Tartalomjegyzék

[**1. Az adatbázis tervezésének bemutatása 2**](#_Toc183983657)

[**2. Adatbázis-tervezési szempontok 2**](#_Toc183983658)

[**3. Kapcsolatok és normalizáció 3**](#_Toc183983659)

[**4. Particionálás 3**](#_Toc183983660)

[**5. Indexek 3**](#_Toc183983661)

[**6. Automatizáció 3**](#_Toc183983662)

[**7. Teljesítményanalízis jelentés 3**](#_Toc183983663)

[**8. Mérések és eredmények 3**](#_Toc183983664)

[**9. Optimalizálási javaslatok 3**](#_Toc183983665)

[**10. Következtetés 4**](#_Toc183983666)

1. Az adatbázis tervezésének bemutatása

Az adatbázis célja egy online könyvesbolt működésének támogatása. A rendszer képes könyvek, vásárlók, rendelések és számlák kezelésére, valamint olyan funkciók megvalósítására, mint készletnyilvántartás és számlázás. A rendszer PostgreSQL adatbázissal készült, amely támogatja a tranzakciókat, particionálást, indexelést, valamint tárolt eljárások és triggerek használatát a műveletek automatizálására.

1. Adatbázis-tervezési szempontok

**Táblák kialakítása**

Az adatbázis a következő táblákból áll:

**Vásárlók (vasarlok)**

* + Tárolja a vásárlók alapvető adatait: név, email, cím és város.
  + A tábla particionálsa a vásárlók városa alapján növelné a lekérdezési teljesítményt, de jelenleg nincs particiónálva.
  + Elsődleges kulcs: vasarlo\_id.

**Könyvek (konyvek)**

* Tárolja a könyvek adatait, mint cím, szerző, ár, készlet, és elérhetőség.
  + Az elerheto oszlop automatikusan frissül egy trigger segítségével, amikor rendelés történik.
  + Elsődleges kulcs: konyv\_id.

**Rendelések (rendelesek)**

* + Tárolja a vásárlók által leadott rendelések adatait, például a rendelés dátumát és státuszát.
  + A vasarlo\_id külső kulcs a vásárlók táblára.
  + Elsődleges kulcs: rendeles\_id.

**Rendelés tételek (rendeles\_tetelek)**

* + Tárolja a rendelésekhez kapcsolódó tételeket, például a könyvek mennyiségét és árát.
  + A konyv\_id külső kulcs a könyvek táblára, míg a rendeles\_id a rendelések táblára.
  + Elsődleges kulcs: tetel\_id.

**Számlák (szamlak)**

* + Tárolja a rendelés alapján generált számlákat, beleértve az összeget, a kiállítás dátumát, és a fizetési módot.
  + A rendeles\_id külső kulcs a rendelések táblára.
  + Elsődleges kulcs: szamla\_id.

1. Kapcsolatok és normalizáció

Az adatbázis a harmadik normálformának (3NF) megfelelően lett megtervezve:

* Minden oszlop közvetlenül az elsődleges kulcshoz kapcsolódik.
* Nincs redundancia: az ismétlődő adatok külön táblákban vannak tárolva (pl. vásárlók és könyvek).

1. Particionálás

A vásárlók táblája város alapján particionálása gyorsítaná a lekérdezésket.

1. Indexek

Indexek létrehozása a gyakran lekérdezett mezőkre:

* idx\_konyv\_cim a konyvek.cim oszlopon.
* idx\_rendeles\_datum a rendelesek.datum oszlopon.

1. Automatizáció

* A tr\_update\_keszlet trigger biztosítja, hogy a könyvek készlete és elérhetősége mindig naprakész legyen rendeléskor.
* Tárolt eljárások, például generate\_szamla és top\_konyvek, segítik az üzleti folyamatokat.

1. Teljesítményanalízis jelentés

Tesztelési környezet

* PostgreSQL verzió: 17.2
* Hardver: 4 CPU mag, 8 GB RAM
* Adatbázis mérete: 10 000 vásárló, 5 000 könyv, 20 000 rendelés és 50 000 rendelés tétel.

1. Mérések és eredmények

**Lekérdezési idők egyszerű és összetett műveletekre**

| **Lekérdezés** | **Idő (ms)** | **Megjegyzés** |
| --- | --- | --- |
| Vásárlók adatainak lekérdezése | 15 | Particionálás segítene a gyors elérésben |
| Könyvek elérhetőségének lekérdezése | 20 | Indexek gyorsítják a keresést |
| Összetett lekérdezés több táblával | 150 | Vásárlók és rendelések statisztikái |

1. Optimalizálási javaslatok

**Indexek további optimalizálása:**

* + Összetett index létrehozásával a rendeles\_tetelek táblában a konyv\_id és rendeles\_id oszlopokra:

CREATE INDEX idx\_konyv\_rendeles ON rendeles\_tetelek (konyv\_id, rendeles\_id);

**Cache használata:**

* + Gyakran keresett könyvek vagy vásárlók esetén érdemes egy caching mechanizmust bevezetni, hogy csökkentsük a lekérdezési terhelést.

**Materializált nézetek:**

* + Materializált nézet létrehozása a legjobban fogyó könyvekhez:

CREATE MATERIALIZED VIEW top\_books AS

SELECT k.konyv\_id, k.cim, SUM(rt.mennyiseg) AS osszes\_mennyiseg

FROM konyvek k

JOIN rendeles\_tetelek rt ON k.konyv\_id = rt.konyv\_id

GROUP BY k.konyv\_id, k.cim;

1. Következtetés

Az adatbázist, nagy adatállományok kezelésekor az indexek nagyban segítik a jó teljesítményt. További optimalizációval, mint például a particionálással és a caching és materializált nézetek bevezetésevel, az adatbázis még nagyobb terhelést is képes lenne hatékonyan kezelni.