Einsatz und Realisierung von Datenbanksystemen

ERDB Übungsleitung

Maximilian {Bandle, Schüle}, Josef Schmeißer

i3erdb@in.tum.de

Folien erstellt von Maximilian Bandle & Alexander Beischl

Organisatorisches
Disclaimer

Die Folien werden von der Übungsleitung allen Tutoren zur Verfügung gestellt.

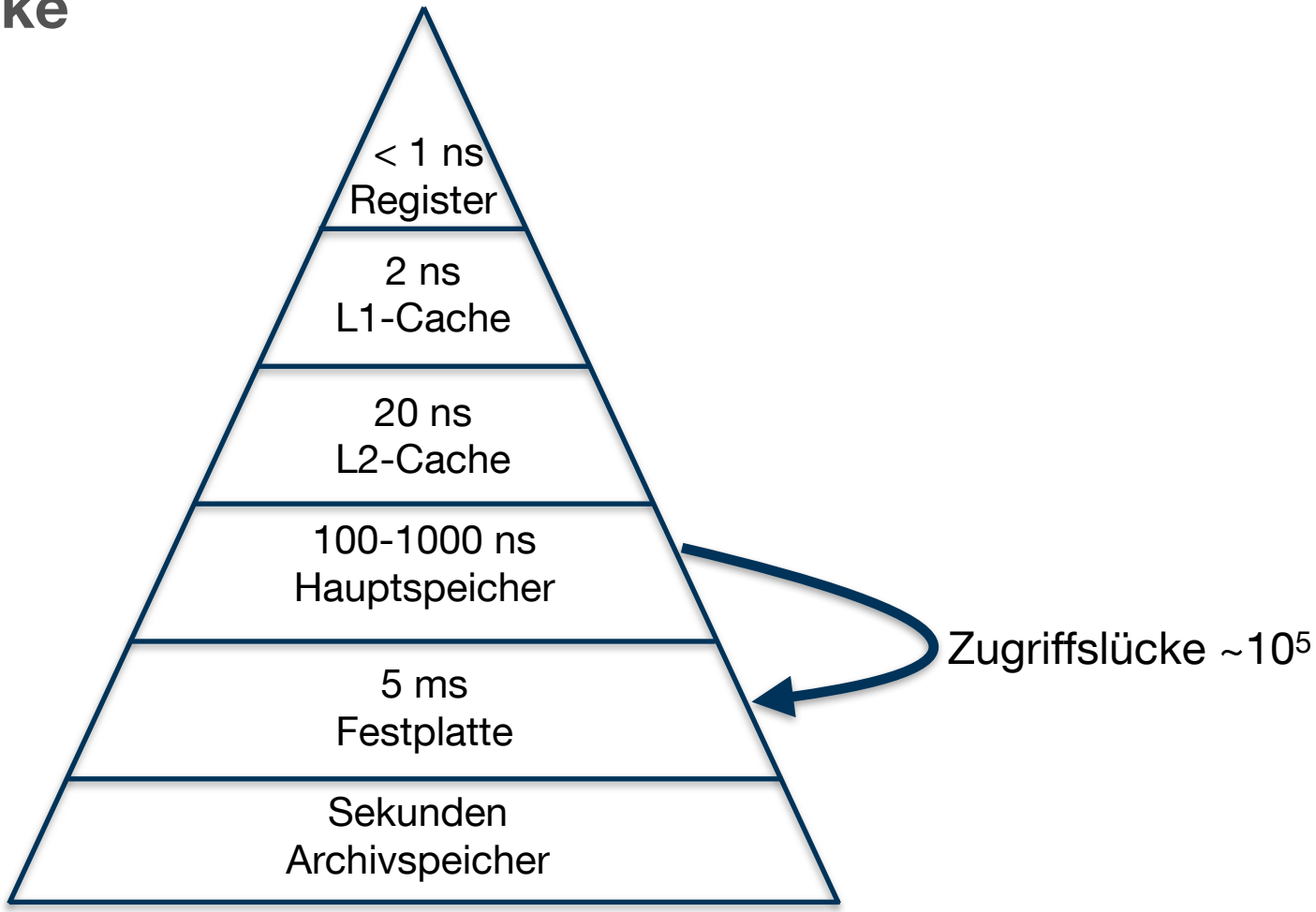
Sollte es Unstimmigkeiten zu den Vorlesungsfolien von Prof. Kemper geben, so sind die Folien aus der Vorlesung ausschlaggebend.

Falls Ihr einen Fehler oder eine Unstimmigkeit findet, schreibt an i3erdb@in.tum.de mit Angabe der Foliennummer.

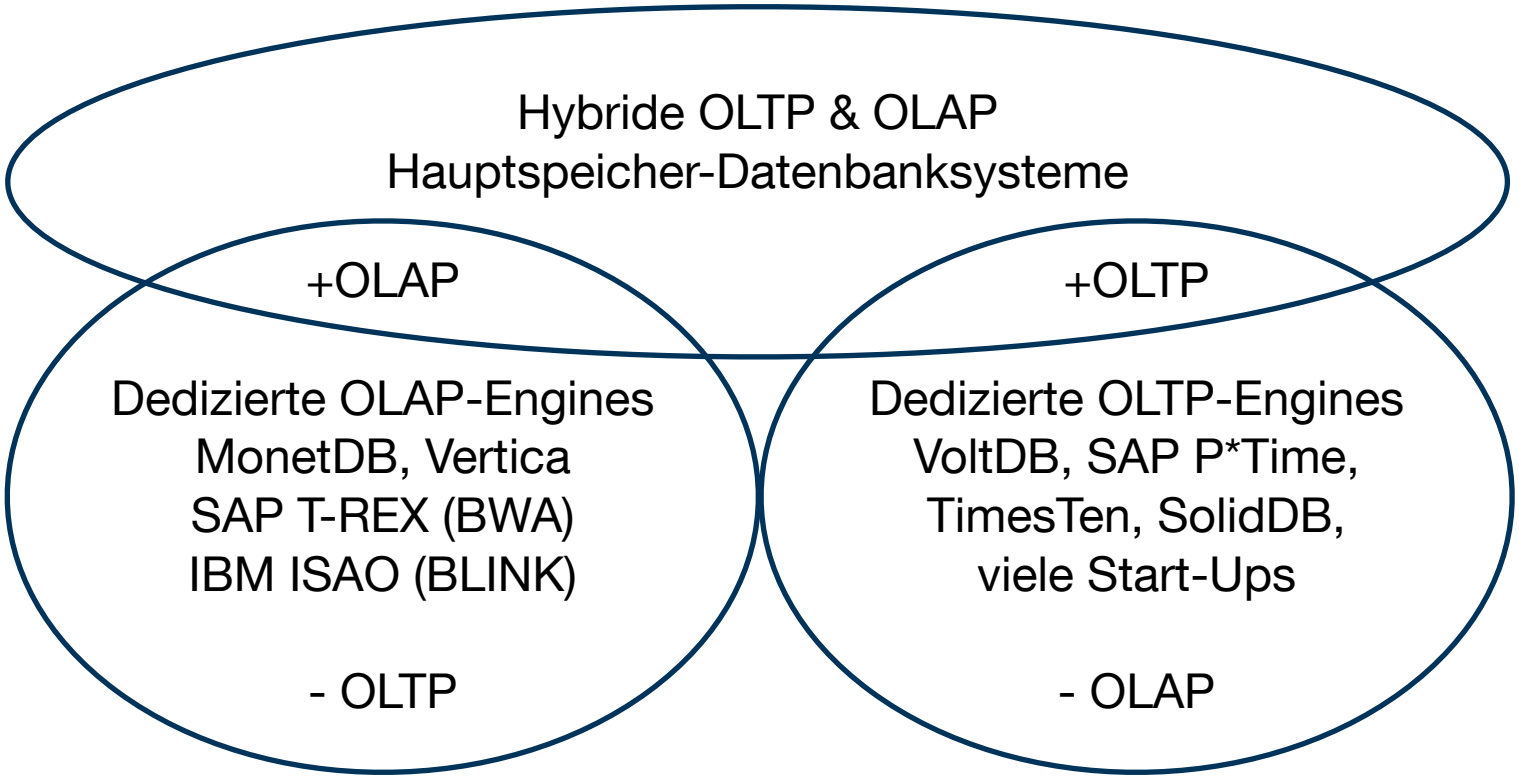
Hauptspeicher-Datenbanken

Hauptspeicher-Datenbanken

Zugriffslücke

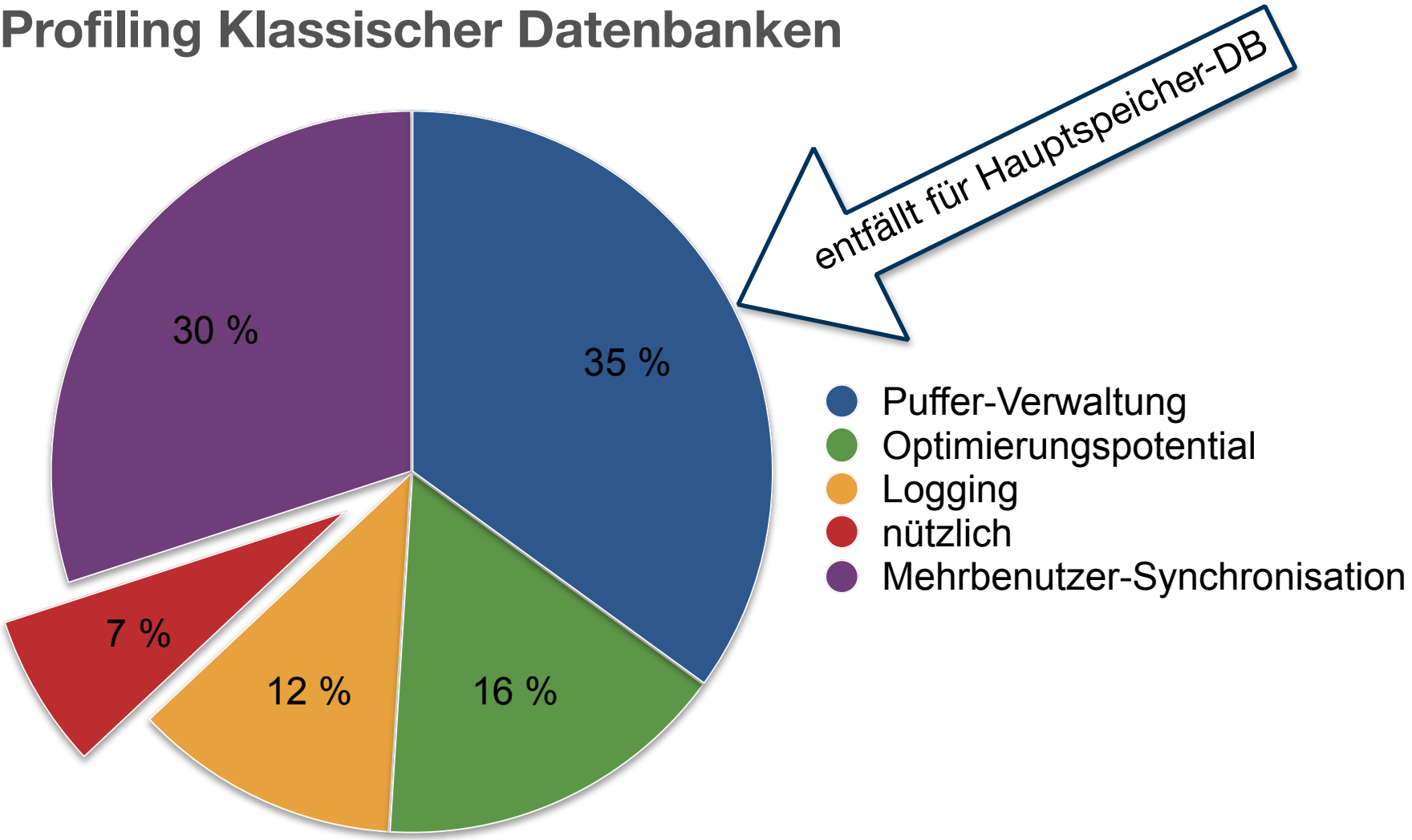


Hauptspeicher-Datenbanken
Einsatz



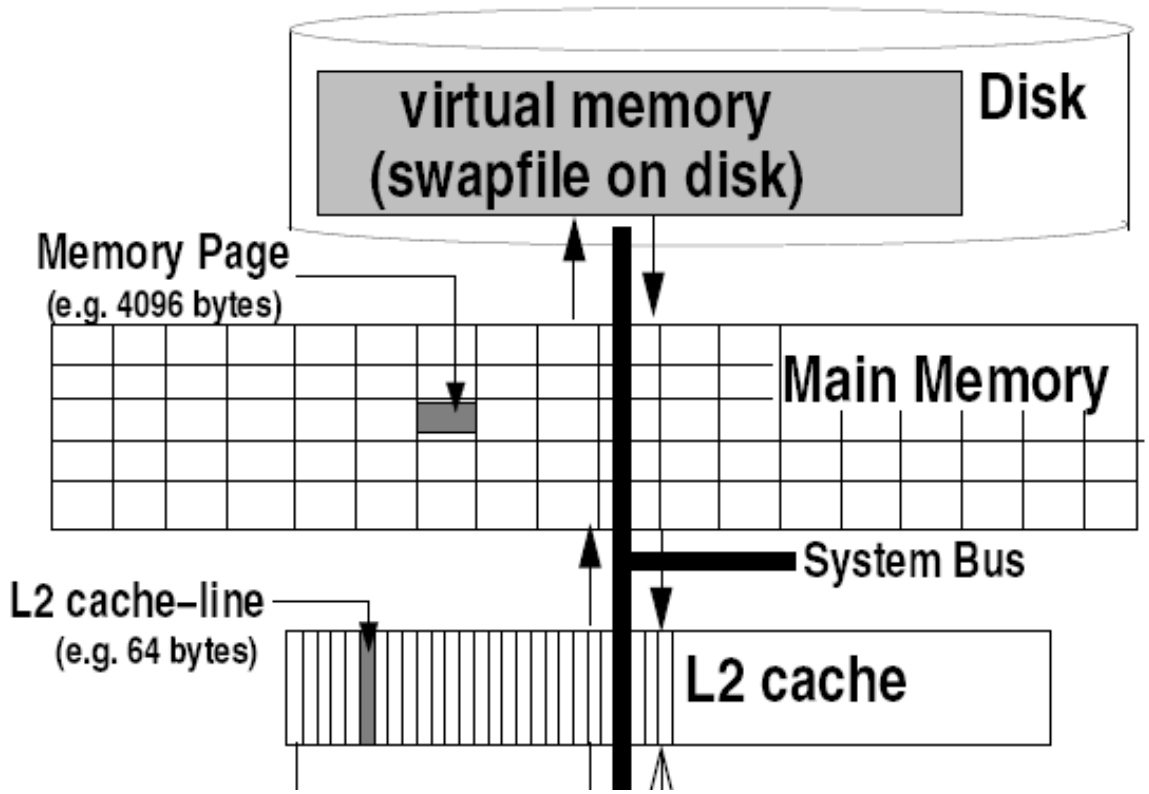
Hauptspeicher-Datenbanken

Profiling Klassischer Datenbanken



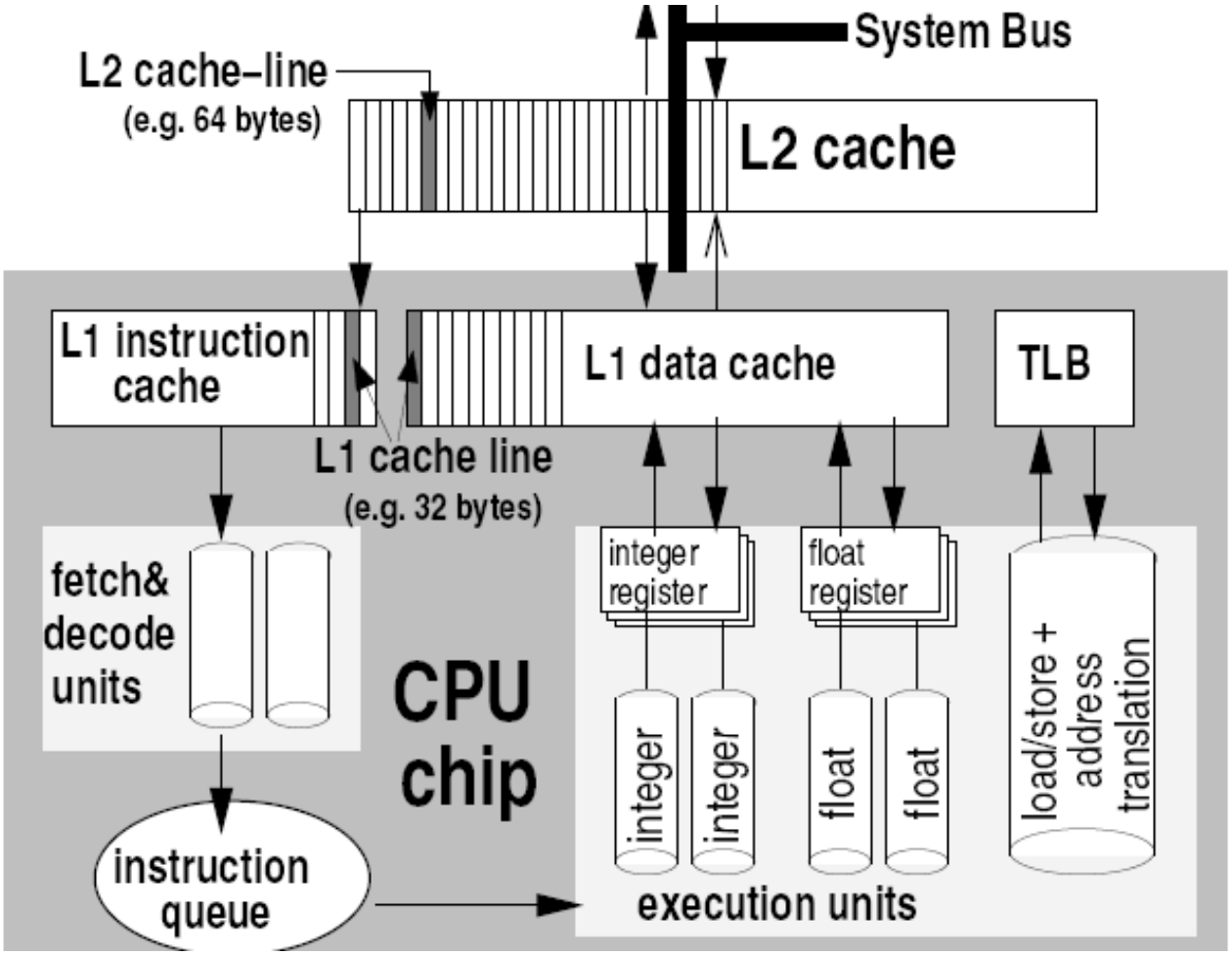
Hauptspeicher-Datenbanken

Speicherhierarchie



Hauptspeicher-Datenbanken

Speicherhierarchie



Aufgabe 1

HyPer schafft 120.000 Transaktionen pro Sekunde. Pro Transaktion werden 120 Byte in die Log geschrieben. Berechnen Sie den benötigten Durchsatz zum Schreiben der Log.

Die Datenbank läuft für einen Monat und stürzt dann ab. Es wurde kein Snapshot erstellt. Berechnen Sie die Recoveryzeit. Gehen Sie davon aus, dass die Recovery durch die Festplatte limitiert ist (100 MiB / s). Wieviel Log Einträge werden pro Sekunde recovert?

Hauptspeicher-Datenbanken

Row- vs Column-Store

Column Store

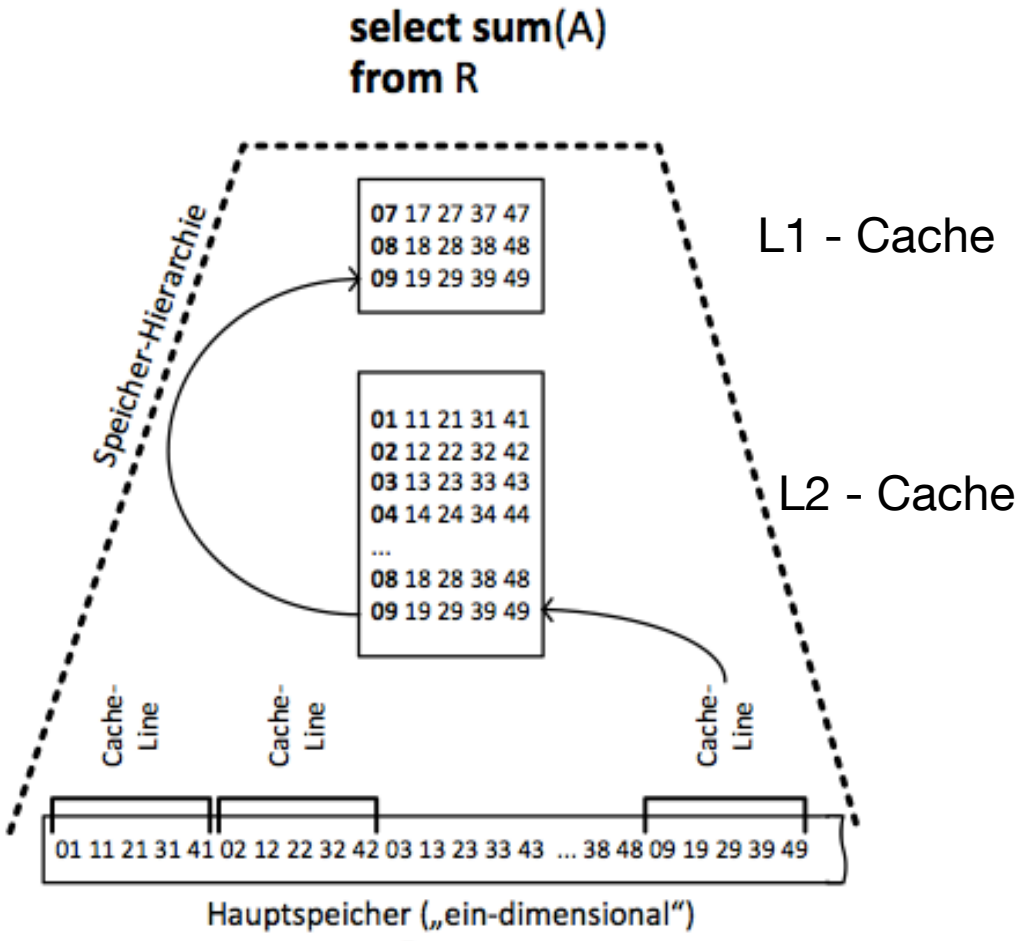
Name	MatrNr	Semester	Fach	Nebenfach
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	362101	10	Info	Mathe

Row Store

Name	MatrNr	Semester	Fach	Nebenfach
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	362101	10	Info	Mathe

Hauptspeicher-Datenbanken

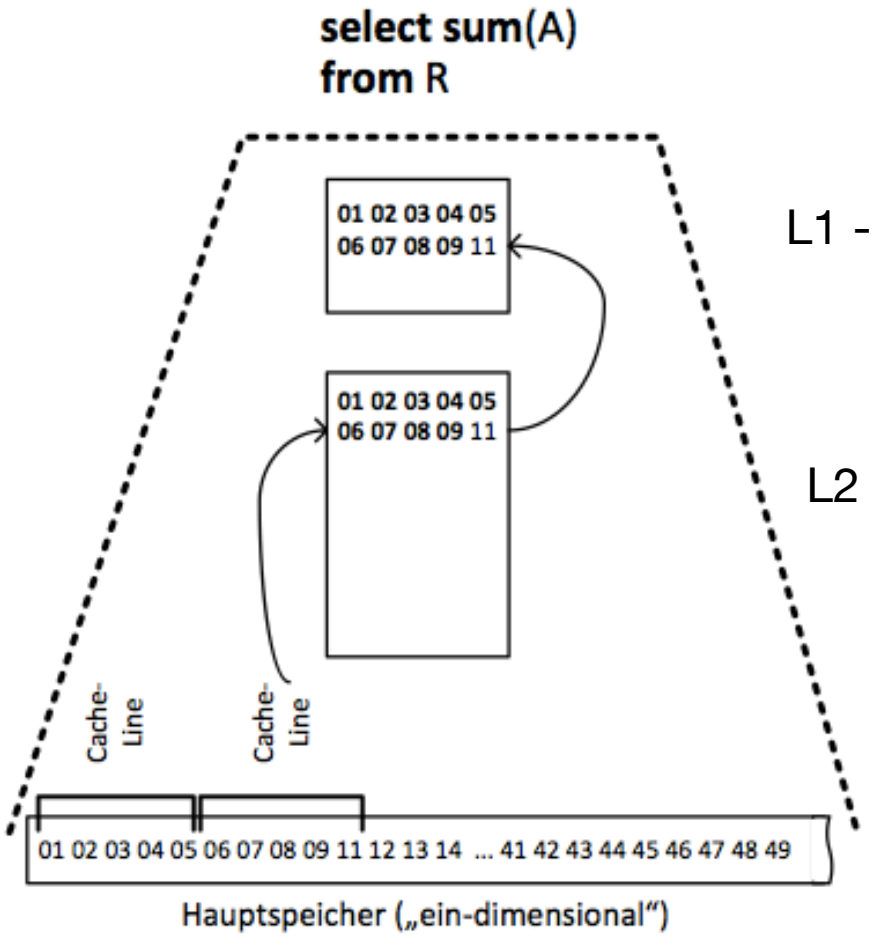
Row-Store



Speicherstruktur

A	B	C	D	E
01	11	21	31	41
02	12	22	32	42
03	13	23	33	43
04	14	24	34	44
05	15	25	35	45
06	16	26	36	46
07	17	27	37	47
08	18	28	38	48
09	19	29	39	49

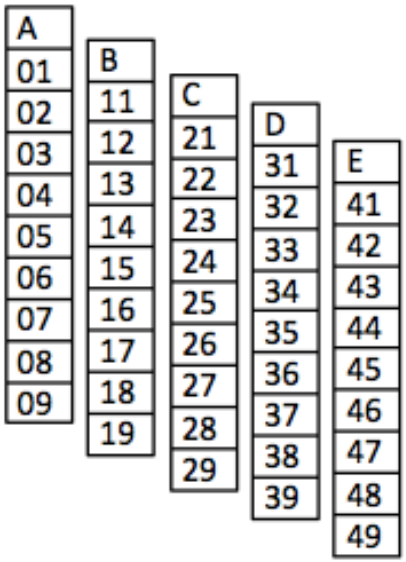
Hauptspeicher-Datenbanken
Column-Store



L1 - Cache

L2 - Cache

Speicherstruktur



Aufgabe 2

In (pseudo) Java kann eine ‘Row-Store-artige’ Datenstruktur wie folgt angelegt werden:

```
class Tuple {
  int MatrNr;
  String Name;
  int Semester;
}
Tuple data[]=new Tuple[10000];
```

Notieren Sie, wie die Daten in Form eines Column Stores gehalten werden können in (pseudo) Java.

Erklären Sie Ihrem Tutor, welche Vor- und Nachteile Row- und Column Stores jeweils haben. Was würden Sie für Amazons Webseite verwenden? Was verwenden Sie für die Controlling Datenbank?

Aufgabe 3

Gegeben eine Tabelle *Produkte* mit folgendem Schema und 10000 Einträgen:

Id (8 Byte) | Name (32 Byte) | Preis (8 Byte) | Anzahl (8 Byte)

Wieviele Daten werden für folgende Queries in die CPU-Caches geladen? Unterscheiden sie jeweils zwischen Row und Column Store.

1. *select * from Produkte*
2. *select Anzahl from Produkte*

Hauptspeicher-Datenbanken
Row vs Column Store

Die Anzahl der Tupel in der Relation Studenten ist nicht bekannt, wir verwenden |S| als Abschätzung.
Für die MatrNr existiert ein Index.

Row Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
...				

Column Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
...	...	...	...	...

Hauptspeicher-Datenbanken
Row vs Column Store

select *

from Studenten;

Row Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
…				

Die Anzahl der Tupel in der Relation Studenten ist nicht bekannt, wir verwenden |S| als Abschätzung.
Für die MatrNr existiert ein Index. 1 B = 1 Byte (8Bit)

Column Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
…	…	…	…	…

Hauptspeicher-Datenbanken
Row vs Column Store

select *

from Studenten;

Row Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
...				

Die Anzahl der Tupel in der Relation Studenten ist nicht bekannt, wir verwenden |S| als Abschätzung.
Für die MatrNr existiert ein Index. 1 B = 1 Byte (8Bit)

RowStore:

1 Tupel: 32B + 3B + 1B + 4B +16B = 56B

$$\#\text{Cachelines} = \lceil |S| * (56\text{Byte}/64\text{Byte}) \rceil$$
$$= \lceil |S| * (7/8) \rceil$$

Hauptspeicher-Datenbanken
Row vs Column Store

```
select *
from Studenten;
```

ColumnStore:

#Cachelines = ⌈|S|*(32B/64B)⌉ + ⌈|S|*(3B/64B)⌉ + ⌈|S|*(1B/64B)⌉ + ⌈|S|*(4B/64B)⌉ + ⌈|S|*(16B/64B)⌉

= ⌈|S|*(32B+3B+1B+4B+16B)/64B⌉

= ⌈|S|*56B/64B⌉

= ⌈|S|*7/8⌉

Die Anzahl der Tupel in der Relation Studenten ist nicht bekannt, wir verwenden |S| als Abschätzung.
Für die MatrNr existiert ein Index. 1 B = 1 Byte (8Bit)

Column Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
…	…	…	…	…

Hauptspeicher-Datenbanken
Row vs Column Store

```
select Name, MatrNr
from Studenten
where Semester = 10;
```

Row Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
…				

Die Anzahl der Tupel in der Relation Studenten ist nicht bekannt, wir verwenden |S| als Abschätzung.
Für die MatrNr existiert ein Index. 1 B = 1 Byte (8Bit)

Column Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
…	…	…	…	…

Hauptspeicher-Datenbanken
Row vs Column Store

```
select Name, MatrNr
from Studenten
where Semester = 10;
```

Row Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
...				

Die Anzahl der Tupel in der Relation Studenten ist nicht bekannt, wir verwenden |S| als Abschätzung.
Für die MatrNr existiert ein Index. 1 B = 1 Byte (8Bit)

RowStore:

$$\#\text{Cachelines} = \lceil |S| * (56\text{Byte}/64\text{Byte}) \rceil$$
$$= \lceil |S| * (7/8) \rceil$$
$$= \lceil |S| * 0{,}875 \rceil$$

Hauptspeicher-Datenbanken

Row vs Column Store

Schätzung der Selektivität von 1/10 ist unrealistisch, insbesondere die Folge das nur 1/10 der CLs gelesen werden. Erfüllt nur den Zweck eines Beispiels.

```
select Name, MatrNr
from Studenten
where Semester = 10;
```

ColumnStore:

$$\#Cachelines = \lceil |S| * 1B/64B \rceil + \lceil |S| * 32B/64B * 1/10 \rceil + \lceil |S| * 3B/64B * 1/10 \rceil$$

$$= \lceil |S| * (1B/64B + 32B/640B + 3B/640B) \rceil$$

$$= \lceil |S| * (10B + 32B + 3B)/640B \rceil$$

$$= \lceil |S| * 45/640 \rceil$$

$$= \lceil |S| * 0{,}070 \rceil$$

Die Anzahl der Tupel in der Relation Studenten ist nicht bekannt, wir verwenden |S| als Abschätzung.
Für die MatrNr existiert ein Index. 1 B = 1 Byte (8Bit)

Column Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
…	…	…	…	…

Hauptspeicher-Datenbanken
Row vs Column Store

```
select Name, MatrNr
from Studenten
where MatrNr = %;
```

Row Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
…				

Die Anzahl der Tupel in der Relation Studenten ist nicht bekannt, wir verwenden |S| als Abschätzung.
Für die MatrNr existiert ein Index. 1 B = 1 Byte (8Bit)

Column Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
…	…	…	…	…

Hauptspeicher-Datenbanken
Row vs Column Store

```
select Name, MatrNr
from Studenten
where MatrNr = %;
```

Row Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
...				

Die Anzahl der Tupel in der Relation Studenten ist nicht bekannt, wir verwenden |S| als Abschätzung.
Für die MatrNr existiert ein Index. 1 B = 1 Byte (8Bit)

RowStore:

#Cachelines = ⌈56B/64B⌉ = 1

Hier wird der Index von MatrNr genutzt. Deshalb muss nur das Tupel mit der gesuchten MatrNr geladen werden. Dieser umfasst 1 Cacheline.

Hauptspeicher-Datenbanken
Row vs Column Store

```
select Name, MatrNr
from Studenten
where MatrNr = %;
```

ColumnStore:

#Cachelines = ⌈32B/64B⌉ + ⌈3B/64B⌉ = 2

Hier wird ebenfalls wieder der Index von MatrNr genutzt, sodass nur der Namen und die MatrNr des Tupels mit der gesuchten MatrNr aus den jeweiligen Tabellengeladen wird.

Die Anzahl der Tupel in der Relation Studenten ist nicht bekannt, wir verwenden |S| als Abschätzung.
Für die MatrNr existiert ein Index. 1 B = 1 Byte (8Bit)

Column Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
…	…	…	…	…

Hauptspeicher-Datenbanken
Row vs Column Store

Insert into Studenten VALUES(…);

Row Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
…				

Die Anzahl der Tupel in der Relation Studenten ist nicht bekannt, wir verwenden |S| als Abschätzung.
Für die MatrNr existiert ein Index. 1 B = 1 Byte (8Bit)

Column Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
…	…	…	…	…

Hauptspeicher-Datenbanken
Row vs Column Store

Insert into Studenten VALUES(…);

Row Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
…				

Die Anzahl der Tupel in der Relation Studenten ist nicht bekannt, wir verwenden |S| als Abschätzung.
Für die MatrNr existiert ein Index. 1 B = 1 Byte (8Bit)

RowStore:

#Cachelines = ⌈56B/64B⌉ = 1

Hauptspeicher-Datenbanken
Row vs Column Store

Insert into Studenten VALUES(…);

ColumnStore:

#Cachelines = ⌈32B/64B⌉ + ⌈3B/64B⌉ +

⌈1B/64B⌉ + ⌈4B/64B⌉ + ⌈16B/64B⌉ = 5

Da jedes Attribut muss einzeln in die jeweilige Tabelle eingefügt werden.

Die Anzahl der Tupel in der Relation Studenten ist nicht bekannt, wir verwenden |S| als Abschätzung.
Für die MatrNr existiert ein Index. 1 B = 1 Byte (8Bit)

Column Store

Name (32Byte)	MatrNr (3Byte)	Semester (1 Byte)	Fach (4Byte)	Nebenfach (16 Byte)
Alex	362148	6	Info	Medizin
Max	362139	6	Info	Physik
David	361299	10	Info	MaschBau
Johannes	362033	8	Info	Mathe
Andre	262101	10	Info	Mathe
…	…	…	…	…

Aufgabe 4

Sie sollen für die Alexander-Maximilians-Universität (AMU) ein Hauptspeicherdatenbanksystem optimieren. In dem System sind die Daten aller Studenten gespeichert. Schätzen Sie für jede der untenstehenden Anfragen einzeln, ob ein Row- oder Column-Store besser geeignet ist.

Relationen

Studenten: MatrNr (8 Byte), Name (48 Byte), Studiengang (4 Byte), Semester (4 Byte)
MatrNr ist der Primärschlüssel der indiziert ist.

Anfragen:

1. select * from Studenten;
2. select Semester, count(*) from Studenten group by Semester;
3. select Name, Studiengang, Semester from Studenten where MatrNr = 42;
4. select Studiengang from Studenten where MatrNr = 42;
5. select * from Studenten where Semester < 5;
6. select * from Studenten where Semester = 25;
7. insert into studenten values(4242, Max Meyer, Info, 7);

Aufgabe 5

Rekonstruieren Sie die ursprüngliche SQL-Anfrage aus dem folgenden (Pseude-)Code eines codegenerierenden Datenbanksystems. Welche Art von Join wurde benutzt? Handelt es sich um Column- oder Row-Store?

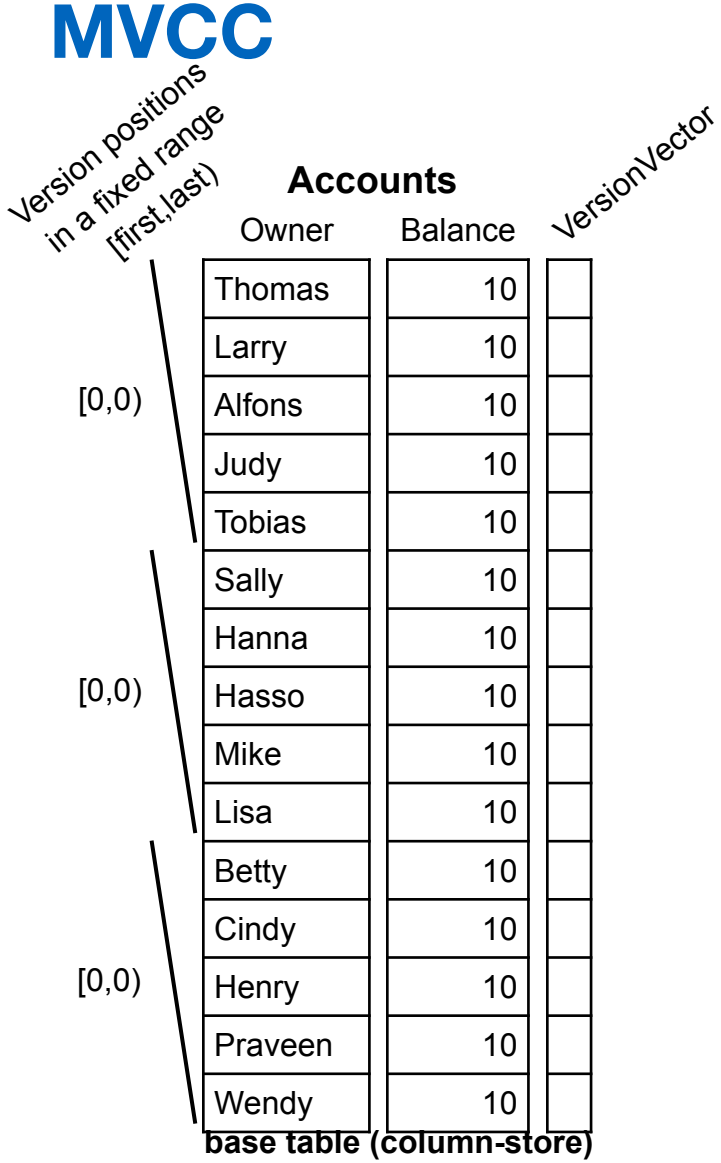
```
class Student { int matrnr; std::string name; int semester; }
class Hoeren { int matrnr; int vorlnr; }
class Result { int vorlnr, a; }

std::vector<Result> compute(std::vector<Student> ses, std::vector<Hoeren> hs){
   std::unordered_multimap<int,Hoeren*,Studenten*> h_map;
   std::unordered_map<int,Studenten*> s_map;
   for(auto h: hes)
      h_map.insert(std::pair<int,Hoeren*>(h.matrnr,&h,nullptr))
   for(auto s: ses)
      s_map.insert(std::pair<int,Studenten*>(s.matrnr,&s))

   // Group by h.vorlnr; avg(s.semester) = sum(s.semester)/count(*)
   std::unordered_map<int,int> count_map;
   std::unordered_map<int,int> sum_map;
   for(auto h: hes){
      count_map.insert(std::pair<int,int>(h.vorlnr,0))
      sum_map.insert(std::pair<int,int>(h.vorlnr,0))
   }
   for(auto h: h_map){
      sum_map[h->second->vorlnr]+=s_map[h->first]->semester
      count_map[h->second->vorlnr]++;
   }
   std::vector<Result> res;
   for(auto r: sum_map)
      res.push_back(r->first,r->second/count_map[r->first]);
   return res;
}
```

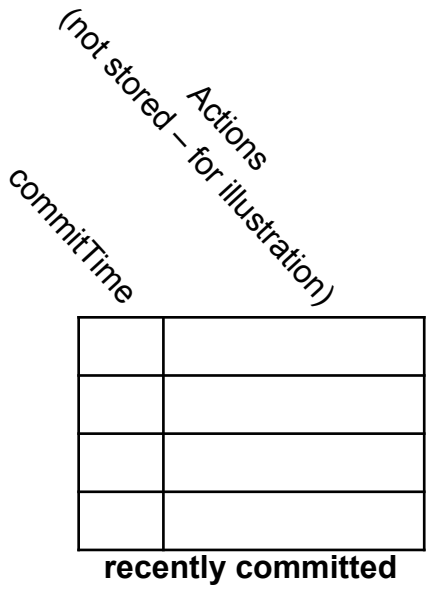
MVCC

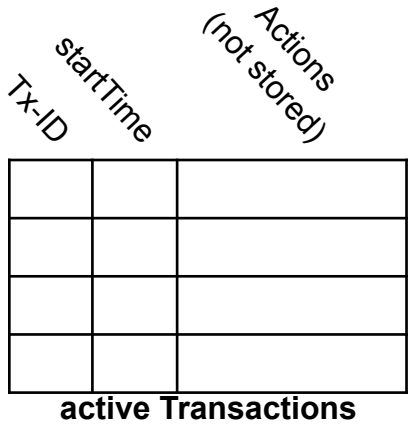
- **M**ulti**v**ersion **C**oncurrency **C**ontrol
- fein granulares Mehrversionsverfahren für die Synchronisation mehrerer paralleler TA
- sehr gut für unterbrechbare TA
- klassische DB nutzen das 2-Phasen-Sperrprotokoll

➡ zu schwerfällig für Hauptspeicher DBs

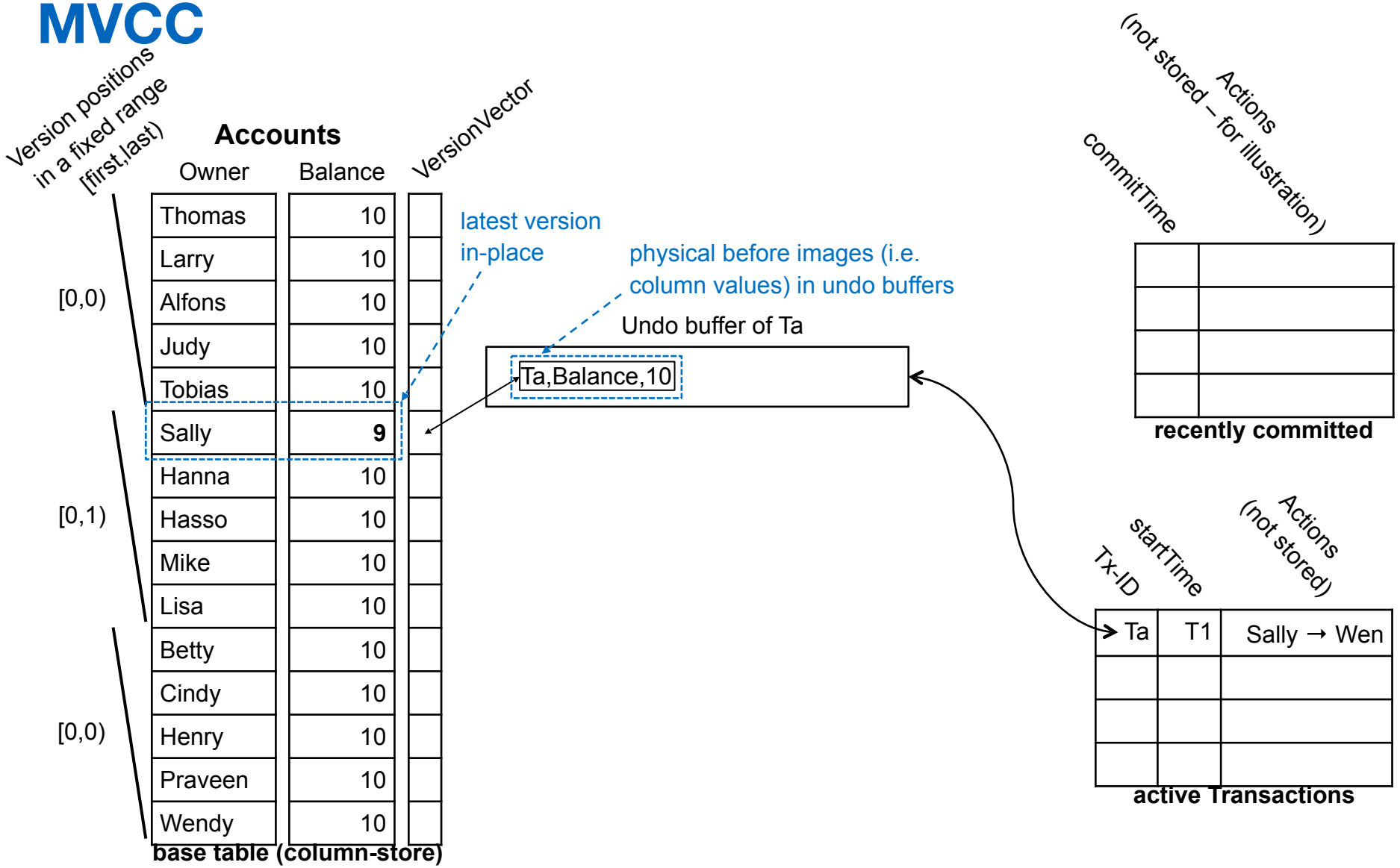
MVCC


Version positions in a fixed range [first,last)
Accounts
Owner
Balance
VersionVector
Thomas 10
Larry 10
Alfons 10
Judy 10
Tobias 10
Sally 10
Hanna 10
Hasso 10
Mike 10
Lisa 10
Betty 10
Cindy 10
Henry 10
Praveen 10
Wendy 10
[0,0)
[0,0)
[0,0)
base table (column-store)

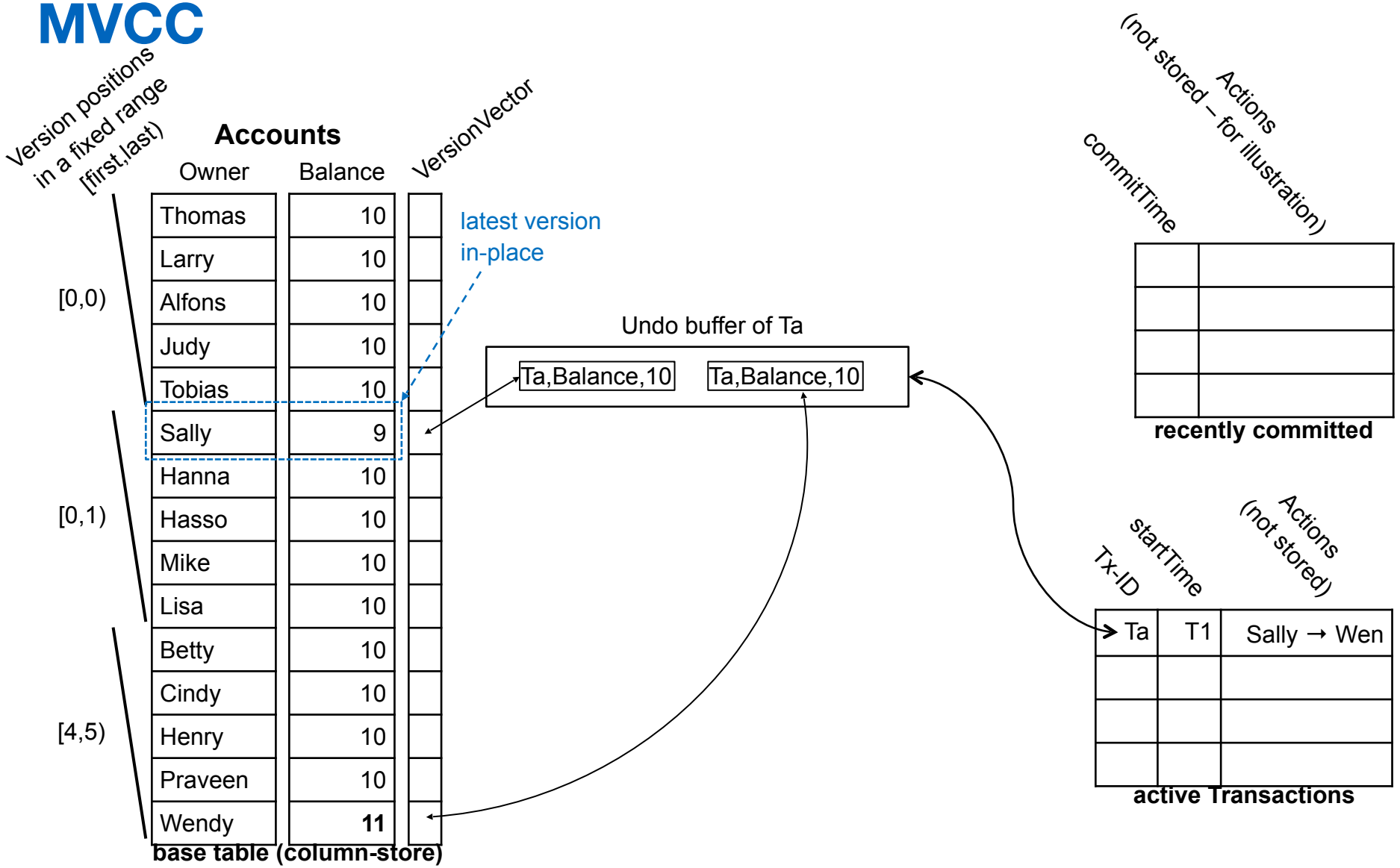

(not stored – for illustration)
Actions
commitTime
recently committed


Actions
(not stored)
startTime
Tx-ID
active Transactions

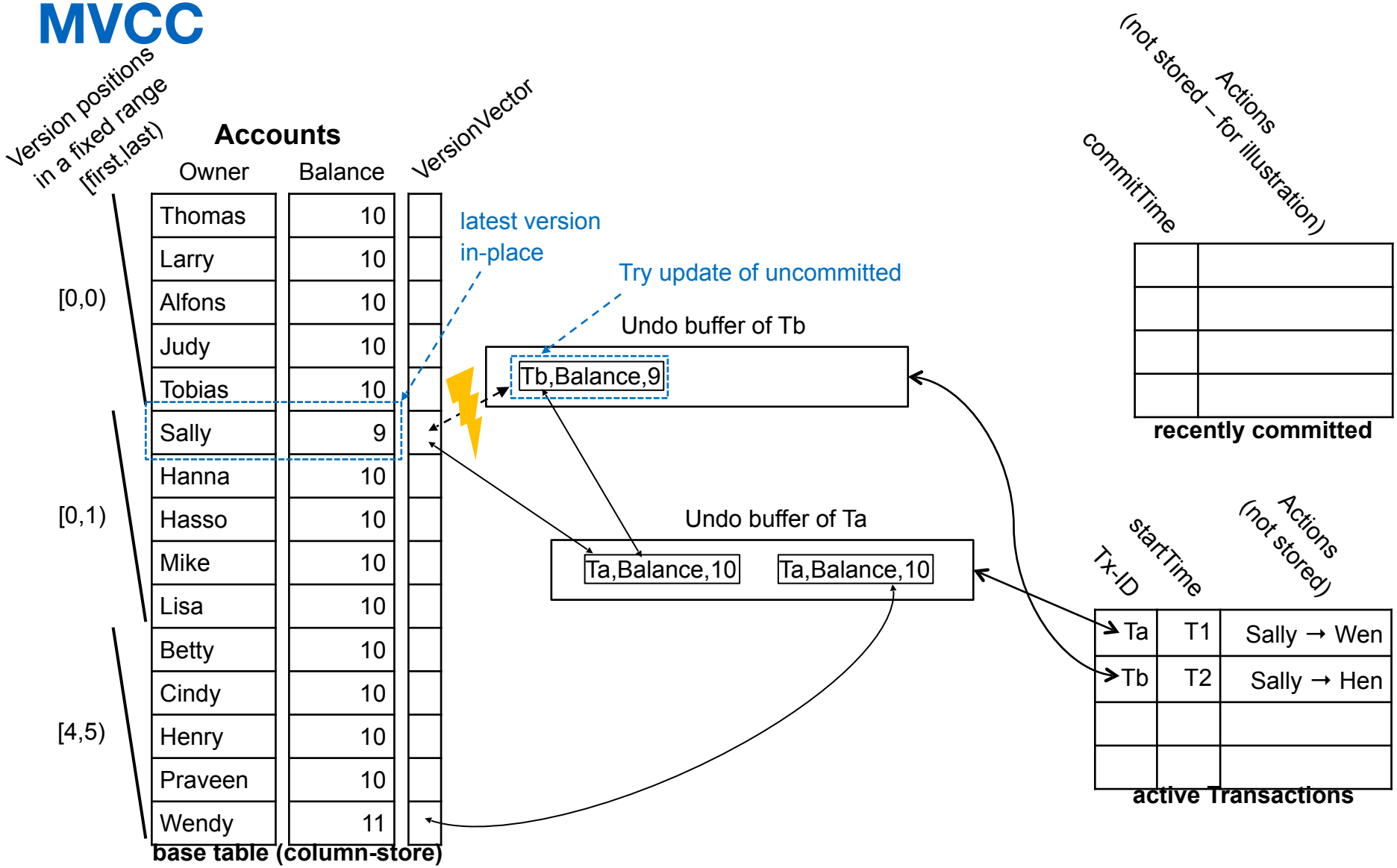
MVCC


Version positions in a fixed range [first,last)
Accounts
Owner
Balance
VersionVector
Thomas 10
Larry 10
Alfons 10
Judy 10
Tobias 10
Sally 9
Hanna 10
Hasso 10
Mike 10
Lisa 10
Betty 10
Cindy 10
Henry 10
Praveen 10
Wendy 10
[0,0)
[0,1)
[0,0)
base table (column-store)
latest version in-place
physical before images (i.e. column values) in undo buffers
Undo buffer of Ta
Ta,Balance,10
(not stored – for illustration)
Actions
commitTime
recently committed
Tx-ID
startTime
Actions (not stored)
Ta
T1
Sally → Wen
active Transactions

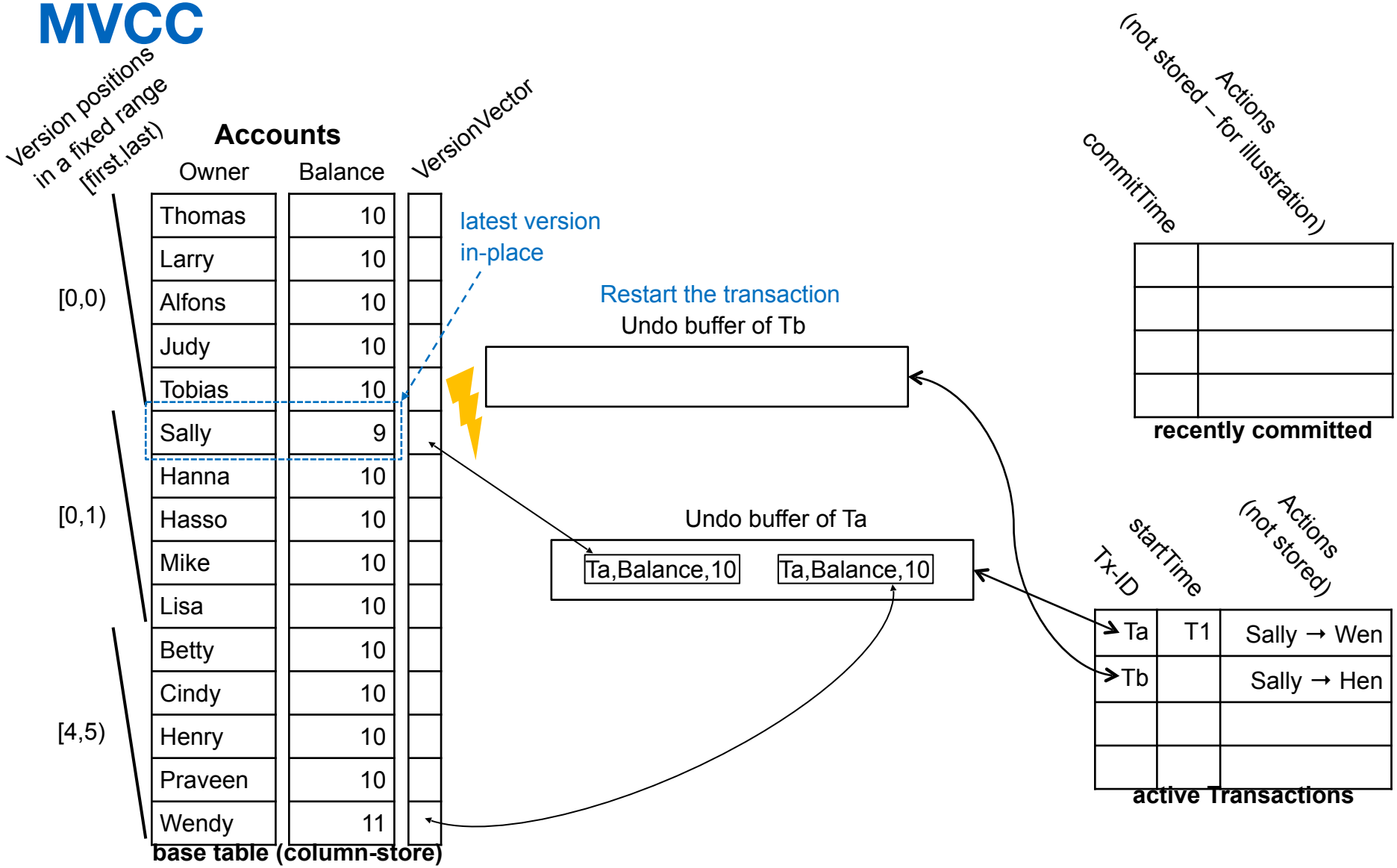
MVCC


Version positions in a fixed range [first,last)
Accounts
Owner
Balance
VersionVector
Thomas 10
Larry 10
Alfons 10
Judy 10
Tobias 10
Sally 9
Hanna 10
Hasso 10
Mike 10
Lisa 10
Betty 10
Cindy 10
Henry 10
Praveen 10
Wendy 11
[0,0)
[0,1)
[4,5)
base table (column-store)
latest version in-place
Undo buffer of Ta
Ta,Balance,10
Ta,Balance,10
(not stored – for illustration)
Actions
commitTime
recently committed
Actions (not stored)
startTime
Tx-ID
Ta
T1
Sally → Wen
active Transactions

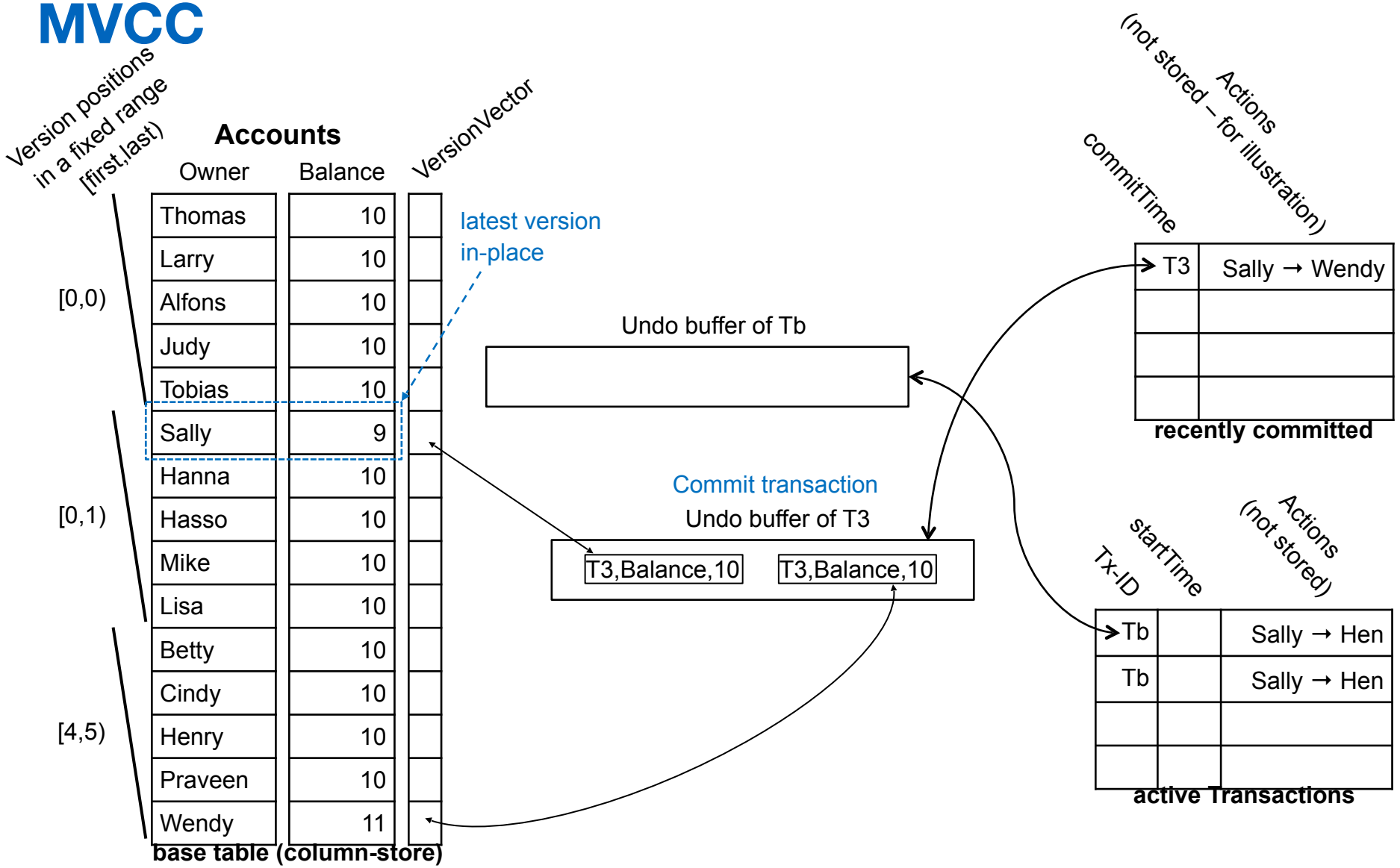
MVCC


Version positions in a fixed range [first,last)
Accounts
Owner
Balance
VersionVector
Thomas 10
Larry 10
Alfons 10
Judy 10
Tobias 10
Sally 9
Hanna 10
Hasso 10
Mike 10
Lisa 10
Betty 10
Cindy 10
Henry 10
Praveen 10
Wendy 11
[0,0)
[0,1)
[4,5)
base table (column-store)
latest version in-place
Try update of uncommitted
Undo buffer of Tb
Tb,Balance,9
Undo buffer of Ta
Ta,Balance,10
Ta,Balance,10
commitTime
Actions (not stored – for illustration)
recently committed
Tx-ID
startTime
Actions (not stored)
Ta T1 Sally → Wen
Tb T2 Sally → Hen
active Transactions

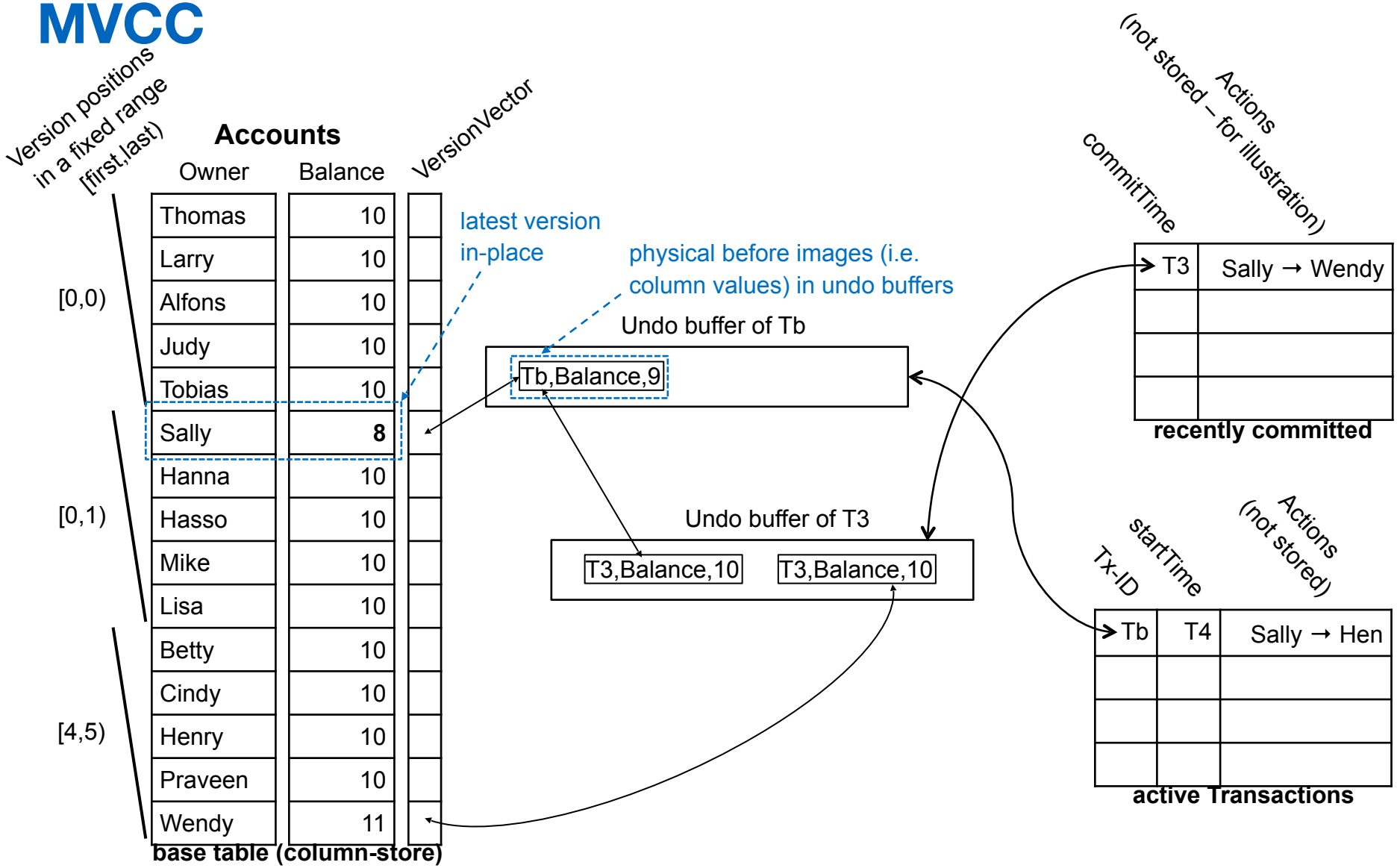
MVCC


Version positions in a fixed range [first,last)
Accounts
Owner
Balance
VersionVector
Thomas 10
Larry 10
Alfons 10
Judy 10
Tobias 10
Sally 9
Hanna 10
Hasso 10
Mike 10
Lisa 10
Betty 10
Cindy 10
Henry 10
Praveen 10
Wendy 11
[0,0)
[0,1)
[4,5)
base table (column-store)
latest version in-place
Restart the transaction
Undo buffer of Tb
Undo buffer of Ta
Ta,Balance,10
Ta,Balance,10
(not stored – for illustration)
Actions
commitTime
recently committed
Actions (not stored)
startTime
Tx-ID
Ta T1 Sally → Wen
Tb Sally → Hen
active Transactions

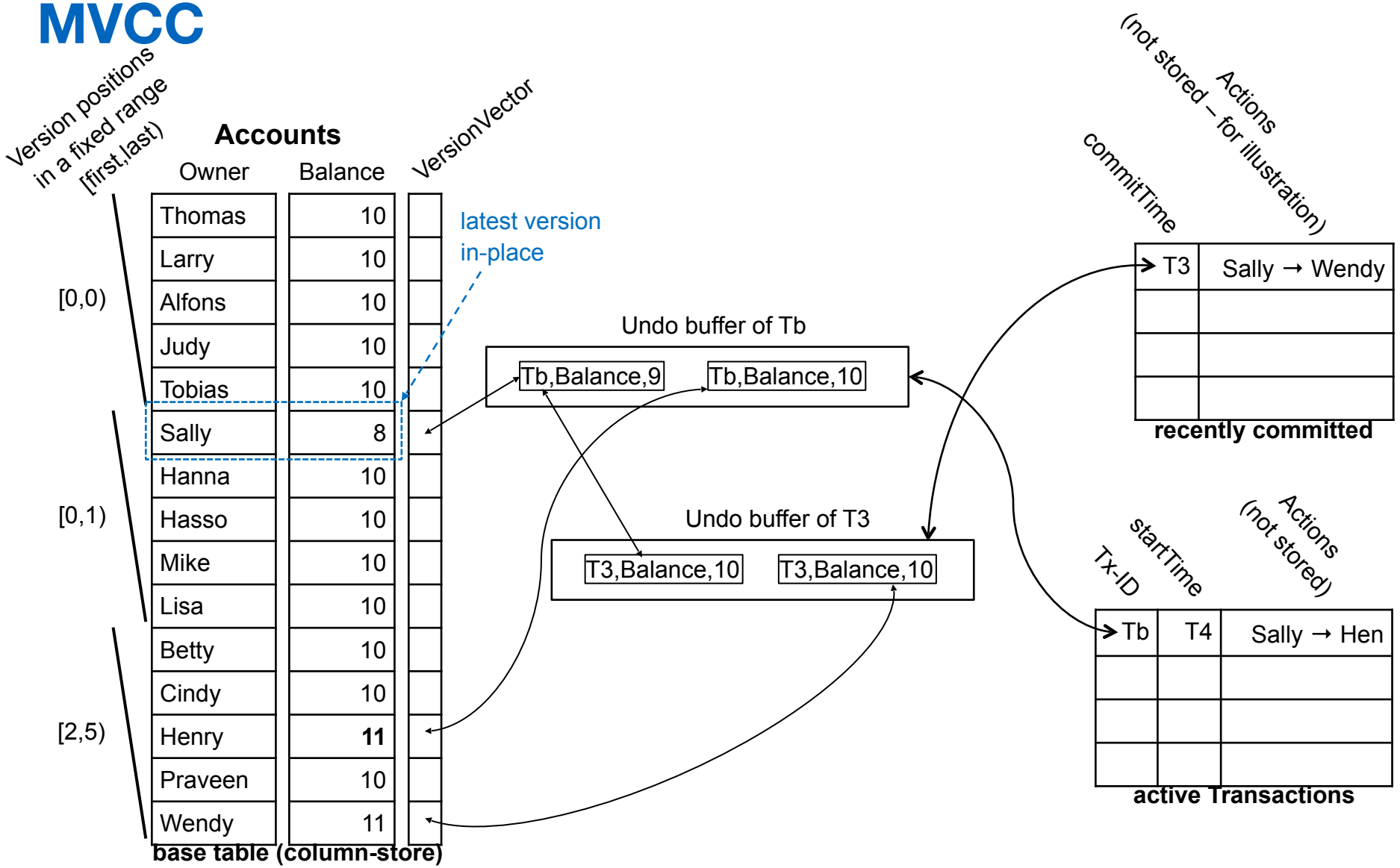
MVCC


Version positions in a fixed range [first,last)
Accounts
Owner
Balance
VersionVector
Thomas 10
Larry 10
Alfons 10
Judy 10
Tobias 10
Sally 9
Hanna 10
Hasso 10
Mike 10
Lisa 10
Betty 10
Cindy 10
Henry 10
Praveen 10
Wendy 11
[0,0)
[0,1)
[4,5)
base table (column-store)
latest version in-place
Undo buffer of Tb
Commit transaction
Undo buffer of T3
T3,Balance,10
T3,Balance,10
(not stored – for illustration)
Actions
commitTime
T3
Sally → Wendy
recently committed
Actions (not stored)
startTime
Tx-ID
Tb
Sally → Hen
Tb
Sally → Hen
active Transactions

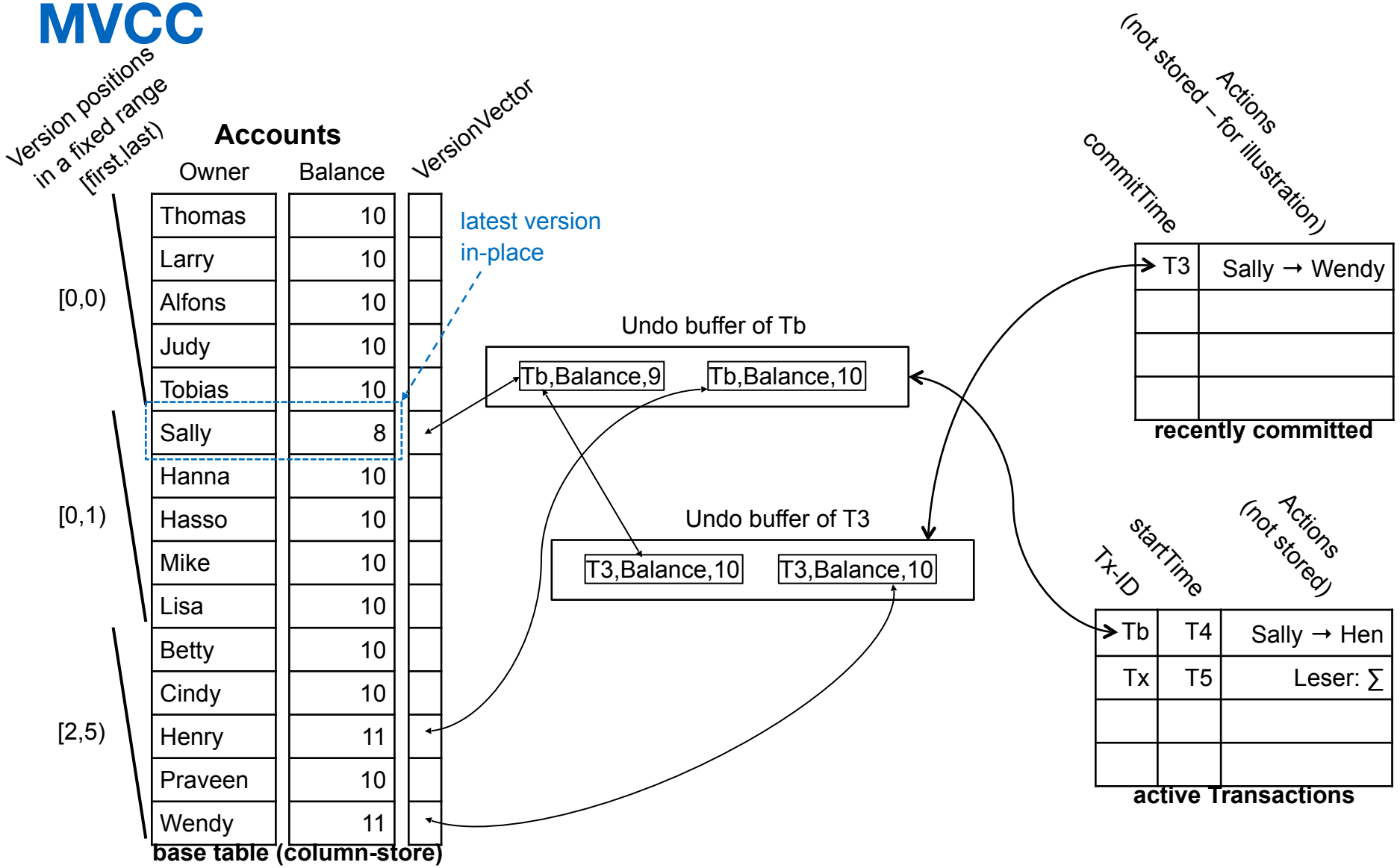
MVCC


Version positions in a fixed range [first,last)
Accounts
Owner
Balance
VersionVector
Thomas 10
Larry 10
Alfons 10
Judy 10
Tobias 10
Sally 8
Hanna 10
Hasso 10
Mike 10
Lisa 10
Betty 10
Cindy 10
Henry 10
Praveen 10
Wendy 11
[0,0)
[0,1)
[4,5)
base table (column-store)
latest version in-place
physical before images (i.e. column values) in undo buffers
Undo buffer of Tb
Tb,Balance,9
Undo buffer of T3
T3,Balance,10
T3,Balance,10
commitTime
Actions (not stored – for illustration)
T3
Sally → Wendy
recently committed
Tx-ID
startTime
Actions (not stored)
Tb
T4
Sally → Hen
active Transactions

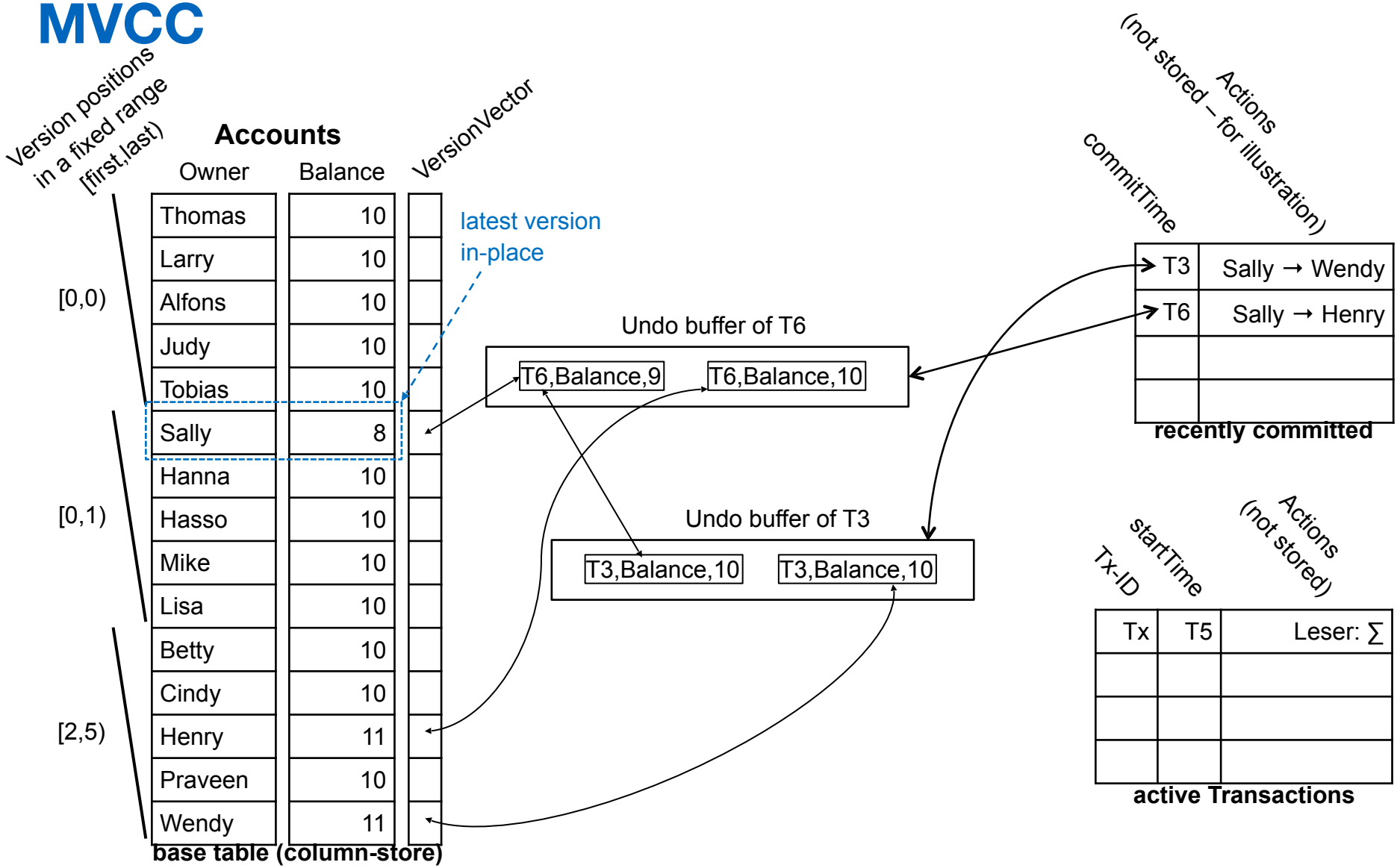
MVCC


Version positions in a fixed range [first,last)
Accounts
Owner
Balance
VersionVector
latest version in-place
[0,0)
[0,1)
[2,5)
Thomas 10
Larry 10
Alfons 10
Judy 10
Tobias 10
Sally 8
Hanna 10
Hasso 10
Mike 10
Lisa 10
Betty 10
Cindy 10
Henry 11
Praveen 10
Wendy 11
base table (column-store)
Undo buffer of Tb
Tb,Balance,9
Tb,Balance,10
Undo buffer of T3
T3,Balance,10
T3,Balance,10
commitTime
Actions (not stored – for illustration)
T3
Sally → Wendy
recently committed
Tx-ID
startTime
Actions (not stored)
Tb
T4
Sally → Hen
active Transactions

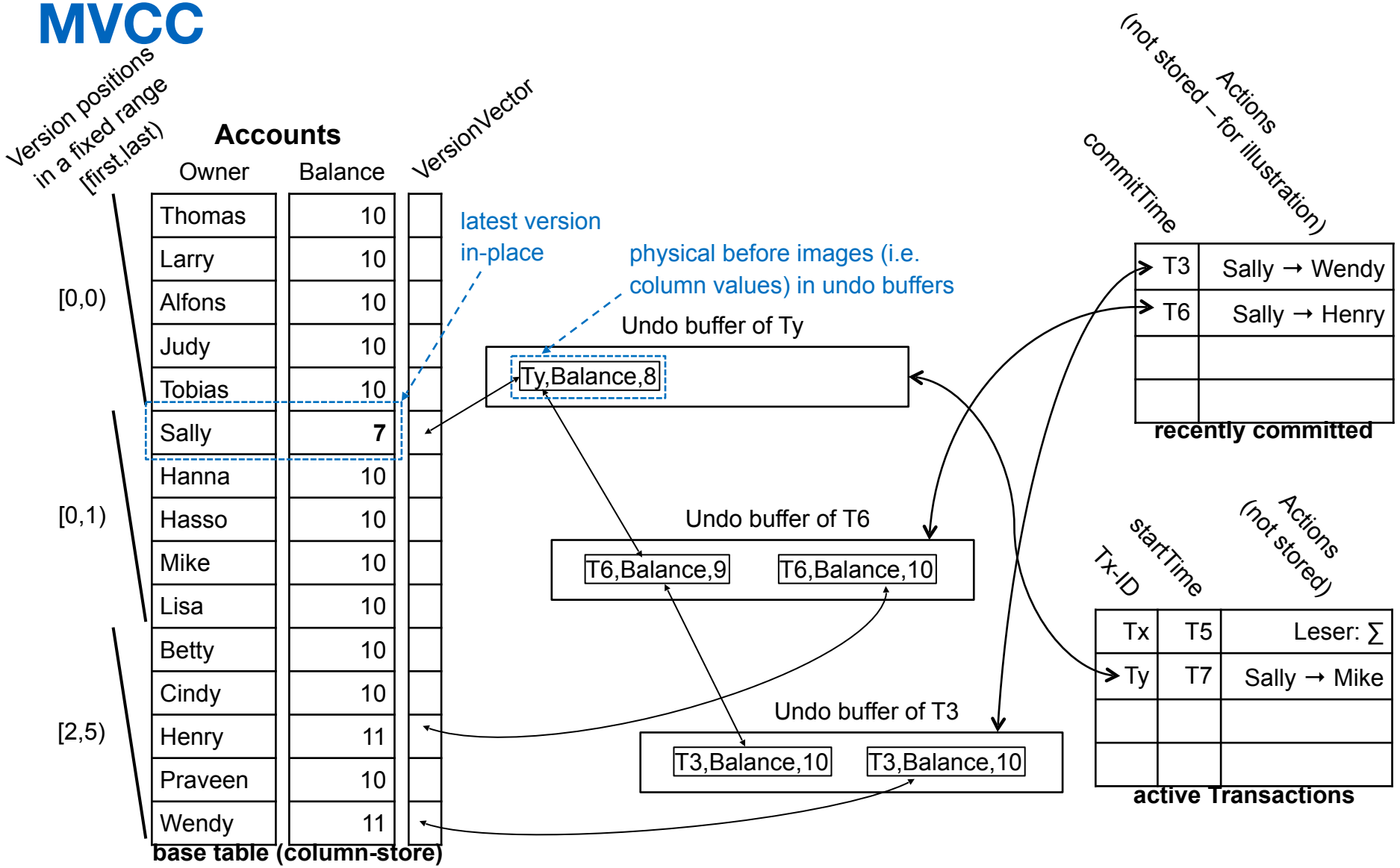
MVCC


Version positions in a fixed range [first,last)
Accounts
Owner
Balance
VersionVector
latest version in-place
[0,0)
[0,1)
[2,5)
Thomas 10
Larry 10
Alfons 10
Judy 10
Tobias 10
Sally 8
Hanna 10
Hasso 10
Mike 10
Lisa 10
Betty 10
Cindy 10
Henry 11
Praveen 10
Wendy 11
base table (column-store)
Undo buffer of Tb
Tb,Balance,9
Tb,Balance,10
Undo buffer of T3
T3,Balance,10
T3,Balance,10
commitTime
Actions (not stored – for illustration)
T3
Sally → Wendy
recently committed
Tx-ID
startTime
Actions (not stored)
Tb
T4
Sally → Hen
Tx
T5
Leser: ∑
active Transactions

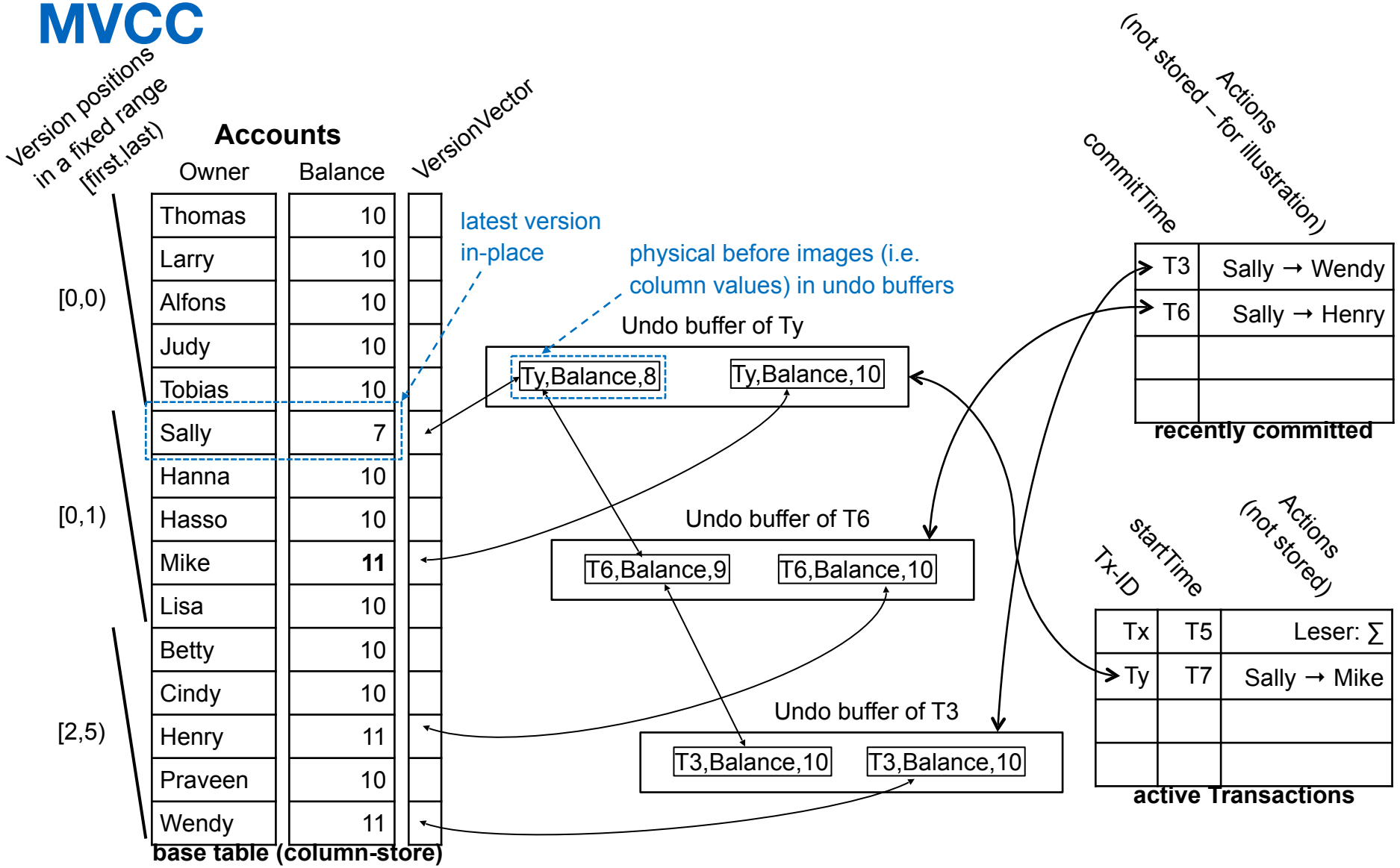
MVCC


Version positions in a fixed range [first,last)
Accounts
Owner
Balance
VersionVector
latest version in-place
[0,0)
[0,1)
[2,5)
Thomas 10
Larry 10
Alfons 10
Judy 10
Tobias 10
Sally 8
Hanna 10
Hasso 10
Mike 10
Lisa 10
Betty 10
Cindy 10
Henry 11
Praveen 10
Wendy 11
base table (column-store)
Undo buffer of T6
T6,Balance,9
T6,Balance,10
Undo buffer of T3
T3,Balance,10
T3,Balance,10
commitTime
Actions (not stored – for illustration)
T3 Sally → Wendy
T6 Sally → Henry
recently committed
Tx-ID
startTime
Actions (not stored)
Tx T5 Leser: ∑
active Transactions

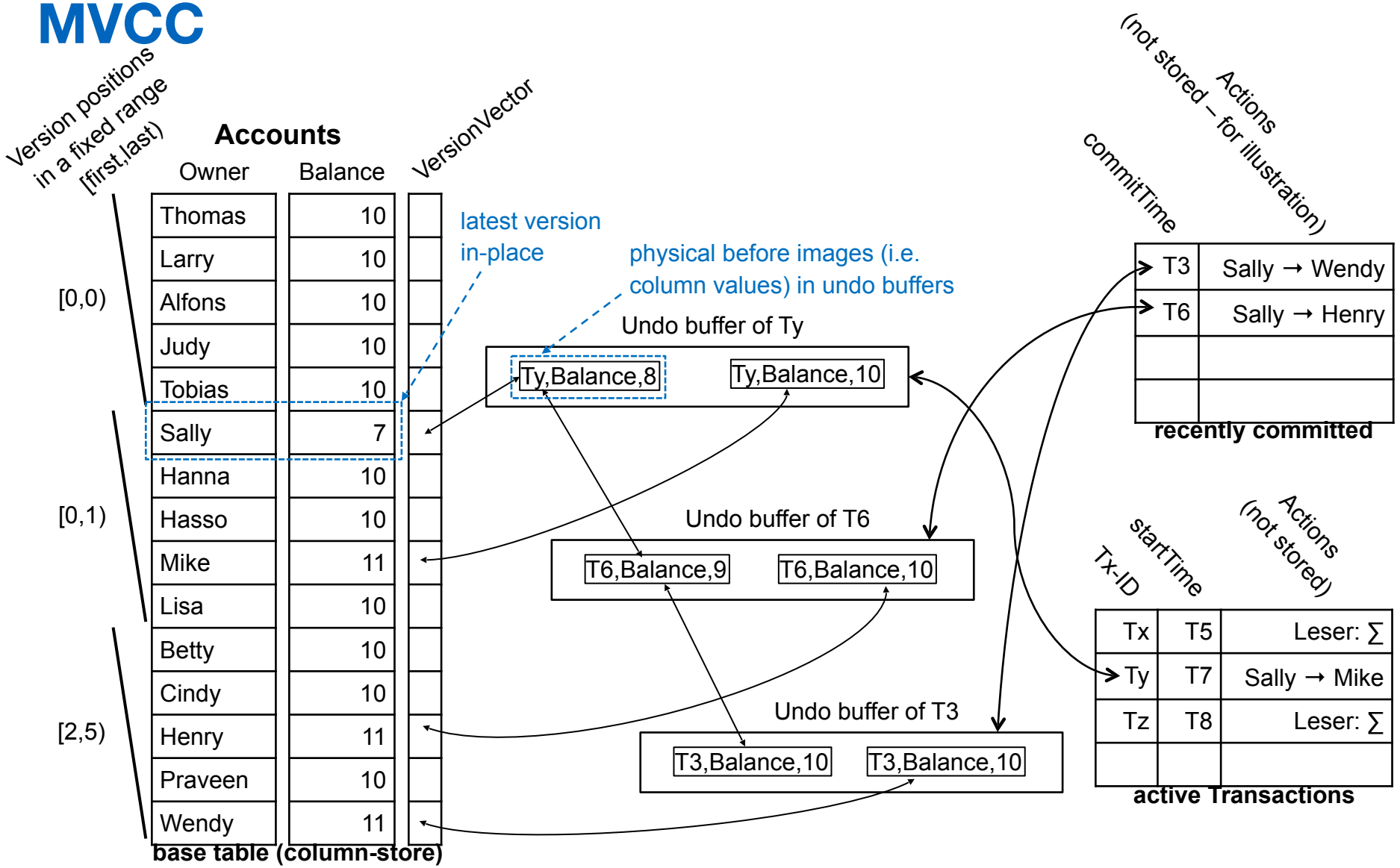
MVCC


Version positions in a fixed range [first,last)
Accounts
Owner
Balance
VersionVector
Thomas 10
Larry 10
Alfons 10
Judy 10
Tobias 10
Sally 7
Hanna 10
Hasso 10
Mike 10
Lisa 10
Betty 10
Cindy 10
Henry 11
Praveen 10
Wendy 11
[0,0)
[0,1)
[2,5)
base table (column-store)
latest version in-place
physical before images (i.e. column values) in undo buffers
Undo buffer of Ty
Ty,Balance,8
Undo buffer of T6
T6,Balance,9
T6,Balance,10
Undo buffer of T3
T3,Balance,10
T3,Balance,10
commitTime
Actions (not stored – for illustration)
T3 Sally → Wendy
T6 Sally → Henry
recently committed
Tx-ID
startTime
Actions (not stored)
Tx T5 Leser: ∑
Ty T7 Sally → Mike
active Transactions

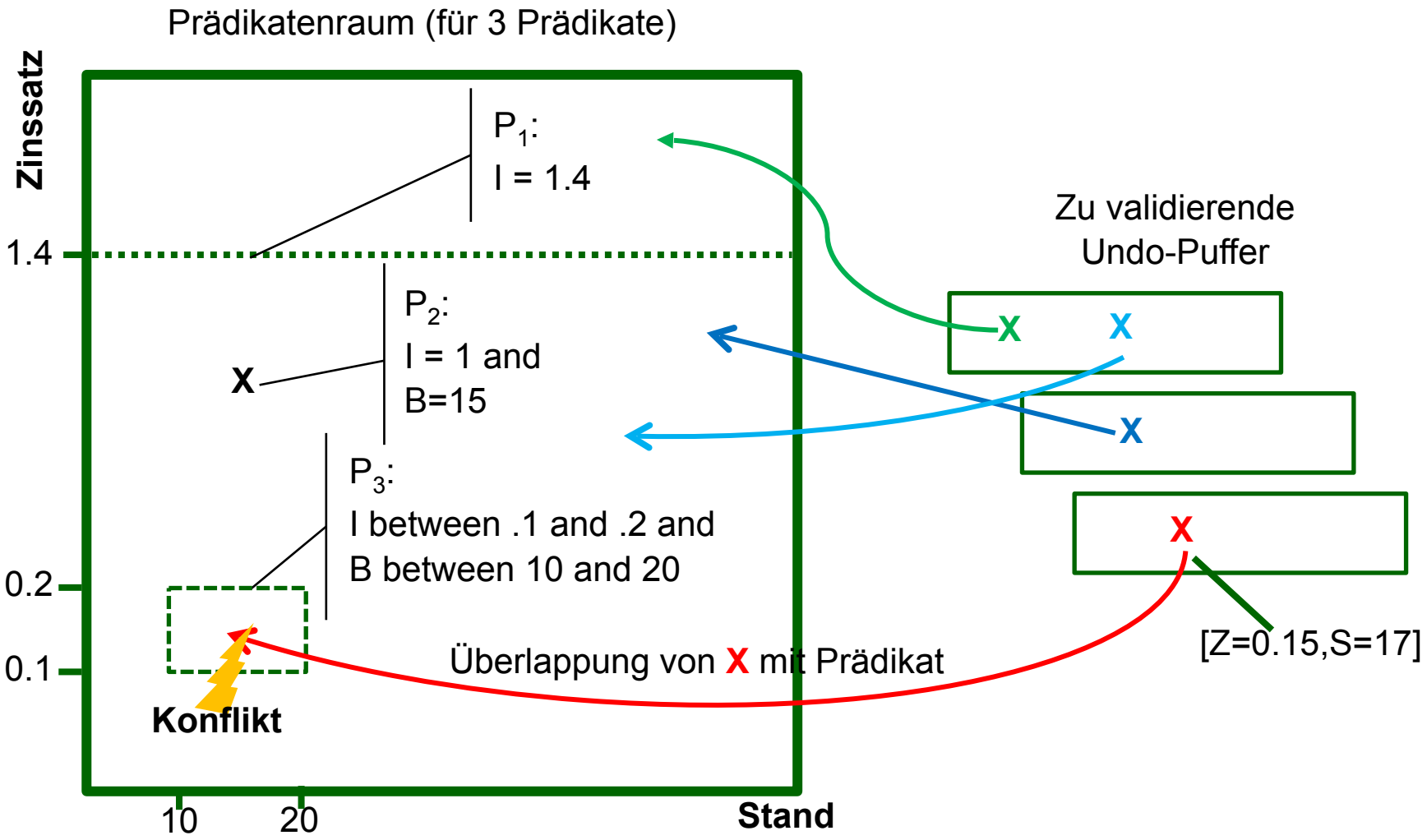
MVCC


Version positions in a fixed range [first,last)
Accounts
Owner
Balance
VersionVector
[0,0)
[0,1)
[2,5)
Thomas 10
Larry 10
Alfons 10
Judy 10
Tobias 10
Sally 7
Hanna 10
Hasso 10
Mike 11
Lisa 10
Betty 10
Cindy 10
Henry 11
Praveen 10
Wendy 11
base table (column-store)
latest version in-place
physical before images (i.e. column values) in undo buffers
Undo buffer of Ty
Ty,Balance,8
Ty,Balance,10
Undo buffer of T6
T6,Balance,9
T6,Balance,10
Undo buffer of T3
T3,Balance,10
T3,Balance,10
commitTime
Actions (not stored – for illustration)
T3 Sally → Wendy
T6 Sally → Henry
recently committed
Tx-ID
startTime
Actions (not stored)
Tx T5 Leser: ∑
Ty T7 Sally → Mike
active Transactions

MVCC


Version positions in a fixed range [first,last)
Accounts
Owner
Balance
VersionVector
Thomas 10
Larry 10
Alfons 10
Judy 10
Tobias 10
Sally 7
Hanna 10
Hasso 10
Mike 11
Lisa 10
Betty 10
Cindy 10
Henry 11
Praveen 10
Wendy 11
[0,0)
[0,1)
[2,5)
base table (column-store)
latest version in-place
physical before images (i.e. column values) in undo buffers
Undo buffer of Ty
Ty,Balance,8
Ty,Balance,10
Undo buffer of T6
T6,Balance,9
T6,Balance,10
Undo buffer of T3
T3,Balance,10
T3,Balance,10
commitTime
Actions (not stored – for illustration)
T3 Sally → Wendy
T6 Sally → Henry
recently committed
Tx-ID
startTime
Actions (not stored)
Tx T5 Leser: ∑
Ty T7 Sally → Mike
Tz T8 Leser: ∑
active Transactions

Precision Locking



MVCC mit Precision Locking

– Lesende Anfragen sind **immer erlaubt**: kein Precision Locking

– Falls schreibende Anfrage: **Überlappender Prädikatbereich**?

– Falls ja, dann **BOT** und **commit-Reihenfolge** beachten

Gruppenaufgabe 6

Name	verfügbar	Versionsvektor
House	ja	-
Green	ja	-
Brinkmann	ja	-

TID	Startzeit	Commitzeit	Aktion
Ta	$T0$	-	$\sum$
Tb	$T2$	-	$(Green)--$
Tc	$T3$	-	$\sum$
Td	$T5$	-	$\sum$

Beschäftigen wir uns mit *Multi-Version Concurrency Control* am Beispiel unserer verfügbaren Ärzte („Doctors on call/duty“), in dem wir sicherstellen wollen, dass immer mindestens ein Arzt verfügbar ist.

Uns stehen drei Operationen zur Verfügung, $\sum$ zählt alle verfügbaren Ärzte, $(X)++$ ändert Xs Status in verfügbar, $(X)--$ zählt alle verfügbaren Ärzte und ändert Xs Status auf nicht verfügbar, wenn mindestens ein Arzt noch anwesend ist.

1. Welche Bedingungen gelten für die Zeitstempel?
2. Green möchte zum Zeitpunkt $T2$ seinen Feierabend antreten. Vervollständigen Sie Tabelle 2 und legen Sie einen geeigneten Undo-Puffer (Zeitstempel, Attribut, Undo-Image) an. Wann muss Tb committen, damit Ta und Td erfolgreich committen? Was lesen Ta und Tb?
3. Brinkmann und House wollen zeitgleich den Feierabend antreten. House startet bei $T8$, Brinkmann bei $T9$. Wer darf gehen? Wie sorgt *Precision Locking* dafür, dass nur ein Arzt das Krankenhaus verlässt? Vervollständigen Sie die Einträge.

Aufgabe 7

T1: insert into foo (select Note from Noten where MatrNr=12345)

T2: insert into bar (select count(*) from Noten where Note<1.5)

T3: insert into Noten(MatrNr,Note) values (54321, 3.0)

T4: update Noten set Note=1.4 where MatrNr=32154

T5: insert into Noten(MatrNr,Note) values (54321, 1.3)

T6: update Noten set Note=1.6 where MatrNr=12345

Analysieren Sie, ob die folgenden Historien unter dem MVCC Model, wie in der Vorlesung vorgestellt, auftreten können. Jede Historie steht für sich selbst und startet jeweils von einem ursprünglichen Datenzustand. Die Buchstaben innerhalb der Klammer entsprechen dabei jeweils den Tupeln auf die zugegriffen wird. Wenn in T2 z.B. drei Werte das 'Prädikat Note<1.5' erfüllen, gäbe es entsprechend drei $r(...)$ Einträge auf die jeweiligen Tupel.

H1 (T1 und T3): $bot_1, r_1(A), bot_3, w_3(B), w_1(C), commit_1, commit_3$

H2 (T2 und T3): $bot_2, r_2(A), bot_3, w_3(B), r_2(C), w_2(D), commit_2, commit_3$

H8 (T2 und T3): $bot_2, r_2(A), bot_3, w_3(B), r_2(C), commit_3, w_2(D), commit_2$

H3 (T2 und T4): $bot_2, r_2(A), r_2(B), bot_4, r_4(B), w_4(B), r_2(C), w_2(D), commit_2, commit_4$

H5 (T2 und T4): $bot_2, r_2(A), bot_4, r_4(B), w_4(B), r_2(C), commit_4, w_2(D), commit_2$

H4 (T1 und T6): $bot_1, r_1(B), bot_6, r_6(B), w_6(B), w_1(C), commit_1, commit_6$

H6 (T1 und T6): $bot_1, r_1(B), bot_6, r_6(B), w_6(B), commit_6, w_1(C), commit_1$

H7 (T2 und T5): $bot_2, r_2(A), bot_5, w_5(D), commit_5, r_2(D), w_2(E), commit_2$

H9 (T2 und T5): $bot_2, r_2(A), bot_5, w_5(B), r_2(C), commit_5, w_2(D), commit_2$

Fragen?