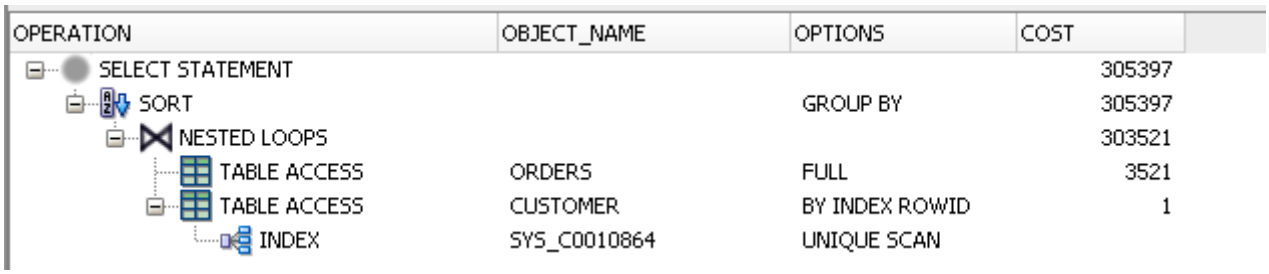
```
create table TIPICORSI (
T_IDCorso int,
T_Nome varchar2(100),
T_Livello int,
T_EtaMin int,
T_EtaMax int,
T_MinPartecipanti int,
primary key (T_IDCorso));

create table ALLIEVI(
A_IDAllievo int,
A_Nome varchar2(100),
A_Eta int,
A_Livello int,
A_SettimanaRichiesta int,
primary key (A_IDAllievo)
);

create table ASSEGNAMENTI(
AS_Corso int,
AS_Allievo int,
primary key (AS_Corso, AS_Allievo),
foreign key (AS_Corso) references TIPICORSI(T_IDCorso),
foreign key (AS_Allievo) references ALLIEVI(A_IDAllievo));


create or replace procedure Assegna(vSettimana int) IS
--cursore
cursor cCorsi is
   select * from TIPICORSI
   order by T_Livello,T_EtaMin;

cursor cAllievi(iLivello int,iEtaMin int,iEtaMax int) is
   select * from ALLIEVI
   where A_SettimanaRichiesta=vSettimana and A_Eta>=iEtaMin and A_Eta<=iEtaMax and
   A_Livello=iLivello;

vNumAllievi int;

begin
--primo cursore
FOR vCorsi IN cCorsi
LOOP
  select count(*) into vNumAllievi from ALLIEVI
   where A_SettimanaRichiesta=vSettimana and A_Eta>=vCorsi.T_EtaMin and
   A_Eta<=vCorsi.T_EtaMax and A_Livello=vCorsi.T_Livello;

  if (vNumAllievi>vCorsi.T_MinPartecipanti) then
    --secondo cursore
    FOR vAllievi IN cAllievi(vCorsi.T_Livello, vCorsi.T_EtaMin,vCorsi.T_EtaMax)
    LOOP
      INSERT INTO ASSEGNAMENTI VALUES (vCorsi.T_IDCorso,vAllievi.A_IDAllievo);
     DBMS_OUTPUT.PUT_LINE('Allievo inserito: ' || vAllievi.A_IDAllievo ||  'in:' ||
     vCorsi.T_IDCorso);
    END LOOP;
 end if;
END LOOP;
end;
```



OPERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			305397
SORT		GROUP BY	305397
NESTED LOOPS			303521
TABLE ACCESS	ORDERS	FULL	3521
TABLE ACCESS	CUSTOMER	BY INDEX ROWID	1
INDEX	SYS_C0010864	UNIQUE SCAN	

Si ricorda nuovamente a tutti gli studenti che il quesito "si disegni l'albero di esecuzione" non significa riportare sul foglio lo schema precedente già prodotto da Oracle, ma tradurlo in operazioni dell'algebra relazionale estesa (es Slide 6 -Ottimizzazione) indicando inoltre le tecniche di join e di accesso a tabella utilizzate. Questi esercizi sono stati ripetutamente svolti a lezione.

In quiesta soluzione la cardinalità delle relazioni è presa dalle statistiche del DBMS. La soluzione con i dati riportati nello schema logico del TPC (NR_C = 150.000, NR_O 1.500.000) è stata valutata come corretta

$NP_C = \lceil 150.631 \times 158 / (4096 \times 0{,}69) \rceil = 8.421$
$NP_O = \lceil 1.500.000 \times 109 / (4096 \times 0{,}69) \rceil = 57.851$
Sel(`o_orderpriority = '1-URGENT'= '1-URGENT'`) = 1/5

Si accede all'indice su C_CUSTKEY (SYS_C0010864) che è la chiave primaria di CUSTOMER al fine di recuperare il record relativi a un valore di chiave (fissato nella tabella esterna). L'operazione è ripetuta per tutti gli ordini selezionati dal predicato su `O_ORDERPRIORITY`

$NL_{C\_CUSTKEY} = \lceil (150.631 \times 4 + 150.631 \times 4) / (4.096 \times 0{,}69) \rceil = 427$
Costo di accesso mediante l'indice su C_CUSTKEY:
$1 + \lceil 1/150.631 \times 427 \rceil + \lceil 1/150.631 \times 8421 \rceil = 1+1+1 = 3$

Costo(Nested Loop O-C) = $57.851 + \lceil 1/5 \times 1.500.000 \rceil \times 3 =$ **957.851**

Il numero di tuple in risultato al join è pari alla cardinalità di ORDINI a cui è applicato il filtro di selezione

$NT_{O+C} = NT_O \times$ Sel(`o_orderpriority = '1-URGENT'`) = $\lceil 1.500.000 / 5 \rceil = 300.000$
$NP_{P+PS} = \lceil 300.000 \times (158+109) / (4096 \times 0{,}69) \rceil = 28.342$

Costo (GB)=$2 \times 28.342 \times (\lceil \log_{100}(28.342) \rceil + 1) = 2 \times 28.342 \times 4 =$ **226.736**

Costo totale = **957.851+ 226.736 = 1.184.587**