```
(p_vData IN ANAGRAFICA_SORGENTE.DataMod%TYPE) IS
  -- Seleziona tutte le righe modificate oltre la data indicata.
  CURSOR c_Sorgente IS
    SELECT *
    FROM ANAGRAFICA_SORGENTE
    WHERE FlagMod = 'M'
    AND DataMod > p_vData;

  v_Trovato number(1);
  v_CodComune COMUNI.CodComune%TYPE;
BEGIN
  -- Per ogni riga interessata
  FOR r_Sorgente IN c_Sorgente LOOP

    -- Cerca il comune.
    SELECT COUNT(*) INTO v_Trovato
    FROM COMUNI
    WHERE Nome=r_Sorgente.Comune
    AND CAP=r_Sorgente.CAP;

    -- Se non esiste un comune con quel nome e quel CAP
    IF v_Trovato = 0 THEN
      INSERT INTO ANAGRAFICA_SCARTI VALUES (r_Sorgente.CodP, r_Sorgente.Nome,
r_Sorgente.Cognome, r_Sorgente.Età, r_Sorgente.Cap, r_Sorgente.Comune, 'R1');
    -- Se l'età è fuori dal range stabilito
    ELSIF r_Sorgente.Età NOT BETWEEN 0 AND 150 THEN
      INSERT INTO ANAGRAFICA_SCARTI VALUES (r_Sorgente.CodP, r_Sorgente.Nome,
r_Sorgente.Cognome, r_Sorgente.Età, r_Sorgente.Cap, r_Sorgente.Comune, 'R2');
    -- Se nessuna regola è violata.
    ELSE
      -- Seleziona il codice del comune. Visto che Nome e Cap sono chiave
secondaria verrà selezionata sempre una e una sola riga.
      SELECT CodComune INTO v_CodComune
      FROM COMUNI
      WHERE Nome=r_Sorgente.Comune
      AND CAP=r_Sorgente.CAP;

      INSERT INTO ANAGRAFICA_DESTINAZIONE VALUES (r_Sorgente.CodP,
r_Sorgente.Nome, r_Sorgente.Cognome, r_Sorgente.Età, v_CodComune);
    END IF;

  END LOOP;
END;
```

```
explain plan for select P_NAME, S_NAME,PS_AVAILQTY
from PART,PARTSUPP,SUPPLIER
where P_PARTKEY=PS_PARTKEY and S_SUPPKEY=PS_SUPPKEY
and P_TYPE='LARGE ANODIZED BRASS' AND S_NATIONKEY=16;
@?/RDBMS/ADMIN/UTLXPLS;

Plan Table
--------------------------------------------------------------------------------
| Operation                 |  Name    |  Rows | Bytes|  Cost  | Pstart| Pstop |
--------------------------------------------------------------------------------
| SELECT STATEMENT          |          |   213 |   22K|   5251 |       |       |
|  HASH JOIN                |          |   213 |   22K|   5251 |       |       |
|   TABLE ACCESS FULL       |SUPPLIER  |   400 |   12K|     65 |       |       |
|   NESTED LOOPS            |          |     5K|  380K|   5164 |       |       |
|    TABLE ACCESS FULL      |PART      |     1K|   78K|   1162 |       |       |
|    TABLE ACCESS BY INDEX R|PARTSUPP  |   800K|    9M|      3 |       |       |
|     INDEX RANGE SCAN      |IX_PART_P |   800K|      |      2 |       |       |
--------------------------------------------------------------------------------
```

$NP_{PART} = \lceil 200.000 \times 130/ 4096 \rceil = \mathbf{6.348}$
$NP_{SUPPLIER} = \lceil 10.000 \times 143/ 4096 \rceil = \mathbf{350}$
$NP_{PARTSUPP} = \lceil 800.000 \times 142 / 4.096 \rceil = \mathbf{27.735}$

$Sel(P\_TYPE='LARGE\ ANODIZED\ BRASS') = 1/150$
$Sel(S\_NATIONKEY=16) = 1/25$
$ET_{P\_TYPE='LARGE\ ANODIZED\ BRASS'} = \lceil 200.000/150 \rceil = 1.334$
$EP_{S\_NATIONKEY=16} = \lceil 350/25 \rceil \cong \lceil 10.000 / 25 \times 143/ 4096 \rceil = 14$

$NL_{IX\_PART\_PARTSUPP} = \lceil (800.000 \times 4 + 4 \times 200.000) / 4096 \rceil = 977$

PARTSUPP è ordinata su PS_PARTKEY quindi l'accesso alla tabella è ordinato.
A ogni tupla di PART corrispondono in media 4 tuple di PARTSUPP (800.000/200.000)

$2\text{-}1 + \lceil 1/200.000 \times 977 \rceil + \lceil 4 \times 27.735 / 800.000 \rceil = 1 + 1 + 1 = 3$

Costo Nested Loop = $6.348 + 1.334 \times 3 =$ **10.350**

$NT_{PARTSUPP+PART} = 1.334 \times 4 = 5.336$
$NP_{PARTSUPP+PART} = \lceil 5.336 \times (142+130) / 4.096 \rceil = 355$

Il costo dell'hybrid hash join è calcolato senza considerare la pipeline

Costo hybrid hash join = $355 + 350 + 2 \times (355 + 14) =$ **1.443**

Costo totale = 10.350 + 1.443 = 11.793