# Efficient UNPIVOT

Had to take a set of repeating attributes that really should have been a many to one child table, and turn it into a child virtual table.

Question was what is the move efficient method of unpivoting data.

## 11g UNPIVOT

**SELECT** alc.comp\_id

,alc.compl\_pri\_id

,alc.compl\_pri\_num

**FROM** asg\_loc\_components

**UNPIVOT** (compl\_pri\_num **FOR** compl\_pri\_id **IN** ( top\_compl\_num **AS** 1

, high\_compl\_num **AS** 2

, medium\_compl\_num **AS** 3

, low\_compl\_num **AS** 4)

) alc

**WHERE** alc.misstyp\_id **IN** (3, 5)

**AND** alc.compsta\_id = 1

**AND** alc.comp\_num > 0;

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Operation | Owner | Object Name | Cost | CPU Cost | IO Cost | Card | Bytes | Filter Predicates | Access Predicates |
| SELECT STATEMENT, GOAL = ALL\_ROWS 433 32431946 432 8700 591600  VIEW SYS 433 32431946 432 8700 591600 "unpivot\_view\_007"."COMPL\_PRI\_NUM" IS NOT NULL  UNPIVOT  TABLE ACCESS FULL MSSW ASG\_LOC\_COMPONENTS 108 8140612 108 2175 47850 "ASG\_LOC\_COMPONENTS"."COMP\_NUM">0 AND ("ASG\_LOC\_COMPONENTS"."MISSTYP\_ID"=3 OR "ASG\_LOC\_COMPONENTS"."MISSTYP\_ID"=5) AND "ASG\_LOC\_COMPONENTS"."COMPSTA\_ID"=1 | | | | | | | | | |

## CROSS JOIN to UNION ALL’d list of literals

**SELECT** alc.comp\_id,

numlist.n **AS** comp\_pri\_id,

(**CASE** **WHEN** numlist.n = 1 **THEN** top\_compl\_num

**WHEN** numlist.n = 2 **THEN** high\_compl\_num

**WHEN** numlist.n = 3 **THEN** medium\_compl\_num

**WHEN** numlist.n = 4 **THEN** low\_compl\_num

**END**) **AS** comp\_pri\_num

**FROM** asg\_loc\_components alc

**CROSS** **JOIN**

(**SELECT** 1 **AS** n **FROM** dual **UNION** **ALL**

**SELECT** 2 **AS** n **FROM** dual **UNION** **ALL**

**SELECT** 3 **AS** n **FROM** dual **UNION** **ALL**

**SELECT** 4 **AS** n **FROM** dual

) numlist

**WHERE** alc.misstyp\_id **IN** (3, 5)

**AND** alc.compsta\_id = 1

**AND** alc.comppri\_id **IS** **NULL**

**AND** alc.comp\_num > 0

**ORDER** **BY** comp\_id, comp\_pri\_id;

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Operation | Owner | Object Name | Cost | CPU Cost | IO Cost | Card | Bytes | Filter Predicates | Access Predicates |
| SELECT STATEMENT, GOAL = ALL\_ROWS 436 65417134 434 8701 217525  SORT ORDER BY 436 65417134 434 8701 217525  MERGE JOIN CARTESIAN 435 33315798 434 8701 217525  VIEW MSSW 8 29086 8 4 12  UNION-ALL  FAST DUAL 2 7271 2 1  FAST DUAL 2 7271 2 1  FAST DUAL 2 7271 2 1  FAST DUAL 2 7271 2 1  BUFFER SORT 436 65417134 434 2175 47850  TABLE ACCESS FULL MSSW ASG\_LOC\_COMPONENTS 107 8321678 107 2175 47850 "ALC"."COMP\_NUM">0 AND ("ALC"."MISSTYP\_ID"=3 OR "ALC"."MISSTYP\_ID"=5) AND "ALC"."COMPSTA\_ID"=1 | | | | | | | | | |

## CONNECT BY LEVEL for list of literals

**SELECT** alc.comp\_id,

n.pri\_id **AS** comp\_pri\_id,

**DECODE**(n.pri\_id

,1,top\_compl\_num

,2,high\_compl\_num

,3,medium\_compl\_num

,4,low\_compl\_num) **AS** comp\_pri\_num

**FROM** asg\_loc\_components alc

**JOIN** (**SELECT** **LEVEL** **AS** pri\_id **FROM** dual **CONNECT** **BY** **LEVEL** <= 4) n **ON** 1=1

**WHERE** alc.misstyp\_id **IN** (3, 5)

**AND** alc.compsta\_id = 1

**AND** alc.comp\_num > 0

**ORDER** **BY** alc.comp\_id, n.pri\_id;

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Operation | Owner | Object Name | Cost | CPU Cost | IO Cost | Card | Bytes | Filter Predicates | Access Predicates |
| SELECT STATEMENT, GOAL = ALL\_ROWS 111 36386372 110 2175 76125  SORT ORDER BY 111 36386372 110 2175 76125  MERGE JOIN CARTESIAN 110 8328949 110 2175 76125  VIEW MSSW 2 7271 2 1 13  CONNECT BY WITHOUT FILTERING LEVEL<=4  FAST DUAL 2 7271 2 1  BUFFER SORT 111 36386372 110 2175 47850  TABLE ACCESS FULL MSSW ASG\_LOC\_COMPONENTS 108 8321678 108 2175 47850 "ALC"."COMP\_NUM">0 AND ("ALC"."MISSTYP\_ID"=3 OR "ALC"."MISSTYP\_ID"=5) AND "ALC"."COMPSTA\_ID"=1 | | | | | | | | | |