Advanced SQL in Oracle and SQL Server

The PIVOT and UNPIVOT Features

Scott L. Hecht
http://www.sheepsqueezers.com
@sheepsqueezers

pluralsight
hardcore developer training

Module Contents

- **The `PIVOT` and `UNPIVOT` Features**
  - *Vintage* Data Transposition
    - Introduction
    - Turning Columns into Rows (Multi-Table INSERT/UNION)
    - Turning Rows into Columns (CASE Statements)
  - *Modern* Data Transposition
    - Introduction
    - Turning Columns into Rows (UNPIVOT)
    - Turning Rows into Columns (PIVOT)
    - Multi-Column PIVOT (Oracle-Specific Syntax)
      - Oracle's Extension to the IN Function Explained
      - Multi-Column PIVOT Example
  - Summary

Data Used in Module

- **Table**
  - CANDYBAR_CONSUMPTION_DATA
- **Columns**
  - CONSUMER_ID – unique identifier of a consumer
  - CANDYBAR_NAME – name of candy bar (e.g., MARS BAR, TWIX BAR, …)
  - SURVEY_YEAR – year of survey responses (e.g., 2009, 2010, …)
  - GENDER – gender of respondent (e.g., M=Male, F=Female)
  - OVERALL_RATING– rating of candy bar ranging from 1=Low to 10=High
  - NUMBER_BARS_CONSUMED – number of candy bars consumed during year
- **Data**

CONSUMER_ID	CANDYBAR_NAME	SURVEY_YEAR	GENDER	OVERALL_RATING	NUMBER_BARS_CONSUMED
1	MARS BAR	2009	M	10	252
1	MARS BAR	2010	M	10	352
1	MARS BAR	2011	M	10	452
1	TWIX BAR	2009	M	10	6
1	TWIX BAR	2010	M	7	60
1	TWIX BAR	2011	M	8	600

…continues on next slide…

Data Used in Module

- Data (*continued*)

CONSUMER_ID	CANDYBAR_NAME	SURVEY_YEAR	GENDER	OVERALL_RATING	NUMBER_BARS_CONSUMED
2	HERSHEY BAR	2009	F	5	2
2	HERSHEY BAR	2010	F	5	3
2	HERSHEY BAR	2011	F	5	1
2	MARS BAR	2009	F	8	25
2	MARS BAR	2010	F	8	12
2	MARS BAR	2011	F	8	13
3	MARS BAR	2009	M	8	25
3	MARS BAR	2010	M	7	12
3	MARS BAR	2011	M	8	13
3	TWIX BAR	2009	M	7	6
3	TWIX BAR	2010	M	8	60
3	TWIX BAR	2011	M	9	600
4	HERSHEY BAR	2009	F	7	20
4	HERSHEY BAR	2010	F	7	30
4	HERSHEY BAR	2011	F	7	10

…continues on next slide…

Data Used in Module

- Data (*continued*)

CONSUMER_ID	CANDYBAR_NAME	SURVEY_YEAR	GENDER	OVERALL_RATING	NUMBER_BARS_CONSUMED
4	MARS BAR	2009	F	7	25
4	MARS BAR	2010	F	7	35
4	MARS BAR	2011	F	7	15
4	TWIX BAR	2009	F	7	20
4	TWIX BAR	2010	F	7	30
4	TWIX BAR	2011	F	7	10
5	HERSHEY BAR	2009	M	8	15
5	HERSHEY BAR	2010	M	8	15
5	HERSHEY BAR	2011	M	6	5
5	SNICKERS BAR	2009	M	8	55
5	SNICKERS BAR	2010	M	8	65
5	SNICKERS BAR	2011	M	8	75
5	TWIX BAR	2009	M	9	75
5	TWIX BAR	2010	M	9	85
5	TWIX BAR	2011	M	9	95

Vintage Data Transposition

- **Introduction**
  - Life without PIVOT and UNPIVOT
  - Turning Columns into Rows
    - Oracle & SQL Server: INSERTs/UNIONs
    - Oracle: Multi-Table INSERT Feature
  - Turning Rows into Columns
    - CASE Statements with SUM() Aggregate

Vintage Data Transposition

- **Turning Columns into Rows**
  - Recall CANDYBAR_CONSUMPTION_DATA has these two columns:
    - OVERALL_RATING
    - NUMBER_BARS_CONSUMED
  - Want data in these columns to appear within a *single* column
  - How do we distinguish the values?
  - Need to create two replacement columns in new table:
    - STAT_TYPE – 1 indicates value is OVERALL_RATING, 2 is NUMBER_BARS_CONSUMED
    - STAT_VALUE – the actual value depending on STAT_TYPE

```
CREATE TABLE T_CANDYBAR_DATA(CONSUMER_ID NUMBER,
                             CANDYBAR_NAME VARCHAR2(50),
                             SURVEY_YEAR NUMBER,
                             GENDER VARCHAR2(1),
                             STAT_TYPE NUMBER,
                             STAT_VALUE NUMBER)
```

Vintage Data Transposition

- **Turning Columns into Rows**
  - One method is to insert the rows into the table with individual INSERTs

```
INSERT INTO T_CANDYBAR_DATA
 SELECT CONSUMER_ID,CANDYBAR_NAME,SURVEY_YEAR,GENDER,
        1 AS STAT_TYPE,OVERALL_RATING AS STAT_VALUE
  FROM CANDYBAR_CONSUMPTION_DATA;

INSERT INTO T_CANDYBAR_DATA
 SELECT CONSUMER_ID,CANDYBAR_NAME,SURVEY_YEAR,GENDER,
        2 AS STAT_TYPE,NUMBER_BARS_CONSUMED AS STAT_VALUE
  FROM CANDYBAR_CONSUMPTION_DATA;
```

Vintage Data Transposition

- **Turning Columns into Rows**
  - Another method is to use a single INSERT and UNIONs

```
INSERT INTO T_CANDYBAR_DATA
 SELECT CONSUMER_ID,CANDYBAR_NAME,SURVEY_YEAR,GENDER,
        1 AS STAT_TYPE,OVERALL_RATING
  FROM CANDYBAR_CONSUMPTION_DATA
 UNION
 SELECT CONSUMER_ID,CANDYBAR_NAME,SURVEY_YEAR,GENDER,
        2 AS STAT_TYPE,NUMBER_BARS_CONSUMED
  FROM CANDYBAR_CONSUMPTION_DATA
```

Vintage Data Transposition

- **Turning Columns into Rows**
  - In Oracle, you can use the Multi-Table INSERT feature:

```
INSERT ALL
 INTO T_CANDYBAR_DATA
  VALUES(CONSUMER_ID,CANDYBAR_NAME,SURVEY_YEAR,GENDER,
         1,OVERALL_RATING)
 INTO T_CANDYBAR_DATA
  VALUES(CONSUMER_ID,CANDYBAR_NAME,SURVEY_YEAR,GENDER,
         2,NUMBER_BARS_CONSUMED)
 SELECT *
  FROM CANDYBAR_CONSUMPTION_DATA
```

Vintage Data Transposition

- **Turning Columns into Rows**
  - In all cases, here is what the data looks like (before and after)

CONSUMER_ID	CANDYBAR_NAME	SURVEY_YEAR	GENDER	OVERALL_RATING	NBC
1	TWIX BAR	2009	M	10	6
1	TWIX BAR	2010	M	7	60
1	TWIX BAR	2011	M	8	600

…snip…

CONSUMER_ID	CANDYBAR_NAME	SURVEY_YEAR	GENDER	STAT_TYPE	STAT_VALUE
1	TWIX BAR	2009	M	1	10
1	TWIX BAR	2010	M	1	7
1	TWIX BAR	2011	M	1	8
1	TWIX BAR	2009	M	2	6
1	TWIX BAR	2010	M	2	60
1	TWIX BAR	2011	M	2	600

…snip…

Vintage Data Transposition

- **Turning Rows into Columns**
  - Given the T_CANDYBAR_DATA table we just produced:
    - Turn STAT_VALUE where STAT_TYPE=1 back into OVERALL_RATING
    - Turn STAT_VALUE where STAT_TYPE=2 back into NUMBER_BARS_CONSUMED
  - Use the CASE Statement to (re-)create these two columns

```
SELECT CONSUMER_ID,CANDYBAR_NAME,SURVEY_YEAR,GENDER,
       CASE
        WHEN STAT_TYPE=1 THEN STAT_VALUE
       END AS OAR,
       CASE
        WHEN STAT_TYPE=2 THEN STAT_VALUE
       END AS NBC
 FROM T_CANDYBAR_DATA
```

Vintage Data Transposition

- **Turning Rows into Columns**
  - Take note of the missing values!

CONSUMER_ID	CANDYBAR_NAME	SURVEY_YEAR	GENDER	OAR	NBC
1	TWIX BAR	2009	M	10	
1	TWIX BAR	2010	M	7	
1	TWIX BAR	2011	M	8	
1	TWIX BAR	2009	M		6
1	TWIX BAR	2010	M		60
1	TWIX BAR	2011	M		600

…snip…

- **How Do We Fix This?**
  - Use SUM() Function to collapse the rows (72 to 36)
  - NULLs have no effect on SUM()

Vintage Data Transposition

- **Turning Rows into Columns**
  - Use SUM() Function to collapse the rows (72 to 36)
  - NULLs have no effect on SUM()
  - SQL Server: *Warning: Null value is eliminated by an aggregate or other SET operation.*

```
INSERT INTO T_T_CANDYBAR_DATA
 SELECT A.CONSUMER_ID,A.CANDYBAR_NAME,A.SURVEY_YEAR,A.GENDER,
        SUM(A.OAR) AS OVERALL_RATING,
        SUM(A.NBC) AS NUMBER_BARS_CONSUMED
  FROM (
        SELECT CONSUMER_ID,CANDYBAR_NAME,SURVEY_YEAR,GENDER,
               CASE
                WHEN STAT_TYPE=1 THEN STAT_VALUE
               END AS OAR,
               CASE
                WHEN STAT_TYPE=2 THEN STAT_VALUE
               END AS NBC
         FROM T_CANDYBAR_DATA
       ) A
  GROUP BY A.CONSUMER_ID,A.CANDYBAR_NAME,A.SURVEY_YEAR,A.GENDER
```

Modern Data Transposition

- **Introduction**
  - Life with PIVOT and UNPIVOT
  - Turning Columns into Rows
    - Use UNPIVOT
  - Turning Rows into Columns
    - Use PIVOT

Modern Data Transposition

- **Turning Columns into Rows (UNPIVOT)**
  - Recall CANDYBAR_CONSUMPTION_DATA has these two columns:
    - OVERALL_RATING
    - NUMBER_BARS_CONSUMED
  - Want these columns to appear within a *single* column
  - How do we distinguish the values?
  - Need to create two replacement columns in new table:
    - STAT_TYPE – 1 indicates value is OVERALL_RATING, 2 is NUMBER_BARS_CONSUMED
    - STAT_VALUE – the actual value depending on STAT_TYPE
  - In this case, we use UNPIVOT

Modern Data Transposition

- **Turning Columns into Rows (UNPIVOT)**
  - Let's use the UNPIVOT Syntax

```
SELECT *
 FROM CANDYBAR_CONSUMPTION_DATA
 UNPIVOT (
          STAT_VALUE
           FOR STAT_TYPE IN (OVERALL_RATING,NUMBER_BARS_CONSUMED)
         ) U
```

  - Both STAT_VALUE and STAT_TYPE are not defined up-front!
  - UNPIVOT defines STAT_VALUE and STAT_TYPE
  - Works with both Oracle and SQL Server
  - Results *similar* as vintage multiple INSERT example

Modern Data Transposition

- **Turning Columns into Rows (UNPIVOT)**
  - But, instead of 1's and 2's, STAT_TYPE holds the textual column names!

CONSUMER_ID	CANDYBAR_NAME	SURVEY_YEAR	GENDER	STAT_TYPE	STAT_VALUE
1	TWIX BAR	2009	M	OVERALL_RATING	10
1	TWIX BAR	2010	M	OVERALL_RATING	7
1	TWIX BAR	2011	M	OVERALL_RATING	8
1	TWIX BAR	2009	M	NUMBER_BARS_CONSUMED	6
1	TWIX BAR	2010	M	NUMBER_BARS_CONSUMED	60
1	TWIX BAR	2011	M	NUMBER_BARS_CONSUMED	600

…snip…

Modern Data Transposition

- **Turning Columns into Rows (UNPIVOT)**
  - Can we replace the textual column names with 1's and 2's?

```
/* Oracle-Specific Syntax */
SELECT *
 FROM CANDYBAR_CONSUMPTION_DATA
 UNPIVOT (
          STAT_VALUE
           FOR STAT_TYPE IN (OVERALL_RATING AS 1,
                             NUMBER_BARS_CONSUMED AS 2)
         ) U
```

Modern Data Transposition

- **Turning Columns into Rows (UNPIVOT)**
  - Can we replace the textual column names with 1's and 2's?

```
/* SQL Server-Specific Syntax */
SELECT CONSUMER_ID,CANDYBAR_NAME,SURVEY_YEAR,GENDER,
       CASE
        WHEN STAT_TYPE='OVERALL_RATING' THEN 1
        WHEN STAT_TYPE='NUMBER_BARS_CONSUMED' THEN 2
       END AS STAT_TYPE,
       STAT_VALUE
 FROM CANDYBAR_CONSUMPTION_DATA
 UNPIVOT (
          STAT_VALUE
           FOR STAT_TYPE IN (OVERALL_RATING,NUMBER_BARS_CONSUMED)
         ) U
```

Modern Data Transposition

- **Turning Columns into Rows (UNPIVOT)**
  - Aliases cause problems in both Oracle and SQL Server
  - You will receive one or more error messages

```
SELECT A.CONSUMER_ID,A.CANDYBAR_NAME,A.SURVEY_YEAR,A.GENDER,
       U.STAT_TYPE,U.STAT_VALUE
 FROM CANDYBAR_CONSUMPTION_DATA A
 UNPIVOT (
          STAT_VALUE
           FOR STAT_TYPE IN (OVERALL_RATING,NUMBER_BARS_CONSUMED)
         ) U

/* Oracle Error Message: */
Error report:
SQL Error: ORA-00904: "A"."GENDER": invalid identifier

/* SQL Server Error Messages: */
Msg 4104, Level 16, State 1, Line 1
The multi-part identifier "A.CONSUMER_ID" could not be bound.
```

Modern Data Transposition

- **Turning Columns into Rows (UNPIVOT)**
  - Aliases cause problems in both Oracle and SQL Server
  - The code below works fine!

```
SELECT U.CONSUMER_ID,U.CANDYBAR_NAME,U.SURVEY_YEAR,U.GENDER,
       U.STAT_TYPE,U.STAT_VALUE
 FROM CANDYBAR_CONSUMPTION_DATA
 UNPIVOT (
          STAT_VALUE
           FOR STAT_TYPE IN (OVERALL_RATING,NUMBER_BARS_CONSUMED)
         ) U
```

Modern Data Transposition

- **Turning Rows into Columns (PIVOT)**
  - Recall the T_CANDYBAR_DATA table
    - Turn STAT_VALUE where STAT_TYPE=1 back into OVERALL_RATING
    - Turn STAT_VALUE where STAT_TYPE=2 back into NUMBER_BARS_CONSUMED
  - In this case, we use PIVOT
  - Syntax differs between databases
  - Note that STAT_TYPE and STAT_VALUE already exist in the table!!
  - Your table may not contain aggregated rows, so…
    - …aggregate your numeric column (usually SUM())
    - …let PIVOT do that for you!

Modern Data Transposition

- **Turning Rows into Columns (PIVOT)**
  - Let's use PIVOT!

```
/* Oracle-Specific Syntax */
SELECT *
 FROM T_CANDYBAR_DATA
 PIVOT (
        SUM(STAT_VALUE)
         FOR STAT_TYPE IN (1,2)
       )
```

```
/* SQL Server-Specific Syntax */
SELECT *
 FROM T_CANDYBAR_DATA
 PIVOT (
        SUM(STAT_VALUE)
         FOR STAT_TYPE IN ([1],[2])
       ) P
```

  - Sadly, cannot use a sub-query with the IN() Function in this case!

Modern Data Transposition

- **Turning Rows into Columns (PIVOT)**
  - In both cases, the transposed columns are named 1 and 2

CONSUMER_ID	CANDYBAR_NAME	SURVEY_YEAR	GENDER	1	2
1	TWIX BAR	2009	M	10	6
1	TWIX BAR	2010	M	7	60
1	TWIX BAR	2011	M	8	600

…snip…

Modern Data Transposition

- **Turning Rows into Columns (PIVOT)**
  - Use the appropriate column names!

```
/* Oracle-Specific Syntax */
SELECT CONSUMER_ID,CANDYBAR_NAME,SURVEY_YEAR,GENDER,
       OVERALL_RATING,NUMBER_BARS_CONSUMED
 FROM T_CANDYBAR_DATA
 PIVOT (
        SUM(STAT_VALUE)
         FOR STAT_TYPE IN (1 AS OVERALL_RATING,
                           2 AS NUMBER_BARS_CONSUMED)
       )

/* SQL Server-Specific Syntax */
SELECT CONSUMER_ID,CANDYBAR_NAME,SURVEY_YEAR,GENDER,
       [1] AS OVERALL_RATING,[2] AS NUMBER_BARS_CONSUMED
 FROM T_CANDYBAR_DATA
 PIVOT (
        SUM(STAT_VALUE)
         FOR STAT_TYPE IN ([1],[2])
       ) P
```

Modern Data Transposition

- **Turning Rows into Columns (PIVOT)**
  - Oracle-Specific Extensions
  - Can give a name to your aggregate column!

```
/* Oracle-Specific Syntax */
SELECT *
 FROM T_CANDYBAR_DATA
 PIVOT (
        SUM(STAT_VALUE) AS SUMSTAT
         FOR STAT_TYPE IN (1,2)
       )
```

CONSUMER_ID	CANDYBAR_NAME	SURVEY_YEAR	GENDER	1_SUMSTAT	2_SUMSTAT
1	TWIX BAR	2009	M	10	6
1	TWIX BAR	2010	M	7	60
1	TWIX BAR	2011	M	8	600

…snip…

Modern Data Transposition

- **Turning Rows into Columns (PIVOT)**
  - Oracle-Specific Extensions
  - Can give a names to the aggregate and STAT_TYPE columns!

```
/* Oracle-Specific Syntax */
SELECT *
 FROM T_CANDYBAR_DATA
 PIVOT (
        SUM(STAT_VALUE) AS SUMSTAT
         FOR STAT_TYPE IN (1 AS OAR,
                           2 AS NBC)
       )
```

CONSUMER_ID	CANDYBAR_NAME	SURVEY_YEAR	GENDER	OAR_SUMSTAT	NBC_SUMSTAT
1	TWIX BAR	2009	M	10	6
1	TWIX BAR	2010	M	7	60
1	TWIX BAR	2011	M	8	600

…snip…

Modern Data Transposition

- **Turning Rows into Columns (PIVOT)**
  - Oracle-Specific Extensions
  - Can use multiple aggregates!

```
/* Oracle-Specific Syntax */
SELECT *
 FROM T_CANDYBAR_DATA
 PIVOT (
        SUM(STAT_VALUE) AS SM,
        MIN(STAT_VALUE) AS MN,
        MAX(STAT_VALUE) AS MX
         FOR STAT_TYPE IN (1 AS OAR,
                           2 AS NBC)
       )
```

CONSUMER_ID	CANDYBAR_NAME	SURVEY_YEAR	GENDER	OAR_SM	OAR_MN	OAR_MX	NBC_SM	NBC_MN	NBC_MX
1	TWIX BAR	2009	M	10	10	10	6	6	6
1	TWIX BAR	2010	M	7	7	7	60	60	60
1	TWIX BAR	2011	M	8	8	8	600	600	600

…snip…

Modern Data Transposition

- **Multi-Column PIVOT**
  - Recall that PIVOT takes rows of data and turns them into columns
  - Only transposed one column so far: STAT_VALUE based on STAT_TYPE
  - Oracle's syntax allows you to pivot more than one column at a time
  - SQL Server's syntax does not allow for this!
  - For example, we can pivot both SURVEY_YEAR and STAT_TYPE at the same time.
    - SURVEY_YEAR contains 3 unique values
    - STAT_TYPE contains 2 unique values
    - The resulting number of columns is 3*2=6
  - Must learn about Oracle's Extension to the IN() Function first!

Modern Data Transposition

- **Oracle's Extension to the IN() Function**
  - We all know how to use the standard IN() Function:

```
SELECT *
 FROM T_CANDYBAR_DATA
 WHERE STAT_TYPE IN (1,2)
```

  - Multiple conditions? We fall back on ANDs and ORs:

```
SELECT *
 FROM CANDYBAR_DATA
 WHERE (CANDYBAR_NAME='MARS BAR' AND SURVEY_YEAR=2010 AND GENDER='M')
    OR (CANDYBAR_NAME='TWIX BAR' AND SURVEY_YEAR=2010 AND GENDER='F')
    OR (CANDYBAR_NAME='HERSHEY BAR' AND SURVEY_YEAR=2010 AND GENDER='M')
```

Modern Data Transposition

- **Oracle's Extension to the IN() Function**
  - Use Oracle's Extension to IN() to avoid ANDs/ORs:

```
SELECT *
 FROM T_CANDYBAR_DATA
 WHERE (CANDYBAR_NAME,SURVEY_YEAR,GENDER) IN (
                                              ('MARS BAR',2010,'M'),
                                              ('TWIX BAR',2011,'F'),
                                              ('HERSHEY BAR',2011,'M')
                                             )
```

  - Cannot use NULLs!
  - But, can use a combination of both IN() and ANDs/ORs:

```
SELECT *
 FROM T_CANDYBAR_DATA
 WHERE (CANDYBAR_NAME,SURVEY_YEAR,GENDER) IN (
                                              ('MARS BAR',2010,'M'),
                                              ('TWIX BAR',2011,'F'),
                                              ('HERSHEY BAR',2011,'M')
                                             )
       OR (CANDYBAR_NAME='HERSHEY BAR' AND SURVEY_YEAR=2011
                                                 AND GENDER IS NULL)
```

Modern Data Transposition

- **Multi-Column PIVOT**
  - Can pivot more than one column at a time
  - Uses Oracle's Extension to IN()
  - Let's pivot the SURVEY_YEAR and the STAT_TYPE:

```
SELECT *
 FROM T_CANDYBAR_DATA
 PIVOT (
        SUM(STAT_VALUE)
        FOR (SURVEY_YEAR,STAT_TYPE) IN (
                                         (2009,1) AS Y2009_OAR,
                                         (2009,2) AS Y2009_NBC,
                                         (2010,1) AS Y2010_OAR,
                                         (2010,2) AS Y2010_NBC,
                                         (2011,1) AS Y2011_OAR,
                                         (2011,2) AS Y2011_NBC
                                        )
       )
```

Modern Data Transposition

- **Multi-Column PIVOT**
  - Can pivot more than one column at a time
  - Uses Oracle's Extension to IN()
  - Let's pivot the SURVEY_YEAR and the STAT_TYPE

CONSUMER_ID	CANDYBAR_NAME	GENDER	Y2009_OAR	Y2009_NBC	Y2010_OAR	Y2010_NBC	Y2011_OAR	Y2011_NBC
1	MARS BAR	M	10	252	10	352	10	452
1	TWIX BAR	M	10	6	7	60	8	600
2	HERSHEY BAR	F	5	2	5	3	5	1
2	MARS BAR	F	8	25	8	12	8	13
3	MARS BAR	M	8	25	7	12	8	13
3	TWIX BAR	M	7	6	8	60	9	600
4	HERSHEY BAR	F	7	20	7	30	7	10
4	MARS BAR	F	7	25	7	35	7	15
4	TWIX BAR	F	7	20	7	30	7	10
5	HERSHEY BAR	M	8	15	8	15	6	5
5	SNICKERS BAR	M	8	55	8	65	8	75
5	TWIX BAR	M	9	75	9	85	9	95

Summary

- Transpose Columns to Rows (UNPIVOT)
- Transpose Rows to Columns (PIVOT)
- Multi-Column PIVOT