1-ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ 3NF

Ο αρχικός πίνακας με τα απαραίτητα γνωρίσματα είναι ο εξής:

booking_id
booking_date
pay_id
payment_mean
customer_surname
customer_name
customer_id
customer_phone
employee_id
employee_surname
employee_name
camp_id
camp_name
total_positions
position_num
position_categ
position_range
position_cost
date_start
date_end
number_people

Σημείωση:
Στον πίνακα δεν έχουμε συμπεριλάβει το κόστος(Ευρώ) ,ούτε το συνολικό κόστος. Είναι και τα δύο παραγόμενα γνωρίσματα και δημιουργούν το καθένα συγκεκριμένες δυσκολίες. Το πρώτο για να υπολογιστεί και για να βρίσκεται στη βάση δεδομένων απαιτεί πράξεις ανάμεσα σε διαφορετικά fields στους πίνακες (φαίνεται εν συνεχεία όταν φτάσουμε στην μορφή 3NF) ,γεγονός που για να

αντιμετωπιστεί χρειάζεται τη δημιουργία function και ο υπολογισμός καθίσταται δαπανηρός. Το δεύτερο είναι ,επίσης ,κοστοβόρο και μπορεί ούτως ή άλλως να υπολογιστεί αλγοριθμικά. Επιπροσθέτως και τα δύο γνωρίσματα έχουν ένα συγκεκριμένο τρόπο υπολογισμού χωρίς αυτό να σημαίνει πως μακροπρόθεσμα δε θα αλλάξει αυτό (π.χ. εκπτώσεις στις κρατήσεις) και έτσι θα χρειάζεται να αλλάξουν οι υπολογισμοί μας.

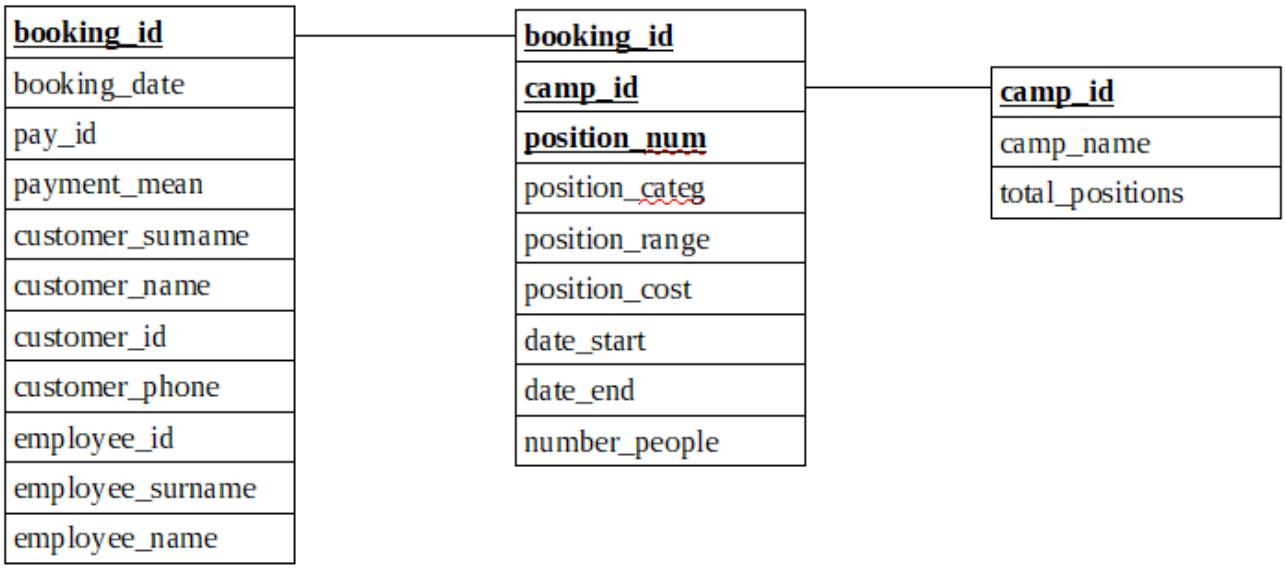
Η μορφή 1NF πρέπει να έχει ατομικές τιμές και το κλειδί που προσδιορίζει τη σχέση είναι (booking_id,camp_id,position_num). Έτσι προκύπτει ο πίνακας:

booking_id
booking_date
pay_id
payment_mean
customer_surname
customer_name
customer_id
customer_phone
employee_id
employee_surname
employee_name
camp_id
camp_name
total_positions
position_num
position_categ
position_range
position_cost
date_start
date_end
number_people

Η μορφή 2NF χρειάζεται κάθε μη πρωτεύον γνώρισμα να εξαρτάται πλήρως από το υποψήφιο κλειδί. Στον παραπάνω πίνακα έχουμε κάποιες εξαρτήσεις που χαλάνε την 2NF :
1) booking_id→ booking_date, pay_id, payment_mean, customer_surname, customer_id, phone, employee_id, employee_surname, employee_name
2)camp_id → camp_name,total_positions

Η σχέση που δε χαλάει την 2NF:
booking_id, camp_id, position_num→ position_categ, position_range, position_cost,date_start,date_end, number_people

Έτσι,έχουμε το εξής σπάσιμο του πίνακα:

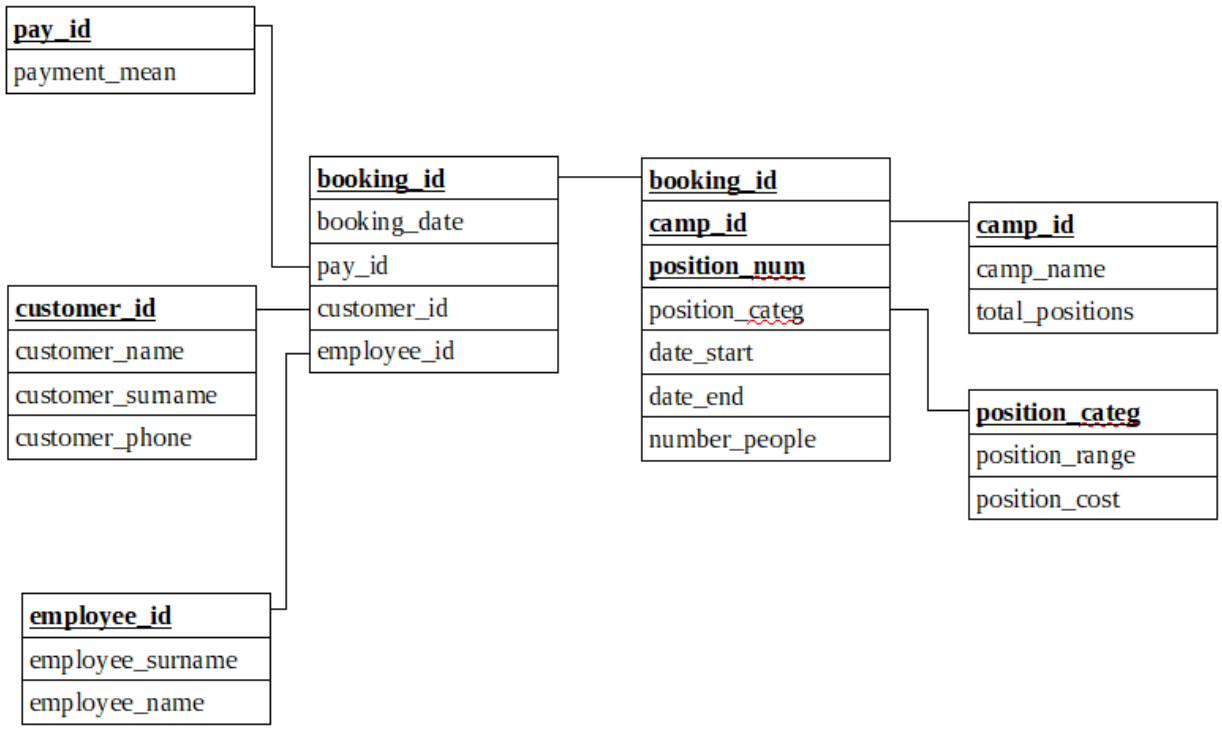


Η μορφή 3NF χρειάζεται κάθε μη πρωτεύον γνώρισμα να εξαρτάται μόνο και μόνο από ολόκληρο το κλειδί. Στους παραπάνω πίνακες έχουμε κάποιες εξαρτήσεις που χαλάνε την 3NF:
1) pay_id → payment_mean

2) customer_id → customer_name, customer_surname, customer_phone
3)employee_id → employee_surname, employee_name
4) position_categ → position_range, position_cost

Και οι σχέσεις που δε χαλάνε την 3NF:
1) booking_id → booking_date, pay_id, customer_id, employee_id
2) camp_id → camp_name, total_positions
3)booking_id, camp_id, position_num → position_categ, date_start, date_end, number_people

Έτσι, οι πίνακες σπάνε ως εξής:



2-Αξιολόγηση των δύο τελευταίων queries με ή χωρίς indexes.

Για ερώτημα d:

- Χωρίς δείκτες:

Ο χρόνος:

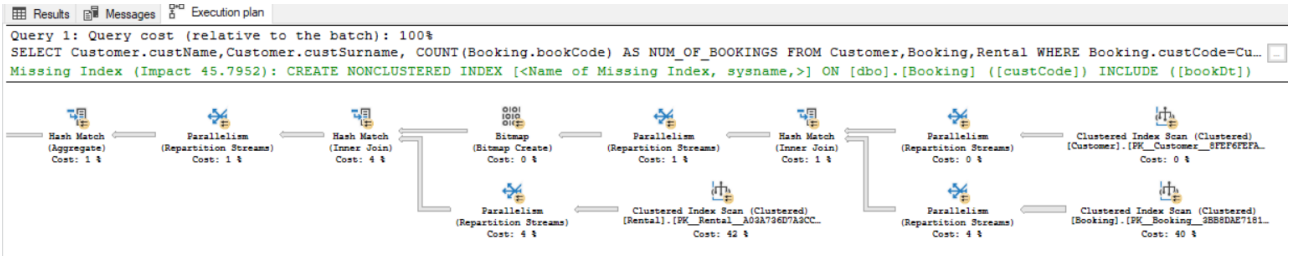
```
SELECT Customer.custName,Customer.custSurname, COUNT(Booking.bookCode) AS NUM_OF_BOOKINGS FROM Customer,Booking,Rental
WHERE Booking.custCode=Customer.custCode AND Rental.bookCode=Booking.bookCode AND YEAR(Booking.bookDt)=2000
GROUP BY Booking.custCode,Customer.custName,Customer.custSurname
ORDER BY Customer.custSurname
```

100 %

Results | Messages | Execution plan

```
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 30 ms, elapsed time = 30 ms.

(4839 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 471 ms,  elapsed time = 329 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
```

Actual Execution Plan:



- Με δείκτες:

Ο χρόνος:

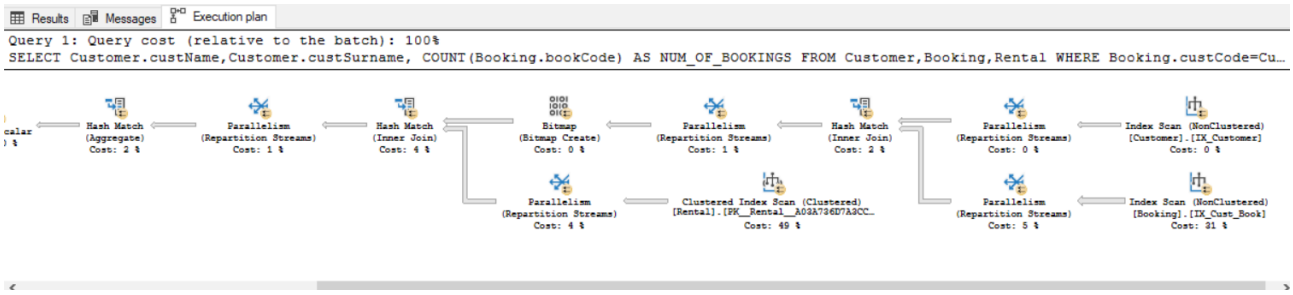
```
SELECT Customer.custName,Customer.custSurname, COUNT(Booking.bookCode) AS NUM_OF_BOOKINGS FROM Customer,Booking,Rental
WHERE Booking.custCode=Customer.custCode AND Rental.bookCode=Booking.bookCode AND YEAR(Booking.bookDt)=2000
GROUP BY Booking.custCode,Customer.custName,Customer.custSurname
ORDER BY Customer.custSurname
```

0 %

Results | Messages | Execution plan

```
(4839 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 453 ms,  elapsed time = 228 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
```

Actual Execution Plan:

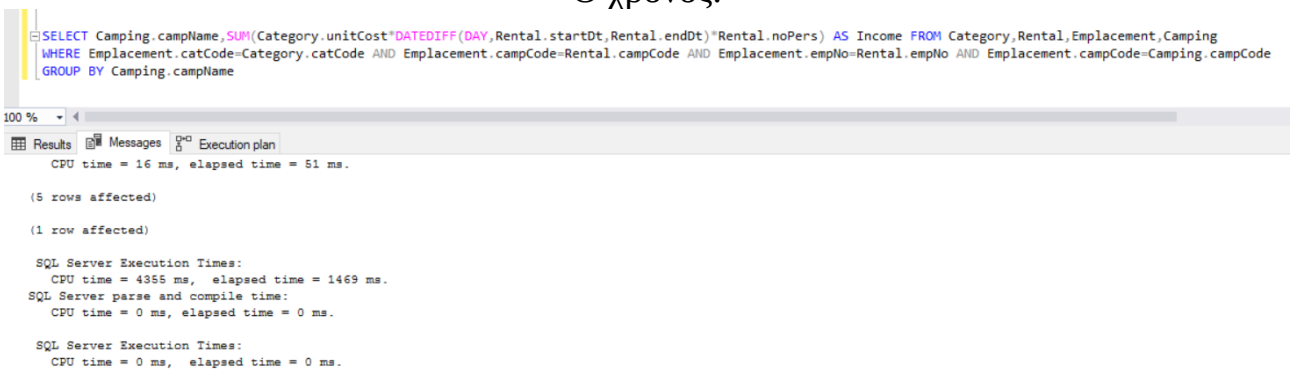


Άρα, στο ερώτημα d συμφέρει χρονικά η δημιουργία των δεικτών IX_Customer και IX_Cust_Book ,αφού ο χρόνος που παρήλθε για την εκτέλεση χωρίς τους δείκτες είναι 329ms ,ενώ με δείκτες 228ms*(και ο χρόνος της CPU χωρίς:471ms ,με:453ms)*. Επίσης,παρατηρώ πως όταν έχω τους δείκτες ,δεν προτιμούνται οι default αλλά αυτοί που δημιούργησα.

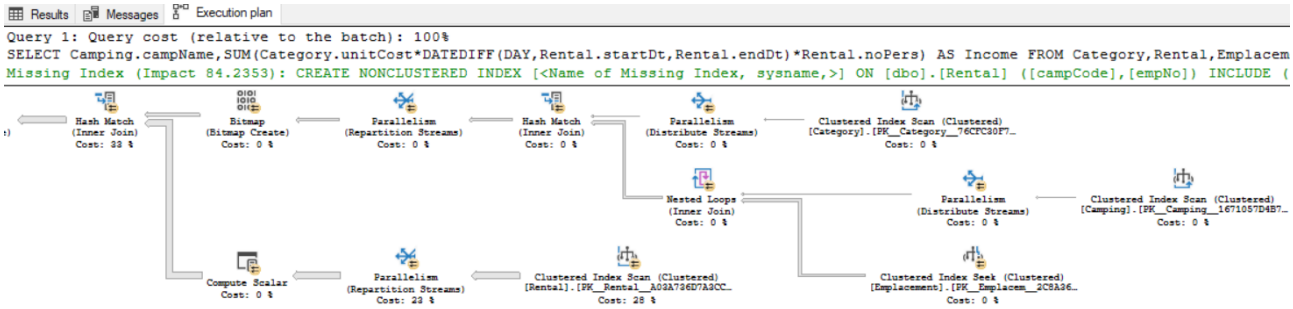
Για ερώτημα e:

- Χωρίς δείκτες:

Ο χρόνος:



Actual Execution Plan:



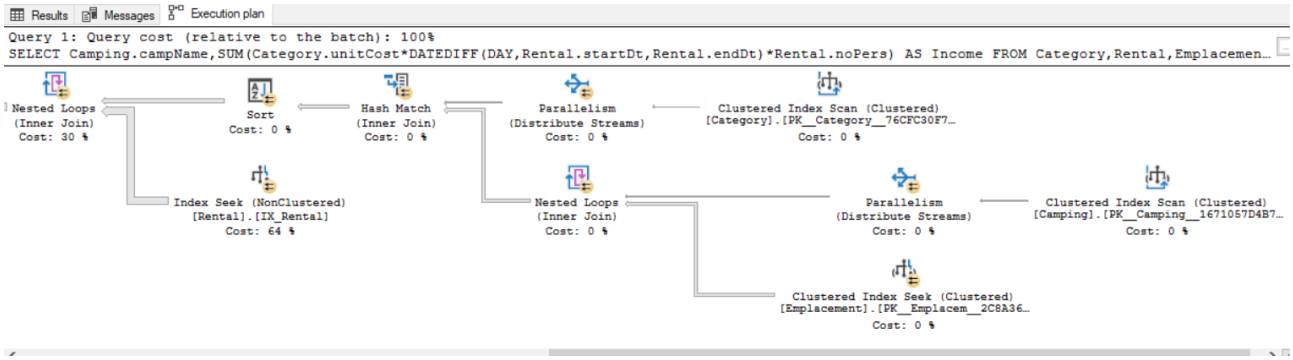
- Με δείκτες:

Ο χρόνος:

```
CREATE INDEX IX_Rental ON Rental(campCode,empNo) INCLUDE (startDt,endDt,noPers)


SELECT Camping.campName,SUM(Category.unitCost*DATEDIFF(DAY,Rental.startDt,Rental.endDt)*Rental.noPers) AS Income FROM Category,Rental,Emplacement,Camping
WHERE Emplacement.catCode=Category.catCode AND Emplacement.campCode=Rental.campCode AND Emplacement.empNo=Rental.empNo AND Emplacement.campCode=Camping.campCode
GROUP BY Camping.campName
```

```
Results  Messages  Execution plan
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 31 ms, elapsed time = 39 ms.

(5 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 1563 ms,  elapsed time = 938 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
```

Actual Execution Plan:



Άρα, στο ερώτημα e συμφέρει χρονικά η δημιουργία του δείκτη IX_Rental,αφού ο χρόνος που παρήλθε για την εκτέλεση χωρίς το δείκτη είναι 1469ms ,ενώ με δείκτη 938ms *(και ο χρόνος της CPU χωρίς:4355ms ,με:1563ms)*. Επίσης,παρατηρώ πως ο δείκτης συμβάλλει σημαντικά περισσότερο από τον default.