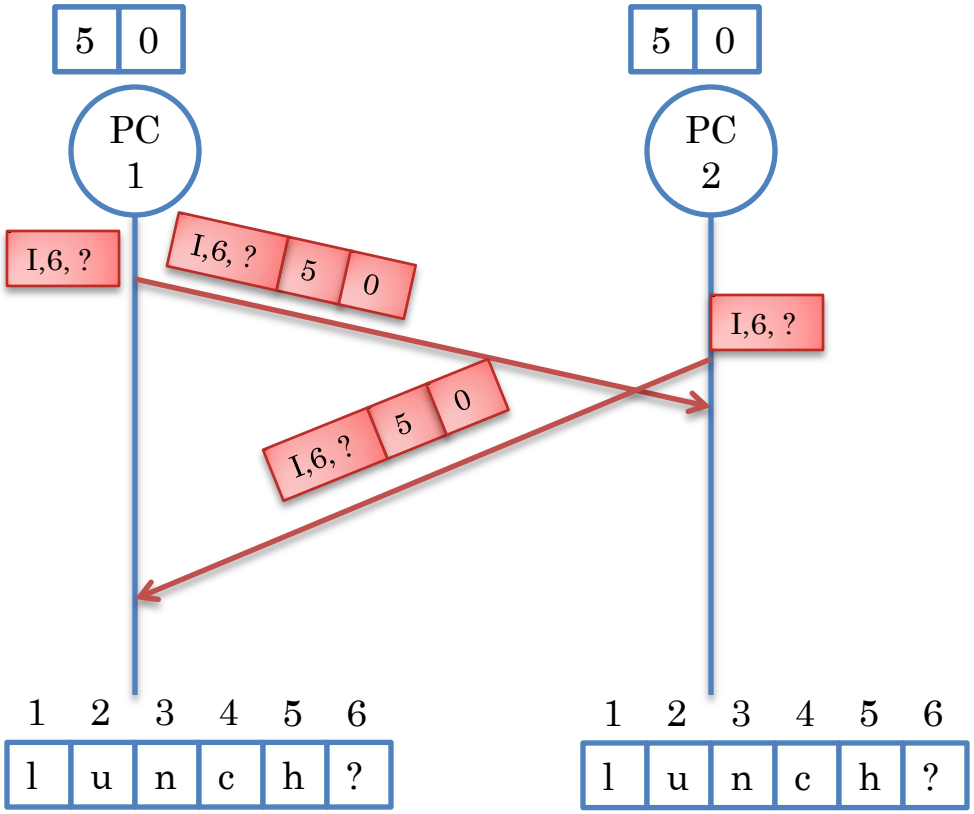
Merge Policies

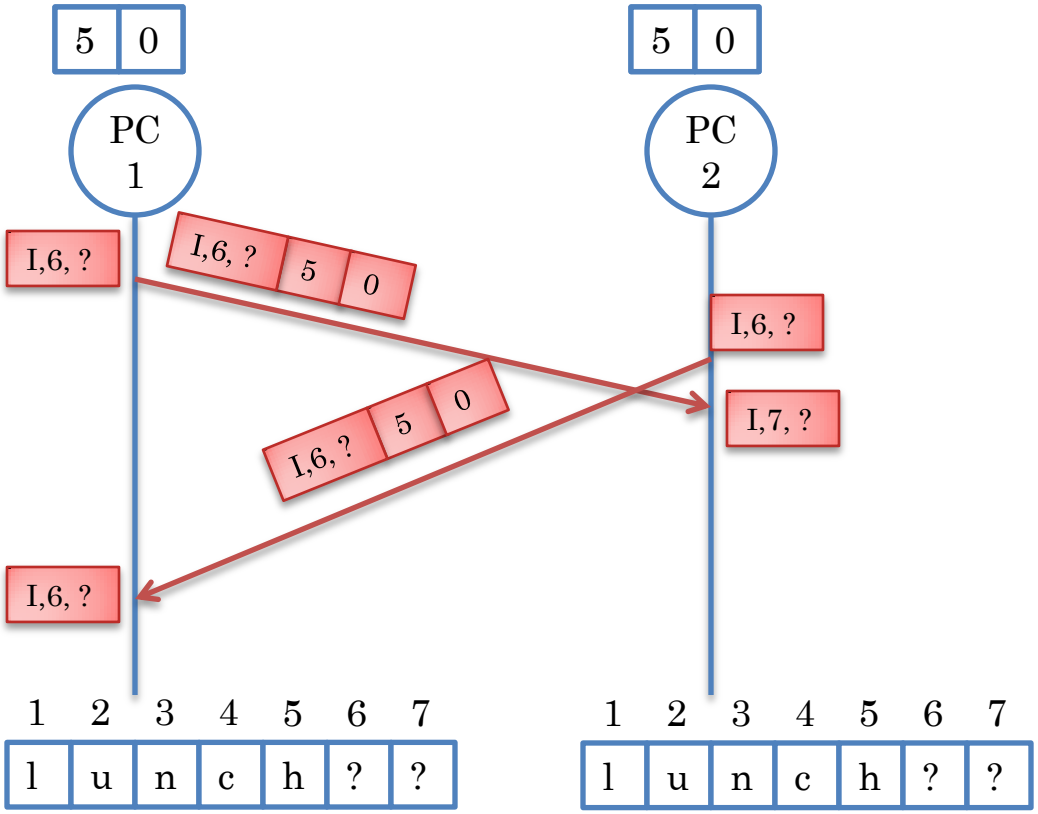
Prasun Dewan
Department of Computer Science
University of North Carolina at Chapel Hill
dewan@cs.unc.edu

Concurrent Interaction



Insert same character at same position?

Concurrent Interaction



Insert same character at same position?

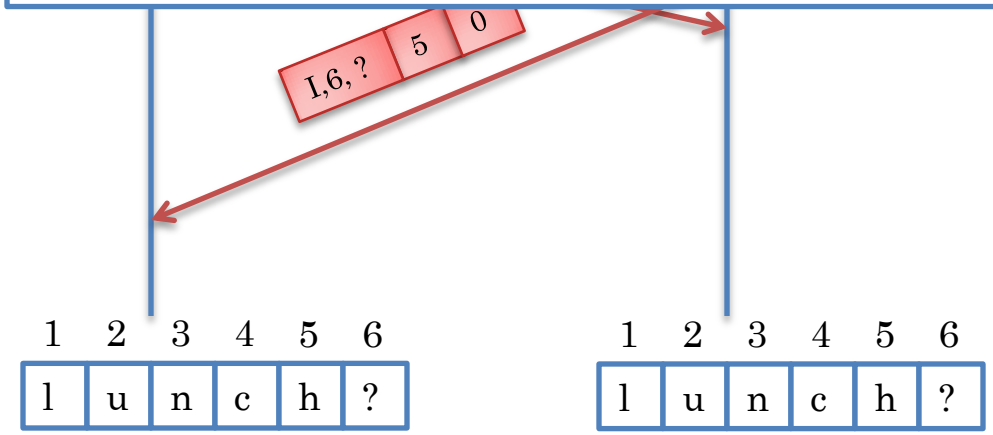
Duplicate character!

Work preserving

Does not meet "user intention"

Concurrent Interaction

```
InsertOperation TransformInsertInsert (InsertOperation Remote, InsertOperation L ocal)  {
        Operation Remote^T = Remote.clone();
        if (Remote.index  == Local.index  && Remote.element.equal(Local.element))
                return NullOperation
        if (Remote.index > Local.index  | |
           (Remote.index === Local.index  &&  Remote.id  < Local.id))
                Remote^T.index = Remote.index + 1;
        return Remote^T ;
}
```



Meets “user intention”

Goal of Consistency

- At quiescence (no user interacting) all displays are the same
  - When concurrent command is executed, could ignore it and clear object
  - Meets TP1 (and TP2!)
- Meets user intention
  - How to describe what it is?
  - Even if we cannot describe it, how to describe what algorithm does
    - More than one "reasonable" merge acceptable
  - Application-specific merger
  - Application-specific merge procedure
  - Declarative scheme?

Assumptions and Intention Issue

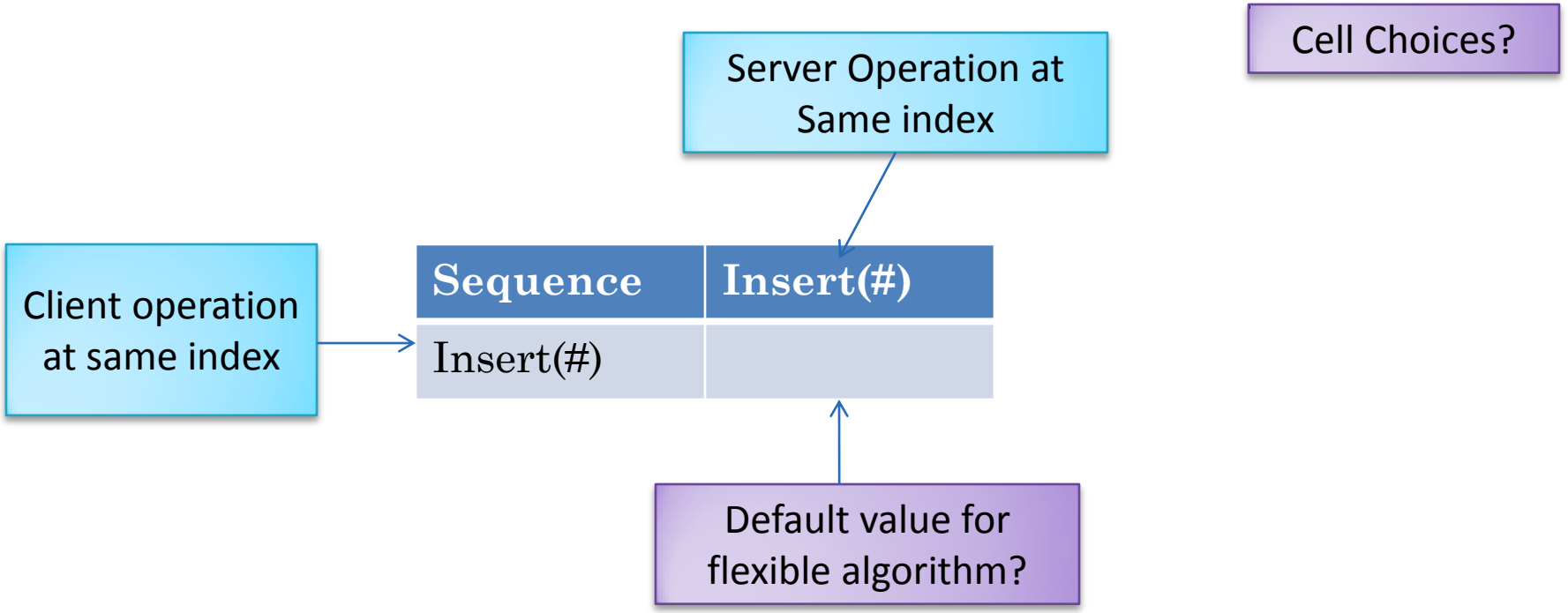
One-level sequence with inserts only.

Inserts at different positions are both accepted (as define by initial transformation functions).

If both users insert the same element at the same position, only one is executed.

What happens when both users insert different elements at the same position?

Merge Matrix For Insertable Sequences

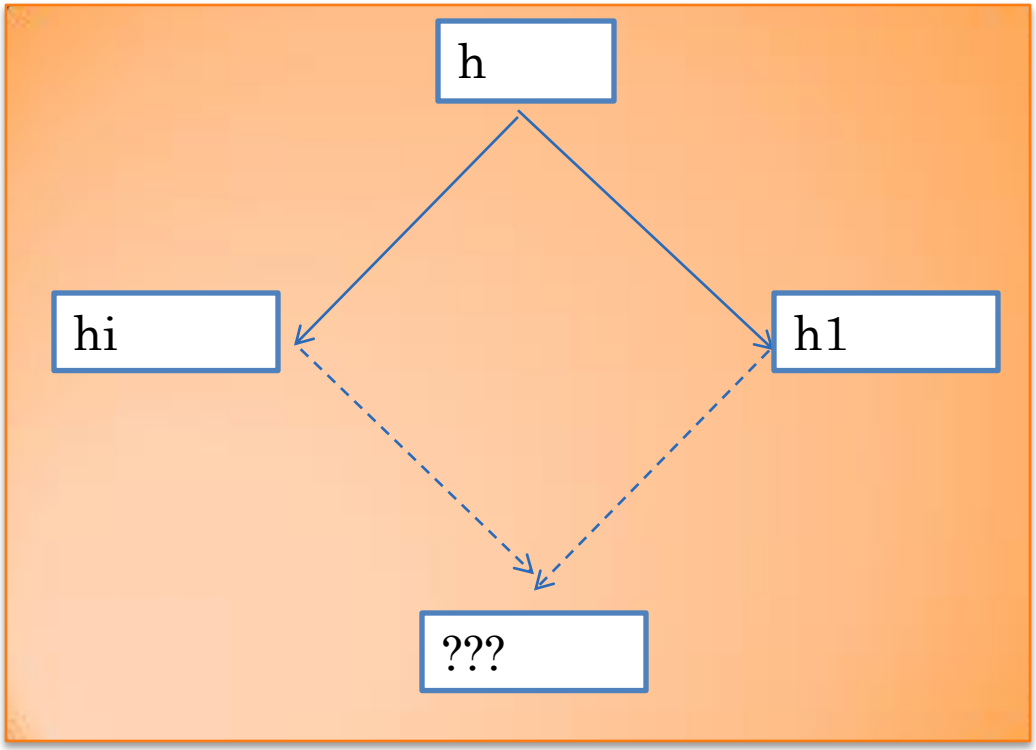


Describes merge semantics of different insert operations at same index

Merge matrix not relevant when indices are different or insertions are the same

A particular algorithm can give the application and/or user a subset of choices in the merge matrix

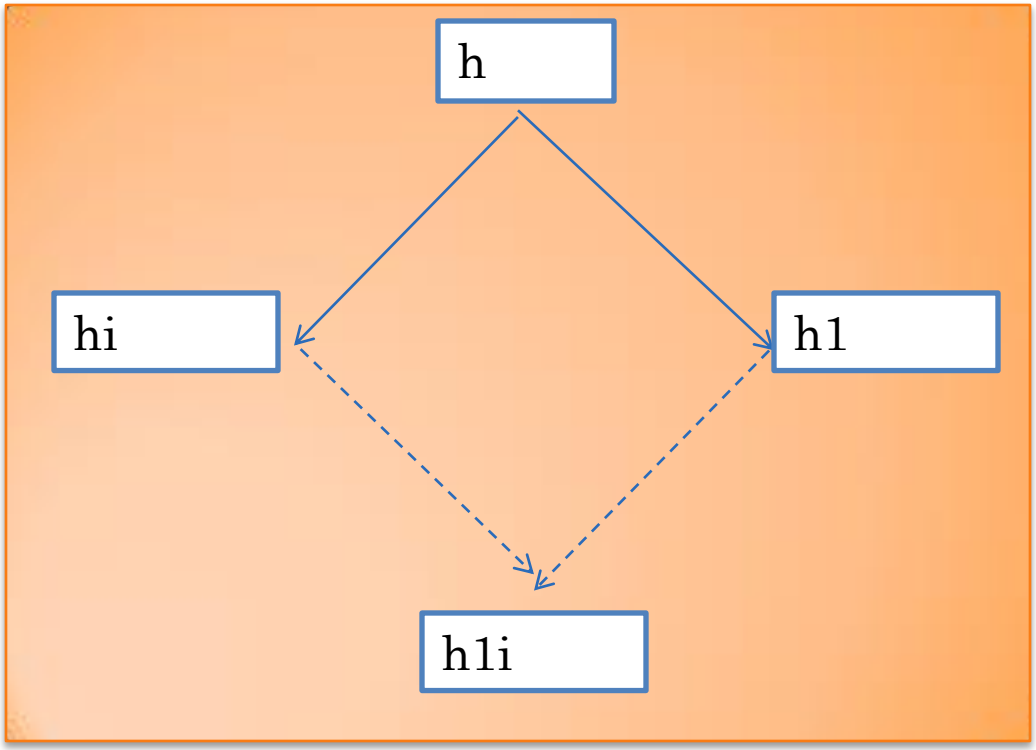
Concurrent Operation



User intention?

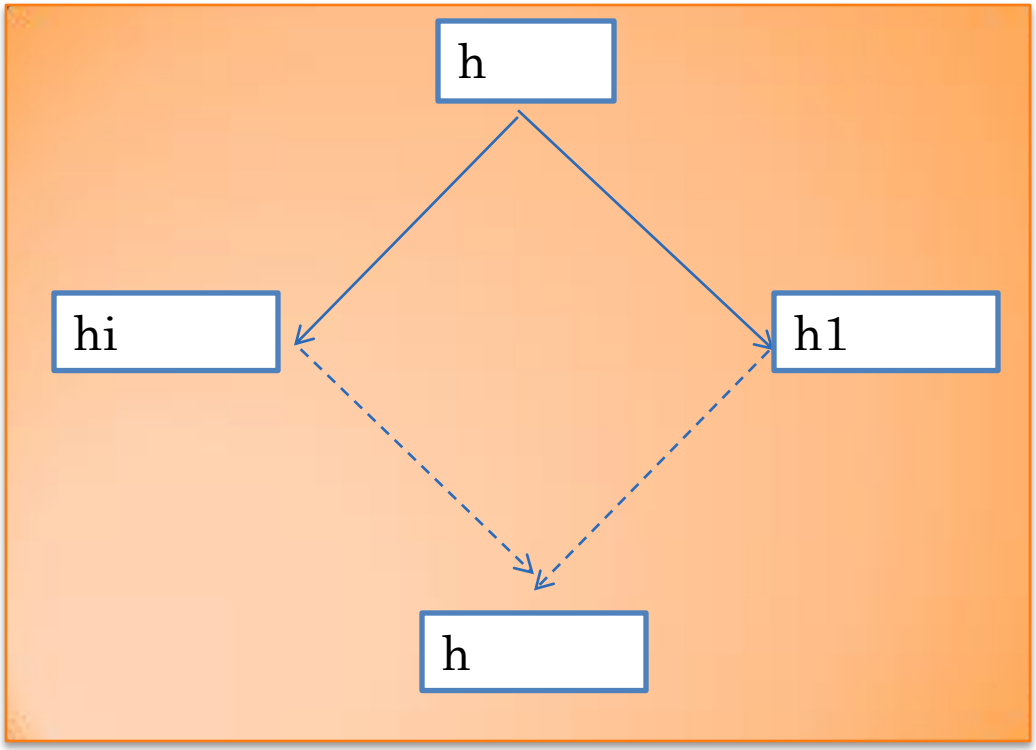
Conflict!

Accept Both



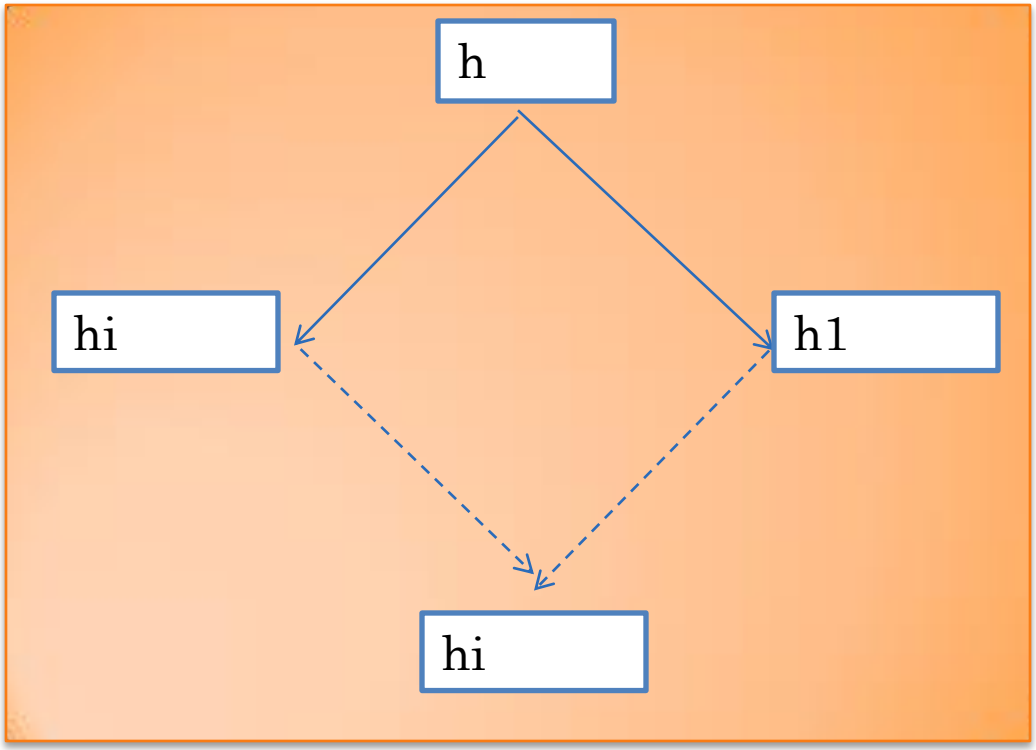
Accept both as they can resolve the conflict

Accept None



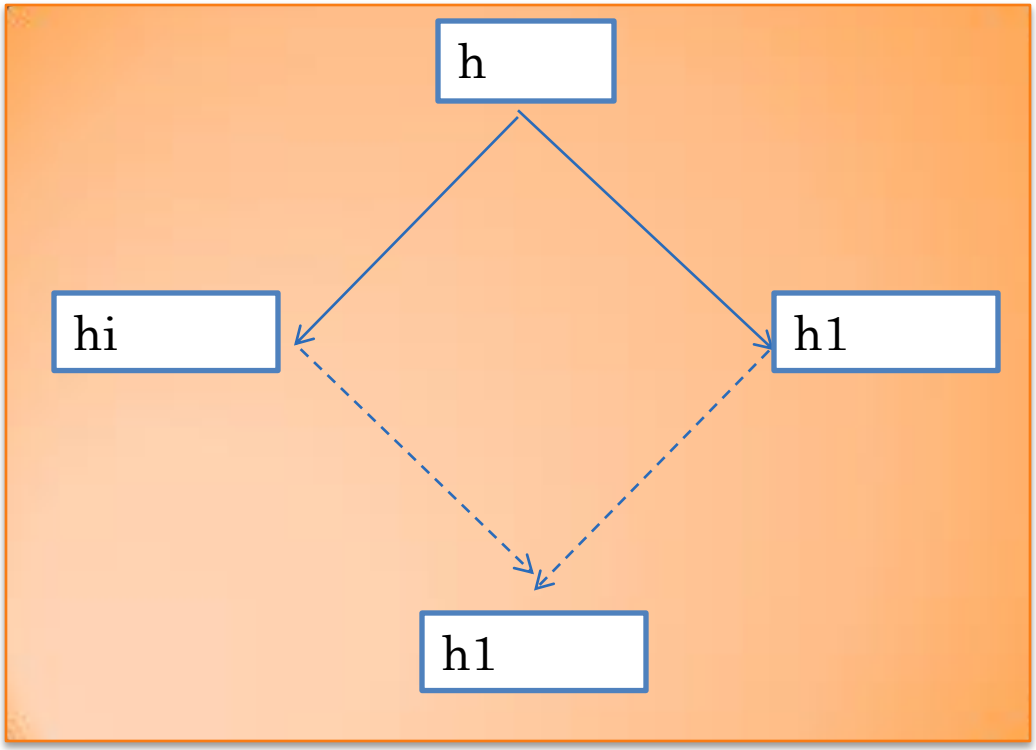
Accept none as there is a conflict and cannot afford a wrong merge

Accept Server



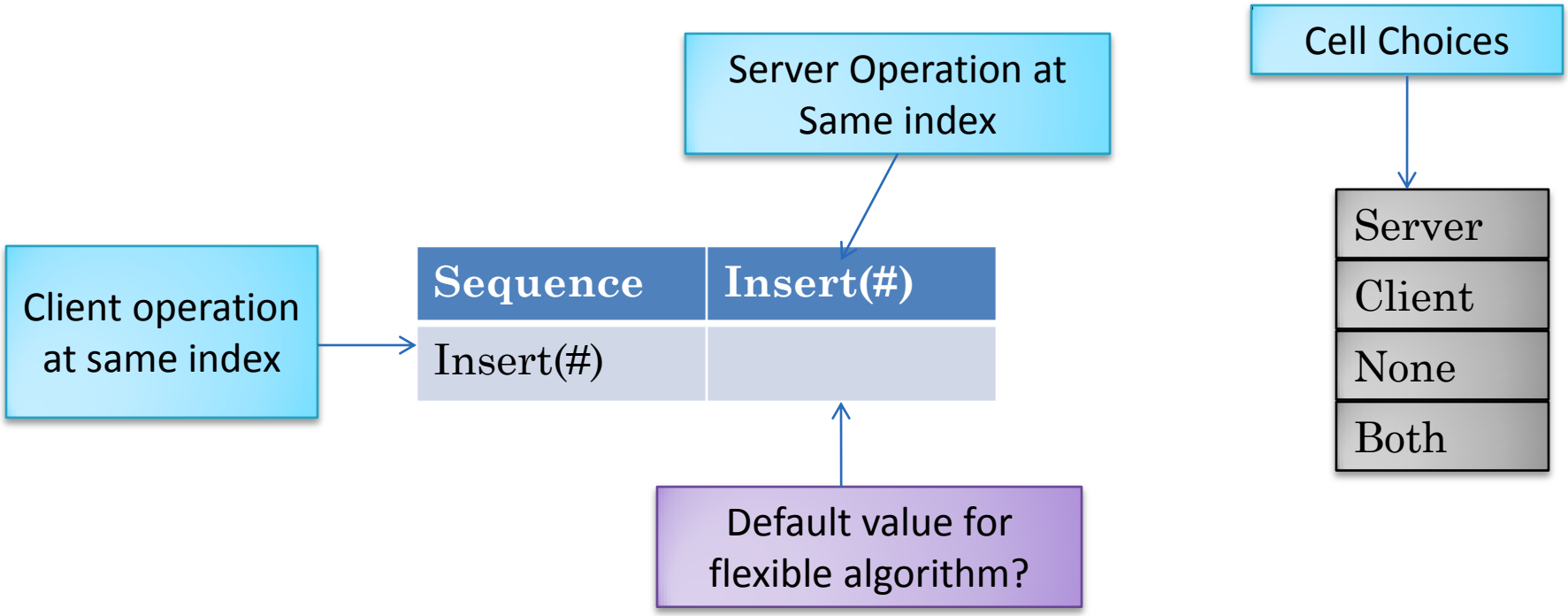
Accept earlier (server) so later person can see it and correct it

Accept Client



Accept later (client) as more recent information available

Merge Matrix For Insertable Sequences



Describes merge semantics of different insert operations at same index

Merge matrix not relevant when indices are different or insertions are the same

A particular algorithm can give the application and/or user a subset of choices in the merge matrix

Default for Insertable Sequences

Sequence	Insert(#)
Insert(#)	Both

Default value for flexible algorithm

- Server
- Client
- None
- Both

Delete and Modify Operations?

Merge Matrix for Insertable Sequences

Sequence	Insert(#)	Delete(#)	Replace(#)
Insert(#)			
Delete(#)			
Replace(#)			

Server
Client
None
Both

Describes merge semantics of different sequence operations at same index

Default values for flexible algorithm?

Defaults for General Sequences

Sequence	Insert(#, a)	Delete(#)	Replace(#)
Insert(#, b)	Both	Both	Both
Delete(#)	Both	NoOp	Server
Replace(#)	Both	Client	Server

Server
Client
None
Both

Tables?

Deletes at the same index always the same operation and hence NoOp

Replacement means it is relevant and perhaps should not be deleted

General Sequences and Replace/Delete

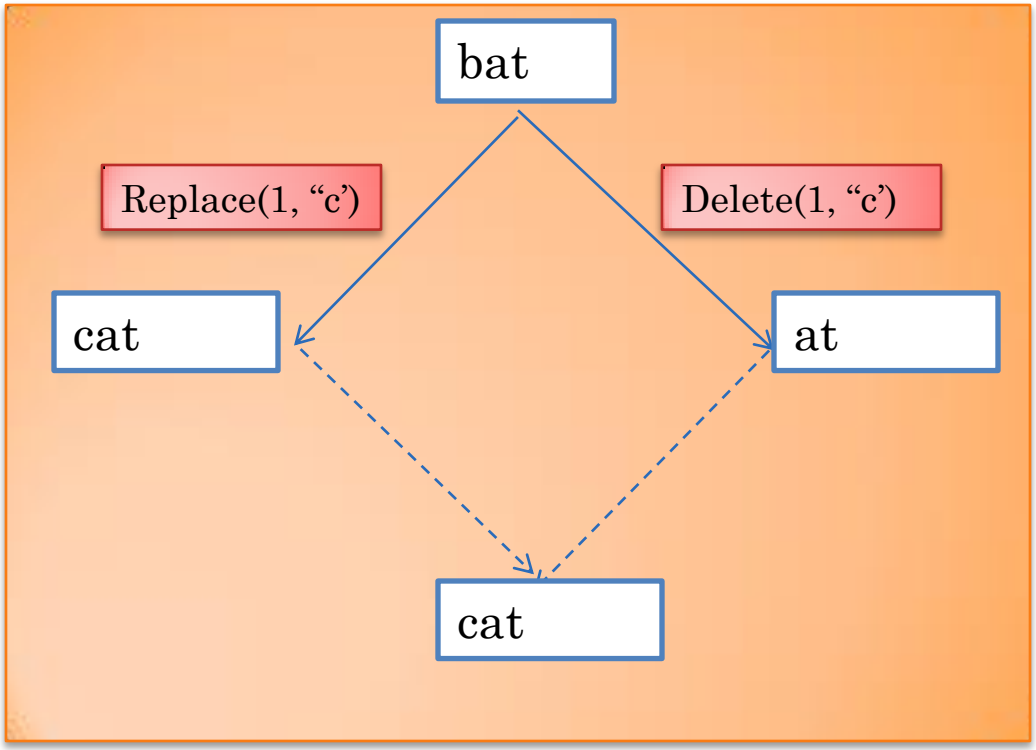


Table Matrix

Table	Put (key)	Delete(key)
Put (key)		
Delete(key)		

Server
Client
None
Both

Assume operations at different keys are non conflicting

Describes merge semantics of different table operations at key

Default values?

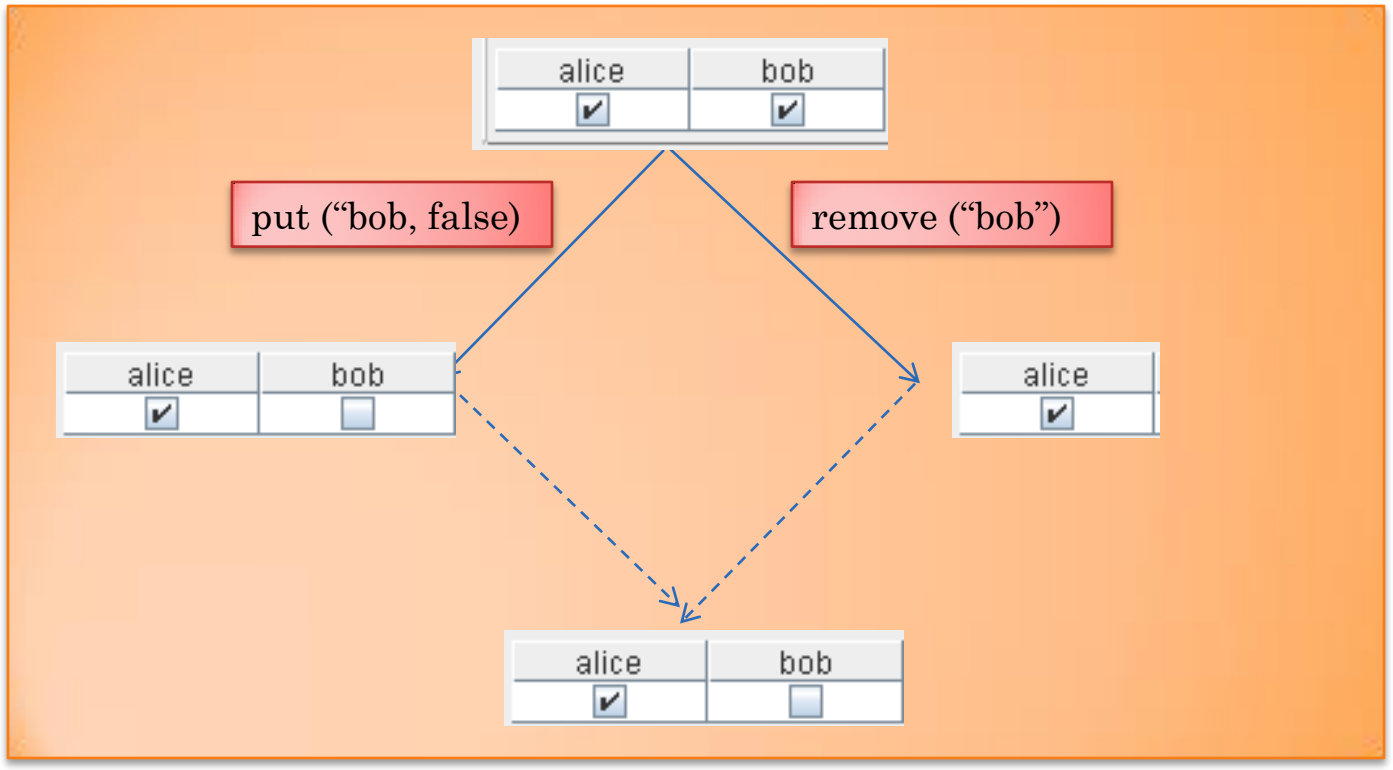
Defaults for General Tables

Table	Put (key)	Delete(key)
Put (key)	Server	Client
Delete(key)	Server	NoOp

- Server
- Client
- None
- Both

Putting the same value at the same key is a NoOP

Tables



Record Matrix

Record	Set (Property)
Set (Property)	

Server
Client
None
Both

Describes merge semantics of different record operations at same property

Putting the same value at the same property is a NoOP

Default value?

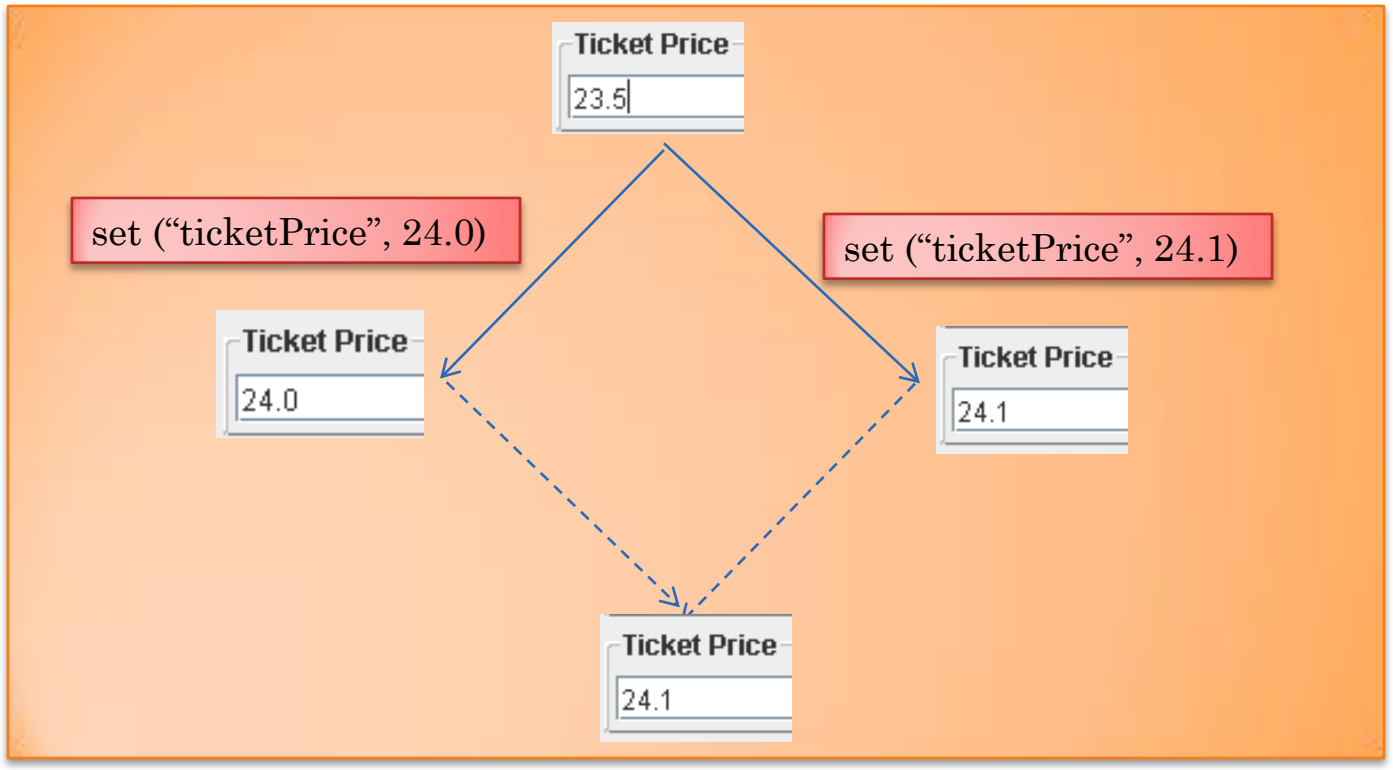
Defaults for Record

Record	Set (Property)
Set (Property)	Server

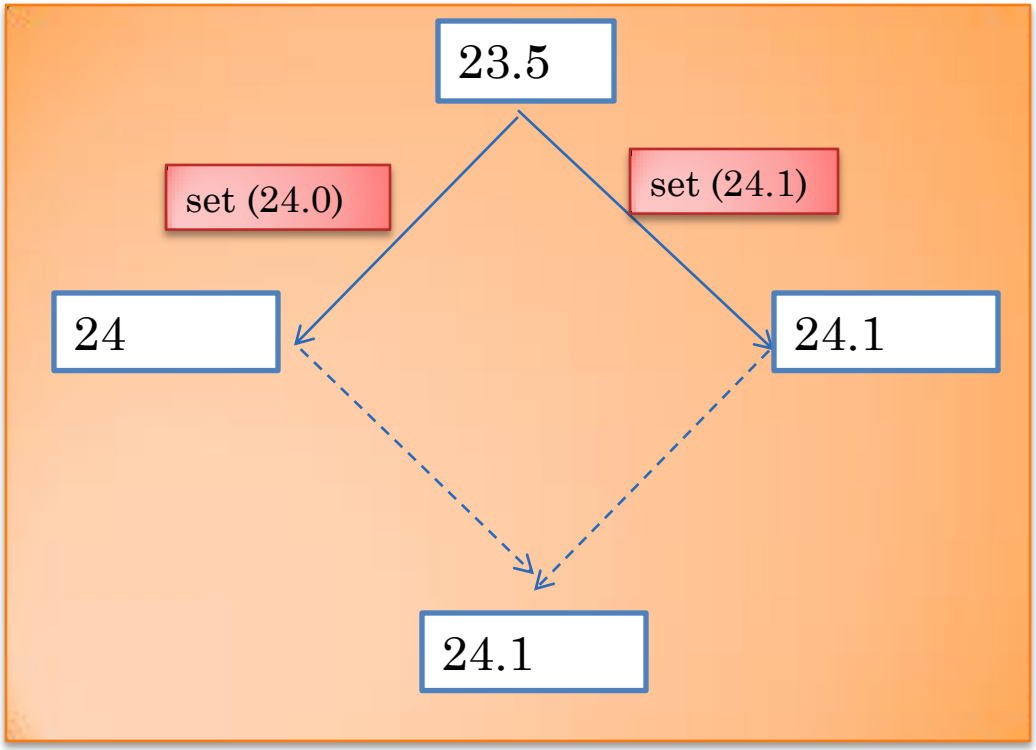
- Server
- Client
- None
- Both

Accept earlier (server) so later person can see it and correct it

Beans/Records



Atomic Objects



Accept earlier (server) so later person can see it and correct it

Atomic Matrix

Atomic	Set ()
Set ()	Server

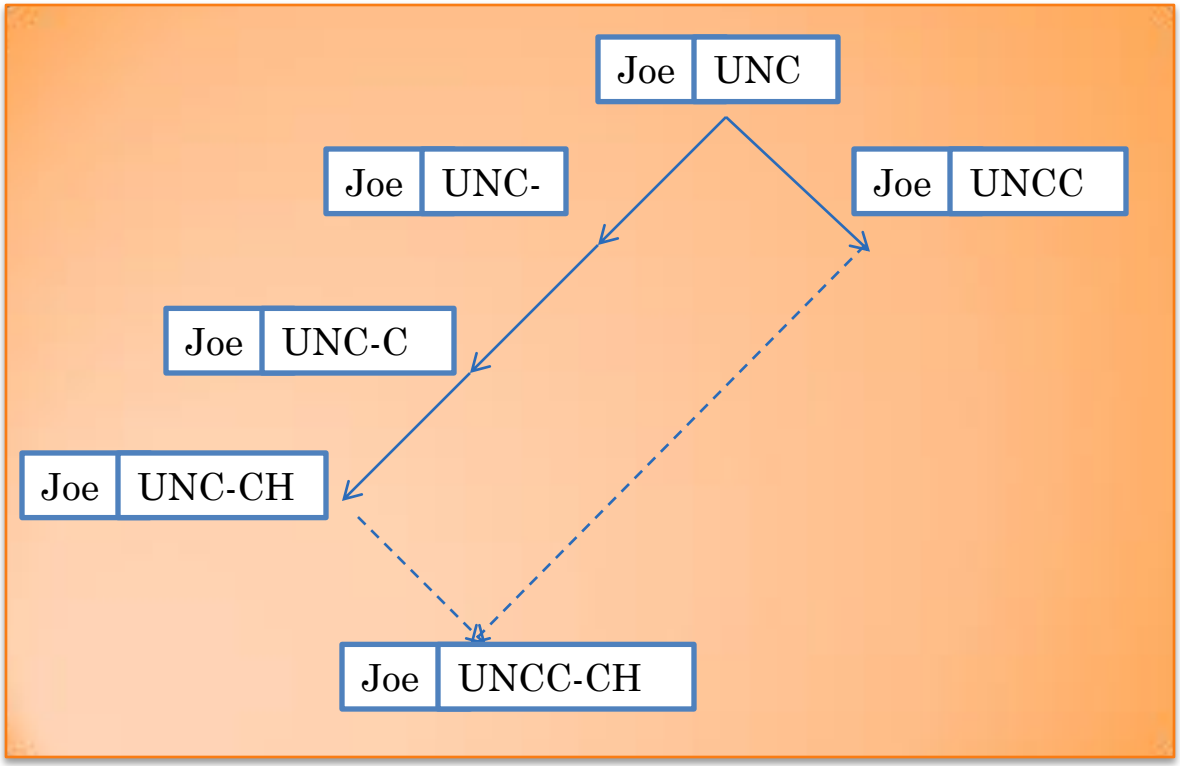
General Merge Matrix

Type	$Operation_1$	…	$Operation_N$
$Operation_1$	$Default_{11}$	…	$Default_{1N}$
…	…	…	…
$Operation_N$	$Default_{N1}$	…	$Default_{NN}$

Server
Client
None
Both

Some type-specific operand (index, key, value) whose value determines when two dissimilar operations are compared

Asynchronous Buffered Changes

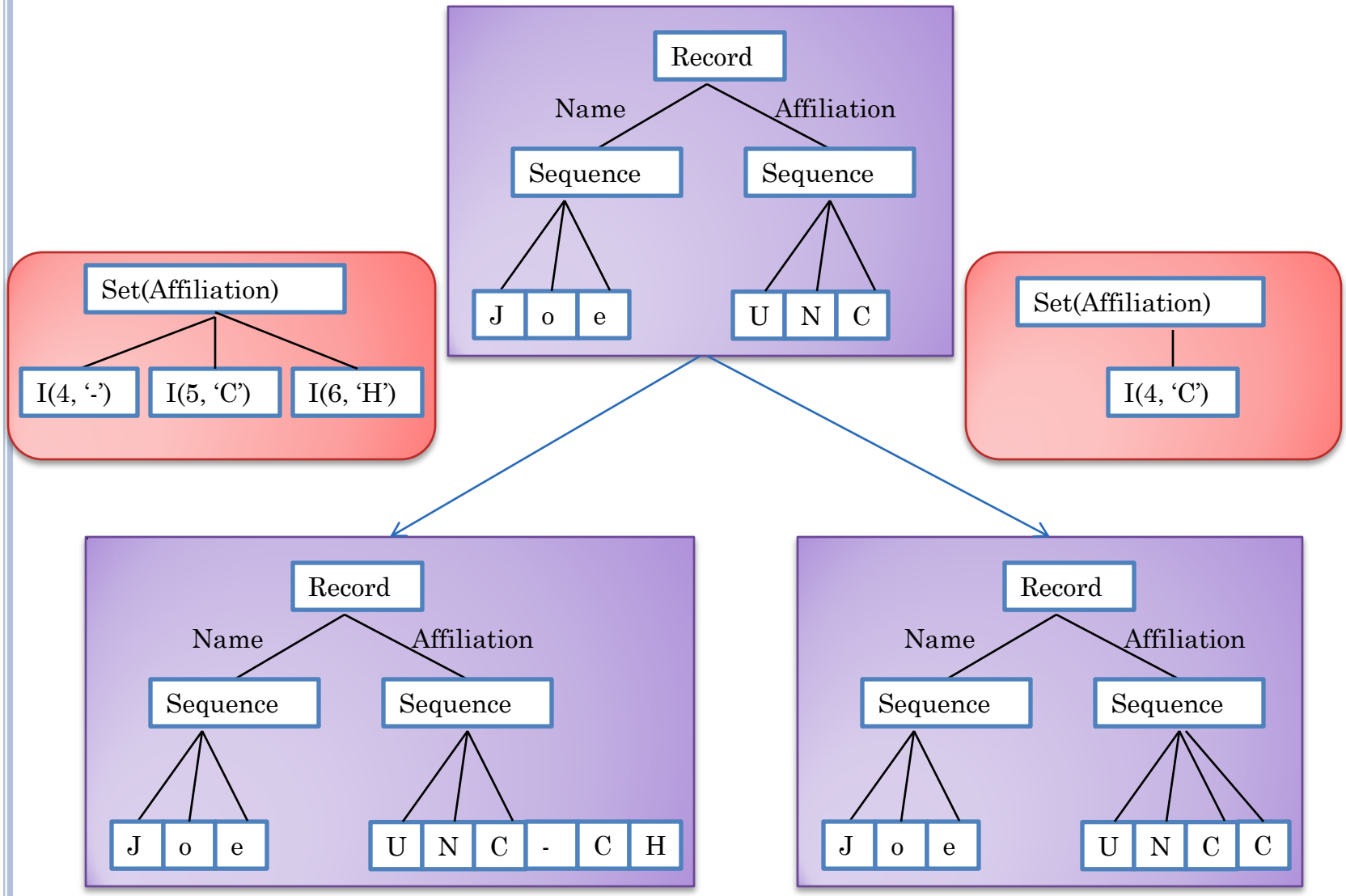


"Joe" and "UNC" are sequences in a record with name and affiliation properties

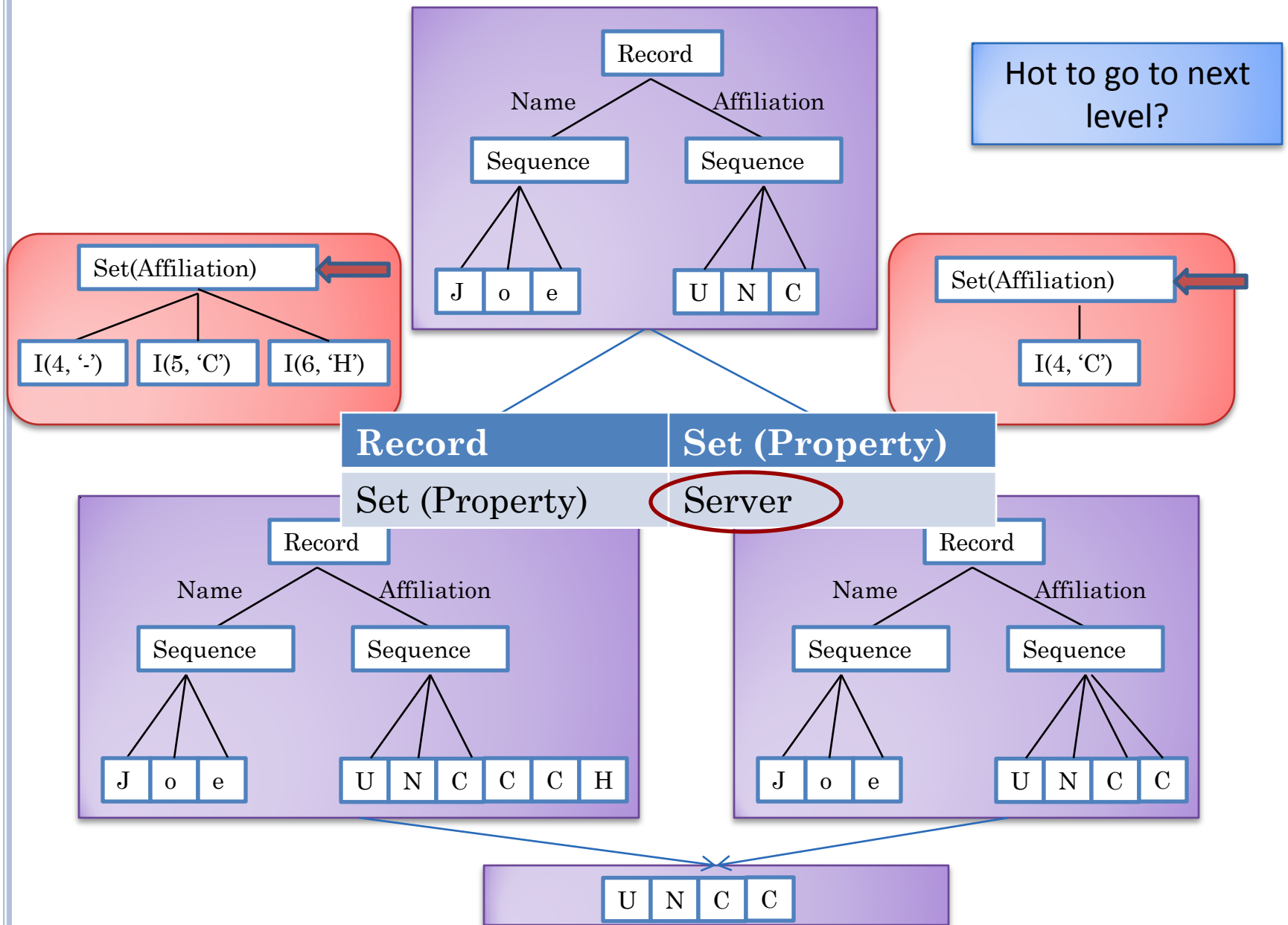
Can we extend model to add the option of accepting all of server or client changes to affiliation?

Need to capture multiple levels of changes

Hierarchical Document



Level 1 Step

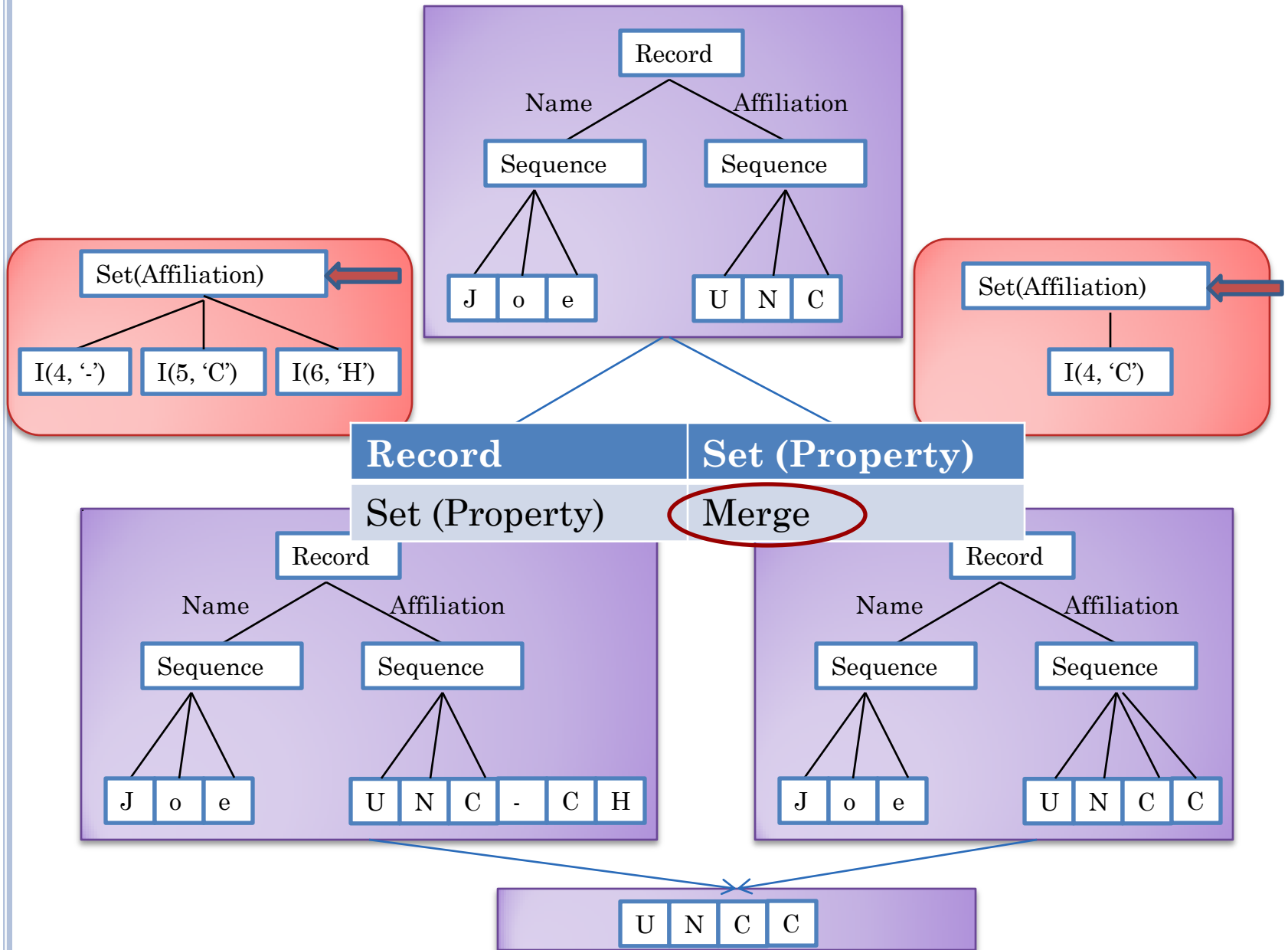

Record
Name
Affiliation
Sequence
Sequence
J o e
U N C
Hot to go to next level?
Set(Affiliation)
I(4, '-')
I(5, 'C')
I(6, 'H')
Set(Affiliation)
I(4, 'C')
Record
Set (Property)
Set (Property)
Server
Record
Name
Affiliation
Sequence
Sequence
J o e
U N C C C H
Record
Name
Affiliation
Sequence
Sequence
J o e
U N C C
U N C C

Merge Next Level Option

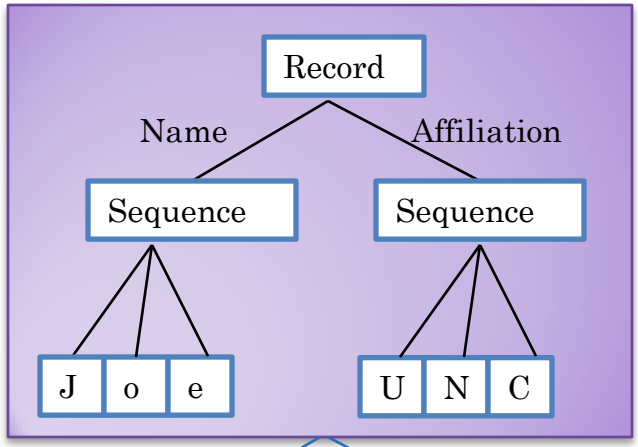
Type	$Operation_1$	…	$Operation_N$
$Operation_1$	$Default_{11}$	…	$Default_{1N}$
…	…	…	…
$Operation_N$	$Default_{N1}$	…	$Default_{NN}$

- Server
- Client
- None
- Both
- Merge

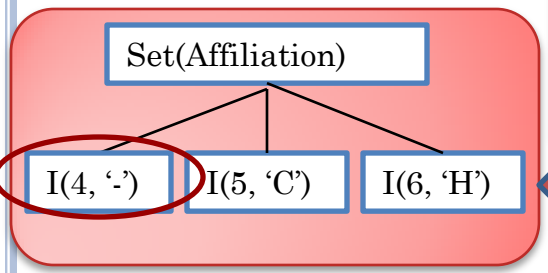
Level 1 Step

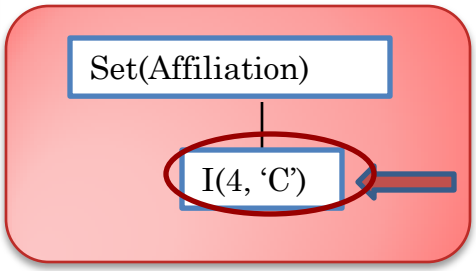

Record
Name
Affiliation
Sequence
Sequence
J
o
e
U
N
C
Set(Affiliation)
I(4, '-')
I(5, 'C')
I(6, 'H')
Set(Affiliation)
I(4, 'C')
Record
Set (Property)
Set (Property)
Merge
Record
Name
Affiliation
Sequence
Sequence
J
o
e
U
N
C
-
C
H
Record
Name
Affiliation
Sequence
Sequence
J
o
e
U
N
C
C
U
N
C
C

Level 2 Step

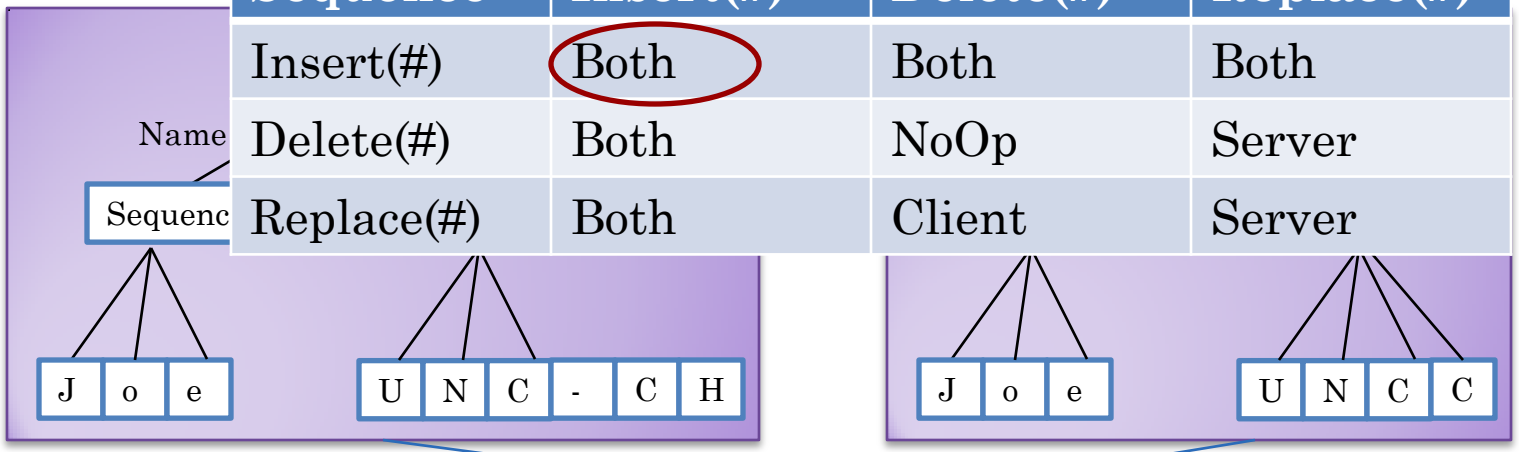


Hot to go to next level?





Sequence	Insert(#)	Delete(#)	Replace(#)
Insert(#)	Both	Both	Both
Delete(#)	Both	NoOp	Server
Replace(#)	Both	Client	Server



Merge Matrix vs. Merge Procedures

```
InsertOperation TransformInsertInsert (InsertOperation R, InsertOperation L) {
        Operation R^T = deepClone();
        if ((R.index > L.index) ||
          (R.index === L.index  && R.isServer()))
                R^T.index = R.index + 1;
        return R^T ;
}
```

Procedural

```
InsertOperation TransformInsertInsert (InsertOperation R, InsertOperation L) {
        Operation R^T = deepClone();
        if ((R.index > L.index) ||
          (R.index === L.index  && !R.isServer()))
                R^T.index = R.index + 1;
        return R^T ;
}
```

Declarative

Declarative is higher level
allows easy customization

But it is less expressive

Sequence	Insert(#)	Delete(#)	Replace(#)
Insert(#)			
Delete(#)			
Replace(#)			

Server
Client
None
Both

Merge Matrix

Covered all asynchronous merge policies known in 94-97

Had mechanisms to extend the default matrix

Reference for Merge Matrix

- Munson and Dewan ‘94, ’97
- Showed that all merge procedures at that time for spreadsheets, file systems, databases, .. could be supported using the merge matrix
- Merge matrix entry could itself have merge procedures in it for a specific combination of operations or level