Bild: Shutterstock

BI braucht funktionierende Datenschnittstellen

Datenlieferungen an BI

Ein Beitrag von
Michael Müller und
Oliver Cramer

Viele Probleme in der BI haben ihre Wurzel in der Anlieferung der Daten. Häufig wird für diese wichtige Übergabe von Daten nicht einmal mehr das Wort Schnittstelle verwendet. Dabei verursacht eine unzulänglich gestaltete Schnittstelle nicht nur massiv Mehrarbeit, weil nicht gelieferte Daten errechnet oder nachgefordert werden müssen. Vielmehr kann eine solche schlechte Schnittstelle bei systematischen Lücken sogar einen bleibenden Informationsverlust auslösen. In Kombination mit einer Warehouse-Automation-Lösung kann sich das Problem noch verstärken. Darüber hinaus gibt es wenig Standardliteratur zur Gestaltung von Schnittstellen und meist noch viel weniger den Willen, Energie und Geld in die Entwicklung von Schnittstellen zur BI zu stecken. In diesem Artikel werden Anforderungen an eine Schnittstelle definiert, die verschiedenen Formen von Schnittstellen vorgestellt und deren Auswirkungen auf das Laden der Daten in BI und die Behandlung von Löschungen dargestellt.

Der folgende Dialog ist oft der Einstieg zur Einführung einer vernünftigen Schnittstellenvereinbarung zwischen operativen Systemen und dem Data Warehouse:

„Warum braucht ihr in BI so lange?“

„Wir mussten [hier komplexes Thema einfügen] berechnen. Das war sehr schwierig.“

„Aber das haben wir doch schon einmal umgesetzt. Warum habt ihr nicht einfach die Daten übernommen?“

„Die Daten waren nicht persistiert und es hieß, wir müssen uns die Daten direkt aus der Datenbank laden. Niemand von [Quellsystem] hat Zeit, etwas für uns zu schreiben, da alle dringend [hier das komplexe Thema von oben einfügen] umsetzen müssen.“

[Betretenes Schweigen]

Ein sehr normaler Dialog: Doppelarbeit aufgrund von nicht persistierten Daten. Dabei ist egal, welche Logik nicht persistiert wurde. Neben der initialen Implementierung sind auch alle Änderungen dauerhaft doppelt auszuführen, da sonst die BI andere Zahlen hat als das operative System. Hinterher wird allen Beteiligten klar, warum man eine Schnittstelle braucht – die entstandenen Kosten als (zu) späte Entscheidungshilfe. In der Folge entstehen mit der Umstellung weitere Kosten.

Aus Sicht der BI muss eine solche Schnittstelle mindestens folgende Anforderungen erfüllen:

1. Komplette Lieferung der Daten, um diese vollständig und historisch speichern zu können.
2. Die Schnittstelle muss robust sein, das heißt, die Daten werden zuverlässig geliefert und Änderungen an der Schnittstelle werden frühzeitig gemeldet, sodass die Ladeprozeduren rechtzeitig angepasst werden können.

Im Grunde gibt es 4 verschiedene Schnittstellentypen, die im Folgenden erläutert werden.

Kompletter Abzug der Daten(-bank)

Alle gespeicherten Daten der Quellsysteme werden für die BI exportiert beziehungsweise auf die BI-Datenbank repliziert:

- Vollständig bis auf die eingangs erwähnten, nicht persistierten Daten. Gelöschte Daten werden über den Vergleich mit der letzten Lieferung ermittelt.
- Im Falle mehrfacher Änderungen von Daten zwischen zwei Lieferungen steht jeweils nur die letzte Version zur Verfügung.
- Es entsteht wenig Aufwand bei der Implementierung.
- Robust, wenn rechtzeitig über Änderungen im Datenmodell informiert wird.

Jeden Tag alle Daten zu exportieren kann sehr zeitaufwendig sein. Zudem fallen oft sehr große Datenmengen an. Es müssen immer mindestens zwei Lieferungen vorgehalten werden. Die Ermittlung der Löschungen erfordert bei Schema-Änderungen im Quellsystem oft manuelle Nacharbeiten.

Change Data Capture (CDC)

Vor allem wegen der hohen Datenmengen, aber auch wegen langer Exportdauer wurde das Verfahren Change Data Capture (CDC) entwickelt. CDC ermittelt die Änderungen pro Satz und übermittelt nur diese Änderungen. Jeder Satz wird mit einem Insert für neue Sätze, einem Update für geänderte Sätze und einem Delete für gelöschte Sätze versehen.

- Vollständig bis auf die eingangs erwähnten, nicht persistierten Daten.
- Es gibt gute Werkzeuge, die dies zuverlässig für das Quellsystem erledigen und keine negativen Auswirkungen auf die Performance des operativen Systems haben.
- Robust, wenn rechtzeitig über Änderungen im Datenmodell informiert wird.

Eine elegante Lösung, die aber Löschungen immer nur bezogen auf einen kompletten Datensatz meldet. Das setzt voraus, dass die Daten im operativen System auch in 3NF abgelegt sind. Ansonsten kann es zu Situationen kommen, bei denen nicht klar ist, auf was sich das Löschkennzeichen bezieht, oder ein Update ist eigentlich eine Löschung (mehr zum Umgang mit diesen Problemen unten im Text).

Implementierte Schnittstelle (komplett oder Delta)

Bei einer implementierten Schnittstelle übernimmt das operative System die komplette Verantwortung für die Vollständigkeit der Lieferungen und die Robustheit der Schnittstelle. BI wird zu einem angeschlossenen System, dem Daten in vereinbarter Form übergeben werden. Eine sehr gute Lösung im Sinne einer klaren Trennung von Aufgaben – das operative System verantwortet die Erfassung und Pflege der Daten und stellt diese bereit.

Diese Schnittstelle ist sehr anspruchsvoll. Die Daten müssen komplett übergeben werden, mit allen Beziehungen untereinander. Sind diese Daten beispielsweise in JSON gespeichert, reicht es meist nicht, diese nur bereitzustellen. Häufig müssen gewisse Beziehungen, die im Programmcode hinterlegt sind (zum Beispiel zwischen zwei JSON-Files), explizit mit exportiert werden. Dasselbe gilt für Konsistenzprüfungen, die zwischen Sätzen gemacht werden. Haben diese Sätze keinen gemeinsamen Schlüssel, muss die Konsistenzlogik aus dem operativen System nachgebildet werden. All das muss die Implementierung der Schnittstelle abbilden.

Im Falle einer Delta-Lieferung muss die Löschung jeweils auf der kleinsten Granularität hinterlegt sein, da sonst – analog zur CDC-Schnittstelle – ein Update auch eine Löschung sein kann oder es zu Mehrdeutigkeiten in Bezug auf die Löschung kommen kann.

Event-Schnittstelle

Häufig sind die operativen Systeme in einer Service-Oriented Architecture (SOA) oder an einem Enterprise Service Bus (ESB) angebunden. Hier werden zwischen den Systemen Nachrichten über Events ausgetauscht. Mit derselben Technik lassen sich auch BI-Systeme anbinden. Der große Unterschied zwischen einem normalen Service und BI liegt in der Frage: „Push oder Pull?“ In diesen Architekturen werden Services angeboten, die andere Services im Pull-Verfahren nutzen: Wenn Sie die Information brauchen, rufen Sie den Service auf. BI ist passiv, hier müssen die Events im Push-Verfahren gesendet werden: BI weiß nicht, was interessant ist, die Benachrichtigung muss aktiv erfolgen. Ansonsten ist ein ganzer Pull-Adapter für BI zu entwickeln, der alle Services regelmäßig nach Änderungen abfragt. Das ist dann eine Bulk-Abfra-

MICHAEL MÜLLER arbeitet seit 20 Jahren im Bereich BI. Sein Fokus liegt auf Datenmodellierung und Architekturen sowie auf Data Vault, DWH-Automatisierung und Einbettung der BI in das Unternehmen. Er ist im Vorstand der Deutschsprachigen Data Vault User Group (DDVUG) e.V.
E-Mail: michael@m2data.de

OLIVER CRAMER ist langjähriger Experte zum Thema Data-Vault-Modellierung und Data Warehouse Automation in Deutschland. Er ist spezialisiert auf Daten-/Informationsmodellierung, Metadaten und modellgetriebene Automation sowie zeitliche Aspekte bei Informationssystemen. Er hat unter anderem Zertifizierungen als Data Vault 2.0 Practioner und Anchor Modeler Version 2014. Er hält Vorträge und gibt Schulungen sowie Workshops zum Thema Data Vault und Data Warehouse Automation.
E-Mail: oliver.cramer@dwh42.de

ge, die wiederum zu Performance-Problemen im Messaging-System führen kann.

Die Event-Schnittstelle ist eine implementierte Schnittstelle, das heißt, alle Aussagen zu implementierten Schnittstellen (siehe oben) gelten auch hier. Zusätzlich bietet die Event-Schnittstelle die Chance, Transaktionen bei der Entstehung in (fast) Realtime aufzunehmen und dann auch zu berichten. Es muss nicht alles persistiert werden, wenn es parallel zur Ausgabe am Bildschirm auch an die BI geht. Desto wichtiger ist es, genau die Vollständigkeit der Schnittstelle zu betrachten.

Diese vier Typen lassen sich beliebig tief betrachten

Diese Betrachtungen lassen sich noch weiter vertiefen, indem man neben Flatfile (Fixed Format oder CSV) weitere Datenformate wie XML oder JSON betrachtet. Oder indem man gesonderte Szenarien wie Internet of Things (IOT) betrachtet. Die Gesamtheit der möglichen Architekturen ist umfangreich und durch die eingehende Betrachtung der vorhandenen Lösungen und deren Architekturen gilt es, eine passende Lösung auszuwählen.

Was beinhaltet nun „vollständig“?

Die Schnittstellenart bestimmt sich häufig direkt aus den Gegebenheiten der vorhandenen IT oder aus den Performance-Anforderungen der operativen Systeme. Es kann sehr gut sein, dass eine einfache und pragmatische Lösung gewählt wird. Mitunter gibt es aber Konstellationen, bei denen nicht genug Informationen vorhanden sind, um die geforderte Vollständigkeit abzudecken. Insbesondere bei nicht normalisierten Schnittstellen ist eine genaue Prüfung wichtig. Hinzu kommt die Frage, bis wann die Daten gültig sind beziehungsweise wann Daten gelöscht wurden.

Arten von Löschungen

Welche Arten von Löschungen gibt es?

- **Logische Löschung:** Nichts wird aus der Datenbank entfernt, es gibt Attribute, die einen Satz als gelöscht kennzeichnen.
- **Physische Löschung:** Daten werden aus der Datenbank entfernt:
  - entweder als Entfernen logisch gelöschter Daten (Housekeeping)
  - oder als tatsächliches Löschen
- **Storno:** Ein Sonderfall, bei dem der Datensatz im Nachhinein für ungültig erklärt wird.

Warum sind Löschungen so wichtig?

BI speichert Daten historisch [Inm96]. Ein Datensatz, der nicht gelöscht wird, kann bis in alle Ewigkeiten ausgewertet werden. Ein Satz, der auf magische Art im operativen System verschwindet, kann auf ewig in den Zahlen der BI verbleiben.

Nicht immer sind alle Sätze relevant für eine Kennzahl. Muss man eine bestehende Schnittstelle auf Vollständigkeit prüfen, hilft es zunächst, auf die Art der Daten zu schauen:

- **Status-Daten** entstehen zu einem fixen Zeitpunkt und ändern sich nicht – hierunter fallen vor allem technische Daten wie Messwerte, Finanztransaktionen, Bestände usw.
- **Historische Daten** sind Daten, die sich im Zeitablauf ändern können, auch wenn sie eigentlich sehr fix erscheinen, hierunter fallen alle Bewegungsdaten (Aufträge, Lieferungen, Rechnungen) im Unternehmen, aber auch einige Stammdaten wie Geburtsdatum oder Geschlecht.

Relevant für Kennzahlen sind aber vor allem die Bewegungsdaten. Betrachtet man nun die Vollständigkeit, sollte man von den Bewegungsdaten ausgehen. Kann ich ermessen, wann ein Eintrag in den Bewegungsdaten nicht mehr gültig ist?

An die Kennzahlen werden die Dimensionsdaten geknüpft. Veraltete Stammdaten werden also ohne Kennzahl nie ausgewertet werden. Dennoch sind die Beziehungen zwischen Kennzahl und Dimension zu prüfen: Was, wenn einzelne Beziehungen nicht mehr gültig sind?

Generell ist eine korrekte Lieferung der Löschungen zu bevorzugen. Denn die Betrachtungen, ob sich eine Nicht-Lieferung auf die Zahlen auswirkt, ist immer eine Momentaufnahme. Bei jeder neuen Auswertung muss diese Prüfung wieder durchgeführt werden. Wird beispielsweise eine Auswertung von Stammdaten gewünscht, ist dies ohne Änderungen nicht möglich.

Gefahr von denormalisierten Daten

Und genau hier treten die Probleme mit denormalisierten Schnittstellen auf:

Eine Schnittstelle liefert in einem Satz alle Daten zum Kunden: CDC-Kennzeichen, Kundennummer, Name, Geschlecht, Geburtsdatum, Adresse, Nummer der Kundenkarte. Zur Kundenkarte werden weitere Daten von einem externen Dienstleister geliefert. Tabelle 1 zeigt, welche Daten geliefert werden.

Aufgrund dieser Lieferung gibt es zwei gültige Kundennummern für einen Kunden. Darf ein Kunde zwei Kundenkarten haben?

Datum	CDC	Kundennr	Name	Geschlecht	Geburtsdatum	Adresse	Kundenkarte
2.1.2019	I	23457	Otto Malus	m	30.9.1978	Nürnberg	B27Z23
3.2.2019	U	23457	Otto Malus	d	30.9.1975	Nürnberg	X23B27
6.2.2019	U	23457	Otto Malus	m	30.9.1975	Fürth	X23B27

Tab. 1: Daten im Zeitverlauf mit CDC-Kodierung

Diese Frage ergibt sich nicht aus der Schnittstelle. Sie lässt sich nur aus dem liefernden logischen Datenmodell beantworten. Wenn nur immer eine Kundenkarte für einen Kunden gültig ist, dann ist dies beim Laden der Daten sicherzustellen.

Die Bewertung einer Schnittstelle kann somit nicht allein aus der Struktur der Schnittstelle erfolgen. Auch das zugrunde liegende Datenmodell muss mit einbezogen werden. Nach dieser Prüfung stellt sich die Frage: Kann die fehlende Logik dennoch abgebildet werden?

Im Beispiel der Kundenkarte muss beim Laden des Kunden immer sichergestellt werden, dass die bereits bestehende Beziehung zwischen Kunde und Kundenkarte abgeschlossen wird, bevor die neue Kundenkarte eingetragen wird.

Diese Prüfung erzeugt ein tiefes Verständnis für die Daten und wird desto wichtiger, wenn Lösungen im Einsatz sind, die Daten automatisch in eine Core-Warehouse-Struktur übernehmen. Im Bereich von Data Vault gibt es hier zumindest einen Namen für dieses Phänomen: Driving Key Szenario [Vos19]. Es tritt nur auf, wenn Fremdschlüssel über eine Deltalieferung mit CDC-Kennzeichen geliefert werden. Das Driving Key Szenario muss explizit umgesetzt werden, damit die Lösung korrekt funktioniert. Die gute Nachricht ist: Es gibt für die meisten Schwierigkeiten gute Lösungen – es ist nur wichtig, dass man weiß, wann diese einzusetzen sind.

Wie muss eine Schnittstellenvereinbarung aussehen?

Nach der Prüfung, ob die Daten bei der gewählten Schnittstellenart auch vollständig geliefert werden können, muss eine Vereinbarung mit dem liefernden System getroffen werden. Der Umfang einer solchen Vereinbarung oder Service Level Agreement (SLA) schließt die zu liefernden Daten, die Häufigkeit der Lieferung und Vereinbarungen zur Qualitätskontrolle mit ein. Generell muss ein Weg vereinbart werden, wie Änderungen im Datenmodell des operativen Systems mitgeteilt werden. Die gleichzeitige Lieferung von Metadaten ermöglicht eine automatisierte Prüfung auf Schema-Änderungen. Damit erweitern sich die Anforderungen an eine Schnittstelle um:

3. Metadaten werden mitgeliefert, um eine Möglichkeit zur Überprüfung zu haben und maschinell auf Veränderungen reagieren zu können.

Die nicht funktionalen Anforderungen für diese Schnittstelle bezüglich Performance, Sicherheit und Skalierbarkeit sind zu formulieren. Final ist noch darauf zu achten, dass die Schnittstelle einfach umzusetzen sein sollte. Die Erfahrung zeigt, dass komplexe Lösungen schnell nach unten priorisiert werden und so die Umsetzung der Schnittstelle – auf Seiten des operativen Verfahrens – immer wieder hinter den Anforderungen an die Weiterentwicklung zurücksteht. Eine schnelle Umsetzung erhöht die Chancen auf eine konsequente Umsetzung.

Auf dieser Basis lässt sich eine gute, technisch orientierte Schnittstelle bauen. Die Verantwortung liegt fast komplett bei der BI. Der Grund hierfür ist meist die Trennung der Weiterentwicklung von operativen Systemen und dispositiven Auswertungen. Eine neue Anforderung an das operative System wird gestellt, und erst kurz vor beziehungsweise nach der Umsetzung wird die entsprechende Anforderung an die BI formuliert.

Wenn die Anforderungen an operativ und dispositiv gemeinsam entstehen, entfällt diese Trennung und eine Änderung ist erst dann vollständig, wenn auch die Auswertung an die neue Situation angepasst ist. Wird in diesem Vorgehen ein logisches Modell der Unternehmensdaten verwendet, kann die Entwicklung der Schnittstelle vereinfacht werden. Langwierige Prüfungen auf Vollständigkeit erübrigen sich. Das logische Modell wird zur Schnittstelle und die Lieferung der Daten in genau dieser Form liegt dann in der Verantwortung des operativen Systems. Die BI nimmt die Daten dann entgegen und bereitet diese „nur noch“ auf. Die Prüfung über die korrekte Bereitstellung liegt auf Seiten der operativen Systeme. Das entstehende Modell ist fachlicher und damit meist resilienter.

Eine detailliertere Darstellung der Schnittstellenvereinbarungen und der Besonderheiten einer Schnittstelle auf Basis des logischen Datenmodells sprengt den Rahmen dieses Artikels. Das Thema wird zu einem späteren Zeitpunkt fortgeführt.

Fazit

Schnittstellen verlangen Sorgfalt und sind häufig von externen Faktoren beeinflusst. Die verschiedenen Schnittstellenarten haben jeweils eigene Stärken und Schwächen. Es gibt nicht die ideale Lösung, nur Lösungen, die in einer bestimmten Umgebung optimal sind.

Unabhängig von der Schnittstellenart gilt es zu prüfen, ob mit der gewählten Lösung auch wirklich die Daten komplett abgebildet sind und sie somit auch korrekt ausgewertet werden können. Hierzu braucht es ein Verständnis des Datenmodells und der Auswertemöglichkeiten, um sich gegen künftige Anforderungen abzusichern. Insbesondere beim Einsatz von Data-Warehouse-Automatisierung ist es wichtig, die Schnittstelle auf Besonderheiten zu prüfen, um Schwierigkeiten frühzeitig zu vermeiden.

Eine Schnittstellenvereinbarung sichert die Bedürfnisse und die Berichtsfähigkeit der BI. Gleichzeitig muss sie eine schnelle Reaktionsfähigkeit aufrechterhalten. Blockaden sind unbedingt zu vermeiden.

Literatur

[Inm96] Inmon, W.: Building the Data Warehouse. John Wiley & Sons 1996, S. 33

[Vos19] Vos, R.: Driving Keys and relationship history, one or more tables? http://roelantvos.com/blog/driving-keys-and-relationship-history-one-or-more-tables/, abgerufen am 25.9.2019