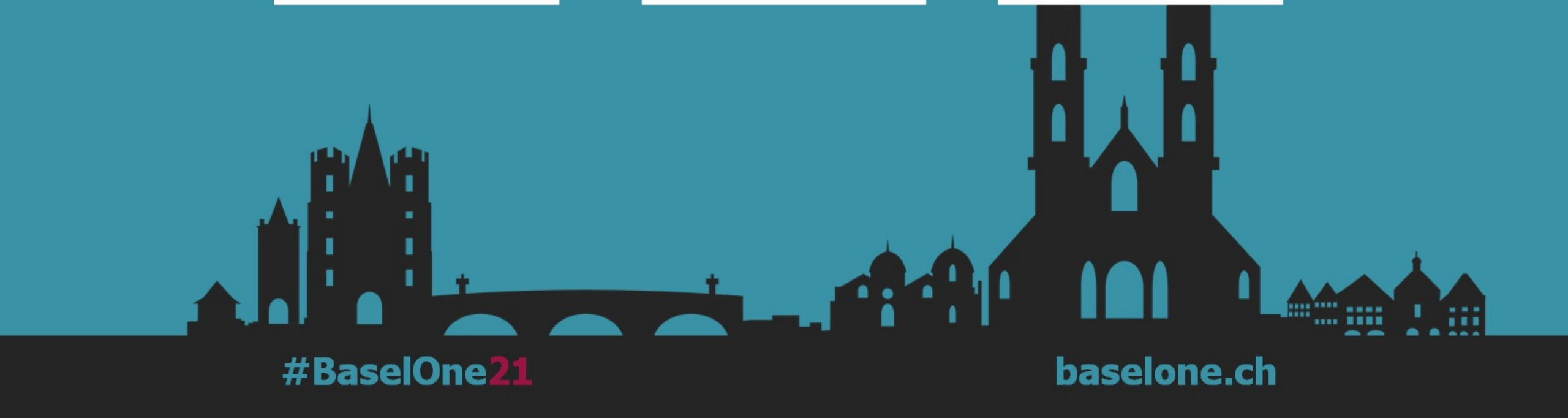
Apache Kafka und GDPR
Freund oder Feind?

Frank Baier, Daniel Grob

Basler Versicherungen

21. Oktober 2021


Basler Versicherungen
bluesky
CONSIDER IT DONE
KARAKUN
sympany versicherungen
magnolia®
JAVA USER GROUP CH
ELCA
We make it work.
Gradle
OPTRAVIS
Transfer Pricing Solutions
#BaselOne21
baselone.ch

Wer sind wir?

Daniel Grob

- Seit 4 Jahren bei der Basler
- Davor bei ELCA, Canoo und edorasware
- Product Owner
- Beteiligt bei Einführung Kafka und GDPR @ Baloise
- Mag alles rund um Sport und Technik

Frank Baier

- Seit 18 Jahren bei der Basler
- Integrationsarchitekt
- Beteiligt bei Einführung Kafka und GDPR @ Baloise
- liebt Musik, Fussball, Mountain Biken, Pokern ...

GDPR

General Data Protection Regulation

Verordnung mit Regeln zur Verarbeitung personenbezogener Daten

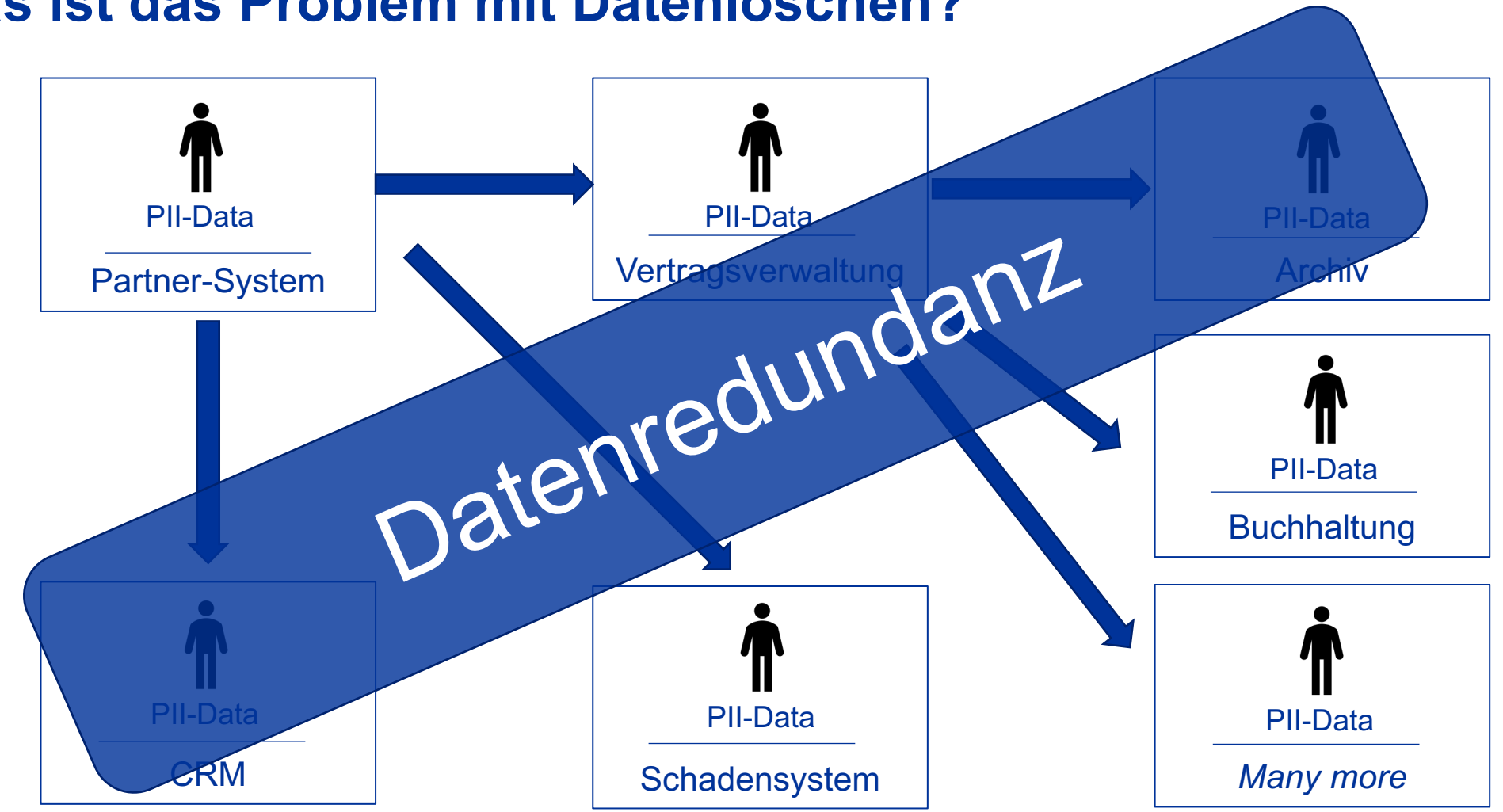
Verpflichtend in der Schweiz

Beinhaltet das Recht auf Vergessenwerden

Daten müssen gelöscht werden, sobald Grundlage für die weitere Verarbeitung entfällt

Dazu gibt es pro Datentyp typische Aufbewahrungsfristen
(z.B. ist diese für Verträge 11 Jahre, für Schäden 31 Jahr)

Was ist das Problem mit Datenlöschen?

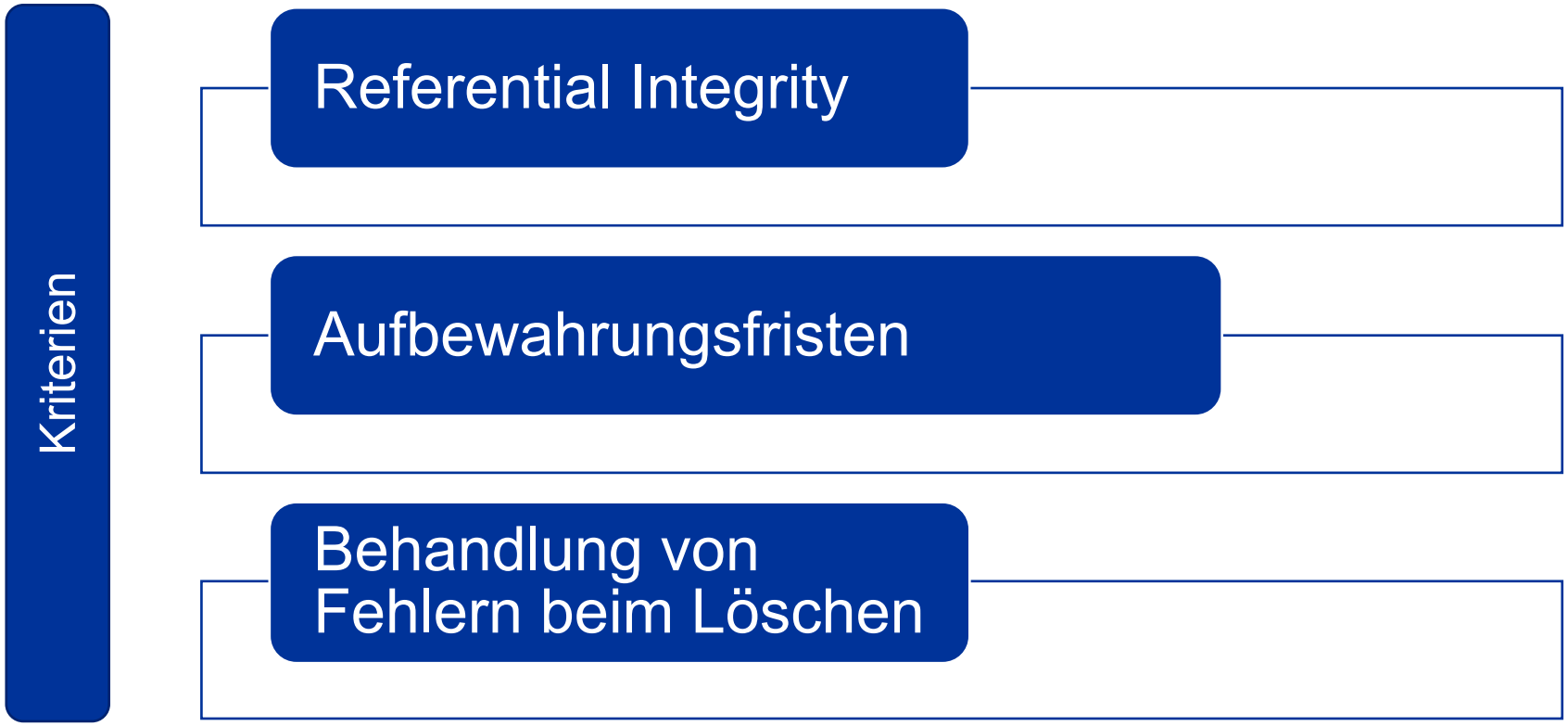


In der Basler: PII-Data in > 100 Systemen verteilt

Datenredundanz

und ihre Folgen für Löschung

- Kann frei gelöscht werden?
- Oder gibt es eine Reihenfolge in der gelöscht werden muss?



Umsetzung Löschen - Anforderungen

Orchestrierung – Reihenfolge festlegen

Löschrückmeldung near real time

Deterministische Löschquittierung

Verschiedene Löschfristen berücksichtigen

Kafka

Grundbegriffe

Kafka stellt **Topics** zur Verfügung

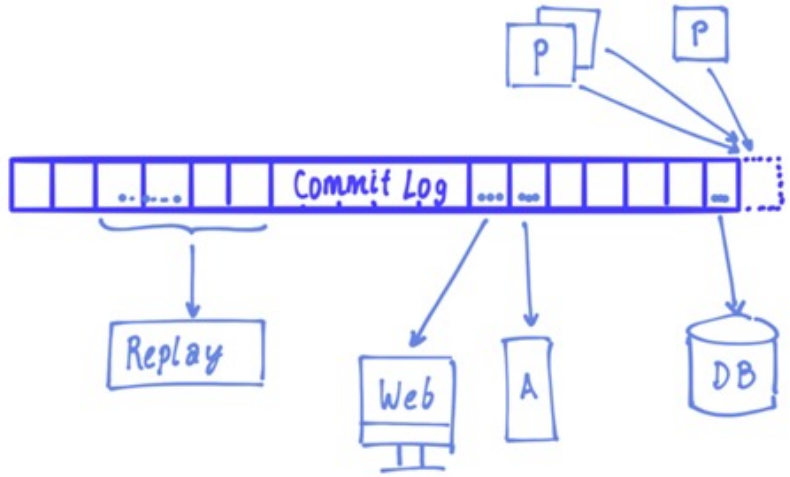
- Kette von Records
- Record hat einen optionalen Key und einen Value
- Achtung: Key identifiziert ein Object und nicht den Record
  => eine Person, einen Vertrag, einen Schaden, …

Producer **senden** Records an ein Topic

- Neue Records werden an das Ende gehängt
- Records können nachträglich nicht geändert werden

Consumer **lesen** Records aus einem Topic

- Können an beliebiger Stelle aufsetzen
- Können beliebig vor- und zurückspulen
- Achtung: Korrekte Reihenfolge ist nur innerhalb desselben Keys garantiert



Löschen mit Crypto-Shredding?

Ansatz

- Provider verschlüsselt die Records
- Consumer entschlüsselt die Records
- Nach Ablauf Aufbewahrungsfrist wird der Schlüssel gelöscht (Shredding)
- Erfordert Einsatz einer Key-Management-Lösung

Einschätzung

- Zu aufwändig und komplex
- Betriebssicherheit gefährdet

Out of scope @ Baloise

Löschen mit Kafka Housekeeping Bordmittel?

Retention Time

Wie lange wird ein Record auf dem Topic vorgehalten?

Nach dieser Zeit **kann** ein Record von Kafka gelöscht werden

Log Compaction

Zu einem Key wird nur der letzte Record vorgehalten

Ältere Records zu einem Key **können** von Kafka gelöscht werden

Retention Time im Kontext Löschung

Retention Time

Lange Durchlaufzeit

10 Systeme in Reihe

jedes System hat 7 Tage retention – gesamte Durchlaufzeit 70 Tage

Gefahr des data respreading

Verkürzung Retention auf z.B. einen Tag

Durchlaufzeit immer noch bei 10 Tagen

SLA Konflikt

Retention Time und der Faktor Wahrscheinlichkeit

Wie wahrscheinlich ist es, dass sich zu löschende PII-Daten bei kurzer Retention im Topic befinden?

Nahezu ausgeschlossen

- Lifecycle
  - Vertragserstellung
  - Vertragsänderung
  - Vertragsauslauf

Wahrscheinlich(er):

- Metaaktionen
  - Stichprobenerhebung

Log Compaction im Kontext Löschen

Log Compaction

KIP-354 nach welcher Zeit soll eine Log-Compaction durchgeführt werden?

Asynchroner Mechanismus

asynch batch run

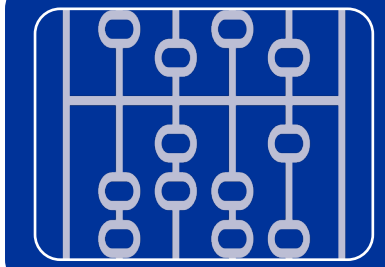
Kick in

Kick in erst nach Empfang einer erneuten Message

Determinismus fraglich

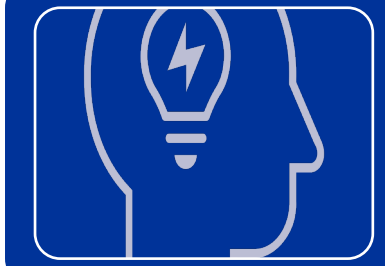
Kafka – Log Compaction

ein paar Hintergründe



Config des Topics

- cleanup.policy: „compact"



Voraussetzung

- Record mit Key
- Key ist bekannt

Tombstone Record

- Key:<yourKey> Value:null

Kafka Log Compaction
in Detail explained

1. Kafka speichert Records in Files - Segmentfiles
2. ein Topic besteht aus einem oder mehreren Segmentfiles
3. ein Segmentfile ist aktiv, alle anderen sind inaktive Segmentfiles
4. Log-Compaction wird nur für inaktive Segmentfiles durchgeführt
5. Log-Compaction auf inaktiven Segmentfiles erfolgt asynchron durch dedizierte Logcleaner-Threads.
6. Kafka erzwingt die Erstellung eines neuen Segmentfiles, wenn
   - max.compaction.lag.ms erreicht ist
   - und ein neuer Datensatz in das Topic geschrieben wird.
     Der Grund dafür ist, dass die Kafka-Implementierung zur Erzwingung eines neuen Segmentfiles nur ausgelöst wird, wenn eine neue Nachricht eintrifft. Es gibt dafür keinen separaten Thread.

Kafka – Log Compaction

Beispiel

Offset	0	1		
Key	K1	K2		
Value	V1	V2		

max.compaction.lag.ms=60000 (1x/min)

cleanup.policy: „compact"

Kafka – Log Compaction

Beispiel

Offset	0	1	2	
Key	K1	K2	K1	
Value	V1	V2	null	

max.compaction.lag.ms=60000 (1x/min)
cleanup.policy: „compact"

Was passiert nach Ablauf einer Minute?

Nichts!

Kafka – Log Compaction

Beispiel

Offset	0	1	2	3	4
Key	K1	K2	K1	K3	K4
Value	V1	V2	null	V3	V4

Roll segment file & LogCleaner Job

Offset	1	3
Key	K2	K3
Value	V2	V3

File-x-1

4
K4
V4

File-x

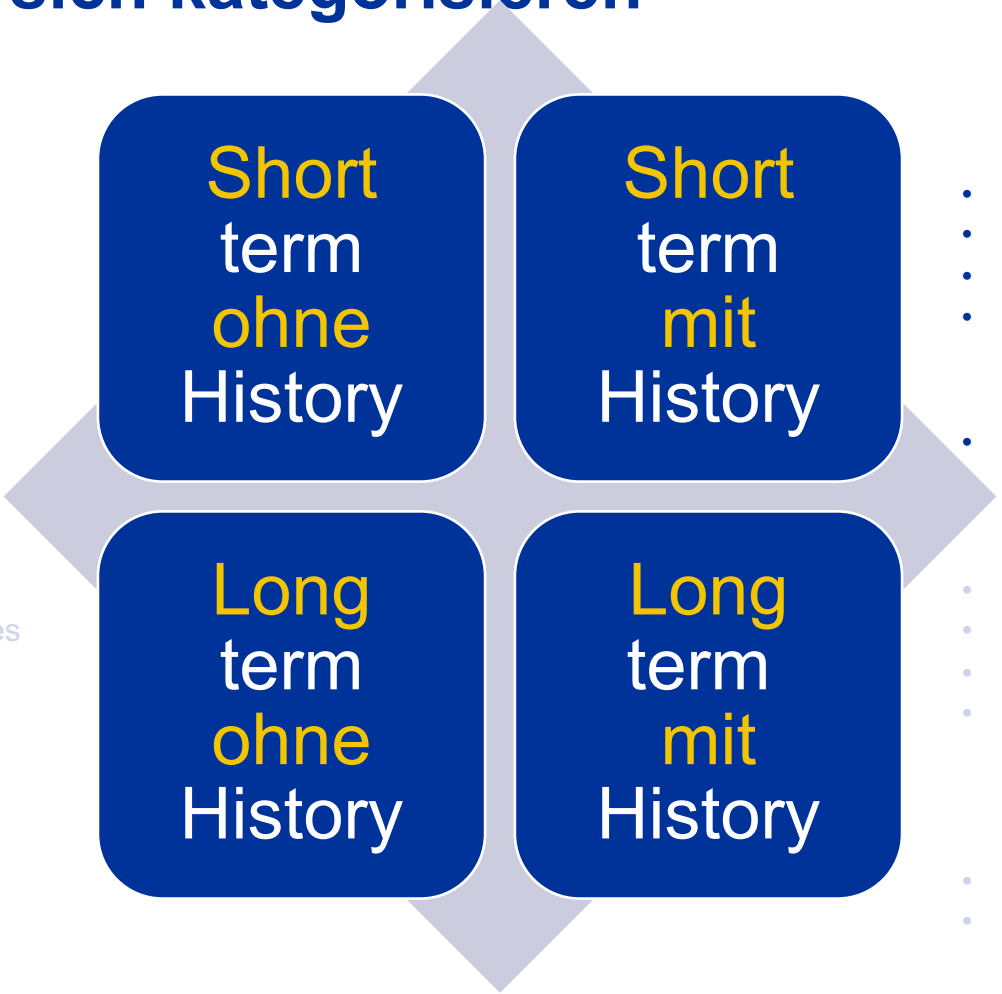
So weit so gut …

eine kleine Zusammenfassung

 Orchestrator Anforderungen	Schnelle Rückmeldung Jedoch erst wenn definitiv gelöscht wurde
 Kafka Realität	Kafka Mechanismen erlauben keinen Determinismus
 Fachliche Realität	Unterschiedliche Anforderungen an die Topic Teilweise durch historische Systemausgangslagen
 Wahrscheinlichkeit	Von nahezu ausgeschlossen bis wahrscheinlich, dass sich zu löschende Daten in einem Topic befinden

Usecases lassen sich kategorisieren

- Eventing
- Verteilen von Vertragsänderungen
- Nur der letzte Stand ist interessant
- Consumer stellen sicher, dass Records innerhalb kurzer Zeit verarbeitet werden (z.B. 14 Tage)
- Verbinden von Systemen



- Eventing
- Verteilen von Vertragsänderungen
- Alle Zwischenstände sind interessant
- Consumer stellen sicher, dass Records innerhalb kurzer Zeit verarbeitet werden (z.B. 14 Tage)
- Anbindung eines Systems ans DWH

- Datenredundanz
- Verteilen des letzten Vertragsstandes
- Nur der letzte Stand ist interessant
- Consumer können zu jeder Zeit dazukommen und alle Records einlesen
- Initial Load beim Verbinden von Systemen

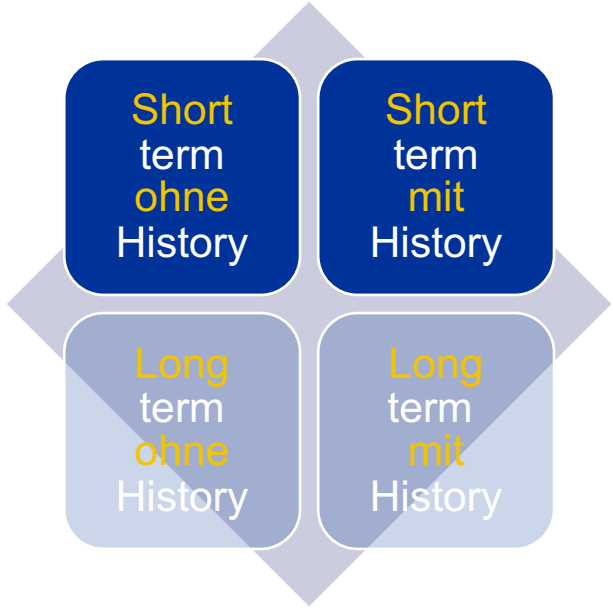
- Datenredundanz
- Verteilen aller Vertragsänderungen
- Alle Zwischenstände sind interessant
- Consumer können zu jeder Zeit dazukommen und alle Zwischenstände aller Records einlesen
- Event Sourcing
- Initial Load beim Anbinden eines Systems ans DWH

Pro Kategorie kommen andere Lösungsoptionen in Betracht

Ansatz 1: Blankolöschrückmeldung

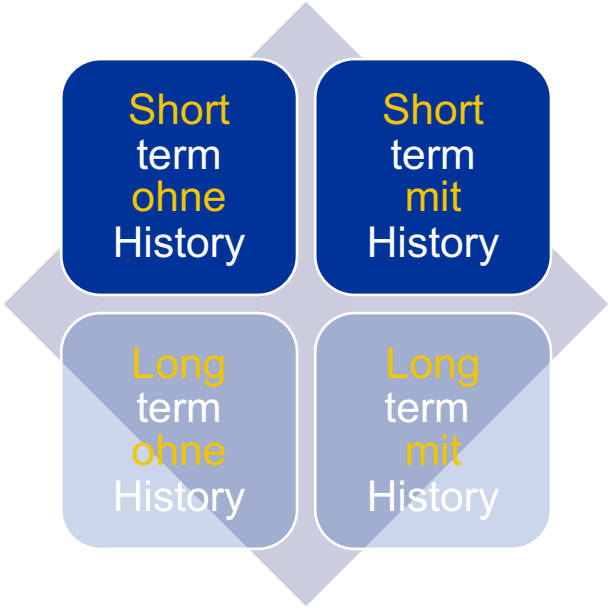
Ohne Aktion: erfolgreiche Löschung zurückmelden

Geeignet, falls es aufgrund der kurzen Retention Time nahezu ausgeschlossen ist, dass ein zu löschender Record im Topic ist



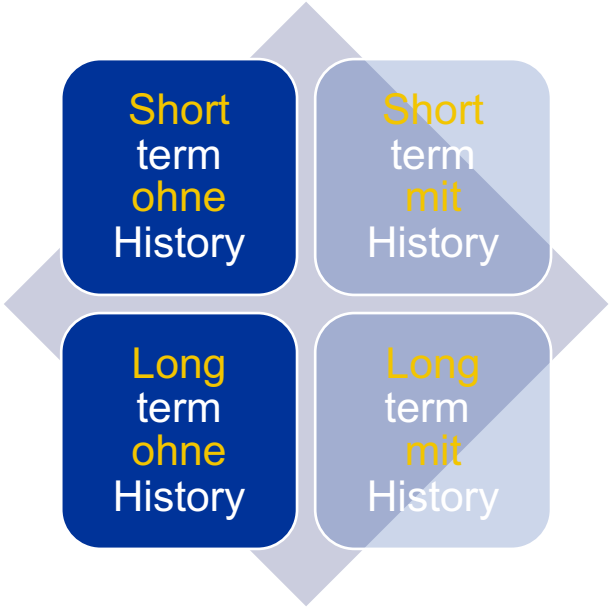
Ansatz 2: Topic durchsuchen

Topic auf zu löschenden Record durchsuchen
Eventuell optimiert durch Index-DB – falls Topic viele Einträge hat
Falls Suche positiv ist, dem Orchestrator zurückmelden, dass Löschen nicht möglich ist
Orchestrator triggert Löschen nach Ablauf der Retention Time nochmals an
Geeignet, falls es aufgrund der kurzen Retention Time sehr unwahrscheinlich ist, dass ein zu löschender Record im Topic ist



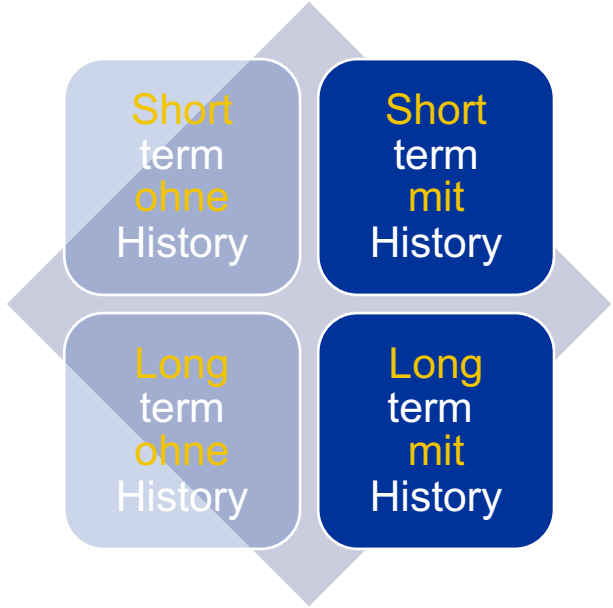
Ansatz 3: Tombstone

Basiert auf dem Compaction Mechanismus von Kafka
Für jeden zu löschenden Record einen Tombstone Record schreiben
max.compaction.lag.ms gering halten
Falls nötig Heartbeat Messages schreiben um inaktive Segment Files und dadurch Compaction zu forcieren



Ansatz 4: Tombstone mit Synthetischem Record Key

Basiert auf dem Compaction Mechanismus von Kafka
Synthetischer Record Key bestehend aus fachlichem Key und Generation definieren
Custom Partioner schreiben, damit Reihenfolge für fachlichen Key garantiert ist
Zu jedem fachlichen Key die zugehörigen synthetischen Record Keys merken
Für zu löschende fachlichen Keys einen Tombstone Record für alle zugehörigen synthetischen Record Keys schreiben
max.compaction.lag.ms gering halten
Falls nötig Heartbeat Messages schreiben um inaktive Segment Files und dadurch Compaction zu forcieren



Fazit

GDPR Umsetzung mit Kafka ist machbar

Bordmittel alleine nur für triviale Fälle ausreichend

Analyse nötig – Lösungen sind vorhanden

Eventual consistency akzeptieren

No best friends – aber genug für ein gemeinsames Bier