SQL projektfeladat dokumentáció

Bookings tábla elkészítése[^1]

[^1]: Részletes scriptek a mellékelt SSMS solution-ben találhatóak

Kezdeti adatok importálása

Az európai városoknevek és az országok ISO kód listái külső forrásból[^2] lettek importálva két segédtáblába.

dasd [^2]: ISO Kódok, Városok

Countries tábla:

```
CREATE TABLE Countries(
    [asciiname] 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 tábla:

```
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.asciiname, 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,
asciiname VARCHAR(255) NOT NULL UNIQUE,
Code CHAR(2) NOT NULL
)

INSERT INTO #selectedCities
SELECT tmp.asciiname, tmp.Code
FROM (
    SELECT DISTINCT TOP 15 Code, asciiname
    FROM #euCountriesIsos eci
    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) Customer
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(*) FRO
INNER JOIN #selectedCities sc on cp.CountryID = sc.ID
ORDER BY CustomerID
```

Bookings feltöltése

(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