Kapitel 7
Operative Analytik

Eine Trennung transaktionaler und analytischer Anwendungsfälle ist im SAP-Umfeld seit jeher üblich. Mit SAP HANA haben Sie nun die Möglichkeit, verschiedene virtuelle Datenmodelle unter Verwendung desselben physikalischen Modells zu definieren.

Im vorliegenden Buch haben Sie sich bisher in erster Linie mit der Bearbeitung transaktionaler Prozesse beschäftigt. Im Backend sind hierbei *Core Data Services* (CDS, siehe Abschnitt 2.3) *Views* und das *Business Object Processing Framework* (BOPF, siehe Abschnitt 4.4) von zentraler Bedeutung, die Visualisierung im Frontend erfolgt über SAP Fiori bzw. SAPUI5. Die entwickelten Applikationen haben in der Regel das Ziel, das Anlegen, Ändern oder Löschen transaktionaler Daten zu ermöglichen.

In diesem Kapitel betrachten Sie einen anderen Aspekt innerhalb Ihres SAP-Systems: die Analyse von Daten und die Ermittlung von Leistungszahlen (besser bekannt unter dem engl. Begriff *Key Performance Indicator*, KPI).

Transaktionaler vs. analytischer Kontext

Entsprechend dem *Principle of One* (siehe Abschnitt 1.1.1, »Entwicklungsprinzipien von SAP S/4HANA«) stützen sich die folgenden Beispiele analytischer Anforderungen auf das gleiche Datenmodell und die gleichen CDS Basic Views, die Ihnen bereits aus den transaktionalen Anwendungsfällen geläufig sind. Eine redundante Datenhaltung oder separate Modellierung der CDS Basic Views für den analytischen Kontext ist nicht notwendig. Basierend auf dem wiederverwendbaren Datenmodell kann es allerdings erforderlich sein, separate Entwicklungsartefakte für analytische Anwendungsfälle zu implementieren, um deren Anforderungen zu bedienen.

In Abschnitt 7.1, »Grundlagen«, gehen wir zunächst kurz auf besondere Merkmale analytischer Szenarien ein und grenzen diese zu transaktionalen ab. In Abschnitt 7.2, »Analytische Core Data Services«, zeigen wir Ihnen konkret den Entwurf eines analytischen Modells. Dieses Modell implementieren Sie im Backend mittels analytischer ABAP CDS Views für unser Beispiel des EPM-Modells (Enterprise Procurement Models). Hierbei erfahren Sie zudem, wie Sie das Datenmodell parametrisieren und durch den Query-Monitor direkt testen. Abschnitt 7.3, »Visualisierung mit der Analytical List

Page«, und Abschnitt 7.4, »Weitere Möglichkeiten der Visualisierung«, geben Ihnen einen Überblick über SAP-Technologien, die Ihnen zur Verfügung stehen, um das von Ihnen entwickelte Modell für Benutzer verwendbar zu machen. Wir legen hierbei mit Abschnitt 7.3 den Schwerpunkt auf die *Analytical List Page* (ALP) von SAP Fiori Elements. Basierend auf dem in Abschnitt 7.2 erstellten Datenmodell implementieren Sie einen vereinfachten Anwendungsfall, um die Besonderheiten des Floorplans vorzustellen. Zusätzlich enthält Abschnitt 7.4 einen groben Überblick über weitere verbreitete Visualisierungsmöglichkeiten und die Integration mit der SAP Analytics Cloud.

7.1 Grundlagen

In Abschnitt 3.1, »Simplifizierung und das Principle of One von SAP«, wurde der Begriff *operative Analytik* definiert und darauf hingewiesen, dass Sie operative Szenarien unter SAP HANA ohne ein Enterprise Data Warehouse in Echtzeit durchführen können. Bevor Sie die praktischen Werkzeuge zur Implementierung erlernen, geben wir Ihnen zunächst ein paar Hintergrundinformationen zur operativen Analytik und der Evolution analytischer Systeme. Seit den 1990er Jahren gibt es grundsätzlich eine klare Abgrenzung zwischen den transaktionalen und den analytischen Anwendungsgebieten im SAP-Bereich.

OLTP und OLAP

Der Unterschied der beiden Szenarien liegt unter anderem im Datenmodell und lässt sich als *Online Transaction Processing* (OLTP) gegenüber *Online Analytical Processing* (OLAP) fassen. OLTP-Szenarien basieren in der Regel auf einem Modell von Entitäten, definiert in der dritten Normalform. Im Gegensatz dazu nutzen OLAP-Szenarien ein Datenmodell, das klassischerweise auf einem Sternschema mit zentralen Fakten und assoziierten Dimensionen basiert.

Operative Analytik in SAP HANA

Operative Analytik verbindet OLTP- und OLAP-Szenarien miteinander. Mit Einführung von SAP HANA ist es möglich, beide Szenarien mit einer einzelnen, zentralen Datenbank abzubilden. Dabei werden über ABAP CDS Views für transaktionale und analytische Anwendungsfälle unterschiedliche *virtuelle Datenmodelle* aufgebaut, die physikalisch auf denselben Datenbanktabellen operieren. Somit ist implizit sichergestellt, dass beide Kontexte auf denselben Datenbestand zugreifen – aufwendige Synchronisierungen aufgrund redundanter Datenhaltung gehören der Vergangenheit an. Wichtig ist dabei jedoch, dass derselbe Datenbestand nicht auch denselben Datenzugriff bedeutet.

Wiederverwendung von CDS Views

Die Erfahrung hat gezeigt, dass eine Wiederverwendung von CDS Views grundsätzlich ein erstrebenswertes Ziel ist, das jedoch nicht um jeden Preis erzwungen werden muss. Unterschiedliche Szenarien haben unterschiedliche Anforderungen an die Selektion der Daten, vor allem wenn es sich sowohl um OLTP- als auch OLAP-Szenarien handelt. Um eine hohe Qualität bezüglich der Performance zu erzielen, kann es notwendig sein, ähnliche Anfragen unterschiedlicher Anwendungsszenarien (z. B. bei unterschiedlichen Granularitätsgraden der Aggregation) separat voneinander zu entwickeln. Lediglich auf Ebene der CDS Interface und Basic Views sollte eine Wiederverwendbarkeit sichergestellt werden, da diese Views unabhängig von der konkreten Applikation entwickelt werden.

Rein technisch handelt es sich bei den Artefakten für den Datenzugriff in OLAP-Szenarien um dieselben wie bei OLTP-Szenarien: ABAP CDS Views. Abschnitt 7.2, »Analytische Core Data Services«, erläutert die Besonderheiten und die entsprechenden *Annotationen* in analytischen Szenarios.

7.2 Analytische Core Data Services

In Abschnitt 2.3, »Core Data Services«, konnten Sie sich bereits einen allgemeinen Überblick über den Datenzugriff mithilfe von Core Data Services (CDS) und deren verschiedener View-Typen verschaffen. Bisher haben Sie allerdings lediglich die transaktionale Sicht und die damit verbundenen Annotationen kennengelernt. Dieser Abschnitt gibt Ihnen einen Überblick darüber, welche CDS-Konzepte sowohl für OLTP- als auch für OLAP-Szenarios gelten und welche Eigenheiten und Annotationen speziell für OLAP relevant sind.

Principle of One beim Datenmodell

SAP empfiehlt, dass Sie sich bei der Erstellung des virtuellen Datenmodells (VDM) am Principle of One orientieren. Mit anderen Worten sollte das Datenmodell möglichst einmal zentral definiert und von allen Verwendern einheitlich konsumiert werden. Dies gilt allerdings nur bis zu einem bestimmten Grad. Liegen CDS Views nah an der Persistenz, ist die Wahrscheinlichkeit hoch, dass Sie sie wiederverwenden können. Liegen Sie dagegen an der konsumierenden Schicht, sind sie in der Regel applikationsspezifischer und demnach schlechter geeignet für die applikationsübergreifende Verwendung.

CDS Views für OLAP-Szenarien

Eine klare Abgrenzung von CDS Views für OLAP-Szenarien ist spätestens auf oberster Ebene essenziell. Zum Konsumieren der Daten sind spezielle OLAP-spezifische Annotationen notwendig, die das sogenannte analytische Modell definieren. Abschnitt 7.2.1 gibt Ihnen einen Überblick, wie Sie

dieses Modell definieren. Sie vertiefen daraufhin das theoretische Wissen anhand eines praktischen Beispiels in Abschnitt 7.2.2.

7.2.1 Das analytische Datenmodell

Struktur des analytischen Modells

Um die bereits erwähnten OLAP-Szenarien abdecken zu können, müssen Sie in Ihrem Backend-System das *analytische Datenmodell* basierend auf Ihrer Persistenzschicht modellieren. Die Implementierung basiert grundsätzlich ausschließlich auf CDS Views. Dabei sollten Sie bei der Modellierung auf eine klare Struktur achten – jeder CDS View hat eine definierte Aufgabe. Es ist also sicherlich kein Best Practice, das analytische Modell in möglichst wenigen CDS Views abzubilden. Dennoch sollten Sie unnötige Abstraktionsschichten vermeiden, um die Komplexität gering zu halten.

CDS Views des analytischen Modells

In diesem Abschnitt erlangen Sie einen Überblick über die verschiedenen Typen von CDS Views, die im analytischen Modell zum Einsatz kommen sollten. Hierbei werden die Typen *bottom-up* gemäß ihrer Aufrufhierarchie erläutert. Das heißt, wir starten nah an der Persistenzschicht und enden mit dem View, der über OData schließlich exponiert wird. Die allgemeine Klassifizierung von CDS Views über Basic, Composite, Interface und Consumption Views bzw. das Konzept von privaten Views kennen Sie bereits aus Abschnitt 2.3, »Core Data Services«. Hierauf aufbauend werden die vorgestellten CDS Views nun innerhalb des analytischen Kontextes eingeordnet.

Typen von CDS Views

Grundsätzlich gilt, dass jeder Typ von CDS View einen anderen Zweck erfüllt. Dies bedeutet allerdings nicht, dass ein CDS View nicht mehr als einen Zweck erfüllen kann. Ein Interface View kann beispielsweise auch als Dimension im analytischen Modell definiert werden. Ein Composite View ist automatisch ein Interface View und kann z. B. ein Cube im Sinne des analytischen Modells sein (Näheres hierzu finden Sie im weiteren Verlauf dieses Abschnitts). Kurz gesagt kann ein CDS View mehrere Aufgaben übernehmen und entsprechend mehreren Typen zugeordnet werden.

Abbildung 7.1 gibt Ihnen einen Überblick darüber, wie die Aufrufhierarchie in Ihrem analytischen Modell aussehen kann. Die dunkelgrau visualisierten CDS Views bilden die Typen des analytischen Modells.

An dieser Stelle sei bemerkt, dass diese zur Ausführungszeit im Gegensatz zu transaktionalen Szenarien unterschiedlich prozessiert werden. Mithilfe der *Analytical Engine*, auch bekannt als analytischer Prozessor, als zentraler Komponente der analytischen Infrastruktur wird eine auf OLAP-Szenarien optimierte interne Verarbeitung sichergestellt.

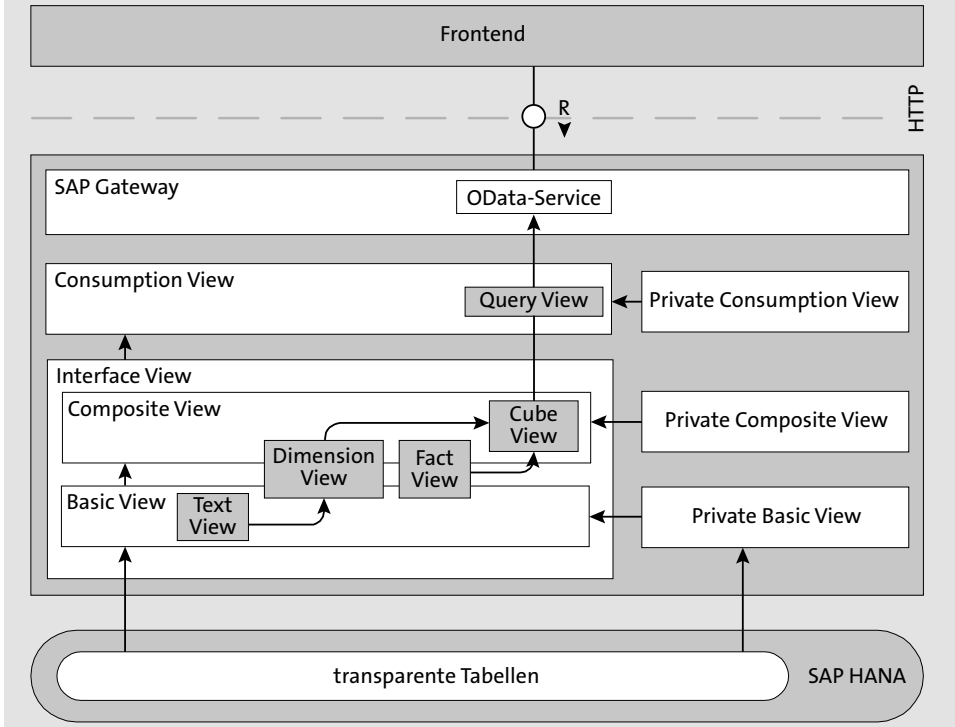


Abbildung 7.1 Überblick über Typen von CDS Views des analytischen Datenmodells

Im weiteren Verlauf dieses Abschnitts stellen wir Ihnen die in Abbildung 7.1 abgebildeten Typen von CDS Views vor und ordnen sie im analytischen Modell ein. In Abschnitt 7.2.2 erstellen Sie dann am Beispiel des Enterprise Procurement Models selbst ein analytisches Modell.

CDS Fact View

Zentrum des Sternschemas

Das analytische Modell ist klassischerweise als Sternschema aufgebaut. Im Zentrum befindet sich der *Fact View*, dessen Aufgabe es ist, die Kennzahlen des Modells in normalisierter Form bereitzustellen. Er selbst enthält keine Verbünde zu Stammdaten und wird durch die Annotation `@Analytics.dataCategory: #FACT` gekennzeichnet. Die bereitgestellten Kennzahlen werden dann im *Cube View* mit deren Dimensionen verbunden. Die Dimensionen bilden die Strahlen des Sterns, der Cube View ist der Stern selbst.

Abbildung 7.2 zeigt das Verhältnis der CDS-View-Typen schematisch. PK bezeichnet hierbei den Primärschlüssel (Primary Key), FK einen Fremdschlüs-

sel (Foreign Key). Der CDS Fact View hat pro Fremdschlüsselfeld eine 1:1-Beziehung zum Primärschlüssel eines CDS Dimension Views.

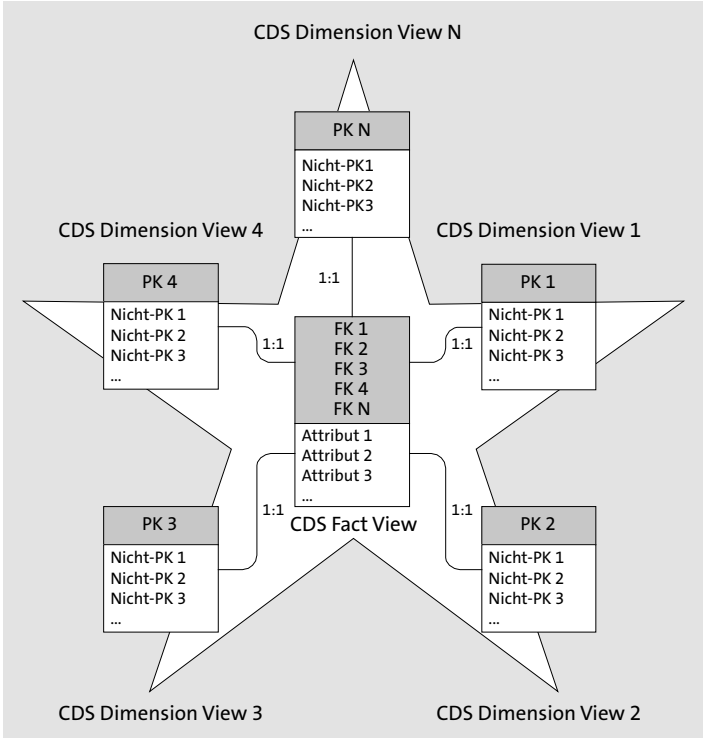


Abbildung 7.2 Aufbau eines CDS Cube Views

Zweck des Fact Views

Der Fact View dient als primäre Datenquelle Ihres analytischen Modells. Abhängig von der Anzahl und Zusammensetzung der konsumierten Datenquellen kann er den Typen Basic View oder Composite View zugeordnet werden. Er wird häufig im Kontext eines Business Warehouses (BW) zur Datenextraktion genutzt. Weiterführende Informationen erhalten Sie z. B. im Buch »SAP BW/4HANA. Konzepte, Prozesse, Funktionen« von T. Lüdtke (SAP PRESS 2017) oder »SAP BW auf SAP HANA. Implementierung und Migration« von M. Merz, T. Hügens, S. Blum (SAP PRESS 2014).

Empfehlung

Rein theoretisch können Sie bei der Erstellung des analytischen Modells die Kennzahlen auch direkt im Cube View selektieren und mit den Dimensionen verbinden. In diesem Fall würden Sie auf die explizite Definition der Kennzahlen innerhalb eines Fact Views verzichten und könnten sich somit Entwicklungszeit und eine zusätzliche Abstraktionsebene sparen. Unserer Ansicht nach sollte der Fact View nichtsdestotrotz grundsätzlich Bestandteil des analytischen Modells sein und den Kern des Sternschemas definieren. Neben der potenziellen Wiederverwendbarkeit des Fact Views ist der Hauptgrund hierfür eine saubere, strukturierte Trennung der Zuständigkeiten. Dieses Prinzip erhöht nicht nur die Übersichtlichkeit und spätere Wartbarkeit, sondern vereinfacht auch den Austausch von Komponenten innerhalb des Modells.

CDS Dimension View

Strahlen des Sternschemas

Wie in Abbildung 7.2 gezeigt, bilden die *Dimension Views* die Strahlen des Sternschemas. Ähnlich wie der Fact View kann der Dimension View je nach Komplexität entweder als Basic View oder als Composite View definiert werden. Er wird mittels der Annotation `@Analytics.dataCategory: #DIMENSION` gekennzeichnet. Der Primärschlüssel des Dimension Views hat hierbei immer eine Fremdschlüsselbeziehung zu einem Attribut des Fact Views, um die Dimension mit den Kennzahlen in Kontext setzen zu können. Die Annotation `@ObjectModel.representativeKey` wird innerhalb des Dimension Views dazu genutzt, das markanteste Merkmal des Primärschlüssels explizit zu definieren. Sie fungiert sozusagen als Anker beim Konsumieren der Dimension von außen. Die Annotation muss auch definiert werden, wenn der Primärschlüssel aus lediglich einem Feld besteht.

Zweck des Dimension Views

Dimensionen sollten zur Assoziation sprachabhängiger Textfelder (Annotation `@ObjectModel.text.element`) und für Hierarchien genutzt werden. Grundsätzlich kann eine Dimension im analytischen Kontext auch als eigenständige Datenquelle genutzt werden. Dimension Views sind prinzipiell mit den Merkmalen (Infoobjekten) des klassischen SAP BW vergleichbar.

CDS Cube View

Einordnung im analytischen Modell

Der Cube View fungiert als zentraler View und Datenquelle für mehrdimensionalen Zugriff auf ein Sternschema – man könnte sagen, der Cube View ist der Stern selbst. Er verbindet den Fact View mit den Dimension Views per Assoziation und enthält demnach denormalisierte Daten, angereichert um zusätzliche Informationen. Innerhalb einer Assoziation müssen alle Schlüsselfelder der Dimension versorgt werden.

Zweck des Cube Views

Zudem müssen die Attribute einer Assoziation über `on` mit einem Ist-gleich-Operator verknüpft werden. Die Projektionsliste enthält alle Dimensionsattribute und Kennzahlen, die für das analytische Modell relevant sind. Die Dimensionsattribute müssen über die Annotation `@ObjectModel.foreignKey.association` mit der entsprechenden Assoziation verknüpft werden. Hierbei müssen Sie das Feld referenzieren, das Sie im Dimension View als repräsentatives Schlüsselfeld annotiert haben. Die Kennzahlen werden in-

nerhalb des Views über die obligatorische Annotation `@DefaultAggregation` identifiziert und an eine Aggregationsfunktion geknüpft. Der Cube View kann auch Formeln zur Berechnung weiterer Kennzahlen umsetzen, die auf den Feldern des Fact Views basieren – oder, wie in unserem Beispiel, eine Währungsumrechnung basierend auf bereits ermittelten Beträgen durchführen. Diese Umrechnung könnte grundsätzlich auch erst im *Query View* implementiert werden. Unserer Ansicht nach sollte Sie aber im Cube View durchgeführt werden, da in der Regel lediglich die umgerechneten Werte für die Konsumenten von Interesse sind. Auf diese Weise können Sie die einmal implementierte Währungsumrechnung direkt in mehreren Query Views verwenden.

Der Cube View wird durch die Annotation `@Analytics.dataCategory: #CUBE` gekennzeichnet. Im klassischen Fall wird er von einem oder mehreren Query Views konsumiert, die dann wiederum vom Frontend, beispielsweise einer Analytical List Page, verwendet werden können. Nach Auflistung der Attribute der Projektionsliste eines Cube Views werden alle verwendeten Assoziationen noch mal aufgeführt, um dem Konsumenten Zugriff auf deren Attribute und Texte zu ermöglichen.

CDS Query View

Einordnung im analytischen Modell

Auf dem Cube View wird abschließend der Query View definiert. Da dieser bereits alle Kennzahlen und Dimensionen publiziert, benötigt der Query View selbst keine Assoziationen. SAP weicht hier von der Klassifizierung eines Views per `@Analytics.dataCategory` ab – der Query View wird per `@Analytics.query: true` identifiziert.

Zweck des Query Views

Query Views ermöglichen ein Konsumieren der Daten aus dem Backend durch das Frontend – sei es direkt über den Query-Namen oder indirekt über einen OData-Service, der durch einen Query View erstellt wurde. Sie können über die Transaktion RSRT getestet werden – hierauf gehen wir in Abschnitt 7.2.2, »Definition des analytischen Datenmodells«, näher ein. In Query Views wird definiert, wie die analytischen Daten zur Verfügung stehen, z. B. über eine Definition der Attributsequenz. Es können Daten in Spalten oder Zeilen dargestellt und gefiltert werden, auch Zwischensummen sind möglich. Sie sind mit transaktionalen Consumption Views und deren UI-Annotationen (siehe Abschnitt 4.6, »Entwicklung einer Benutzeroberfläche mit SAP Fiori Elements«) vergleichbar.

Analytical Engine

Der Query View wird von der auf analytische Anwendungszwecke spezialisierten Analytical Engine prozessiert. Sie kann mit speziellen analytischen Anforderungen wie Formelberechnung, Einschränkung von Kennzahlen oder Fehlerbehandlung während der Aggregation umgehen.

Im folgenden Abschnitt 7.2.2 geben wir Ihnen einen Überblick über die Definition eines analytischen Datenmodells, das von einer Analytical List Page (SAP Fiori Elements) konsumiert werden kann.

7.2.2 Definition des analytischen Datenmodells

Wie bereits in den vorangegangenen Abschnitten erläutert, ist ein analytisches Datenmodell ab einem bestimmten Punkt in der CDS-Aufrufkette sehr anwendungsspezifisch. Dementsprechend sollte vor dem Aufbau des Modells der Anwendungsfall klar definiert werden.

Fachlich unvollständig definierte Modelle

Da dem Fachbereich zu Projektstart oft entweder die Zeit und/oder das nötige Wissen fehlt, das große Ganze sowie Abhängigkeiten zu definieren, ist es in der Praxis oft unausweichlich, mit fachlich nur mittelmäßig durchdachten Modellen zu starten. Aus Entwicklungssicht können wir Ihnen dennoch nur empfehlen, ein fachlich möglichst vollständig durchdachtes Konzept einzufordern, bevor mit der Implementierung eines analytischen Modells begonnen wird. Hierbei sind Mock-ups und ein enger Austausch zwischen IT-Architekt und Fachbereich hilfreich. Zumindest die Basis des Modells, die Entitäten und deren Assoziationen inklusive Kardinalitäten untereinander sollten feststehen. Eine nachträgliche Änderung des Modells während der Entwicklung kann zu massiven Refaktorierungskosten führen. Es empfiehlt sich, bei unvollständigem Fachkonzept darauf hinzuweisen, dass das Modell lediglich unter speziellen, abgestimmten Annahmen implementiert wird.

Anforderungen an das analytische Modell

Für unser Szenario ist die fachliche Anforderung glücklicherweise voll spezifiziert. Unser analytisches Modell soll folgende fachliche Anforderungen abbilden:

1. **Kundenaufträge**:
   Das Datenmodell soll die Analyse von Kundenaufträgen und deren Positionen erlauben. Besonderes Augenmerk soll dabei auf dem bezahlten Bruttobetrag für die Produkte einzelner Positionen in aggregierter Form liegen. Zusätzlich soll es möglich sein, Informationen über den Netto- bzw. Steuerbetrag abzugreifen. Hierbei sollen die Beträge der bezahlten Produkte in einem ersten Schritt immer in Euro berücksichtigt werden – als optionale Anforderung soll die Währung vom Benutzer konfigurierbar sein.

2. **Kundenauftragspositionen**:
   Die Beträge der Kundenauftragspositionen sollen im Kontext des gesamten Kundenauftrags evaluiert werden. Hierzu soll nach den einzelnen Statusfeldern eines Kundenauftrags aggregiert werden können. Relevant sind Lebenszyklus-, Bestätigungs-, Bestell- und Gesamtstatus.
3. **Produkte**:
   Jedes Produkt einer Kundenauftragsposition fällt unter eine bestimmte Produktkategorie. Nach dieser Kategorie soll im analytischen Modell aggregiert werden können.
4. **Lieferanten**:
   Jedes Produkt einer Kundenauftragsposition hat einen Lieferanten, der in einem konkreten Land beheimatet ist. Dieses Land stellt ebenfalls eine wichtige Dimension dar, um Rückschlüsse auf möglicherweise große Entfernungen von Lieferanten ziehen zu können.
5. **Kunden**:
   Dasselbe gilt für die Kunden, die ein Produkt über die Kundenauftragsposition in Auftrag gegeben haben. Auch deren Land ist für die Auswertung von Interesse.

Der Query View soll später zudem von einer Analytical List Page (SAP Fiori Elements) per OData konsumiert werden können. Dies setzen Sie in Abschnitt 7.3 um.

Wiederverwendbare CDS Basic Views

Beginnen Sie zur Umsetzung der Anforderung mit der Erstellung des analytischen Modells. Sinngemäß wird das Modell von unten nach oben aufgebaut, das bedeutet, Sie beginnen mit der Definition der CDS Basic Views, die den Direktzugriff auf die Datenbanktabellen abbilden.

Anstelle oft kryptischer Datenbankfeldnamen verwenden Sie für unser Szenario bereits erstellte CDS Basic Views, die für das analytische Modell wiederverwendet werden können. Auf Ebene dieser CDS Views nahe an der Persistenz ist deren Wiederverwendung problemlos möglich, selbst wenn diese ursprünglich für einen transaktionalen Kontext implementiert wurden. Folgende CDS Basic Views sind für das Szenario relevant:

- `ZI_SalesOrderHeader`
- `ZI_SalesOrderItem`
- `ZI_Product`
- `ZI_Text`
- `ZI_BusinessPartner`
- `ZI_Address`

In Anhang B, »Das Enterprise Procurement Model«, finden Sie eine Auflistung der einzelnen Felder und Zusammenhänge der Views, die Ihnen einen detaillierten Überblick über das Datenmodell gibt. Die Implementierung der Views finden Sie in den Downloadmaterialien zum Buch.

Legen Sie in einem ersten Schritt (falls noch nicht aufgrund vergangener Übungen bereits geschehen) die aufgelisteten CDS Basic Views an. Basierend auf diesen Views implementieren Sie im nächsten Schritt alle weiteren View-Typen, die Sie für die Definition eines analytischen Modells benötigen.

Namen analytischer CDS Views

Nach der Klassifizierung des CDS Views (Consumption oder Interface Views) kennzeichnen wir analytische Views noch einmal explizit mit `A`, gefolgt vom ersten Buchstaben des View-Typs, beispielsweise also `ZI_AF_*` für Fact Views oder `ZC_AQ_*` für Query Views. Diese Konventionen wurden von uns frei gewählt und basieren auf keinerlei Vorgaben von SAP.

Fact View implementieren

Beginnen Sie mit der Implementierung des Fact Views, der der Definition der Kennzahlen dient (für die Umsetzung von Anforderung 1 relevant). Hierzu implementieren Sie den View `ZI_AF_SalesOrderItem` und selektieren die Kennzahlen aus dem wiederverwendbaren Basic View `ZI_SalesOrderItem`.

Kennzahlen definieren

Listing 7.1 zeigt die Implementierung des Fact Views.

```
@AbapCatalog.sqlViewName: 'ZI_AF_SOI'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'EPM: Fact Sales Order Item'
@Analytics.dataCategory: #FACT
@VDM.viewType: #BASIC
define view ZI_AF_SalesOrderItem
  as select from ZI_SalesOrderItem {
key SalesOrderItemKey,
    SalesOrderKey,
    SalesOrderItemPosition,
    @Semantics.currencyCode: true
    Currency,
    @Semantics.amount.currencyCode: 'Currency'
    GrossAmount,
```

```
    @Semantics.amount.currencyCode: 'Currency'
    NetAmount,
    @Semantics.amount.currencyCode: 'Currency'
    TaxAmount
}
```

Listing 7.1 Fact View »ZI_AF_SalesOrderItem«

Beachten Sie, dass die umgerechneten Felder per `@Semantics`-Annotation wieder mit dem entsprechenden Zielwährungsfeld verknüpft werden.

Dimension View implementieren

Um die Kennzahlen abhängig von anderen Attributen aggregieren und visualisieren zu können, werden CDS Dimension Views implementiert. Hierbei gehen Sie die einzelnen Punkte der Anforderungen durch und legen entsprechende Views an.

Statusfelder der Kundenaufträge hinzufügen

Laut Anforderung 2 sollen die Beträge der Kundenauftragspositionen im Kontext der eigentlichen Kundenaufträge nach deren Status evaluiert werden können. Sie modellieren zu diesem Zweck einen CDS Dimension View `ZI_AD_SalesOrderHeader` und lesen die Statusfelder aus dem Kundenauftrag. Listing 7.2 zeigt, wie die Implementierung dazu aussieht. Hierbei ist zu bemerken, dass Sie grundsätzlich auch den Basic View `ZI_SalesOrderHeader` direkt verwenden könnten. Sie wollen allerdings das Prinzip der Trennung von Zuständigkeiten einhalten und legen deshalb einen eigenen View für den analytischen Zweck einer Dimension an. So ist sichergestellt, dass Sie diesen ohne Bedenken später erweitern können.

```
@AbapCatalog.sqlViewName: 'ZI_AD_SOH'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'EPM: Dimension Sales Order Header'
@Analytics.dataCategory: #DIMENSION
@VDM.viewType: #COMPOSITE
@ObjectModel.representativeKey: 'SalesOrderHeaderKey'
define view ZI_AD_SalesOrderHeader
  as select from ZI_SalesOrderHeader {
    key SalesOrderHeaderKey,
        SalesOrderID,
        LifecycleStatus,
        BillingStatus,
        DeliveryStatus,
```



```
        OverallStatus
}
```

Listing 7.2 Dimension View »ZI_AD_SalesOrderHeader«

Neben den technisch notwendigen Schlüsselfeldern und den Dimensionsattributen ist es sinnvoll, auch die semantischen Schlüsselfelder (hier `SalesOrderID`) zur fachlichen Identifikation der Datensätze mit zu selektieren.

Produktkategorie definieren

Anforderung 3 gibt vor, dass eine Kundenauftragsposition über die Kategorie ihres Produkts im analytischen Modell aggregiert werden können soll. Listing 7.3 enthält die Implementierung der zu diesem Zweck notwendigen Dimension `ZI_AD_Product`.

```
@AbapCatalog.sqlViewName: 'ZI_AD_PRD'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'EPM: Dimension Product'
@Analytics.dataCategory: #DIMENSION
@VDM.viewType: #BASIC
@ObjectModel.representativeKey: 'ProductKey'
define view ZI_AD_Product
  as select from ZI_Product {
    key ProductKey,
        ProductID,
        Category
}
```

Listing 7.3 Dimension View »ZI_AD_Product«

Zusätzlich wird die Projektionsliste unseres Fact Views `ZI_AF_SalesOrderItem` um den `ProductKey` erweitert, der zum späteren Verbinden der Daten mit der neuen Produktdimension erforderlich ist.

Land des Lieferanten implementieren

Anforderung 4 ist nun etwas anspruchsvoller: Das Land des Lieferanten eines Produkts soll als Dimension bereitgestellt werden. Um den Lieferanten eines Produkts später mit der Kundenauftragsposition verbinden zu können, müssen Sie in einem ersten Schritt das Feld `SupplierKey` in `ZI_AD_Product` in die Projektionsliste aufnehmen. Lieferanten sind im EMP grundsätzlich immer Geschäftspartner, das bedeutet, unsere Kerndaten können per Basic View `ZI_BusinessPartner` konsumiert werden. Aus diesem Grund fügen Sie in `ZI_AD_Product` eine Assoziation zu diesem CDS View ein. Da der assoziierte CDS View allerdings das Land des Lieferanten noch nicht enthält, benötigen Sie eine weitere Assoziation zur Adresse des Lieferanten (`ZI_Address`), über dessen Feld `AddressKey`. Die Adressdaten werden innerhalb

des Views vollständig nach außen propagiert, sodass der Konsument auf dessen Attribute zugreifen kann.

Composite View verwenden

Da die Produktdimension nun auch Aspekte des Geschäftspartners beinhaltet, ändern Sie den View-Typ von Basic auf Composite. Die angepasste Dimension finden Sie in Listing 7.4.

```
@AbapCatalog.sqlViewName: 'ZI_AD_PRD'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'EPM: Dimension Product'
@Analytics.dataCategory: #DIMENSION
@VDM.viewType: #COMPOSITE
define view ZI_AD_Product
  as select from ZI_Product as Product
    association[1..1] to ZI_BusinessPartner as _Supplier
      on Product.SupplierKey = _Supplier.BusinessPartnerKey
    association[1..1] to ZI_Address as _SupplierAddress
      on SupplierAddressKey = _SupplierAddress.AddressKey {
    key Product.ProductKey,
        Product.ProductID,
        Product.Category,
        Product.SupplierKey,
        _Supplier.BusinessPartnerID,
        _Supplier.AddressKey as SupplierAddressKey,
        _SupplierAddress
}
```

Listing 7.4 Angepasster Dimension View »ZI_AD_Product«

Land des Kunden implementieren

Anforderung 5 besagt, dass auch das Land des Kunden für ihr analytisches Modell von Interesse sein soll. Der Kunde ist Attribut des Kundenauftrags und über die Dimension `ZI_AD_SalesOrderHeader` im Zugriff. Sie verwenden demnach diesen CDS View zur Erfüllung von Anforderung 5. Zunächst fügen Sie das Attribut `BuyerKey` hinzu. Dieses wird nun analog zu Anforderung 4 verwendet, um Assoziationen zu `ZI_BusinessPartner` und `ZI_Address` zu implementieren. Listing 7.5 zeigt die angepasste Implementierung von `ZI_AD_SalesOrderHeader`.

```
@AbapCatalog.sqlViewName: 'ZI_AD_SOH'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'EPM: Dimension Sales Order Header'
```



```
@Analytics.dataCategory: #DIMENSION
@VDM.viewType: #COMPOSITE
define view ZI_AD_SalesOrderHeader
  as select from ZI_SalesOrderHeader
    association[1..1] to ZI_BusinessPartner as _Buyer
      on ZI_SalesOrderHeader.BuyerKey =
                  _Buyer.BusinessPartnerKey
    association[1..1] to ZI_Address as _BuyerAddress
      on BuyerAddressKey = _BuyerAddress.AddressKey {
    key SalesOrderHeaderKey,
        SalesOrderID,
        LifecycleStatus,
        BillingStatus,
        DeliveryStatus,
        OverallStatus,
        BuyerKey,
        _Buyer.BusinessPartnerID,
        _Buyer.AddressKey as BuyerAddressKey,
        _BuyerAddress
}
```

Listing 7.5 Angepasster Dimension View »ZI_AD_SalesOrderHeader«

Die Attribute der Assoziation `_Buyer` sind auf diese Weise für den Konsumenten (also unseren später implementierten Cube View) bei Bedarf selektierbar.

Text CDS View referenzieren

Bevor wir auf den Cube View eingehen, wollen wir an dieser Stelle noch einen weiteren CDS-View-Typ ansprechen: den *Text View* (`@ObjectModel.dataCategory: #TEXT`). Sie haben im Dimension View die Möglichkeit, ähnlich wie in einem transaktionalen Szenario einen Text View über eine Textassoziation zu referenzieren. Näheres zu der konkreten Implementierung finden Sie in Abschnitt 6.3, »Annotationen«.

Cube View implementieren

Nachdem Sie für die einzelnen Anforderungen Dimension bzw. Fact Views implementiert haben, bringen Sie diese innerhalb eines Cube Views `ZI_AC_SalesOrderItem` in Abhängigkeit zueinander. Der Cube View selektiert die Daten unseres Fact Views, angereichert durch Assoziationen auf die Dimension Views. Die Projektionsliste enthält die Attribute zur Identifikation einer Kundenauftragsposition, alle Dimensionsattribute sowie die Kennzahlen aus dem Fact View.

Währungsumrechnung einführen

Um sicherzustellen, dass die Kennzahlen vom Benutzer auch miteinander verglichen werden können, führen Sie an dieser Stelle eine Währungsumrechnung durch. Auf diese Weise vereinheitlichen Sie das Format der Kennzahlen, sodass SAP HANA die entsprechenden Werte aggregieren kann. Sie nutzen hierzu die durch SAP bereitgestellte SAP-HANA-Funktion `CURRENCY_CONVERSION`. Auf diese Weise wird die Umrechnung direkt von der SAP-HANA-Datenbank im Sinne des *Code Pushdowns* (siehe Abschnitt 2.5, »Code Pushdown auf die SAP-HANA-Datenbank«) durchgeführt. Der Einfachheit halber entscheiden wir uns an dieser Stelle für eine Visualisierung in »Euro« und definieren dies hart codiert im View. Für den produktiven Einsatz sollte dies selbstverständlich vom Benutzer definiert werden können – Abschnitt 7.2.3, »Parametrisierung des analytischen Modells und Filterung«, beschreibt die notwendigen Anpassungen am Modell. Für die Währungsumrechnung müssen Sie der Funktion ein Datum übergeben. Dieses Datum wird zur internen Ermittlung des zu verwendenden Umrechnungskurses benötigt. Sollten Sie auf einem SAP-NetWeaver-Release ≥ 7.51 arbeiten, können Sie `$session.system_date` (nach einem Cast auf den Typ `dats`) übergeben, die Variable hält das aktuelle Datum, ähnlich dem bekannten ABAP-Systemfeld `sy-datum`. Wenn Sie sich auf einem älteren Release befinden, übergeben Sie hart codiert einfach das aktuelle Datum in der Form JJJJMMTT – Sie optimieren auch dies später in Abschnitt 7.2.3, um eine benutzergesteuerte Kursermittlung zu ermöglichen. Für die umgerechneten Kennzahlen definieren Sie mittels `@DefaultAggregation` als Aggregationsfunktion `#SUM`. Listing 7.6 zeigt den implementierten Cube View.

```
@AbapCatalog.sqlViewName: 'ZI_AC_SOI'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'EPM: Cube Sales Order Items'
@Analytics.dataCategory: #CUBE
@VDM.viewType: #COMPOSITE
define view ZI_AC_SalesOrderItem
 as select from ZI_AF_SalesOrderItem as SOI
  association[1..1] to ZI_AD_SalesOrderHeader as _SOH
   on SOI.SalesOrderKey = _SOH.SalesOrderHeaderKey
  association[1..1] to ZI_AD_Product as _Product
   on SOI.ProductKey = _Product.ProductKey {
  key _SOH.SalesOrderID,
  key SOI.SalesOrderItemPosition,
   _SOH.LifecycleStatus,
   _SOH.BillingStatus,
   _SOH.DeliveryStatus,
```



```
   _SOH.OverallStatus,
   _Product.ProductID,
   _Product.Category,
   _SOH.BusinessPartnerID as BuyerID,
   _SOH._BuyerAddress.Country as BuyerCountry,
   _Product.BusinessPartnerID as SupplierID,
   _Product._SupplierAddress.Country as SupplierCountry,
   @Semantics.currencyCode: true
   SOI.Currency as OriginalCurrency,
   @Semantics.amount.currencyCode: 'OriginalCurrency'
   @DefaultAggregation: #SUM
   SOI.GrossAmount as GrossAmount_OC,
   @Semantics.amount.currencyCode: 'OriginalCurrency'
   @DefaultAggregation: #SUM
   SOI.NetAmount as NetAmount_OC,
   @Semantics.amount.currencyCode: 'OriginalCurrency'
   @DefaultAggregation: #SUM
   SOI.TaxAmount as TaxAmount_OC,
   @Semantics.currencyCode: true
   cast('EUR' as abap.cuky(5))     as TargetCurrency,
   @Semantics.amount.currencyCode: 'TargetCurrency'
   @DefaultAggregation: #SUM
   currency_conversion(
    amount             => GrossAmount,
    source_currency    => Currency,
    target_currency    => cast('EUR' as abap.cuky(5)),
    exchange_rate_date => cast($session.system_date as dats),
    error_handling     => 'SET_TO_NULL') as GrossAmount_TC,
   @Semantics.amount.currencyCode: 'TargetCurrency'
   @DefaultAggregation: #SUM
   currency_conversion(
    amount             => NetAmount,
    source_currency    => Currency,
    target_currency    => cast('EUR' as abap.cuky(5)),
    exchange_rate_date => cast($session.system_date as dats),
    error_handling     => 'SET_TO_NULL') as NetAmount_TC,
   @Semantics.amount.currencyCode: 'TargetCurrency'
   @DefaultAggregation: #SUM
   currency_conversion(
    amount             => TaxAmount,
    source_currency    => Currency,
    target_currency    => cast('EUR' as abap.cuky(5)),
    exchange_rate_date => cast($session.system_date as dats),
```

```
    error_handling     => 'SET_TO_NULL') as TaxAmount_TC,
  @ObjectModel.foreignKey.association: '_SOH'
  SOI.SalesOrderKey,
  _SalesOrderHeader,
  @ObjectModel.foreignKey.association: '_Product'
  SOI.ProductKey,
  _Product
}
```

Listing 7.6 Cube View »ZI_AC_SalesOrderItem«

Innerhalb des Cube Views definieren Sie abschließend die Bezeichnungen der einzelnen Felder, die dem Benutzer angezeigt werden. Sie setzen hierzu für jedes einzelne Attribut die Bezeichnung per Annotation, beispielsweise `@EndUserText.label: 'Gross Amount'` für das neu berechnete Attribut `GrossAmount_TC`.

[!]

Kennzahlen aus Dimension View

Für bestimmte Anforderungen ist es grundsätzlich auch möglich, einen Dimension View statt eines Cube Views als Datenquelle zu nutzen. In der Regel enthält dieser View keine Kennzahlen – man bedient sich in diesem Fall häufig der Aggregationsfunktion `Count`. Die Anzahl der Datensätze, die die Dimension zurückliefert, fungiert auf diese Weise als Hilfskennzahl. Sollte Ihr Dimension View echte Kennzahlen besitzen, können diese selbstverständlich ebenfalls an den Query View weitergereicht werden.

Query View implementieren

Vorschau des Cube Views

Vor der Erstellung der Query zum Konsumieren der Daten kann der Cube View als *transienter Provider* über die Transaktion RSRTS_ODP_DIS (Vorschau transienter Provider) geprüft werden. Die Transaktion dient beispielsweise der Überprüfung von Assoziationen, Texten oder Hierarchien. Auf diese Art und Weise kann das Zusammenspiel der im analytischen Modell definierten Attribute und deren Annotationen kontrolliert werden. Zur Überprüfung des Cube Views kopieren Sie den von Ihnen definierten SQL-View-Namen des Cube Views (`@AbapCatalog.sqlViewName`) in das Feld **ODP-Name** (siehe Abbildung 7.3).

Klicken Sie den Button **Auswahl über Query**. Ihnen wird eine detaillierte Liste Ihrer Attribute dargestellt. Die Attribute sind nach ihrem Typ (z. B. Schlüssel, Einheit, Kennzahl) gruppiert dargestellt, wie Abbildung 7.4 zeigt.

Im rechten Bereich sehen Sie, welchen Attributen eine Dimension per Fremdschlüssel und/oder assoziiertem Text zugewiesen ist.

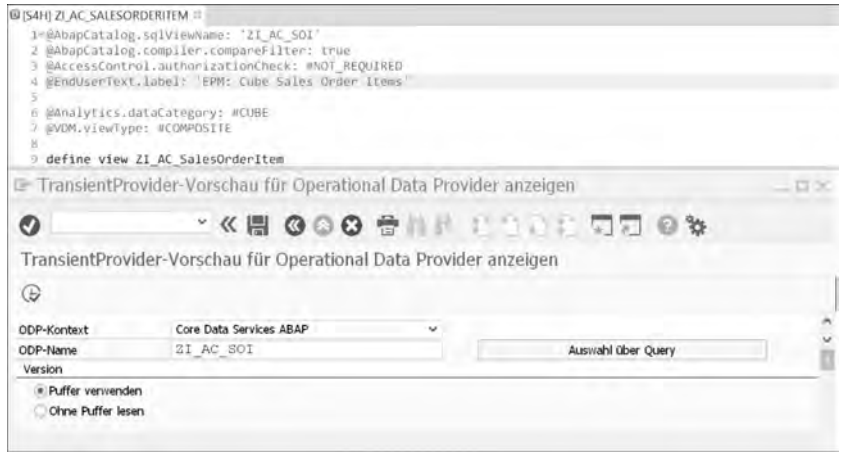


Abbildung 7.3 Selektion eines transienten Providers (Cube View)

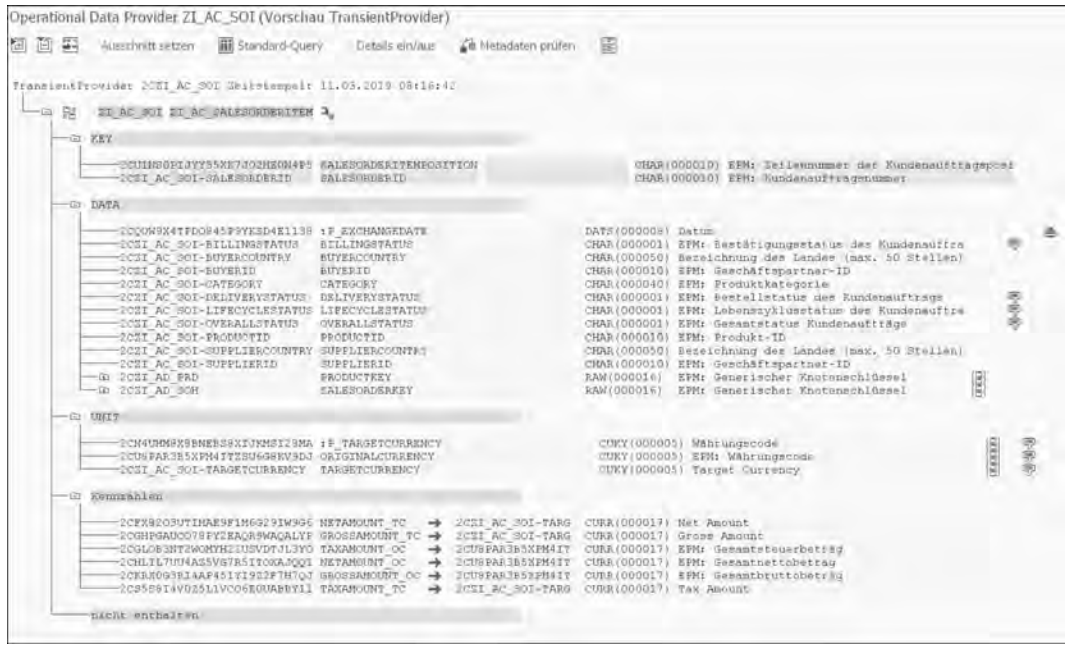


Abbildung 7.4 Attribute des transienten Providers (Cube View)

Query View implementieren

Zur Vervollständigung des analytischen Modells setzen Sie nun noch einen Query View oben auf das Datenmodell. Dieser Query View dient dem Frontend als zentraler Einstiegspunkt. Der Query View selbst enthält keine Asso-

ziationen und konsumiert alle Dimensionen und Kennzahlen vom Cube View. Der Query View ist ein Consumption View. Annotationen, die für das Frontend (beispielsweise für die Analytical List Page) relevant sind, werden innerhalb dieses Views implementiert.

OData-Service generieren

In unserem Beispiel dient der Query View der Generierung des OData-Service mithilfe der Annotation `@OData.publish: true` (Details zur Generierung von OData-Services finden Sie in Abschnitt 2.4, »Entwicklung von OData-Services mit SAP Gateway«). Zur Kennzeichnung des Query Views vergessen Sie nicht, `@Analytics.query: true` zu setzen. Basierend auf diesem Query View sollen die umgerechneten Betragsfelder zusätzlich als Leistungskennzahlen ausgewertet werden. Zu diesem Zweck annotieren Sie die drei Felder bereits an dieser Stelle noch zusätzlich mittels `@UI.dataPoint.title` und kennzeichnen sie damit als Leistungskennzahl.

Listing 7.7 zeigt Ihnen die Implementierung des CDS Views.

```
@AbapCatalog.sqlViewName: 'ZI_AQ_SOI'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'EPM: Query Sales Order Item'
@Analytics.query: true
@VDM.viewType: #CONSUMPTION
@OData.publish: true
define view ZI_AQ_SalesOrderItem
  as select from ZI_AC_SalesOrderItem {
    SalesOrderID,
    SalesOrderItemPosition,
    LifecycleStatus,
    BillingStatus,
    DeliveryStatus,
    OverallStatus,
    ProductID,
    Category,
    BuyerID,
    BuyerCountry,
    SupplierID,
    SupplierCountry,
    TargetCurrency,
    @UI.dataPoint.title: 'GrossAmount'
    GrossAmount_TC,
    @UI.dataPoint.title: 'NetAmount'
    NetAmount_TC,
    @UI.dataPoint.title: 'TaxAmount'
    TaxAmount_TC
}
```

Listing 7.7 Query View »ZI_AQ_SalesOrderItem«

OData-Service registrieren

Nachdem Sie den Query View aktiviert haben, müssen Sie den generierten OData-Service registrieren, bevor Sie ihn extern über SAP Gateway konsumieren können. Die erforderliche Registrierung ist in Abschnitt 2.4, »Entwicklung von OData-Services mit SAP Gateway«, beschrieben.

Query View testen

Zur Prüfung der erstellten Query auf syntaktische Korrektheit empfiehlt sich der Report `RSRTS_CDS_QUERY_CHECK`. Sollte es zu Problemen kommen, finden Sie im Applikationslog des Reports hilfreiche Informationen. Um die selektierten Daten zu kontrollieren, kann in einem ersten Schritt für den Query View die Option **Datenvorschau** F8 direkt auf den CDS View in Eclipse verwendet werden. Die Daten liegen hier allerdings immer auf demselben Aggregationslevel vor – alle Dimensionen werden berücksichtigt. Um eine aggregierte Sicht auf die Daten zu bekommen, bietet sich die Transaktion RSRT (Query-Monitor) an. Geben Sie den SQL-View-Namen des Query Views mit dem Präfix »2C« in das Feld **Query** ein, und führen Sie die Suche per F8 aus. Sie sollten nun tabellarisch die drei definierten Kennzahlen **Gross Amount**, **Net Amount** und **Tax Amount** aufsummiert der zentralen Tabelle entnehmen können. Auf der linken Seite können Sie anschließend beliebig Dimensionen (über den Pfeil nach unten) hinzufügen und das Aggregationslevel ändern. Abbildung 7.5 zeigt beispielsweise die drei Beträge aggregiert nach Produktkategorien für die Länder der Käufer.

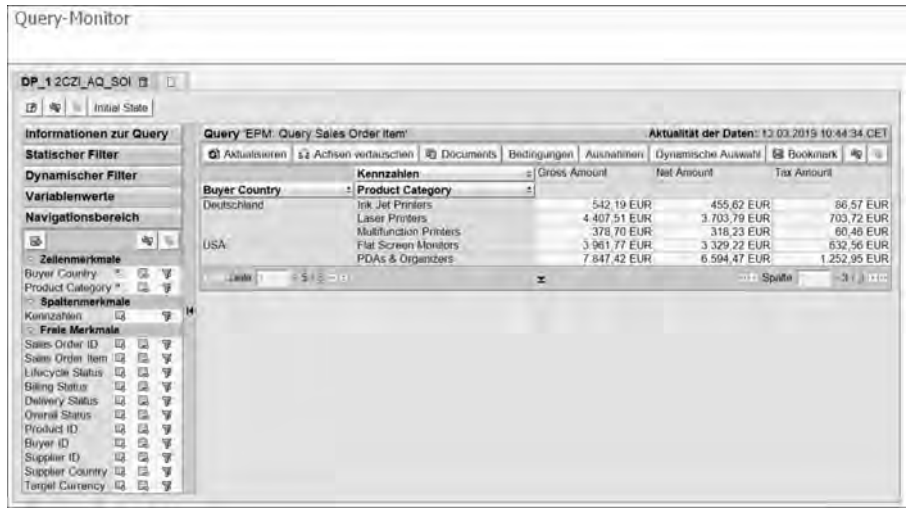


Abbildung 7.5 Testen des Query Views per Transaktion RSRT (Query-Monitor)

Alternative in SAP S/4HANA

Befinden Sie sich bereits auf einem SAP-S/4HANA-Release, haben Sie alternativ die Möglichkeit, die SAP-Fiori-App *View Browser* zu aktivieren. Sie können auf die App über Ihr SAP Fiori Launchpad zugreifen, sofern Ihnen die SAP-Standardrolle `SAP_BR_ANALYTICS_SPECIALIST` zugewiesen ist.

[!]

Zugriffskontrolle auf Query Views

Für analytische Query Views ergibt sich eine Besonderheit bezüglich des in Abschnitt 3.4.3, »Zugriffsschutz für CDS Views«, erläuterten Konzepts der *Data Control Language* (DCL) für CDS Views. Grundsätzlich wird die DCL des konsumierten Views zur Laufzeit ausgewertet. Der Query View stellt hier jedoch eine Ausnahme dar, da er lediglich der Definition der Eigenschaften einer Query dient. Die Selektion selbst nutzt den unterliegenden Cube View und die entsprechenden Dimension Views. Aus diesem Grund müssen Sie bei der Implementierung der Zugriffskontrolle innerhalb eines analytischen Szenarios diese Datenquellen separat schützen – die Zugriffskontrolle eines Query Views wird zur Laufzeit nicht ausgewertet.

Eingeschränkte Darstellung von Kennzahlen erreichen

In Kundenprojekten kommt es häufig zur Anforderung, eine Kennzahl auf bestimmte Werte eingeschränkt darstellen zu müssen. Diese Einschränkung kann z. B. abhängig von einer zeitlichen Begrenzung oder eines Dimensionswertes definiert sein. In unserem Beispiel soll der Gesamtbetrag durch insgesamt drei Kennzahlen folgendermaßen ausgewertet werden:

- voller Gesamtbetrag
- Gesamtbetrag mit `BillingStatus = 'P'` (bezahlt)
- Gesamtbetrag mit `OverallStatus = 'N'` (neu)

Um dies zu erreichen, können Sie im Query View zusätzliche Felder definieren, die Sie abhängig von der entsprechenden Bedingung mithilfe einer bestehenden Kennzahl (in unserem Fall dem Gesamtbetrag) befüllen. Listing 7.8 zeigt Ihnen, wie die Feldliste Ihres Query Views aussehen könnte:

```
{…
TargetCurrency,
GrossAmount_TC,
@EndUserText.label: 'Gesamtbetrag (bezahlter Buchungsstatus)'
@Semantics.amount.currencyCode: 'TargetCurrency'
case when BillingStatus = 'P' then GrossAmount_TC
     else 0
end as PaidBillStatusGrossAmount_TC,
@EndUserText.label: 'Gesamtbetrag (neuer Gesamtstatus)'
@Semantics.amount.currencyCode: 'TargetCurrency'
```



```
case when OverallStatus = 'N' then GrossAmount_TC
     else 0
end as NewOverallStatusGrossAmount_TC,
…}
```

Listing 7.8 Implementierung eingeschränkter Kennzahlen

Sie haben nun gelernt, wie Sie Ihr analytisches Modell im Backend definieren. In Abschnitt 7.3 zeigen wir Ihnen anhand des Beispiels einer Analytical List Page, wie Ihr Datenmodell im Frontend konsumiert werden kann.

Zunächst wollen wir Ihnen jedoch in Abschnitt 7.2.3 erläutern, wie Sie Ihr analytisches Modell mit weiteren Eingabeparametern bzw. initialen Filtern verfeinern können, und verschaffen Ihnen abschließend in Abschnitt 7.2.4 einen Überblick über die wichtigsten Annotationen im analytischen Kontext.

7.2.3 Parametrisierung des analytischen Modells und Filterung

Aus der klassischen ABAP-Entwicklung sind Sie mit Funktionsbausteinen oder Methoden und deren Signaturen in ABAP Objects (ABAP OO) vertraut. Auch innerhalb unseres Szenarios ist eine Parametrisierung der Views sinnvoll, denn die Kennzahlen berechnen sich aus Betragsfeldern, die an eine Währung geknüpft sind – in diesem Buch handelt es sich um die Brutto-, Netto- und Steuerbeträge einzelner Kundenauftragspositionen. Um diese in einem analytischen Kontext sinnvoll aggregieren zu können, müssen die Beträge in einer einheitlichen Währung dargestellt werden, da sie ansonsten nur schwer vergleichbar sind. Selbstverständlich kann man dies innerhalb der Implementierung hart codieren, allerdings beraubt man sich damit der Flexibilität, die gewünschte Zielwährung von außen ändern zu können.

Eingabeparameter definieren

ABAP CDS bietet Ihnen die Möglichkeit, einzelne Felder als Eingabeparameter für den CDS View zu definieren (tabulare Parameter werden nicht unterstützt). Diese Parameter können in der Aufrufhierarchie der CDS Views über den konsumierten CDS View per `SELECT`-Befehl beliebig tief nach unten propagiert werden und an entsprechender Stelle eine konkrete Aufgabe übernehmen.

Währungsumrechnung flexibel gestalten

Mithilfe dieses Parameters werden Sie im nächsten Schritt die Währungsumrechnung nach den Wünschen des Benutzers konfigurierbar implementieren. Zu diesem Zweck führen Sie die Zielwährung selbst als Parameter ein. Zusätzlich ermöglichen Sie dem Benutzer noch, über einen zweiten Parameter das Umrechnungsdatum festzulegen. Dies ist vor allem in pro-

duktiven Szenarien mit großen Beträgen eine essenzielle Anforderung – alternativ ist es auch möglich, das Umrechnungsdatum innerhalb Ihrer CDS Views aus einem fachlichen Datumsfeld abzuleiten und die Währungsumrechnung auf diese Weise noch flexibler zu gestalten.

Da die Parameter vom Benutzer eingegeben werden sollen, müssen diese logischerweise bereits im Query View auf oberster Ebene implementiert werden. Die Währungsumrechnung selbst findet im Cube View statt, aus diesem Grund muss zudem sichergestellt werden, dass die Parameter in der CDS-Aufrufkette nach unten weitergegeben werden. Listing 7.9 zeigt, wie Sie dies implementieren.

```
define view ZI_AQ_SalesOrderItem
  with parameters
    @Consumption.defaultValue: 'EUR'
    p_TargetCurrency :  snwd_curr_code,
    @Environment.systemField: #SYSTEM_DATE
    p_ExchangeDate    : dats
  as select from ZI_AC_SalesOrderItem(
        p_TargetCurrency: $parameters.p_TargetCurrency,
        p_ExchangeDate:   $parameters.p_ExchangeDate)
{ ... }
```

Listing 7.9 Implementierung der Eingabeparameter im Query View

Die Annotationen der beiden Parameter sind nicht verpflichtend, ermöglichen Ihnen allerdings, Default-Werte für Ihre Parameter zu vergeben und damit ihre Wiederverwendbarkeit zu verbessern. Soll der Default-Wert zur Laufzeit vom System ermittelt werden, bietet Ihnen die Annotation `@Environment.systemField` einen Auszug an Systemfeldern an.

Parameter der Schnittstelle hinzufügen

Bevor Sie den CDS View erfolgreich aktivieren können, müssen Sie die Parameter innerhalb des Cube Views auch der Schnittstelle hinzufügen. Nun können Sie innerhalb der berechneten Beträge Ihre initial implementierte Datumslogik durch den Parameter austauschen. Als Kurzschreibweise könn en Sie zur Identifikation einer Variablen statt `$parameters.` auch das Präfix : verwenden.

Listing 7.10 verdeutlich dies anhand eines Auszuges aus der Cube-View-Implementierung und des Betragsfeldes `GrossAmount`.

```
define view ZI_AC_SalesOrderItem
  with parameters
    @Consumption.defaultValue: 'EUR'
    p_TargetCurrency :  snwd_curr_code,
    @Environment.systemField: #SYSTEM_DATE
    p_ExchangeDate   : dats
 as select from ZI_AF_SalesOrderItem as SalesOrderItem
  …
  @EndUserText.label: 'Target Currency'
  @Semantics.currencyCode: true
  :p_TargetCurrency as TargetCurrency,
  @EndUserText.label: 'Gross Amount'
  @Semantics.amount.currencyCode: 'TargetCurrency'
  @DefaultAggregation: #SUM
  currency_conversion(
    amount             => GrossAmount,
    source_currency    => Currency,
    target_currency    => :p_TargetCurrency,
    exchange_rate_date => :p_ExchangeDate,
    error_handling     => 'SET_TO_NULL') as GrossAmount_TC,
…
```

Listing 7.10 Verwendung der Eingabeparameter im Cube View

Eingabeparameter testen

Testen Sie Ihre Implementierung erneut mittels Transaktion RSRT. Sie sollten nun die Möglichkeit haben, die Eingabeparameter über den Bereich **Variablen** mit den gewünschten Werten zu versorgen. Die Bezeichnung der Parameter wird in unserem Fall abhängig von der Anmeldesprache des Benutzers aus den entsprechenden Datenelementen ermittelt. Abbildung 7.6 zeigt, wie der Query-Monitor die Benutzereingabe visualisiert.

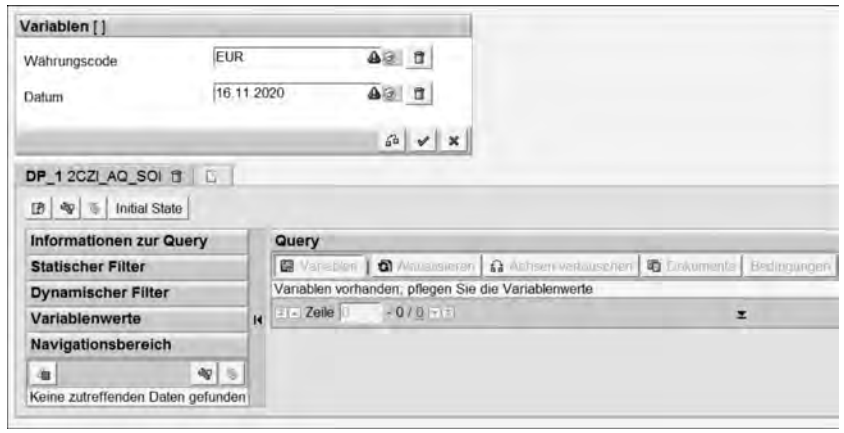


Abbildung 7.6 Eingabeparameter des CDS Views im Query-Monitor

Hierbei ist zu bemerken, dass die Visualisierung eines Query Views und der zugehörigen Parameter eigenverantwortlich durch das verwendete Front-

end-Tool übernommen wird. Eine standardisierte Darstellung ist nicht gewährleistet. Abhängig vom Customizing der Währungstabellen Ihres Systems sollten sich die Beträge nun ändern, wenn Sie das Umrechnungsdatum anpassen – in jedem Fall, wenn Sie die Währung ändern.

Filter definieren

Zusätzlich zu Eingabeparametern ist es möglich, innerhalb des Query Views für einzelne Dimensionsattribute Filter zu definieren. Dies kann vor allem dann sinnvoll sein, wenn die Datenmenge nur im Kontext einer bestimmten Ausprägung einer Dimension sinnvoll ist. Auch Performanceprobleme beim Laden des vollständigen Datenbestandes können auf diese Weise umgangen werden. Sie haben dabei die Möglichkeit, Filter als obligatorisch oder optional zu definieren. Die Definition eines optionalen Filters innerhalb des Query Views sehen Sie in Listing 7.11.

```
@Consumption.filter: {
    selectionType: #SINGLE,
    multipleSelections: false,
    mandatory: false
}
BuyerCountry,
```

Listing 7.11 Annotation zur Definition eines Filters im Query View

Die Annotationen `@Consumption.filter.*` bieten Ihnen hierbei zusammen mit der Annotation `@Consumption.derivation.*` weitere Möglichkeiten zur Definition Ihrer Filterkriterien.

Filter testen

Testen Sie nun Ihren Query View erneut. Abbildung 7.7 zeigt Ihnen den angepassten Selektionsbildschirm innerhalb des Bereichs **Variablen**.

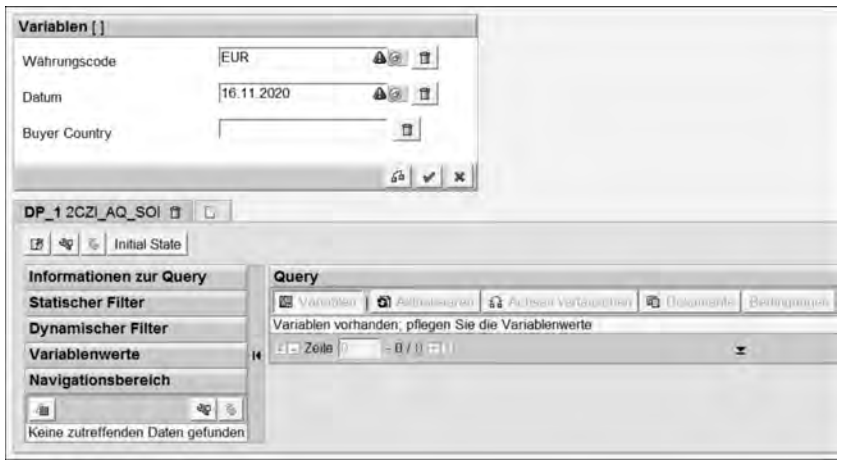


Abbildung 7.7 Eingabeparameter und Filter des CDS Views im Query-Monitor

Der Query-Monitor setzt für obligatorische Felder (in diesem Fall die Eingabeparameter) ein rotes Warndreieck. Der angezeigte Text des zu filternden Attributs wird mittels der Annotation `@EndUserLabel.text` aus Ihrem Cube View ermittelt und ist aus diesem Grund im Gegensatz zu den automatisch abgeleiteten Bezeichnern der Eingabeparameter auf Englisch. In einem produktiven Szenario können Sie für die Texte mittels Transaktion SE63 Übersetzungen pflegen.

Geben Sie beispielsweise **Deutschland** in das Feld **Buyer Country** ein, und prüfen Sie, ob der Filter korrekt angewandt wird. Entfernen Sie den optionalen Filter, wird der komplette Datenbestand analysiert.

Zurückschreiben von Daten ermöglichen

Sie haben nun über CDS Views der Typen Dimension, Fakt, Cube und Query das analytische Datenmodell erstellt. Der Vollständigkeit halber sei noch erwähnt, dass es zudem Views gibt, die sich auf ein Aggregationslevel beziehen. Die Datenquelle solcher Views ist klassischerweise der Cube View. Der »Aggregation Level View« selbst enthält – ähnlich dem Query View – keine Assoziationen. Die Besonderheit des Aggregation Level Views ist, dass er als einziger Typ Benutzern die Möglichkeit bietet, Daten zurückzuschreiben. Der View muss dann die Annotation `@Analytics.planning.enabled` enthalten. Der zugrunde liegende Cube View wiederum benötigt die Annotation `@Analytics.writeBack.className` und referenziert auf eine zu implementierende Klasse. Diese muss das Interface `IF_RODPS_ODP_WRITEBACK` implementieren. Sie kümmert sich um die Speicherung, beispielsweise direkt in eine separate Datenbanktabelle oder alternativ über ein Geschäftsobjekt, um die entsprechenden BOPF-Funktionalitäten nutzen zu können. Auf diese Funktionalität gehen wir im Kontext dieses Buches jedoch nicht weiter ein.

Der folgende Abschnitt 7.2.4 gibt Ihnen schließlich noch einmal einen Überblick über die aus unserer Sicht wichtigsten Annotationen für die Erstellung des analytischen Datenmodells.

7.2.4 Wichtige Annotationen

Wie in den vorangegangenen Abschnitten deutlich geworden ist, sind Annotationen für CDS essenziell. Da es mittlerweile Hunderte verschiedener Annotationen in der SAP-Hilfe zur Auswahl gibt, stellt es unserer Erfahrung nach für den Entwickler eine große Herausforderung dar, hier den Überblick zu bewahren. Aus diesem Grund listen wir in Tabelle 7.1 die aus unserer Sicht wichtigsten Annotationen zur Erstellung des analytischen Modells auf. Die Spalte »Typ« klassifiziert den Typ des relevanten analytischen CDS Views über dessen Anfangsbuchstaben: Query View (Q), Cube View (C), Fact View (F) oder Dimension View (D).

Annotation	Typ	Funktion
`@VDM.viewType.*`	alle	allgemeine (nicht analytisch-spezifische) CDS-View-Klassifizierung, primär informatorische Funktion
`@Analytics.*`	alle	Ermöglicht Definition des analytischen Modells und multidimensionalen Datenzugriff.
`@Analytics.query`	Q	Klassifiziert einen Query View.
`@Analytics.dataCategory`	C, F, D	Klassifiziert einen Dimension, Cube oder Fact View.
`@AnalyticsDetails.*`	Q, C	Dienen der Anpassung der analytischen Query (z. B. Formeln, Aggregationsverhalten, Sortierungen, Ausblendungen).
`@AnalyticsDetails.query.display`	Q	Definiert die Visualisierungsstrategie von Schlüssel- und Textfeldern (erst ab SAP-NetWeaver-Release 7.52 verfügbar).
`@AnalyticsDetails.query.axis`	Q	Definiert die Grundeinstellung für die Achse (Zeile/Spalte) einer Dimension.
`@AnalyticsDetails.exceptionAggregationSteps.*`	Q	Definiert das Aggregationsverhalten für bestimmte Elemente der Dimensionen spezieller.
`@AnalyticsDetails.query.formula`	Q	Dient der Definition von Formeln, die über SQL nicht abbildbar sind.
`@DefaultAggregation`	C	Definiert für ein Feld die Aggregationsfunktion und klassifiziert es als Kennzahl.
`@ObjectModel.foreignKey.association`	C	Definiert die Assoziation, deren repräsentatives Schlüsselfeld auf das annotierte Feld verweist.
`@ObjectModel.representativeKey`	D	Definiert das Feld des Primärschlüssels, das vom Konsumenten des Views als Anker in dessen Assoziation genutzt wird.
`@ObjectModel.text.*` & `@Semantics.text`	D	Verbindet ein Schlüsselfeld/eine ID mit einer Assoziation zur Ermittlung des Textfeldes – oder direkt mit dem Textfeld.

Tabelle 7.1 Für die Erstellung eines analytischen Datenmodells wichtige Annotationen

Annotation	Typ	Funktion
`@Semantics.*`	C, F, D	Dient allgemein der Klassifizierung einzelner Felder, als z. B. Adress-, Text-, Währungsfelder usw.
`@Consumption.filter.*`	Q	Ermöglicht die Filterung auf Elemente des unterliegenden Views.
`@Consumption.defaultValue`	Q, C	Ermöglicht die Definition eines Default-Wertes für einen Eingabeparameter.
`@Consumption.filter.defaultValue`	Q, C	Ermöglicht die Definition eines Default-Wertes für ein Feld.
`@Consumption.derivation`	Q, C	Ermöglicht die Ermittlung von Default-Werten für Filter (die gegebenenfalls auch ausgeblendet werden können).
`@Environment.systemField`	Alle	Ermöglicht die Definition eines Default-Wertes für einen Eingabeparameter basierend auf dem Wert eines Systemfeldes.
`@OData.publish`	Q	Dient dem Auto-Exposure des OData-Service (siehe Abschnitt 2.3, »Core Data Services«).

Tabelle 7.1 Für die Erstellung eines analytischen Datenmodells wichtige Annotationen (Forts.)

Eine vollständige Dokumentation aller CDS-Annotationen finden Sie in der SAP-Hilfe unter »ABAP-Programmiermodell für SAP Fiori«. Da sich der Umfang der verfügbaren Annotationen in CDS mit jedem SAP-NetWeaver-Release ändert, stellen Sie sicher, dass Sie die richtige Dokumentation verwenden.

Sollten Sie nicht auf dem aktuellsten Release arbeiten, ist es wahrscheinlich, dass Sie früher oder später auf Annotationen stoßen, die Sie gut verwenden könnten – die jedoch für Ihr Release noch nicht verfügbar sind. In diesem Fall haben Sie – abhängig von der verwendeten Frontend-Technologie – die Möglichkeit, Ihre Attribute an anderer Stelle mit Annotationen zu versehen. In Abschnitt 7.3 zeigen wir dies beispielhaft für die Analytical List Page anhand des *Annotation Modelers*.

7.3 Visualisierung mit der Analytical List Page

In Abschnitt 7.2, »Analytische Core Data Services«, haben Sie im Backend Ihr analytisches Datenmodell implementiert und über die Annotation `@OData.publish: true` als OData-Service über SAP Gateway exponiert. In diesem Abschnitt geben wir Ihnen nun einen Überblick über die aus unserer Sicht gängigsten Möglichkeiten, diesen OData-Service im Frontend zu konsumieren und damit das analytische Modell für den Benutzer verwendbar zu machen.

Benutzeroberflächen mit SAP Fiori Elements

Wie bereits in Kapitel 6, »Entwicklung transaktionaler Benutzeroberflächen«, erläutert, stellt SAP Fiori Elements eine zentrale Säule der künftigen Frontend-Entwicklung dar. SAP stellt Ihnen dabei kontinuierlich neue Floorplans und Möglichkeiten zur Verfügung, um auch ohne tiefes Wissen in JavaScript bzw. SAPUI5 SAP-Fiori-Apps zu entwickeln. Im Folgenden veranschaulichen wir Ihnen dies anhand des Floorplans Analytical List Page (ALP).

Analytical List Page und SAP Business Suite

Die Analytical List Page ermöglicht eine Vielzahl an Interaktionen, über die ein Analyst verschiedenste Blickwinkel auf Daten und Kennzahlen erhält. Der Floorplan ist vor allem in neu veröffentlichten Apps von SAP, wie beispielsweise der Custom Code Migration App, sehr verbreitet. Die ALP ist ab SAPUI5-Release 1.48 verfügbar. Das bedeutet, dass Sie die ALP grundsätzlich bereits auf der SAP Business Suite verwenden können, sofern Sie eine entsprechende SAPUI5-Version im Einsatz haben. Dabei empfehlen wir Ihnen jedoch mindestens SAP-NetWeaver-Release 7.50, da vorher wichtige CDS-Annotationen, wie beispielsweise `@OData.publish: true`, noch nicht zur Verfügung stehen. Sie werden im Folgenden feststellen, dass zudem weitere CDS-Annotationen, wie beispielsweise `@UI.presentationVariant`, hilfreich sind, die erst ab SAP-NetWeaver-Release 7.51 zur Verfügung stehen. Dieses Release steht Ihnen auf der SAP Business Suite nicht zur Verfügung. Wie in Kapitel 6 erläutert, können Sie in dieser Situation ein eigenständiges Annotationsfile hochladen oder den Annotation Modeler einsetzen, um innerhalb der ALP zu Ihren extern konsumierten Annotationen lokale Annotationen hinzuzufügen.

Analytical List Page – Überblick

In unserem Beispiel entwickeln Sie die Analytical List Page exemplarisch unter SAP-NetWeaver-Release 7.50. Abbildung 7.8 zeigt Ihnen zunächst die wichtigsten Bestandteile und Funktionen der ALP im Rahmen der Applikation, die Sie innerhalb dieses Abschnitts selbst implementieren.

Abbildung 7.8 Vollständig implementierte Analytical List Page

Die Applikation gliedert sich in folgende Bereiche:

- **Variantenmanagement**:
  Im linken oberen Bereich finden Sie den Punkt **Standard**, über den sich die Darstellungsvariante verwalten lässt ❶. Der Benutzer kann hier unterschiedlichste persönliche Änderungen an der Visualisierung der Daten oder der Konfiguration von Filtern als Variante abspeichern. Er kann eine Konfiguration für sich als Standard festlegen und diese auch als eigene Kachel auf dem SAP Fiori Launchpad ablegen oder die aktuelle Anzeige per E-Mail an andere Benutzer verschicken.
- **Kennzahlenübersicht**:
  Rechts neben dem Variantenmanagement können Sie bis zu drei spezielle Kennzahlen einbinden ❷. Im Beispiel implementieren Sie die Kennzahl »Nettobetrag eines Käufers« bzw. »Land des Käufers«. Der Benutzer erhält hierdurch unmittelbar wichtige Informationen – die Werte können über die Farbgebung als kritisch angezeigt werden.
- **Kennzahlendetails**:
  Bewegt der Benutzer die Maus über die Kennzahl, so öffnet sich eine Karte mit detaillierten Informationen zur Kennzahl ❸. Technisch ist es möglich, mittels dieser Karte eine weiterführende Navigation in eine externe Applikation zu implementieren, die mit dem Kontext der Karte bzw. eines vom Benutzer markierten Bereichs gestartet wird.

- **Filter**:
  Direkt unter dem Variantenmanagement befindet sich ein Bereich zum Filtern der Daten. Der Benutzer kann *visuelle* Filter mit Diagrammen nutzen ❹, um direkt auf diejenigen Ausprägungen einer Dimension zu filtern, die die maximalen (oder minimalen) Werte enthalten. Dies ist vor allem hilfreich, wenn der Benutzer weniger an konkreten Ausprägungen als an Ausreißern einer Dimension interessiert ist.

  Rechts oben kann der Benutzer auf sogenannte *kompakte* Filter umschalten ❺ – hier hat er gewöhnliche Selektionsfelder zur Verfügung, in denen er konkrete Ausprägungen einer Dimension eingeben kann.

  Visuelle und kompakte Filter wirken gemeinsam als angewandter Filter.

- **Datenvisualisierung**:
  Unter dem versteckbaren Filterbereich befindet sich der Hauptbereich der Analytical List Page. Der Benutzer hat ein Diagramm ❻ und/oder eine Tabelle ❼ zur Auswahl, um die Daten zu visualisieren. In unserer Standardkonfiguration werden sowohl ein Diagramm als auch eine Tabelle dargestellt – der Benutzer kann dies in der Toolbar rechts oberhalb der Visualisierung umschalten ❽.

  Abbildung 7.8 visualisiert die Daten auf dem Aggregationslevel, das durch den Entwickler per entsprechenden Annotationen implementiert wurde. Dies ist allerdings nur die Standardansicht. Der Benutzer ist maximal flexibel. Er kann in Dimensionen beliebig tief einsteigen und diese jederzeit ändern. Er kann zusätzliche implementierte Kennzahlen nach Belieben hinzufügen und auch den Diagrammtyp wechseln. Hier wird ihm eine Vielzahl an Möglichkeiten »out of the box« zur Verfügung gestellt.

  Die Tabelle, klassischerweise unter dem Diagramm angesiedelt, visualisiert die gefilterten Daten ebenfalls. Der Benutzer kann auch hier das Level der Aggregation selbst beeinflussen, indem er sich zusätzliche Spalten in die Ansicht holt oder andere entfernt. Der Benutzer kann über einen Klick die aktuelle Ansicht der Daten nach Excel exportieren. Für den Entwickler besteht die Möglichkeit, für die Zeilen der Tabelle eine Navigation zu implementieren, sodass dem Benutzer weitere Aktionen für ausgewählte Datensätze (auch in externen Applikationen) zur Verfügung stehen.

7.3.1 Vorbereitende Schritte

Voraussetzungen im Backend

Wir gehen in diesem Abschnitt davon aus, dass Sie in Abschnitt 7.2 das analytische CDS-Datenmodell implementiert haben. Somit haben Sie einen

funktionalen OData-Service, den Sie nun als Basis der Frontend-Entwicklung nutzen können. Sollten Sie das Modell nicht implementiert haben, nutzen Sie den CDS Query View `ZI_AQ_SalesOrderItem`, und generieren Sie basierend darauf den OData-Service, wie in Abschnitt 2.4, »Entwicklung von OData-Services mit SAP Gateway«, beschrieben.

Im Folgenden führen wir Sie Schritt für Schritt durch die Implementierung der SAP-Fiori-App. Spezielles Augenmerk legen wir dabei auf die konkrete Anwendung verschiedener CDS-Annotationen, mit denen Sie die Visualisierung der Daten beeinflussen können. Bevor wir hier tiefer einsteigen, geben wir Ihnen jedoch einen allgemeinen Überblick über den Aufbau und die Möglichkeiten des Floorplans Analytical List Page.

Analytisches Datenmodell simplifizieren

Zunächst wollen wir das Szenario für die ALP noch etwas simplifizieren. Das analytische Modell ist mit finalem Stand aus Abschnitt 7.2 in der Lage, eine vom Benutzer gesteuerte Währungsumrechnung durchzuführen. Dies haben Sie bereits durch Eingabeparameter im Query View erzielt. Durch die Parameter hat der generierte OData-Service ein spezielles Metadatenmodell mit eigenen Entitäten für Parameter und das Ergebnis (technische `Results`-Entität), das von den gängigen Metadatenmodellen abweicht. Der Umgang mit diesem speziellen Modell ist nach aktuellem Stand in der Dokumentation nicht behandelt und hat zahlreiche Sonderfälle in der Implementierung als Ergebnis, beispielsweise bei der Definition von visuellen Filtern oder Suchhilfen für kompakte Filter. Um zu vermeiden, dass Sie sich im folgenden Beispiel mit diesen Sonderfällen auseinandersetzen müssen, entfernen Sie in Ihrem analytischen Modell die Eingabeparameter. Auf diese Weise können Sie sich auf die Implementierung der wesentlichen Bestandteile der ALP konzentrieren. Listing 7.12 zeigt Ihnen, wie Sie den Query View anpassen müssen, um den daraus generierten OData-Service zu vereinfachen.

```
define view ZI_AQ_SalesOrderItem
/*
  with parameters
    @Consumption.defaultValue: 'EUR'
    p_TargetCurrency :  snwd_curr_code,
    @Environment.systemField: #SYSTEM_DATE
    p_ExchangeDate    : dats
 */
  as select from ZI_AC_SalesOrderItem(
  /*
        p_TargetCurrency: $parameters.p_TargetCurrency,
        p_ExchangeDate:   $parameters.p_ExchangeDate)
```

```
  */
        p_TargetCurrency: 'EUR',
        p_ExchangeDate:   '20181231')
{
  …
    /*
    @Consumption.filter: {
        selectionType: #SINGLE,
        multipleSelections: false,
        mandatory: false
    }
    */
    BuyerCountry,
  …
}
```

Listing 7.12 Vereinfachung des Query Views durch Auskommentieren der Eingabeparameter

Führen Sie diesen Schritt unbedingt durch, da die folgende Beschreibung ansonsten nicht zu Ihrem Szenario passt.

7.3.2 Analytical List Page implementieren

Zur Implementierung der SAP-Fiori-App verlassen Sie die Entwicklungsumgebung Eclipse mit ihren ABAP Development Tools und verwenden für die Frontend-Entwicklung die bereits in Kapitel 6, »Entwicklung transaktionaler Benutzeroberflächen«, angesprochene SAP Web IDE Full-Stack.

Analytical List Page anlegen

Um eine neue Analytical List Page anzulegen, führen Sie folgende Schritte durch:

1. Öffnen Sie das Kontextmenü Ihres Arbeitsbereichs, und wählen Sie die Option **New • Project from Template**.
2. Wählen Sie die **Analytical List Page/Object Page** aus SAP Fiori Elements, und klicken Sie auf **Next**.
3. Als Projektnamen vergeben Sie »SalesOrderItems«, der Namensraum ist »Z«. Vergeben Sie noch einen geeigneten Titel der App, und klicken Sie auf **Next**.
4. Wählen Sie nun Ihr angebundenes Backend-System aus, und melden Sie sich am System an. Finden Sie den Service `ZI_AQ_SALESORDERITEM_CDS`, wählen Sie ihn aus, und klicken Sie auf **Next**.
5. Markieren Sie alle Annotationsquellen, und klicken Sie erneut **Next**.

6. Im nächsten Schritt müssen Sie noch finale Konfigurationsschritte für Ihre Applikation durchführen. Zunächst legen Sie das Data Binding auf Ihre OData Collection fest. In der Auswahl sollte sich nun neben **Additional Metadata** lediglich die Entitätsmenge `ZI_AQ_SALESORDERITEM` befinden. Wählen Sie diese aus. Sollten Sie in der Auswahl andere Möglichkeiten, wie beispielsweise Entitäten für **Parameter** oder **Results**, finden, so hat die Vereinfachung des Query Views im letzten Schritt nicht funktioniert. Stellen Sie dann sicher, dass der Query View aktiviert ist, führen Sie ein vollständiges Aktualisieren (Strg + ⇧ + R) auf die SAP Web IDE durch, und starten Sie den Wizard zur Applikationserstellung erneut.
7. Setzen Sie den **Qualifier des App Descriptors** auf **Default**. Sie können hier einen beliebigen Wert wählen, notieren Sie sich diesen aber für die spätere Implementierung.
8. Als Tabellentyp wählen Sie **Analytical**. Sie können diesen Typ später per *App Descriptor* (besser bekannt als **manifest.json**) bei Bedarf anpassen. Beenden Sie die Erstellung der Applikation über **Finish**.

Wahl des richtigen Tabellentyps

Für unser Beispiel haben wir uns für die *Analytical Table* entschieden. Dieser Typ ist besonders geeignet, wenn es darum geht, große Datenmengen darzustellen und Zeilen miteinander zu vergleichen. Die Alternative ist die *Responsive Table*. Bei diesem Typ liegt der Fokus eher auf ganzen Zeilen und weniger auf einzelnen Zellen über mehrere Zeilen hinweg, wie bei der Analytical Table. Die Responsive Table passt sich dem jeweiligen Anwendergerät an und ist damit auch auf mobilen Endgeräten nutzbar, bietet aber keine Unterstützung für Gruppierungs- und Aggregationsfunktionalitäten. Demnach ist in analytischen Anwendungsfällen die Analytical Table zu bevorzugen. Müssen auch Smartphones unterstützt werden, ist die Responsive Table zu empfehlen – wobei Sie hierbei die Anzahl an dargestellten Datensätzen reduzieren und auf analytische Tabellenfunktionen verzichten müssen.

Sie haben nun eine neue Analytical-List-Page-Anwendung erstellt – in Ihrem Arbeitsbereich sollten Sie das Projekt `SalesOrderItems` finden.

Sie können es über den Button **Run** (oder alternative Alt + F5) ausführen und lokal per SAP Web IDE testen – wählen Sie hierzu **flpSandbox.html** aus. Obwohl Ihre Applikation lokal liegt, konsumieren Sie bereits die Daten Ihres Backend-Systems – aus diesem Grund müssen Sie sich noch einmal authentifizieren. Die lokale Ausführung der Applikation ist auch im produktiven Szenario während der Entwicklungsphase legitim. Um die Appli-

kation dann ins gewohnte SAP-Transportwesen einzubinden, müssen Sie die Applikation auf Ihr Backend-System deployen.

Wenn Sie alle Schritte korrekt durchgeführt haben, zeigt die Applikation an dieser Stelle noch keine Daten. Dies ist kein Problem. Die Analytical List Page benötigt zur Visualisierung der Daten noch weitere Informationen, wie die Daten aufbereitet werden sollen. Dies implementieren Sie im folgenden Abschnitt 7.3.3 anhand von Annotationen.

7.3.3 Annotationen für die Analytical List Page implementieren

Alle bisher implementierten CDS-Annotationen beziehen sich auf die Definition des analytischen Datenmodells. Die im Folgenden erstellten Annotationen befassen sich dagegen mit der Analytical List Page. In unserem Beispiel implementieren Sie diese mithilfe lokaler Annotationen über die SAP Web IDE, da diese Variante auch unter SAP NetWeaver 7.50 möglich ist. Sollten Sie sich bereits auf einem aktuelleren Release befinden, können Sie die Annotationen auch innerhalb des CDS Query Views oder einer entsprechenden Metadatenerweiterung (siehe Abschnitt 2.3.3, »CDS-Annotationen«) basierend auf dem CDS Query View definieren.

Annotation Modeler und Code Editor

Ihr in Abschnitt 7.3.2, »Analytical List Page implementieren«, erstelltes Projekt enthält die Datei **annotations.xml**. Öffnen Sie die Datei. Sie haben die Wahl zwischen dem formularbasierten Annotation Modeler und dem *Code Editor*. Beide Tools zeigen dieselbe Datei und sind synchronisiert – fügen Sie eine Annotation hinzu und wechseln das Tool, ist diese weiterhin sichtbar. Sie müssen sich demnach nicht für ein Tool entscheiden. Im Folgenden zeigen wir die Implementierung der Annotationen in XML-Schreibweise, also in der Visualisierungsform des Code Editors.

[+]

Werkzeug wählen

Unserer persönlichen Erfahrung nach unterstützt der Annotation Modeler Sie vor allem als Einsteiger, wenn Sie sich mit den zu implementierenden Annotationen vertraut machen wollen. Sie erhalten über Auswahlhilfen eine Übersicht, welche Annotationen zur Verfügung stehen und an welchen Stellen *Qualifier* vergeben werden sollten. Für erfahrenere Entwickler ist dagegen der Code Editor interessanter. Sie sollten bei seiner Verwendung bereits mit den Annotationen vertraut sein und wissen, welche Sie wann benötigen. Der Code lässt sich im Code Editor schneller schreiben – z. B. können Sie unkompliziert Stellen kopieren und einfügen.

Annotationen mit dem Code Editor implementieren

Im ersten Schritt öffnen Sie im Code Editor die Datei **annotations.xml**. Abhängig von Ihrer SAPUI5-Version kann die konkrete Implementierung der generierten Objekte im Folgenden leicht abweichen.

Sollten Sie im unteren Bereich ein XML-Tag `<Annotations Target="">` finden, so entfernen Sie dieses zusammen mit der darin eingebetteten Annotation vom Typ `UI.Facets`, da diese aufgrund des fehlenden Ziels ansonsten zu Problemen führen. Auch innerhalb des Ziels `Target="ZI_AQ_SALESORDERITEM_CDS.ZI_AQ_SALESORDERITEMType"` befindet sich eine Annotation vom Typ `UI.Facets`. Diesen Block lassen Sie unverändert – er steuert die Navigation von den Zeilen Ihrer Tabelle zu einer *Object Page*.

[!]

Navigationsmöglichkeiten

Die Object Page als Navigationsziel von SAP Fiori Elements ist direkt verfügbar – der Einfachheit halber verwenden Sie diese auch für das Beispiel dieses Buches. Dies ist im produktiven Szenario zwar ebenfalls ein valides Szenario, unserer Erfahrung nach wird meist aber eher ein Absprung in eine separate App implementiert. Dieser App werden dann die Daten der aktuellen Zeile Ihrer Tabelle (oder Teile davon) als *Start-up-Parameter* übergeben.

Auf diese Weise können Sie zunächst die ALP für analytische Zwecke nutzen, um bestimmte Sichten auf die Daten zu realisieren und erste Erkenntnisse zu ermöglichen. In einem weiteren Schritt kann der Benutzer auf Datensätze navigieren, die für ihn von hoher Relevanz sind und beispielsweise eine transaktionale Änderung innerhalb einer weiterführenden Applikation durchführen.

Qualifier definieren

Beim Erstellen der SAP-Fiori-App müssen Sie erneut einen Qualifier für den App Descriptor definieren. Sie finden diesen Wert, wenn Sie die Datei **manifest.json** öffnen und sich die generierte Konfiguration Ihrer App genauer ansehen (siehe Listing 7.13).

```
"sap.ui.generic.app": {
  "_version": "1.3.0",
  "pages": {
    "AnalyticalListPage|ZI_AQ_SALESORDERITEM": {
      "entitySet": "ZI_AQ_SALESORDERITEM",
      "component": {
        "name":"sap.suite.ui.generic.
                    template.AnalyticalListPage",
        "list": true,
        "settings": {
          "tableType": "AnalyticalTable",
```

```
            "multiSelect": false,
            "qualifier": "Default",
            "autoHide": true,
            "showGoButtonOnFilterBar": false,
            "condensedTableLayout": true,
            "keyPerformanceIndicators": {}
        }
    },
```

Listing 7.13 Konfiguration der Analytical List Page innerhalb der Datei »manifest.json«

Annotationen für die ALP

Dieser Qualifier dient als Ankerpunkt für die zentrale Annotation, mit der Sie die Aspekte der Visualisierung Ihrer Analytical List Page steuern können. Weitere Details zur Implementierung des technischen Qualifiers sind in Kapitel 6, »Entwicklung transaktionaler Benutzeroberflächen«, beschrieben. Das Konzept der Referenzierung über Qualifier wird von den meisten Annotationsgruppen unterstützt. Abbildung 7.9 gibt Ihnen zunächst einen Überblick über die Annotationen, die wir im Folgenden implementieren.

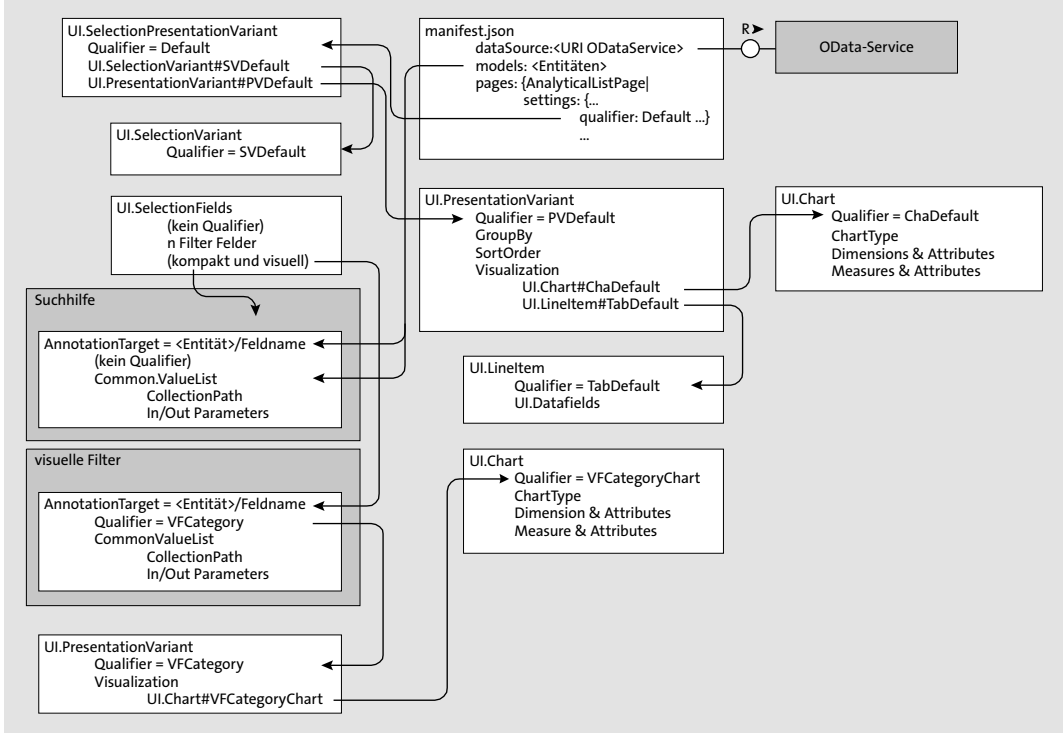


Abbildung 7.9 Zusammenhang der Annotationen für die Analytical List Page

Annotation »UI.selectionPresentationVariant«

Zentrale Annotation definieren

Zu Beginn benötigen Sie die zentrale Annotation `@UI.SelectionPresentationVariant`, die dann wiederum auf `@UI.SelectionVariant` und `@UI.PresentationVariant` referenziert. Diese Annotation wurde für transaktionale Applikationen bereits in Kapitel 6, »Entwicklung transaktionaler Benutzeroberflächen«, beschrieben. Im Folgenden prägen Sie sie für einen analytischen Anwendungsfall aus.

Kopieren Sie Listing 7.14 in die Datei **annotations.xml**, oder legen Sie die Annotation über den Annotation Modeler an. Die Annotation muss sich (wie alle anderen Annotationen, sofern nicht explizit anders beschrieben) innerhalb von `ZI_AQ_SALESORDERITEMType` befinden.

```
<Annotations Target="
    ZI_AQ_SALESORDERITEM_CDS.ZI_AQ_SALESORDERITEMType">
 <Annotation Term="UI.SelectionPresentationVariant"
             Qualifier="Default">
  <Record>
   <PropertyValue Property="Text" String="Default"/>
   <PropertyValue Property="SelectionVariant"
                  Path="@UI.SelectionVariant#SVDefault"/>
   <PropertyValue Property="PresentationVariant"
                  Path="@UI.PresentationVariant#PVDefault"/>
  </Record>
 </Annotation>
...
</Annotations>
```

Listing 7.14 Definition der zentralen Annotation »SelectionPresentationVariant«

In der zweiten Zeile von Listing 7.14 wird per `Qualifier="Default"` die Verbindung der Annotation zum Qualifier des App Descriptors als zentraler Einstiegspunkt für die Applikation hergestellt.

Diese Annotation beinhaltet zwei Referenzen auf `SelectionVariant` und `PresentationVariant`. Hierbei wird innerhalb des Pfades ebenfalls ein Qualifier `#SVDefault` bzw. `#PVDefault` angegeben. Dies bedeutet, dass es auch eine Annotation `@UI.SelectionVariant` (bzw. `@UI.PresentationVariant`) mit den entsprechenden Qualifier geben muss, auf die referenziert werden kann.

Annotation »UI.SelectionVariant«

Selektionsvariante definieren

Die Annotationen der Selektionsvariante dienen dazu, eine spezielle Kombination zwischen Parametern und Filtern zu kennzeichnen, worüber sich

eine initiale Anfrage eingrenzen lässt. In unserem Beispiel definieren Sie die Annotationsvariante zwar, nehmen jedoch keine Spezifizierung vor (siehe Listing 7.15).

```
<Annotation Term="UI.SelectionVariant" Qualifier="SVDefault">
  <Record>
    <PropertyValue Property="Text" String="Default"/>
  </Record>
</Annotation>
```

Listing 7.15 Definition der Selektionsvariante

Beim Wechsel auf die Annotationsansicht im Annotation Modeler erhalten Sie hier möglicherweise einen Fehler – er tritt aktuell bei Selektions- und Präsentationsvarianten vereinzelt auf und weist darauf hin, dass die entsprechende Variante nicht valide ist. Die Meldung hat jedoch keine Auswirkung auf die Funktionalität unserer App und kann im Weiteren ignoriert werden.

Annotation »UI.PresentationVariant«

Präsentationsvariante definieren

Die Präsentationsvariante wiederum ist deutlich spannender. Hier definieren Sie folgende Dinge:

- **Positionen nach Kundenaufträgen gruppieren**:
  Die Datensätze sind im analytischen Modell auf Basis der Kundenauftragspositionen definiert. Für die Anzeige der Daten in der Applikation ist es sinnvoll, diese nach ihren zugehörigen Kundenaufträgen zu gruppieren.
- **Positionen nach Bruttobeträgen sortieren**:
  Die wichtigste Kennzahl innerhalb unserer Analyse stellt der Bruttogesamtbetrag dar. Zur besseren Übersicht sortieren Sie die angezeigten Datensätze nach diesem Betrag absteigend.
- **Daten grafisch aufbereiten**:
  Die Applikation soll die hybride Ansicht innerhalb des Hauptbereichs unterstützen. Aus diesem Grund werden Sie sowohl einen Chart als auch eine Tabelle zur Visualisierung definieren und die notwendigen Annotationen über einen Qualifier referenzieren.

Weitere Informationen zu den Möglichkeiten der Präsentationsvariante finden Sie in Abschnitt 6.3, »Annotationen«.

Listing 7.16 zeigt, wie Sie die genannten Punkte in der Annotation `@UI.PresentationVariant` implementieren. Sie benötigen hierbei innerhalb der Annotation jeweils eine Eigenschaft `Property` mit entsprechender Ausprägung.

```
<Annotation Term="UI.PresentationVariant"
     Qualifier="PVDefault">
 <Record>
  <PropertyValue Property="Text" String="Default"/>
   <PropertyValue Property="GroupBy">
    <Collection>
     <PropertyPath>SalesOrderID</PropertyPath>
    </Collection>
   </PropertyValue>
   <PropertyValue Property="SortOrder">
    <Collection>
     <Record>
      <PropertyValue Property="Property"
                     PropertyPath="GrossAmount_TC"/>
      <PropertyValue Property="Descending"
                     Boolean="true" Bool="true"/>
     </Record>
    </Collection>
   </PropertyValue>
   <PropertyValue Property="Visualizations">
    <Collection>
     <AnnotationPath>@UI.Chart#ChaDefault</AnnotationPath>
     <AnnotationPath>@UI.LineItem#TabDefault</AnnotationPath>
    </Collection>
   </PropertyValue>
  </Record>
 </Annotation>
```

Listing 7.16 Definition der Präsentationsvariante

Annotation »UI.Chart«

Chart definieren

Sie haben die Definition der notwendigen Varianten abgeschlossen und mithilfe des Attributs `Visualizations` festgelegt, dass es eine Annotation `@UI.Chart` mit Qualifier `ChaDefault` gibt. Diese Annotation wird genutzt, um die Eigenschaften des Charts für die initiale Visualisierung zu definieren:

- Es gibt diverse Typen von Charts. Siewählen für die initiale Anzeige das Säulendiagramm (mit technischer Bezeichnung `Column`).
- Als initiale Dimensionen soll der Chart das Land des Käufers und die Produktkategorie zur Visualisierung verwenden.
- Auch die Kennzahlen müssen festgelegt werden. Hierbei sollen alle drei Beträge, also Brutto-, Netto- und Steuerbetrag, angezeigt werden.

Das folgende Listing 7.17 zeigt, wie die Annotation `@UI.Chart` innerhalb der Datei **annotations.xml** implementiert wird.

```
<Annotation Term="UI.Chart" Qualifier="ChaDefault">
 <Record Type="UI.ChartDefinitionType">
  <PropertyValue Property="ChartType"
       EnumMember="UI.ChartType/Column"/>
  <PropertyValue Property="Dimensions">
   <Collection>
    <PropertyPath>BuyerCountry</PropertyPath>
    <PropertyPath>Category</PropertyPath>
   </Collection>
  </PropertyValue>
  <PropertyValue Property="DimensionAttributes">
   <Collection>
    <Record Type="UI.ChartDimensionAttributeType">
     <PropertyValue Property="Dimension"
          PropertyPath="BuyerCountry"/>
     <PropertyValue Property="Role"
          EnumMember="UI.ChartDimensionRoleType/Category"/>
    </Record>
    <Record Type="UI.ChartDimensionAttributeType">
     <PropertyValue Property="Dimension"
          PropertyPath="Category"/>
     <PropertyValue Property="Role"
          EnumMember="UI.ChartDimensionRoleType/Category"/>
    </Record>
   </Collection>
  </PropertyValue>
  <PropertyValue Property="Measures">
   <Collection>
    <PropertyPath>GrossAmount_TC</PropertyPath>
    <PropertyPath>NetAmount_TC</PropertyPath>
    <PropertyPath>TaxAmount_TC</PropertyPath>
   </Collection>
  </PropertyValue>
  <PropertyValue Property="MeasureAttributes">
   <Collection>
    <Record Type="UI.ChartMeasureAttributeType">
     <PropertyValue Property="Measure"
          PropertyPath="GrossAmount_TC"/>
     <PropertyValue Property="Role"
          EnumMember="UI.ChartMeasureRoleType/Axis1"/>
    </Record>
    <Record Type="UI.ChartMeasureAttributeType">
     <PropertyValue Property="Measure"
          PropertyPath="NetAmount_TC"/>
     <PropertyValue Property="Role"
          EnumMember="UI.ChartMeasureRoleType/Axis1"/>
    </Record>
    <Record Type="UI.ChartMeasureAttributeType">
     <PropertyValue Property="Measure"
          PropertyPath="TaxAmount_TC"/>
     <PropertyValue Property="Role"
          EnumMember="UI.ChartMeasureRoleType/Axis1"/>
    </Record>
   </Collection></PropertyValue></Record></Annotation>
```

Listing 7.17 Definition des Charts

Hierbei sei bemerkt, dass der Chart innerhalb der ALP eine Vielzahl an Konfigurationsmöglichkeiten bietet, die dem Benutzer während der Ausführung der Applikation zur Verfügung stehen. Es lassen sich beispielsweise Anzahl und Art der Dimensionen ändern und der Chart-Typ wechseln.

Annotation »UI.LineItem«

Tabelle definieren

Nachdem der Chart definiert ist, bleibt noch die Tabelle. Hierfür haben Sie (in Listing 7.16) bereits den Qualifier `TabDefault` definiert. Die anzuzeigenden Spalten werden über die Annotation `UI.DataField` (eingebettet innerhalb einer Annotation `@UI.LineItem`) definiert. Für die initiale Anzeige der Daten wählen Sie die Felder `SalesOrderID`, `SalesOrderItemPosition`, `OverallStatus`, `ProductID`, `Category` und `GrossAmount_TC`. Der Benutzer hat zur Ausführungszeit die Möglichkeit, sich die Felder selbst zusammenzustellen und damit das Granularitätslevel der Daten anzupassen.

Listing 7.18 zeigt Ihnen die Implementierung der initialen Spaltenkonfiguration.

```
<Annotation Term="UI.LineItem" Qualifier="TabDefault">
 <Collection>
  <Record Type="UI.DataField">
   <PropertyValue Property="Value" Path="SalesOrderID"/>
  </Record>
  <Record Type="UI.DataField">
   <PropertyValue Property="Value"
                  Path="SalesOrderItemPosition"/>
  </Record>
  <Record Type="UI.DataField">
   <PropertyValue Property="Value" Path="OverallStatus"/>
  </Record>
  <Record Type="UI.DataField">
   <PropertyValue Property="Value" Path="ProductID"/>
  </Record>
  <Record Type="UI.DataField">
   <PropertyValue Property="Value" Path="Category"/>
  </Record>
  <Record Type="UI.DataField">
   <PropertyValue Property="Value" Path="GrossAmount_TC"/>
  </Record>
 </Collection>
</Annotation>
```

Listing 7.18 Definition der angezeigten Tabellenspalten

7.3.4 Analytical List Page testen

Sie haben nun die Spalten der Tabelle definiert, die beim initialen Laden der ALP angezeigt werden. Über Ihre Präsentationsvariante `PVQualifier` haben Sie zudem definiert, dass die Ergebnisse in der Spalte nach dem Attribut `SalesOrderID` gruppiert dargestellt werden sollen. Nachdem Sie den Chart, die Tabelle und auch die weiterführende Navigation implementiert haben, testen Sie die ALP. Starten Sie Ihre Applikation aus der SAP Web IDE. Öffnen Sie zu diesem Zweck das Kontextmenü Ihres Projekts, und wählen Sie **Run as SAP Fiori Launchpad Sandbox**. Die Applikation startet mit einem leeren Filterbereich. Minimieren Sie diesen zunächst mit dem zentralen Pfeil, der nach oben zeigt. Ihre App sollte in etwa aussehen, wie in Abbildung 7.10 gezeigt – sowohl im Chart als auch in der Tabelle sind Daten zu sehen.

Abbildung 7.10 Analytical List Page mit Daten im Chart- und Tabellenbereich

Fehlermeldung »Unable to load data«

Sollten Sie an dieser Stelle die Fehlermeldung »Unable to load data« erhalten, kann eine Ursache dafür sein, dass Sie eine zu große Datenmenge verarbeiten wollen und das Time-out Ihres Browsers zu schnell eintritt. In diesem Fall können Sie entweder das Intervall anpassen oder (für das Beispiel sinnvoller) die Datenmenge einschränken. Öffnen Sie zu diesem Zweck erneut Ihren Cube View `ZI_AC_SalesOrderItem` in Eclipse, und schränken Sie die Datensätze nach folgendem Schema ein (siehe Listing 7.19):

```
where
 _SalesOrderHeader.SalesOrderID = '0500000000' or
 _SalesOrderHeader.SalesOrderID = '0500000001' or
 _SalesOrderHeader.SalesOrderID = '0500000002'
```

Listing 7.19 Datensätze einschränken

Sollte das Problem weiterbestehen, öffnen Sie die *Chrome DevTools* F12, um im Bereich **Console** eine detailliertere Fehlerbeschreibung zu erhalten.

Grundsätzlich ist die Ausführung der Applikation aus der SAP Web IDE auch mit anderen Varianten (z. B. **Run as flpSandbox.html**) möglich. Wir empfehlen zum lokalen Testen einer Analytical List Page jedoch die Option **Run as SAP Fiori Launchpad Sandbox**, da es bei anderen Optionen häufig zu Anzeigefehlern kommt, z. B. in der Farbgebung von visuellen Filtern.

Analytical List Page als Benutzer bedienen

Sie haben in einer Analytical List Page verschiedene Möglichkeiten, als Benutzer die Sicht auf Ihre Daten zu ändern:

- **Chart**:
  Sie können beispielsweise den Typ des Charts, die Stufe des Drilldowns, die dargestellten Dimensionen usw. innerhalb des Charts ändern.
- **Tabellen**:
  In der Tabelle können Sie durch einen Klick auf den Pfeil einer Gruppierung im linken Bereich diese Gruppe detailliert ansehen. Über die Einstellungen können Sie zusätzliche Spalten hinzufügen oder andere ausblenden. Sie können die Daten zudem sortieren oder anderweitig gruppieren.
- **Navigation**:
  Auf der rechten Seite der Tabelle finden Sie einen Pfeil mit dem Tooltip **Details**. Klicken Sie darauf – dieser Pfeil realisiert die implementierte Navigation. Sie sollten auf eine aktuell noch relativ leere Object Page gelangen. Je nach Anforderung kann diese vom Entwickler per UI-Annotationen mit Daten eines OData-Service gefüllt oder eine andere Applikation als Ziel definiert sein. Es ist selbstverständlich auch möglich die Navigation innerhalb der **mainfest.json**-Datei zu unterdrücken.

Da dies den Umfang des Buches übersteigen würde, können wir an dieser Stelle nicht auf alle Möglichkeiten der Analytical List Page im Detail eingehen. Weitere Informationen erhalten Sie in den SAP Fiori Guidelines (siehe *http://s-prs.de/v678860*) sowie der Dokumentation zur Analytical List Page (siehe *http://s-prs.de/v678861*).

7.3.5 Filter implementieren

Wie bereits angesprochen, wird der obere Filterbereich Ihrer neuen Applikation aktuell leer dargestellt. Das liegt daran, dass Sie weder einen visuellen noch kompakten Filter zur initialen Anzeige implementiert haben. Da eine Sicht auf den Gesamtbestand nur in seltenen Fällen von Interesse ist, implementieren wir im Folgenden einen kompakten Filter. Sowohl der Chart als auch die Tabelle berücksichtigen die vom Benutzer definierten Filter zu jedem Zeitpunkt.

Kompakten Filter implementieren

Für unser Beispiel implementieren Sie exemplarisch einen kompakten Filter für das Feld `SalesOrderID`, mit dem sich die Visualisierung der Daten auf einzelne Kundenaufträge einschränken lässt.

Selektionsfeld definieren

Damit das Feld als kompakter Filter zur Verfügung steht, kennzeichnen Sie es als Selektionsfeld in Ihrer Datei **annotations.xml** (siehe Listing 7.20).

```
<Annotation Term="UI.SelectionFields">
  <Collection>
    <PropertyPath>SalesOrderID</PropertyPath>
  </Collection>
</Annotation>
```

Listing 7.20 Kompakter Filter für das Feld »SalesOrderID«

Aktualisieren Sie Ihre Applikation, und wechseln Sie den Filtermodus von **visueller Filter** auf **kompakter Filter**. Es erscheint nun ein Feld **Sales Order ID**. Geben Sie einen validen Wert für die Kundenauftragsnummer ein (Sie können einen Wert aus der Tabelle verwenden), und prüfen Sie, ob sich die Daten im Chart und in der Tabelle aktualisiert haben.

Suchhilfe implementieren

Wie Sie aus der klassischen ABAP-Oberflächenentwicklung wissen, sollte an jedes Eingabefeld auch eine entsprechende Unterstützung für den Benutzer implementiert sein. Das Anbinden einer Suchhilfe ist in Listing 7.21 beschrieben.

Hierbei ist zu beachten, dass diese Annotationen in CDS zum aktuellen Zeitpunkt nicht zur Verfügung stehen. Implementieren Sie Ihre Annotationen in CDS, so müssen Sie an dieser Stelle auf die lokalen Annotationen Ihrer Applikationen in der SAP Web IDE zurückgreifen. Beachten Sie, dass diese Annotationen nicht im Block `Annotations` des Ziels `Target="ZI_AQ_SALESORDERITEM_CDS.ZI_AQ_SALESORDERITEMType"` enthalten sind, sondern als separater Block mit eigenem Ziel hinzugefügt werden müssen.

```
<Annotations Target="ZI_AQ_SALESORDERITEM_CDS.ZI_AQ_
SALESORDERITEMType/SalesOrderID">
 <Annotation Term="Common.ValueList">
  <Record Type="Common.ValueListType">
   <PropertyValue Property="CollectionPath"
                  String="ZI_AQ_SALESORDERITEM"/>
   <PropertyValue Property="SearchSupported" Bool="true"/>
   <PropertyValue Property="Parameters">
    <Collection>
     <Record Type="Common.ValueListParameterOut">
      <PropertyValue Property="LocalDataProperty"
                     PropertyPath="SalesOrderID"/>
      <PropertyValue Property="ValueListProperty"
                     String="SalesOrderID"/>
     </Record>
```

```
    <Record Type="Common.ValueListParameterIn">
     <PropertyValue Property="LocalDataProperty"
                    PropertyPath="SalesOrderID"/>
     <PropertyValue Property="ValueListProperty"
                    String="SalesOrderID"/>
    </Record></Collection></PropertyValue></Record>
 </Annotation></Annotations>
```

Listing 7.21 Suchhilfe für kompakten Filter »SalesOrderID«

Führen Sie Ihre Applikation erneut aus, und prüfen Sie, ob die Suchhilfe korrekt angezeigt wird und funktional ist.

[!]

Unterschied zu Suchhilfen in transaktionalen Szenarien

Wenn Sie bereits mit der Entwicklung transaktionaler Applikationen mit SAP Fiori Elements vertraut sind, also beispielsweise List Reports mit Object Pages angelegt haben, haben Sie sicher festgestellt, dass die Anbindung von Suchhilfen in analytischen Applikationen auf andere Weise erfolgt. Im transaktionalen Szenario können Sie im CDS Consumption View über Fremdschlüsselbedingungen oder explizit modellierte Views (Annotation `@Consumption.valueHelp`) Suchhilfen relativ komfortabel einbinden. Dieses Vorgehen kann auf das analytische Szenario nicht einfach übertragen werden, da der Query View (der das Pendant zum Consumption View bildet) bei seiner Aktivierung keine Assoziationen erlaubt.

Visuellen Filter implementieren

Zusätzlich zum kompakten Filter implementieren Sie noch einen visuellen. Anwender sollen direkt beim Einstieg in unsere Applikation erkennen, welche Produktkategorien die höchsten Gesamtbeträge in Kundenaufträgen aufweisen, um diese im nächsten Schritt konkreter analysieren zu können. Zu diesem Zweck wählen Sie ein Kreisdiagramm (**Donut**) als Chart-Darstellungstyp. Es ermöglicht, die beiden stärksten Abweichungen in Relation zu den restlichen Kategorien darzustellen. Aktuell werden von visuellen Filtern noch die Typen **Bar** (zeigt die drei stärksten Abweichungen) und **Line** (visualisiert die Entwicklung einer Kennzahl über einen bestimmten Zeitraum) unterstützt.

Selektionsfeld definieren

Um einen visuellen Filter zu implementieren, sind mehrere Schritte notwendig (siehe Abbildung 7.9). Zunächst müssen Sie ein Feld als Selektionsfeld definieren – dieser Schritt ist analog zu dem der Definition eines kompakten Filters und in Listing 7.22 veranschaulicht.


```
<Annotation Term="UI.SelectionFields">
  <Collection>
    <!-- Compact Filter Fields -->
    <PropertyPath>SalesOrderID</PropertyPath>
    <!-- Visual Filter Fields -->
    <PropertyPath>Category</PropertyPath>
  </Collection>
</Annotation>
```

Listing 7.22 Definition des Feldes »Category« als Selektionsfeld

Visuellen Filter definieren

Im nächsten Schritt implementieren Sie die Produktkategorie als visuellen Filter. Dies ist der Definition einer Suchhilfe des kompakten Filters sehr ähnlich. Sie müssen ebenfalls mit konkretem Ziel auf das Feld »Produktkategorie« die Quellen der Daten (`CollectionPath`) und die entsprechenden Parameter definieren. Der entscheidende Unterschied ist, dass Sie für die Definition eines visuellen Filters noch einen Qualifier vergeben, den Sie im Folgenden dann aufgreifen. Listing 7.23 zeigt, wie die Implementierung aussieht.

```
<Annotations Target=
"ZI_AQ_SALESORDERITEM_CDS.ZI_AQ_SALESORDERITEMType/Category">
 <Annotation Term="Common.ValueList">
  <Record Type="Common.ValueListType">
   <PropertyValue Property="CollectionPath"
                  String="ZI_AQ_SALESORDERITEM"/>
   <PropertyValue Property="Parameters">
    <Collection>
     <Record Type="Common.ValueListParameterInOut">
      <PropertyValue Property="LocalDataProperty"
                     PropertyPath="Category"/>
      <PropertyValue Property="ValueListProperty"
                     String="Category"/>
     </Record>
    </Collection>
   </PropertyValue>
  <PropertyValue Property="PresentationVariantQualifier"
                 String="VFCategory"/>
  </Record>
 </Annotation>
</Annotations>
```

Listing 7.23 Definition des Feldes »Category« als visueller Filter

Achten Sie darauf, die Annotation wie bei der Suchhilfe zuvor in einem eigenen Block zu definieren und nicht fälschlicherweise eingebettet innerhalb des Ziels `ZI_AQ_SALESORDERITEM_CDS.ZI_AQ_SALESORDERITEMType`.

Präsentationsvariante definieren

Für den erstellten visuellen Filter muss nun eine Präsentationsvariante angelegt werden. Die Verbindung wird über den Qualifier `VFCategory` implementiert. Listing 7.24 zeigt die Definition der Präsentationsvariante, in der Sie den Chart als Visualisierungsart festlegen und auch hierfür einen Qualifier `VFCategoryChart` vergeben. Die Präsentationsvariante (und auch der spätere Chart) muss innerhalb des Ziels `ZI_AQ_SALESORDERITEM_CDS.ZI_AQ_SALESORDERITEMType` implementiert werden.

```
<Annotation Term="UI.PresentationVariant"
            Qualifier="VFCategory">
 <Record>
  <PropertyValue Property="Text"
                 String="Filter on Product Category"/>
  <PropertyValue Property="Visualizations">
   <Collection>
   <AnnotationPath>@UI.Chart#VFCategoryChart</AnnotationPath>
   </Collection>
  </PropertyValue>
 </Record>
</Annotation>
```

Listing 7.24 Präsentationsvariante für den visuellen Filter des Feldes »Category«

Chart definieren

Abschließend müssen Sie für den visuellen Filter noch den Chart und dessen Konfiguration implementieren. Die Verbindung wird durch den Qualifier `VFCategoryChart` implementiert (siehe Listing 7.25). Neben dem Typ des Charts müssen die Dimension »Produktkategorie« und die Kennzahl »Gesamtbetrag« definiert werden.

```
<Annotation Term="UI.Chart" Qualifier="VFCategoryChart">
 <Record Type="UI.ChartDefinitionType">
  <PropertyValue EnumMember="UI.ChartType/Donut"
       Property="ChartType"/>
  <PropertyValue Property="Dimensions">
   <Collection>
    <PropertyPath>Category</PropertyPath>
   </Collection>
  </PropertyValue>
  <PropertyValue Property="DimensionAttributes">
   <Collection>
```



```
    <Record Type="UI.ChartDimensionAttributeType">
     <PropertyValue Property="Dimension"
          PropertyPath="Category"/>
     <PropertyValue Property="Role"
          EnumMember="UI.ChartDimensionRoleType/Category"/>
    </Record>
   </Collection>
  </PropertyValue>
  <PropertyValue Property="Measures">
   <Collection>
    <PropertyPath>GrossAmount_TC</PropertyPath>
   </Collection>
  </PropertyValue>
  <PropertyValue Property="MeasureAttributes">
   <Collection>
    <Record Type="UI.ChartMeasureAttributeType">
     <PropertyValue Property="Measure"
          PropertyPath="GrossAmount_TC"/>
     <PropertyValue Property="Role"
          EnumMember="UI.ChartMeasureRoleType/Axis1"/>
    </Record></Collection></PropertyValue></Record></Annotation>
```

Listing 7.25 Chart des visuellen Filters von »Category«

Visuellen Filter testen

Im Filterbereich wird Ihr visueller Filter angezeigt, wie in Abbildung 7.11 zu sehen. Sie können den Filter nun nutzen und beispielsweise die stärkste Abweichung markieren.

Abbildung 7.11 Analytical List Page mit visuellem Filter auf die Produktkategorie

Die Daten des Charts und der Tabelle werden auf die Produktkategorie gefiltert. Um basierend auf der Produktkategorie weitere Einblicke in die Daten zu gewinnen, können Sie im Chart beispielsweise die Produktkategorie als Dimension entfernen (klicken Sie dazu auf die erste Dimension **Buyer Country** im linken Bereich des Charts), und fügen Sie über die Option **View By** (in älteren Versionen auch **Drill Down**) beispielsweise **Supplier Country** hinzu.

Mit mehr als einem Filter arbeiten

Es ist auch möglich, mehr als einen Filter für eine Applikation zu definieren. Mittels der Parameter, die Sie innerhalb der Annotation `@Common.ValueList` angeben, können Sie auch ein Zusammenspiel der Filter untereinander implementieren. Das bedeutet, dass durch die Auswahl innerhalb des ersten Filters der zweite visuelle Filter angepasst wird, da dessen Daten bereits durch den ersten Filter eingeschränkt werden. Dies ist aber optional, die Filter können auch unabhängig voneinander agieren. In jedem Fall wirken sich alle ausgewählten Filter (inklusive der kompakten Filter) immer auf die angezeigten Daten im Chart und in der Tabelle aus.

Nutzen von visuellen Filtern

In unseren bisherigen Projekten kamen visuelle Filter durchweg sehr gut an. Die optische Aufbereitung eines Selektionsfeldes erhöht die Attraktivität der Applikation erheblich, sie erfährt höhere Akzeptanz bei Anwendern. Darüber hinaus haben visuelle Filter jedoch einen weiteren entscheidenden Vorteil: Wie in unserem Beispiel ersichtlich, erhält der Benutzer direkt die Information, welche Produktkategorien die höchsten (oder alternativ auch die geringsten) Beträge in Kundenaufträgen aufweisen. Das erspart Anwendern die aufwendige Analyse der Produktkategorien. Über den Filter können sich Benutzer zudem leicht übermäßige Abweichungen vom Durchschnitt anzeigen lassen und sich diese detaillierter in ihrer Zusammensetzung ansehen.

7.3.6 Daten aus der Analytical List Page exportieren

Excel-Export aktivieren

In einem nächsten Schritt wollen wir Benutzern unserer Analytical List Page ermöglichen, Daten einer Tabelle in eine Excel-Datei zu exportieren.

1. Öffnen Sie den in Kapitel 6, »Entwicklung transaktionaler Benutzeroberflächen«, angesprochenen SAPUI5 Visual Editor über das Kontextmenü Ihres Projekts in der SAP Web IDE, und wechseln Sie direkt in den Bearbeitungsmodus.
2. Markieren Sie im mittleren Bereich die Tabelle – sollten dabei Probleme auftreten, können Sie im linken Bereich **Outline** auch die **VBox** mit folgender **Element-ID** markieren: `Z.SalesOrderItems::sap.suite.ui.generic.template.AnalyticalListPage.view.AnalyticalListPage::ZI_AQ_SALESORDERITEM–table`
3. Scrollen Sie im rechten Bereich nach unten bis zur Option **Use Export to Excel**. Ändern Sie den Wert auf **true**, und speichern Sie Ihre Einstellung über die Tastenkombination Strg + S.
4. Starten Sie Ihre Applikation erneut, und achten Sie auf den rechten Bereich über Ihrer Tabelle. Sie sollten hier nun einen neuen Button mit der Funktion **Export to Spreadsheet** finden. Er ermöglicht Ihnen, die aktuell angezeigten Daten (abhängig von den gesetzten Filtern) direkt nach Excel zu exportieren.

Ihre Änderungen werden dabei vom SAPUI5 Visual Editor automatisch in einem neu erstellten Ordner **changes** innerhalb Ihres Projekts gespeichert (siehe Abbildung 7.12).

Dateien mit Änderungen

Abbildung 7.12 Ordner »changes« mit Änderungsdateien des SAPUI5 Visual Editors

Mit jeder Änderung, die Sie über den SAPUI5 Visual Editor durchführen, wird eine solche Datei erzeugt. Zur Laufzeit, wenn die Applikation über das SAP-Fiori-Elements-Framework neu generiert wird, werden diese Dateien ausgewertet und in den Generierungsprozess eingearbeitet. Wollen Sie eine Änderung rückgängig machen, können Sie die entsprechende Datei einfach wieder löschen.

Entwicklung der Features im Auge behalten

SAP hat in den vergangenen Jahren erkannt, dass die Verwendung von Annotationen für die Entwickler oft nicht trivial oder selbsterklärend ist. Nach der Möglichkeit, Annotationen über einen formularbasierten Editor in der SAP Web IDE zu definieren, war der SAPUI5 Visual Editor der nächste Schritt auf dem Weg, die Frontend-Entwicklung intuitiver zu gestalten.

SAP arbeitet stetig an der Weiterentwicklung dieses und anderer Werkzeuge, um eine effiziente Entwicklung auch für unerfahrenere Entwickler bestmöglich zu unterstützen.

Beispielsweise ist es mittlerweile möglich, über den SAPUI5 Visual Editor im Stil eines »What you see is what you get«-Editors die Oberfläche der Appli-

kation ohne tiefes technisches Wissen zu modellieren. Ihnen als Entwickler werden die notwendigen Änderungen in der Datei **manifest.json** und an den lokalen Annotationen direkt generiert.

Die Entwicklung geht allerdings klar weg von der SAP Web IDE und hin zum SAP Business Application Studio mit integrierten SAP Fiori Tools (siehe Details hierzu in Abschnitt 6.4), das beispielsweise eine Befehlszeilenschnittstelle bietet oder zur schnellen Direktauswahl häufiger Tätigkeiten wie das Aktivieren des Excel-Exports einer Tabelle.

SAP behält es sich an dieser Stelle allerdings selbstverständlich vor, angedachte Features der Roadmaps jederzeit zu ändern oder zu streichen. Aus diesem Grund wollen wir Ihnen nahelegen, sich stets über neue Möglichkeiten zu informieren.

7.3.7 Key Performance Indicators implementieren

Wie zu Beginn dieses Kapitels angesprochen, ist es auch möglich, globale Key Performance Indicators (KPIs) und eine Detailansicht auf diese (KPI-Karte) in die Analytical List Page zu integrieren. Da dies recht aufwendig ist, können wir Sie an dieser Stelle nicht wie sonst Schritt für Schritt durch die notwendigen Punkte führen.

KPI-Annotationen

Wir wollen Ihnen jedoch einen Überblick über die zu implementierenden Annotationen geben (siehe Abbildung 7.13).

Die Annotationen zur Implementierung von KPIs sind sehr ähnlich denen, die wir in Abschnitt 7.3.3 für die Analytical List Page vorgestellt haben. Auch hier referenzieren Sie über die zentrale Annotation `@UI.SelectionPresentationVariant` auf die Selektions- und Präsentationsvariante und bestimmen die Arten der Visualisierung. Neben Charts, die später auf der KPI-Karte angezeigt werden, können Sie einen `DataPoint` als KPI definieren. Dabei haben Sie zahlreiche Möglichkeiten, die Farbe der angezeigten Kennzahl zu beeinflussen, sodass Sie kritische Situationen hervorheben können.

KPI-Modell anlegen

Eine Besonderheit der Implementierung von KPIs ist die Notwendigkeit eines eigenen Modells innerhalb Ihres App Descriptors. Öffnen Sie zu diesem Zweck die Datei **manifest.json**, und wechseln Sie auf den **Descriptor Editor**. Navigieren Sie auf die Registerkarte **Models**, und legen Sie ein neues Modell an. Vergeben Sie einen passenden Namen (den Sie im nächsten Schritt referenzieren werden), wählen Sie als Quelle `mainService`, und speichern Sie das Modell.

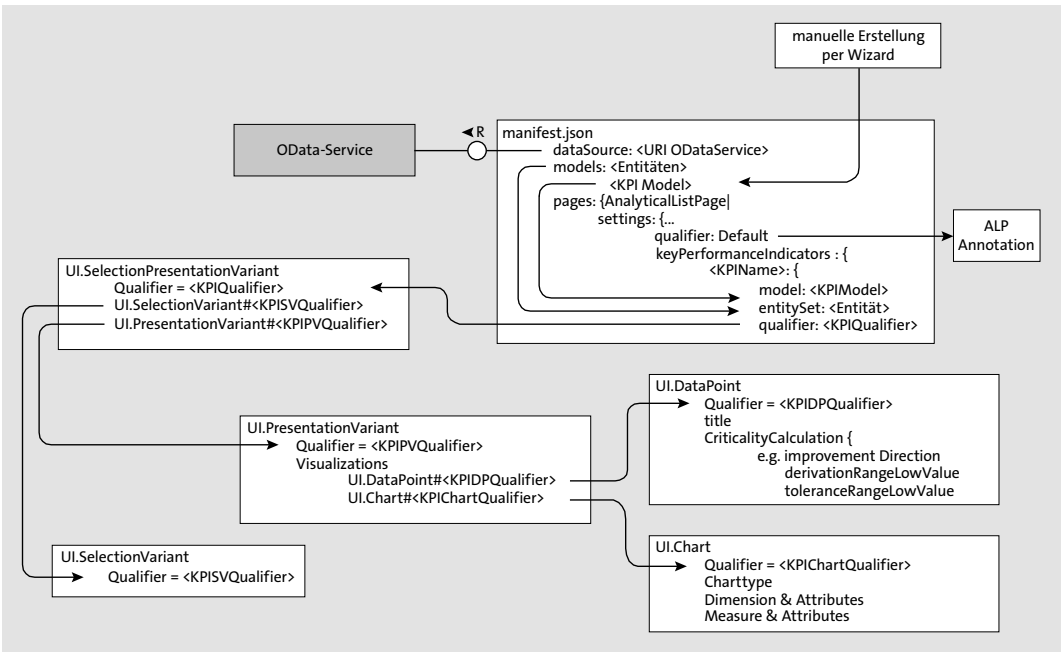


Abbildung 7.13 Annotationen zur Implementierung eines KPIs mit KPI-Karte in der Analytical List Page

KPI in die Analytical List Page integrieren

Um die zentrale Annotation `@UI.SelectionPresentationVariant` mit der Applikation zu verbinden und Ihr neu erstelltes Modell zu referenzieren, müssen Sie nun noch den Code aus Listing 7.26 innerhalb Ihrer Datei **manifest.json** hinzufügen. Im Bereich der Analytical List Page füllen Sie hierzu den Bereich für KPIs mit Ihrem Modell, der zugrunde liegenden Entität und dem Qualifier Ihrer Selektions-Präsentationsvariante.

```
"keyPerformanceIndicators": {
  "KPINetAmountBuyer": {
    "model": "KPINetAmountBuyer",
    "entitySet": "ZI_AQ_SALESORDERITEM",
    "qualifier": "KPI_SPV_NetAmount_BuyerID"
 }
}
```

Listing 7.26 Definition der KPI im App Descriptor der Analytical List Page

Ihre App sollte anschließend aussehen, wie in Abbildung 7.14 gezeigt.

Abbildung 7.14 Analytical List Page mit KPI

Bewusster Umgang mit Möglichkeiten von SAP Fiori Elements

KPIs sind für eine produktiv eingesetzte Analytical List Page nicht zwingend notwendig. Unsere Erfahrung hat gezeigt, dass Sie teilweise als störend empfunden werden, gerade, wenn Sie nicht vollumfänglich zu einem Anwendungsfall passen. Im passenden Szenario kann eine Kennzahl aber deutlichen Mehrwert bringen. Besonders dann, wenn noch eine passende KPI-Karte implementiert wurde, die eine Navigation auf weiterführende Applikationen ermöglicht.

Achten Sie daher bei der Implementierung von KPIs wie auch bei der Implementierung visueller oder kompakter Filter darauf, sie auf die konkrete Anforderung abzustimmen. Produktive Applikationen sind nicht umso besser, je mehr Möglichkeiten von SAP Fiori Elements Sie ausschöpfen. Wichtiger ist, dass Sie dem Anwender bei seiner Arbeit helfen.

Im abschließenden Abschnitt 7.4 geben wir Ihnen nun noch einen Überblick über weitere Möglichkeiten, Anwendern Ihr analytisches CDS-Datenmodell zu visualisieren.

7.4 Weitere Möglichkeiten der Visualisierung

Generierung nativer BW-Objekte

Wie in den vorherigen Abschnitten beschrieben, wird bei der Verwendung analytischer Annotationen ein transienter Provider sowie darauf basierend eine transiente Query erstellt. Diese generierten Objekte sind Standard-

objekte des SAP Business Warehouses (SAP BW). Auch als BW-Entwickler werden Sie einen Composite Provider und darauf basierend eine transiente Query anlegen. Entsprechend haben Sie über analytische Annotationen – ohne es aktiv auszusteuern – native BW-Objekte angelegt. Diese können zwar über BW-Werkzeuge nicht bearbeitet werden, doch die Funktionalitäten, die Sie über die Annotation `@AnalyticalDetails` konfigurieren können, entsprechen denen der BW-Werkzeuge.

Werkzeuge im Frontend

Dies bietet einen großen technischen wie fachlichen Mehrwert, da auf Visualisierungswerkzeuge von BW-Objekten zurückgegriffen werden kann. Das Produktportfolio für Frontend-Tools zur Datenvisualisierung von BW-Objekten ändert sich rasant. Zum Zeitpunkt der Bucherstellung (Ende 2020) werden die Produkte *SAP Analytics Cloud* und *SAP Analysis for Microsoft Office* empfohlen. Daneben bietet SAP die beiden Produkte *SAP Lumira Discovery* und *SAP Lumira Designer* zur Erstellung von Reporting Dashboards und Berichten an. Für beide Produkte stellt die SAP Analytics Cloud den logischen Nachfolger dar. Sowohl SAP Lumira Discovery als auch SAP Lumira Designer unterstützen den Zugriff auf analytische CDS Views – aufgrund der strategischen Ausrichtung von SAP auf die SAP Analytics Cloud gehen wir in diesem Buch aber nicht näher auf SAP Lumira Discovery und SAP Lumira Designer ein.

Neben den SAP-Produkten ermöglichen auch Produkte von Drittanbietern, wie beispielsweise *Power BI* von Microsoft, den Zugriff auf die Berichte der CDS Views.

Im Folgenden zeigen wir Ihnen, wie Sie den Zugriff auf analytische CDS Views in SAP Analytics Cloud und SAP Analysis realisieren.

Anpassung von Berichten im Frontend

Die meisten Frontend-Tools erlauben die direkte Anpassung des Querys. So können beispielsweise Visualisierungsoptionen im Frontend lokal geändert werden. Auch wenn dies sehr komfortabel erscheint, empfehlen wir, so viele Einstellungen wie möglich im CDS Query View im Backend durchzuführen und nur berichtspezifische Einstellungen im Frontend vorzunehmen. Jede lokal durchgeführte Einstellung im Frontend geht bei einem Umzug auf ein anderes Tool verloren.

7.4.1 SAP Analytics Cloud

Funktionalitäten SAP Analytics Cloud

SAP Analytics Cloud ist der logische Nachfolger der Produkte SAP Lumira Discovery und SAP Lumira Designer. SAP Analytics Cloud bietet ein Self-

Service-Tool, um Anwendern eine komfortable und simplifizierte Erstellung von Dashboard-Berichten zu ermöglichen (ehemalige Funktion von SAP Lumira Discovery). Zusätzlich bietet die SAP Analytics Cloud die Möglichkeit zur Erstellung von professionellen und geskripteten analytischen Applikationen (ehemalige Funktion von SAP Lumira Designer). Neben diesen beiden Kernfunktionalitäten bietet SAP Analytics Cloud eine Vielzahl von Funktionen, beispielsweise im Bereich der Dashboard-Erstellung (Digital Boardroom), Machine Learning und Predictive Analytics (Smart Insights, Smart Discovery) sowie Planung und Konnektivität.

Hilfsressourcen für SAP Analytics Cloud

SAP Analytics Cloud ist das strategische Frontend-Tool für analytische Anforderungen. Wir empfehlen Ihnen, SAP Analytics Cloud zu evaluieren – unabhängig davon, ob Sie ein BW- oder BI-Entwickler sind. Zum Zeitpunkt der Bucherstellung bietet SAP einen 90-tägigen kostenlosen Probezugriff (siehe *http://s-prs.de/v787814*). Da die umfangreiche Beschreibung aller Aktivitäten und Möglichkeiten über den Umfang dieses Buches hinausgehen würde, verweisen wir auf die sehr gute Dokumentation im SAP Help Portal (siehe *http://s-prs.de/v787815*). Gerade für Einsteiger bieten die geführten Entwicklungsvideos eine hervorragende Unterstützung (siehe *http://s-prs.de/v787816*).

Datenquellen

SAP Analytics Cloud bietet eine Vielzahl an unterschiedlichen Datenquellen für den Direktzugriff und für den Abzug von Daten. Dabei werden neben einem großen Teil Technologien der SAP-Produktlandschaft auch SAP-fremde Konnektivitäten (beispielsweise für Google Drive) angeboten. Ein Teil der möglichen Datenquellen wird in Abbildung 7.15 gezeigt.

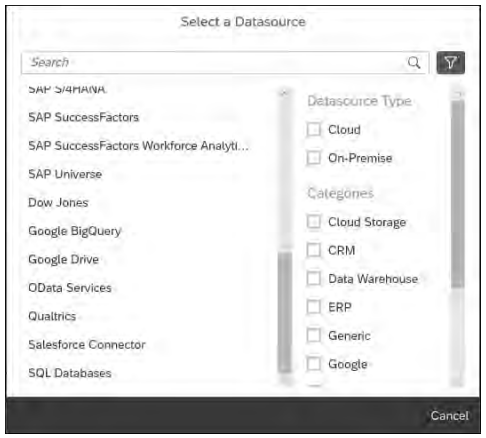


Abbildung 7.15 Auszug aus den Datenquellen von SAP Analytics Cloud

Durch die breite Konnektivität auch außerhalb des SAP-Universums ist SAP Cloud Analytics als firmenweite Lösung für analytische Anwendungen zu

verstehen. So erlaubt beispielsweise der Zugriff auf Google Drive, Microsoft OneDrive und Microsoft-SharePoint-Ordner ein simplifiziertes Live-Reporting auf häufig verteilte Excel-Dateien verschiedener Fachabteilungen – ohne hohe Kosten einer aufwendigen Systemintegration.

Online- und Batchverbindungen

SAP Analytics Cloud unterscheidet zwischen *Live-Verbindungen* und *Batchverbindungen*. Bei Live-Verbindungen werden Daten direkt im Quellsystem ausgewertet. Die Daten bleiben sicher in Ihrem Quellsystem – die Analyse- und Transformationsmöglichkeiten sind aber beschränkt. Bei Batchverbindungen werden die Daten in die SAP Analytics Cloud repliziert und können dort mit allen nativen Funktionen der SAP Analytics Cloud transformiert und analysiert werden. Im Folgenden werden Sie Live-Verbindungen nutzen, um direkt über die Analytical Engine Ihres SAP-S/4HANA-Systems auf die CDS Views zuzugreifen.

Im Folgenden wollen wir einen sehr einfachen Bericht auf unseren analytischen CDS View erstellen. Analytische CDS Views, die in SAP Analytics Cloud konsumiert werden, müssen folgende Voraussetzungen erfüllen:

1. Die Annotation `@Analytics.query` ist auf `true` gesetzt.
2. Der analytische CDS View hat den API-Status `released`.

API-Status pflegen

Während die Annotation bereits in den vorherigen Abschnitten gesetzt wurde, müssen Sie hier noch den API-Status des analytischen CDS Views pflegen. Öffnen Sie hierzu den analytischen CDS View in Eclipse, und navigieren Sie, wie in Abbildung 7.16 zu sehen, im Eigenschaftenfenster zum API-Status (engl. **API State**).

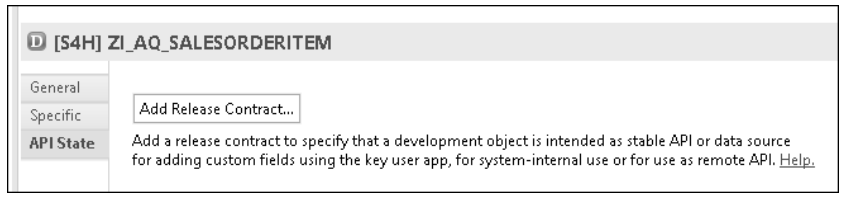


Abbildung 7.16 API-Status analytischer CDS Views

Betätigen Sie den Button **Add Release Contract**, und konfigurieren Sie im anschließenden Dialog einen systeminternen (C1-)Kontrakt im Status `Released`. Durch diese Konfiguration darf der CDS View durch die SAP Analytics Cloud verwendet werden.

Erstellung eines Modells in SAP Analytics Cloud

Zur Definition eines neuen Berichts in SAP Analytics Cloud erstellen Sie zunächst ein neues Modell, das auf die Live-Data-Connection Ihres Systems zugreift. Innerhalb der Auswahl der Datenquelle können Sie nach dem Query View (Annotation `AbapCatalog.sqlViewName`) suchen. Wie bereits in

Abschnitt 7.2.2 erklärt, wird dem technischen BW-Query-Namen das Präfix 2C vorangestellt. In Abbildung 7.17 sehen Sie den fertig ausgefüllten Dialog zur Erstellung des Modells. Beachten Sie, dass die Live-Datenverbindung zu Ihrem System vorab von einem Administrator erstellt werden muss. Sollten Sie den Browser Chrome verwenden, beachten Sie zudem SAP-Hinweis 2890576. Dieser beschreibt ein bekanntes Konnektivitätsproblem ab Chrome 80. Zur Behebung müssen recht komplexe Korrekturen im Backend durchgeführt werden. Wenn Sie diese Korrekturen nicht einspielen wollen, empfehlen wir die Nutzung von SAP Business Client mit Chromium oder Microsoft Edge.

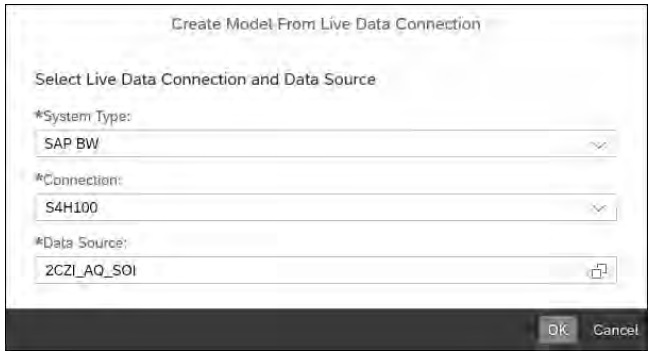


Abbildung 7.17 Erstellung eines Modells

Erstellen des Dashboards

Erstellen Sie im Anschluss eine neue Story mit Zugriff auf Ihr gerade erstelltes Modell. Designen Sie anschließend den Bericht mit SAP Analytics Cloud.

Abbildung 7.18 Umsatz-Dashboard in SAP Analytics Cloud

Das beispielhafte Ergebnis des Dashboards ist in Abbildung 7.18 dargestellt. Die Umsätze pro Land und Kategorie wurden in einer Heatmap auf der linken Seite platziert. Auf der rechten Seite wurden die Umsätze der letzten fünf Tage sowie die Umsätze der zwei erfolgreichsten Produktkategorien dargestellt.

7.4.2 SAP Analysis for Microsoft Office

SAP Analysis for Microsoft Office erlaubt es, Berichte in Microsoft Office anzuzeigen, zu konfigurieren und mit lokalen Daten anzureichern. Das Tool kann als Add-in für die MS-Office-Produkte Word, Excel und PowerPoint installiert werden. Da es Anwendern die Möglichkeit bietet, mit den Ihnen bekannten Oberflächen und Funktionalitäten zu arbeiten, wird es von Fachbereichen gern eingesetzt.

Neuen Bericht erstellen

Die Anbindung an CDS Views erfolgt wie in SAP Analytics Cloud (siehe Abschnitt 7.4.1). In Abbildung 7.19 ist dargestellt, wie Sie in SAP Analysis über den Menüpunkt **Select Data Source** nach den SQL-View-Namen eines CDS Views suchen können.

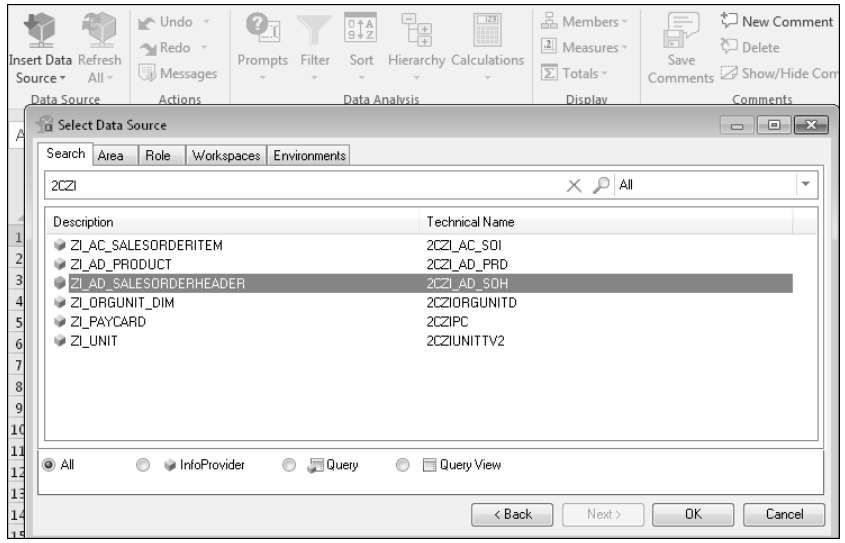
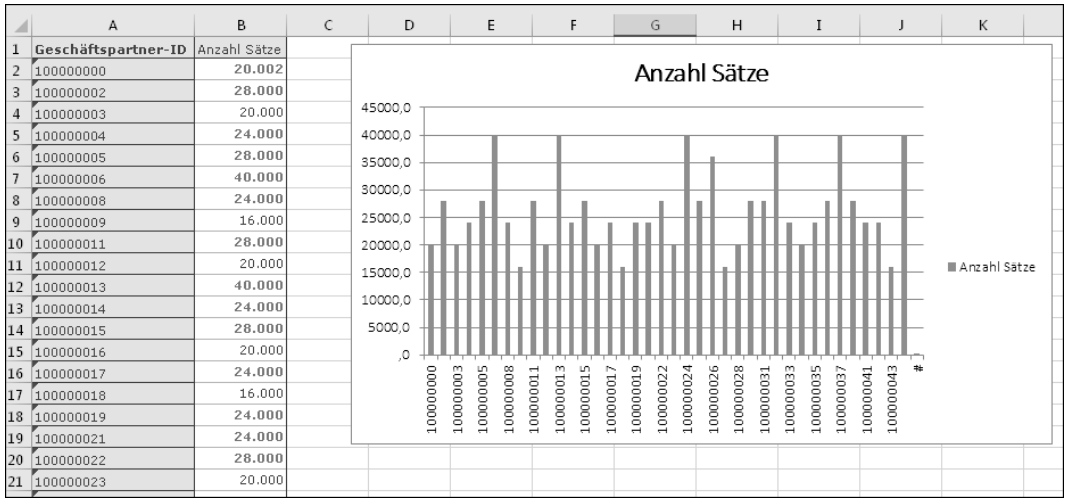


Abbildung 7.19 Datenquelle in SAP Analysis auswählen

Visualisierung konfigurieren

Anschließend können Sie über die Seitenleiste die Visualisierung der angebundenen Daten konfigurieren. Ihnen steht eine Vielzahl an Funktionalitäten zur Verfügung, z. B. Filter, dynamisches Hinzufügen und Entfernen von Aggregationsebenen, semantische Färbung von Einträgen. Abbildung 7.20

zeigt eine mögliche Darstellung als Tabelle und Chart innerhalb von Microsoft Excel.



	A	B
1	Geschäftspartner-ID	Anzahl Sätze
2	100000000	20.002
3	100000002	28.000
4	100000003	20.000
5	100000004	24.000
6	100000005	28.000
7	100000006	40.000
8	100000008	24.000
9	100000009	16.000
10	100000011	28.000
11	100000012	20.000
12	100000013	40.000
13	100000014	24.000
14	100000015	28.000
15	100000016	20.000
16	100000017	24.000
17	100000018	16.000
18	100000019	24.000
19	100000021	24.000
20	100000022	28.000
21	100000023	20.000

Abbildung 7.20 Bericht in SAP Analysis