Ressort Technologie

Orientierungsleitfaden 1.0

SAP HANA
SQL DWH

Oktober 2020

DSAG

Zielsetzung des Leitfadens

Das Ziel des Leitfadens ist es, dem Leser eine Orientierung zum SAP HANA SQL Data-Warehouse (DWH) zu geben. Dazu wird zunächst definiert, was unter einem SQL DWH mit SAP HANA zu verstehen ist und wie es in der DWH-Strategie neben dem SAP BW und der DWC von SAP einzuordnen ist. Da eine DWH-Umsetzung mit SAP HANA sehr flexibel ist, möchten wir eine Einführung geben und auf die wichtigsten Aspekte aus unserer Sicht eingehen. Demzufolge ist dieser Orientierungsleitfaden keine erschöpfende Anleitung zur Implementierung eines SAP HANA SQL DWH.

Wir wollen dem Leser aufzeigen, welche Komponenten zu einem SAP HANA SQL DWH zählen und wie DWH-Architekturen aussehen können. Anhand von drei Beispielen soll gezeigt werden, wie Unternehmen ein solches DWH bei sich einsetzen und betreiben. Dabei gehen die Unternehmen auch auf Aspekte bei der Einführung des DWH ein und teilen ihre Erkenntnisse.

Der Orientierungsleitfaden soll den Mehrwert und die Potenziale eines SAP HANA SQL DWH vermitteln und somit dem Leser eine Alternative zu anderen DWH-Lösungen aufzeigen. Dabei wird aber keine Empfehlung gegeben, sich für ein Produkt aus dem DWH-Portfolio von SAP zu entscheiden.

Zielgruppe

Der Orientierungsleitfaden wendet sich an alle, die sich in Anwenderunternehmen und IT-Systemhäusern mit Data-Warehousing auseinandersetzen.

Vorwort

In der heutigen Zeit nehmen die Anzahl und die Art der Datenquellen sowie die Frequenz der Änderungen in der IT-Landschaft kontinuierlich zu. Getrieben von immer neuen Anforderungen durch das Management und durch sich stets weiterentwickelnde Analysewerkzeuge und -methoden wird eine dynamische und offene Datengrundlage fast Voraussetzung für jedes Unternehmen, um die Masse an Daten nicht in einem Sumpf aus Daten zu verlieren. Um diesem Bedarf zu entsprechen, reicht es nicht aus, ein zentrales Data-Warehouse als Monolith zu entwickeln, das dann über Jahre Bestand hat und kaum erweitert wird. Es wird vielmehr immer wichtiger, unternehmensweit eine offene Datenplattform zu schaffen, die es erlaubt, dass die Konsumenten eigenständig, ohne großes spezifisches Vorwissen, Daten auswerten: Self-Service-BI. Damit eine solche Datenplattform auch mit Leben gefüllt werden kann, sind offene Schnittstellen für die Datenbeladung, den Datenzugriff sowie die Verwaltung und Entwicklung notwendig. Nur auf diese Weise ist es möglich, flexibel und schnell auf die sich ständig ändernden Anforderungen zu reagieren.

Bedingt durch den Wettbewerb – gerade in der Software-Branche, aber auch als interne IT-Abteilung – ist es oft erforderlich, in kurzer Zeit neue Funktionalitäten zu entwickeln und anbieten zu können. Somit wird die Produkteinführungszeit (Time-to-Market) immer kritischer, um den Erfolg nicht zu verpassen. Ebenso ist eine frühe Einbindung der Endnutzer in den Entwicklungsprozess wichtig, damit es nicht zu kostspieligen Fehlentwicklungen kommt. Nichtsdestotrotz ist es schon fast eine selbstverständliche Voraussetzung, dass eine solche komplexe Datenplattform performant ist, um Abfragen in Echtzeit durchzuführen und auch Auswertungen von Live-Daten zu ermöglichen.

Autoren

Die Inhalte dieses Dokuments wurden von DSAG-Mitgliedsunternehmen erarbeitet und zusammengestellt. Folgende Autoren haben daran gearbeitet:

Name	Vorname	Firma	Mail-Adresse
Drieschner	Clemens	TU München	clemens.drieschner@tum.de
Hess	Stefan	MHP GmbH	stefan.hess@mhp.com
Köppel	Ralph	AOK BaWü	ralph.koeppel@bw.aok.de
Krenn	Johannes	Q_Perior AG	johannes.krenn@q-perior.de
Niemeyer	Frederik	ISR Information Products AG	frederik.niemeyer@isr.de
Seinsche	Marcel	Q_Perior AG	marcel.seinsche@q-perior.com
Weingart	Volker	UK Würzburg	weingart_v@ukw.de
Unterstützung der SAP SE durch			
Meier	Axel		ax.meier@sap.com
Saggau	Volker		volker.saggau@sap.com
Sauer	Klaus-Peter		klaus-peter.sauer@sap.com

Inhaltsverzeichnis

Zielsetzung des Leitfadens **1**
Zielgruppe **1**
Vorwort **1**
Autoren **2**
Abbildungsverzeichnis **5**
1. HANA SQL DWH – Erweiterung der SAP-DW-Strategie **6**
1.1 Unterschiedliche Ansätze 6
1.1.1 Applikationsgetriebenes Data-Warehouse: SAP BW 6
1.1.2 S/4HANA: embedded BW 6
1.1.3 SQL-getriebener Ansatz 7
1.1.4 Data-Warehouse-Cloud 7
1.1.5 SAP-Analytic-Cloud 7
1.2 Strategie 8
1.2.1 SAP-Strategie vs. Kundenstrategie 9
2. Was verstehen wir unter einem SAP HANA SQL DWH? **10**
2.1 Unterschiede zum SAP BW 12
2.2 Unterschiede zu Any-DB 13
3. Bausteine eines HANA SQL DWH **17**
3.1 Technische Werkzeuge 17
3.1.1 Web IDE 17
3.1.2 PowerDesigner 18
3.1.3 Smart Data Integration 19
3.1.4 Calculation-Views 19
3.1.5 Flowgraphs 20
3.1.6 SAP Data Warehousing Foundation 21
3.1.7 Continuous Integration & Delivery (CI/CD) 23
3.1.8 Core Data Services 25
3.1.9 Git 26
3.2 Entwicklungsmethodik 27
3.3 Sicherheit 30

4. Use-Cases ... 31

4.1 AOK Baden-Württemberg: Natives SAP HANA SQL Data-Warehousing 31

4.2 Hybrides Szenario mit einem SAP HANA SQL DWH und einem SAP BW/4HANA ... 35

4.3 Migration eines SQL-Data-Warehouse in die HANA-Datenbank ... 38

4.3.1 Ist-Analyse ... 40

4.3.2 Zielarchitektur ... 42

5. Fazit ... 44

6. Anhang ... 45

6.1 Glossar ... 45

Impressum ... 46

Abbildungsverzeichnis

Abbildung 1: Überblick der SAP-DWH-Strategie 8
Abbildung 2: Vereinfachte Darstellung des SAP HANA SQL DWH 11
Abbildung 3: Screenshot von der SAP HANA Web IDE 17
Abbildung 4: Screenshot vom SAP PowerDesigner 18
Abbildung 5: Anbindung von Datenquellen an die SAP HANA via SAP Smart Data Integration 19
Abbildung 6: Calculation-View 20
Abbildung 7: Flowgraph zum Laden von Daten in ein NDSO 21
Abbildung 8: Grafische UI zur Verwaltung des NDSO 22
Abbildung 9: Grafische UI zur Erstellung von Task-Chains 23
Abbildung 10: Data Lifecycle Manager 23
Abbildung 11: Continuous Integration & Delivery 24
Abbildung 12: Screenshot von Jenkins 24
Abbildung 13: Unterschied SQL vs. CDS 25
Abbildung 14: Vereinfachte Darstellung von Git 26
Abbildung 15: Continuous Integration & Delivery 28
Abbildung 16: Modellgetriebene Entwicklung 28
Abbildung 17: Multi-Layer-DWH-Architektur 32
Abbildung 18: Grobübersicht der Systemlandschaft 33
Abbildung 19: Ausgangslage Bechtle vor Einführung eines SAP HANA SQL DWH. 35
Abbildung 20: Bechtles Argumente für eine Mixed Architecture 36
Abbildung 21: Beziehungen zwischen Komponenten aus der Mixed Architecture von Bechtle 37
Abbildung 22: Data-Warehouse vor der Migration 39
Abbildung 23: Zielarchitektur 42

Bildnachweise:
Bechtle AG, SAP SE (Grafiken/Produktabbildungen), ISR Information Products AG, eigene Darstellungen

1. HANA SQL DWH – Erweiterung der SAP-DW-Strategie

Zu Beginn des Leitfadens soll das SAP HANA SQL DWH gegenüber den anderen DWH-Ansätzen von SAP abgegrenzt werden. Anschließend erfolgt eine Einordnung des SAP HANA SQL DWH in die SAP-Strategie aus der Perspektive der DSAG-Themengruppe SAP HANA SQL DWH.

1.1 Unterschiedliche Ansätze

Die folgenden Beschreibungen der DWH-Ansätze beziehen sich auf die wesentlichen Punkte, die für eine Abgrenzung notwendig sind. Auf eine detaillierte Beschreibung wird verzichtet. Für weitere Informationen möchten wir gerne auf die zugehörigen DSAG-Arbeitsgruppen bzw. -Themengruppen hinweisen.

1.1.1 Applikationsgetriebenes Data-Warehouse: SAP BW

Vor mehr als 20 Jahren kam SAP mit einem Business-Data-Warehouse auf den Markt, das sich noch heute vor allem durch den Ansatz „Ein Data-Warehouse als abgeschlossene Applikation“ von anderen Lösungen unterscheidet. Dieser Applikationsansatz hat die Aspekte Modellierung, Scheduling/Monitoring, Versionierung/Lifecyle etc. mit bekannten SAP-Mitteln (ABAP-Entwicklungen, Jobmonitoring etc.) abgedeckt – also ein Data-Warehouse als SAP-Applikation. Dies hat zur Konsequenz, dass typische SAP-Skills und weniger Datenbank-Know-how gefordert sind.

Der mitausgelieferte Business-Content ist ein weiteres Alleinstellungsmerkmal und hat den beträchtlichen Vorteil, dass SAP-ERP-Kunden mit Extraktoren, die von SAP bereitgestellt und gewartet wurden, Daten aus ihren SAP-Systemen in ebenfalls darauf abgestimmte Datenmodelle laden können. Wenn dieser Business-Content – wie in vielen Fällen – für die spezifischen Kundenanforderungen nicht wirklich ausreicht, dient er zumindest als Vorlage bzw. als Ausgangspunkt für kundenindividuelle Erweiterungen. So sind im Laufe der Zeit beachtliche Data-Warehouse-Lösungen entstanden, die als solche schon einen gewissen Asset darstellen.

Um ein zeitgemäßes UI erweitert, ist das BW/4HANA die aktuelle Weiterentwicklung dieses applikationsgetriebenen Ansatzes, der mit Hilfe der HANA eine verbesserte Konnektivität bietet.

1.1.2 S/4HANA: embedded BW

Mit der Vermarktung von S/4HANA wird immer wieder beworben, dass ein abgesetztes Data-Warehouse nicht mehr notwendig sei – das embedded BW in S/4 sei ausreichend. Sobald eine Historisierung der Daten oder Cross-Reporting zwischen mehreren SAP-Systemen als Anforderung auftaucht, wird aber schnell klar, dass der Embedded-Ansatz dies nicht erfüllen kann.

1.1.3 SQL-getriebener Ansatz

Da SAP mit HANA nun auch Datenbankhersteller ist – also keine datenbankneutrale Strategie mehr verfolgt –, eröffnet sich dadurch die Möglichkeit, auch Applikationen, insbesondere Data-Warehouses, auf der HANA selbst bzw. der HANA-Applikationsschicht XSA (Einzelheiten s. u.) zu bauen. Damit einhergehend sind andere Skills notwendig bzw. erstmals einsetzbar. Eine SQL-getriebene Entwicklung erfolgt mit anderen Methoden, wie z. B. DevOps, bzw. eröffnet die Möglichkeit, diese einzusetzen. Mehr dazu in den folgenden Kapiteln.

1.1.4 Data-Warehouse-Cloud

Die neueste Entwicklung im Data-Warehouse-Umfeld stellt die Data-Warehouse-Cloud (DWC) dar. Die DWC stellt die HANA-spezifischen Features wie Connectivity, Predictive etc. als Cloud-Lösung zur Verfügung und ist mehr für die Fachabteilung konzipiert. Hier finden sich vor allem Ansätze wieder, die früher unter dem Stichwort „Workspaces“ zu finden waren. Es werden voneinander unabhängige sogenannte Spaces definiert, die sich in erster Linie aus BW-Daten speisen und um eigene Daten erweitert werden können. Man muss abwarten, wohin diese Entwicklung geht.

1.1.5 SAP-Analytic-Cloud

Die SAP-Analytic-Cloud (SAC) ist zwar kein Data-Warehouse, aber das strategische Frontend. Also zumindest was die Frontend-Tools angeht, ist eine SAP-Strategie klar ersichtlich.

1.2 Strategie

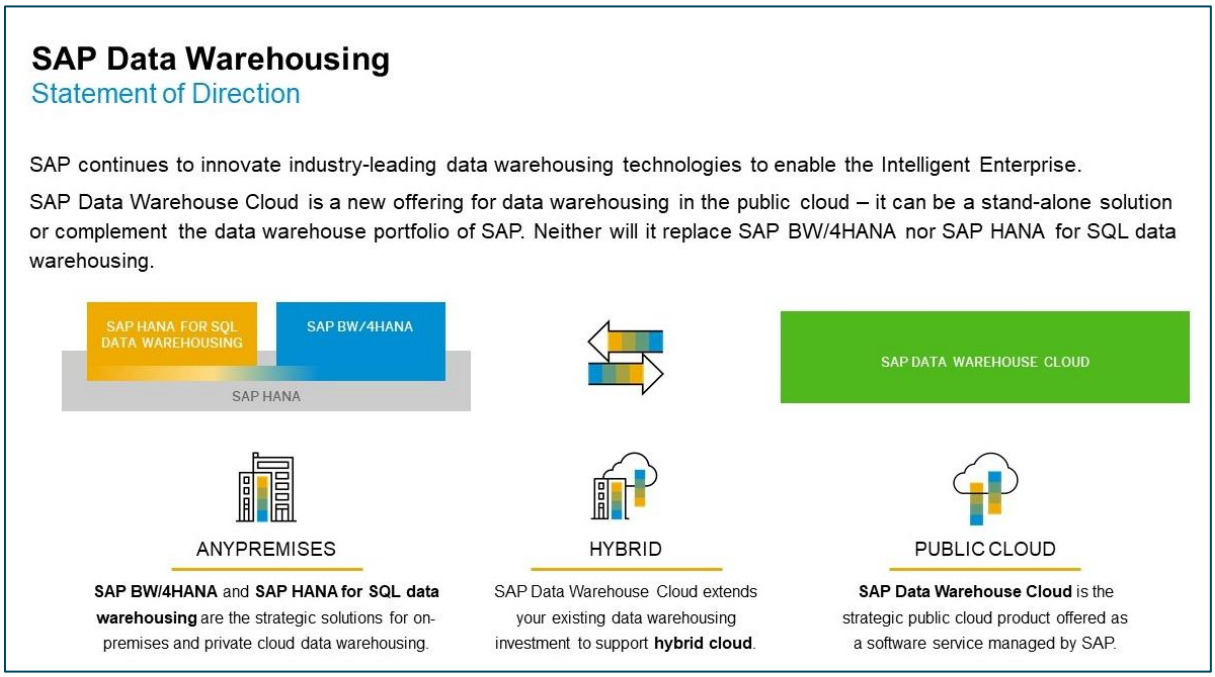

Abbildung 1: Überblick der SAP-DWH-Strategie

Die im vorhergehenden Kapitel beschriebenen Erweiterungen des SAP-Data-Warehouse-Portfolios machen es eher schwerer, eine klare Strategie von SAP zu erkennen. Während die Abbildung 1 eine Koexistenz bzw. gegenseitige Ergänzung nahelegt, entsteht schnell der Eindruck, dass jeweils das eine Produkt als Nachfolger des anderen konzipiert ist. Vor dem Hintergrund, dass sich SAP als Cloud-Company versteht, ist die Argumentation nachvollziehbar, dass das DWC das strategische SAP-Tool sei und alle anderen damit eigentlich schon abgekündigt.

Diverse Gespräche mit SAP-Kunden zeigen eher eine abwartende Haltung. Insbesondere aufgrund der Erfahrung mit anderen SAP-Produkten und einer abrupt endenden Roadmap herrscht die Haltung vor, zunächst abzuwarten, welcher der vielen, vermeintlich widerstrebenden Ansätze „überleben“ wird.

Für diese Haltung sehen wir keinen Anlass, sondern wir sind der Meinung, dass jetzt der Zeitpunkt ist, sich mit den neuen Möglichkeiten auseinanderzusetzen. Dabei soll dieser Leitfaden helfen.

1.2.1 SAP-Strategie vs. Kundenstrategie

Das erweiterte Portfolio von SAP ermöglicht es SAP-BW-Bestandskunden, mittels kleiner Ausflüge auf die SQL-Seite erste Erfahrungen zu sammeln. Für Kunden mit einem stärkeren SQL-Hintergrund wird eine HANA als Data-Warehouse-Plattform interessant. Am spannendsten wird aber vermutlich ein gemischter Ansatz sein, der die Stärken von beiden Seiten miteinander verbindet.

Es ist auch nicht zu erwarten, dass SAP hier einen Best-Practice-Weg beschreiben oder gar vorgeben wird, da diese Entscheidungen sehr kundenindividuell sein werden. Es sind unterschiedliche Skills notwendig bzw. abhängig davon, welche Skills im Haus vorhanden sind, wird die Annäherung an gemischte Szenarien eine andere sein.

Die Komplexität bzw. Heterogenität der Quellsystemlandschaft hat ebenfalls einen Einfluss darauf, wie man in Zukunft seine Data-Warehouse-Strategie ausprägt. Die Anforderungen an ein Data-Warehouse sind inzwischen auch deutlich heterogener geworden. Insbesondere sind agile, schnelllebige, mit großen Datenvolumina verbundene Anforderungen sind wohl eher mit einem SQL-Ansatz zu erfüllen, während das starrere Unternehmensberichtswesen mit größerer Wahrscheinlichkeit im applikationsgetriebenen Ansatz beheimatet sein wird. Ein weiterer Treiber wird die Kundenstrategie bzgl. On-Premise vs. Cloud sein.

Dieser Leitfaden soll einen Einblick geben, mit welchen Punkten man sich auseinandersetzen muss, um die jeweilige Data-Warehouse-Ausprägung angehen zu können – die Entscheidung über „Ob“ und „Wie“ wird immer eine kundenindividuelle bleiben.

2. Was verstehen wir unter einem SAP HANA SQL DWH?

Das SAP HANA SQL DWH ist, wie der Name schon sagt, ein SQL-Data-Warehouse auf der SAP-HANA-Plattform. Besondere Betonung liegt dabei auf dem Wort „SQL“. Sowohl Entwicklung als [1]auch Betrieb des SAP HANA SQL DWH erfolgen entweder direkt auf Basis von SQL oder indirekt auf Basis von HANA-nativen Datenbankobjekten wie Calculation-Views – die ihrerseits wiederum natives SQL produzieren. Zentrale Charakteristika des SAP-HANA-SQL-DWH-Ansatzes sind sein Fokus auf modellgetriebene Entwicklung, seine Agilität, seine Bereitschaft für On-Premise- oder Cloud-Betrieb sowie seine grundsätzliche Offenheit für Technologien und Werkzeuge aller Art (mehr zu diesen Charakteristika auch in Kapitel 3).

Architektur des SAP HANA SQL DWH

Auf den ersten Blick unterscheidet sich ein HANA SQL DWH wenig von anderen typischen DWH (siehe Abbildung 2 auf der folgenden Seite). Seine Hauptaufgabe ist es, Daten aus unterschiedlichen Quellen zu integrieren und diese integrierten Daten für Frontend-Werkzeuge zur Verfügung zu stellen. Die zugrundeliegenden Datenmodelle und ETL-Prozesse basieren dabei auf SQL. Dies trifft auf ein HANA SQL DWH in ähnlicher Weise zu wie für Data-Warehouses auf Basis anderer SQL-Datenbanken.

Bei näherer Betrachtung wird jedoch schnell ein essenzieller Unterschied sichtbar: SAP HANA Extended Application Services Advanced Model, kurs XSA. Das HANA SQL DWH basiert nicht bloß auf einer einfachen SQL-Datenbank, sondern nutzt die Möglichkeiten der SAP HANA als Plattform. Innerhalb dieser Plattform arbeitet – neben einer ganzen Reihe weiterer nützlicher Services – ein eigenständiger Application-Service, der von der Datenbank losgelöst ist.

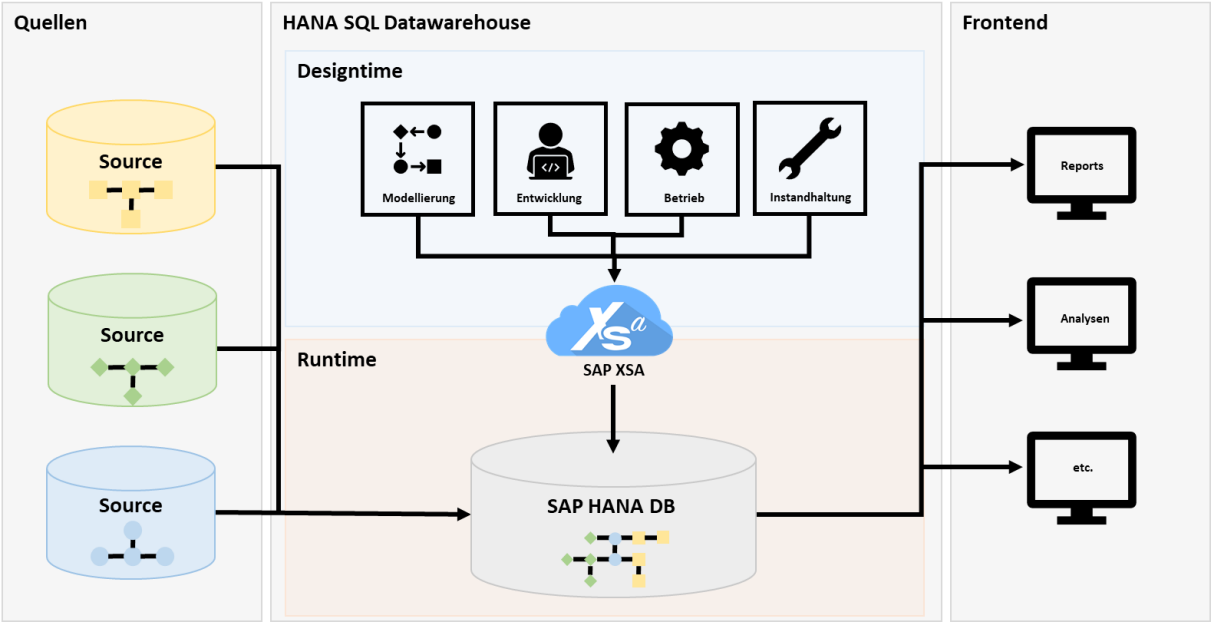

Abbildung 2: Vereinfachte Darstellung des SAP HANA SQL DWH

Dieser dedizierte Application-Service birgt den entscheidenden Vorteil, dass Prozesse zur Modellierung und Entwicklung sowie Betrieb und Instandhaltung des Data-Warehouse nicht mehr direkt auf der Runtime der Datenbank erfolgen, sondern anhand von Designtime-Objekten, wie zum Beispiel Tabellen, Views und Prozeduren. Diese Designtime-Objekte werden mittels der SAP XSA in der Runtime der Datenbank ausgeführt. Im übertragenen Sinne agiert die SAP XSA als Pseudo-Compiler, der von Entwicklern programmierte, abstrakte Designtime-Objekte in ausführbare Runtime-Objekte auf der Datenbank übersetzt. Für die Überführung der Designtime-Objekte in die Datenbank behilft sich die XSA außerdem der sogenannten HANA Deployment Infrastructure (dazu unten im Text mehr). Aus der Unterscheidung zwischen Designtime und Runtime ergeben sich folgende Vorteile:

Komplexitätsreduktion

Der Entwickler kann sich voll und ganz auf die Definition der Designtime-Objekte konzentrieren. Wie genau die Definition übersetzt wird, ob zum Beispiel ein CREATE- oder ein UPDATE-Statement ausgeführt wird, entscheidet die XSA eigenständig.

Kollaboration

Das Sperren von Objekten für die Bearbeitung, wie es beispielsweise im Business-Warehouse praktiziert wird, ist nicht notwendig. Einzelne Designtime-Objekte können von mehreren Entwicklern gleichzeitig verwendet und bearbeitet werden. Hierzu bietet die XSA die Möglichkeit, in isolierten Laufzeiten, sogenannten Sandboxes, zu arbeiten. Jeder Entwickler erhält eine solche Sandbox und ist damit unabhängig vom bereits

deployten DWH und von den Entwicklungen der Kollegen. Die jeweiligen Entwicklungsstände werden zu einem späteren Zeitpunkt an einer zentralen Stelle zusammengeführt.

Versionierung und Auditierfähigkeit

Da sich die Entwicklungsarbeit auf Designtime-Objekte beschränkt, die letztlich nichts anderes als Quellcode darstellen, können Objekte exakt versioniert und auditiert werden. Hierzu bedient sich das HANA SQL DWH des Source-Code-Management-Werkzeugs Git, das in der Software-Entwicklung bereits seit Jahrzehnten für genau diesen Zweck genutzt wird. Mit Git ist es beispielsweise möglich, genau nachzuvollziehen, welcher Entwickler welches Objekt wann erstellt hat und wann es weiterentwickelt wurde.

Vereinfachtes Deployment

Für das Deployment bietet die SAP-HANA-Plattform eine fest vorgegebene und strikt standardisierte Infrastruktur (HANA Deployment Infrastructure, kurz HDI), die das Bündeln der Designtime-Objekte und die konsistente Einhaltung von Abhängigkeiten für das Deployment regelt. Mithilfe der HDI ist es möglich, plattformunabhängig zu deployen. Das heißt, neben dem konventionellen On-Premise-Deployment sind auch Cloud-Deployments möglich – einzige Voraussetzung ist das Vorhandensein der HDI. Ebenso ist es möglich, Deployments inkl. dazugehöriger Tests vollständig zu automatisieren. Weitere Informationen zu HDI im Kontext Sicherheit finden Sie in Kapitel 3.3.

2.1 Unterschiede zum SAP BW

Mit dem HANA SQL Data-Warehouse bietet die SAP einen neuen Ansatz für den Aufbau eines Data-Warehouse. Das HANA SQL Data-Warehouse bietet nicht nur einen SQL-getriebenen Ansatz für das Data-Warehouse, sondern ermöglicht auch die Integration eines „Best of Breed“-Ansatzes in der Data-Warehouse-Architektur. SAP stellt für die meisten Aufgaben im Data-Warehouse eigene Tools zur Verfügung, aber ermöglicht es dem Kunden, auch eigene Tools für gewisse Aufgaben im DWH einzubinden. Nicht für alle benötigten Funktionalitäten bietet SAP eigene Tools an, sodass zumindest für CI/CD (Erläuterung in Kapitel 3.1.7) ein GIT-Server eingerichtet werden sollte. Durch die Einbindung der verschiedenen Tools kann in der Datenmodellierung und der Implementierung ein gewisser Grad an Freiheit ermöglicht werden, sodass man während der Implementierung nicht in eine bestimmte Architektur oder Vorgehensweise gezwungen wird. Das benötigte Know-how der Mitarbeiter unterscheidet sich beim SQL Data-Warehouse vom BW/4HANA, sodass für das SQL Data-Warehouse SQL-Skills und keine ABAP-Skills benötigt werden.

Das BW/4HANA hingegen bietet einen „Best in Class“-Ansatz für die Implementierung des Data-Warehouse und bietet eine komplette und vordefinierte Tool-Landschaft für die Entwicklung des Data-Warehouse. Das BW/4HANA ist als applikationsgetriebenes Data-Warehouse der SAP sehr gut in die SAP-Systemlandschaft integriert, sodass für viele Produkte der SAP ein Business-Content angeboten wird. Der Business-Content bietet eine Auswahl an vordefinierten Schnittstellen und Datenmodellen für die verschiedenen SAP-Systeme, welche über User-Exits jederzeit erweitert werden können. Für das BW/4HANA bietet die SAP eine hybride Variante an, sodass die Funktionen des BW/4HANA und des HANA SQL Data-Warehouse verbunden werden können. Die hybride Option bietet die Vorteile beider Ansätze, allerdings muss diese auch extra lizenziert werden und ist nicht in der BW/4HANA-Lizenz enthalten.

Tabelle 1: Unterschiede zwischen BW/HANA und DWH

	Anforderungen der Kunden	Mitarbeiterskills der Kunden
SAP BW/4HANA	• Ein „Best in Class“-Tool • SAP-lastige Systemland-schaft • Nutzen von Business-Content der SAP	• Weitreichendes SAP- und ABAP-Know-how
HANA SQL Data-Warehouse	• Ein „Best of Breed“-Tool • Ausgeprägte Non-SAP-Systemlandschaft • Ersetzen von Third-Party-DWH • Aufbau von Applikationen aus dem DWH	• SQL-Know-how

2.2 Unterschiede zu Any-DB

Die Unterschiede eines SAP HANA SQL DWH zu einem SQL DWH mit einer Any-DB sind verständlicherweise von der Any-DB abhängig. Wir verzichten darauf, alle relevanten Funktionen aller Datenbankhersteller mit der HANA zu vergleichen. Wir konzentrieren uns in diesem Kapitel auf Themen, die uns beim Umstieg auf HANA aufgefallen sind und den Umstieg auf HANA erleichtern sollen. Weil es sich um den Orientierungsleitfaden für ein SQL DWH handelt, lassen wir an der Stelle alle NoSQL-Datenbanken außen vor.

Relationale Datenbanken sind für **OLTP** optimiert, um möglichst viele Schreib- und Lesezugriffe auf Einzelzeilenbasis in kurzer Zeit zu bedienen. Dies ist praktisch für transaktionale Systeme wie HANA. Für Systeme, die zudem noch analytische Auswertungen durchführen sollen, eignet sich **OLAP** mehr. Mit HANA können die Vorteile von OLTP und OLAP kombiniert werden. Dieses ERP-Systembringt Objekte

(zeilen-/spaltenbasierte Tabellen) und Werkzeuge mit (siehe Kapitel 3.1.4 Calculation-Views). Auf diese Weise kann man nicht nur Data-Marts erzeugen, sondern auch gleich Cubes für eine multidimensionale Analyse der Daten vorbereiten.

Der Zugriff während der Entwicklung erfolgt wie bei vielen anderen Datenbanken über eine **Console**. Bei der SAP-HANA-Datenbank ist das der Client „SAP HANA Administration Console“ oder der Web-Client „Web IDE“. Mit den Erfahrungen aus Client-Tools anderer relationaler Datenbankhersteller findet man sich in der HANA-Datenbank sehr schnell zurecht. Auf der linken Seite des Bildschirms befindet sich ein Navigationsbaum, über den die Objekte eingesehen werden können. Auf der rechten Seite befindet sich die sogenannte SQL-Console, über die SQL-Statements abgesetzt werden. Je nach Statement erfolgt das Feedback über separate Fenster. Hierzu gehört zum Beispiel die Liste der Daten, die abgefragt oder berechnet wurden, oder eine Fehlermeldung, wenn die Syntax nicht korrekt ist.

Die Objekte in der Datenbank sind in **Schemas** strukturiert und können über den Navigationsbaum oder über System-Views der Datenbank angezeigt werden. Zu den Objekten in der Datenbank zählen Tabellen, Views, Trigger, Funktionen, Prozeduren, Libraries, Synonyms, Grants etc. Bei den System-Views überrascht es nicht, dass jeder Hersteller (so auch SAP) seine eigenen Bezeichnungen hat. Ein Dictionary, und damit eine Liste mit allen System-Views und deren Beschreibung, ist in der Datenbank nicht vorhanden. Diese Liste befindet sich in der technischen Dokumentation. Bei der Verwendung der XSA werden die Objekte in HDI-Containern (ähnlich dem Datenbankschema) angelegt.

Bei der Erstellung der Create-Statements für Tabellen gibt es Unterschiede zu anderen Datenbankhersteller bei der Bezeichnung der **Datentypen**. Ein SQL-Select-Statement mit einer Liste von Attributen, Tabellen und Joins unterscheidet sich nicht von anderen Datenbanken, solange in diesem Statement keine Berechnungen mit Datenbankfunktionen durchgeführt werden und der ANSI-Standard eingehalten wird. Größere Unterschiede gibt es bei Berechnungen mit zeitlichem Bezug. So kann z. B. bei Oracle zu einem beliebigen Tag eine Zahl vom Typ integer addiert werden. Das Ergebnis ist wiederum ein Datum mit der Anzahl Tage später. Hierfür hat S/4HANA eine Funktion ADD_DAYS. Bei der Verwendung der Konkatenationsfunktionen CONCAT oder || liefert HANA, wenn einer der Werte den Wert NULL hat, immer NULL im Ergebnis zurück. Verwenden wir eine der beiden Funktionen in einer anderen Datenbank, erhalten wir andere Ergebnisse. Unterschiede gibt es bei der Programmierung von Libraries und deren Prozeduren und Funktionen mit HANAN SQLScript. In einer Library können Prozeduren und Funktionen gekapselt werden. Diese gibt es in anderen Datenbanken unter dem Namen „Package“, oder es gibt keine entsprechende Funktionalität. Obwohl die Ablaufsteuerung nahezu identisch ist, gibt es bei der Programmierung Unter-

schiede, wie zum Beispiel bei der Deklaration von Variablen oder bei Ausnahmebehandlungen. Bei diesen Themen ist der Unterschied von einer beliebigen Datenbank zu HANA nicht größer als die Unterschiede zwischen den anderen Datenbankherstellern. Die Tabelle DUMMY gibt es bei anderen Datenbankherstellern entweder unter einem anderen Namen, wie zum Beispiel DUAL, oder Datenbankhersteller wie Microsoft oder Teradata verzichten auf diese Dummy-Tabelle.

Alle Tools für die Erstellung eines HANA SQL Data-Warehouse sind auf der SAP-HANA-Plattform verfügbar. Die Datenhaltung und -verarbeitung findet in der SAP HANA **In-Memory** statt. Auf diese Weise können sehr große Datenmengen schnell verarbeitet werden, da sich diese bereits im Hauptspeicher befinden und nicht vom Festplattenspeicher geladen werden müssen. Somit hat man mit der SAP HANA einen deutlichen Performance-Gewinn im Vergleich zu einer Datenbank, die keine In-Memory-Technologie verwendet. Bei der Anlage einer Tabelle kann bei der HANA DB zwischen „row-based“ und „columnar“ unterschieden werden. Bei der Implementierung verwenden die Datenbankhersteller unterschiedliche Konzepte und unterscheiden sich somit auch in der Syntax.

Wichtig für eine schnelle Datenverarbeitung ist es, dass die Daten möglichst da verarbeitet werden, wo sie auch gespeichert sind, und nicht erst in eine Applikationsschicht geladen werden müssen. Die oben genannten Werkzeuge der SAP HANA reichen dafür Verarbeitungsschritte an die Datenbank weiter, also die SAP HANA, wo sich die Daten befinden. Der **SQL Optimizer** der SAP HANA bereitet die Abfragen passend für die verschiedenen **Engines** (Join, OLAP, Calc, Row Store) in der SAP HANA vor. Diese Engines können je nach Zweck die Daten optimiert und somit schneller und gezielter verarbeiten. Gemeinsam mit anderen Datenbanken ist der Ausführungsplan, der darstellt, welche Tabellen in welcher Reihenfolge und über welche Mechanismen gelesen und verarbeitet werden. Bei der Interpretation der Ausführungspläne gibt es wiederum herstellerspezifische Ausprägungen.

Die Abbildung der **ETL**-Prozesse in einer relationalen Datenbank kann über Stored Procedures erfolgen, was aber nicht sehr einfach zu modellieren, prozessieren und überwachen ist. SAP HANA bietet hingegen auch dafür Werkzeuge (siehe Kapitel 3.1.5 Flowgraphs), um schnell und einfach ETL-Prozesse zu modellieren, diese einzuplanen und auch zu überwachen. Je nach Funktionalität der Any-DB sind zusätzliche Systeme notwendig, um ETL-Prozesse zu steuern.

Die **Versionierung** von Objekten und **Historisierung** von Daten werden von einfachen Datenbanken nicht übernommen. Dafür benötigt es immer zusätzliche Funktionalitäten. SAP HANA bietet die Möglichkeit, Objekteversionen zu erstellen und bspw. in einem Git-Repository zu speichern (siehe Kapitel 3.1.9 Git). Des Weiteren gibt es

bei einer SAP HANA Objekte, die die Historisierung von Daten übernehmen und somit Ladevorgänge rückgängig gemacht werden können (siehe Kapitel 3.1.6 DWF).

Die Erstellung von neuen Objekten (**Deployment**) wird bei der SAP HANA von der XSA (siehe Dokumentation zu XSA) durchgeführt. Dabei übernimmt die XSA Schritte, die bei einer SQL-Datenbank in der Regel vom Entwickler unternommen werden müssen. Dazu zählt bspw. das Anpassen von Tabellen, wenn sich die Struktur ändert. Mit der XSA wird automatisch das Delta zwischen den Tabellendefinitionen ermittelt und ein ALTER-Befehl im Hintergrund generiert und ausgeführt. Somit können eigenständige Erstellungen von ALTER-Befehlen und eventuelle Neubeladungen eingespart werden.

Zuletzt soll noch der Punkt **Integration** von Third-Party-Applikationen angesprochen werden. Ein HANA SQL DWH profitiert von der Offenheit der HANA-Plattform, auf der das DWH entwickelt wird. Mit Hilfe von standardisierten Treibern (bspw. ODBC/JDBC) oder Schnittstellen (bspw. Odata) können Quellsysteme einfach angeschlossen und konsumierte Systeme/Applikationen bedient werden.

3. Bausteine eines HANA SQL DWH

Im Kern basiert das SAP HANA SQL DWH auf der SAP-HANA-Plattform. Darüber hinaus kommen jedoch weitere technische Werkzeuge sowie eine Entwicklungsmethodik zum Einsatz, die im Folgenden beschrieben werden.

3.1 Technische Werkzeuge

Grundsätzlich zeigt das SAP HANA SQL DWH eine hohe Offenheit für unterschiedlichste technische Werkzeuge – darunter sowohl SAP-proprietäre Werkzeuge als auch Werkzeuge von Drittanbietern. Im Folgenden wird eine Auswahl von technischen Werkzeugen beschrieben, die sich für Entwicklung und Betrieb des SAP HANA SQL DWH bewährt haben.

3.1.1 Web IDE

Die SAP HANA Web IDE ist eine web-basierte Entwicklungsumgebung, die auf der XSA läuft und mit dieser ausgeliefert wird. Sie ist auf die Entwicklung von XSA-Anwendungen zugeschnitten und bietet daher einige spezielle Funktionen und Werkzeuge. Hierzu gehören unter anderem grafische Editoren für SDI-Flowgraphs, Calculation-Views, CDS-Objekte, DWF-Task-Chains sowie integrierte Funktionen, um diese Objekte auszuführen und zu testen. Abbildung 3 zeigt die Web IDE mit einem Workspace, in dem projektbezogene Daten angelegt und gespeichert werden.

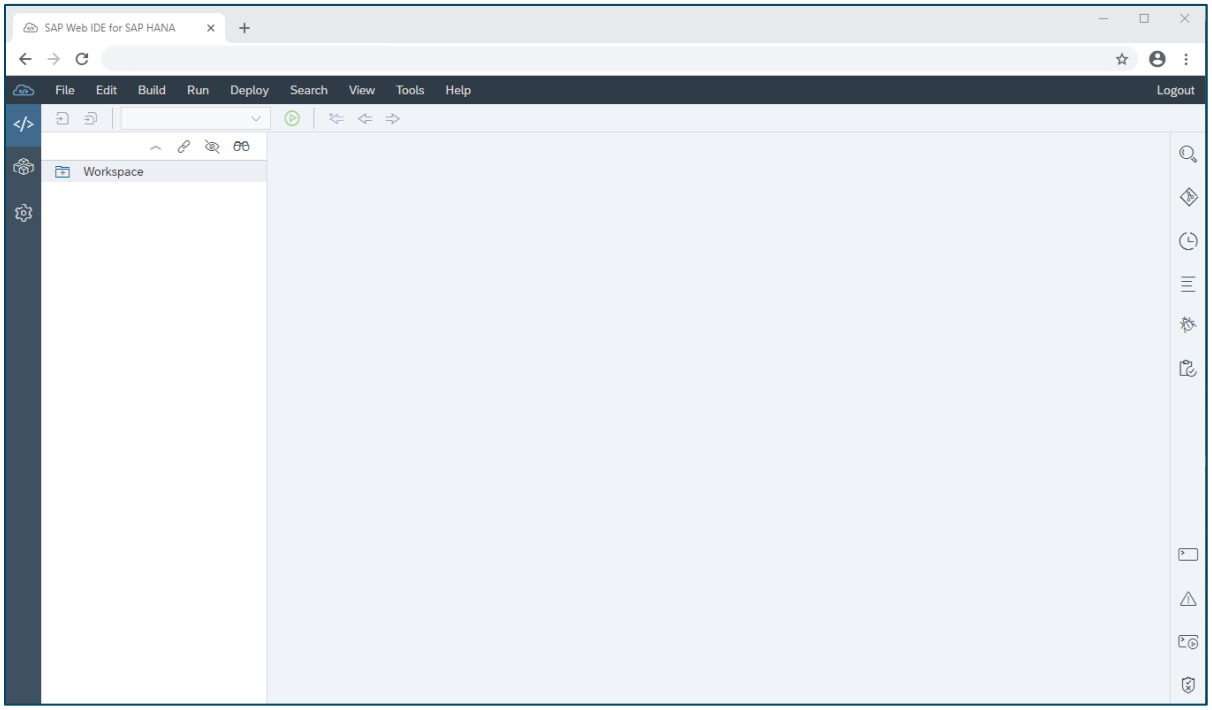

Abbildung 3: Screenshot von der SAP HANA Web IDE

Neben einer Entwicklungsoberfläche verfügt die SAP HANA Web IDE über einen integrierten Database-Explorer, mit dem sich entweder ganze Datenbanken oder auch einzelne Datenbank-Container für Entwicklungszwecke anbinden und explorieren lassen. Des Weiteren bietet die SAP HANA Web IDE gängige Werkzeuge der Software-Entwicklung wie eine Git-Integration, einen Debugger, Testwerkzeuge und vieles mehr.

3.1.2 PowerDesigner

Der SAP PowerDesigner (siehe Abbildung 4) ist nicht Teil der SAP-HANA-Plattform, sondern ein eigenständiges Modellierungswerkzeug aus dem SAP-Portfolio. Mit dem PowerDesigner ist es möglich, komplexe Datenmodelle zu erstellen und zu pflegen. Dies ist nicht bloß aus technischen Gesichtspunkten hilfreich, sondern auch mit Blick auf die Entwicklungsmethodik. Mehr dazu in Kapitel 3.2.

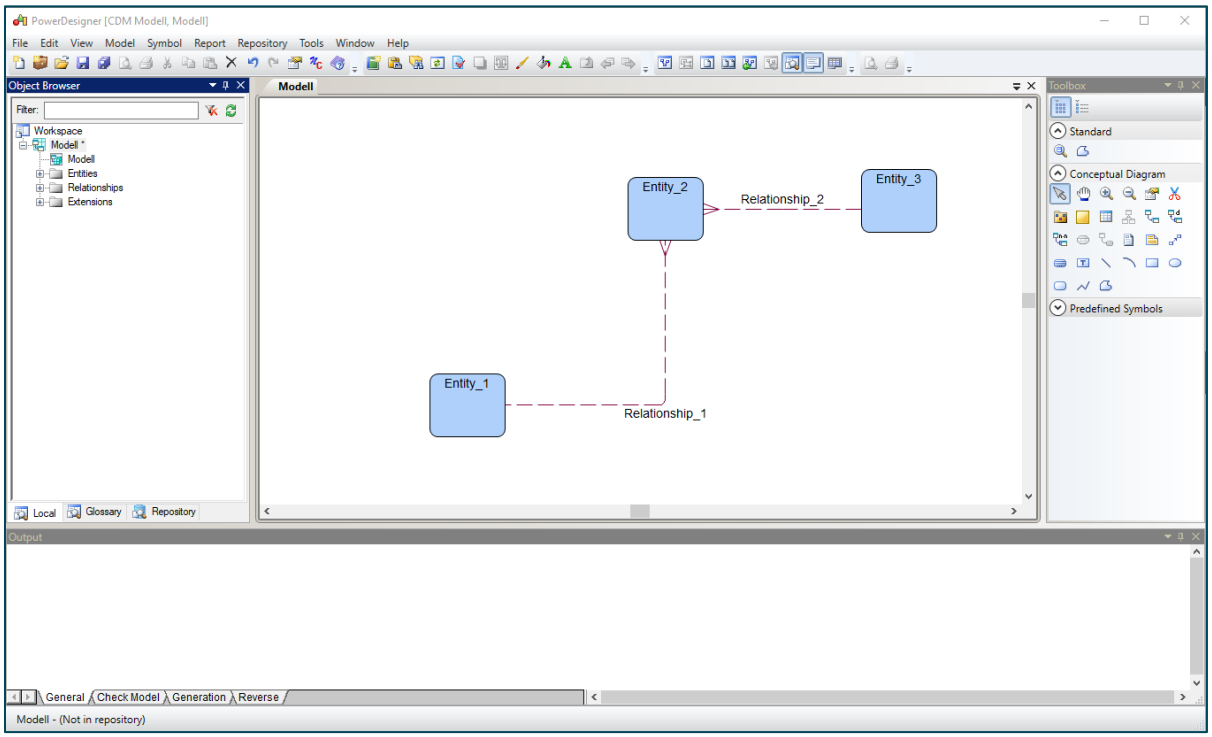

Abbildung 4: Screenshot vom SAP PowerDesigner

Seine besonderen Stärken zeigt der PowerDesigner zum einen im Reverse-Engineering. Das bedeutet, dass über eine Datenbankverbindung bestehende Modelle direkt in den PowerDesigner importiert und dort weiterentwickelt werden können. Des Weiteren bietet der PowerDesigner umfangreiche Lineage-Analyse-Funktionen, um Ursprünge und Ziele der Daten sowie zwischengelagerte Transformationen genau nachvollziehen zu können. Eine weitere nützliche Funktionalität des PowerDesigner liegt in den verschiedenen Exportmöglichkeiten für die Datenmodelle. So ist es möglich – je

nach Zieldatenbank –, die modellierten Artefakte als CDS-Objekte (für SAP-HANA-Datenbanken) zu exportieren oder SQL-Befehle (für Datenbanken von Drittanbietern) generieren zu lassen. Die generierten SQL-Befehle können wahlweise auch direkt aus dem PowerDesigner heraus ausgeführt werden.

3.1.3 Smart Data Integration

SAP Smart Data Integration (SDI) ist in die Entwicklungsumgebung Web IDE integriert und findet sich dort in dem grafischen Editor für sogenannte Flowgraphs wieder. Mittels SDI ist es möglich, nahezu jede Datenquelle an die HANA-Datenbank anzubinden – ob SQL-Datenbank, No-SQL-Datenbank, verteilte Datenbanken à la Hadoop oder Social-Media-Webseiten. Hierzu steht standardmäßig eine große Auswahl von Konnektoren zur Verfügung, die herkömmliche Szenarien wie oben beschrieben abdecken. Für weitere Szenarien steht ein Software-Development-Kit zur Verfügung, mit dem individuelle Konnektoren erstellt werden können.

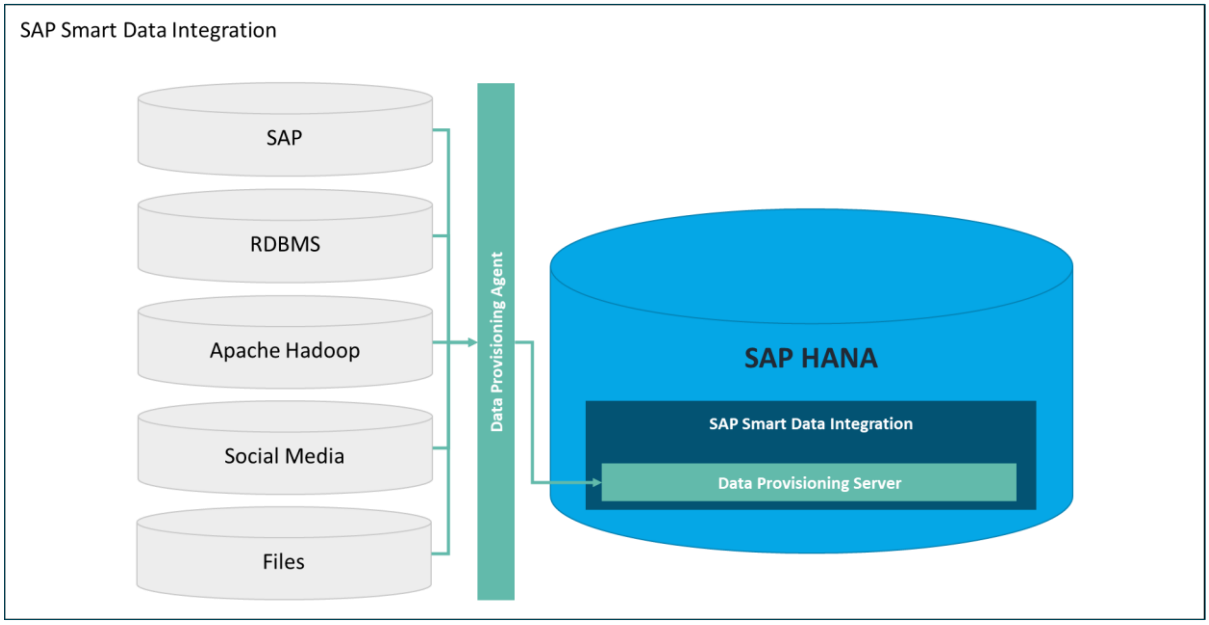

Abbildung 5: Anbindung von Datenquellen an die SAP HANA via SAP Smart Data Integration

Technisch ist SDI voll integriert in die SAP-HANA-Plattform. Zur Nutzung muss auf dem Quellsystem lediglich ein sogenannter Data Provisioning Agent installiert und mit dem entsprechenden Adapter angebunden werden. Anschließend kann der Data Provisioning Server der SAP-HANA-Plattform mit dem Data Provisioning Agent kommunizieren, wie in Abbildung 5 dargestellt.

3.1.4 Calculation-Views

Die Calculation-View ist eine besondere Form von Datenbank-Views, welche grafisch über die Web IDE implementiert werden kann. Die Calculation-View unterscheidet sich

von herkömmlichen Datenbank-Views im Umfang der Funktionalität und kann auch semantische Informationen für den Fachbereich oder Reportingtools bereitstellen. Die semantischen Informationen werden in einem Semantikknoten der Calculation-View gepflegt und können die folgenden Informationen enthalten:

- Aggregationsanweisungen für Kennzahlen (Summation, Max-Funktionen, AVG etc.)
- Verknüpfung von Text- und Schlüsselspalten
- Aufbau von Level- und Parent-Child-Hierarchien
- Data-Masking

Ein weiterer Unterschied zur herkömmlichen Datenbank-View ist die Ausführung der Calculation-View in der Calculation-Engine der HANA statt der SQL-Engine. In der Calculation-Engine wird die View wie ein Auswertungs-Cube behandelt und entsprechend werden Kennzahlen je nach Aufriss aggregiert, und die Ausführung der View wird für das Aggregationsszenario optimiert.

Calculation-Views können noch in Dimension- und Cube-Views unterteilt werden. Durch diese Unterscheidung können mit den Calculation-Views auch Star- oder Snowflake-Modelle aufgebaut werden. Die Dimension-Calculation-View bietet auch eine Option, um spezielle Zeitdimensionen aufzubauen. In den Cube-Calulation-Views können die Bewegungsdaten mit den verschiedenen Dimensionen verjoint werden und somit ein Auswertungs-Cube für ein Reporting-Werkzeug erzeugt werden.

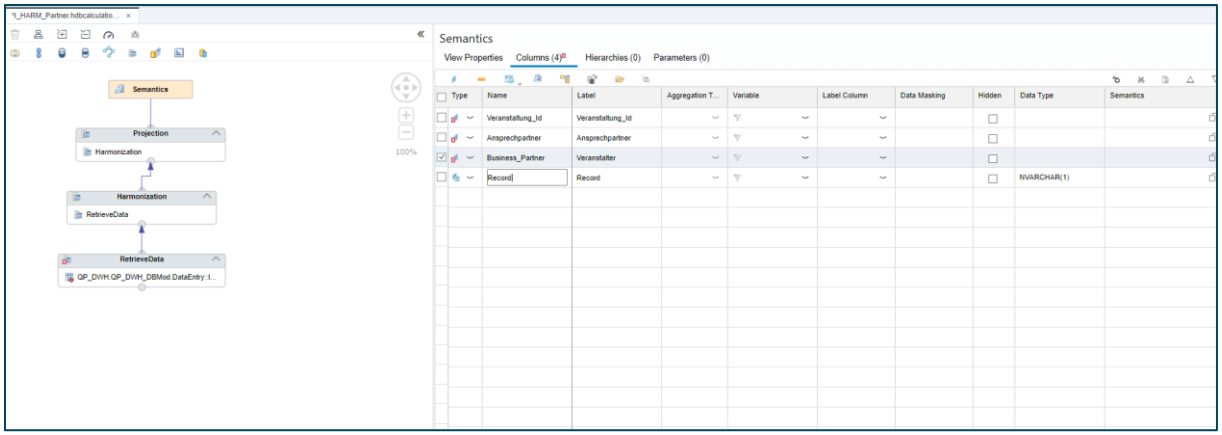

Abbildung 6: Calculation-View

3.1.5 Flowgraphs

Der Flowgraph ist ein Smart-Data-Integration-Tool (SDI-Tool) und wird im SQL Data-Warehouse verwendet, um Daten zu laden, zu transformieren und zu persistieren. Somit dienen Flowgraphs als ETL-Tool innerhalb des Data-Warehouse. Sie können direkt in der Web IDE über einen grafischen Editor angelegt werden. Hierbei wird eine Datei

mit der Endung „.hdbflowgraph“ erstellt, welche im Anschluss gebuilded und als Runtime-Objekt ausgeführt werden kann. Ein Flowgraph besteht aus mehreren Knoten, die in der grafischen Oberfläche der Web IDE intuitiv modelliert und konfiguriert werden können. So kann im Knoten „Data Source“ die Datenquelle (z. B. über ein Synonym) hinterlegt und im Knoten „Data Target“ das Ziel der Transformation (z. B. Tabelle oder NDSO) angegeben werden. In einem „Projection“-Knoten kann ein Filter bzw. eine Projektion (z. B. zusätzliche Spalte oder angepasstes Mapping) eingefügt werden. Außerdem gibt es noch einige weitere mächtige Knoten wie „LookUp“, „Join“ oder „History Preserving“, auf welche jedoch in diesem Kapitel nicht näher eingegangen wird.

In Flowgraphs können auch mehrere Datenquellen, Transformationen und Datenziele eingebaut werden, was die Abbildung eines ganzheitlichen Datenflusses ermöglicht. Außerdem werden Flowgraphs im Gegensatz zu anderen ETL-Tools vollständig auf der HANA-Datenbank ausgeführt. Dadurch ist keine Engine beteiligt, welche Teile der Logik ausführt, sondern es erfolgt ein kompletter Pushdown auf die HANA-Datenbank. Dies ermöglicht eine sehr performante Ausführung.

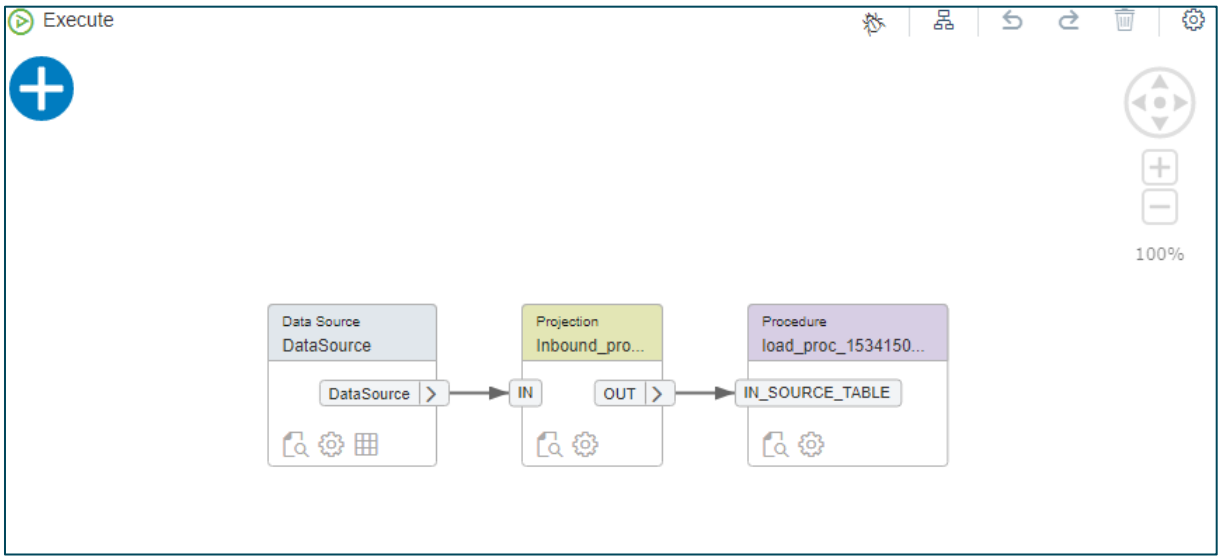

Abbildung 7: Flowgraph zum Laden von Daten in ein NDSO

3.1.6 SAP Data Warehousing Foundation

Die SAP HANA Data Warehousing Foundation (DWF) ist ein Toolset zur Unterstützung von SAP HANA SQL Data-Warehouse Use Cases. Die DWF stellt somit mehrere Funktionalitäten zum Datenmanagement im SQL DWH bereit, u.a. das Native Data Store Object (NDSO), den Scheduler (DWS) oder den Data Lifecycle Manager (DLM).

Das NDSO wurde mit SAP HANA Data Warehousing Foundation 2.0 (SP01) eingeführt und ist ein natives HANA SQL Object, das auf dem HANA XSA Framework implementiert wird. Es ist ein Persistenzobjekt im HANA SQL Data-Warehouse und ist vollständig in die Web IDE und in den Enterprise Architecture Designer (EAD) integriert. Außerdem lässt sich das NDSO in Flowgraphs und Task-Chains einbauen. Das NDSO stellt ein vollumfängliches Request-Handling und Delta-Management bereit und lässt sich in einer grafischen Oberfläche in der Web IDE verwalten. Es besteht, analog zum Data Store Object im SAP BW, aus einer (oder mehreren) „Inbound Queue“, einer „Active Data“ und einer „Change Log“-Tabelle. In der grafischen Oberfläche ist eine Verwaltung der Requests und das Durchführen eventueller Rollbacks möglich. Außerdem bietet das NDSO die Möglichkeit der selektiven Löschung von Daten.

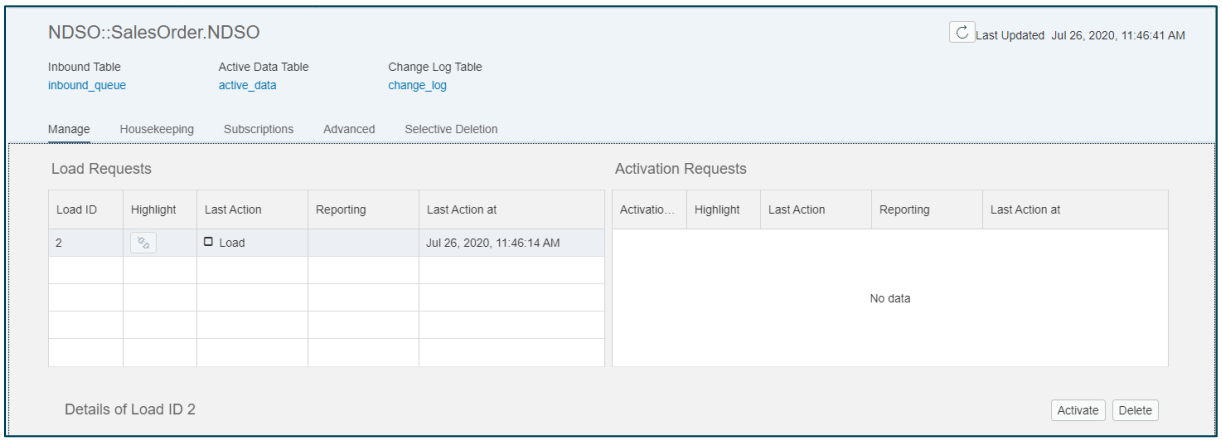

Abbildung 8: Grafische UI zur Verwaltung des NDSO

Der Data Warehouse Scheduler (DWS) ermöglicht es, Task-Chains über eine grafische UI zu implementieren und zu monitoren. Dieses Framework ist ebenfalls vollständig in die Web IDE integriert. In einer Task-Chain können verschiedene Tasks, wie z. B. die Ausführung eines Flowgraphs oder einer Procedure, das Löschen von Logs oder die Aktivierung eines NDSO in einer bestimmten Reihenfolge ausgeführt und auch zeitlich geplant werden. Die Erstellung erfolgt ähnlich wie bei Flowgraphs in einer grafischen UI, in welcher die verschiedenen Tasks angeordnet werden können.

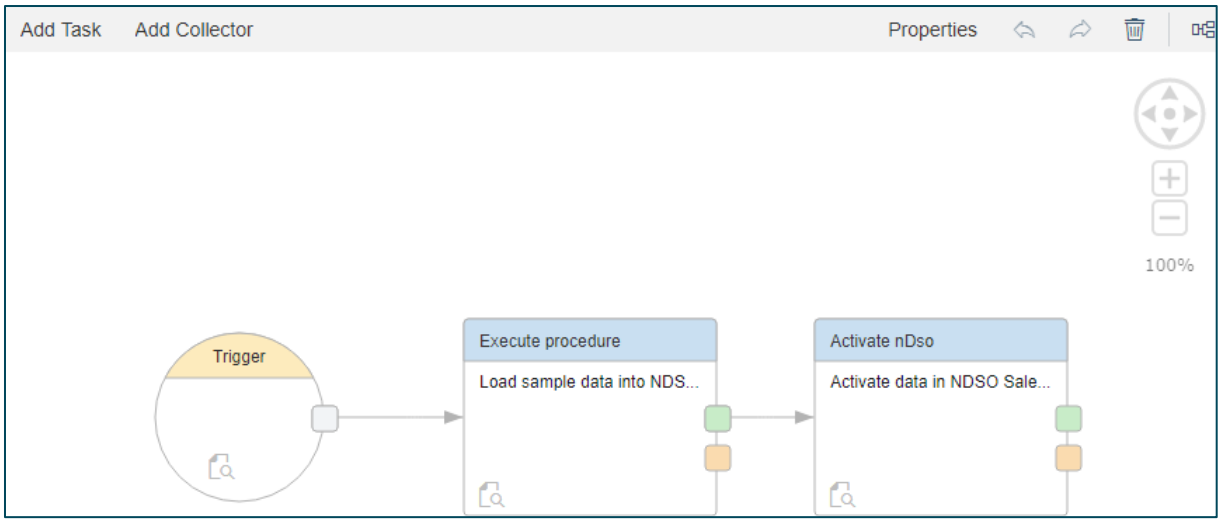

Abbildung 9: Grafische UI zur Erstellung von Task-Chains

Der Data Lifecycle Manager (DLM) ist ein Web-basiertes Daten-Management-Tool, welches es erlaubt, Data-Tiering-Mechanismen zu implementieren. Dabei geht es um die Klassifizierung und Speicherung von Daten abhängig von der Zugriffshäufigkeit (Unterscheidung in Hot/Warm/Cold). Somit können veraltete oder weniger häufig verwendete Daten aus SAP-HANA-Tabellen ausgelagert werden.

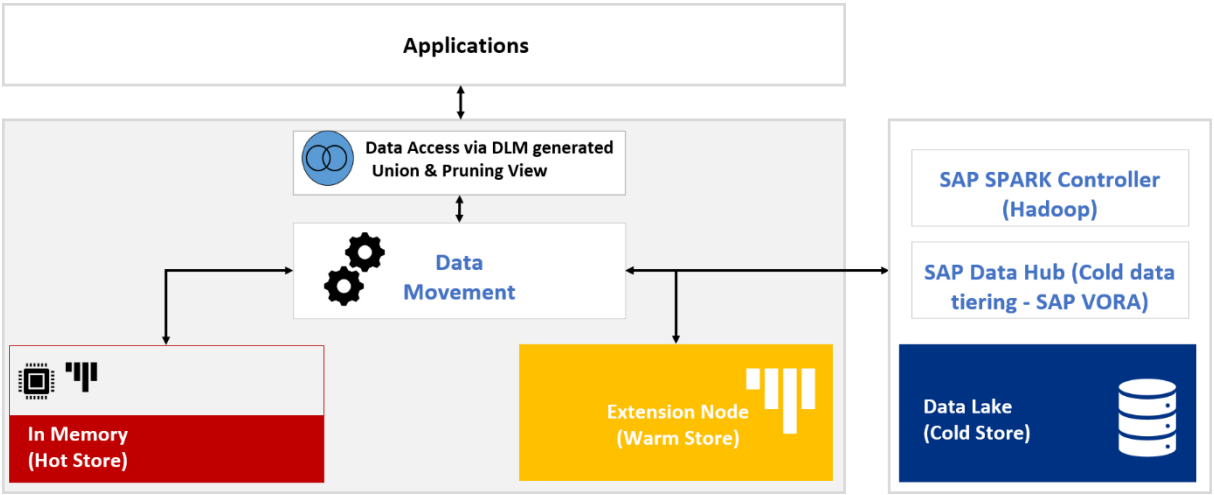

Abbildung 10: Data Lifecycle Manager

3.1.7 Continuous Integration & Delivery (CI/CD)

Der Begriff CI/CD beschreibt die Kombination von Continuous Integration (CI) und Continuous Delivery (CD; siehe Abbildung 11) – beides Kernaspekte der DevOps-Philosophie, die wie Git ebenfalls einen Best Practice der Software-Entwicklung darstellt und zum Ziel hat, die Entwicklung des SAP HANA SQL DWH möglichst effizient zu gestalten.

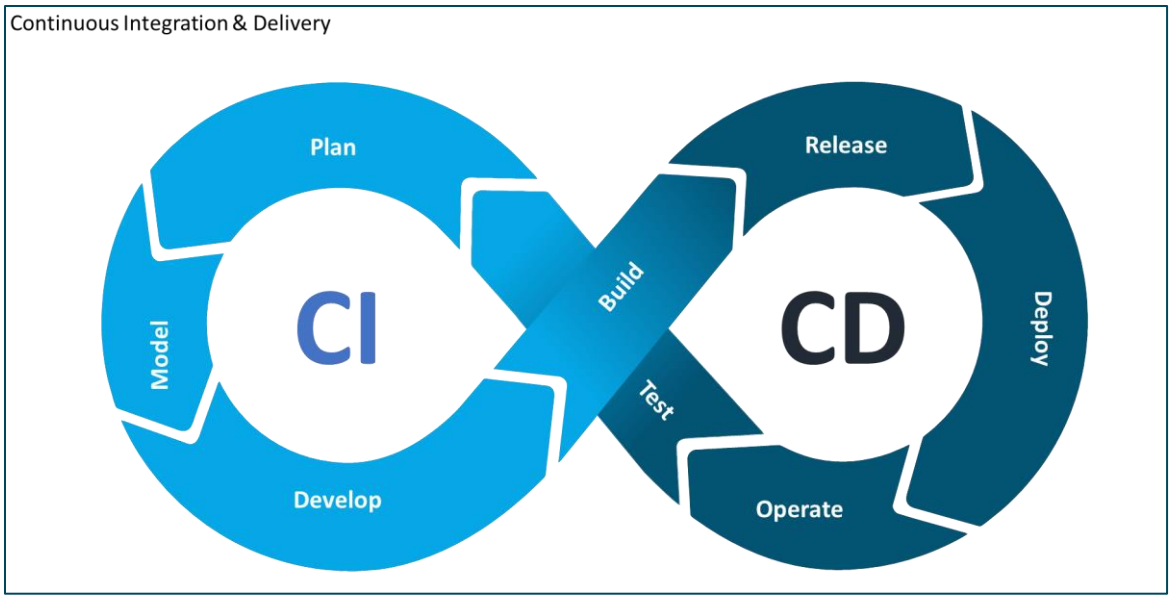

Abbildung 11: Continuous Integration & Delivery

Vereinfacht ausgedrückt ist CI/CD eine Vorgehensweise, bei der Quellcode in iterativen Wiederholungsschleifen (auch als Sprints bezeichnet) entwickelt und deployed wird. Details zu dieser Vorgehensweise finden Sie im Kapitel 3.2 zu den Methoden.

Um CI/CD anzuwenden, werden in aller Regel spezielle CI/CD-Werkzeuge genutzt. Zu den populärsten Vertretern gehört sicherlich das Open-Source-Projekt Jenkins (siehe Abbildung 12). Daneben wächst aber auch stetig das Angebot anderer Anbieter – z. B. mit Atlassian Bamboo, GitLab CI/CD, JetBrains Teamcity und vielen mehr.

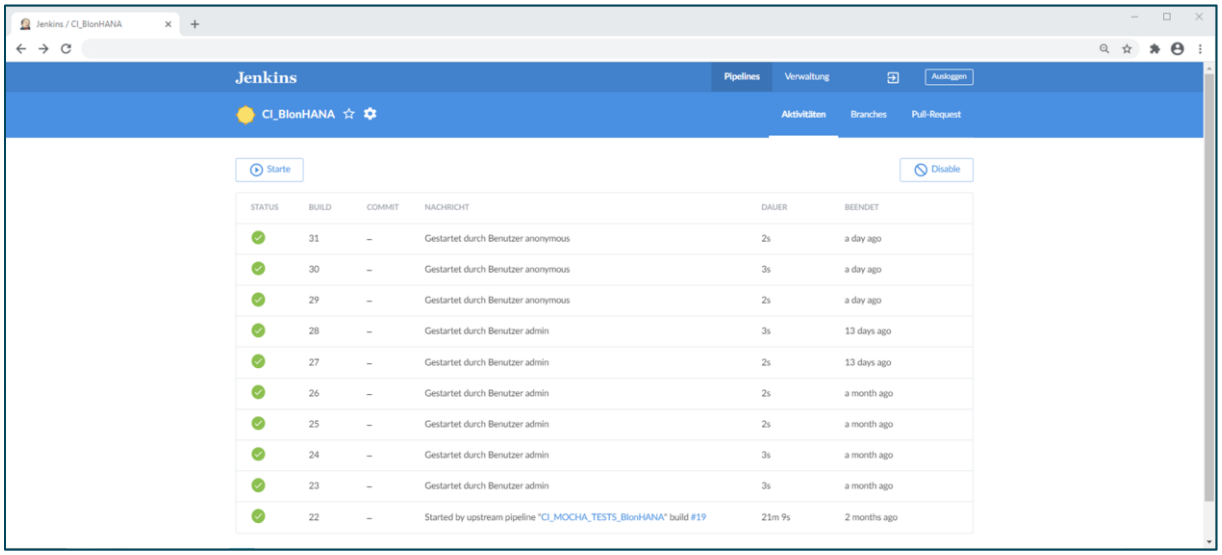

Abbildung 12: Screenshot von Jenkins

Die Funktionsweise ist bei all diesen Werkzeugen ähnlich: Prozesse, die im Sinne der Continuous Integration und Continuous Delivery automatisiert stattfinden sollen, werden entweder mittels einer grafischen Oberfläche (siehe Abbildung) oder als ausführbare Skripte erstellt. Das CI/CD-Werkzeug liefert hierfür zum einen die Entwicklungsumgebung, zum anderen aber auch die Infrastruktur für das Ausführen und Monitoren der Automationsprozesse.

Einige Beispiele für Automationsprozesse im HANA-SQL-DWH-Kontext sind:

- Designtime-Tests: Neu entwickelte Objekte werden auf Entwicklungskonventionen geprüft (Namenskonventionen eingehalten, Datentypen korrekt etc.)
- Build-Tests: Neu entwickelte Objekte werden auf ihre Lauffähigkeit getestet
- Deployment: Basierend auf dem zentralen Git-Repository wird das DWH automatisch deployed
- Runtime-Test: Runtime-Objekte werden auf inhaltliche Konsistenz geprüft (z. B. Anzahl Datensätze, Durchschnittswerte etc.)

3.1.8 Core Data Services

Persistenzmodelle im SAP-HANA-SQL-DWH-Kontext werden mittels Core Data Services (CDS) definiert. CDS ist eine von SAP entwickelte Infrastruktur für die Definition von Modellpersistenzen und bietet als solche vielfältige Möglichkeiten, Datenobjekte zu definieren und weiterzuentwickeln.

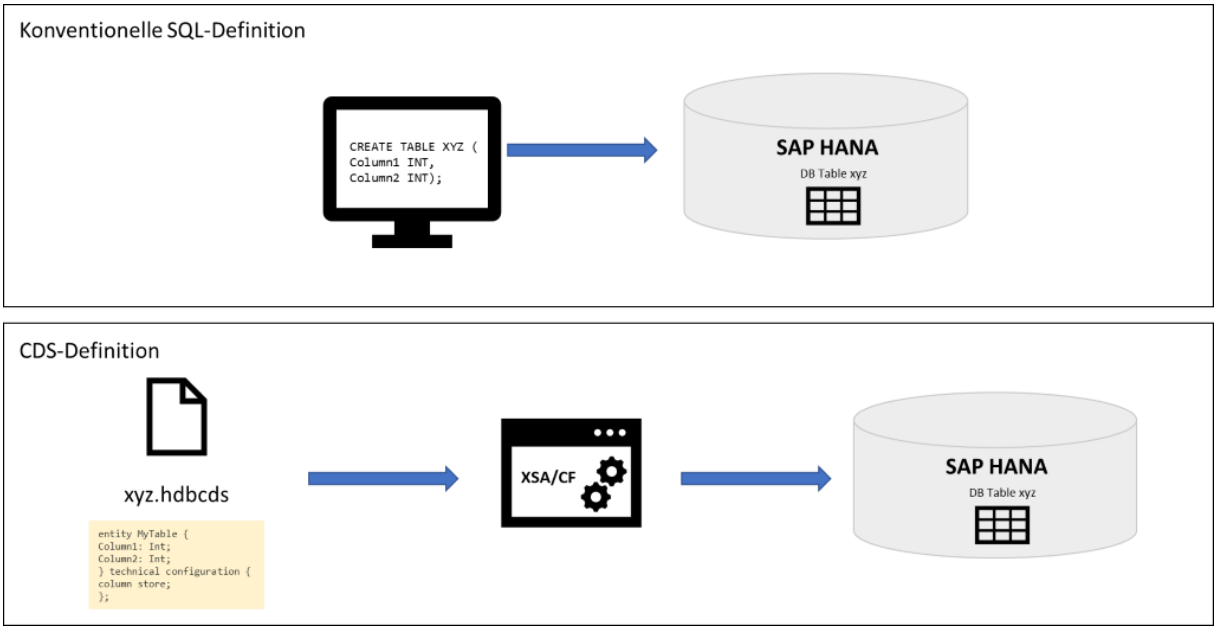

Abbildung 13: Unterschied SQL vs. CDS

CDS liefert dabei den besonderen Vorteil, dass Persistenzmodelle nicht – wie im konventionellen SQL-Ansatz – manuell in die Datenbank programmiert werden müssen.

Stattdessen werden sogenannte CDS-Designtime-Objekte erstellt (siehe Abbildung 13). Diese Designtime-Objekte übersetzt die HANA-Datenbank in native SQL-Statements, die sie dann ausführt. Diese Vorgehensweise erweist sich als besonders mächtig, wenn Änderungen an einem Modell vorgenommen werden oder wenn ein Modell auf ein anderes System transportiert wird. Das CDS-Modell stiftet als Design-time-Äquivalent zum tatsächlichen Modell auf der Datenbank nicht nur maximale Transparenz und Auditierfähigkeit, sondern auch eine hohe Flexibilität bei der Weiter-entwicklung.

Um Entwicklern entgegenzukommen, die Datenmodelle nicht gerne programmieren möchten, bietet SAP verschiedene Möglichkeiten an, CDS-Modelle grafisch zu entwik-keln. Dies ist zum Beispiel mithilfe des SAP PowerDesigners möglich – oder direkt über den CDS Editor in SAP Web IDE.

3.1.9 Git

Git ist ein System für die Versionsverwaltung von Quellcode. Es wurde bereits 2005 veröffentlicht und ist in der Software-Entwicklung schon seit Jahren unerlässlich. Ver-einfacht ausgedrückt wird mit Git der Quellcode für ein Projekt, in Git auch Repository genannt, verwaltet und historisiert.

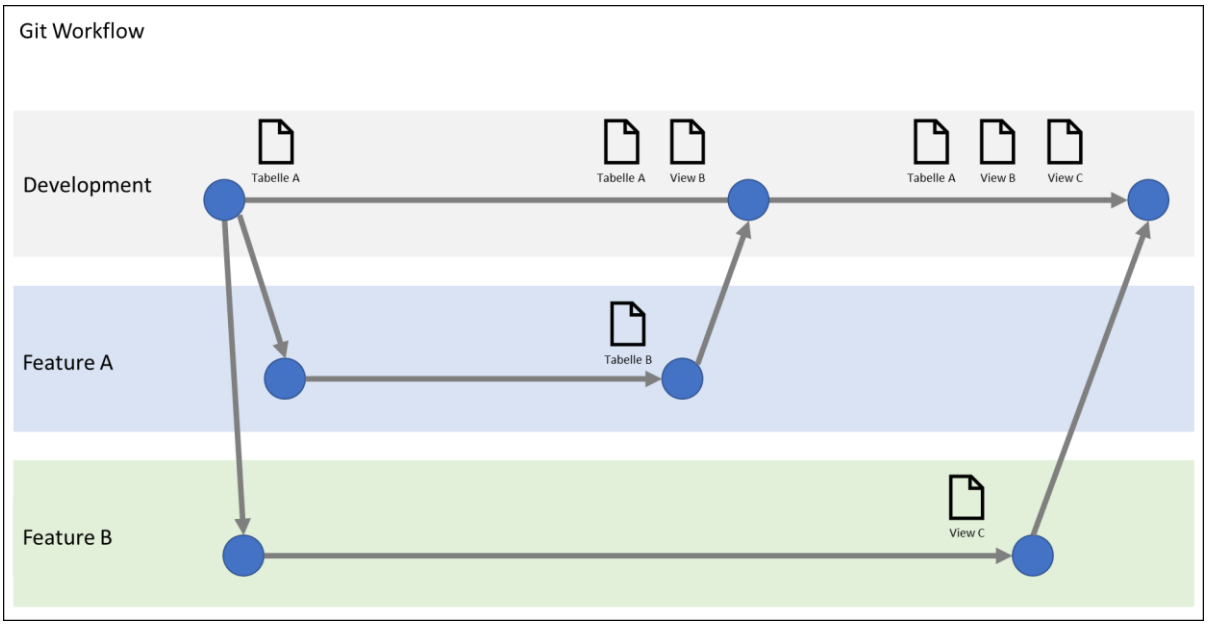

Abbildung 14: Vereinfachte Darstellung von Git

Die wichtigsten Funktionalitäten, die Git hierzu verwendet, sind Commits und Bran-ches. **Commits** (Kreise in der Abbildung) sind Speicherstände des Quellcodes zu ei-nem bestimmten Zeitpunkt. Der Quellcode entwickelt sich sozusagen von Commit zu Commit immer weiter. Branches (Development, Feature A, Feature B in der Abbildung)

sind notwendig, um mehrere Entwicklungen an ein und demselben Repository gleichzeitig voranzutreiben. Dies ist beispielsweise der Fall, wenn mehrere Bestandteile des Data-Warehouse, in Git auch Features genannt, gleichzeitig entwickelt werden und wenn mehrere Entwickler gleichzeitig an diesen Features arbeiten. In Abbildung 14 werden über die Branches Feature A und B die Views A und B erzeugt, die beide auf der Tabelle A basieren. Beide Objekte werden unabhängig voneinander entwickelt und anschließend wieder in die Branch-Development überführt, die als Resultat alle drei Objekte Tabelle A, View A und View B beinhaltet. Diesen Überführungsvorgang bezeichnet man auch als „Merge“.

3.2 Entwicklungsmethodik

In Kapitel 2 wurde bereits erwähnt, dass ein essenzieller Unterschied zu anderen Data-Warehouses darin besteht, dass das HANA SQL DWH über eine Applikationsschicht namens XSA verfügt. Dank dieser Applikationsschicht ist es möglich, das DWH auf Basis von Designtime-Objekten zu entwickeln, die erst durch die XSA (als Pseudo-Compiler) zu Laufzeitobjekten in der HANA-Datenbank werden.

Auf dieses Designtime-/Runtime-Paradigma ist die Software-Programmierung (im Gegensatz zur DWH-Entwicklung) seit jeher ausgerichtet. Mit Git und den CI/CD-Werkzeugen in Kapitel 3.1.9 und 3.1.7 wurden bereits Best-Practice-Entwicklungswerkzeuge vorgestellt, die aus der Software-Programmierung entlehnt sind.

Doch auch in Bezug auf die Entwicklungsmethodik lassen sich bewährte Konzepte für das DWH adaptieren. Eine besonders empfehlenswerte Methodik ist CI/CD.

In Kapitel 3.1.7 wurde das Thema CI/CD (siehe Abbildung 15) im Kontext der verschiedenen Werkzeuge betrachtet, die für dessen Anwendung zur Verfügung stehen. Wie dort bereits erwähnt, stellt CI/CD eine Entwicklungsmethodik dar, bei der Quellcode in iterativen Wiederholungsschleifen entwickelt und deployed wird (vgl. Abbildung 15). Dabei fasst Continuous Integration die Schritte zusammen, die üblicherweise zur Entwicklung gehören (Planen, Modellieren, Implementieren). Continuous Delivery hingegen umfasst die Schritte, die für das Deployen notwendig sind (Build, Release, Deploy, Operate, Monitor).

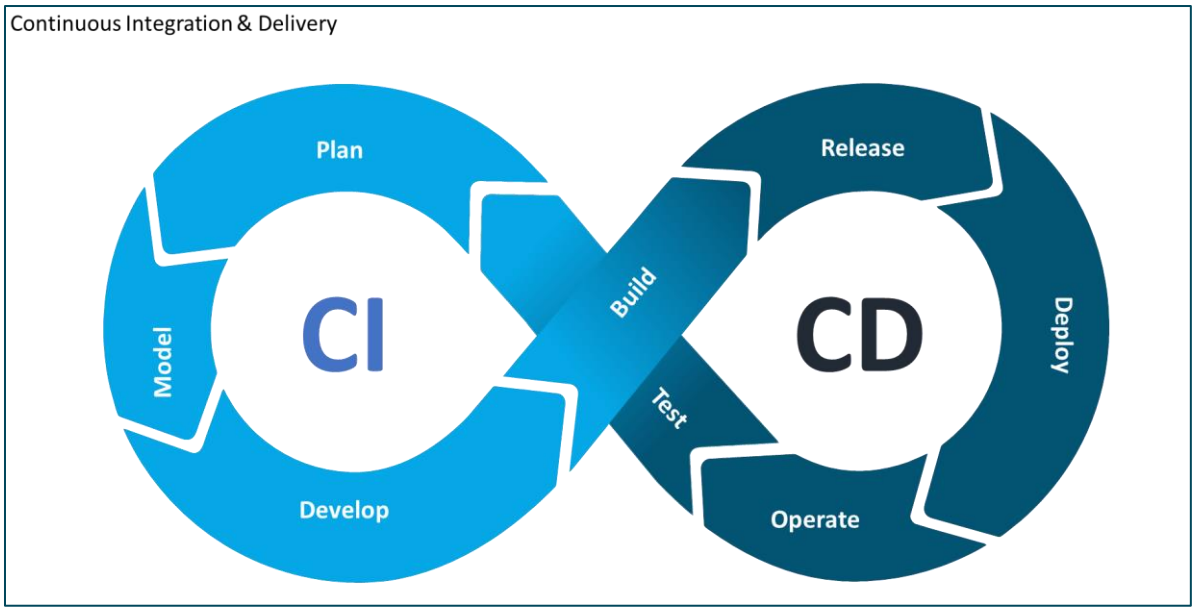

Abbildung 15: Continuous Integration & Delivery

In einer CI/CD-Schleife (auch als Sprint bezeichnet) wird in der Regel ein möglichst kleinteiliges Arbeitspaket bearbeitet – und nicht das gesamte Data-Warehouse. Beispielsweise könnten in einem Arbeitspaket nur die Business-Entität-Kunden bearbeitet werden – und alle dazu benötigten Data-Warehouse-Artefakte, wie Tabellen, ETL-Strecken und Views. Jede CI/CD-Schleife beginnt mit der Planung als ersten Schritt.

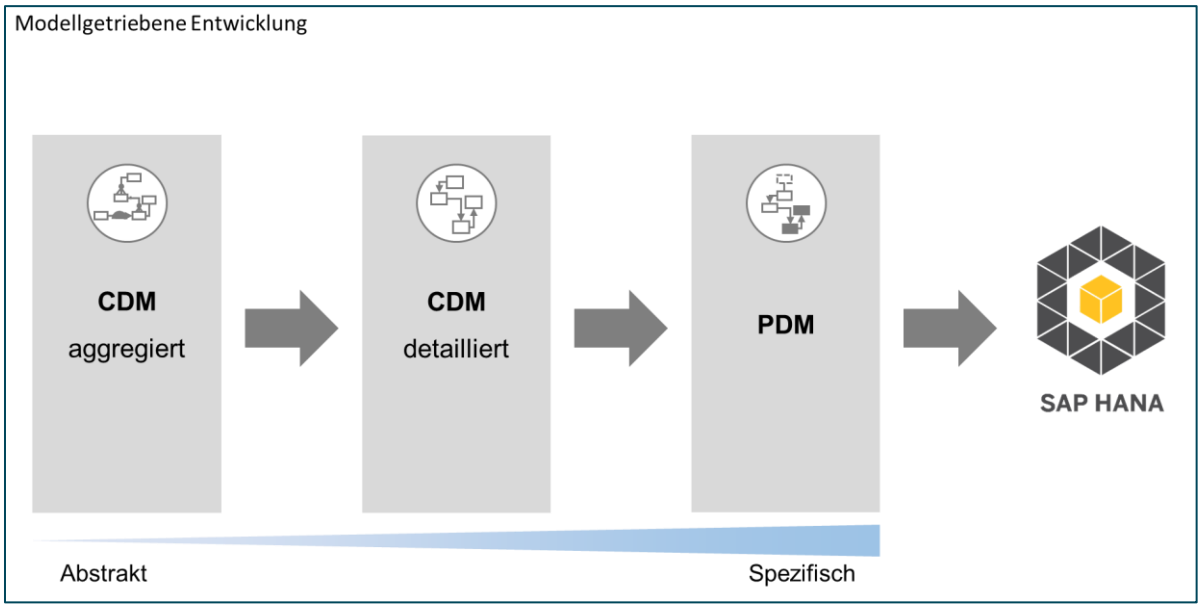

Abbildung 16: Modellgetriebene Entwicklung

Der Planung sollte dabei eine besondere Bedeutung zukommen. Hier empfiehlt es sich, die Business-Anforderungen für das Arbeitspaket in einem konzeptionellen Datenmodell (CDM) zu sammeln (siehe Abbildung 16), welches für alle an der Entwicklung des DWH Beteiligten die „Single Source of Truth“ darstellt. In diesem Zusammenhang wird auch von modellgetriebener Entwicklung gesprochen. Erst auf Basis dieses CDM sollte ein physisches Datenmodell (PDM) erstellt werden, das die technische Blaupause für die benötigten Datenbankartefakte darstellt und damit die Arbeitsgrundlage für weitere Implementierungsschritte.

Eine CI/CD-Schleife besteht somit aus folgenden einzelnen Schritten:

1. Plan: Anforderungen werden gesammelt und in einem konzeptionellen Datenmodell festgehalten.

2. Model: Auf Grundlage des konzeptionellen Datenmodells werden die konkret benötigten Datenbankartefakte in Form eines physischen Datenmodells erstellt.

3. Develop: Die Datenbankartefakte werden in den Entwickler-Sandboxes implementiert und getestet.

4. Build: Die Einzelentwicklungen werden zusammengeführt und als Bündel erneut auf Lauffähigkeit geprüft.

5. Release: Der neue Entwicklungsstand wird mit dem bereits vorhandenen Stand zusammengeführt. Somit entsteht ein neues Release.

6. Deploy: Das neue Release wird deployed.

7. Operate: Der neue Stand des DWH wird in Betrieb genommen und für User zugänglich gemacht.

8. Test: User testen den neuen Stand und formulieren Fehler und/oder neue Anforderungen.

Ist eine CI/CD-Schleife durchlaufen, beginnt – im Sinne des iterativen Charakters – direkt eine neue Schleife. Die Grundlage für die neue Schleife stellt dann entweder eine Liste mit Fehlern in einem ausgelieferten Arbeitspaket oder eine Liste mit Anforderungen für ein neues Arbeitspaket dar.

3.3Sicherheit

BW-generierte Objekte
BW generiert u.a. Calculation-Szenarien, die auch außerhalb des BW genutzt werden können. Diese benötigen dann Zugriffsberechtigungen in der HANA, um z. B. in einem HDI-Context in einem Calculation-View weiter verwendet werden zu können.

Nativer SQL-Ansatz
Datenbankinhalte wie Tables, Views, Procedures etc. werden über SQL (Data-Definition-Language, kurz DDL) angelegt. Der Entwickler muss hier das Recht haben, ein Schema anzulegen oder in einem Schema Artefakte anzulegen. Konzept: Ein Administrator gibt einem Entwickler die Rechte, Artefakte anzulegen. Dann müssen die Artefakte über die Landschaften, z. B. Qualität und Produktion, verteilt werden. Hier sind dann entsprechende Berechtigungen bereitzustellen.

HANA Deployment Infrastructure (HDI)
Die HDI vereint drei Ansätze:

- Anlegen und Ändern der Tabellenstruktur (und anderer Datenbankartefakte)
- Verteilung der Inhalte in der Landschaft (Dev, Test, Prod) durch kontinuierliche Integration/kontinuierliche Verteilung über Versionskontrollwerkzeuge.
- Beschreiben der Sicherheit durch den Entwickler und/oder Sicherheitsbeauftragten in Form von Designtime-Rollen. Konzept: Jeder Entwickler kann HDI-Container anlegen. Ohne weitere Berechtigungsdefinierungen bekommt keiner den Zugriff außerhalb des Containers oder der Entwicklungsumgebung des jeweiligen Entwicklers. Es gilt das Konzept „Zero Privileges by Default“.

Um auf Objekte außerhalb des HDI-Containers zuzugreifen, müssen entsprechende Synonyme erzeugt werden. Zusätzlich benötig man für diesen Zugriff entsprechende Berechtigungen. Bei HDI hat jeder Entwickler seinen eigenen HDI-Container. Er kann dort alle Artefakte anlegen, ohne bei der Aktivierung mit parallelen Entwicklern zu konkurrieren. Vereint werden die Artefakte über einen Austausch mit einem Versionswerkzeug, wie z. B. GIT.

Global-Risk-and-Compliance (GRC) für die Berechtigung nutzen
Im Sinne einer übergeordneten Sicherheit in einem Unternehmen kann man die Kontrolle der Berechtigung auch an zentraler Stelle in GRC hinterlegen. GRC kann dann für die jeweilige HANA-Instanz Rechte sowohl in der HANA als auch zum Teil im XSA zuweisen.

4. Use-Cases

Im folgenden Kapitel beschreiben wir einige Use-Cases für SAP HANA SQL DWH.

4.1 AOK Baden-Württemberg: Natives SAP HANA SQL Data-Warehousing

Die AOK Baden-Württemberg betreut als gesetzliche Krankenversicherung rund 4,5 Millionen Versicherte und verfügt über ein Haushaltsvolumen von rund 18 Milliarden Euro. Seinen Hauptsitz hat das Unternehmen in Stuttgart.

Das Referat Business Intelligence der AOK Baden-Württemberg ist betraut mit der permanenten Verbesserung der Analyse- und Controlling-Fähigkeit des Unternehmens, der Bereitstellung der hierfür erforderlichen Instrumente, Werkzeuge und Datengrundlagen in standardisierter und strukturierter Form sowie mit der Festlegung einer an der Unternehmensstrategie ausgerichteten BI-Strategie.

Im Jahr 2017 hat sich das Referat Business Intelligence der AOK Baden-Württemberg dazu entschieden, das bestehende, in die Jahre gekommene Data-Warehouse abzulösen und zu ersetzen. Die Motivation für eine Umstellung bestand dabei hauptsächlich in den folgenden Punkten:

- Technologische Angleichung an die Hauptquelle (oscare BW)
- Ablösung des bestehenden Oracle-basierten DWH
- Reduktion der persistenten Datenhaltung
- Performance-Optimierung

Bei der Auswahl eines geeigneten Nachfolge-DWH fiel die Wahl auf das SAP HANA SQL DWH, da dieses die oben genannten Motivationspunkte ohne Einschränkungen bedient. Darüber hinaus bildet das SAP HANA SQL DWH mit seiner Architektur, den Systemkomponenten und der Entwicklungsmethodik den Startpunkt für eine agile, offene Architektur, die vorbereitet ist für Erweiterungen, die zum Teil heute noch nicht bekannt sind. Wobei die generelle Richtung mit Themen wie Big-Data-Integration, Data-Science-Applikationen und Cloud-Orientierung durchaus abzusehen ist.

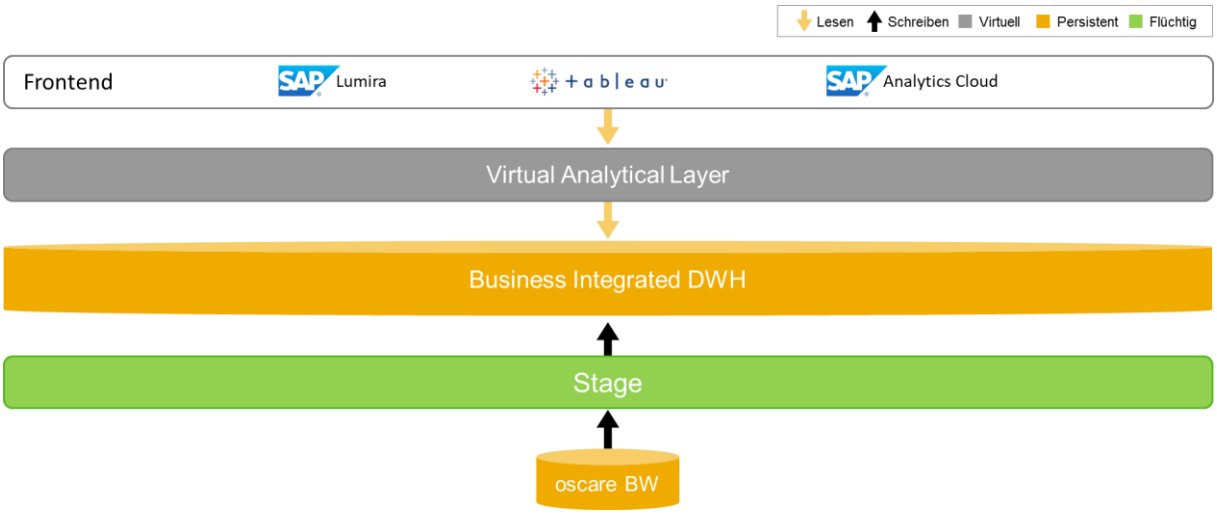

Abbildung 17: Multi-Layer-DWH-Architektur

Das bei der AOK Baden-Württemberg eingesetzte SAP HANA SQL DWH baut auf eine klassische Multi-Layer-DWH-Architektur, die für die HANA-Plattform optimiert ist (siehe Abbildung). Zentraler Baustein dieser Architektur ist das Business-Integrated-DWH (kurz BID), in dem die klassischen Aufgaben eines Data-Warehouse (wie Integration, Historisierung etc.) wahrgenommen werden.

Um hier eine möglichst hohe Flexibilität für die Zukunft sicherzustellen, erfolgt die Modellierung im BID nach dem Data-Vault-Ansatz (mehr dazu unten im Text). Der Zugriff auf die Daten erfolgt über den Virtual-Analytical-Layer. Hier werden virtuelle Sichten auf die persistierten Daten erstellt, um Anforderungen für unterschiedliche Analyseszenarien zu bedienen. Dies führt zu einer deutlichen Reduktion der persistierten Daten – bei gleichzeitig erhöhter Abfrage-Performance.

Als Frontend-Werkzeuge kommen Tableau, SAP Lumira und künftig SAP Analytics Cloud zum Einsatz. Perspektivisch sollen auch Data-Science-Werkzeuge wie R und Python zum Einsatz kommen.

Die Hauptquelle für das bestehende und auch für das neue DWH bildet die SAP-Erweiterung oscare. Dabei handelt es sich um eine integrative IT-Plattform der AOK Systems GmbH, die auf der SAP Business Suite basiert und sämtliche Geschäftsprozesse der gesetzlichen Krankenversicherung abbildet. Die Schnittstelle zu oscare bildet ein integriertes SAP-Business-Warehouse (oscare BW).

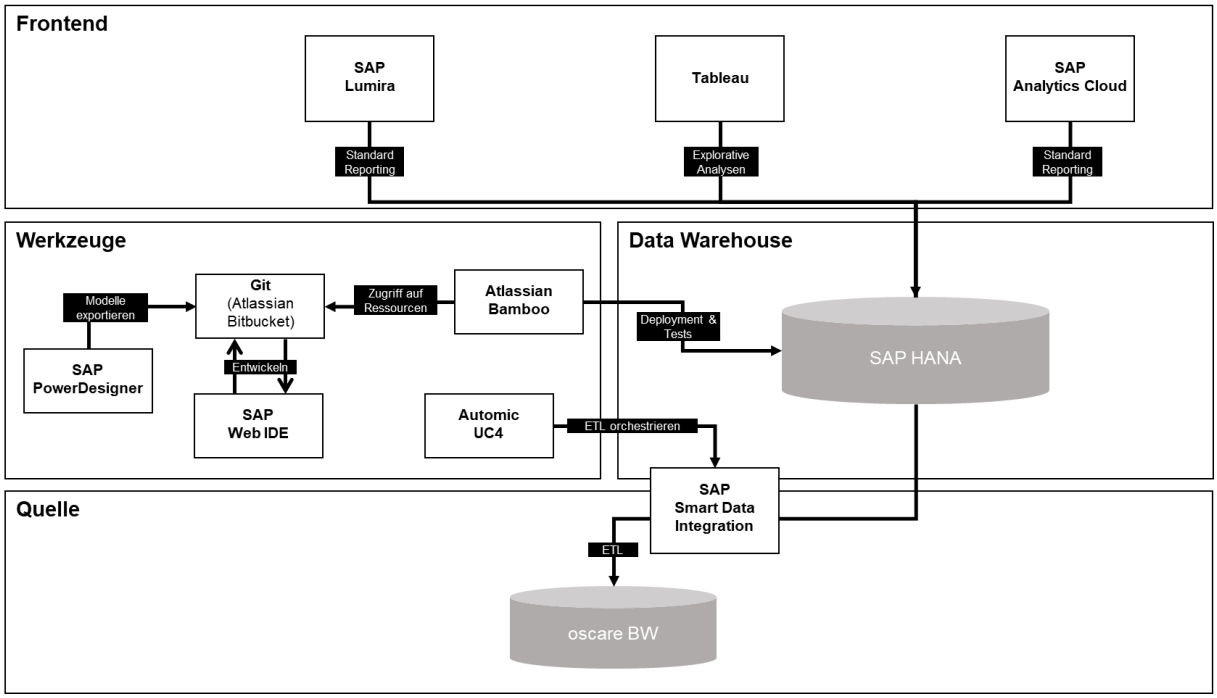

Abbildung 18: Grobübersicht der Systemlandschaft

Neben der SAP-HANA-Plattform nutzt das SAP HANA SQL DWH der AOK Baden-Württemberg eine Reihe weiterer Werkzeuge (siehe Abbildung 18). Zu den wichtigsten Werkzeugen gehören dabei:

- **SAP PowerDesigner**: Wird genutzt für die Modellierung des Data-Warehouse und nimmt somit unter den Werkzeugen eine zentrale Rolle ein.
- **SAP Web IDE**: Das zentrale Entwicklungswerkzeug, mit dem z. B. Calculation-Views und SDI-Flowgraphs entwickelt werden.
- **SAP HANA Studio**: Wird für das Administrieren der SAP-HANA-Datenbanken genutzt.
- **SAP Smart Data Integration**: Wird als ETL-Werkzeug genutzt, mit dem die Datenquellen angesprochen werden, um die Daten in das DWH zu übertragen.
- **Atlassian Bitbucket**: Wird als Git-Server genutzt.
- **Atlassian Bamboo**: Wird als Automations-Server genutzt, mit dem Deployments und Tests automatisch ausgeführt werden.
- **Automic UC4**: Wird für die Orchestrierung von Beladungsprozessen verwendet.

Bei der Entwicklung des SAP HANA SQL DWH wendet die AOK Baden-Württemberg einen modellgetriebenen Ansatz im Sinne der CI/CD-Methodik (siehe Kapitel 3.1.7) an. Das heißt, die zu entwickelnden Datenbankartefakte werden mit Hilfe von Modellen definiert und erst dann implementiert. Dabei kommen sowohl konzeptionelle Datenmo-

delle zum Einsatz, in denen semantische Informationen über die benötigten Daten gesammelt werden, als auch physische Datenmodelle, die als technische Blaupause für die Datenbank genutzt werden. Für die Modellierung der physischen Datenmodelle für das SAP HANA SQL DWH setzt die AOK Baden-Württemberg die Data-Vault-Methodik ein – eine Modellierungsmethodik, die genau für den Zweck des Data-Warehousing konzipiert wurde und sich durch hohe Flexibilität und vereinfachte Historisierungsmöglichkeiten auszeichnet.

Mit der vorgestellten Architektur, der Entwicklungsmethodik sowie den verwendeten Werkzeugen beweist sich das SAP HANA SQL DWH bei der AOK Baden-Württemberg als äußerst flexibles System, das schnell und unkompliziert auf Veränderungen in den Quellsystemen reagieren kann. Des Weiteren zeigt es dank der virtuellen Analyseschichten eine hohe Abfrage-Performance und einen optimierten Speicherbedarf, der auf die Kompressions- und Virtualisierungsfähigkeiten der HANA-Plattform zurückzuführen ist.

4.2 Hybrides Szenario mit einem SAP HANA SQL DWH und einem SAP BW/4HANA

Die Bechtle AG mit Sitz in Neckarsulm ist das größte IT-Systemhaus in Deutschland. Sie bietet IT-Dienstleistungen an sowie den Vertrieb von IT-Produkten. Mit ca. 100 Tochtergesellschaften ist die Bechtle Gruppe europaweit tätig. Durch die Vielzahl von Unternehmen hat sich im Laufe der Zeit eine heterogene, verteilte und redundante Datenlandschaft entwickelt (siehe Abbildung 19).

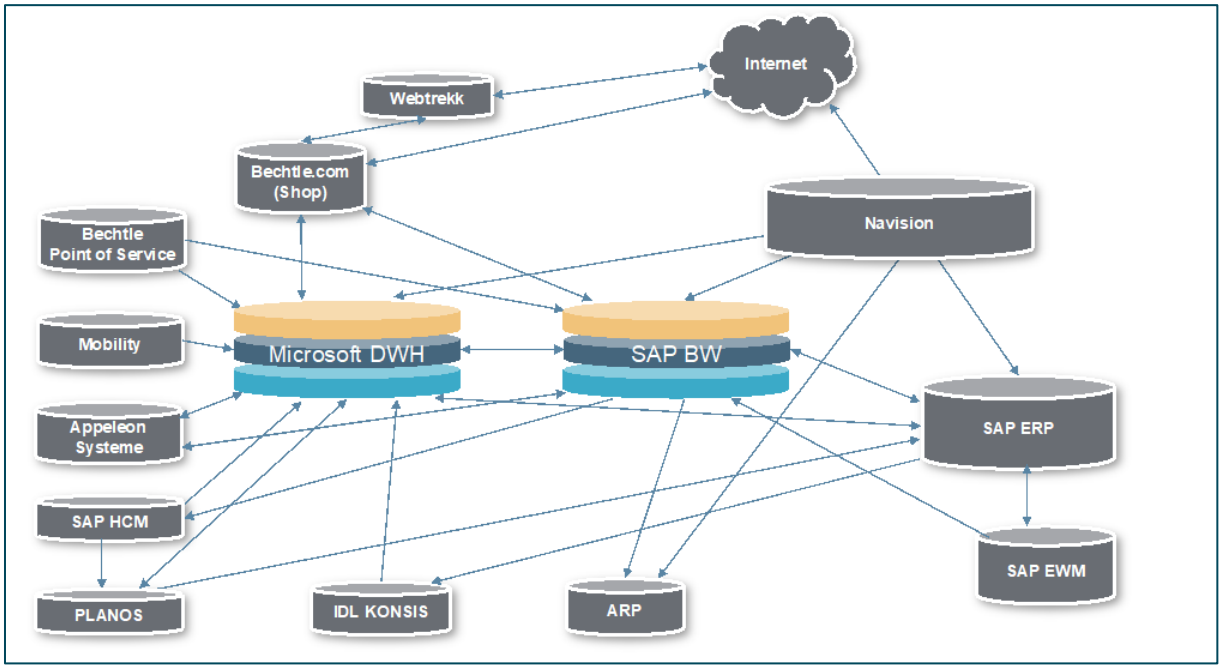

Abbildung 19: Ausgangslage Bechtle vor Einführung eines SAP HANA SQL DWH

Da eine solche IT-Landschaft schwer zu betreiben ist, hat sich die Bechtle Gruppe das Ziel gesetzt, ein konsolidiertes DWH einzuführen. Diese konsolidierte DWH-Lösung soll alle Reporting-Anforderungen der Bechtle Gruppe abdecken. Eine besondere Herausforderung dabei ist es, ein eindeutiges Informationsmodell mit gemeinsamen Stammdaten und ein einheitliches Kennzahlenmodell für mehr Transparenz und Konsistenz in Auswertungen und Vergleichen zu schaffen. Eine konsolidierte und validierte Datensicht an zentraler Stelle schafft Sicherheit und Verlässlichkeit. Die Abbildung der Anforderungen aller Fachbereiche sowie das Einbeziehen und Verknüpfen der einzelnen Geschäftsmodelle ermöglicht einen ganzheitlichen Blick auf die gesamte Wertschöpfungskette. Da die Anforderungen sehr divers sind, kann ein einziges Frontend-Werkzeug nicht allen Anforderungen genügen. Aus diesem Grund muss sichergestellt sein, dass bei der Bereitstellung verschiedener Frontends für unterschiedliche Anforderungen auf gleicher Datenbasis gearbeitet werden kann.

Die Bechtle Gruppe hat sich entschieden, ein BW/4HANA mit einer „Mixed Architecture“ einzuführen (siehe Abbildung 20). Als Mixed Architecture wird eine Architektur beschrieben, die sich aus einen SAP BW/4HANA und einem SAP HANA native SQL DWH zusammensetzt. Auf diese Weise werden Vorteile aus beiden Welten vereint. Daten, die aus SAP-Quellen kommen, können via BICS-Schnittstelle komfortabel in das BW/4HANA persistiert werden. Außerdem kann das bereits vorhandene BW-Know-how in der Unternehmensgruppe genutzt werden. Die geführte Modellierung in einem BW/4HANA-System vereinfacht zudem die Abbildung von fachlichem Wissen in Form von Infoobjekten, die es erlauben, Metainformationen wie Hierarchien und Zeitabhängigkeiten zu pflegen. Die SAP-Schnittstellen ermöglichen einen einfachen Abruf der Daten aus dem BW/4HANA durch andere SAP-Systeme und -Frontends. Mit dem HANA SQL DWH werden Nicht-SAP-Quellen eingebunden sowie Nicht-SAP-Frontends bedient. Die SQL-Schnittstelle ist universell und von jeder anderen Anwendung nutzbar. Somit wird die Erweiterung mit Third-Party-Anwendungen deutlich vereinfacht. Auch an dieser Stelle war bereits SQL-Know-how in der Unternehmensgruppe vorhanden.

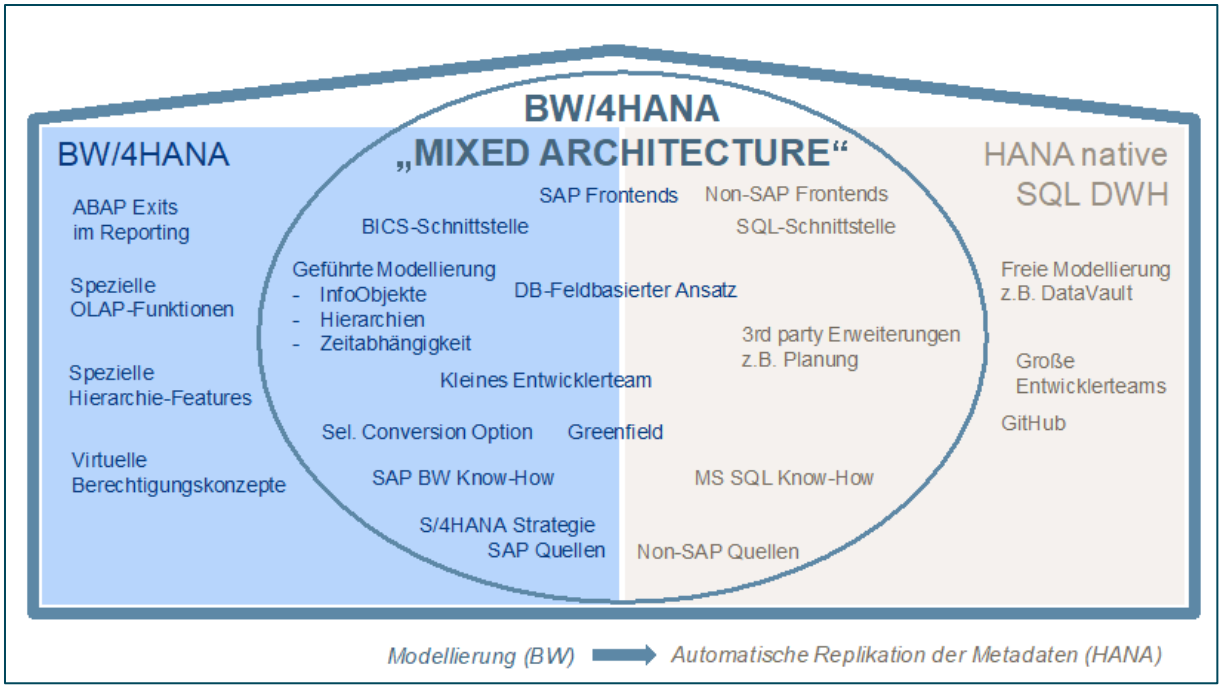

Abbildung 20: Bechtles Argumente für eine Mixed Architecture

Die Bechtle Gruppe betreibt ihr BW/4HANA und HANA SQL DWH auf der gleichen HANA Enterprise Cloud. Auf diese Weise können Calculation-Views basierend auf BW/4HANA-Objekten direkt in das HANA SQL DWH generiert werden. Somit kann sichergestellt werden, dass die Informationen aus dem BW/4HANA sich auch genauso im HANA SQL DWH befinden oder virtuell verfügbar sind. Auch werden die Nutzer und Berechtigungen von dem BW/4HANA als HANA-User generiert und erzwingen somit

keine doppelte Pflege in beiden Systemen. Innerhalb des HANA SQL DWH werden nun die Daten aus dem BW/4HANA mit weiteren Daten veredelt und für weitere Zwecke bereitgestellt. Auf diese Weise können sowohl SAP-Anwendungen, wie Lumira oder Analysis for Office, als auch Third-Party-Anwendungen, wie Tableau, auf die gleiche Datenbasis zugreifen. Die Calculations-Views sind von allen Frontend-Anwendungen konsumierbar, weil sie einen SQL-Zugriff erlauben. Abbildung 21 zeigt den beschriebenen Aufbau der „Mixed Architecture“ bei der Bechtle Gruppe.

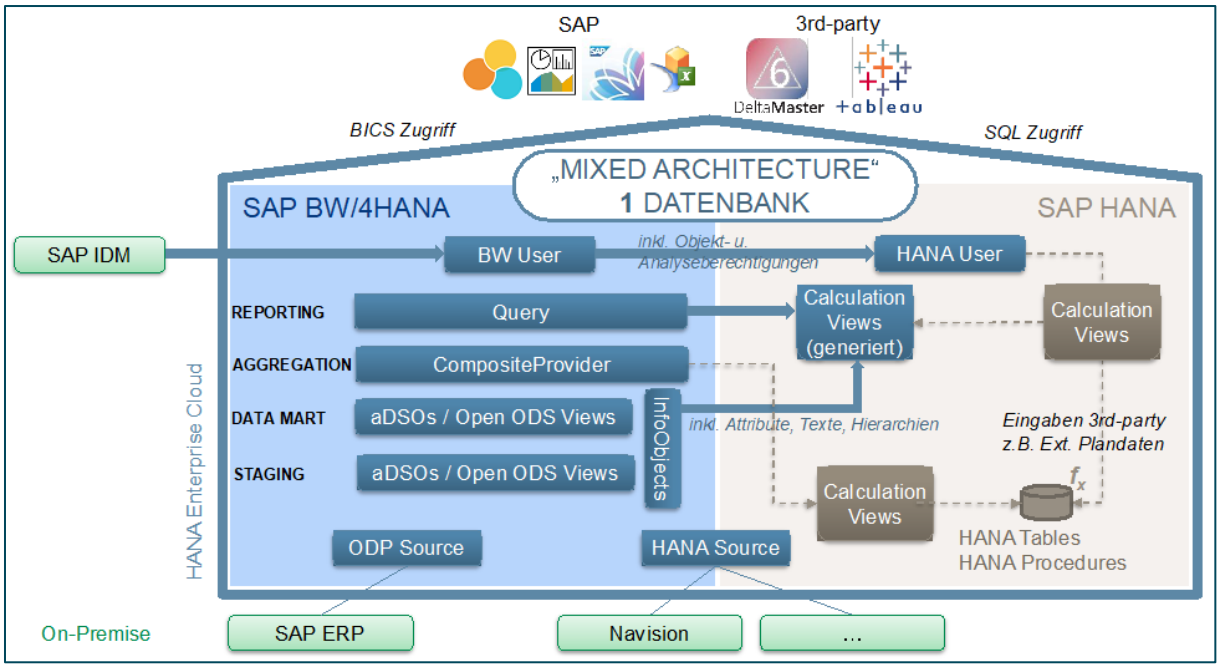

Abbildung 21: Beziehungen zwischen Komponenten aus der Mixed Architecture von Bechtle

Siehe auch:

https://www.dsag.de/dokumente/v167-tag-3-business-analytics-kinzel-bechtle-dr-seitz-prodato-integration-technology

4.3 Migration eines SQL-Data-Warehouse in die HANA-Datenbank

SQL-Data-Warehousing mit einer Layer-Architektur, Fakten, Dimensionen, Star-Schema, einer relationalen Datenbank und ETL-Prozessen gibt es bereits seit vielen Jahrzenten. Solche Systeme werden außerhalb der SAP-Welt erstellt. Wenn SAP oder SAP BW vorhanden sind, können diese Systeme als mögliche weitere Quellsysteme eingebunden werden. Ansonsten erfolgt die Datenextraktion aus einem der vielen anderen operativen Systemen. All diese SQL-Data-Warehouses sind in einer Datenbank entstanden wie Oracle, IBM DB/2, MS SQLServer, Teradata etc.

Mit der Einführung von HANA stellt sich für viele SAP-Anwender die Frage, ob der vorhandene Datenbank-Zoo weiterhin betrieben werden muss, oder ob man diese Funktionalität in einer HANA-Datenbank abbilden kann. Die Antwortet lautet: Ja, es ist möglich, ein SQL-Data-Warehouse in der HANA-Datenbank zu erstellen, und es ist auch möglich, ein bestehendes Data-Warehouse in die SAP-HANA-Datenbank zu migrieren und dieses darin zu betreiben. In dem vorliegenden Artikel wird beschrieben, welche Schritte notwendig sind, ein bestehendes Data-Warehouse mit einer Layer-Architektur zu migrieren. In Abbildung 22 ist das Ergebnis der Ist-Analyse dargestellt. Auf dieser Basis wird eine Zielarchitektur entworfen und das bestehende Data-Warehouse stufenweise in die SAP-HANA-Datenbank integriert.

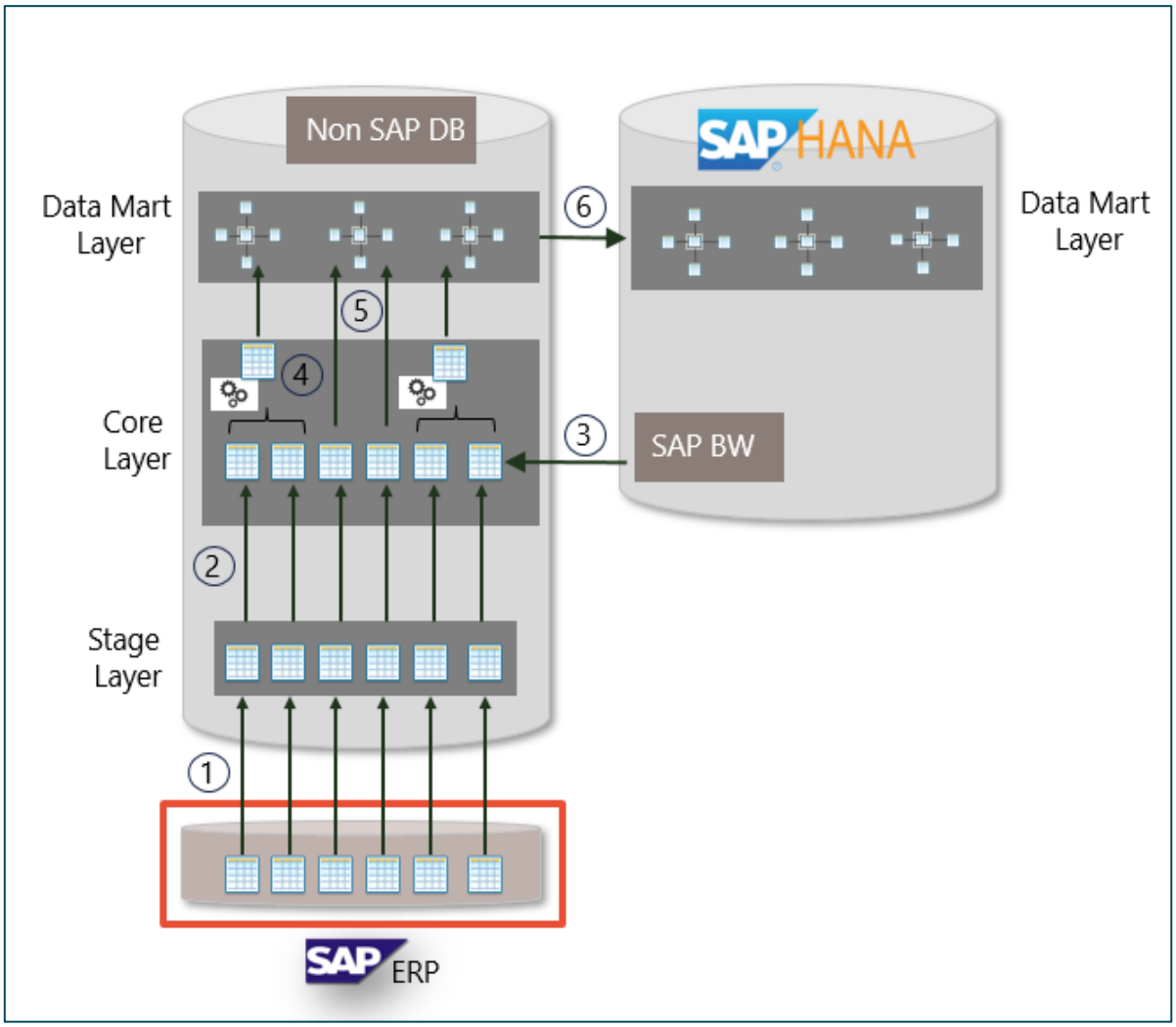

Abbildung 22: Data-Warehouse vor der Migration

1. Extraktion der SAP-Rohdaten mit IBM Data Manager SAP Connector
2. Ermittlung des Deltas seit dem letzten erfolgreichen Lauf. Geänderte oder neue Datensätze werden identifiziert und im Core-Layer mit Datenbank-Prozeduren zusammengeführt.
3. Extraktion der Hierarchien aus SAP BW mit dem IBM Data Manager.
4. Anreicherung der Fakten mit Geschäftslogik.
5. Überführung der Daten in Star-Schemas (Fakten, Dimensionen) mit Datenbank-Prozeduren.
6. Die Data-Marts werden bereits mit SAP Data Services in die bestehende HANA-Datenbank kopiert.

4.3.1 Ist-Analyse

Wir starteten die Ist-Analyse mit folgenden Fragestellungen: Welche Layer befinden sich im Data-Warehouse und sind für die Zielstruktur relevant? Welche Quellsysteme sind aktuell am Data-Warehouse angebunden/integriert? Welche ETL-Tools und Sprachen werden für die Extraktion und Transformation verwendet? Auf dieser Basis wird die Zielarchitektur für die anschließende Implementierung entworfen.

	Ist-Situation	Ziel
Datenbank	Um welche Datenbank handelt es sich, in der aktuell das Data-Warehouse betrieben wird?	SAP-HANA-Datenbank
Quellsysteme	Welche Quellsysteme sind eingebunden? Welche Systeme sind für die Zielarchitektur relevant? Zum Beispiel SAP ERP, Siebel CRM, Sales Force etc.	Alle aktiven Quellsysteme müssen in die Zielarchitektur angebunden werden.
Schnittstellen	Aus der Liste der Quellsysteme, die in der Zielumgebung angebunden werden, ergibt sich die Anzahl der Schnittstellen.	Können die Schnittstellen in der Zielumgebung 1:1 übernommen werden oder sollten diese optimiert werden? Operative Systeme liefern oft unterschiedliche Möglichkeiten, die Daten auszulesen. Nach der Migration müssen die Schnittstellen für die nachfolgenden Systeme dieselben Daten und Strukturen aufweisen. Hierzu gehören Reporting-Tools und Schnittstellen, bei dem das Data-Warehouse als Datenlieferant dient.
ETL	Wie werden aktuell die Daten im DWH gelesen, transformiert und geschrieben? Welche ETL-Tools oder Sprachen werden hierfür verwendet? Je nach Quellsystem können für das Beladen des Stage-Layers unterschiedliche Technologien oder Tools zum Einsatz kommen.	Die Entscheidung über ETL-Tools, gekauft oder selbst entwickelt, muss sehr sorgfältig ausgearbeitet werden. Es handelt sich ähnlich wie bei der Datenbank um eine langfristige Entscheidung. In unserem Fall werden die Transformationen mit SQL-Views in der Datenbank implementiert und über SAP Data Services ausgeführt.

Layer-Architektur	Im Data-Warehouse befindet sich aktuell eine Drei-Layer-Architektur für die ETL-Prozesse: Stage-Layer Core-Layer Mart-Layer	Sollen alle Layer erhalten bleiben? Ist es sinnvoll, einen weiteren Layer hinzuzufügen?
Tabellen	Welche Strukturen befinden sich in der Datenbank?	Von allen relevanten Tabellen müssen die DDL-Skripte extrahiert, nach HANA transformiert und implementiert werden.
Datentypen	Jede Spalte in einer Tabelle hat einen definierten Datentyp. Alle relevanten Datentypen inkl. Precision und Scale müssen extrahiert werden.	Die bestehenden Datentypen müssen auf die bestmöglichen HANA-Datentypen gemappt werden.
Reporting	Einer der wichtigsten Schnittstellen ist die Zugriffsschicht für die Reports und Dashboards.	In unserem Scenario bleibt die Implementierung der Reports und Dashboards erhalten. Somit muss der Zugriff des BI-Servers auf die neue Datenbank sichergestellt werden.
Berechtigungen	Berechtigungen auf Daten können entweder in der Datenbank oder in der Reporting-Lösung implementiert sein.	In unserem Fall bleibt die bestehende Lösung im BI-Server erhalten.

4.3.2 Zielarchitektur

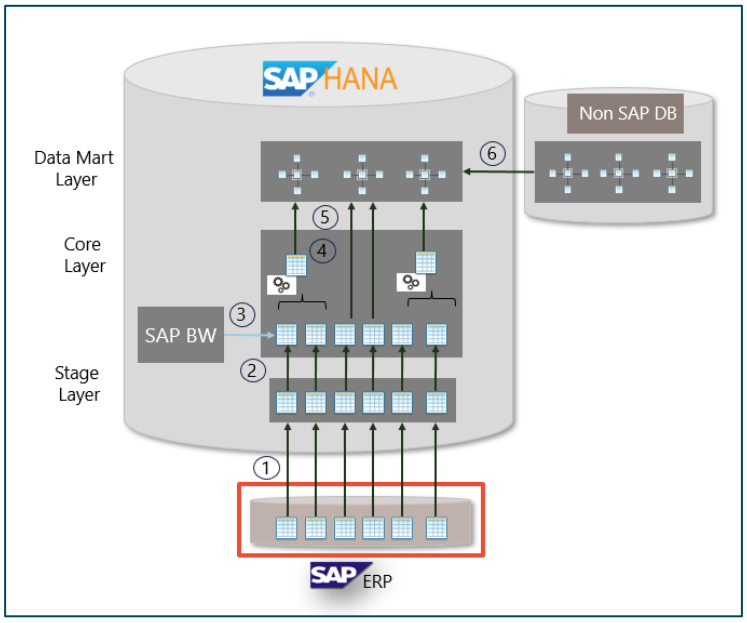

Abbildung 23: Zielarchitektur

1. Extraktion von SAP-Rohdaten mit SAP SLT.
2. Datentypkonvertierung.
3. Hierarchien werden innerhalb der HANA-Datenbank von SAP BW in das SQL DWH kopiert.
4. Implementierung der Geschäftslogik mit SQL-Skript/Views und SAP Data Services.
5. Aufbau von Dimensionen, Fakten und Beziehungen für ein Star-Schema.
6. Kopie der Data-Marts aus der alten abzulösenden Datenbank.

Datenmodell und Layer

Als Erstes muss das Datenmodell aus der bestehenden Datenbank extrahiert und transformiert werden, bevor diese Skripte in die HANA eingespielt werden können. Die Extraktion des bestehenden Modells kann auf unterschiedliche Weise geschehen. Mit dem PowerDesigner kann das bestehende Datenmodell über Reverse-Engineering ausgelesen und nach der Transformation in HANA eingespielt werden. Eine weitere Variante ist das manuelle Auslesen der Metadatentabellen der Datenbank per SQL oder über herstellerspezifische Funktionen. Unabhängig davon, welcher Weg gewählt wird, müssen die bestehenden Objekte aus der zu migrierenden Datenbank extrahiert und auf HANA SQL überführt werden. Obwohl sich alle Datenbankhersteller am SQL-Standard anlehnen, hat jede Datenbank doch ihre eigene Historie und somit auch Varianten in der Syntax und Bezeichnungen. Während der Extraktion müssen die verwendeten Datentypen inklusive Precision und Scale ausgelesen und der gewünschte Zieldatentyp definiert werden.

Extraktion von Rohdaten

(1) In diesem gewählten Szenario handelt es sich um ein SAP-ERP-System, das über SLT angebunden wird. Somit werden die Rohdaten zunächst unverändert in das SQL DWH repliziert. (2) Im nächsten Schritt werden diese Daten auf die Datentypen der HANA-Datenbank angepasst. Beispielsweise werden Daten vom Typ Datum als CHAR in der SAP-ERP-Datenbank gespeichert. Diese Daten werden zu SECONDDATE konvertiert. Der Standardwert für das Datum ist „00000000“ in der Quelle und muss auch entsprechend angepasst werden. In HANA sind im Datentype SECONDDATE nur Werte zwischen 0001-01-01 00:00:01 und 9999-12-31 24:00:00 möglich. Die Transformation erfolgt mittels SQL-Views.

Hierarchien aus SAP BW

(3) SAP BW ist für das SQL-Data-Warehouse eine weitere Datenquelle. Diese wird mit SQL ausgelesen und in das DWH übertragen. Ein weiteres ETL-Tool, wie in der Ausgangssituation beschrieben, kommt nicht mehr zum Einsatz.

Geschäftslogik und Star-Schema im Mart-Layer

(4+5) Die Aufbereitung der Daten für das Reporting erfolgt über Views in der Datenbank. Der Transport wie das Auslesen aus den Tabellen im Core DWH und das Schreiben im Mart-Layer erfolgt mit SQL-Script in der Datenbank oder mit dem ETL-Tool SAP Data Services.

Kopie Data-Mart

Zu Beginn der Migration des Data-Warehouse in die SAP-HANA-Datenbank wird ein Batchjob erstellt, der regelmäßig die in der abzulösenden Datenbank berechneten Data-Marts in die HANA kopiert. In diesem Fall erfolgt die Implementierung mit SAP Data Services.

Die regelmäßige Synchronisation der Data-Marts aus der bestehenden produktiven, abzulösenden Umgebung in die neue Entwicklungsumgebung ermöglicht eine frühzeitige Anbindung und einen frühzeitigen Test der bestehenden BI-Tools und Schnittstellen. In der Übergangsphase kann diese Funktionalität in die produktive Umgebung übernommen werden, ohne dass bereits alle Berechnungen in der HANA-Datenbank erfolgen müssen. Im weiteren Verlauf der Entwicklung und Testphase kann die Kopie der Data-Marts in der Entwicklungsumgebung auch für den maschinellen automatisierten Test herangezogen werden. Differenzen, die es bei einer Migration immer geben wird, müssen analysiert und bewertet werden. Nicht jede Abweichung ist zwingend ein Fehler bei den ETL-Jobs in der Zielumgebung.

Nach einer Phase der Stabilisierung kann der Parallelbetrieb auf das Data-Warehouse in der SAP-HANA-Datenbank reduziert werden und in den Regelbetrieb übergehen.

5. Fazit

Wir haben mit diesem Leitfaden einen Einblick in das SAP HANA SQL DWH gegeben und stellen wichtige Punkte und Aspekte heraus. In der Differenzierung zu anderen DWH-Produkten zeigt sich, dass SAP versucht, aktuellen Kundenanforderungen gerecht zu werden. Mit dem SAP HANA SQL DWH bietet SAP einen Ansatz, der jedem Kunde möglichst viel Freiheit gibt. Dies hat aber zur Folge, dass es von SAP kein Best Practice für die Implementierung eines DWH mit SAP HANA gibt. Auch für hybride Szenarien (mit SAP BW) gibt es keine Anleitung, und jeder Kunde muss individuell eine Architektur entwickeln. Moderne agile Entwicklungsmethoden werden durch viele Tools (SA-eigene oder Third Party) unterstützt und befriedigen die Anforderungen einer flexiblen Systemlandschaft im Unternehmen.

Die eigentliche Herausforderung bei der Einführung eines SAP HANA SQL DWH liegt darin, den passenden Aufbau und die passenden Methoden für das eigene Unternehmen zu finden. Viele Faktoren, wie z. B. eine komplexe bzw. heterogene Quellsystemlandschaft, Fertigkeiten der Mitarbeiter oder die Strategie bzgl. On-Premise vs. Cloud, beeinflussen die Gestaltung des SAP HANA SQL DWH.

Wir glauben, dass es SAP uns Kunden einfacher machen würde, wenn ein Best-Practice-Guide für die Implementierung eines SAP HANA SQL DWH angeboten würde. Vielleicht würde die Bereitstellung eines Business-Contents (analog zum BW) die Anwender befähigen, die durch die gewonnene Freiheit auftretende Komplexität etwas zu vermindern und damit die Akzeptanz zu steigern.

Unsere Themengruppe wird sich weiter mit dem SAP HANA SQL DWH beschäftigen. Dazu möchten wir gern tiefere Einblicke in die Technologien und Methoden geben, die SAP in diesem Kontext anbietet. Auch wollen wir mit einem folgenden Leitfaden einmal exemplarisch beschreiben, wie man bei der Implementierung vorgehen kann. Sie dürfen also gespannt sein und bei Interesse sind Sie eingeladen mitzuwirken.

6. Anhang

6.1 Glossar

Begriff	Beschreibung
ABAP	Advanced Business Application Programming
ADSO	Advanced DataStore Object
BI	Business Intelligence
BW	Business Warehouse
CRM	Customer-Relationship-Management
DB	Database
DLM	Data Lifecycle Manager
DWC	SAP Data Warehouse Cloud
DWF	SAP HANA Data Warehousing Foundation
DWH	Data Warehouse
EAD	SAP Enterprise Architecture Designer
ERP	Enterprise Resource Planning
ETL	Extract, Load, Transform
HANA DB	SAP HANA Datenbank
IT	Informationstechnik
OLAP	Online Analytic Processing
OLTP	Online Transactional Processing
SDI	SAP HANA smart data integration
SLT	SAP Landscape Transformation
SQL	Structured Query Language
UI	User Interface

Impressum

Wir weisen ausdrücklich darauf hin, dass das vorliegende Dokument nicht jeglichen Regelungsbedarf sämtlicher DSAG-Mitglieder in allen Geschäftsszenarien antizipieren und abdecken kann. Insofern müssen die angesprochenen Themen und Anregungen naturgemäß unvollständig bleiben. Die DSAG und die beteiligten Autoren können bezüglich der Vollständigkeit und Erfolgsgeeignetheit der Anregungen keine Verantwortung übernehmen.

Die vorliegende Publikation ist urheberrechtlich geschützt (Copyright).

Alle Rechte liegen, soweit nicht ausdrücklich anders gekennzeichnet, bei:

Deutschsprachige SAP® Anwendergruppe e.V.
Altrottstraße 34 a
69190 Walldorf | Deutschland
Telefon +49 6227 35809-58
Telefax +49 6227 35809-59
E-Mail info@dsag.de

dsag.de

Jedwede unerlaubte Verwendung ist nicht gestattet. Dies gilt insbesondere für die Vervielfältigung, Bearbeitung, Verbreitung, Übersetzung oder die Verwendung in elektronischen Systemen/digitalen Medien.

© Copyright 2020 DSAG e.V.