

# Information Engineering 2

## Dimensionale Datenmodellierung 2

Prof. Dr. Kurt Stockinger

SW	Datum	Vorlesungsthema	Praktikum
1	23.02.2022	Data Warehousing Einführung	Praktikum 1: KNIME Tutorial
2	02.03.2022	Dimensionale Datenmodellierung 1	Praktikum 1: KNIME Tutorial (Vertiefung)
3	09.03.2022	Dimensionale Datenmodellierung 2	Praktikum 2: Datenmodellierung
4	16.03.2022	Datenqualität und Data Matching	Praktikum 3: Star-Schema, Bonus: Praktikum 4: Slowly Changing Dimensions
5	23.03.2022	Big Data Einführung	DWH Projekt - Teil 1
6	30.03.2022	Spark - Data Frames	DWH Projekt - Teil 2 (Abgabe: 4.4.2022 23:59:59)
7	06.04.2022	Data Storage: Hadoop Distributed File System & Parquet	Praktikum 1: Data Frames
8	13.04.2022	Query Optimization	Praktikum 2: Data Storage
9	20.04.2022	Spark Best Practices & Applications	Praktikum 3: Query Optimization & Performance Analysis
10	27.04.2022	Machine Learning mit Spark 1	Praktikum 3: Query Optimization & Performance Analysis (Vertiefung)
11	04.05.2022	Machine Learning mit Spark 2 + Q&A	Praktikum 4: Machine Learning (Regression)
12	11.05.2022	NoSQL Systems	Big Data Projekt - Teil 1
13	18.05.2022	Keine Vorlesung (Arbeit am Projekt)	Big Data Projekt - Teil 2
14	25.05.2022	Keine Vorlesung (Arbeit am Projekt)	Big Data Projekt - Teil 3 (Abgabe: 30.5.2022 23:59:59)

# Ziele der heutigen Lektion

- Kennen von
  - Sternschema Fakten- und Hierarchietypen
- Verstehen von
  - Historisierung
  - Slowly Changing Dimensions
- Beherrschen von
  - Modellierung von Hierarchien
  - Laden und Nutzen von Slowly Changing Dimensions Typ 2

# Wiederholung

- Geben Sie Beispiele für unterschiedliche Dimensionstabellen an
- Geben Sie Beispiele für Faktentabellen an
- Was ist ein Star-Schema und wofür wird es eingesetzt?



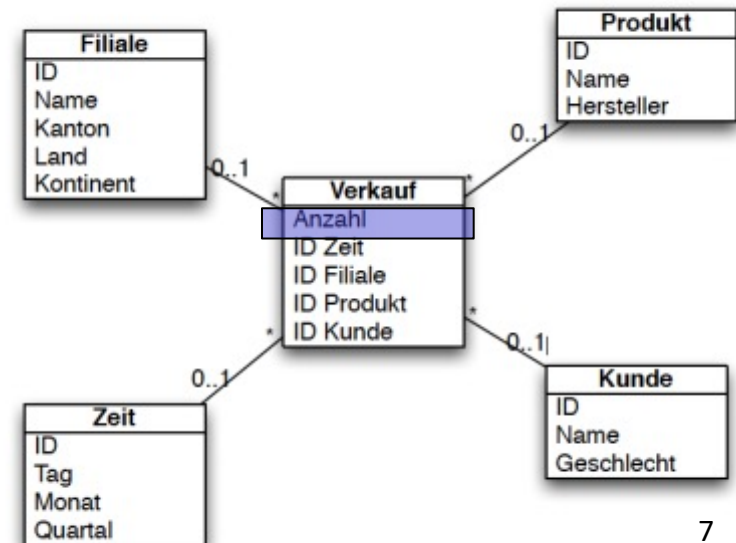
# Faktentypen

# Typen von Fakten

- Fakten sind **quantifiziert**:
  - Meist betriebswirtschaftliche Grössen:
    - z.B. Umsatz, Gewinn, Rentabilität
  - Beschreibende Attribute sind ebenfalls relevant:
    - z.B. Einheit, Wertebereich, Berechnungsvorschrift
- Fakten sind **aggregierbar**:
  - **Additiv**: Addition entlang aller Dimensionen möglich
    - z.B. Produktionsdaten, Verkaufszahlen,...
  - **Semi-additiv**: Addition nur entlang ausgewählter Dimensionen gestattet
    - z.B. Kontostand, Anzahl von Versicherungsverträgen
  - **Nicht-additiv**:
    - z.B. Durchschnittswerte, Prozentanteile, Extrema (Max/Min)

# Charakteristika von Fakten

- Charakteristik:
  - Oft nur eine konkrete „**Anzahl**“
    - z.B. Grösse der Transaktion
  - Kontext durch unabhängige Dimensionen gegeben
- Faktentabelle beschreibt eine atomare Aktion zu einem gegebenen Zeitpunkt
- Keine Updates, nur Inserts



# Modellierung von Fakten

- Wie speichern Sie folgende Fakten?
  - Einzahlung von 3000 CHF am 1.2.2015
  - Abhebung von 200 CHF am 4.2.2015
  - Einzahlung von 1000 CHF am 9.2.2015





# Beispiel: Transaktionen am Bankkonto

Transaktionen:

Date	Amount
1. 2. 2015	3000
4. 2. 2015	-200
9. 2. 2015	1000

3 unterschiedliche Abfragen:

- a) Kontostand am 4. 2. 2015
- b) Kontostand am 9. 2. 2015
- c) Wieviel wurde am 9.2. abgehoben?

# Beispiel: Modellierungstyp Transaction

Date	Amount
1. 2. 2015	3000
4. 2. 2015	-200
9. 2. 2015	1000

3 unterschiedliche Abfragen:

- a) Kontostand am 4. 2. 2015
- b) Kontostand am 9. 2. 2015
- c) Wieviel wurde am 9.2.  
abgehoben?

SQL Statements?

# Beispiel: Modellierungstyp Periodic Snapshot

Date	Amount
1. 2. 2015	3000
4. 2. 2015	-200
9. 2. 2015	1000



Date	Amount
1. 2. 2015	3000
2. 2. 2015	3000
3. 2. 2015	3000
4. 2. 2015	2800
...	
8. 2. 2015	2800
9. 2. 2015	3800

Annahme: Periodische Snapshots  
werden täglich erstellt

3 unterschiedliche Abfragen:

- a) Kontostand am 4. 2. 2015
- b) Kontostand am 9. 2. 2015
- c) Wieviel wurde am 9.2.  
abgehoben?

SQL Statements?

# Bestände: Status zu gegebenem Zeitpunkt

- Bestände := Inhalt / Stand einer Faktentabelle
- „**Periodic Snapshot**“: Periodischer Abzug
  - z.B. Monatsstand
  - Repräsentiert **definierte Zeitspanne**
  - Zeitdimension gibt meist das Ende der Spanne an
  - z.B. **pro Police/Konto existiert eine Faktenzeile**
    - **Auch dann, wenn keine Transaktionen durchgeführt wurden**
  - Viele Kennzahlen
    - **Kennzahlen, die schwer auf Basis von Transaktionen zu berechnen wären**
    - z.B. verdiente Prämie, Kundenklasse, ...
  - Ein Record pro Kombination der relevanten Dimensionskeys:
    - z.B. Kontotyp x Kunde x Zeitraum

# Vorteil und Nachteil von Periodic Snapshots



# Vorteil und Nachteil von Periodic Snapshots

- Vorteil:
  - Kontostandabfragen sind einfach
- Nachteil:
  - Einzelne Transaktionen nicht direkt ersichtlich,
  - Wenn es wenig Änderungen gibt, muss Tabelle trotzdem jeden Tag aufgebaut werden (wird gross)



# Beispiel: Modellierungstyp Accumulating Snapshot

Date	Amount
1. 2. 2015	3000
4. 2. 2015	-200
9. 2. 2015	1000

Nur neuer Record, wenn es  
Änderungen gibt



Date	Amount	Effective Date	Expiration Date
1. 2. 2015	3000	1. 2. 2015	4. 2. 2015
4. 2. 2015	2800	4. 2. 2015	9. 2. 2015
9. 2. 2015	3800	9. 2. 2015	

# Bestände

- „**Accumulating Snapshot**“: Akkumulierter Abzug
  - **Zeitperiode** oft **nicht fest definiert**
    - Startzeitpunkt bei Kontoeröffnung/Vertragsabschluss
    - Endzeitpunkt: Gegenwart
  - z.B. jedes Konto verfügt über eine oder mehrere Faktenzeilen
    - Mindestens eine Eröffnungszeile
    - **Eine Zeile für jedes Ereignis**
  - Updates auf Fakten werden appliziert
    - z.B. Kollisionsdeckung wird verändert
    - dieser Wert wird von einer einzigen Zeile angegeben



# Nutzung der Typen

- Fragestellungen bei **Transaktionstyp**:
  - Analyse von Zeitpunkten
    - Wie lange dauert es vom Kundenantrag bis zur Kontoeröffnung?
    - z.B. Prozesseffizienz
  - Detaillierte **Verlaufsanalysen**
  - Unvorhersehbares, repetitives Verhalten
- Fragestellung bei **periodischem Abzug**:
  - **Geschäftsübersicht**
    - z.B. Revenue, Cost
- Fragestellungen bei **akkumulierendem Abzug**:
  - Beobachtung von **Produkten mit endlichem Lebenszyklus**
    - Versicherungsverträge, Konten
    - Deckt Aktivitäten um das Produkt ab

# Modellierungstypen für Fakten

- Insgesamt 3 Modellierungstypen
- Nutzung, um Geschäftsprozesse abzubilden
  - Genutzt werden meist einer oder zwei der drei Typen
- Die Dimensionen sind jeweils durch dieselben Tabellen abgebildet

## Transaction

	Dimensionskeys:  Transaction Key (Typ der Transaction)  Time: Timestamp  Fakten:  Grösse der Transaktion (Amount)

## Periodic Snapshot

	Dimensionskeys  Status Key (Status des Bestands)  Time: Reporting Month  Fakten:  Anzahl Transaktionen  Bestandszahlen (Earned premium, incurred claims...)

## Accumulating Snapshot

	Dimensionskeys  Time: Snapshot Timestamp  Effective Date Expiration Date First Transaction Date Last Transaction Date Status Fakten: Bestände bis zum Snapshot Timestamp

# Beispiele

Transaction Grain Fact	Periodic Snapshot Grain Fact	Accumulating Snapshot Grain Fact
Transaction Date Key (FK)	Reporting Month Key (FK)	Snapshot Date Key (FK)
Policy Key (FK)	Policy Key (FK)	Effective Date Key (FK)
Customer Key (FK)	Customer Key (FK)	Expiration Date Key (FK)
Agent Key (FK)	Agent Key (FK)	First Claim Date Key (FK)
Coverage Key (FK)	Coverage Key (FK)	Last Payment Date Key (FK)
Covered Item Key (FK)	Covered Item Key (FK)	Policy Key (FK)
Transaction Key (FK)	Status Key (FK)	Customer Key (FK)
Amount	Earned Premium	Agent Key (FK)
	Incurred Claims	Coverage Key (FK)
	Change in Reserve	Covered Item Key (FK)
	Reserve Balance	Status Key (FK)
	Number Transactions	Earned Premium to Date
		Number Claims to Date
		Claim Payments to Date

Quelle: <http://www.kimballgroup.com/2008/11/05/fact-tables/>

# Dimensionen mit Hierarchien

# Dimensionen und Hierarchien

- Dimensionen:
  - Dimensionen enthalten **Referenzdaten** in **Sternschemata**
  - **Referenzdaten** werden in **Hierarchien** gruppiert
  - **Gruppierungen** über mehrere Hierarchieebenen gestatten **Aggregationen/Drill-down**
- Hierarchien:
  - Hierarchien sind **Gruppen von Entitäten**, die in eine Abfolge nach Grad, Klasse etc. gebracht werden
  - z.B.: Organisationsstrukturen, Produktkataloge, Zeit

# Parent-Child Hierarchien

- Typisch ist sie für **Organisationen** oder **Produkte**
- Parent-Child Hierarchien sind typischerweise **Baumstrukturen**
- Baumstrukturen lassen sich in Tabellen abbilden, indem ein Datensatz auf seinen Vorgänger verweist
- Diese Darstellung hat **verschiedene Namen**:
  - rekursive Hierarchie
  - parent-child hierarchy (Eltern-Kind)
  - value-based hierarchy

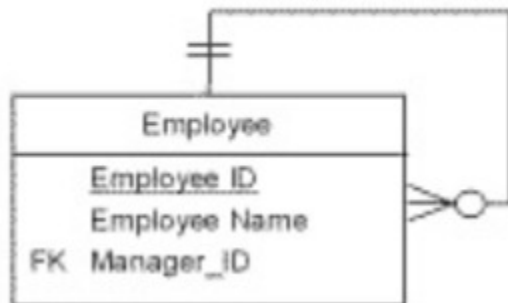
ID	OrgEinheit	Parent
1	Inst. f. angewandte IT	3
2	Inst. f. angewandte Physik	3
3	School of Engineering	5
4	School of Management & Law	5
5	ZHAW	-

# Logische Modellierung

- (Normalisierte) Datenmodellierung:
  - Rekursion
- Allgemeine Beziehungstypen:
  - 1:1
    - z.B. Ehepartner
  - 1:n
    - z.B. Manager - Mitarbeiter
  - n:m
    - z.B. Produkt und Teile; ein Zylinder kann mehrfach in einem Motor vorkommen

# 1:n Hierarchie

- Ein **Mitarbeiter hat einen Vorgesetzten**. Dieser ist selbst Mitarbeiter
  - „Jeder Mitarbeiter kennt seinen Vorgesetzten“
- Vorsicht:
  - zirkuläre Referenzen sind möglich
  - maximale Tiefe als Abbruchkriterium festlegen
- Es können direkte Eltern und Shortcuts abgelegt werden (falls kein PK Eindeutigkeit erzwingt)

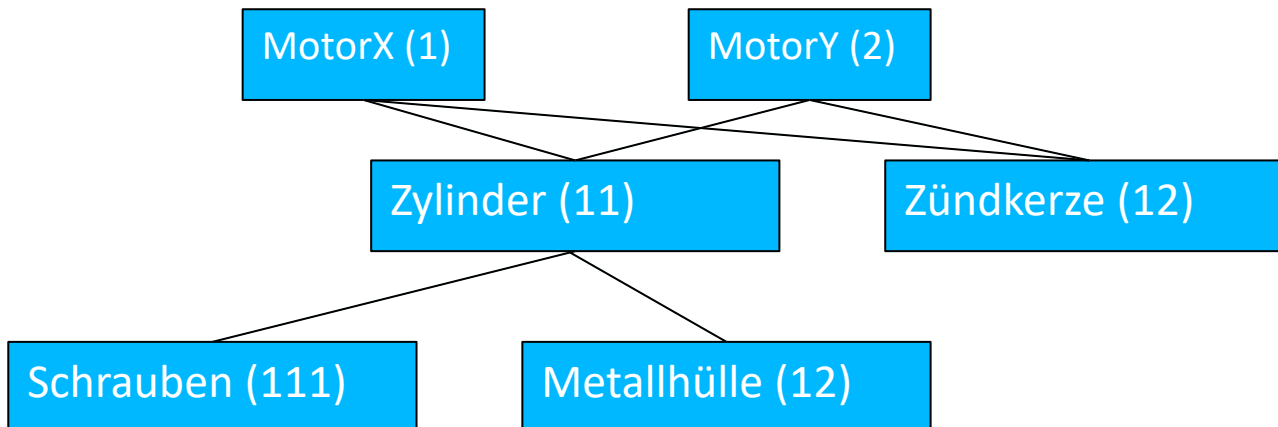


Emp_ID	Name	Manager_ID
1492	The Boss	
777	Boeing	1492
007	James B.	777

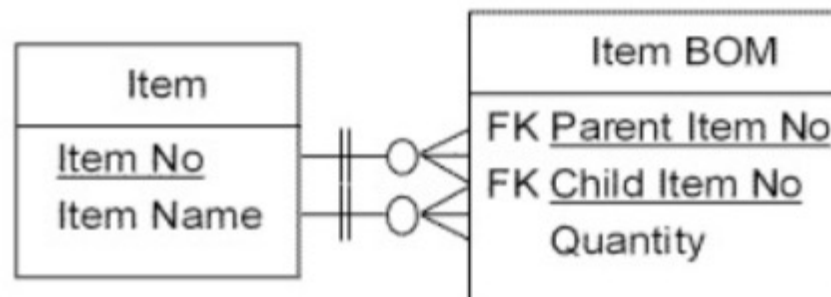
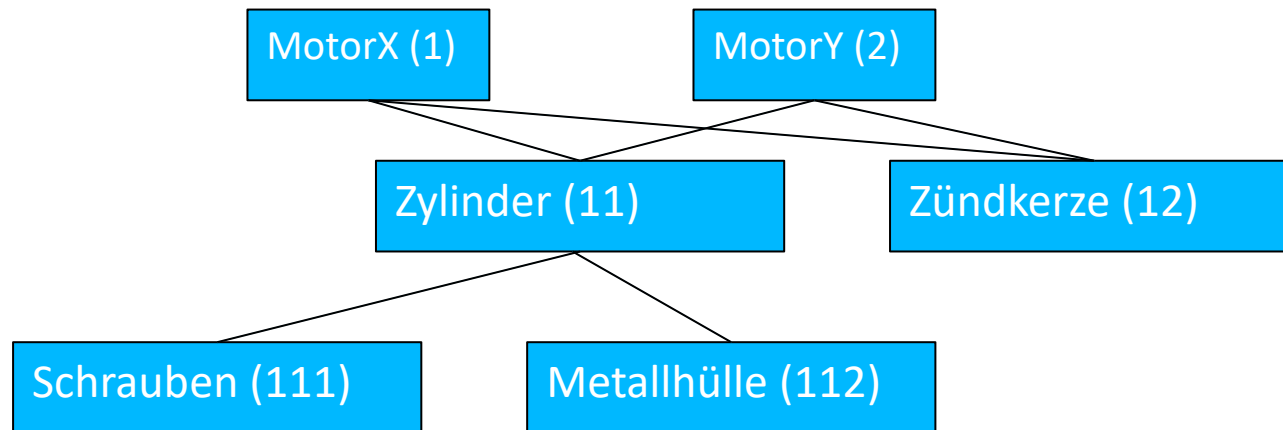


# m:n Hierarchie

- Zwei Strukturen:
  - Entität selbst
  - Beziehungen: ein Gegenstand kann mehrere Eltern/Kinder haben.
  - z.B.: „Bill of materials“ (BOM)
    - Tabelle ist die Beziehungstabelle (associative entity)



# m:n Hierarchie #2



Physische Modellierung?

# Physische Modellierung

- Ansatz zur **Performancesteigerung** und zur **Abfragevereinfachung**:  
**Fixierung**
- Varianten:
  - **Flachstruktur** („flattened structure“)
  - **Schneeflocke** („snowflake“)
- Beide Ansätze sind anwendbar bei
  - Festen Hierarchieebenen
  - Rekursion (nur bedingt)

# Fixierung - Flachstruktur

- Fixierung der Hierarchien ist der **populärste Ansatz** bei der Dimensionsbildung:
  - Die Abfrage braucht **keine rekursiven Elemente** mehr zu verwenden
- z.B. via Flachstruktur:
  - Alle Ebenen werden in einer Tabelle zusammengefasst
  - pro Ebene wird ein Attribut eingeführt

Country	State	City
CH	ZH	Zurich
CH	ZH	Winterthur
BRA	RJ	Rio de Janeiro

# Fixierung - Schneeflocke

- z.B. via Schneeflocke:
  - Jede Ebene wird in einer separaten Tabelle gehalten
  - Die Hierarchie ist also durch die Struktur fixiert

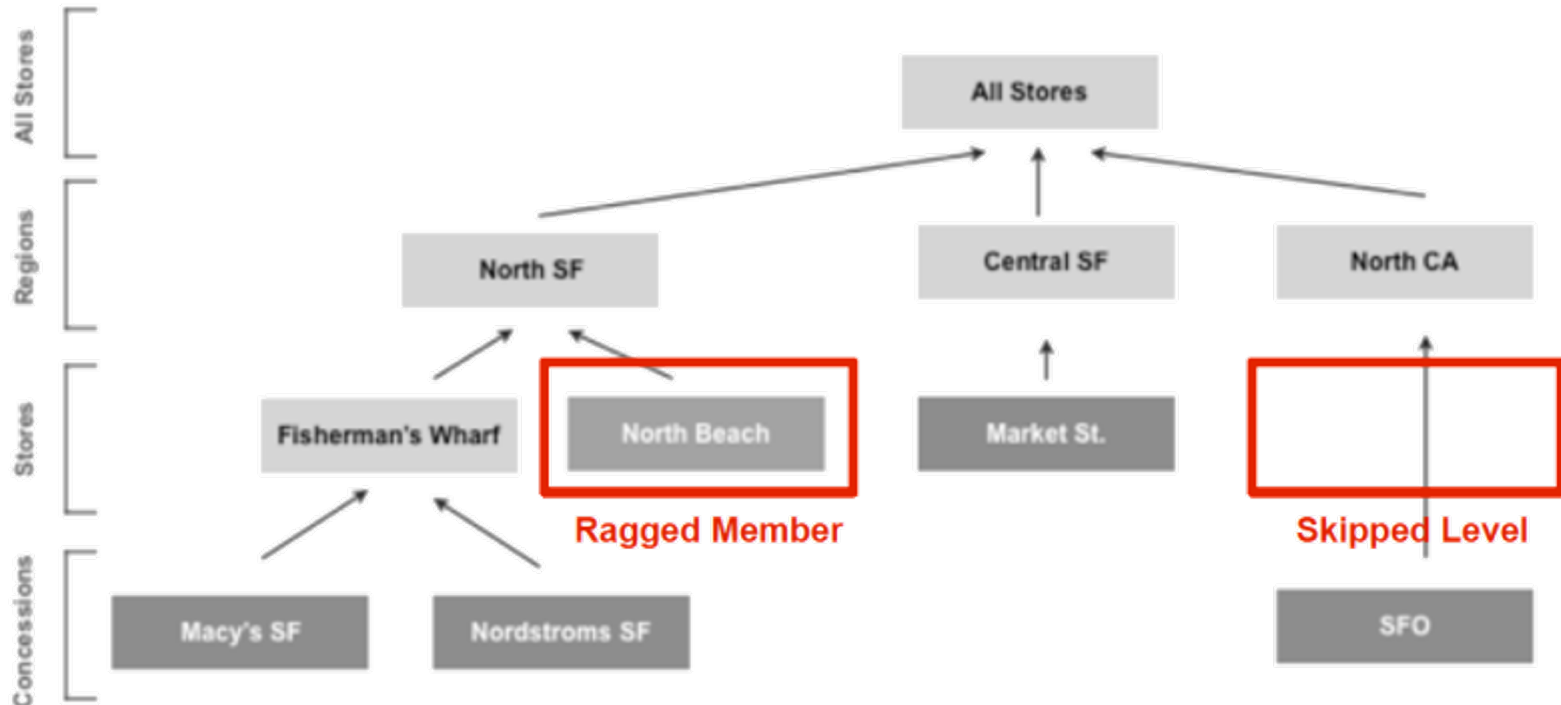
Country_ID	Country	State_ID
1	CH	10
3	BRA	30

State_ID	State	City_ID
10	ZH	100
30	RJ	300
10	ZH	101

City_ID	City
100	Zurich
300	Rio de Janeiro
101	Winterthur

# Unbalancierte Hierarchien

- Hierarchien mit unterschiedlichen Ebenenanzahlen
- Nicht alle Blätter haben denselben Abstand zur Wurzel
  - **Ragged hierarchy** („abgerissen“)
  - **Skip-level hierarchy** („fehlende Ebene“)



# Ragged Hierachy: Physische Modellierung

- Auf der untersten Ebene können **leere Einträge** auftreten (NULL Werte)
- Abfragen müssen berücksichtigen, dass die unterste(n) Ebenen nicht notwendigerweise befüllt sind
- Die meisten BI Tools unterstützen diesen Typ Hierarchie
- Vorteil:
  - Jeder „**Parent**“ **hat** ein direktes „**Child**“
- Nachteil:
  - Im Beispiel ist Washington DC und Vatikanstadt auf einer Ebene mit Kalifornien
  - eine **Konzepthierarchie** liegt also **nicht** korrekt vor (Vatikanstadt ist kein föderativer Staat)

Level 1	Level 2	Level 3
USA	CA	San Francisco
USA	CA	Los Angeles
USA	Washington DC	<NULL>
Vatican City	Vatican City	<NULL>

# Skip-Level Hierachy:

## Physische Modellierung

- Eine skip-level hierarchy liegt vor, wenn
  - das Parent-Element von mindestens einem Element **eine Ebene überspringt**
- Eine skip-level hierarchy **kann balanciert sein**
- Übersprungene Ebenen haben typischerweise NULL-Einträge
- **Konzepthierarchien** können so **korrekt** abgehandelt werden:
  - jede Ebene setzt einen begrifflichen Kontext (siehe Beispiel)

Country	State	City
USA	CA	San Francisco
USA	CA	Los Angeles
USA	<NULL>	Washington DC
Vatican City	<NULL>	Vatican City



# Historisierung

# Warum braucht man Datenhistorisierung?

All Datenänderungen im DWH müssen

- nachvollziehbar sein:
  - e.g. Wann änderte sich der Zinssatz von 1% auf 1.5%?
- audit-fähig sein:
  - e.g. Wer waren die Kundenberater der Person X in den letzten 10 Jahren?
- erklärbar sein:
  - e.g. Warum wurden bestimmte Daten im Q2 2013 angepasst?



# Slowly Changing Dimensions

- **SCD Type 1 (overwrite):**
  - Der ursprüngliche Attributwert wird überschrieben
  - Keine Versionierung, as-is-reporting
- **SCD Type 2 (new record):**
  - Bei Änderungen wird ein neuer Eintrag in der Dimensionstabelle generiert
  - Volle Versionierung, as-was-reporting
- **SCD Type 3 (current value field):**
  - Der neue Wert verdrängt den alten
  - Der ursprüngliche Wert wird in einer speziellen Spalte gespeichert
  - Aufbewahren des Vorgängers (selten verwendet)

# SCD1 - Beispiel

Juni 2004: Maria S. wohnt in San Francisco.

Name	Address
Maria S.	San Francisco

Nov. 2004: Maria S. zieht nach New York City.

Name	Address
Maria S.	New York City

# SCD3 - Beispiel

Juni 2004: Maria S. wohnt in San Francisco.

Name	Address
Maria S.	San Francisco

Nov. 2004: Maria S. zieht nach New York City.

Name	Current Address	Previous Address
Maria S.	New York City	San Francisco

Mai 2011: Maria S. zieht nach Zürich.

Name	Current Address	Previous Address
Maria S.	Zurich	New York City

Wir unterscheiden unterschiedliche Historisierungskonzepte:

- **Monotemporale Historisierung:**
  - Gibt an, wann Ereignisse dem System bekannt sind (**system time**)
  - **ORDER** wann sie in der realen Zeit stattgefunden haben (**business time, validity**)
- **Bitemporale Historisierung:**
  - Gibt an, wann Ereignisse dem System bekannt sind (**system time**)
  - **UND** wann sie in der realen Zeit stattgefunden haben (**business time, validity**)

# Monotemporale Historisierung

- Jeder Datensatz enthält zwei Spalten über die Gültigkeit
- Erfolgt ein Update:
  - neuen Datensatz einfügen
  - Gültigkeit des alten Datensatzes anpassen
- Beispiel: Kunde ist zunächst Privatkunde.

Customer Key	Contract Type	Valid From	Valid Until
A	Privatkunde	01-03-2005	31-12-9999

# Monotemporale Historisierung

- Jeder Datensatz enthält zwei Spalten über die Gültigkeit
- Erfolgt ein Update:
  - neuen Datensatz einfügen
  - Gültigkeit des alten Datensatzes anpassen
- Beispiel: Kunde ist zunächst Privatkunde. Ab 1. Mai 2011 wird er Geschäftskunde.

Customer Key	Contract Type	Valid From	Valid Until
A	Privatkunde	01-03-2005	01-05-2011
A	Geschäftskunde	01-05-2011	31-12-9999



# Bitemporale Historisierung

## Wie behandeln wir folgende Fehlerfälle?

Maria S zieht von New York nach Zürich.

Adressänderung wird falsch ins System eingetragen:

e.g. Zurich, **CA**


anstatt Zurich, **CH**



Bitemporale Historisierung wird benötigt, um zu  
vermerken, wann das DWH über die reale Welt  
bescheid weiss.

# Bitemporale Historisierung

- Juni 2004: Maria S. wohnt in San Francisco. Juli 2004: Daten kommen ins DWH.




Name	Address	Valid from	Valid until	System created	System outdated
Maria S.	San Francisco, CA	2004-06-01	9999-12-31	2004-07-01	9999-12-31

- Nov. 2004: Maria S. zieht nach New York City. Dez. 2004: Daten kommen ins DWH.

Name	Address	Valid from	Valid until	System created	System outdated
Maria S.	San Francisco, CA	?	?	?	?

# Bitemporale Historisierung

- Juni 2004: Maria S. wohnt in San Francisco. Juli 2004: Daten kommen ins DWH.

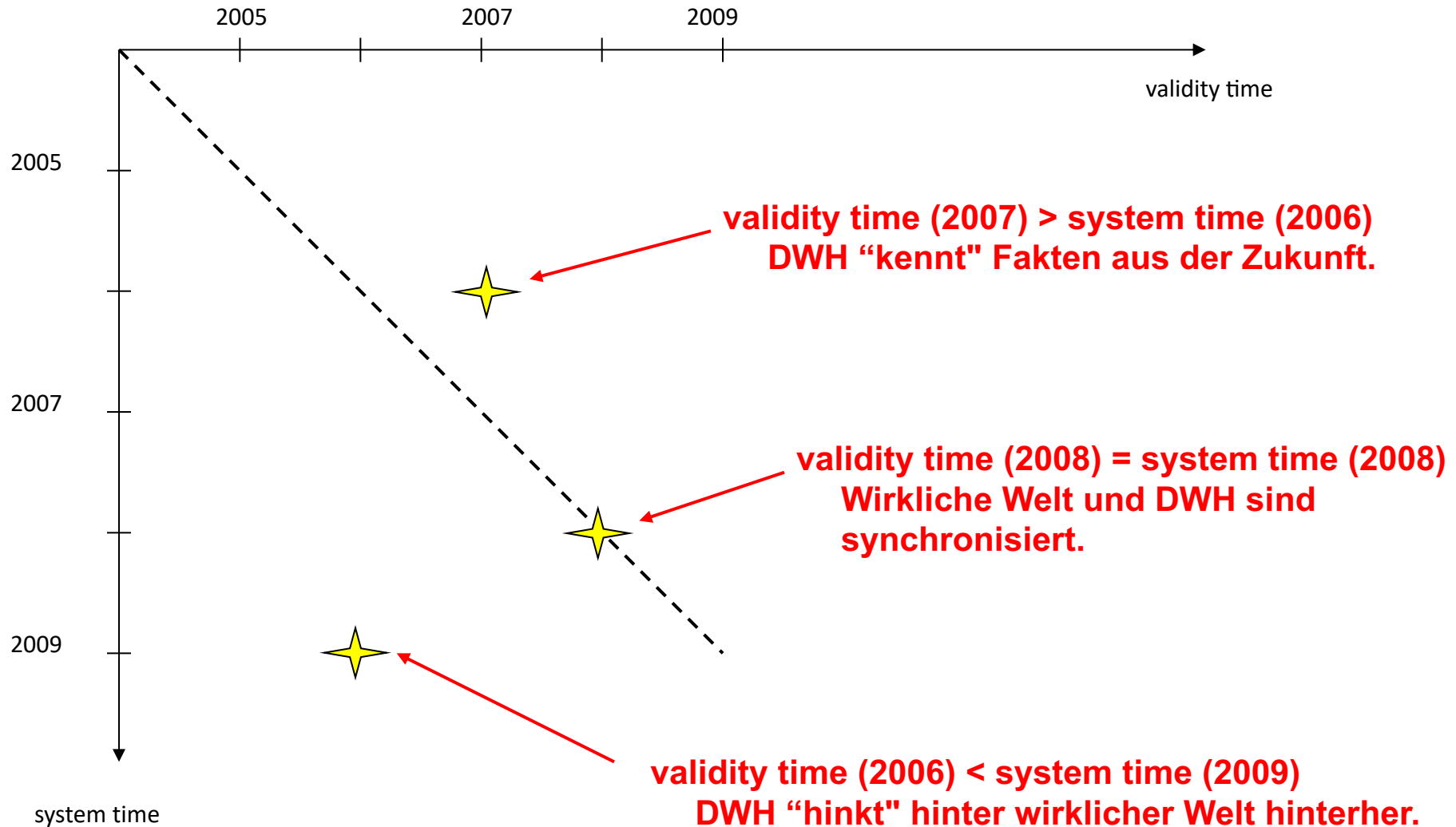


Name	Address	Valid from	Valid until	System created	System outdated
Maria S.	San Francisco, CA	2004-06-01	9999-12-31	2004-07-01	9999-12-31

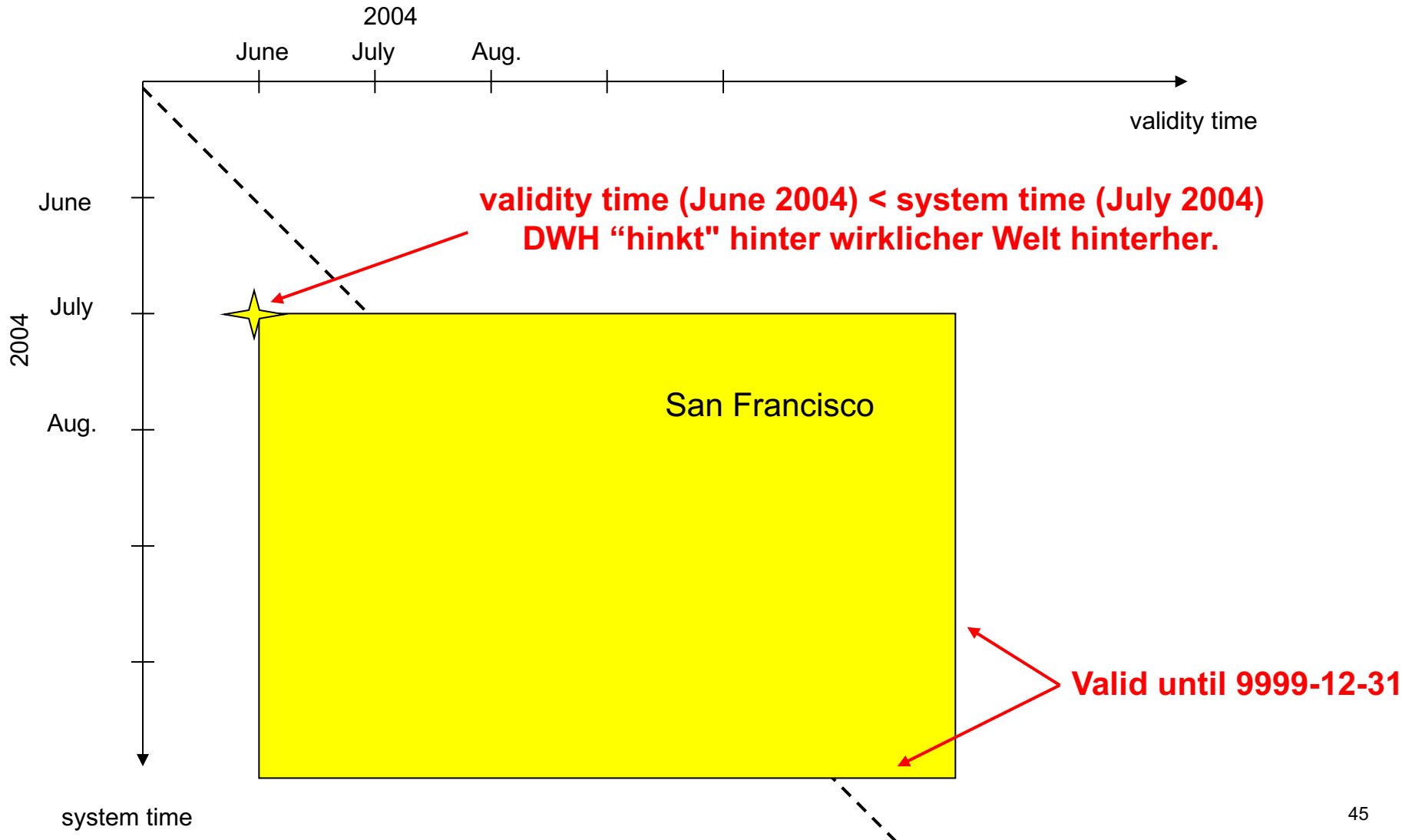
- Nov. 2004: Maria S. zieht nach New York City. Dez. 2004: Daten kommen ins DWH.

Name	Address	Valid from	Valid until	System created	System outdated
Maria S.	San Francisco, CA	2004-06-01	9999-12-31	2004-07-01	2004-12-01
Maria S.	San Francisco, CA	2004-06-01	2004-11-01	2004-12-01	9999-12-31
Maria S.	New York, NY	2004-11-01	9999-12-31	2004-12-01	9999-12-31

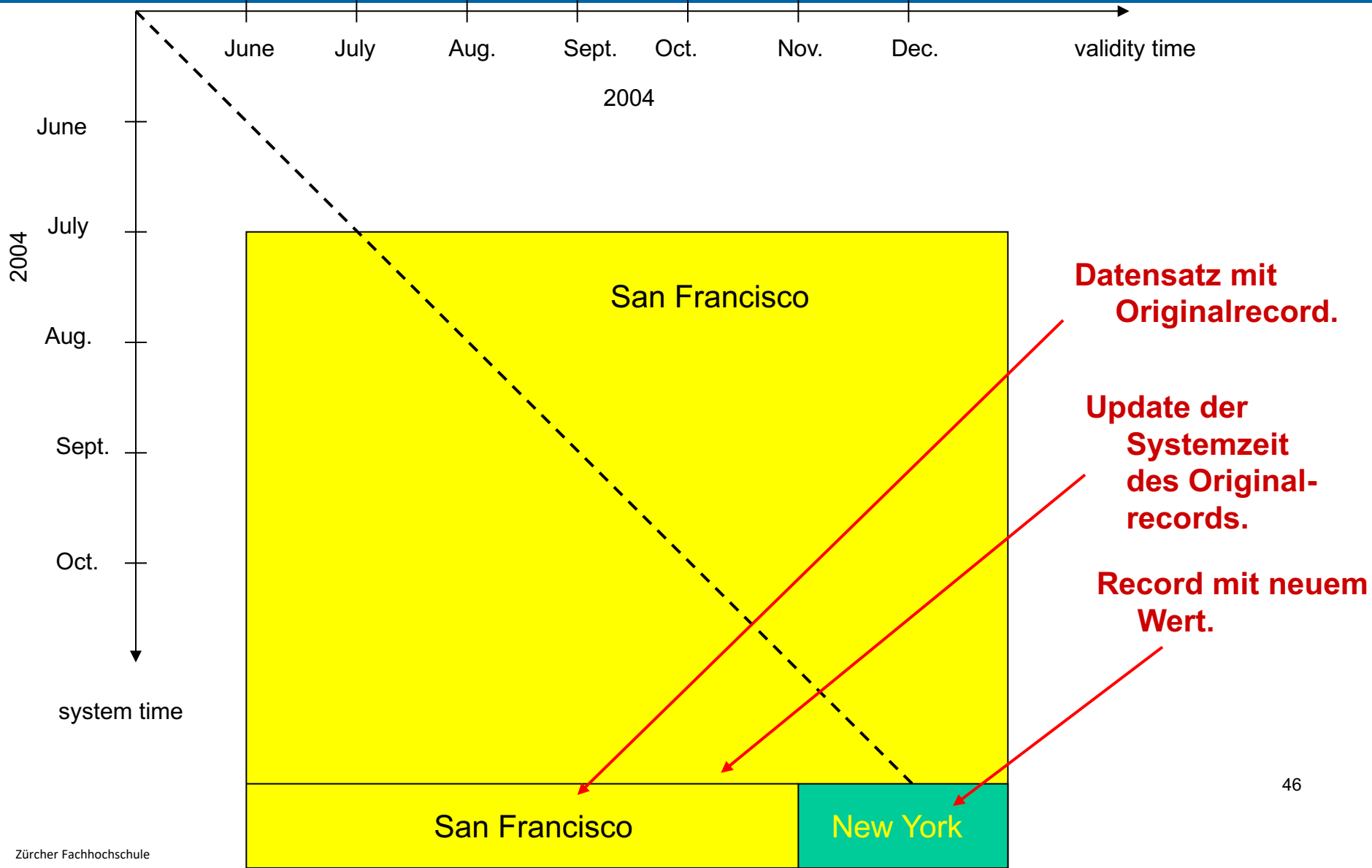
# Graphische Interpretation von 2 Zeitdimensionen



Juni 2004: Maria S. wohnt in San Francisco.  
Juli 2004: Daten kommen ins DWH.

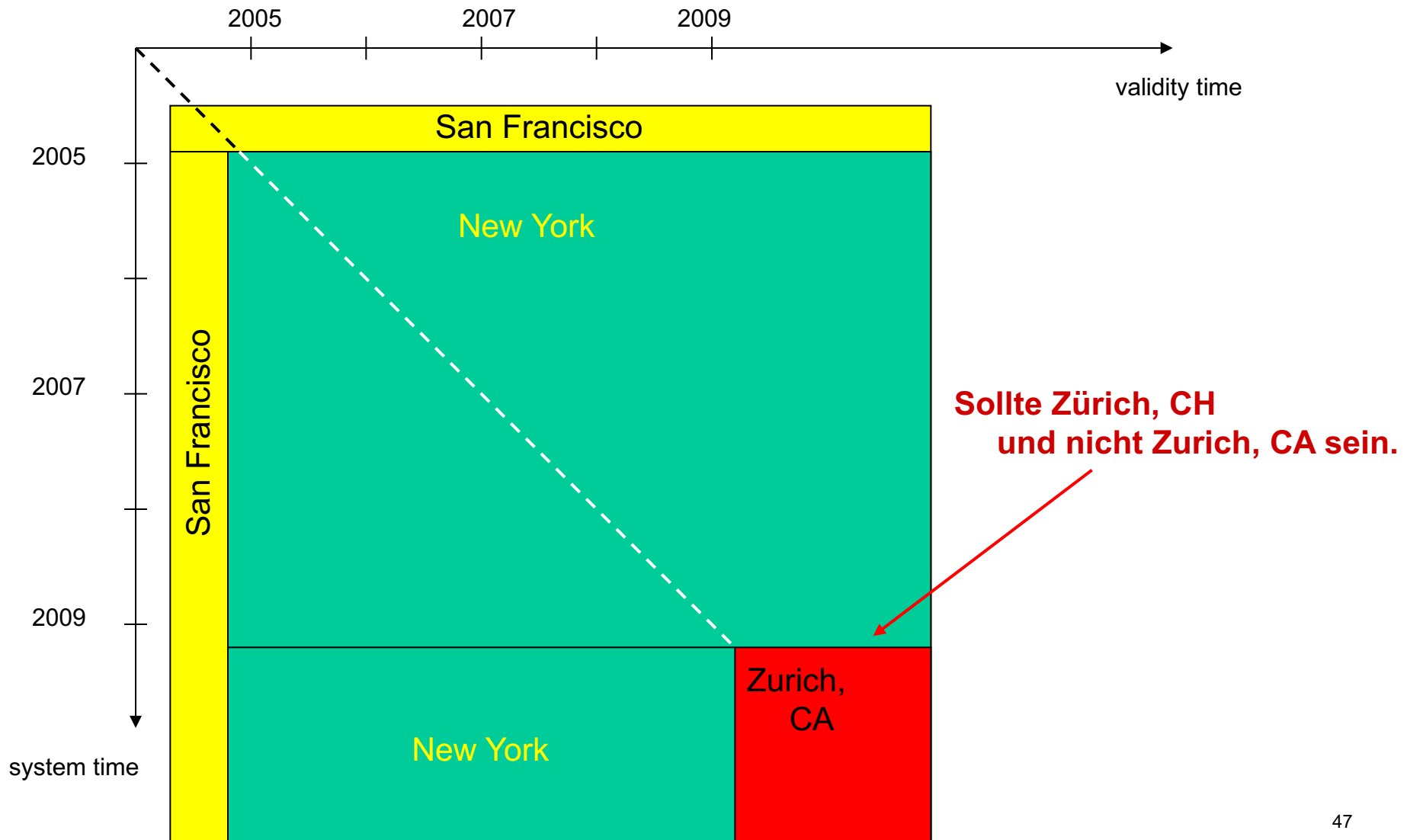


Nov. 2004: Maria S. zieht nach New York City.  
Dez. 2004: Daten kommen ins DWH.

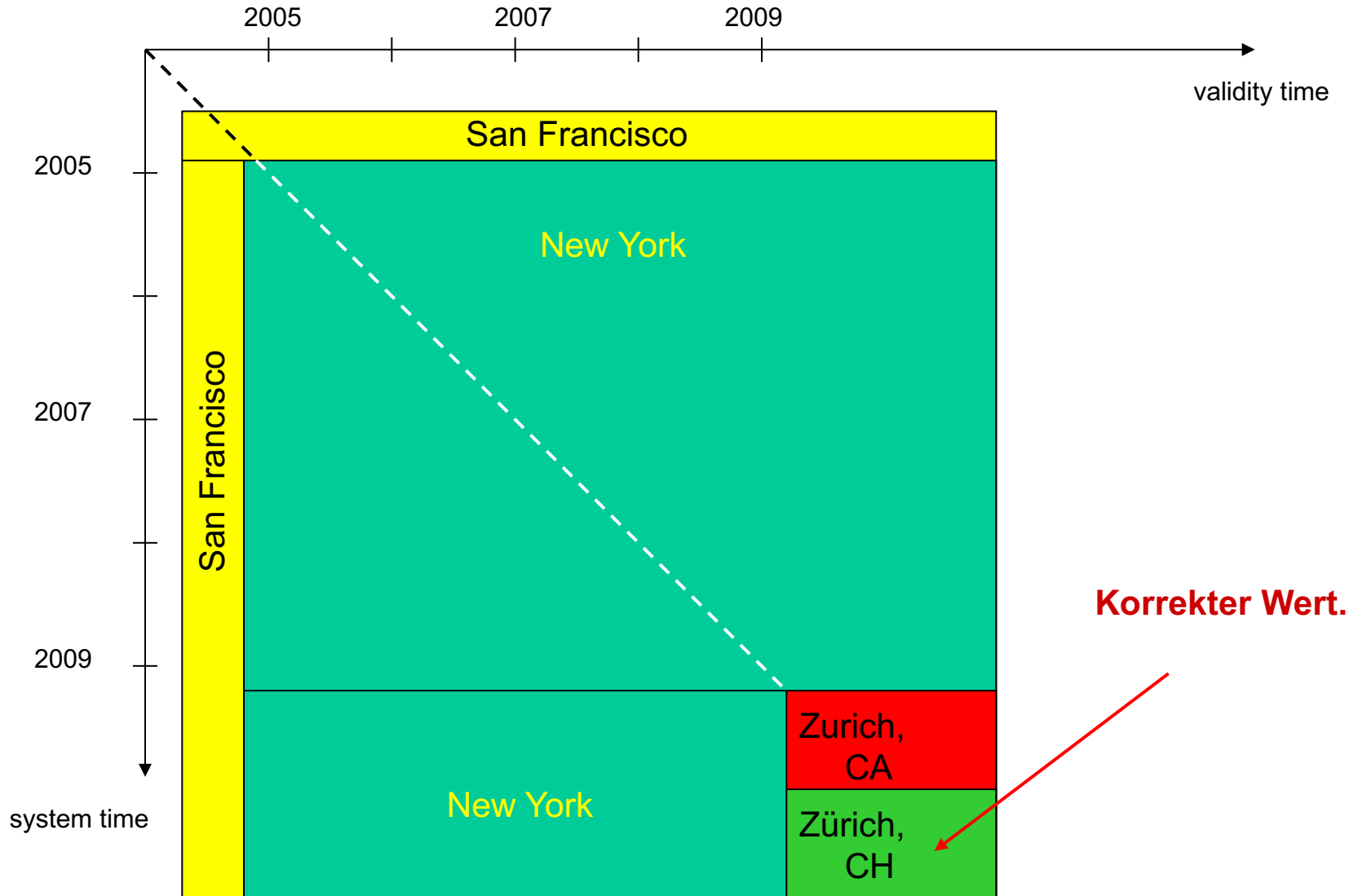


März 2009: Maria S. zieht nach Zürich.

März 2009: DWH Daten werden falsch eingetragen.

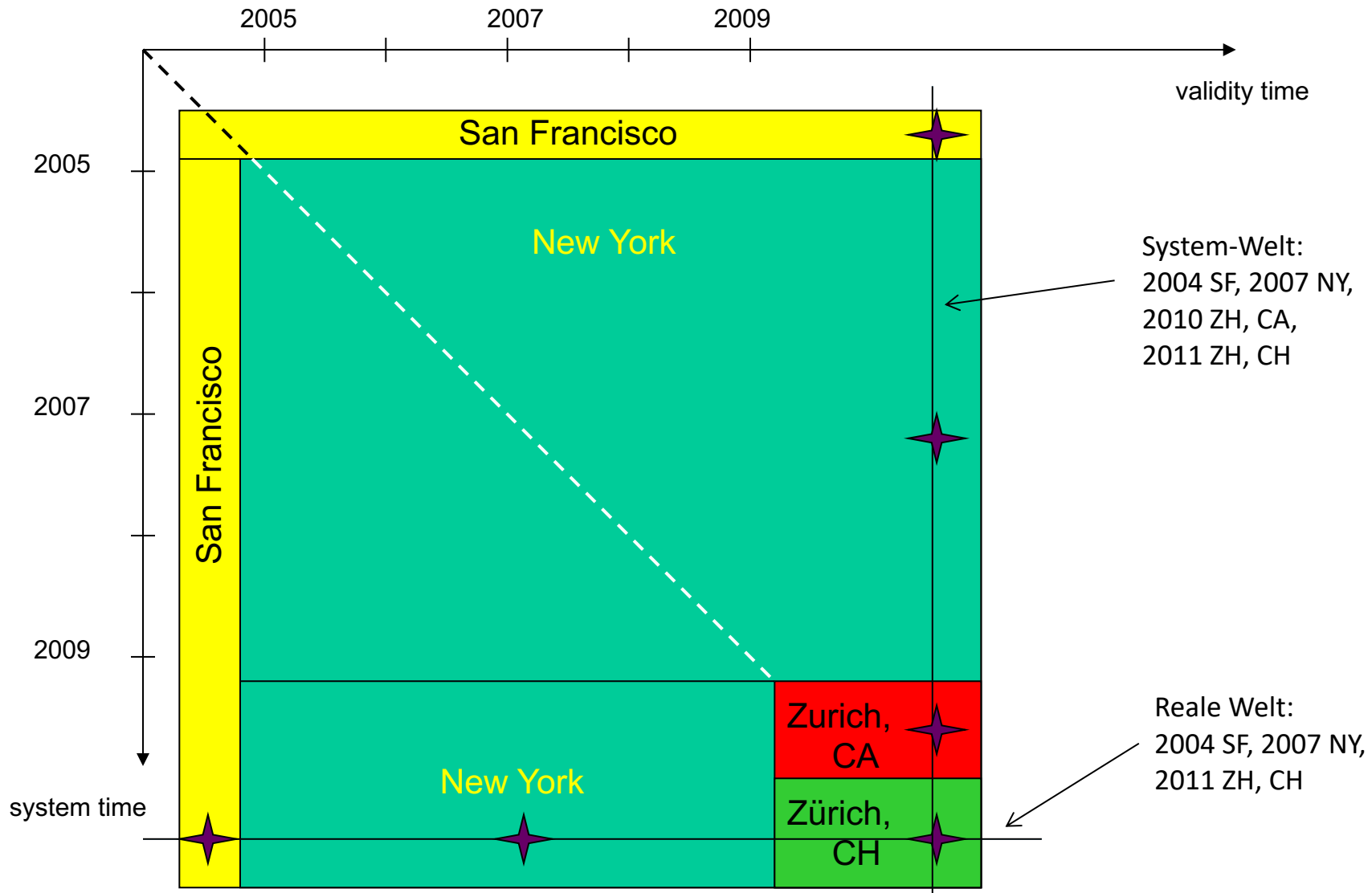


März 2009: Maria S. zieht nach Zürich.  
Jan 2010: DWH Daten werden korrigiert.





# Datenanalyse aus unterschiedlichen Blickpunkten: System-Welt vs. Reale Welt



# Zusammenfassung

- **Faktentabellen** können auf unterschiedliche Weise implementiert werden:
  - **Physische Implementierung** hängt von der Art der gewünschten Abfragen ab
  - Optimierung ohne **Kenntnisse über Abfragen** nur bedingt möglich
- **Dimensionstabellen:**
  - Hierarchien können **logisch und physisch** unterschiedlich modelliert werden
- **Historisierung** ist wichtig für die Nachvollziehbarkeit von Datenänderungen:
  - **Mono-temporale Historisierung:** erfüllt minimale Anforderung
  - **Bi-temporale Historisierung:** erfüllt maximale Anforderung (Umsetzung komplex)