Fragment Sequence f, the primary key attributes of each Fragment are identical: $pk(f) := pk(f^{(0)}) = \ldots = pk(f^{(z)})$.

Definition 3. A *Segment* $s^{(v)}(f^{(w)})$ contains tuples that have the schema $f^{(w)}$. Again, v and w denote version numbers.

Definition 4. A *Segment Sequence* $s(f)$ is a sequence of Segments $s^{(0)}(f^{(i)}), \ldots, s^{(z)}(f^{(j)})$. The Fragments $f^{(i)}, \ldots, f^{(j)}$ form part of the same Fragment Sequence f.

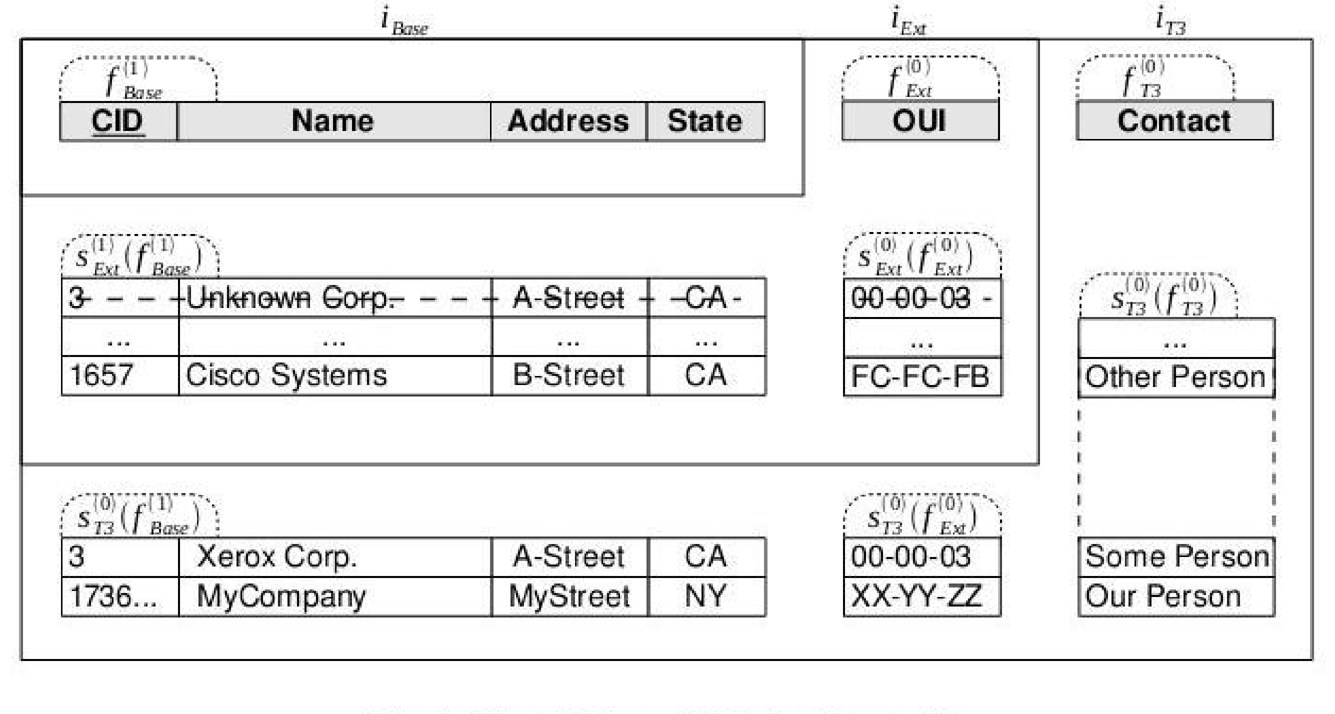
Definition 5. An *Instance* i combines Fragment Sequences and Segment Sequences. Each Instance contains zero or one Fragment Sequences, as well as a set of Segment Sequences. Each Segment has either a local Fragment as schema or inherits the schema definition from other Instances.

Definition 6. A *Polymorphic Relation* $R = (I, E, r)$ can be represented as a rooted tree. Its vertices are the Instances I, and its edges E the inheritance relationships. Furthermore, there is one distinguished Instance $r \in I$ that forms the root of the Polymorphic Relation.

Definition 7. A *Polymorphic Database* D is a set of Polymorphic Relations.

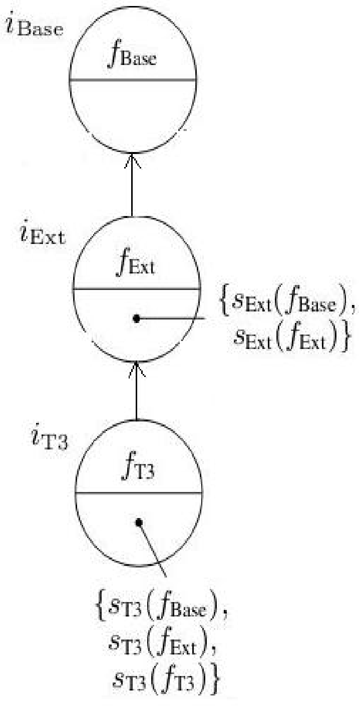
FlexScheme

- The relation for a given tenant
- *(e.g. T3)*


i_{Base}
i_{Ext}
i_{T3}
$f^{(1)}_{Base}$
$f^{(0)}_{Ext}$
$f^{(0)}_{T3}$
$s^{(1)}_{Ext}(f^{(1)}_{Base})$
$s^{(0)}_{Ext}(f^{(0)}_{Ext})$
$s^{(0)}_{T3}(f^{(0)}_{T3})$
$s^{(0)}_{T3}(f^{(1)}_{Base})$
$s^{(0)}_{T3}(f^{(0)}_{Ext})$

CID	Name	Address	State	OUI	Contact
~~3~~	~~Unknown Corp.~~	~~A-Street~~	~~CA~~	~~00-00-03~~	
...	...	...	...	...	...
1657	Cisco Systems	B-Street	CA	FC-FC-FB	Other Person
3	Xerox Corp.	A-Street	CA	00-00-03	Some Person
1736...	MyCompany	MyStreet	NY	XX-YY-ZZ	Our Person

Fig. 4: Virtual Private Table for Tenant T_3


i_{Base}
f_{Base}
i_{Ext}
f_{Ext}
$\{s_{\mathrm{Ext}}(f_{\mathrm{Base}}), s_{\mathrm{Ext}}(f_{\mathrm{Ext}})\}$
i_{T3}
f_{T3}
$\{s_{\mathrm{T3}}(f_{\mathrm{Base}}), s_{\mathrm{T3}}(f_{\mathrm{Ext}}), s_{\mathrm{T3}}(f_{\mathrm{T3}})\}$

Physical Data Organization Of Segment Sequences

- Snapshot Approach

- Data Structures

- Pointwise Access
- Scan Access

- Creation of New Segments
- Purge Operation

- Space Efficiency

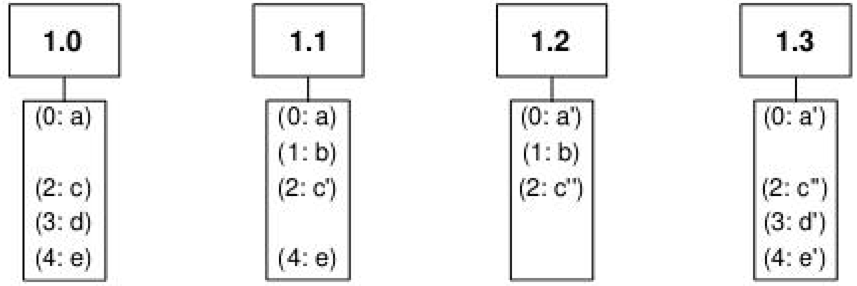


Fig. 5: Snapshot Approach

Physical Data Organization Of Segment Sequences

- Dictionary Approach

- Data Structures

- Pointwise Access
- Scan Access

- Creation of New Segments
- Purge Operation

- Space Efficiency

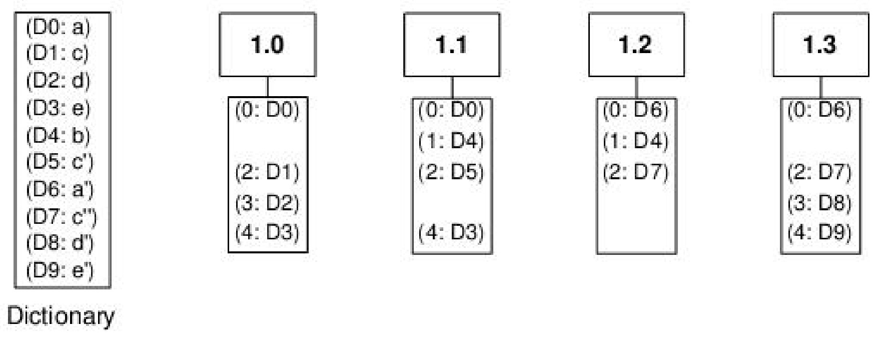


Fig. 6: Dictionary Approach

Physical Data Organization Of Segment Sequences

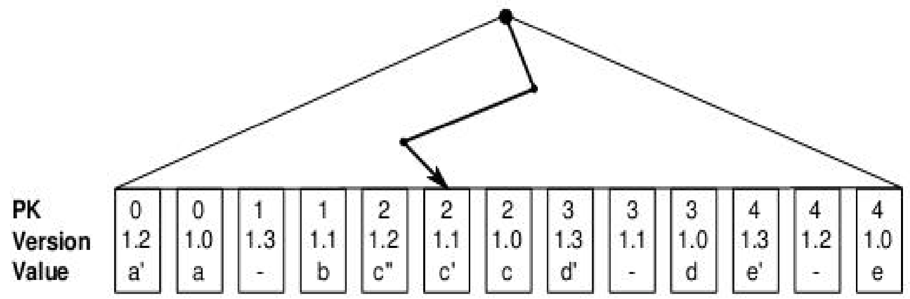
- Temporal Approach

- Data Structures

- Pointwise Access
- Scan Access

- Creation of New Segments
- Purge Operation

- Space Efficiency



PK	0	0	1	1	2	2	2	3	3	3	4	4	4
Version	1.2	1.0	1.3	1.1	1.2	1.1	1.0	1.3	1.1	1.0	1.3	1.2	1.0
Value	a'	a	-	b	c"	c'	c	d'	-	d	e'	-	e

Fig. 7: Temporal Approach

Physical Data Organization Of Segment Sequences

- Differential Delta Approach

- Data Structures

- Pointwise Access
- Scan Access

- Creation of New Segments
- Purge Operation

- Space Efficiency

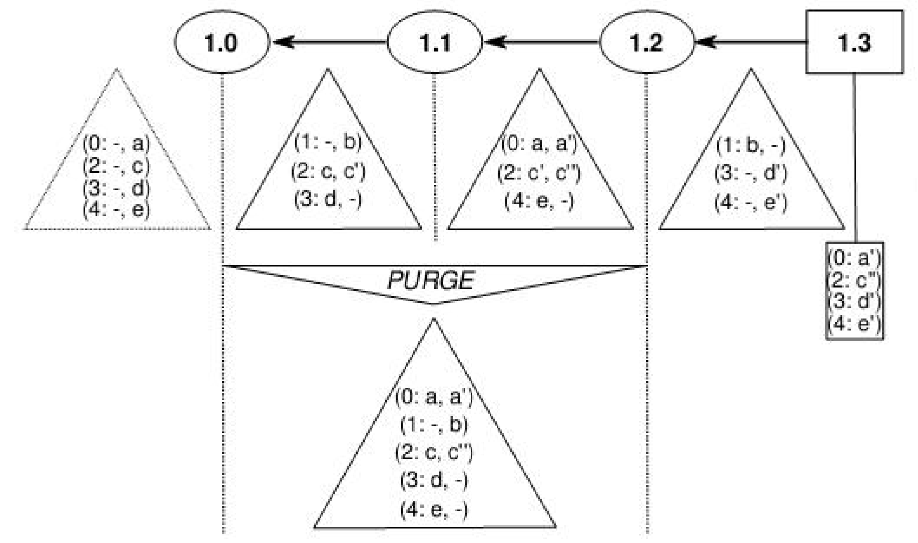


Fig. 8: Differential Delta Data Approach

Physical Data Organization Of Segment Sequences

- XOR Delta Approach

- Data Structures

- Pointwise Access
- Scan Access

- Creation of New Segments
- Purge Operation

- Space Efficiency

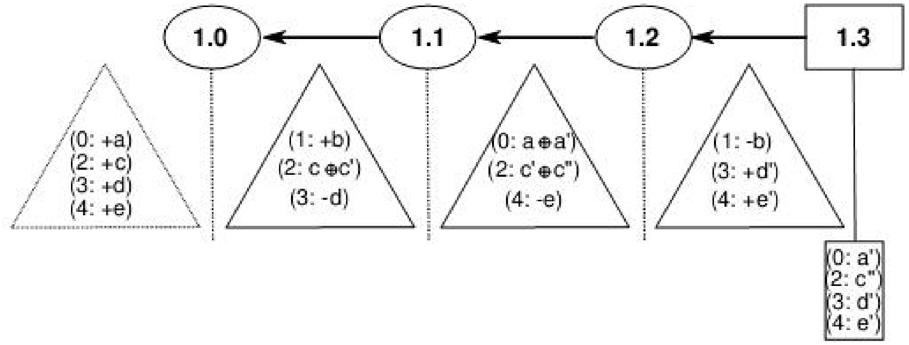


Fig. 9: XOR Delta Approach

XOR Delta Approach

- XOR Delta Approach

- Pointwise Access

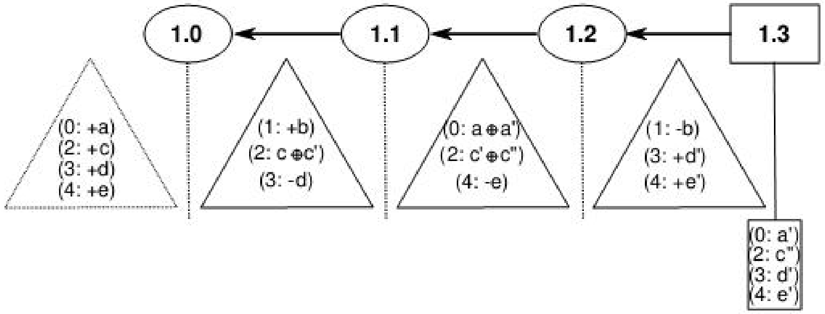
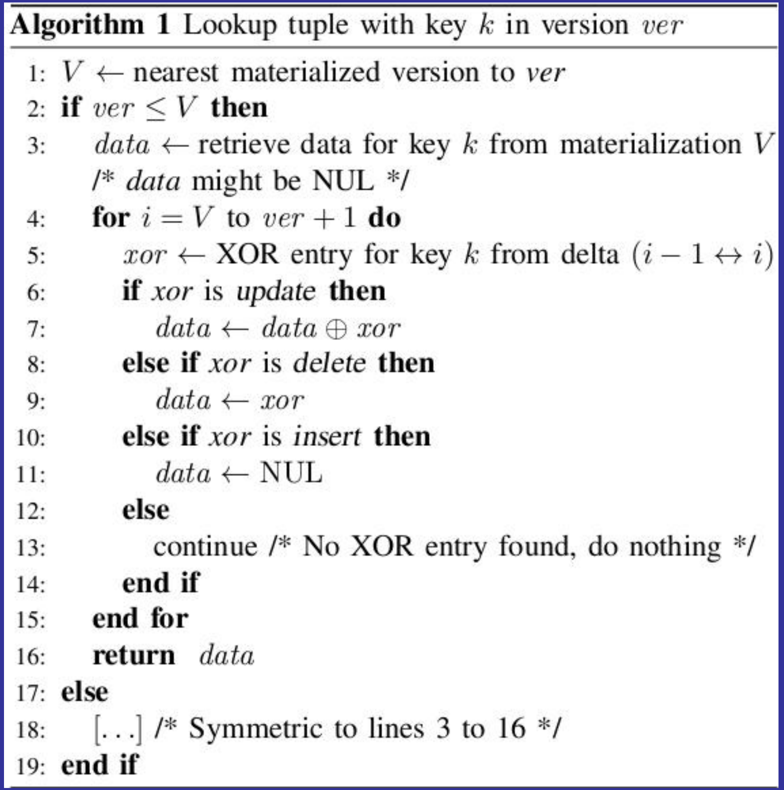


Fig. 9: XOR Delta Approach

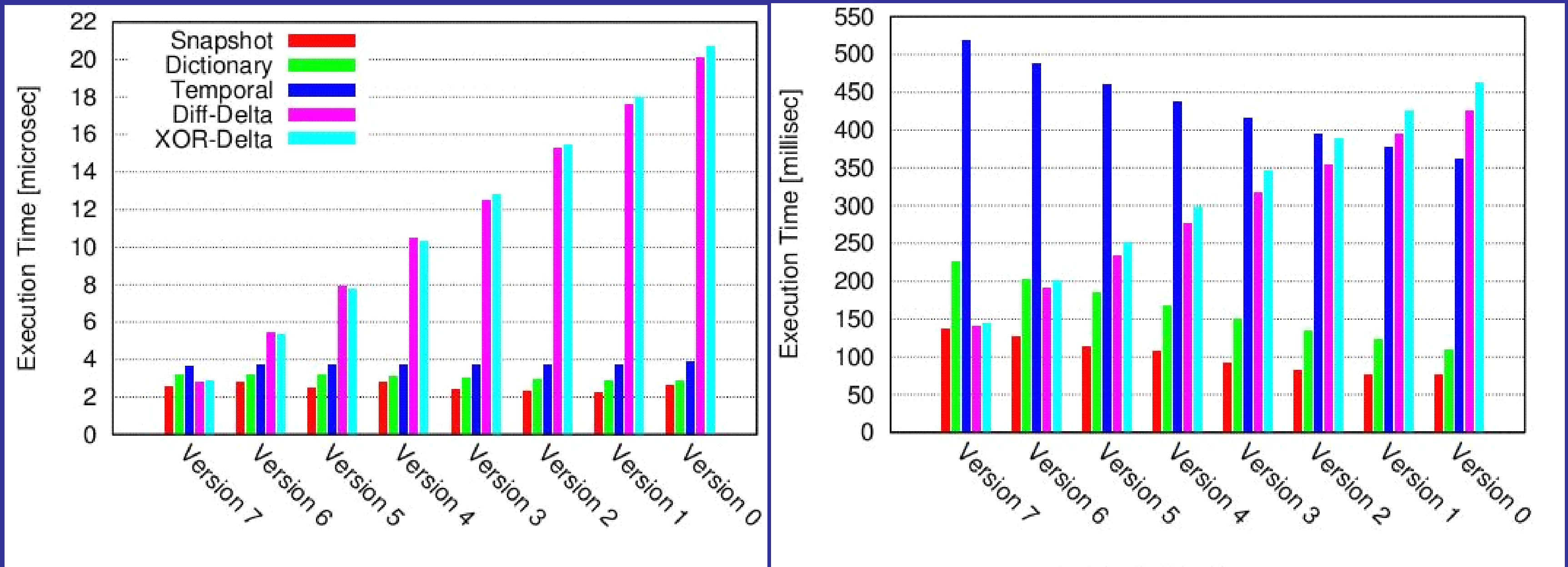


Algorithm 1 Lookup tuple with key k in version ver

```
1:  $V \leftarrow$ nearest materialized version to $ver$
2:  if $ver \leq V$ then
3:      $data \leftarrow$ retrieve data for key $k$ from materialization $V$
        /* $data$ might be NUL */
4:      for $i = V$ to $ver + 1$ do
5:          $xor \leftarrow$ XOR entry for key $k$ from delta $(i - 1 \leftrightarrow i)$
6:          if $xor$ is update then
7:              $data \leftarrow data \oplus xor$
8:          else if $xor$ is delete then
9:              $data \leftarrow xor$
10:         else if $xor$ is insert then
11:             $data \leftarrow$ NUL
12:         else
13:             continue /* No XOR entry found, do nothing */
14:         end if
15:     end for
16:     return $data$
17: else
18:     [...] /* Symmetric to lines 3 to 16 */
19: end if
```

Evaluation

- Data Access



(a) Pointwise Lookup

(b) Full Table Scan

Evaluation

- Space Requirements

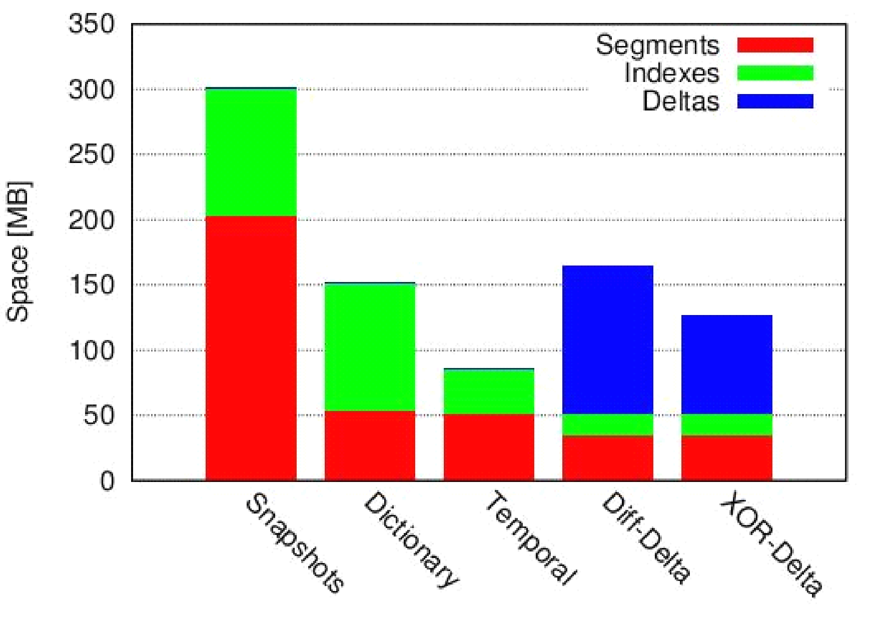
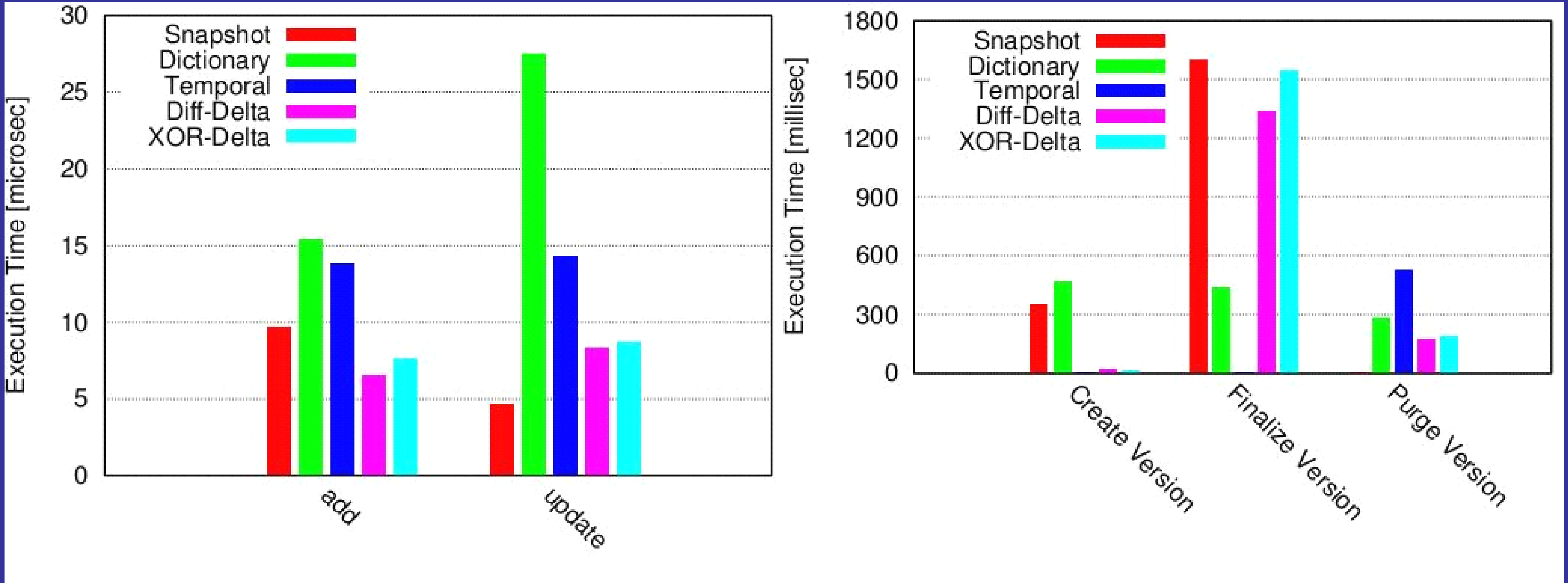


Fig. 11: Space Requirements

Evaluation

- Maintenance Operations



(a) Data Changes

(b) Versioning Operations

Thank You !