**Adatbázis-szerverek**

Név: Tábi Eszter

Neptunkód: F02F8P

Szakirány: Mérnökinformatikus

Github-link: https://github.com/Eszto94/DataBase

**Adatbázis tervezés**

Az Online Könyvesbolt adatbázis tervezésénél átgondoltam, hogy milyen adatokra lehet szüksége egy könyvesboltnak. Kellenek a könyvek, a vásárlók, a címek, a rendelések adatai és a számlák. A rendelésben vannak a rendelés tételek, amiket külön táblában kell kezelni. A rendelést és a rendelés tételt tranzakcióban kell rögzíteni.

Vagyis először létrehoztam a következő táblákat a következő adatokkal:

1. Book
   1. Id
   2. Author
   3. Title
   4. PublishedIn
   5. Publisher
   6. ISBN
   7. Translator
   8. Language
2. Customer
   1. Id
   2. Name
   3. Email
   4. PhoneNumber
   5. Password
   6. BillingAddressId (ez természetesen az Address táblára mutat, mint idegen kulcs)
3. Address
   1. Id
   2. Country
   3. ZipCode
   4. City
   5. PublicPlaceType
   6. PublicPlaceName
   7. HouseNumber
   8. OtherAddress
4. Order
   1. Id
   2. CustomerId (idegen kulcsa a Customernek)
   3. DeliveryAddressId (idegen kulcsa az Addressnek)
   4. OrderDate
5. Invoice
   1. Id
   2. InvoiceNumber
   3. OrderId (idegen kulcsa az Ordernek)
6. OrderItem
   1. Id
   2. OrderId (idegen kulcsa az Ordernek)
   3. BookId (idegen kulcsa a Booknak)

Ezek után végiggondoltam, hogy milyen adat lehet még fontos. Ekkor eszembe jutott, hogy érdemes lehet nyomon követni, hogy mikor és ki hozta létre és módosította az adatokat, így minden egyes fenti táblához még a következő 4 oszlopot adtam hozzá:

1. CreatedOn
2. CreatedBy
3. LastModifiedOn
4. LastModifiedBy

Viszont valahol számon kéne tartani a könyvek darabszámát és árát, így a Book táblát a következő két oszloppal egészítettem ki:

1. Price
2. Piece

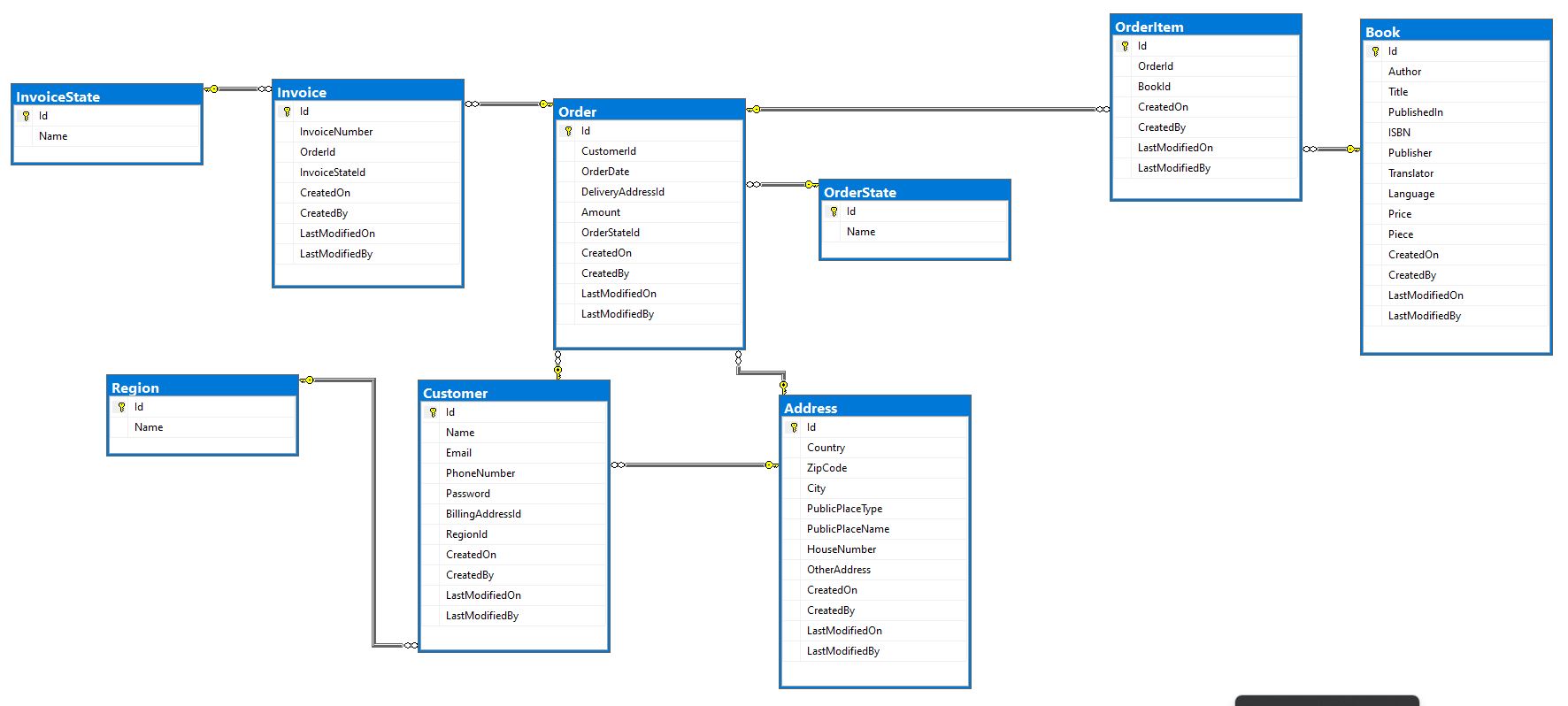
Valaminek kell, hogy tartalmazza a végösszeget is, amit az Order táblába tettem be, vagyis hozzáadtam az Amount oszlopot.

Végiggondoltam, hogy a rendelések meg a számlák is különböző státuszban lehetnek. Így létrehoztam két kód-táblát:

1. InvoiceState
   1. Id
   2. Name
2. OrderState
   1. Id
   2. Name

Az Invoice táblába az InvoiceState oszlopot, az Order oszlopban az OrderState oszlopot. Ezek természetesen mind külső kulcsok, melyek sorra az InvoiceState és az OrderState táblákra mutatnak.

Vagyis az adatbázisom végső felépítése:



Már az elején létrehoztam 2 indexet. Az egyiket az Author-ra, a másikat a Title-re, mivel ezekre keresnek a legtöbbször a vásárlók.

A Vásárlók tábla (Customer) partícionálása miatt felvettem egy új oszlopot a Customer táblába RegionId néven és létrehoztam egy Region táblát öt rekorddal, ami Észak-, Dél-, Kelet-, Nyugat- és Közép-Magyarországgal.

**Adatbázis teljesítmény analízis**

Az adatbázis teljesítmény analízishez írtam egy lekérdezést, ami a vásárlók legutóbbi rendeléseit és a rendelésükhöz tartozó könyvek adatait tartalmazzák.

A lekérdezés a következő:

explain analyze **select** c.name, c.email, o.orderdate, b.author, b.title, b.price **from** customer c

**inner** **join** **"Order"** o **on** c.id = o.customerid

**inner** **join** orderitem oi **on** o.id = oi.orderid

**inner** **join** book b **on** b.id = oi.id

**where** o.orderdate = (**select** **max**(orderdate) **from** **"Order"** **where** customerid = c.id)

Az analízis eredménye a következő:

Merge Join (cost=0.86..897235.49 rows=9 width=67) (actual time=272.024..9606.444 rows=1941 loops=1)

Merge Cond: (oi.id = b.id)

-> Nested Loop (cost=0.57..5579874.38 rows=56 width=52) (actual time=272.008..9600.746 rows=1942 loops=1)

Join Filter: (o.id = oi.orderid)

Rows Removed by Join Filter: 29938545

-> Index Scan using orderitem\_pkey on orderitem oi (cost=0.29..2135.18 rows=60126 width=16) (actual time=0.013..5.071 rows=10001 loops=1)

-> Materialize (cost=0.29..5565112.77 rows=14 width=52) (actual time=0.027..0.839 rows=2994 loops=10001)

-> Nested Loop (cost=0.29..5565112.70 rows=14 width=52) (actual time=271.989..7378.856 rows=2994 loops=1)

-> Seq Scan on "Order" o (cost=0.00..336.00 rows=15000 width=24) (actual time=0.007..1.867 rows=15000 loops=1)

-> Memoize (cost=0.29..1869.01 rows=5 width=44) (actual time=0.491..0.491 rows=0 loops=15000)

Cache Key: o.customerid, o.orderdate

Cache Mode: logical

Hits: 103 Misses: 14897 Evictions: 0 Overflows: 0 Memory Usage: 1373kB

-> Append (cost=0.28..1869.00 rows=5 width=44) (actual time=0.475..0.476 rows=0 loops=14897)

-> Index Scan using customer\_eszak\_pkey on customer\_eszak c\_1 (cost=0.28..373.82 rows=1 width=34) (actual time=0.120..0.120 rows=0 loops=14897)

Index Cond: (id = o.customerid)

Filter: (o.orderdate = (SubPlan 1))

Rows Removed by Filter: 0

SubPlan 1

-> Aggregate (cost=373.51..373.52 rows=1 width=8) (actual time=0.467..0.468 rows=1 loops=14897)

-> Seq Scan on "Order" (cost=0.00..373.50 rows=5 width=8) (actual time=0.075..0.465 rows=6 loops=14897)

Filter: (customerid = c\_1.id)

Rows Removed by Filter: 14994

-> Index Scan using customer\_del\_pkey on customer\_del c\_2 (cost=0.28..373.82 rows=1 width=34) (actual time=0.116..0.116 rows=0 loops=14897)

Index Cond: (id = o.customerid)

Filter: (o.orderdate = (SubPlan 1))

Rows Removed by Filter: 0

-> Index Scan using customer\_kelet\_pkey on customer\_kelet c\_3 (cost=0.28..373.82 rows=1 width=34) (actual time=0.121..0.121 rows=0 loops=14897)

Index Cond: (id = o.customerid)

Filter: (o.orderdate = (SubPlan 1))

Rows Removed by Filter: 0

-> Index Scan using customer\_nyugat\_pkey on customer\_nyugat c\_4 (cost=0.28..373.82 rows=1 width=34) (actual time=0.116..0.116 rows=0 loops=14897)

Index Cond: (id = o.customerid)

Filter: (o.orderdate = (SubPlan 1))

Rows Removed by Filter: 0

-> Index Scan using customer\_kozep\_pkey on customer\_kozep c\_5 (cost=0.14..373.69 rows=1 width=344) (actual time=0.001..0.001 rows=0 loops=14897)

Index Cond: (id = o.customerid)

Filter: (o.orderdate = (SubPlan 1))

-> Index Scan using book\_pkey on book b (cost=0.29..444.37 rows=10000 width=31) (actual time=0.008..4.237 rows=10000 loops=1)

Planning Time: 0.393 ms

JIT:

Functions: 68

Options: Inlining true, Optimization true, Expressions true, Deforming true

Timing: Generation 1.731 ms (Deform 0.768 ms), Inlining 11.597 ms, Optimization 150.138 ms, Emission 109.738 ms, Total 273.205 ms

Execution Time: 9608.526 ms

Az analízis eredményét kiértékelve azt láttam, hogy az Order tábla OrderDate mindig filterrel hozza, és az Order táblában CustomerId mezőjét úgy keresi, hogy az egész táblán végigmegy a keresés során. Ez a végrehajtási terv 9608,526 miliszekundumot vesz igénybe. Ezek alapján úgy gondoltam, hogy az Order tábla OrderDate és CustomerId mezőjére is indexet teszek. Ezek után újra lefutattam a teljesítményanalízist ugyanazzal a lekérdezéssel, aminek az eredménye a következő:

Merge Join (cost=22.81..14232.00 rows=9 width=67) (actual time=11.944..2790.548 rows=1941 loops=1)

Merge Cond: (oi.id = b.id)

-> Nested Loop (cost=22.52..85515.82 rows=56 width=52) (actual time=11.929..2787.053 rows=1942 loops=1)

Join Filter: (o.id = oi.orderid)

Rows Removed by Join Filter: 29938482

-> Index Scan using orderitem\_pkey on orderitem oi (cost=0.29..2135.18 rows=60126 width=16) (actual time=0.014..3.535 rows=10001 loops=1)

-> Materialize (cost=22.23..70754.22 rows=14 width=52) (actual time=0.000..0.105 rows=2994 loops=10001)

-> Nested Loop (cost=22.23..70754.15 rows=14 width=52) (actual time=0.095..56.818 rows=2994 loops=1)

-> Append (cost=0.00..112.50 rows=3100 width=44) (actual time=0.013..0.815 rows=3000 loops=1)

-> Seq Scan on customer\_eszak c\_1 (cost=0.00..21.59 rows=759 width=34) (actual time=0.013..0.179 rows=759 loops=1)

-> Seq Scan on customer\_del c\_2 (cost=0.00..21.39 rows=739 width=34) (actual time=0.007..0.138 rows=739 loops=1)

-> Seq Scan on customer\_kelet c\_3 (cost=0.00..21.65 rows=765 width=34) (actual time=0.008..0.161 rows=765 loops=1)

-> Seq Scan on customer\_nyugat c\_4 (cost=0.00..21.37 rows=737 width=34) (actual time=0.007..0.120 rows=737 loops=1)

-> Seq Scan on customer\_kozep c\_5 (cost=0.00..11.00 rows=100 width=344) (actual time=0.005..0.005 rows=0 loops=1)

-> Index Scan using ix\_order\_orderdate on "Order" o (cost=22.23..22.78 rows=1 width=24) (actual time=0.007..0.012 rows=1 loops=3000)

Index Cond: (orderdate = (SubPlan 1))

Filter: (c.id = customerid)

Rows Removed by Filter: 40

SubPlan 1

-> Aggregate (cost=21.94..21.95 rows=1 width=8) (actual time=0.006..0.006 rows=1 loops=3000)

-> Bitmap Heap Scan on "Order" (cost=4.32..21.93 rows=5 width=8) (actual time=0.003..0.004 rows=5 loops=3000)

Recheck Cond: (customerid = c.id)

Heap Blocks: exact=14796

-> Bitmap Index Scan on ix\_order\_customerid (cost=0.00..4.32 rows=5 width=0) (actual time=0.001..0.001 rows=5 loops=3000)

Index Cond: (customerid = c.id)

-> Index Scan using book\_pkey on book b (cost=0.29..444.37 rows=10000 width=31) (actual time=0.012..1.911 rows=10000 loops=1)

Planning Time: 0.785 ms

Execution Time: 2790.779 ms

Az új végrehajtási terv 2790,779 miliszekundum lett, vagyis 70,95 %-os lekérdezési javulást hozott.

A beadandó szkripteket a következő helyre töltöttem fel: https://github.com/Eszto94/DataBase

A szkriptek funkcióját az első számú melléklet tartalmazza:

1. **Melléklet**

|  |  |
| --- | --- |
| **Script neve** | **Script feladata** |
| DB\_Create.sql | Adatbázis létrehozása |
| Insert.sql | Adatbázisba rekordokat beszúró script |
| Konyv\_elerhetosege.sql | Könyvek elérhetőségéről információt nyújtó script |
| Vasarlok\_vasarlasi\_elozmenyei.sql | Vásárlók vásárlási előzményeiről információt nyújtó script |
| Rendelesek\_statusza.sql | Rendelések státuszáról információt nyújtó script |
| Konyv\_keszlet\_trigger.sql | Könyv készletét és a megrendelés összegét frissítő trigger |
| Create\_Invoice\_SP.sql | Tárolt eljárás a számla adatbázisban való létrehozására |
| Generate\_Invoice.sql | Tárolt eljárás a számla előállításához |
| BookStat\_SP.sql | A legjobban fogyó könyvek tárolt eljárás |
| CustomerStat\_SP.sql | A legaktívabb vásárlók tárolt eljárás |
| Customer\_Partition.sql | Vásárlók tábla partícionálása régiók alapján |
| Insert\_Customer.sql, Insert\_Invoice.sql, Insert\_Order\_OrderItem.sql, InsertAddress.sql, InsertBook.sql | Adatbázis feltöltése a teljesítmény teszthez |
| Teljesítmeny\_analizis.sql | Lekérdezés a teljesítmény teszthez |
| javasolt\_index.sql | Indexek létrehozása a jobb teljesítmény érdekében |