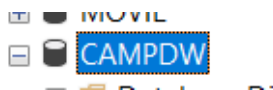




### Ζήτημα Πρώτο

Δημιουργούμε την βάση δεδομένων και τον πίνακα που μας ζητείται και φορτώνουμε τα δεδομένα με τις παρακάτω εντολές:



```
1 CREATE TABLE campdata (  
2     custID int,  
3     fname varchar(30),  
4     lname varchar(30),  
5     cID int,  
6     country varchar(30),  
7     bookID int,  
8     bookDate date,  
9     campCode char(3),  
10    campName varchar(50),  
11    empno int,  
12    catCode char(1),  
13    category varchar(20),  
14    unitCost numeric(4,2),  
15    startDate date,  
16    overnights int,  
17    persons int  
18 );  
19  
20  
21 BULK INSERT campdata  
22 FROM 'C:\Users\Μαρίνα Σαμ\Desktop\CAMPDATA.TXT'  
23 WITH (FIRSTROW =2, FIELDTERMINATOR='|', ROWTERMINATOR = '\n');
```

100 %

Messages

(1311107 rows affected)

Completion time: 2021-05-24T20:09:21.4120546+03:00

2. Παρακάτω περιγράφονται τα tables που υλοποιήσαμε.

Στα dimension tables περιέχονται πεδία όπως ονόματα, διευθύνσεις, πεδία που αναφέρονται σε εξαρτήσεις με άλλους πίνακες. Τέλος, τα primary keys τους αποτελούν τα foreign keys του fact table. Στο Fact Table περιέχονται όλα τα κλειδιά των dimension tables καθώς και διάφορα αριθμητικά πεδία.

```
create table customers  
(custID int primary key,  
fname varchar(30),  
lname varchar(30),  
cID int,  
country varchar(30)  
);
```

```
create table camp  
(campCode char(3) primary key,  
campName varchar(50),  
bookID int,  
startDate date  
);
```

```
create table category  
(catCode char(1) primary key,  
category varchar(20),  
);
```

```
create table timeinfo  
(time_key datetime primary key,  
t_year int,  
t_month int,  
t_dayofmonth int,  
t_quarter int,  
t_week int,  
t_dayofyear int,  
t_dayofweek int  
);
```

```
create table booking  
( custID int,  
    campCode char(3),  
    catCode char(1),  
    time_key datetime,  
    unitCost numeric(4,2),  
    overnights int,  
    persons int,
```

```
primary key(custID,campCode,catCode,time_key),  
foreign key (custID) references customers(custID),  
foreign key (campCode) references camp(campCode),  
foreign key (catCode) references category(catCode),  
foreign key (time_key) references timeinfo(time_key)  
);
```

### 3. Εισαγωγή των δεδομένων:

Για τον πίνακα customers:

```
insert into customers select distinct custID,fname,lname,cID,country from campdata group by  
custID,fname,lname,cID,country
```

Για τον πίνακα camp:

```
insert into camp select distinct campCode,campName,max(bookID),max(startDate) from  
campdata group by campCode,campName
```

Σε αυτό το σημείο πρέπει να σημειώσουμε πως η συνάρτηση max τοποθετήθηκε γιατί το dataset περιείχε διπλότυπα και το query δε μπορούσε να εκτελεστεί.

Για τον πίνακα category:

```
insert into category select distinct catCode,category from campdata group by  
catCode,category
```

Για τον πίνακα timeinfo:

```
set datefirst 1;  
insert into timeinfo select distinct startDate, datepart(year, startDate), datepart(month,  
startDate), datepart(day,startDate),  
datepart(quarter,startDate),  
datepart(week,startDate),datepart(dayofyear,startDate),datepart(dw,startDate)  
from campdata order by startDate
```

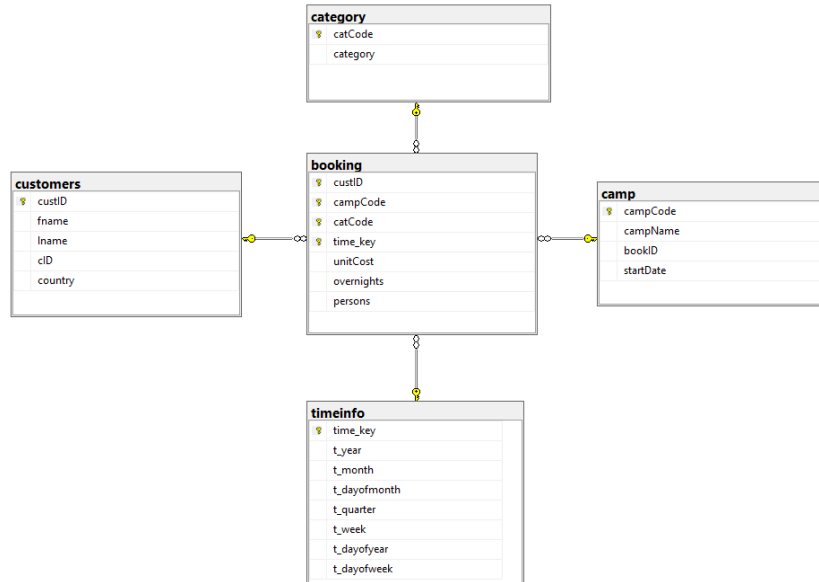
Για τον πίνακα booking:

```
insert into booking  
select custID, campCode,catCode, startDate, sum(unitCost), sum(overnights), sum(persons)  
from campdata  
group by custID, campCode, catCode,startDate
```

Το συνολικό κόστος για κάθε πελάτη υπολογίζεται μέσω της παρακάτω εντολής:

```
select custID,  
overnights * persons * CONVERT(numeric(4,2), unitCost) as totalCost  
from campdata  
order by custID
```

4. Το σχήμα της αποθήκης φαίνεται παρακάτω:



## Ζήτημα Δεύτερο

1.

```
select TOP 100 country,fname,lname,  
overnights * persons * CONVERT(numeric(4,2), unitCost) as totalCost  
from camp,customers,booking  
order by fname,lname,totalCost desc
```

The screenshot shows a SQL query being executed in SQL Server Enterprise Manager. The query is: `select TOP 100 country,fname,lname, overnights * persons * CONVERT(numeric(4,2), unitCost) as totalCost from camp,customers,booking order by totalCost DESC`. The results are displayed in a table with columns: country, fname, lname, and totalCost. The table contains 100 rows of data, all from Belgium, sorted by totalCost in descending order. The status bar at the bottom indicates the query was executed successfully.

	country	fname	lname	totalCost
1	Belgium	Aarika	JOHNSON	4455.00
2	Belgium	Abagael	WILLIAMS	4455.00
3	Belgium	Abigail	JONES	4455.00
4	Belgium	Abbe	BROWN	4455.00
5	Belgium	Abbey	DAVIS	4455.00
6	Belgium	Abbi	MILLER	4455.00
7	Belgium	Abbie	WILSON	4455.00
8	Belgium	Abby	MOORE	4455.00
9	Belgium	Abbye	TAYLOR	4455.00
10	Belgium	Abigail	ANDERSON	4455.00
11	Belgium	Abigail	THOMAS	4455.00
12	Belgium	Abigale	JACKSON	4455.00
13	Belgium	Abra	WHITE	4455.00
14	Belgium	Ada	HARRIS	4455.00
15	Belgium	Adah	MARTIN	4455.00
16	Belgium	Adaline	THOMPSON	4455.00
17	Belgium	Aden	GARRIA	4455.00
18	Belgium	Adara	MARTINEZ	4455.00
19	Belgium	Adda	ROBINSON	4455.00
20	Belgium	Addi	CLARK	4455.00
21	Belgium	Addia	RODRIGUEZ	4455.00
22	Belgium	Addie	LEWIS	4455.00
23	Belgium	Addy	LEE	4455.00
24	Belgium	Adel	WALKER	4455.00
25	Belgium	Adela	HALL	4455.00

Η εντολή convert χρησιμοποιήθηκε γιατί το πεδίο unicast είναι numeric και δεν μπορούσε να πολλαπλασιαστεί με int.

2.

```
select camp.campCode,category.catCode,category,timeinfo.t_year,  
       sum(overnights * persons * CONVERT(numeric(4,2), unitCost)) as totalCost  
from camp,booking,category,timeinfo  
where t_year = '2000'  
group by camp.campCode,category.catCode,category.category,t_year
```

The screenshot shows a SQL query window with the following text:

```
12 select camp.campCode,category.catCode,category,timeinfo.t_year,  
13     sum(overnights * persons * CONVERT(numeric(4,2), unitCost)) as totalCost  
14 from camp,booking,category,timeinfo  
15 where t_year = '2000'  
16 group by camp.campCode,category.catCode,category.category,t_year  
17
```

Below the query window, the 'Results' tab is active, displaying a table with 5 columns: campCode, catCode, category, t\_year, and totalCost. The table contains 15 rows of data.

campCode	catCode	category	t_year	totalCost
APL	A	Tent	2000	55544844075.00
DIS	A	Tent	2000	55544844075.00
ELB	A	Tent	2000	55544844075.00
KIS	A	Tent	2000	55544844075.00
ROS	A	Tent	2000	55544844075.00
APL	B	Caravan	2000	55544844075.00
DIS	B	Caravan	2000	55544844075.00
ELB	B	Caravan	2000	55544844075.00
KIS	B	Caravan	2000	55544844075.00
ROS	B	Caravan	2000	55544844075.00
APL	C	Camper Van	2000	55544844075.00
DIS	C	Camper Van	2000	55544844075.00
ELB	C	Camper Van	2000	55544844075.00
KIS	C	Camper Van	2000	55544844075.00
ROS	C	Camper Van	2000	55544844075.00

At the bottom of the window, a status bar indicates: 'Query executed successfully. LAPTOP-PHIU6JS3 (15.0 RTM) LAPTOP-PHIU6JS3\Mapiva... second 00:00:00 15 rows'.

3.

```
SELECT DISTINCT timeinfo.t_month, camp.campCode,  
               sum(overnights * persons * CONVERT(decimal(4,2), unitCost)) as totalCost  
FROM camp,booking,timeinfo  
WHERE t_year = '2018'  
GROUP BY timeinfo.t_month, camp.campCode
```

The screenshot shows a SQL query window with the following text:

```
18 SELECT DISTINCT timeinfo.t_month, camp.campCode,  
19     sum(overnights * persons * CONVERT(numeric(4,2), unitCost)) as totalCost  
20 FROM camp,booking,timeinfo  
21 WHERE t_year = '2018'  
22 GROUP BY timeinfo.t_month, camp.campCode  
23
```

Below the query window, the 'Results' tab is active, displaying a table with 3 columns: t\_month, campCode, and totalCost. The table contains 36 rows of data.

t_month	campCode	totalCost
9	DIS	4565329650.00
10	DIS	4717507305.00
11	DIS	4565329650.00
12	DIS	4717507305.00
1	ELB	4717507305.00
2	ELB	4260974340.00
3	ELB	4717507305.00
4	ELB	4565329650.00
5	ELB	4717507305.00
6	ELB	4565329650.00
7	ELB	4717507305.00
8	ELB	4717507305.00
9	ELB	4565329650.00
10	ELB	4717507305.00
11	ELB	4565329650.00
12	ELB	4717507305.00
1	KIS	4717507305.00
2	KIS	4260974340.00
3	KIS	4717507305.00

At the bottom of the window, a status bar indicates: 'Query executed successfully. LAPTOP-PHIU6JS3 (15.0 RTM) LAPTOP-PHIU6JS3\Mapiva... second 00:00:00 60 rows'.

4.

```
select timeinfo.t_year,camp.campCode,category.catCode,sum(cast (persons as bigint))
from timeinfo,camp,category,booking
group by rollup (camp.campCode,category.catCode,timeinfo.t_year)
```

The screenshot shows a SQL query executed successfully, displaying a rollup of persons by camp and category over time. The query is as follows:

```
select timeinfo.t_year,camp.campCode,category.catCode,sum(cast (persons as bigint))
from timeinfo,camp,category,booking
group by rollup (camp.campCode,category.catCode,timeinfo.t_year)
```

The results table has four columns: t\_year, campCode, catCode, and a column with no name. The data is grouped by campCode and category, and then by year. The status bar indicates the query was executed successfully and returned 771 rows.

t_year	campCode	catCode	(No column name)
1970	APL	A	1047971200
1971	APL	A	1195342150
1972	APL	A	1195342150
1973	APL	A	1195342150
1974	APL	A	1195342150
1975	APL	A	1195342150
1976	APL	A	1195342150
1977	APL	A	1195342150
1978	APL	A	1195342150
1979	APL	A	1195342150
1980	APL	A	1195342150
1981	APL	A	1195342150
1982	APL	A	1195342150
1983	APL	A	1195342150
1984	APL	A	1195342150
1985	APL	A	1195342150
1986	APL	A	1195342150
1987	APL	A	1195342150
1988	APL	A	1195342150
1989	APL	A	1195342150
1990	APL	A	1195342150
1991	APL	A	1195342150
1992	APL	A	1195342150
1993	APL	A	1195342150
1994	APL	A	1195342150
1995	APL	A	1195342150
1996	APL	A	1195342150
1997	APL	A	1195342150
1998	APL	A	1195342150

5.

```
create view temp as
select booking.campCode,booking.time_key,sum(booking.persons) as totalPersons
from booking
join camp on booking.campCode = camp.campCode
group by booking.campCode,booking.time_key

select eighteen.campCode
from temp as eighteen,temp as seventeen
where eighteen.campCode = seventeen.campCode
and eighteen.time_key = '2018' and seventeen.time_key = '2017'
and eighteen.totalPersons > seventeen.totalPersons
```

The screenshot shows a SQL query executed successfully, displaying the creation of a temporary view and a comparison of total persons between 2017 and 2018. The query is as follows:

```
create view temp as
select booking.campCode,booking.time_key,sum(booking.persons) as totalPersons
from booking
join camp on booking.campCode = camp.campCode
group by booking.campCode,booking.time_key

select eighteen.campCode
from temp as eighteen,temp as seventeen
where eighteen.campCode = seventeen.campCode
and eighteen.time_key = '2018' and seventeen.time_key = '2017'
and eighteen.totalPersons > seventeen.totalPersons
```

The results table has one column: campCode. The data shows two rows: APL and ELB. The status bar indicates the query was executed successfully and returned 2 rows.

campCode
APL
ELB

### Ζήτημα Τρίτο

α. Κάθε κελί του κύβου περιέχει το έτος, την κατηγορία της θέσης, τον κωδικό της κράτησης, καθώς και τον σύνολο του κόστους διανυκτέρευσης ανά άτομο.

```
select t_year, category.catCode, bookID, sum(unitCost)
from timeinfo, category, booking, camp
group by cube (timeinfo.t_year, category.catCode, bookID)
order by timeinfo.t_year desc, category.catCode asc
```

t_year	catCode	bookID	(No column name)
2019	NULL	630059	3658030050.00
2019	NULL	885920	3658030050.00
2019	NULL	1162982	3658030050.00
2019	NULL	1277808	3658030050.00
2019	NULL	2646220	3658030050.00
2019	NULL	NULL	16290150250.00
2019	A	NULL	6096716750.00
2019	A	2646220	1219343350.00
2019	A	630059	1219343350.00
2019	A	885920	1219343350.00
2019	A	1162982	1219343350.00
2019	A	1277808	1219343350.00
2019	B	1162982	1219343350.00
2019	B	885920	1219343350.00
2019	B	630059	1219343350.00
2019	B	2646220	1219343350.00
2019	B	1277808	1219343350.00
2019	B	NULL	6096716750.00
2019	C	NULL	6096716750.00
2019	C	1277808	1219343350.00
2019	C	2646220	1219343350.00
2019	C	630059	1219343350.00
2019	C	1162982	1219343350.00
2019	C	885920	1219343350.00
2018	NULL	630059	18805365750.00
2018	NULL	885920	18805365750.00
2018	NULL	1162982	18805365750.00
2018	NULL	1277808	18805365750.00
2018	NULL	2646220	18805365750.00
2018	NULL	NULL	94026828750.00
2018	A	NULL	31342276250.00
2018	A	2646220	6268455250.00

Επαληθεύουμε ότι έγιναν σωστά τα group by ψάχνοντας στα αποτελέσματα που έδωσε το query, όλους τους συνδυασμούς του NULL.

β. Έχουμε 3 columns, άρα ο κύβος θα επιστρέψει 8 διαφορετικά group by.

```
(select t_year, category.catCode, bookID, sum(unitCost)
from timeinfo, category, booking, camp
group by timeinfo.t_year, category.catCode, bookID)
```

```
(select null, category.catCode, bookID, sum(unitCost)
from timeinfo, category, booking, camp
group by category.catCode, bookID)
```

```
(select t_year, null, bookID, sum(unitCost)
from timeinfo, category, booking, camp
group by timeinfo.t_year, bookID)
```

```
(select t_year, category.catCode, null, sum(unitCost)
from timeinfo, category, booking, camp
group by timeinfo.t_year, category.catCode)
)
```

```
(select t_year, null, null, sum(unitCost)
from timeinfo, category, booking, camp
group by timeinfo.t_year)
```

```
(select null, category.catCode, null, sum(unitCost)
from timeinfo, category, booking, camp
group by category.catCode)
```

```
(select null, null, bookID, sum(unitCost)
from timeinfo, category, booking, camp
group by bookID)
```

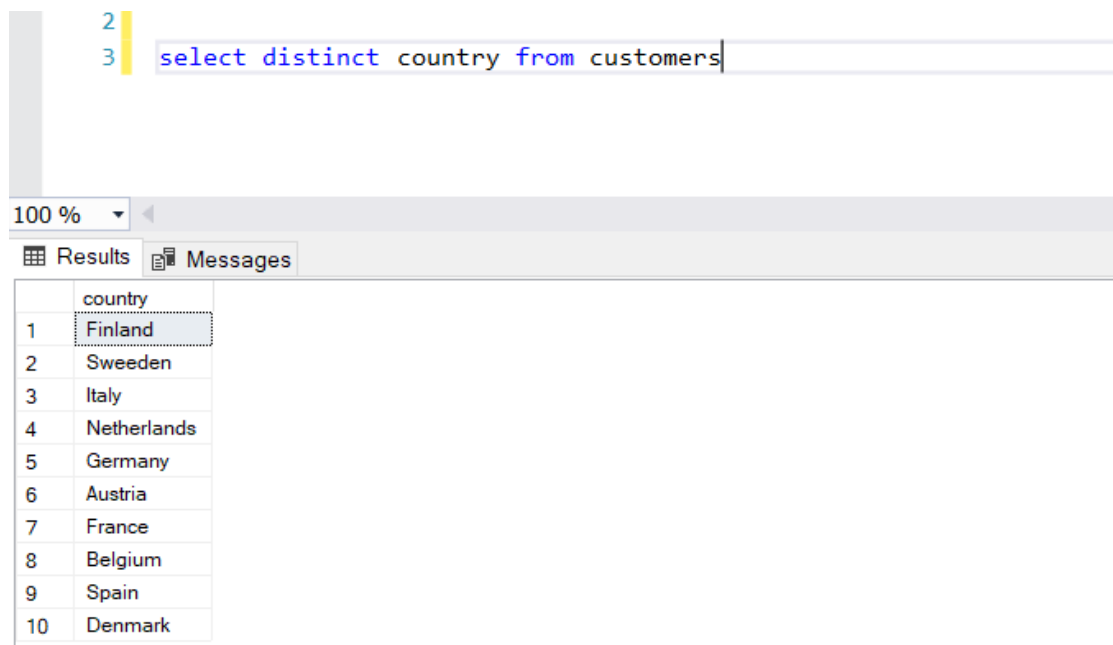
```
(select null, null, null, sum(unitCost)
from timeinfo, category, booking, camp
group by none)
```

### Ζήτημα Τέταρτο

Στα Bitmap ευρετήρια, κάθε διαφορετική τιμή ενός συγκεκριμένου πεδίου, αντιστοιχίζεται με ένα bitmap, 0 ή 1. Στο συγκεκριμένο παράδειγμα, οι χώρες είναι αρκετές, οπότε θα μπορούσαμε να χρησιμοποιήσουμε Bitmap Join Index.

Αρχικά εκτελούμε το παρακάτω query για να πάρουμε τις διαφορετικές χώρες που περιέχονται στο dataset.

```
select distinct country from customers
```



The screenshot shows a database query interface. At the top, the query `select distinct country from customers` is entered. Below the query, there are tabs for 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with 10 rows of distinct countries. The table has a single column named 'country'.

	country
1	Finland
2	Sweeden
3	Italy
4	Netherlands
5	Germany
6	Austria
7	France
8	Belgium
9	Spain
10	Denmark



Οι χώρες είναι 10 και οι ενδεικτικές γραμμές που μας δίνονται είναι 8. Ο κανόνας λέει ότι γράφουμε 1 στην χώρα που περιέχεται στην εκάστοτε εγγραφή και 0 αλλιώς. Συνεπώς, καταλήγουμε στον παρακάτω πίνακα που αναπαριστά την μορφή και το περιεχόμενο του ευρετηρίου.

#rowID	Finland	Sweden	Italy	Netherlands	Germany	Austria	France	Belgium	Spain	Denmark
69	0	0	0	0	0	0	0	0	1	0
70	0	0	0	0	0	0	0	0	1	0
71	0	0	0	0	0	0	0	0	0	1
72	0	0	0	0	0	1	0	0	0	0
73	0	0	0	0	0	1	0	0	0	0
74	0	0	0	0	1	0	0	0	0	0
75	0	0	0	0	1	0	0	0	0	0
76	0	0	0	0	1	0	0	0	0	0