Analytic Views: Einsatzgebiete im Data Warehouse

Dani Schnider
Trivadis AG
Zürich/Glattbrugg, Schweiz

Einleitung

Analytic Views sind eine der wesentlichen Erweiterungen in Oracle 12c Release 2. Durch zusätzliche Metadatenobjekte können nun hierarchische Zusammenhänge in Dimensionen sowie abgeleitete Kennzahlen in Faktentabellen elegant definiert werden. Doch brauchen wir diese Erweiterung? Solche Informationen werden typischerweise in BI-Werkzeugen und OLAP-Cubes definiert. Wo liegen die Vorteile von Analytic Views gegenüber Metadaten in einem BI-Tool, wann bringen sie einen Mehrwert und wie werden sie optimal eingesetzt?

Was sind Analytic Views?

Eine Analytic View ist ein Metadatenobjekt in der Datenbank, welches die dimensionalen und hierarchischen Zusammenhänge in einem Star Schema repräsentiert und für die Vereinfachung von Abfragen in BI-Applikationen verwendet werden kann. Ähnlich wie bei einer klassischen View wird die Komplexität in den Metadaten versteckt, der Anwender muss sich nicht darum kümmern. Die Definition einer Analytic View erfolgt jedoch auf eine andere Art als bei einer normalen View, wie wir später sehen werden.

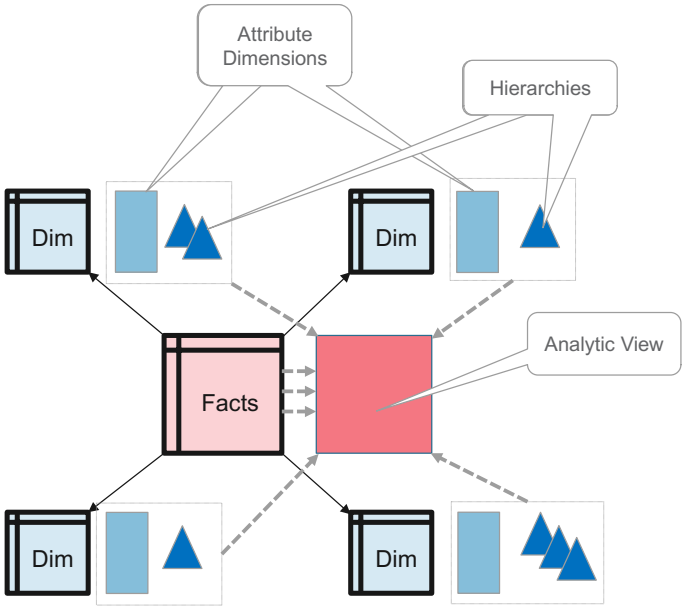

Abbildung 1: Analytic Views verwenden Attribute Dimensions und Hierarchies

Basis für eine Analytic View ist ein Star oder Snowflake Schema mit einer Fakten- und mehreren Dimensionstabellen. Für jede Dimensionstabelle wird zuerst eine *Attribute Dimension* erstellt. Darauf aufbauend können eine oder mehrere Hierarchien (*Hierarchies*) definiert werden. Diese dienen dann als Grundlage für die *Analytic View*, in welcher die Basiskennzahlen sowie abgeleitete Kennzahlen über mehrere Hierarchiestufen der einzelnen Dimensionen definiert werden können.

Für die Abfragen auf das Star Schema werden dann nicht mehr Joins zwischen Dimensionen und Fakten geschrieben, sondern es werden nur noch Abfragen auf die Analytic View gemacht. Die notwendigen Joins und Aggregationen werden von Oracle im Hintergrund ausgeführt.

Dimensionen und Hierarchien

Dimensionen enthalten üblicherweise eine oder mehrere Hierarchien, die für Drill-up und Drill-down in OLAP-Tools verwendet werden können. Diese können zum Beispiel mit der ADAPT-Notation dargestellt werden.

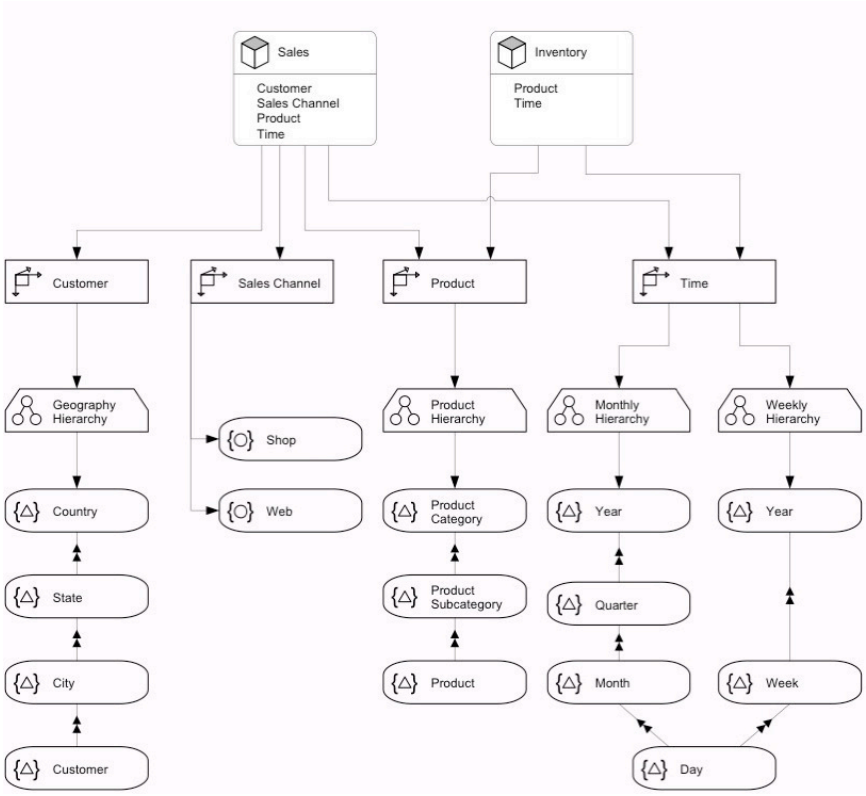

Abbildung 2: Hierarchien in Dimensionen, dargestellt als ADAPT-Diagramm[1]

Wie Abbildung 2 zeigt, gibt es für die Dimension *Product* eine Hierarchie mit den Hierarchiestufen Produkt, Subkategorie und Kategorie. Für jede Hierarchiestufe gibt es mehrere Attribute in der Dimensionstabelle. Doch welches Attribut gehört zu welcher Hierarchiestufe? Diese Zusammenhänge müssen wir kennen, um sinnvolle Abfragen auf die Dimension ausführen zu können.

[1] ADAPT (Application Design for Analytical Processing Technologies), siehe www.symcorp.com

Um Oracle die Hierarchien einer Dimension mitzuteilen, gibt es zwei Arten von Metadatenobjekten für Dimensionen: *Attribute Dimensions* und *Hierarchies*. In der *Attribute Dimension* werden alle fachlich relevanten Attribute der Dimensionstabelle aufgeführt und mit zusätzlichen Informationen ergänzt. Dazu gehören Kurz- und Langbezeichnungen, Zuordnung zu einer Hierarchiestufe sowie Sortierreihenfolge innerhalb einer Hierarchiestufe. Listing 1 zeigt, wie die Produktdimension im *Sales History* (SH) Schema von Oracle definiert werden kann.[2]

```
CREATE OR REPLACE ATTRIBUTE DIMENSION sh_products_attr_dim
USING sh.products
ATTRIBUTES (
  prod_id
    CLASSIFICATION caption VALUE 'Product'
    CLASSIFICATION description VALUE 'Product',
  prod_name
    CLASSIFICATION caption VALUE 'Product'
    CLASSIFICATION description VALUE 'Product',
  prod_subcategory
    CLASSIFICATION caption VALUE 'Subcategory'
    CLASSIFICATION description VALUE 'Subcategory',
  prod_category
    CLASSIFICATION caption VALUE 'Category'
    CLASSIFICATION description VALUE 'Category'
    )
  LEVEL PRODUCT
    CLASSIFICATION caption VALUE 'Product'
    CLASSIFICATION description VALUE 'Product'
    KEY prod_id
    MEMBER NAME prod_name
    MEMBER CAPTION prod_name
    MEMBER DESCRIPTION prod_name
    ORDER BY prod_name
    DETERMINES (prod_subcategory)
  LEVEL SUBCATEGORY
    CLASSIFICATION caption VALUE 'Subcategory'
    CLASSIFICATION description VALUE 'Subcategory'
    KEY prod_subcategory
    MEMBER NAME prod_subcategory
    MEMBER CAPTION prod_subcategory
    MEMBER DESCRIPTION prod_subcategory
    ORDER BY prod_subcategory
    DETERMINES (prod_category)
  LEVEL CATEGORY
    CLASSIFICATION caption VALUE 'Category'
    CLASSIFICATION description VALUE 'Category'
    KEY prod_category
    MEMBER NAME prod_category
    MEMBER CAPTION prod_category
    MEMBER DESCRIPTION prod_category
    ORDER BY prod_category
  ALL MEMBER NAME 'ALL PRODUCTS';
```

Listing 1: Definition der Attribute Dimension für PRODUCTS

[2] Die Beispiele stammen aus dem Oracle Live SQL Tutorial *Create attribute dimensions, hierarchies and analytic views for SH schema*

Als nächstes muss nun eine Hierarchie definiert werden. Dazu wird ein *Hierarchy*-Objekt erstellt. Dieser Schritt muss separat ausgeführt werden, weil es möglich ist, dass für eine *Attribute Dimension* mehrere Hierarchien erstellt werden können (siehe Dimension *Time* in Abbildung 2). Für unser Beispiel der Produktdimension wird eine Hierarchie festgelegt, wie in Listing 2 dargestellt. Als Basis dient die zuvor erstellte *Attribute Dimension*.

```
CREATE OR REPLACE HIERARCHY sh_products_hier
  CLASSIFICATION caption VALUE 'Products'
  CLASSIFICATION description VALUE 'Products'
USING sh_products_attr_dim
 (product CHILD OF
  subcategory CHILD OF
  category);
```

Listing 2: Definition eine Hierarchie

Auf die gleiche Weise werden auch die anderen Dimensionen des Star Schemas mit Metadaten angereichert. Für jede Dimensionstabelle wird eine *Attribute Dimension* sowie mindestens eine *Hierarchy* definiert. Für Dimensionen, die aus mehreren Tabellen bestehen (Snowflake Schema) muss zuerst eine View erstellt werden, die dann als Grundlage für die *Attribute Dimension* verwendet werden kann.

Erstellen einer Analytic View

Nach diesen Vorbereitungen sind wir nun in der Lage, unsere erste einfache *Analytic View* zu erstellen. Auf der Faktentabelle *SALES* im *Sales History* Schema wird eine Analytic View mit fünf Dimensionen erstellt. Die Dimension *Times* hat zwei Hierarchien, alle anderen Dimensionen je eine. Als Kennzahlen (*Measures*) werden die Basiskennzahlen *AMOUNT_SOLD* und *QUANTITY_SOLD* definiert.

```
CREATE OR REPLACE ANALYTIC VIEW sh_sales_history_av
USING sh.sales
DIMENSION BY
 (
  sh_times_attr_dim KEY time_id REFERENCES time_id
    HIERARCHIES (sh_times_calendar_hier DEFAULT,
                 sh_times_fiscal_hier),
  sh_products_attr_dim KEY prod_id REFERENCES prod_id
    HIERARCHIES (sh_products_hier DEFAULT),
  sh_customers_attr_dim KEY cust_id REFERENCES cust_id
    HIERARCHIES (sh_customers_hier DEFAULT),
  sh_channels_attr_dim KEY channel_id REFERENCES channel_id
    HIERARCHIES (sh_channels_hier DEFAULT),
  sh_promotions_attr_dim KEY promo_id REFERENCES promo_id
    HIERARCHIES (sh_promotions_hier DEFAULT)
  )
MEASURES (
  amount_sold FACT amount_sold,
  quantity_sold FACT quantity_sold
);
```

Listing 3: Definition einer einfachen Analytic View

Eine Analytic View bezieht sich immer auf genau eine Faktentabelle, die mittels USING spezifiziert wird. Nach DIMENSION BY wird eine Reihe von Dimensionen angegeben, jeweils identifiziert durch die entsprechende Attribute Dimension, nicht die Dimensionstabelle. Pro Hierarchie können eine oder mehrere Hierarchien angegeben werden. Über KEY und REFERENCES werden die Fakten mit den Dimensionen verbunden.

Auch hier könnten wieder mittels *CLASSIFICATION* Kurz- und Langbezeichnungen sowie weitere Angaben für Formatierung, Aggregationsregeln, etc. definiert werden. Um das Beispiel in Listing 3 möglichst übersichtlich zu halten, wurden diese Angaben hier weggelassen.

Die wahre Stärke der Analytic Views zeigt sich aber erst bei der Definition von berechneten Kennzahlen (*Calculated Measures*). So können Berechnungen über die Basiskennzahlen direkt in der Analytic View definiert und nachher als zusätzliche Kennzahlen angewendet werden. Beispiele sind Year-to-Date-Berechnungen, Vorjahresvergleiche, prozentuale Anteile und alle Arten von analytischen Funktionen. Die zuvor erstellte Analytic View wird in Listing 4 um drei berechnete Kennzahlen erweitert:

- AMT_SOLD_SHR_PARENT_PROD: Prozentualer Anteil des Umsatzes in Bezug auf die übergeordnete Hierarchiestufe. Beispiel: Wieviel Umsatz wird mit einem Produkt im Verhältnis zum gesamten Umsatz der Produktsubkategorie gemacht?
- SALES_CAL_YTD: Year-to-Date-Umsatz, d.h. kumulierter Umsatz seit Anfang des Kalenderjahres.
- SALES_CAL_YEAR_AGO: Umsatz in der entsprechenden Zeitperiode (Tag, Monat, Quartal, Jahr) des Vorjahres.

```
CREATE OR REPLACE ANALYTIC VIEW sh_sales_history_av
USING sh.sales
DIMENSION BY
 (...)
MEASURES (
-- Base Measures
  amount_sold FACT amount_sold,
  quantity_sold FACT quantity_sold,
-- Calculated Measures
  amt_sold_shr_parent_prod AS
    (SHARE_OF(amount_sold HIERARCHY sh_products_hier PARENT)),
  sales_cal_ytd AS
    (SUM(amount_sold) OVER (HIERARCHY sh_times_calendar_hier
                      BETWEEN UNBOUNDED PRECEDING
                      AND CURRENT MEMBER
                      WITHIN ANCESTOR AT LEVEL calendar_year)),
  sales_cal_year_ago AS
    (LAG(amount_sold) OVER (HIERARCHY sh_times_calendar_hier
                      OFFSET 1 ACROSS ANCESTOR
                      AT LEVEL calendar_year))
)
DEFAULT MEASURE amount_sold;
```

Listing 4: Analytic View mit Calculated Measures

Dieses relativ einfache Beispiel zeigt die zahlreichen Möglichkeiten von analytischen Berechnungen nur ansatzweise, gibt aber bereits einen Hinweis auf die Mächtigkeit der Funktionalität.

Abfragen auf Star Schema

Wie können nun die erstellten Datenbankobjekte für Abfragen verwendet werden? Die *Attribute Dimensions* gar nicht – sie dienen nur als Grundlage für die *Hierarchies* und *Analytic Views*. Eine *Hierarchy* kann jedoch direkt selektiert werden. Die SQL-Abfrage in Listing 5 stellt sämtliche Subkategorien und Produkte der Produktkategorie „Electronics“ als hierarchischen Baum dar.

```
SELECT rpad(' ', depth * 2)||member_description AS PROD_HIER
  FROM sh_products_hier
 WHERE prod_category = 'Electronics'
ORDER BY hier_order;
```

Listing 5: Abfrage auf Dimensionshierarchie

In Listing 5 wird eine SQL-Abfrage auf die *Analytic View* ausgeführt. Und zwar werden die Kennzahlen AMOUNT_SOLD und QUANTITY_SOLD über die Kalender- und Produkthierarchie aggregiert. Für die Kalenderhierarchie sollen nur die Hierarchiestufen *Jahr* und *Quartal* angezeigt werden, für die Produkthierarchie alle Stufen, jedoch nur für die Kategorie „Electronics“.

```
SELECT sh_times_calendar_hier.member_name AS calendar_hier
     , sh_products_hier.member_name AS products_hier
     , amount_sold
     , quantity_sold
  FROM sh_sales_history_av
       HIERARCHIES (sh_times_calendar_hier, sh_products_hier)
 WHERE sh_times_calendar_hier.level_name
       IN ('CALENDAR_YEAR', 'CALENDAR_QUARTER')
   AND sh_products_hier.prod_category = 'Electronics'
ORDER BY
       sh_times_calendar_hier.hier_order
     , sh_products_hier.hier_order;
```

Listing 6: Abfrage auf Analytic View

Die Abfrage enthält weder Joins zwischen Fakten und Dimensionen noch Aggregationen von Kennzahlen. Auch ein GROUP BY ist nicht notwendig. All diese Operationen werden automatisch innerhalb der Analytic View ausgeführt. Anhand der Hierarchiedefinitionen und der verwendeten Dimensionen weiß Oracle, welche Dimensionstabellen gejoined und nach welchen Attributen das Resultat gruppiert werden muss. Als Aggregationsfunktion wird hier SUM verwendet, da es sich um additive Kennzahlen handelt. Dies ist der Default, wenn nichts Anderes spezifiziert wird. Beim Erstellen der Analytic View könnte aber auch eine andere Aggregationsfunktion definiert werden.

Das Beispiel in Listing 6 bezieht sich nur auf die Basiskennzahlen, aber auf die gleiche Weise könnten auch berechnete Kennzahlen verwendet werden. Für den Anwender ändert sich nichts – die Komplexität ist in der Analytic View versteckt.

Natürlich könnten all diese Auswertungen auch mit SQL-Abfragen auf die Tabellen des Star Schemas implementiert werden. Mit Hilfe von Analytischen Funktionen, Subqueries, Self-Joins, etc. können die gleichen Resultate ermittelt werden, jedoch mit deutlich komplexeren SQL-Befehlen. Ziel der Analytic Views ist es, die Komplexität von Abfragen im Star Schema zu verringern.

Einsatzgebiete im Data Warehouse

Abfragen auf Analytic Views sind vergleichbar mit den Möglichkeiten von BI-Werkzeugen, die mittels multidimensionalen Sprachen (z.B. MDX) auf OLAP-Cubes zugreifen. Bei der relationalen Implementierung von dimensionalen Datenmodellen (sprich: in einem Star Schema) waren die Abfragen bisher eher schwerfällig. Im Praxiseinsatz ist dies üblicherweise kein Nachteil, da die SQL-Abfragen von den BI-Werkzeugen generiert werden. Doch auch dazu sind entsprechende Metadaten über die Dimensionen, Hierarchien und Kennzahlen notwendig. Diese Informationen werden normalerweise in den BI-Tools definiert.

Wer mit Werkzeugen wie OBIEE, SAP BusinessObjects oder IBM Cognos arbeitet, wird nicht viel Neues an den Möglichkeiten von Analytic Views finden. In all diesen Tools können Dimensions-hierarchien, Drill-Pfade, Aggregationsfunktionen und berechnete Kennzahlen definiert werden. Ob diese Informationen im *Business Model and Mapping Layer* von OBIEE, im *Universe* von BusinessObjects oder im *MetaManager* von Cognos definiert werden, macht keinen großen Unterschied. Warum sollen nun die gleichen Zusammenhänge auch noch in der Datenbank definiert werden? Dies würde höchstens dann attraktiv, wenn die Metadaten nachträglich ins BI-Tool importiert werden können. Solange dies nicht möglich ist, werden Anwender der erwähnten BI-Produkte kaum mit Analytic Views arbeiten.

Interessant werden Analytic Views jedoch, wenn unterschiedliche BI-Werkzeuge zum Einsatz kommen. Anstatt die Metadaten in jedem Tool separat zu erfassen, ist es einfacher, dies einmal auf der Datenbank zu tun. Auch für Adhoc-Abfragen mit Tools ohne spezifische Metadatenschicht bieten Analytic Views komfortable Möglichkeiten, um Abfragen in der dimensionalen Welt auszuführen. Damit wird es beispielsweise möglich, einfache BI-Applikationen mit APEX zu entwickeln oder Adhoc-Abfragen mit SQL Developer durchzuführen. Je mehr Logik in der Datenbank definiert ist, desto weniger muss in den Tools nochmals definiert werden.

Werden Analytic Views die umfangreichen und komplexen BI-Werkzeuge überflüssig machen? Wohl kaum, aber mit den neuen Möglichkeiten in Oracle 12c Release 2 können bisher schwerfällige SQL-Abfragen stark vereinfacht werden. Die Komplexität ist nach wie vor vorhanden, aber sie ist in den Analytic Views versteckt und für die Anwender unsichtbar.

Kontaktadresse:

Dani Schnider
Trivadis AG
Sägereistrasse 29
CH-8152 Glattbrugg

Telefon: +41 58 459 50 81
Fax: +41 58 459 56 95
E-Mail dani.schnider@trivadis.com
Internet: www.trivadis.com