Voll-Integrierte DG
(Aspekt 1)

Data Governance

Thilo Riegel, München

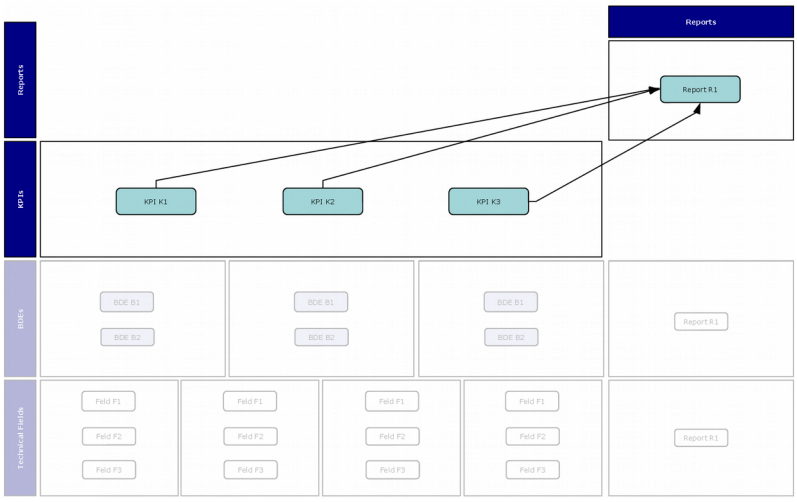
Zusammenfassung

Data-Governance-Serie, Teil 6.1: Wir haben in uns den letzten Teilen mit verschiedenen Dimensionen der DG befasst: der Informations-Dimensions, der horizontalen, der vertikalen sowie mit dem DQ-Management. In diesem Teil sehen wir uns an, wie die geschickte Kombination diese Dimensionen zu einem effektiven und ganzheitlichen Lösungsansatz für einige der Probleme führt, die wir im ersten Teil aufgezählt haben.

Integration der Dimensionen

Horizontal und vertikal „schnurren“ zusammen

Erinnern Sie sich kurz an eine Bemerkung aus einem der früheren Teile – dort hatten wir folgendes Bild, das merkwürdig anmutete:



Was haben wir denn jetzt da? Ist das nun eine vertikale oder eine horizontale Beziehung? Da stimmt doch was nicht im Schema? Auf den ersten Blick ist das so, aber das ist nur scheinbar ein Widerspruch.

Wir können es so erklären: Das Top-Management weiß nichts / braucht nichts zu wissen von den einzelnen Verarbeitungs-Schritten, somit werden diese auf dieser Ebene auch nicht dargestellt bzw. sie schnurren zusammen in einen einzigen Verarbeitungs-Schritt (und somit verschmelzen auf der höchsten Ebene auch horizontale und vertikale Dimension). Außerdem dient es hier, in diesem Arti-

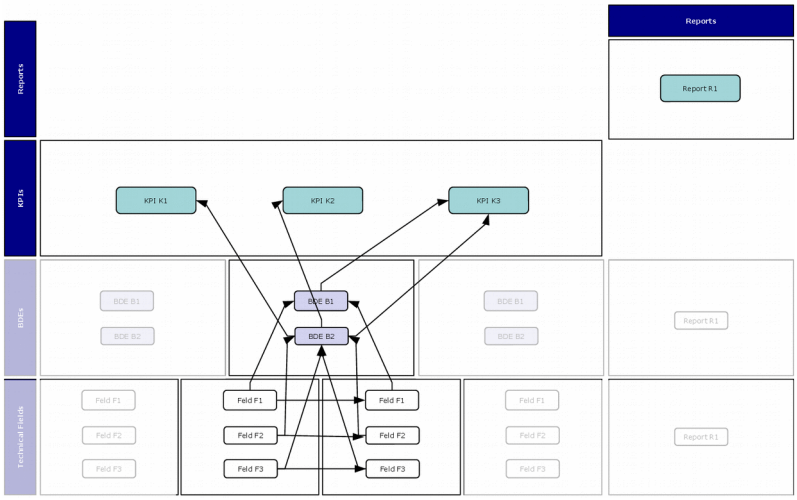
kel, der vereinfachten Darstellung; ein bisschen differenzierter (und somit korrekt über zwei Dimensionen) geht es dann in der Praxis doch.

Allgemeiner: Detaillierungsgrad nimmt nach unten zu

Besonders effektiv und „schön“ wird die Kombination von horizontaler und vertikaler Dimension für den Aspekt des Navigierens durch das Modell. Man kann im Prinzip von jedem Element aus frei nach oben, unten, links und rechts navigieren und sich Informationen über Zusammenhänge holen, weil ja alles *explizit* miteinander verknüpft ist.

Z.B. erlaubt es einem, von einem KPI (oben) über die BDEs (Mitte) zu den entsprechenden technischen Feldern (unten) zu gelangen, wobei man ggf. auch die Elemente auf den verschiedenen Stages betrachten kann. Mit jedem Schritt nach unten werden die Informationen technischer, detaillierter und reichhaltiger. Umgekehrt kann man auch durch Heraufnavigieren herausfinden, zu welchen KPIs / Reports dieses und jedes technische Feld direkt beiträgt.

Die folgende Abbildung verdeutlicht das schematisch.



Die von mir geforderte sofortige Auskunftsfähigkeit im Detail des Data Governors wird auf diese Weise (und ich behaupte: *nur* auf diese Weise) möglich. Ein Data Governor, der kein Metadaten-Repository hat, mit der er in der hier skizzierten Art durch seine Meta-Daten navigieren kann, kann seinen Job nicht vollständig ausüben.

These

Weiteres zu BDEs

An dieser Stelle müssen wir auch eine Klarstellung nachholen, die wir in einem der vorherigen Teile absichtlich ausgelassen haben. BDEs fallen typischerweise in drei Haupt-Kategorien:

- *Staging BDE*: i.W. eine redundante Kopie eines Felds in einem Quellsystem.
- *Core oder Vault BDE*: Teil eines konsolidierten, normalisierten und redundanzfreien, fachlichen Datenmodells, das zur Datenpflege und für fachliche Berechnungen optimiert ist. Das sind tpyischerweise Snowflake-Schemas.
- *Mart BDE*: Kopien oder Projektionen von Core/Vault BDEs. Sie gehen in die Reports sein, und hier bestehen auch wesentliche Redundanzen. Das sind typischerweise Star-Schemas oder Cubes. Auf diese BDEs werden wir später noch einmal zurück kommen.

Eine häufige Frage in diesem Zusammenhang ist dann: Sind diese BDEs gleich den DWH-Fakten? Nein, aber sie sind, sagen wir, eng verwandt. Etwas vereinfacht gesagt, ist ein Fakt das low-level-technische Repräsentierung eines BDE, das (typischerweise direkt) in ein oder mehrere KPIs eingeht. Oder, etwas weniger schwülstig ausgedrückt: Eine Zeitreihe transaktionaler Geschäftszahlen, die zur Unternehmens(bereichs)-Steuerung aggregiert werden. Also: Längst nicht jedes technische Feld ist ein Fakt, aber jeder Fakt ist ein technisches Feld.

Mit dieser Klarstellung und Unterteilung können wir unterschiedliche Teile des Gesamt-Repositories mit seinen verschiedenen Unter-Repositories sinnvoll nach Projekt-Streams und typischen Anforderungs-Profilen unterteilen:

- Um die Modellierung der *Staging-BDEs* kümmern sich Analysten bzw. die Verantwortlichen der Vorsysteme,
- um die *Core BDEs* kümmern sich „Hardcore"-Analysten, und
- um die *Mart-BDEs* kümmern sich eher die Cube- und Reporting-Spezialisten.

Jede Gruppe hat ihre spezifischen Bedürfnisse. Durch diese Trennung (die ja keineswegs akademisch-künstlich ist, sondern sich im richtigen Leben auch „natürlich" ergibt) ist es einerseits möglich, organisatorisch zu entzerren, und andererseits kann die Projektleitung per Auswertung[1] feststellen, ob wirklich alles abgedeckt ist – denn wir verwenden das Repository ja nicht nur für Ist-Stände (Dokumentation), sondern auch für Soll-Stände (Plan) und entspr. Soll-Ist-Vergleich.

These

Projekt-Controlling einerseits und Linienarbeit für das Governance-Team andererseits sind einem BI/DWH-Projekt ist mit einem BDE-basierten Ansatz, wie hier beschrieben, viel zeitnaher, zuverlässiger und tiefergehend möglich als mit herkömmlichen Methoden. Das wird in naher Zukunft auch gar nicht mehr anders möglich sein.

Fachliche Funktionen

Komplexe fachliche Funktionen (die typischerweise auf dem Kernmodell der BDEs ausgeführt werden) müssen nicht unbedingt im Detail in die Abhängigkeiten aufgenommen werden (obwohl ich das befürworte), aber wenigstens müssen sie grob dokumentiert sein, wie oben angedeutet: „BDE A hängt ab von BDE C und B" wäre also die grobe Aussage (die zur Projekt-Steuerung und Governance u.U. ausreicht), während detailliertere Vorgaben („... und zwar im Detail wie folgt: nimm B und addiere...") in einem Zwischenschritt ausgelagert werden können.

Ich schlage allerdings vor, dies nur auf der oberen Ebene (Entitäten und KPIs) zu tun und nicht mehr auf der BDE-Ebene. Warum? Weil man ansonsten durch den Medienbruch garantiert Inkonsistenzen hat, und diese bemerkt man leider sehr spät. Andererseits darf man es mit der Detaillierung auch nicht übertreiben, sonst ist man quasi schon auf der technischen Ebene und macht die Arbeit der Programmierer doppelt.

These

Wie beim fachlichen BDE-Modell selbst, so ist es auch für die Low-Level-Fachfunktionen im Repository (also derjenigen, die BDEs als Ein- und Ausgangsgrößen nehmen) nötig, eine angemessene Balance zu finden zwischen „zu grob" und „zu detailliert / fast schon eine zweite Implementierung". Bei allem Ehrgeiz, den man in die Dokumentations-Qualität legen kann: Man kann es auch übertreiben. Schließlich sollen die Entwickler mit dem Übergang von der fachlichen BDE- auf die technische Ebene auch noch ein bisschen was zu Denken haben. Gute Governance zeichnet sich u.a. dadurch aus, dass sie diese Balance findet!

1 Über die expliziten Verbindungen, die in 90% der Fälle auch ziemlich einfach und eindeutig sind.