



Synchronisierung und Immutability

Erfahrungsbericht aus der Praxis von

Dr. Andreas Bernauer, Active Group GmbH

@lysium

Immutability erleichtert Synchronisierung

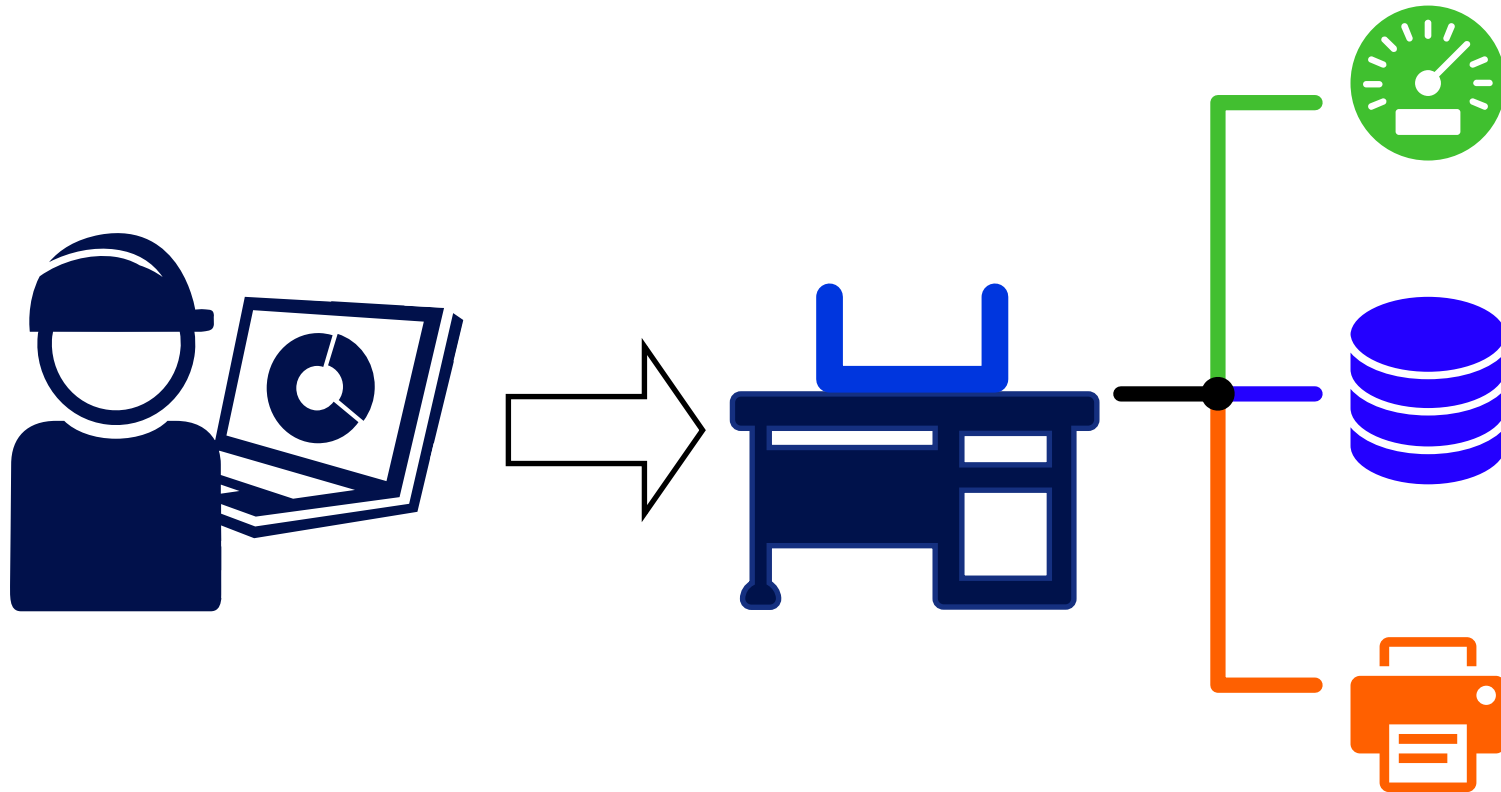
Kanzlerwechsel stellt Konsistenz sicher

Merkle-Trees synchronisieren schnell

Monaden erleichtern Umgang mit Datenbank

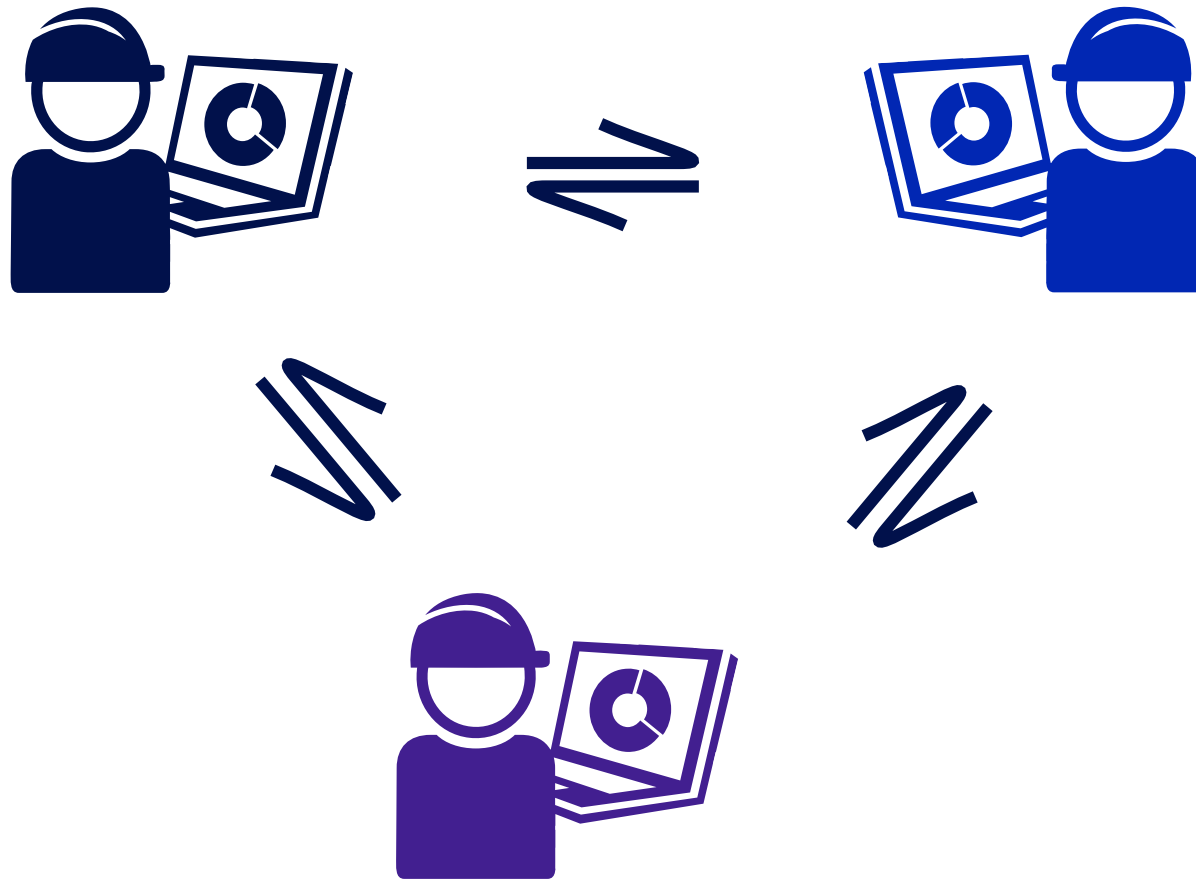
Kundenwunsch

Mobile Geräte für aktuelle Prüfstation konfigurieren



Kundenwunsch

Mobile Geräte untereinander synchron halten



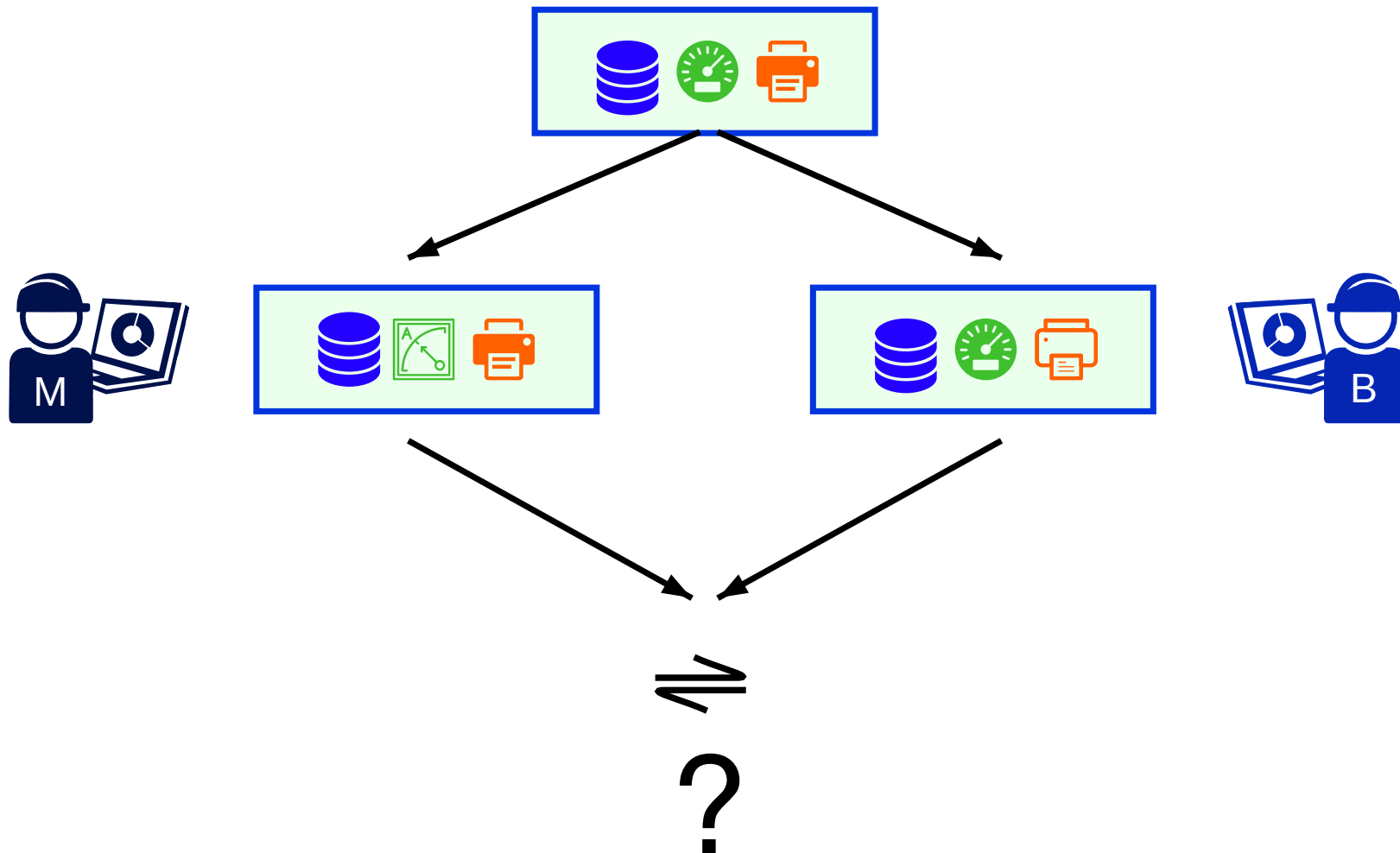
Synchronisierung

Wie synchronisieren?



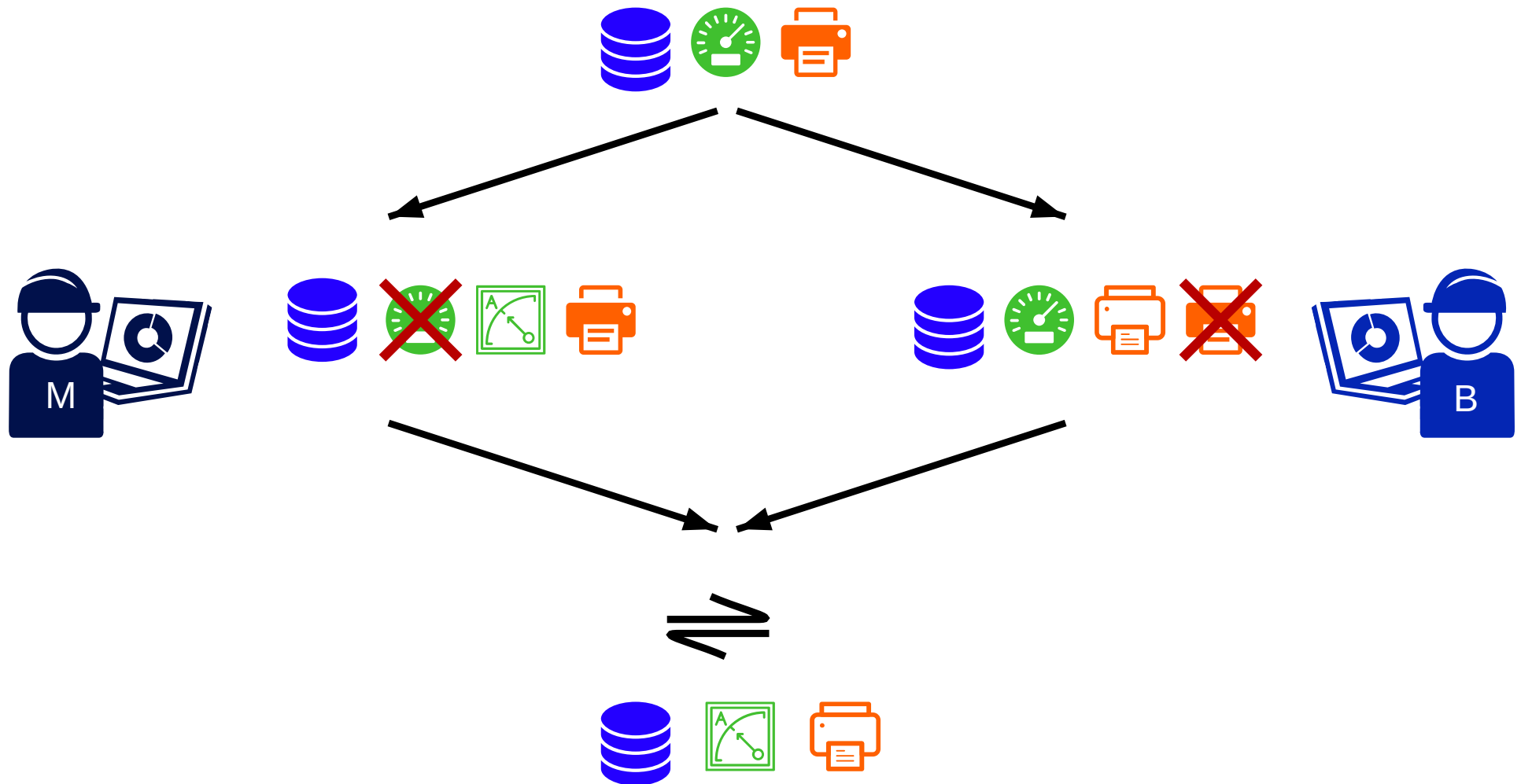
Fakten

Was synchronisieren?



Fakten

Möglichst kleinteilig synchronisieren



API, v1

Lesen und Schreiben

`get` : GuidT → PropertyT → ValueT
 ↑ ↑ ↑
 System.Guid „string“ byte[]

`put` : GuidT → PropertyT → ValueT → HashT
 ↑
 „ID“

Beispiel:

```
let station1Printerid = put station1Guid  
                                          (Property „drucker“)  
                                          (utf8Bytes „HP2130“)
```

Löschen

Wie mit Löschen umgehen?



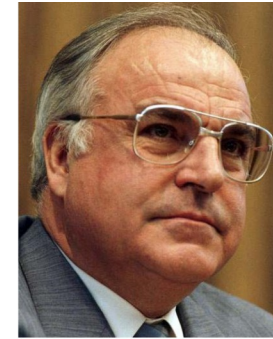
Löschen „geht nicht“ bei verteilter Datenhaltung: **Immutability**

Kanzlerwechsel

Wie überschreiben?



Kanzlerwechsel



Kanzlerwechsel: „Löschen mit Hinzufügen“ als atomare Operation

API, v2

mit Suche und Kanzlerwahl

`get` : GuidT → PropertyT → ValueT

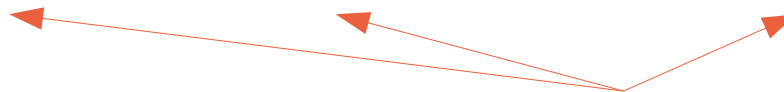
`put` : GuidT → PropertyT → ValueT → **ObsoletesT** → HashT

↑
HashT[]

`search` : (GuidT option) * (PropertyT option) * (ValueT option) →

HashT[]

suchen, falls gesetzt



API, v2

Beispiel:

```
let oldPrinterIds = search (Some station1Guid,  
                          Some printerProperty,  
                          None)
```

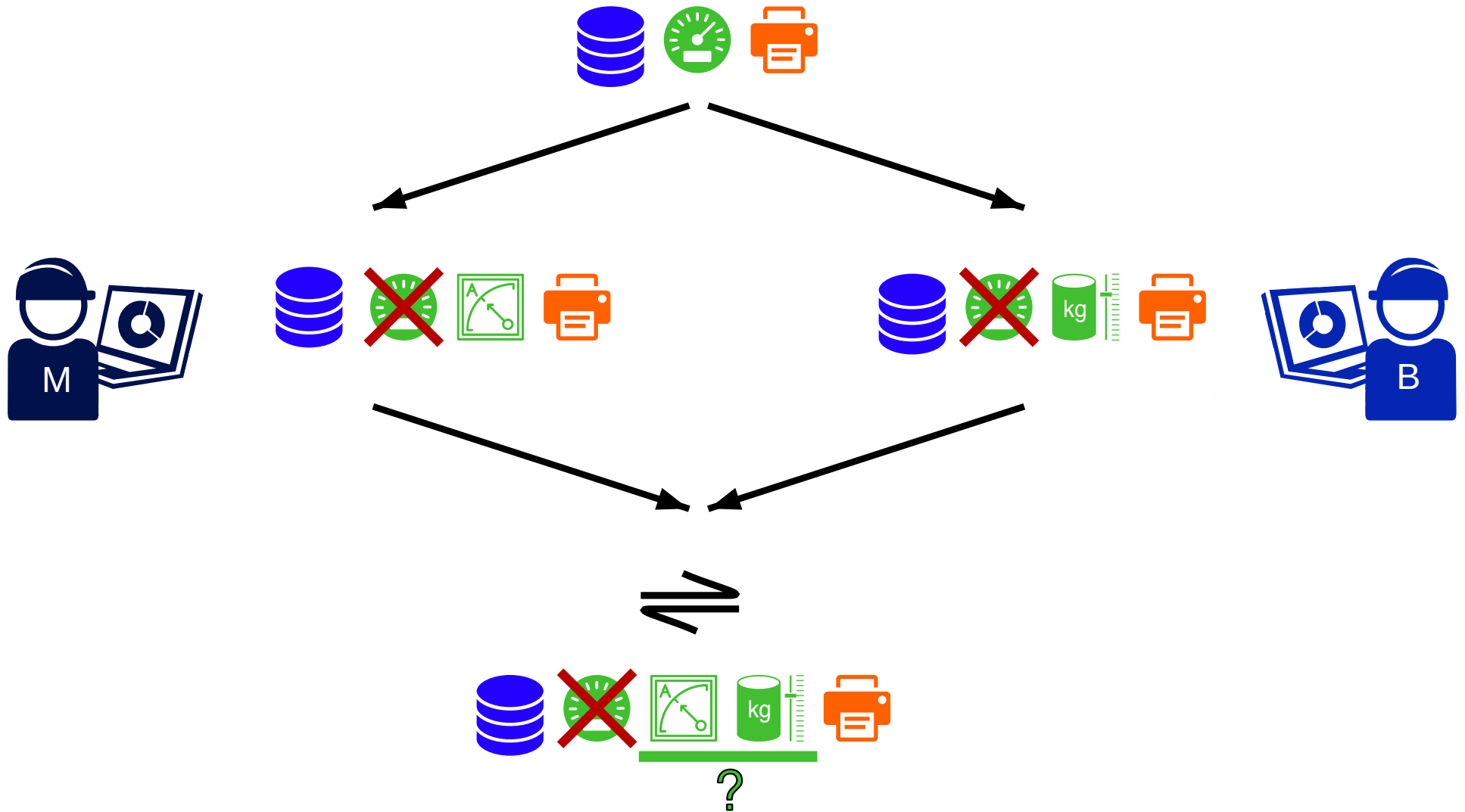
```
let newPrinterId = put station1Guid  
                    printerProperty  
                    (utf8Bytes „MFC-J46“)  
                    oldPrinterIds // id of „HP2130“
```

Abstrahiert:

```
let update guid prop value = // „Kanzlerwahl“  
  let oldIds = search (Some guid, Some prop, None)  
  put guid prop value oldIds
```

Konflikte

Wann entstehen Konflikte?



Konflikte

Wie Konflikte auflösen?

Durch zeitgleiche Änderungen gibt es für die Eigenschaft
"Meßgerät" der Station "Prüfstation 1"
folgende Werte, die sich widersprechen:

Bitte wählen Sie den Wert, der übernommen werden soll:

Wert	Mitarbeiter	MD	Datum
Barometer 92B	Michaela	HNPC1	2015-09-25T09:58:40.8784083+00:00
Waage TR12	Ben	HNPC2	2015-10-16T09:29:23.8363926+00:00

Übernehmen

Manuell, beim Lesen

API, v3

erlaubt Konflikte

`get` : GuidT → PropertyT → ValueT[]

Bei Konflikt ist size > 1

`put` : GuidT → PropertyT → ValueT →

MetaT → ObsoletesT → HashT

{user; machine; datetime}

Beispiel:

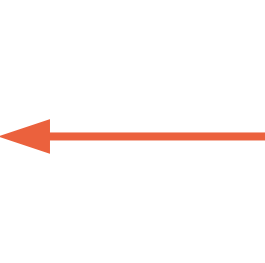
```
let newPrinterId = put station1Guid
                    printerProperty
                    (utf8Bytes „MFC-J46“)
                    (Meta.create())
                    oldPrinterIds
```

Implementierung

Datenbank

- .Net-Paket `Active.Net.AddOnlyDb` (F#, via nuget)
- `SQLite` als backend
- 2 Tabellen: `facts` und `deletes`

facts		deletes	
hash	BLOB	key_hash	BLOB
guid	BLOB	obsoleted_hash	BLOB
property	STRING		
value	BLOB		
meta	STRING		



Implementierung

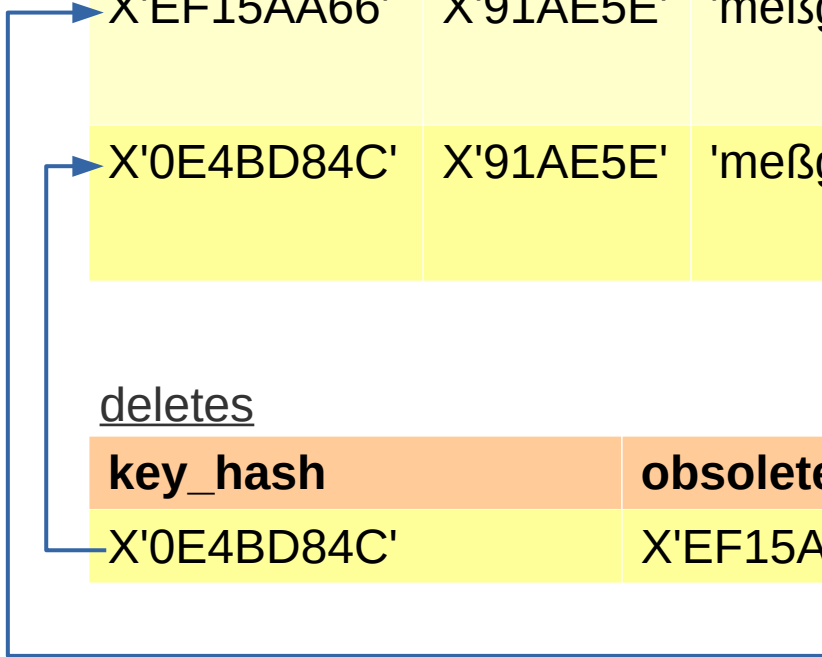
Tabellen

facts

hash	guid	property	value	meta
X'C83BEB05'	X'91AE5E'	'drucker'	X'485032313330' („HP2130“)	'{"user":"michael","machine":"bob","datetime":"2015-10-20T09:25:10.4393052+00:00"}'
X'EF15AA66'	X'91AE5E'	'meßgerät'	X'393242' („92B“)	'{"user":"michael","machine":"bob","datetime":"2015-10-20T09:25:10.2885825+00:00"}'
X'0E4BD84C'	X'91AE5E'	'meßgerät'	X'54523132' („TR12“)	'{"user":"ben","machine":"alice","datetime":"2015-10-24T05:41:08.9448352+00:00"}'

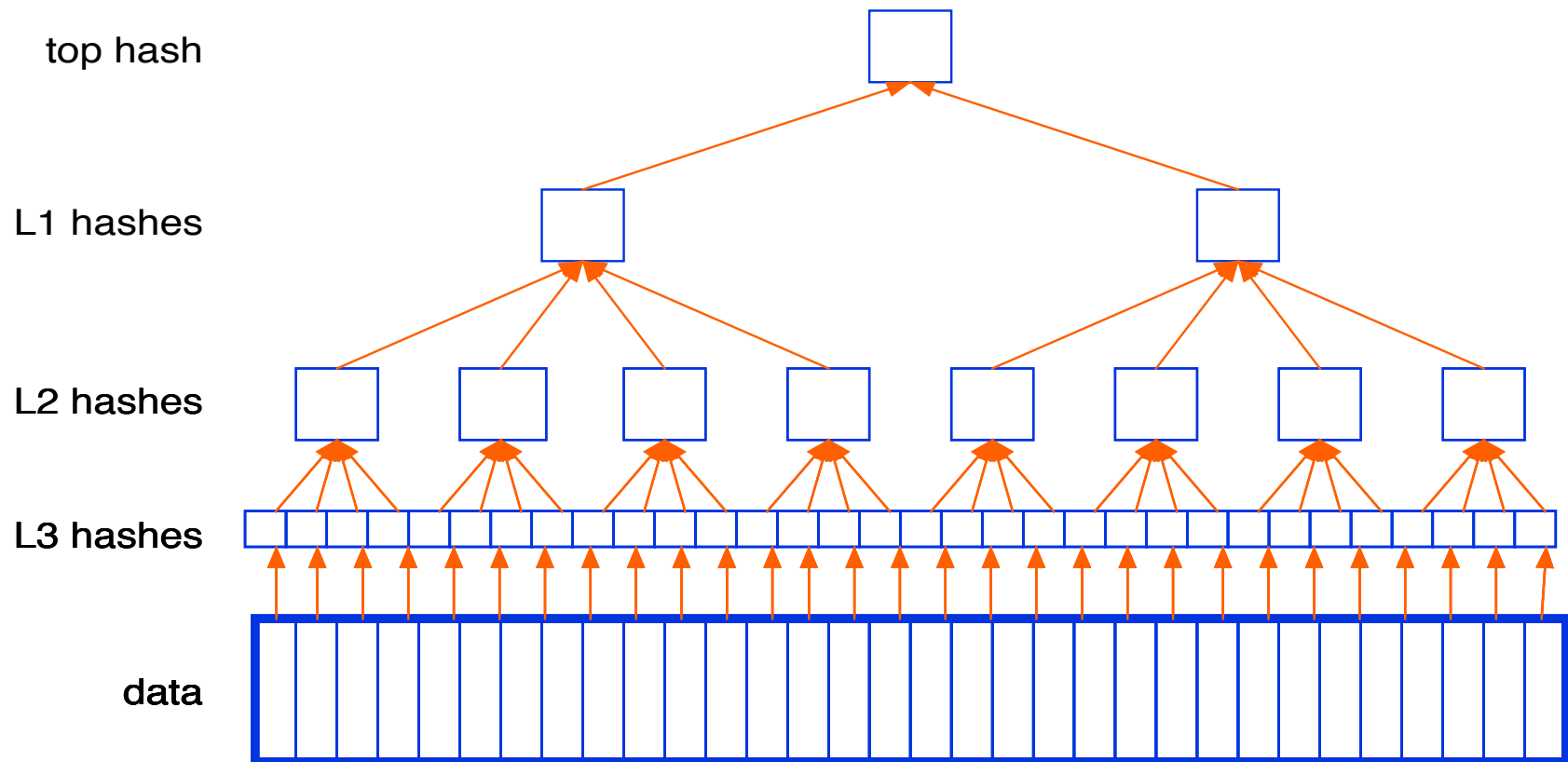
deletes

key_hash	obsoleted_hash
X'0E4BD84C'	X'EF15AA66'



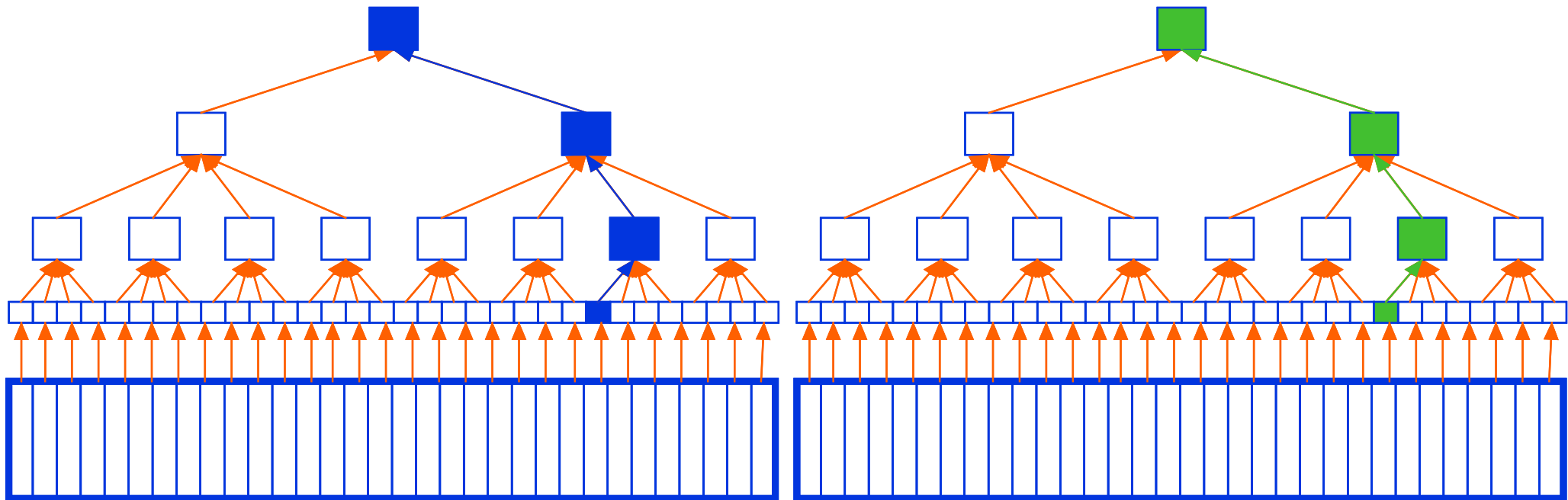
Synchronisierung

Merkle-Tree



Synchronisierung

Merkle-Tree



Naiver Code

```
let update guid prop value =  
  let oldIds = search (Some guid, Some prop, None)  
  put guid prop value (Meta.create()) oldIds
```

Mit Verweis auf Datenbank

```
let update db guid prop value =  
  let oldIds = search db (Some guid, Some prop, None)  
  put db guid prop value (Meta.create()) oldIds
```

Mit Transaktion

```
let update db guid prop value =  
  startTransaction db  
  let oldIds = search db (Some guid, Some prop, None)  
  put db guid prop value (Meta.create()) oldIds  
  commitTransaction db
```

db-Monade

aka „Builder“ in .Net

```
let update guid prop value =  
  atomically  
  db {  
    let! oldIds = search (Some guid, Some prop, None)  
    do! put guid prop value (Meta.create()) oldIds  
  }
```

Anwendung:

```
let db = openDatabase ...  
let op = update guid prop value  
run db op
```

db-Monade

Basis-Typen und Funktionen

„continuation“ /
„callback“



```
type 'a Op =  
  | Result of 'a  
  | Get of GuidT * PropertyT * (ValueT [] -> 'a Op)  
  | Put of FactT * (HashT -> 'a Op)  
  | Atomically of unit Op * (unit -> 'a Op)
```

```
let rec bind (opB: 'b Op) (next: 'b -> 'a Op) : ('a Op) =  
  match opB with  
  | Result b          -> next b  
  | Get (g, p, cb) -> Get (g, p, (fun v ->  
                                let cbOp = cb v  
                                bind cbOp next))  
  | Put (d, cb)      -> Put (d, (fun v -> bind (cb v) next))  
  | Atomically (op, cb) ->  
    Atomically (op, (fun () -> bind (cb ()) next))
```

db-Monade

F#-Implementierung

```
let db = DbBuilder()

type DbBuilder() =
    member this.Return(v) = Result v // : 'a → 'a Op
    member this.Bind(op, next) = bind op next
                                   // : 'b Op → ('b → 'a Op) → 'a Op
```

```
db {
    let! oldIds = search (Some guid, Some prop, None)
    do! put guid prop value (Meta.create()) oldIds
}
```

==>

```
Bind(search (Some guid, Some prop, None), (fun oldIds →
    Bind(put guid prop value (Meta....) oldIds, (fun x →
        Return ())))))
```

db-Monade

Runner

```
let run db (op0:'a Op) : 'a =
  runLoop db false op0

let rec runLoop<'a> db (inTransaction:bool) (op:'a Op) : 'a =
  match op with
  | Result v -> v
  | Get (guid, prop, cb) ->
      let resultOfGet = get db guid prop
      let nextOp = cb resultOfGet
      runLoop db inTransaction nextOp)
  | Put ((guid, prop, value, meta, obsoletes), cb) -> // ...
  | Atomically (op, cb) ->
      let res = runTransaction db inTransaction op
      runLoop db inTransaction (cb res)
```

db-Monade

Runner

```
and runTransaction<'b> db (inTrans:bool) (op: 'b Op) : 'b =
  if inTrans
  then runLoop db true opB
  else
    try
      openTransaction db
      let res = runLoop db true opB
      commitTransaction db
      res
    with
    | e ->
      rollbackTransaction db
      raise e
```


Zusammenfassung: AddOnlyDb

Immutability	erleichtert Synchronisierung
Kanzlerwechsel	stellt Konsistenz sicher
Merkle-Trees	synchronisieren schnell
Monaden	erleichtern Umgang mit Datenbank