

SQL projektfeladat dokumentáció

Bookings tábla elkészítése¹

Kezdeti adatok importálása

Az európai városoknevek és az országok ISO kód listái külső forrásból² lettek importálva két segédtáblába.

Countries tábla³ :

```
CREATE TABLE Countries(  
    [asciiiname] VARCHAR(255) NOT NULL PRIMARY KEY,  
    [country] VARCHAR(255) NOT NULL,  
    [population] int NOT NULL  
)  
BULK INSERT Countries  
FROM '<Elérési Útvonal>\countries.csv'  
WITH (FORMAT='CSV',  
    FIRSTROW=2,  
    FIELDTERMINATOR = '|',  
    ROWTERMINATOR = '0x0a')
```

Iso2Codes tábla⁴ :

```
CREATE TABLE Iso2Codes(  
    [Code] VARCHAR(2) NOT NULL,  
    [Name] VARCHAR(255) NOT NULL PRIMARY KEY  
)  
BULK INSERT Iso2Codes  
FROM '<Elérési Útvonal>\iso2.csv'  
WITH (FORMAT='CSV',  
    FIRSTROW=2,  
    FIELDTERMINATOR = '|',  
    ROWTERMINATOR = '0x0a')
```

Kiegészítő adathalmazok létrehozása

A kiegészítő adatok ideiglenes táblában vannak tárolva a felhasználásukig, az átláthatóság kedvéért.

Az ISO kódok hozzárendelése csak az európai országokhoz:

```
SELECT c.asciiiname, ic.Code  
INTO #euCountriesIsos  
FROM Iso2Codes ic  
INNER JOIN Countries c on ic.Name = c.country
```

Városok kiválasztása:

```
CREATE TABLE #selectedCities (  
    ID int NOT NULL IDENTITY(1,1) PRIMARY KEY,  
    asciiiname VARCHAR(255) NOT NULL UNIQUE,  
    Code CHAR(2) NOT NULL  
)  
  
INSERT INTO #selectedCities  
SELECT tmp.asciiiname, tmp.Code  
FROM (  
    SELECT DISTINCT TOP 15 Code, asciiiname  
    FROM #euCountriesIsos eci
```

¹A teljes scriptek a mellékelt SSMS solution-ben találhatóak

²ISO Kódok, Városok

³ helyettesítendő a kívánt fájl helyével

⁴ helyettesítendő a kívánt fájl helyével

```
WHERE (ABS(CAST((CHECKSUM(*) * RAND()) as int)) % 100) < 30) tmp
UNION
(SELECT 'Luton' asciiname, 'GB' Code)
```

A *TABLESAMPLE* utasítás pontatlansága miatt a *WHERE (ABS(CAST((CHECKSUM(*) * RAND()) as int)) % 100) < 30** segítségével veszünk véletlenszerű mintát a *#euCountriesIsos* táblából.

Ezek után generálunk 2500, hatjegyű *CustomerID*-t:

```
WITH
  l0(i) AS (SELECT 0 UNION ALL SELECT 0),
  l1(i) AS (SELECT 0 FROM l0 a, l0 b),
  l2(i) AS (SELECT 0 FROM l1 a, l1 b),
  l3(i) AS (SELECT 0 FROM l2 a, l2 b),
  l4(i) AS (SELECT 0 FROM l3 a, l3 b),
  numbers(i) AS (SELECT ROW_NUMBER() OVER(ORDER BY (SELECT 0)) FROM l4)
SELECT DISTINCT TOP(2500) (CEILING(RAND(CHECKSUM(NEWID()))*899999)+100000) CustomerID
INTO #cids
FROM numbers
ORDER BY CustomerID
```

CustomerID-khoz *CCountry* azonosítókat rendelünk:

```
SELECT cp.CustomerID CustomerID, sc.Code CCountry
INTO #cidPairs
FROM (SELECT cids.CustomerID, (CEILING(RAND(CHECKSUM(NEWID()))*(SELECT COUNT(*) FROM #selectedCities)))
INNER JOIN #selectedCities sc on cp.CountryID = sc.ID
ORDER BY CustomerID
```

Bookings feltöltése

Bookings tábla létrehozása:

```
CREATE TABLE Bookings(
[BookingID] int NOT NULL IDENTITY(1,1) PRIMARY KEY,
[CustomerID] int NOT NULL,
[CCountry] varchar(2) NOT NULL,
[DepartureStation] varchar(30) NOT NULL,
[Date] datetime NOT NULL DEFAULT (GETDATE()),
[Price] money NOT NULL,
[Seats] int NOT NULL
)
```

Egy kezdeti adathalmaz létrehozása:

```
WITH
  l0(i) AS (SELECT 0 UNION ALL SELECT 0),
  l1(i) AS (SELECT 0 FROM l0 a, l0 b),
  l2(i) AS (SELECT 0 FROM l1 a, l1 b),
  l3(i) AS (SELECT 0 FROM l2 a, l2 b),
  numbers(i) AS (SELECT ROW_NUMBER() OVER(ORDER BY (SELECT 0)) FROM l3)
SELECT TOP(1000000)
  cp.CustomerID,
  cp.CCountry,
  sc.asciiname DepartureStation,
  (DATEADD(ms,RAND(CHECKSUM(NEWID())) * 86400000,DATEADD(d,RAND(CHECKSUM(NEWID()))*365,'
  (ROUND(RAND(CHECKSUM(NEWID()))*200+10,2)) Price,
  (CEILING(RAND(CHECKSUM(NEWID()))*50)) Seats
INTO #bookingsAll
from numbers
CROSS APPLY (
  SELECT
    cp.CustomerID,
```

```

cp.CCountry,
(
    SELECT CEILING(RAND(CHECKSUM(NEWID()))*(SELECT COUNT(*)
    FROM #selectedCities))
) DepartureStationID
FROM #cidPairs AS cp
) AS cp
INNER JOIN #selectedCities sc
ON sc.ID = cp.DepartureStationID

```

Az adathalmazból véletlenszerűen kiválasztásra kerül tízezer bejegyzés a végleges *Bookings* táblába:

```

INSERT INTO Bookings
SELECT TOP(10000) *
FROM #bookingsAll
WHERE (ABS(CAST((CHECKSUM(*) * RAND()) as int)) % 100) < 30

```

Indexek létrehozása

BookingID mezőn alapértelmezetten létrejön egy clustered index a *PRIMARY KEY* feltétel miatt Non Clustered Index a *CCountry* és *CustomerID* párokon:

```

CREATE NONCLUSTERED INDEX NC_Bookings_countryCid ON [dbo].[Bookings]
(
    [CCountry] ASC,
    [CustomerID] ASC
)

```

Ezen kívül a feladat lekérdezéseihez a Database Engine Tuning Advisor által ajánlott optimális non clustered indexek:

```

CREATE NONCLUSTERED INDEX NC_Bookings_countryCidDate ON [dbo].[Bookings]
(
    [Date] ASC,
    [CCountry] ASC,
    [CustomerID] ASC
)
INCLUDE([Price])

CREATE NONCLUSTERED INDEX NC_Bookings_countryDateStation ON [dbo].[Bookings]
(
    [DepartureStation] ASC,
    [Date] ASC,
    [CCountry] ASC
)
INCLUDE([Price])

```

Lekérdezések

```
SELECT b.CustomerID, b.CCountry , SUM(b.Price) 'Total Amount'
FROM Bookings b
WHERE b.[Date] >= '2019-05-01' AND b.[Date] < '2019-06-01'
GROUP BY b.CustomerID,b.CCountry
ORDER BY 'Total Amount' DESC
```

2. feladat - 2019 májusi megrendelések csökkenő sorrendben

```
SELECT b.CCountry Country,
       ISNULL(SUM(case when DATEPART(m,b.[Date]) = 4 THEN b.Price END),0) '2019-04',
       ISNULL(SUM(case when DATEPART(m,b.[Date]) = 5 THEN b.Price END),0) '2019-05',
       ISNULL(SUM(case when DATEPART(m,b.[Date]) = 6 THEN b.Price END),0) '2019-06'
FROM Bookings b
WHERE b.DepartureStation = 'Luton' AND b.[Date] >= '2019-04-01' AND b.[Date] < '2019-07-01'
GROUP BY b.CCountry
ORDER BY SUM(b.Price) DESC
```

3. feladat - Második negyedév Luton-ból induló megrendeléseinek összegei

Megvalósítható PIVOT használatával is, ez azonban csak kevésbé rugalmas megjelenítést tesz lehetővé

```
SELECT * FROM
(SELECT b.CCountry Country, FORMAT(b.[Date], 'yyyy-MM') dpart, b.Price
 FROM Bookings b
 WHERE b.DepartureStation = 'Luton' AND b.[Date] >= '2019-04-01' AND b.[Date] < '2019-07-01'
 PIVOT (
 SUM(x.Price)
 FOR dpart IN ([2019-04],[2019-05],[2019-06])
 ) PivotTable;
```

```
SELECT cntQry.CCountry Country,
       COUNT(case when cntQry.buyCnt = 1 THEN 1 ELSE null END) [1],
       COUNT(case when cntQry.buyCnt = 2 THEN 1 ELSE null END) [2],
       COUNT(case when cntQry.buyCnt = 3 THEN 1 ELSE null END) [3],
       COUNT(case when cntQry.buyCnt BETWEEN 4 AND 9 THEN 1 ELSE null END) [4-10],
       COUNT(case when cntQry.buyCnt BETWEEN 10 AND 100 THEN 1 ELSE null END) [10-100]
FROM (
 SELECT COUNT(b.BookingID) buyCnt, b.CustomerID, b.CCountry
 FROM Bookings b
 GROUP BY b.CustomerID,b.CCountry
 ) cntQry
GROUP BY cntQry.CCountry
ORDER BY SUM(cntQry.buyCnt)
```

4. feladat - Régiónkénti ügyfél vásárlási számok kimutatása

6. feladat - Helymegtakarítás az indexek eldobásával

Az indexek méretét több dolog is befolyásolja, így a konkrét tábla és indexek ismerete nélkül nem lehet meghatározni a helymegtakarítást. Befolyásoló szempontok lehetnek például a tábla mérete, a pageek fill-factora vagy indexben használt oszlopok típusai és száma. További nehezen meghatározható tényező a tömörített indexek és LOB-ok tárhely igénye

Az indexek által foglalt hely lekérdezhető az alábbi queryvel[^3]: [^3]: helyettesítendő a keresett tábla nevével

```
SELECT
OBJECT_NAME(i.OBJECT_ID) AS TableName,
i.name AS IndexName,
i.index_id AS IndexID,
8 * SUM(a.used_pages) AS 'Indexsize(KB)'
FROM sys.indexes AS i
JOIN sys.partitions AS p ON p.OBJECT_ID = i.OBJECT_ID AND p.index_id = i.index_id
JOIN sys.allocation_units AS a ON a.container_id = p.partition_id
WHERE OBJECT_NAME(i.OBJECT_ID) = <TÁBLA>
GROUP BY i.OBJECT_ID,i.index_id,i.name
ORDER BY OBJECT_NAME(i.OBJECT_ID),i.index_id
```


random jegyzetek

SNAPSHOT ISOLATION ON tempdb-be az update előtti adatok bekerülnek egy row version számmal p: nincs lock

c: tempdb igénybevétele nő

Read committed Snapshot On olvasni csak commitelt adatokból

Mire érdemes figyelni: több update-nél UPDLock hinttel zárható az updatelt sor kiadott update-eknél van e select uplockkal, vagy hibakezelés (begin try/end try)

(INDEX) 300MB tábla, 2 nonCL, 1 CL index eldobás hatása a méretre (%ban)

[BookingID] int 4b

[CustomerID] int 4b

[CCountry] varchar(2) 4b (2+2)

[DepartureStation] varchar(30) 32b (30+2)

[Date] datetime 8b

[Price] money 8b

[Seats] int 4b

Row req: 70b (28 + [2+2*2+32] + 4 rowHeader)

Clustered a BookingID-n automatikusan

Row req 11b (4+1 rowHeader+6 childID)

NC 2 szűk DepState & CID

Row reqs 43b <- 11b (4+1+6) & 32b (2+30)

NEM számol vele: page veszteség, non-leaf page méret, tömörítés

arányok: 70:54 -> 300M/124*70 -> 164M adat -> 45% csökkenés

Tartalomjegyzék

Bookings tábla elkészítése[^1]	2
Kezdeti adatok importálása	2
Kiegészítő adathalmazok létrehozása	2
Bookings feltöltése	3