Berechnung von Kennzahlen mit der SQL Model Clause

Thomas Mauch

12.07.2018 DOAG

AGENDA

1. **Einführung**
2. SQL Model Clause Syntax
3. Performance
4. Architektur
5. Framework
6. Zusammenfassung

Einführung

Berechnung von Kennzahlen im DWH

- Anforderungen an Berechnungen:
  - Umsetzung von verschachtelten Berechnungen
  - nachvollziehbar
  - Von Power Benutzer anpassbar

- Einfache Syntax, zentral in DB abgelegt

- Umsetzung mit SQL Model Clause

AGENDA

1. Einführung
2. **SQL Model Clause Syntax**
3. Performance
4. Architektur
5. Framework
6. Zusammenfassung

SQL Model Clause

Beispieldaten: SH – Schema von Oracle

- Daten aus View:

Dimensionen **Measure Dimension** **Measure**

COUNTRY	PRODUCT	YEAR	MEASURE_CODE	MEAS_VALUE
Italy	Bounce	1999	cnt_sold	112
Italy	Bounce	2000	cnt_sold	202
Italy	Bounce	2001	cnt_sold	237
Italy	Mouse Pad	1998	cnt_sold	283
Italy	Mouse Pad	1999	cnt_sold	437
Italy	Mouse Pad	2000	cnt_sold	351
Italy	Mouse Pad	2001	cnt_sold	470
Japan	Bounce	1999	cnt_sold	135
Japan	Bounce	2000	cnt_sold	239
Japan	Bounce	2001	cnt_sold	309
Japan	Mouse Pad	1998	cnt_sold	287
Japan	Mouse Pad	1999	cnt_sold	510
Japan	Mouse Pad	2000	cnt_sold	412
Japan	Mouse Pad	2001	cnt_sold	507

SQL Model Clause Syntax

Berechnung Wert für 2002:

```
SELECT country, product, year, measure_code, meas_value
  FROM sh.v_sales_view_measured
 MODEL
 DIMENSION BY ( country, product, year, measure_code )
 MEASURES ( meas_value AS meas_value )
 RULES
( meas_value['Italy', 'Bounce', 2002, 'cnt_sold'] =
   meas_value['Italy','Bounce',2001,'cnt_sold']
   + meas_value['Italy','Bounce', 2000, 'cnt_sold']  );
```

COUNTRY	PRODUCT	YEAR	MEASURE_CODE	MEAS_VALUE
Italy	Bounce	1999	cnt_sold	112
Italy	Bounce	2000	cnt_sold	202
Italy	Bounce	2001	cnt_sold	237
Italy	Bounce	2002	cnt_sold	439
Japan	Bounce	1999	cnt_sold	135
Japan	Bounce	2000	cnt_sold	239
Japan	Bounce	2001	cnt_sold	309

.....

SQL Model Clause Syntax

PARTITION BY: Berechnung Wert für 2002 und alle Länder

```
SELECT country, product, year, measure_code, meas_value
  FROM sh.v_sales_view_measured
 MODEL
 PARTITION BY ( country )
 DIMENSION BY ( product, year, measure_code )
 MEASURES ( meas_value AS meas_value )
 RULES
(meas_value['Bounce',2002,'cnt_sold'] =
 meas_value['Bounce',2001,'cnt_sold'] + meas_value['Bounce',2000,'cnt_sold']);
```

COUNTRY	PRODUCT	YEAR	MEASURE_CODE	MEAS_VALUE
Italy	Bounce	1999	cnt_sold	112
Italy	Bounce	2000	cnt_sold	202
Italy	Bounce	2001	cnt_sold	237
Italy	Bounce	2002	cnt_sold	439
Japan	Bounce	1999	cnt_sold	135
Japan	Bounce	2000	cnt_sold	239
Japan	Bounce	2001	cnt_sold	309
Japan	Bounce	2002	cnt_sold	548

.....

SQL Model Clause Syntax

Berechnung neue Measure mit CV()

```
SELECT country, product, year, measure_code, meas_value
  FROM sh.v_sales_view_measured
 MODEL
 PARTITION BY ( country )
 DIMENSION BY ( product, year, measure_code )
 MEASURES ( meas_value AS meas_value )
 RULES
 ( meas_value['Bounce',1999,'cnt_sld2']
 = meas_value['Bounce', CV(),'cnt_sold'] + 50  );

COUNTRY PRODUCT         YEAR    MEASURE_CODE    MEAS_VALUE
-------  -----------    -------  -----------     ---------------
Italy   Bounce          1999    cnt_sold        112
Italy   Bounce          1999    cnt_sld2        162
Japan   Mouse Pad       2000    cnt_sold        412
Japan   Mouse Pad       2001    cnt_sold        507
……
```

SQL Model Clause Syntax

Berechnung: Multi Cell Reference linke Seite, UPSERT ALL

```
SELECT country, product, year, measure_code, meas_value
  FROM sh.v_sales_view_measured
 MODEL
 PARTITION BY ( country )
 DIMENSION BY ( product, year, measure_code )
 MEASURES ( meas_value AS meas_value )
 RULES UPSERT ALL
 ( meas_value['Bounce', ANY ,'cnt_sld2']
 = meas_value['Bounce', CV(),'cnt_sold'] + 50  );
```

COUNTRY	PRODUCT	YEAR	MEASURE_CODE	MEAS_VALUE
Italy	Bounce	1998	cnt_sld2	
Italy	Bounce	1999	cnt_sold	112
Italy	Bounce	1999	cnt_sld2	162
Italy	Bounce	2000	cnt_sld2	252
Italy	Bounce	2000	cnt_sold	202
Italy	Bounce	2001	cnt_sold	237
Italy	Bounce	2001	cnt_sld2	287

.....

SQL Model Clause Syntax

Modus Datenänderung:

- UPDATE: Vorhandene Zelle wird überschrieben
- UPSERT: (Default) Vorhandene Zelle wird überschrieben, neue Zelle wird erzeugt. Nicht bei symbolischer Referenz !
- UPSERT ALL: Neue Zelle auch mit symbolischer Referenz.

- Positionale Referenz:
  - meas_value['Bounce', 2005] =meas_value…
  - FOR LOOP
- Symbolische Referenz:
  - meas_value[product = 'Bounce', YEAR = 2005] =meas_value…
  - >, >=, IN, ANY, BETWEEN

SQL Model Clause Syntax

Berechnung: Multi Cell Reference auf rechter Seite + Gruppenfunktion

```
SELECT country, product, year, measure_code, meas_value
  FROM sh.v_sales_view_measured
 MODEL
 PARTITION BY ( country )
 DIMENSION BY ( product, year, CAST( measure_code AS VARCHAR2(200)) AS
measure_code )
 MEASURES ( meas_value AS meas_value )
 RULES UPSERT ALL
 (meas_value['Bounce',2007,'cnt_sold_avg'] =
   AVG(meas_value)['Bounce',year BETWEEN 2000 and 2001 ,'cnt_sold']  )
```

COUNTRY	PRODUCT	YEAR	MEASURE_CODE	MEAS_VALUE
Italy	Bounce	2000	cnt_sold	202
Italy	Bounce	2001	cnt_sold	237
Italy	Bounce	2007	cnt_sold_avg	219,5
Japan	Bounce	2000	cnt_sold	239
Japan	Bounce	2001	cnt_sold	309
Japan	Bounce	2007	cnt_sold_avg	274

.....

SQL Model Clause Syntax

Berechnung: Multi Cell Reference für alle Jahre und Produkte

```
SELECT country, product, year, measure_code, meas_value
  FROM sh.v_sales_view_measured
 MODEL
 PARTITION BY ( country )
 DIMENSION BY ( product, year, CAST( measure_code AS VARCHAR2(200)) AS
measure_code )
 MEASURES ( meas_value AS meas_value )
 RULES UPSERT ALL
 (meas_value[ ANY ,  ANY , 'cnt_sold_avg'] =
    AVG(meas_value)[ CV(), year BETWEEN CV() -1 and CV() , 'cnt_sold'] )
```

COUNTRY	PRODUCT	YEAR	MEASURE_CODE	MEAS_VALUE
Italy	Bounce	1999	cnt_sold	112
Italy	Bounce	1999	cnt_sold_avg	112
Italy	Bounce	2000	cnt_sold	202
Italy	Bounce	2000	cnt_sold_avg	157
Italy	Bounce	2001	cnt_sold	237
Italy	Bounce	2001	cnt_sold_avg	219,5
Italy	Mouse Pad	1998	cnt_sold	283
Italy	Mouse Pad	1998	cnt_sold_avg	283

.....

SQL Model Clause Syntax

Analytische Funktionen in MEASURE und DIMENSION Clause

```
SELECT  country, product, year, year2, measure_code, meas_value, sum_pro_prd
FROM v_sales_view_measured
WHERE year BETWEEN 1998 AND 2000
MODEL
PARTITION BY ( country )
DIMENSION BY ( product, year,
    MIN(year) OVER ( PARTITION BY product) AS year2,  measure_code )
 MEASURES ( meas_value meas_value,
       SUM(meas_value) OVER(PARTITION BY product) sum_pro_prd )
     RULES   (  )
```

COUNTRY	PRODUCT	YEAR	YEAR2	MEASURE_CODE	MEAS_VALUE	SUM_PRO_PRD
Italy	Bounce	1999	1999	cnt_sold	112	688
Italy	Bounce	2000	1999	cnt_sold	202	688
Japan	Bounce	1999	1999	cnt_sold	135	688
Japan	Bounce	2000	1999	cnt_sold	239	688
Italy	Mouse Pad	1998	1998	cnt_sold	283	2280
Italy	Mouse Pad	1999	1998	cnt_sold	437	2280
Italy	Mouse Pad	2000	1998	cnt_sold	351	2280
Japan	Mouse Pad	1998	1998	cnt_sold	287	2280

.....

SQL Model Clause Syntax

Analytische Funktionen in Berechnung von neuen Measures

```
SELECT  country, product , year, measure_code , meas_value, sum_pro_prd
FROM v_sales_view_measured
WHERE year BETWEEN 1998 AND 2000
MODEL
PARTITION BY ( country )
DIMENSION BY ( product, year, measure_code )
  MEASURES ( meas_value meas_value,
                 SUM(meas_value) OVER(PARTITION BY product) sum_pro_prd )
     RULES  UPSERT ALL
( meas_value[ 'Bounce',ANY,'ratio'] =
ROUND(meas_value[cv(),cv(),'cnt_sold'] / sum_pro_prd [cv(),cv(),'cnt_sold'], 2)
                              )
```

COUNTRY	PRODUCT	YEAR	MEASURE_CODE	MEAS_VALUE	SUM_PRO_PRD
Italy	Bounce	1998	ratio		
Italy	Bounce	1999	cnt_sold	112	688
Italy	Bounce	1999	ratio	0,16	
Italy	Bounce	2000	cnt_sold	202	688
Italy	Bounce	2000	ratio	0,29	
Japan	Bounce	1998	ratio		
Japan	Bounce	1999	cnt_sold	135	688
Japan	Bounce	1999	ratio	0,2	

SQL Model Clause Syntax

Verwendung von berechneter Measure in weiterer Berechnung

```
SELECT  country, product , year, measure_code , meas_value, sum_pro_prd
FROM v_sales_view_measured
WHERE year BETWEEN 1998 AND 2000
MODEL
PARTITION BY ( country )
DIMENSION BY ( product, year, measure_code )
MEASURES ( meas_value meas_value,
              SUM(meas_value) OVER(PARTITION BY product) sum_pro_prd )
     RULES  UPSERT ALL
 meas_value['Bounce',ANY,'ratio'] = ROUND(meas_value[cv(),cv(),'cnt_sold' ] /
       sum_pro_prd [ cv() , cv() , 'cnt_sold' ], 2),
 meas_value[ 'Bounce',ANY,'sum_rat'] = SUM(meas_value)[cv(),
       year BETWEEN cv() - 1 AND cv() , 'ratio' ] )
```

COUNTRY	PRODUCT	YEAR	MEASURE_CODE	MEAS_VALUE	SUM_PRO_PRD
Italy	Bounce	1999	cnt_sold	112	688
Italy	Bounce	1999	ratio	0,16	
Italy	Bounce	1999	sum_rat	0,16	
Italy	Bounce	2000	cnt_sold	202	688
Italy	Bounce	2000	ratio	0,29	
Italy	Bounce	2000	sum_rat	0,45	

SQL Model Clause Syntax

Optionen

- RULES_ORDER
  - SEQUENTIAL : Regeln werden in Reihenfolge der Definition abgearbeitet
  - AUTOMATIC: Reihenfolge wird aus Regeln abgeleitet
- IGNORE_NAV
  - Default für NULL und fehlende Zellen:
  - 0 / 01.01.2001 / Leerer String für NUMBER / DATE / VARCHAR Datentyp
- RETURNING
  - ALL ROWS : Alle Zellen werden zurückgegeben
  - UPDATED ROWS: nur berechnete Zellen werden zurückgegeben

SQL Model Clause Syntax

Weiter Möglichkeiten:

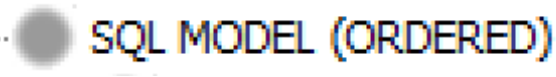
- FOR LOOP
  - Wiederholen von Rules für Werte, auch mit Subqueries
  - Bis max. 10.000 Durchläufe
- ITERATE
  - Iteratives Durchlaufen von Rechenregeln
- Reference Model
  - Verwendung von read-only Reference Model in dem Main SQL Model

AGENDA

1. Einführung
2. SQL Model Clause Syntax
3. **Performance**
4. Architektur
5. Framework
6. Zusammenfassung

SQL Model Clause: Performance

Berechnungen sind schnell

- Daten werden einmal gelesen
  - Operator im Execution Plan: 

- Parallelisierung
  - Partitionen (PARTITON BY) können parallel verarbeitet werden
- Aber: Bedarf an Session Memory (PGA + UGA)
  - Abhängig vom Platzbedarf der berechneten Werte

SQL Model Clause: Performance

Abfrage auf DB mit 96 GB Memory Target:

- Abhängigkeit von Anzahl berechneter Werte
- Datentyp berechneter Wert: NUMBER

Anz. Formel	0 (ohne Mod. C.)	0 (mit Mod. C.)	1	50	100	150	200	200 (Par.2)	200 (Par.6)
Anz. Rows (Mio)	1,87	1,87	1,89	3,02	4,18	5,32	6,47	6,47	6,47
Ausf. Zeit (sec)	0,6	2,9	3,4	7,6	11,2	14,8	18,0	3,00	2,10
PGA + UGA max. (MB)	0,2	295	396	406	426	443	450	1,4	0,2

SQL Model Clause: Performance

Abfrage auf DB mit 96 GB Memory Target:

- Abhängigkeit von deklarierter Länge berechneter Werte
- Datentyp berechneter Wert: VARCHAR2
  - 200 Formeln / 6,47 Mio Rows

Deklarierte Länge berechneter Wert	100	500	1000	2000	4000
Ausf. Zeit (sec)	18,5	28,5	39,5	85	132
PGA + UGA max. (MB)	759	2.302	2.352	2.381	2.399

- Wenn Session Memory Grenze erreicht wird , nimmt Ausführungszeit stark zu

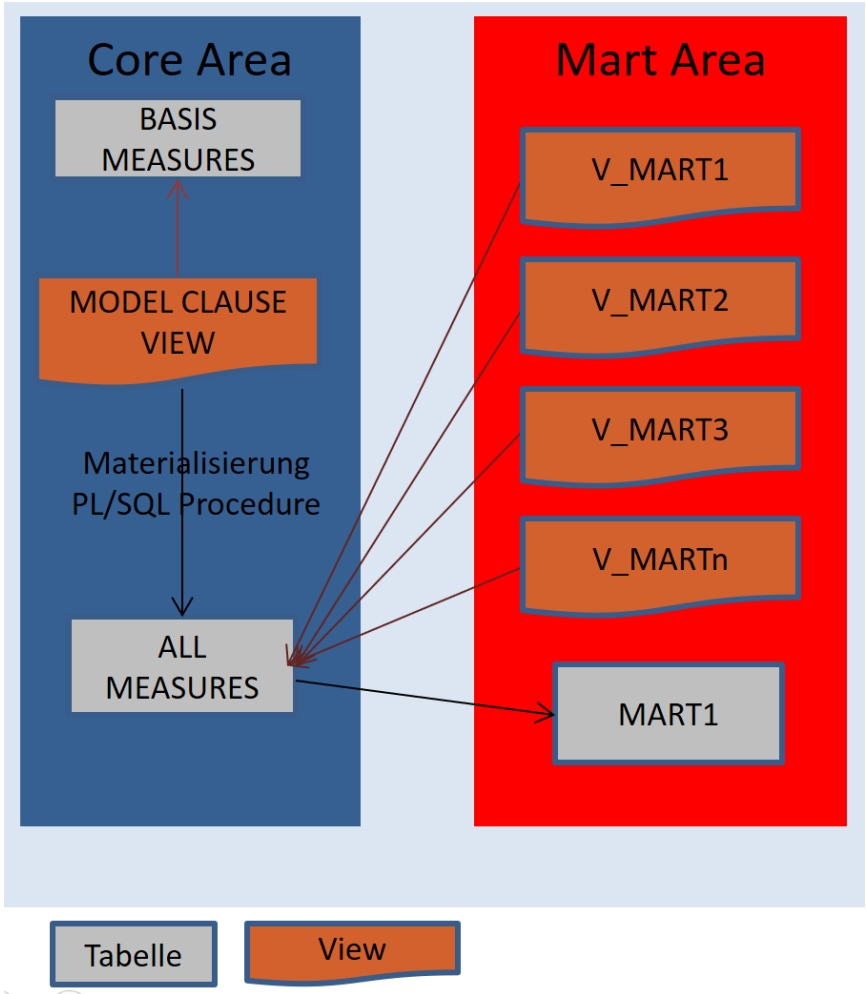
AGENDA

1. Einführung
2. SQL Model Clause Syntax
3. Performance
4. **Architektur**
5. Framework
6. Zusammenfassung

Architektur

View mit SQL Model Clause liefert berechnete Measures

- Materialisierung im ETL-Prozess
  - Performance
  - Evtl. ergänzende Berechnungen in PL/SQL Procedure
  - Anpassungen in Model Clause View wirken sich nicht auf vorhandene Daten aus



AGENDA

1. Einführung
2. SQL Model Clause Syntax
3. Performance
4. Architektur
5. **Framework**
6. Zusammenfassung

Framework

PL/SQL Framework mit APEX Oberfläche

- Repository in Datenbank
  - Versionierte Definition der SQL Model Views
  - Protokollierung der Anpassungen
- APEX Oberfläche
  - Unterstützung bei der Definition der Views
  - Zugriff für Power User
- PL/SQL Package
  - Syntax Check der View Definition
  - Anlegen und Anpassen der Views

Framework

Model Clause View Definition über APEX Oberfläche

- Eingabe Viewname / Basistabelle / Schema
- Auswahl von Optionen
- Auswahl Spalten und Zuordnung von Funktion (DIMENSION / MEASURE / PARTITION)
- Eingabe der Formeln und Syntaxcheck
- Anlegen der View
  - Über Button in Maske
  - Automatisiert: Datum ‚gültig von' definiert Zeitpunkt, an dem die View angelegt wird

Framework

Model Clause View Anpassung über APEX Oberfläche (Versionierung)

- Vorhandene Definition wird über Button kopiert
- Neue Version kann angepasst und validiert werden
- Produktivsetzung der Version
  - Über Button in Maske
  - Automatisiert

AGENDA

1. Einführung
2. SQL Model Clause Syntax
3. Performance
4. Architektur
5. Framework
6. **Zusammenfassung**

Zusammenfassung

- Einfache Syntax für verschachtelte Berechnungen
  - Excel - ähnliche Syntax
  - Viele Optionen
  - Aber Grenzen bei Funktionalität
- Performance bei großen Datenmengen
  - Session Memory Bedarf beachten
- Zentral in Datenbank abgelegt
  - In Views definiert
  - Einfache Einbindung in ETL - Prozess
- Einfache Anpassungen über Benutzeroberfläche
  - Zugriff von Power User über APEX Oberfläche
  - Unterstützung bei Definition und Anpassungen
  - Versionierung der Berechnungen

Dokumentation

- Database Data Warehousing Guide 12c Rel. 1
  - Kap 21: SQL for Modeling