Nested Table Types als Ergänzung zu Pivot XML

Thomas Strub

22. November 2012

DOAG Konferenz Nürnberg

CGI

Logica is now part of CGI.

logica

Thomas Strub,
Consultant BI

Beratungsschwerpunkte & Technologien

- Optimierung bestehender DWH Prozesse
- Technisches DWH Architekturdesign
- Entwicklung dynamischer Reports
- Kampagnenmanagement und E-Commerce Reporting
- Oracle DB, OWB, ODI, PL/SQL

Qualifikationen & Zertifizierungen

- Diplom-Informationswirt (Uni)
- Zertifizierung BI-Framework

Agenda

Motivation
Exkurs Pivot
Exkurs Unpivot
Weiterverarbeitung
Laufzeitvergleich (mit Livedemo)
Fazit

Motivation Pivot

LOC	JOB_SUF	CNT	AMOUNT
1700	MAN	1	11.000
1800	REP	1	6.000
1700	VP	2	34.000
2700	REP	1	10.000
1700	PRES	1	24.000
2500	MAN	5	61.000
1700	ASST	1	4.400
1700	MGR	2	24.000
1700	ACCOUNT	6	47.900
1500	MAN	5	36.400
1500	CLERK	40	120.000
1400	PROG	5	28.800
2400	REP	1	6.500
1700	CLERK	5	13.900
2500	REP	29	243.500
1800	MAN	1	13.000

- Reporting übersichtlicher
  - Zeilen zusammengefasst
  - Zwischensummen
- Weitere Kalkulationen in der Zeile möglich
- Vorverarbeitung von Daten
- Datenaustausch

SUM AMOUNT	Spalten									
Zeilen	ACCOUNT	ASST	CLERK	MAN	MGR	PRES	PROG	REP	VP	Gesamt
1400							28.800			28.800
1500			120.000	36.400						156.400
1700	47.900	4.400	13.900	11.000	24.000	24.000			34.000	159.200
1800				13.000				6.000		19.000
2400								6.500		6.500
2500				61.000				243.500		304.500
2700								10.000		10.000
Gesamt	**47.900**	**4.400**	**133.900**	**121.400**	**24.000**	**24.000**	**28.800**	**266.000**	**34.000**	**684.400**

Grunddaten – HR.EMPLOYEES

```
Create view v_pivot_grunddaten as
SELECT location_id ,
  SUBSTR(job_id,4,10) job_suf ,
  COUNT(*) cnt ,
  SUM(salary) amount
FROM hr.employees e
JOIN hr.departments d USING (department_id)
GROUP BY location_id ,
  SUBSTR(job_id,4,10);
```

	LOCATION_ID	JOB_SUF	CNT	AMOUNT
1	1700	MAN	1	11000
2	1800	REP	1	6000
3	1700	VP	2	34000
4	2700	REP	1	10000
5	1700	PRES	1	24000
6	2500	MAN	5	61000
7	1700	ASST	1	4400
8	1700	MGR	2	24000
9	1700	ACCOUNT	6	47900

Pivot

Einfaches pivot mit zwei Wertspalten

```
select * from
  v_pivot_grunddaten
PIVOT (sum(cnt) anz_ma,
     sum(amount) sum_geh
FOR job_suf IN ('ACCOUNT' as Account, 'CLERK' as Clerk, 'MAN' as Manager))
    order by location_id;
```

	LOCATION_ID	ACCOUNT_ANZ_MA	ACCOUNT_SUM_GEH	CLERK_ANZ_MA	CLERK_SUM_GEH	MANAGER_ANZ_MA	MANAGER_SUM_GEH
1	1400	(null)	(null)	(null)	(null)	(null)	(null)
2	1500	(null)	(null)	40	120000	5	36400
3	1700	6	47900	5	13900	1	11000
4	1800	(null)	(null)	(null)	(null)	1	13000
5	2400	(null)	(null)	(null)	(null)	(null)	(null)

Pivot XML

```
SELECT *
FROM v_pivot_grunddaten
PIVOT XML (SUM (cnt) anzahl, SUM(amount) wert FOR job_suf IN (any) )
ORDER BY location_id;
```

	LOCATION_ID	JOB_SUF_XML
1	1400	<PivotSet><item><column name = "JOB_SUF">PROG</column><column name = "ANZAH...
2	1500	<PivotSet><item><column name = "JOB_SUF">CLERK</column><column name = "ANZA...
3	1700	<PivotSet><item><column name = "JOB_SUF">ACCOUNT</column><column name = "AN...
4	1800	<PivotSet><item><column name = "JOB_SUF">MAN</column><column name = "ANZAHL...
5	2400	<PivotSet><item><column name = "JOB_SUF">REP</column><column name = "ANZAHL...
6	2500	<PivotSet><item><column name = "JOB_SUF">MAN</column><column name = "ANZAHL...
7	2700	<PivotSet><item><column name = "JOB_SUF">REP</column><column name = "ANZAHL...

```
<PivotSet>
<item><column name = "JOB_SUF">ACCOUNT</column><column name = "ANZAHL">6</column><column name = "WERT">47900</column></item>
<item><column name = "JOB_SUF">ASST</column><column name = "ANZAHL">1</column><column name = "WERT">4400</column></item>
<item><column name = "JOB_SUF">CLERK</column><column name = "ANZAHL">5</column><column name = "WERT">13900</column></item>
<item><column name = "JOB_SUF">MAN</column><column name = "ANZAHL">1</column><column name = "WERT">11000</column></item>
<item><column name = "JOB_SUF">MGR</column><column name = "ANZAHL">2</column><column name = "WERT">24000</column></item>
<item><column name = "JOB_SUF">PRES</column><column name = "ANZAHL">1</column><column name = "WERT">24000</column></item>
<item><column name = "JOB_SUF">VP</column><column name = "ANZAHL">2</column><column name = "WERT">34000</column></item>
</PivotSet>
```

Pivot – Nested Table – collect transponiert

```
create type hr_pivot_obj as object
(job_suf varchar2(10), anzahl number(10,0), summe  number(20,2));
create type hr_pivot_ntt as table of hr_pivot_obj;

SELECT location_id ,
     CAST(COLLECT(hr_pv_obj(job_suf,cnt,amount))AS hr_pv_ntt) AS example_ntt
FROM
 v_pivot_grunddaten
GROUP BY location_id;
```

	LOCATION_ID	EXAMPLE_NTT
1	1400	HR.HR_PV_NTT('HR.HR_PV_OBJ('PROG',5,28800)')
2	1500	HR.HR_PV_NTT('HR.HR_PV_OBJ('MAN',5,36400)','HR.HR_PV_OBJ('CLERK',40,120000)')
3	1700	HR.HR_PV_NTT('HR.HR_PV_OBJ('MAN',1,11000)','HR.HR_PV_OBJ('CLERK',5,13900)','HR.HR_PV_OBJ('ACCOUNT',6,47900)','H
4	1800	HR.HR_PV_NTT('HR.HR_PV_OBJ('REP',1,6000)','HR.HR_PV_OBJ('MAN',1,13000)')
5	2400	HR.HR_PV_NTT('HR.HR_PV_OBJ('REP',1,6500)')
6	2500	HR.HR_PV_NTT('HR.HR_PV_OBJ('MAN',5,61000)','HR.HR_PV_OBJ('REP',29,243500)')
7	2700	HR.HR_PV_NTT('HR.HR_PV_OBJ('REP',1,10000)')

Unpivot

```
SELECT *
FROM
  (SELECT location_id, account_sum_geh, clerk_sum_geh,manager_sum_geh
 FROM v_pivot_s
  ) unpivot  /* include nulls */
(summe FOR job_suf IN ( account_sum_geh as 'account'
,clerk_sum_geh as 'clerk', manager_sum_geh as 'manager')
);
```

	LOCATION_ID	JOB_SUF	SUMME
1	1500	clerk	120000
2	1500	manager	36400
3	1700	account	47900
4	1700	clerk	13900
5	1700	manager	11000
6	1800	manager	13000
7	2500	manager	61000

Unpivot mit Nested Tables

```
select v.location_id, m.job_suf, m.anzahl, m.summe
from v_pv_ntt_standard v, table(EXAMPLE_NTT) m;
```

	LOCATION_ID	JOB_SUF	ANZAHL	SUMME
1	1400	PROG	5	28800
2	1500	MAN	5	36400
3	1500	CLERK	40	120000
4	1700	MAN	1	11000
5	1700	CLERK	5	13900
6	1700	ACCOUNT	6	47900
7	1700	MGR	2	24000
8	1700	ASST	1	4400
9	1700	PRES	1	24000

Weiterverarbeitung von Nested Tables

- Summenbildung, Durchschnitt, Top-N Werte, usw.
- Nachaggregation von Daten über custom aggregation function
- Umsortierungen
- Listenabgleiche
- Projektionen

Objekte - Weiterverarbeitung

```
 create table million_rows
(PIVOTING_COL     NUMBER(10,0),
GROUPING_COL      NUMBER(10,0),
SUMMING_COL       NUMBER(10,2),
PADDING_COL       VARCHAR2(70)
)    NOLOGGING ;
/


 -- 10 x 10 - 0.5 Mio
insert into million_rows
SELECT TRUNC
(DBMS_RANDOM.VALUE(1,10)) AS
pivoting_col
 , MOD(ROWNUM,10)+10   AS grouping_col
 , DBMS_RANDOM.VALUE   AS summing_col
 , RPAD('X',70,'X')    AS padding_col
  FROM    dual
 CONNECT BY ROWNUM <= 500000;
```

```
-- 10 col example - 4_weiterverarbeitung
create table t_4_w_pivot_table
(GROUPING_COL number,
v0 number,v1 number,v2 number,v3 number,
v4 number,v5 number,v6 number,v7 number,
v8 number,v9 number);


-- für NTT Beispiel
create type time_obj as object
 (grouping_col number
, wert number);
 /
create type time_ntt as table of time_obj;
  /

create table t_4_w_ntt_table
(grouping_col number
, my_ntt time_ntt)
 nested table my_ntt store as t4w_ntt return value;
```

Weiterverarbeitung – Daten laden

```
-- 4_weiterverarbeitung - pivot
insert into t_4_w_pivot_table
 SELECT  *
   FROM
              (SELECT pivoting_col,
              grouping_col,summing_col
              FROM million_rows)
PIVOT (SUM(summing_col) AS sum
  FOR    pivoting_col IN
        (0,1,2,3,4,5,6,7,8,9))
  ORDER  BY grouping_col;
```

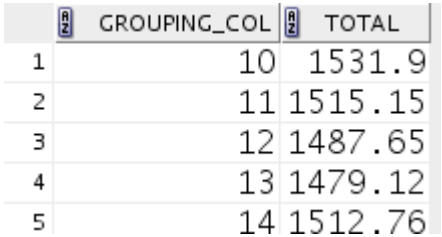
```
-- 4_weiterverarbeitung - ntt
insert into t_4_w_ntt_table
SELECT grouping_col ,
  CAST( collect(
time_obj(pivoting_col,summing_col))AS time_ntt)
 AS my_ntt
FROM
        (SELECT pivoting_col, grouping_col,
          SUM (summing_col) summing_col
        FROM million_rows
        GROUP BY pivoting_col, grouping_col)
GROUP BY grouping_col;
```

	GROUPING_COL	0_SUM	1_SUM	2_SUM
1	10	9936.29	10054.25	9944.75
2	11	9985.39	10157.07	10052.71
3	12	9624.81	10181.68	9933.94
4	13	9783.88	9700.94	9844.71

	GROUPING_COL	MY_NTT
1	10	HR.TIME_NTT('HR.TIME_OBJ(4,9986.33)','HR.TI
2	11	HR.TIME_NTT('HR.TIME_OBJ(1,10157.07)','HR.TI
3	12	HR.TIME_NTT('HR.TIME_OBJ(5,10215.21)','HR.T
4	13	HR.TIME_NTT('HR.TIME_OBJ(1,9700.94)','HR.TIME

Weiterverarbeitung Summenbildung

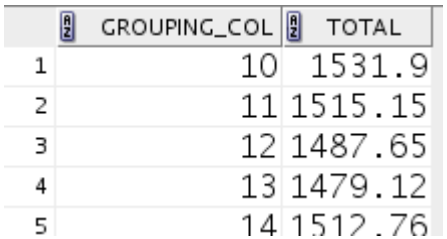
```
select grouping_col,
   nvl(v0,0)+nvl(v1,0)+nvl(v2,0)+nvl(v3,0)
+nvl(v4,0)+nvl(v5,0)+nvl(v6,0)+nvl(v7,0)
+nvl(v8,0)+nvl(v9,0)total
  from t_4_w_pivot_table
```



	GROUPING_COL	TOTAL
1	10	1531.9
2	11	1515.15
3	12	1487.65
4	13	1479.12
5	14	1512.76

vs NTT:

```
select t.grouping_col
, FCT_SUM_VALUES2(my_ntt)total
from t_4_w_ntt_table t
```



	GROUPING_COL	TOTAL
1	10	1531.9
2	11	1515.15
3	12	1487.65
4	13	1479.12
5	14	1512.76

Laufzeitvergleich – Livedemo

Ergebnisse:

	Pivot	Pivot XML	Pivot NTT
P 10 x 10 – 500.000	1,6s	0,7s	0,65 s
P 10 x 10.000 – 1.000.000	6 s	13 s	4 s
P 100 x 10 – 500.000	1,8s	0,8s	0,9s
P 100 x 10.000 – 500.000	4,3s	43s	7,6s
W Summe 10 x 10.000	0,02s		0,08s

Fazit

- Nicht immer schneller
- Eleganter zu programmieren
- Wo möglich normalisiertes Datenmodell verwenden
- Austausch zu Frontend möglich, im Projekt greift ein Ext JS 3.x Frontend auf die Daten zu.

Thomas Strub
Consultant BI
Thomas.Strub@logica.com
+49 151 402 34 650

Thank you

Weitere Logica Vorträge auf der DOAG

Mittwoch 21.11.2012

Oracle Messaging:
AQ vs WebLogic JMS
Bertrand Caradec
9:00 Uhr
Konferezraum EG

BI Lifecycle –
Wildwuchs oder
klare Vorgaben?
Dirk Braunecker
14:00 Uhr
Stockholm

Donnerstag 22.11.2012

Nested Tables Types
als Ergänzung zu Pivot XML
Thomas Strub
9:00 Uhr
Konferenzraum EG

Forecasting Oracle Performance
– Better than a crystal ball
Yuri van Buren
15:00 Uhr
St. Petersburg

ODI Projekt Lifecycle - Segen
oder Fluch?
Michael Klose
16:00 Uhr
Kiew