

# On Deriving Net Change Information From Change Logs – The DeltaLayer-Algorithm –

Stefanie Rinderle-Ma<sup>1</sup>, **Martin Jurisch**<sup>1</sup>, Manfred Reichert<sup>2</sup>

<sup>1</sup>Institute DBIS, Ulm University, Germany

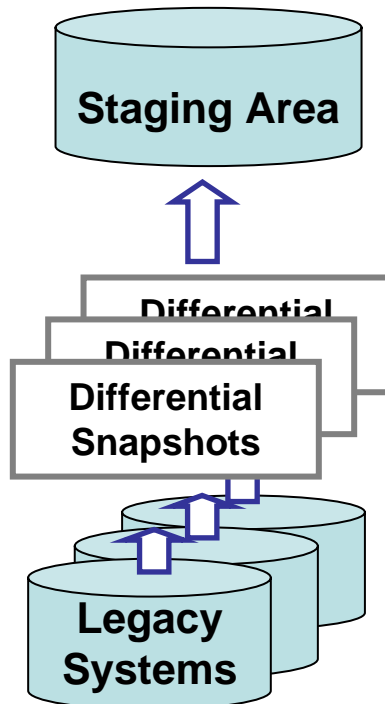
<sup>2</sup>IS Group, University of Twente, The Netherlands



# Motivation

- (interne) Log-Informationen schon immer äußerst wichtig
- zunehmende Bedeutung externer Verarbeitung von Log-Information

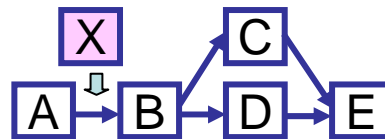
## a) DWH: efficient update



S. Rinderle-Ma, M. Jurisch, M. Reichert

## b) Correctness Checks in Adaptive PMS

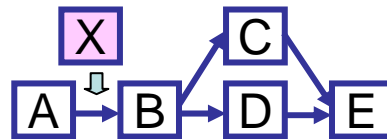
Process Template S:



Change Log for S:  
 INSERT((X, (A,B))),  
 DELETE((B, NULL)),  
 INSERT((B, NULL))

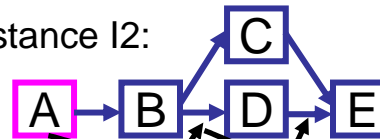
Process Instances:

Instance I1:



Change Log for I1:  
 INSERT((X, (A,B)))

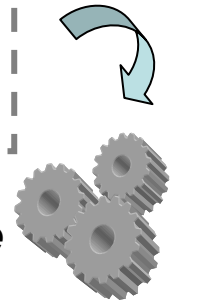
Instance I2:



Change Log for I2:  
 UPDATE((A, (B,D))),  
 UPDATE((A, (D,E)))



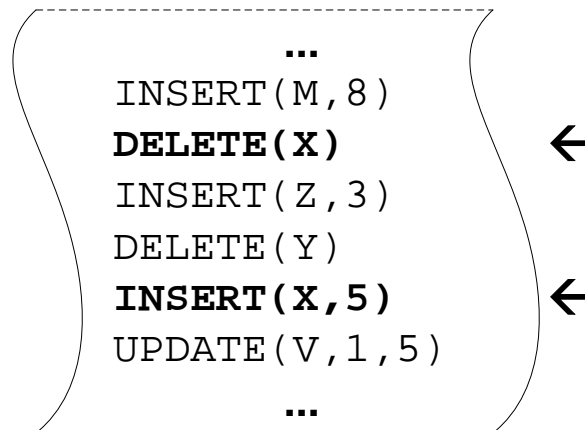
WF-Engine



# Motivation

- Entscheidend für eine effiziente Verarbeitung ist die Güte der Änderungsinformation
- Häufiges Problem:
  - ➔ Änderungslogs enthalten viel unnötige Information
- z.B. DWH:

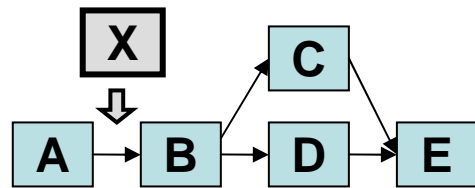
DELETE/INSERT-Paare in “differential snapshots” [LGM 96]



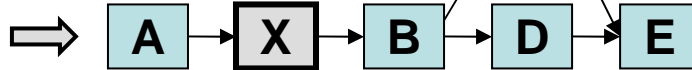
# Motivation

- Unnötige Informationen in DWH “nur” performanzkritisch, im PMS-Umfeld inkorrekte Ergebnisse!

Process Template S

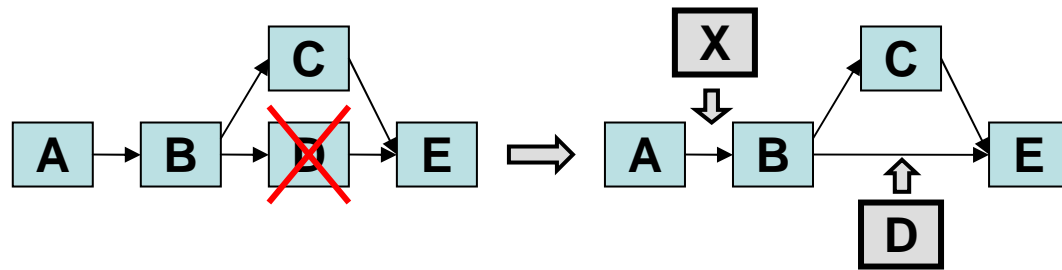


Process Template S'



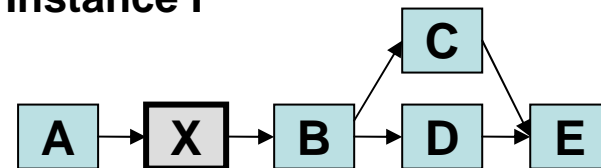
Change log for S:  
INSERT(X, (A,B))

Instance I



Change log for I:  
DELETE(D)  
INSERT(X, (A,B))  
INSERT(D, (B, E))

Instance I'



# Motivation

- Unnötige Informationen verursachen...

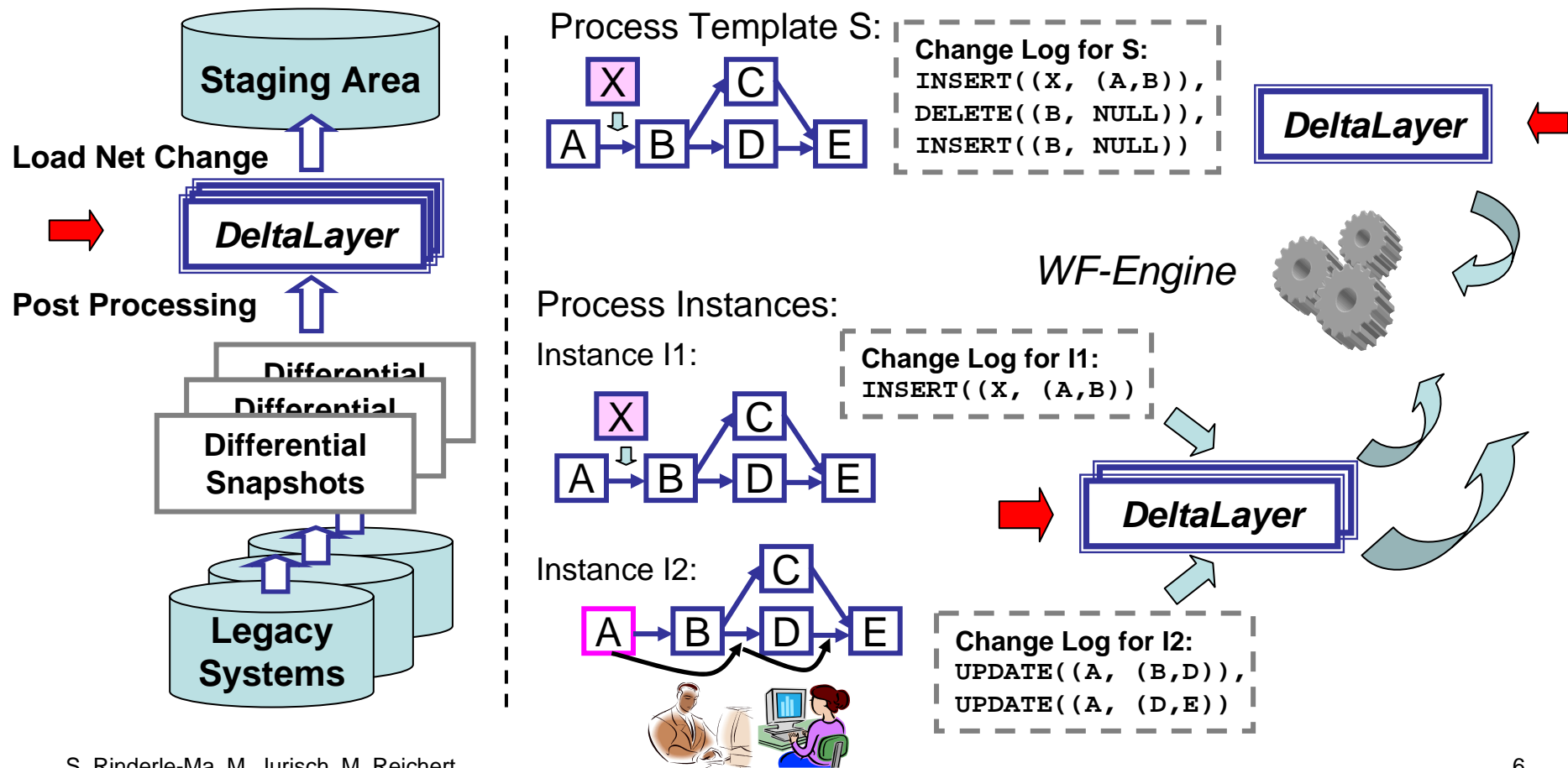
 Performanzprobleme

 Datenqualitätsprobleme

 Vergleichbarkeits-/Korrektheitsprobleme

# Ziel

- Berechnung eines minimalen Deltas aus beliebigen Änderungslogs  $\Rightarrow$  Delta-Schicht (Delta-Layer)



# Herausforderungen

- **Berechnung eines minimalen Deltas aus beliebigen Änderungslogs erfordert:**

➔ Identifikation problematischer Änderungskombinationen

➔ Formale Definition eines Minimalitätskriteriums

➔ Algorithmus zur Berechnung eines minimalen Deltas

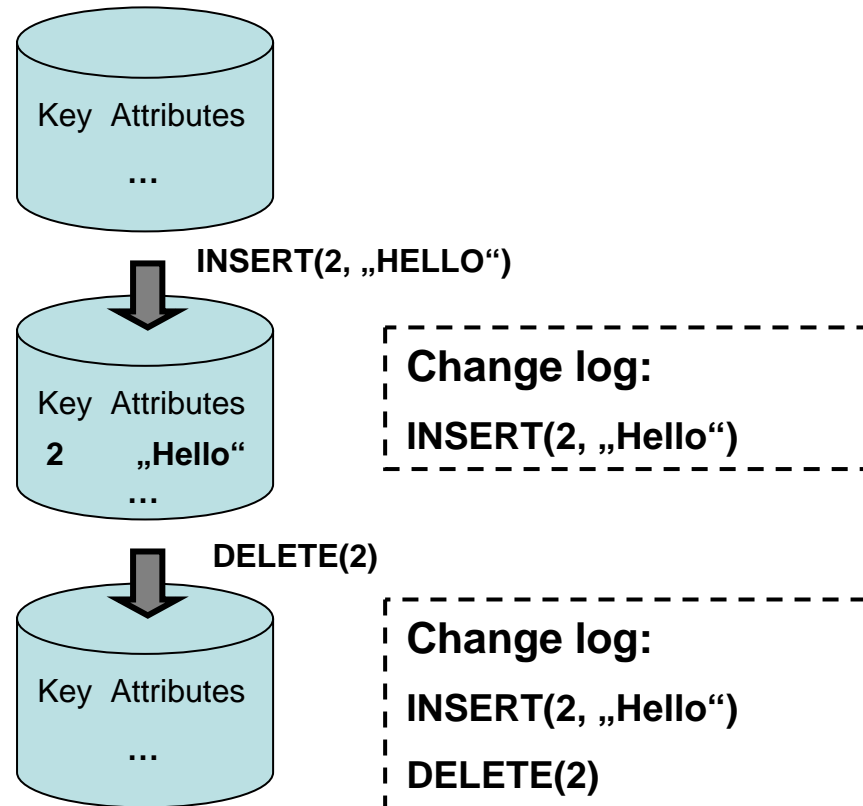
**(The DeltaLayer-Algorithm)**

mit folgenden Eigenschaften:

- effizient
- in der Praxis leicht umsetzbar
- Minimalität der Ausgabe beweisbar

# Problematische Änderungskombinationen

➔ INSERT(T)/DELETE(T)

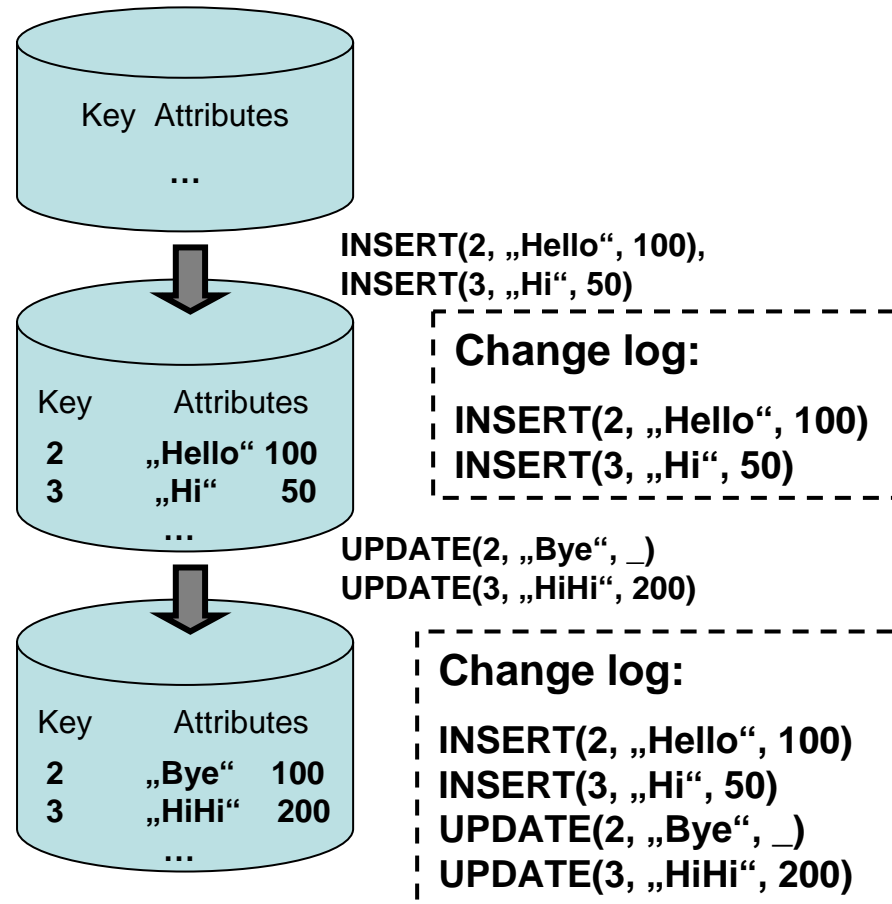


➔ **kein Effekt**

# Problematische Änderungskombinationen

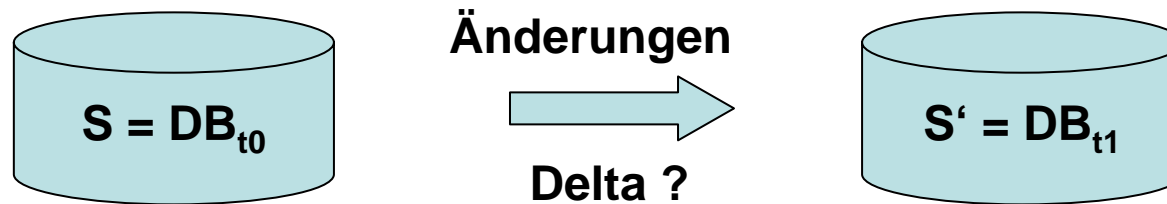
- INSERT(T)/DELETE(T)
- UPDATE(T)/DELETE(T)
- DELETE(T)/INSERT(T)
- UPDATE(T)/UPDATE(T)

➔ INSERT(T)/UPDATE(T)



➔ **Vereinigung für 2 in INSERT(2, „Bye“, 100),  
INSERT(3, „HiHi“, 200) für 3**

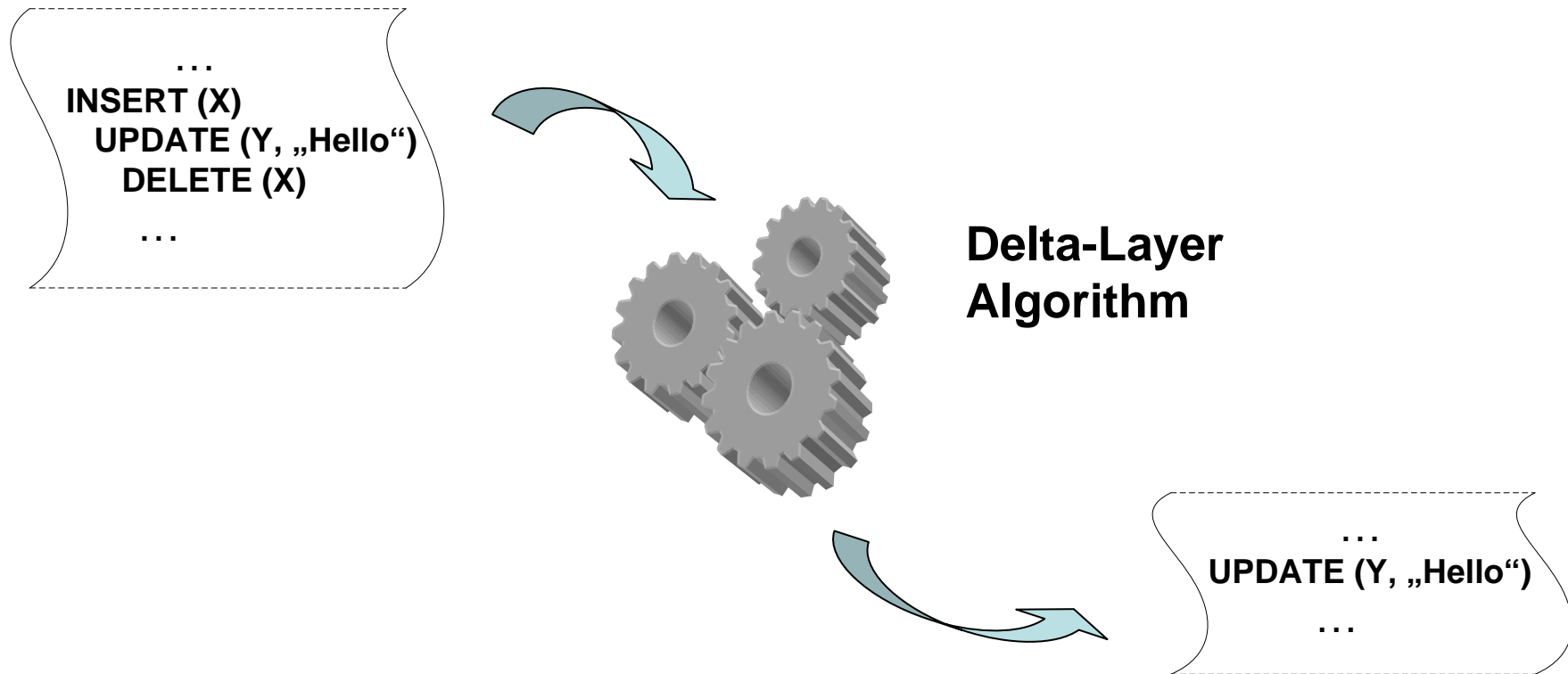
# Minimalitätskriterium



- Delta/Änderungslog  $D$  ist genau dann minimal wenn gilt:
  - $D.deletedTuples =$   
 $\nabla cL := \{t \mid t \in S \setminus S' : \nexists t' \in S' \text{ with } t'.Key = t.Key\}$
  - $D.insertedTuples =$   
 $\Delta cL := \{t \mid t \in S' \setminus S : \nexists t' \in S \text{ with } t'.Key = t.Key\}$
  - $D.updatedTuples =$   
 $\triangleright cL := \{t \mid t \in S \setminus S' \vee t \in S' \setminus S : \exists t' \in S' \text{ with } t'.Key = t.Key\}$

# Delta-Layer Algorithmus

⇒ beliebiges Änderungslog als Eingabe



⇒ Netto-Effekt (“Delta-Schicht”) als Ausgabe

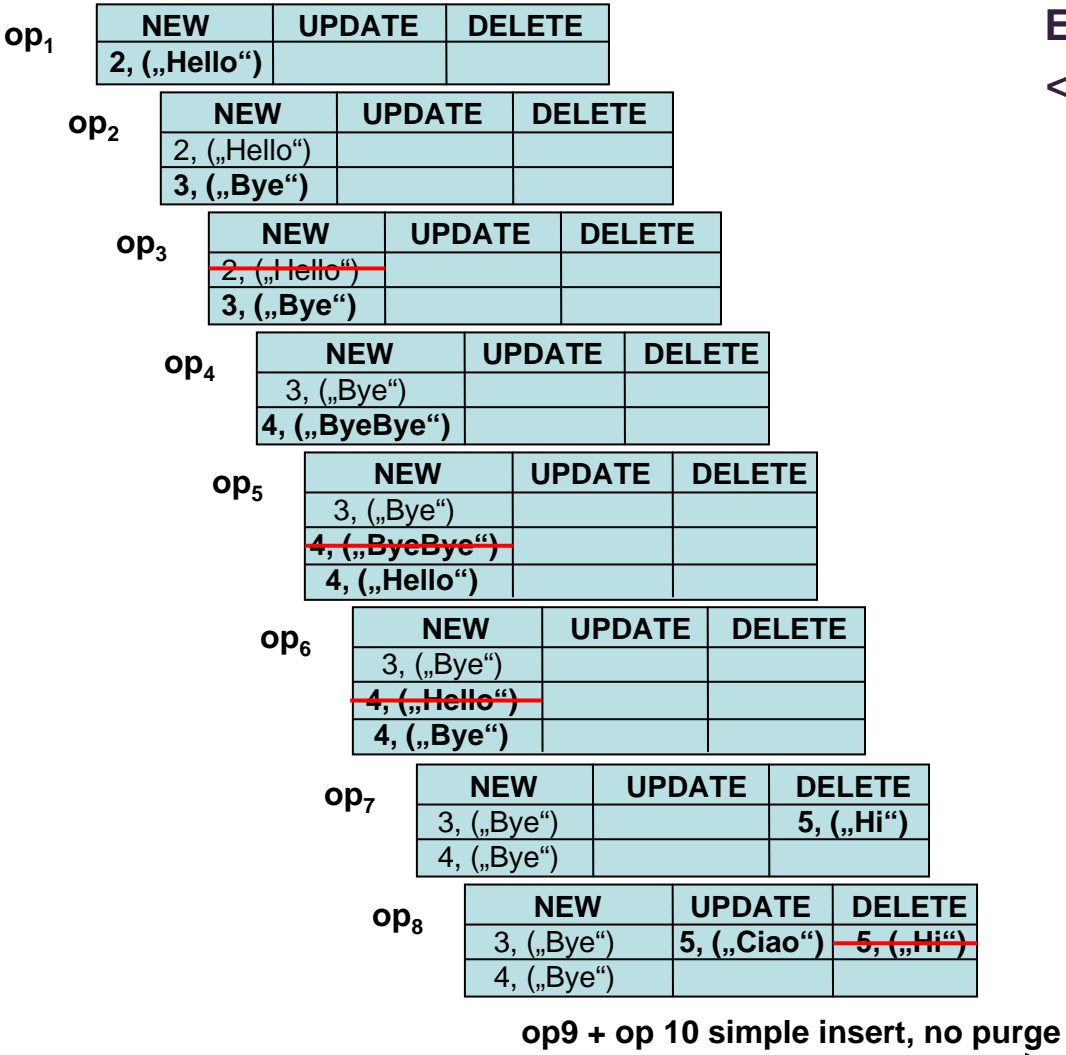
# Delta-Layer Algorithmus

- Beispiel:

Änderungslog:

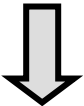
```
< op1 = INSERT(2, "Hello"), ←  
  op2 = INSERT(3, "Bye"),  
  op3 = DELETE(2), ←  
  op4 = INSERT(4, "ByeBye"), ←  
  op5 = UPDATE(4, "Hello"), ←  
  op6 = UPDATE(4, "Bye"), ←  
  op7 = DELETE(5), ←  
  op8 = INSERT(5, "CIAO"), ←  
  op9 = INSERT(6, "Hi"),  
  op10 = INSERT(7, "HiHi") >
```

# Delta-Layer Algorithmus



## Eingabe-Änderungslog:

- < op<sub>1</sub> = INSERT(2, „Hello“), ←
- op<sub>2</sub> = INSERT(3, „Bye“), ←
- op<sub>3</sub> = DELETE(2), ←
- op<sub>4</sub> = INSERT(4, „ByeBye“), ←
- op<sub>5</sub> = UPDATE(4, „Hello“), ←
- op<sub>6</sub> = UPDATE(4, „Bye“), ←
- op<sub>7</sub> = DELETE(5), ←
- op<sub>8</sub> = INSERT(5, „CIAO“), ←
- op<sub>9</sub> = INSERT(6, „Hi“), ←
- op<sub>10</sub> = INSERT(7, „HiHi“) > ←



## Ausgabe

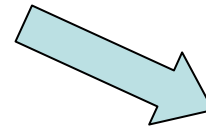
NEW	UPDATE	DELETE
3, („Bye“)	5, („Ciao“)	
4, („Bye“)		
6, („Hi“)		
7, („HiHi“)		

# Einsatz in existierenden Systemen

- IBM Data Propagator [IBM95]: “*Condensed Consistent Change Data Table*”

Tupel	Action
(2, „Hello“)	„I“
...	

DELETE(2, „Hello“)



Tupel	Action
(2, „Hello“)	„D“
...	



- Oracle *Data Warehousing Component* [Dat03]: “*Merge-Statement*”

# Zusammenfassung und Ausblick

- **Delta-Layer-Algorithmus...**
  - ... ermöglicht die Berechnung der reinen Netto-Änderungen
  - ... ist effizient
  - ... leicht in existierende Systeme zu integrieren
  - Minimalität des Outputs ist bewiesen
- ➔ schafft die Grundlage für effiziente und korrekte Verarbeitung von Änderungsinformation

Danke für Ihre Aufmerksamkeit.

# Referenzen

- [LGM96] W. Labio and H. Garcia-Molina. Efficient Snapshot Differential Algorithms for Data Warehousing. In *Proc. Int'l Conf. VLDB*, pages 63–74, 1996.
- [IBM95] IBM. Data Where You Need It, The DPropR Way! DataPropagator Relational Solutions Guide. Technical Report GG24-4492-00, IBM International Technical Support Organization, San Jose Center, 1995.
- [Dat03] Oracle Database. *Data Warehousing Guide, 10g Release 1 (10.1)*, December 2003.