

FELADATKIÍRÁS

Ennek az oldalnak a helyére a tanszéktől kapott, aláírt és lepecsételt eredeti feladatkiírás kerül. Az elektronikusan feltöltött dolgozatba nem kell beletenni.



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Mesterséges Intelligencia és Rendszertervezés Tanszék

A dolgozat címe

Szakdolgozat

Készítette

Hallgató Hanna

Konzulens

dr. Konzulens Kata

2026. január 1.

Tartalomjegyzék

Kivonat	i
Abstract	ii
Bevezetés	1
1. Dokumentumok közös szerkesztése	2
1.1. A rendszerrel szemben támasztott követelmények	3
2. A szinkronizáció megközelítései	4
2.1. Operációs transzformáció	4
2.1.1. Transzformációs függvény	4
2.2. Konfliktusmentes replikált adattípusok	4
3. Tervezés és megvalósítás	6
3.1. Az előzmények tárolása	6
4. Kiértékelés	7
4.1. Az eredmények megvitatása	8
Összegzés	9
Köszönetnyilvánítás	10
Irodalomjegyzék	11
Függelék	12
F.1. Nyilatkozat generatív mesterséges intelligencia alkalmazásáról	13

HALLGATÓI NYILATKOZAT

Alulírott *Hallgató Hanna*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem. A Függelékben egyértelműen megjelöltem, hogy a mesterséges intelligencia eszközeit alkalmaztam-e a dolgozat elkészítéséhez; amennyiben igen, annak módját és mértékét a táblázatban közöltem. Tudomásul veszem, hogy a mesterséges intelligenciával generált tartalomért, annak mértékétől függetlenül, teljes felelősséggel tartozom.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK az interneten nyilvánosan hozzáférhetővé tegye, a munka teljes szövege pedig a BME Címtárban regisztrált személyek számára elérhető legyen. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2026. január 1.

Hallgató Hanna
hallgató

Kivonat

Ide kerül a dolgozat tartalmi összefoglalója magyarul: a feladat, a választott megoldás és az eredmények néhány mondatban.

Abstract

Here comes the abstract of the thesis in English: the task, the chosen solution and the results in a few sentences.

Bevezetés

A szöveges dokumentumok ma már csak kivételesen születnek egyetlen szerző kezében. A hallgatók párban írják szemináriumi dolgozataikat, a fejlesztők közösen tartják karban a műszaki dokumentációt, a kutatócsoportok pedig az utolsó pillanatban, egyszerre több helyszínről fejezik be cikkeiket. Ahhoz, hogy az ilyen írás működjön, a szerkesztőnek biztosítania kell, hogy minden résztvevő ugyanazt a szöveget lássa, és egyetlen módosítás se vesszen el.

Jelen dolgozat célja, hogy összehasonlítsa a dokumentumok valós idejű közös szerkesztése során alkalmazott szinkronizáció két fő megközelítését, az operációs transzformációt és a konfliktusmentes replikált adattípusokat, és egy webes szerkesztő prototípusán ellenőrizze őket. Az első fejezet meghatározza az ilyen rendszerrel szemben támasztott követelményeket, a második bemutatja mindkét megközelítést, a harmadik a prototípus tervét, a negyedik pedig a mérések eredményeit.

1. fejezet

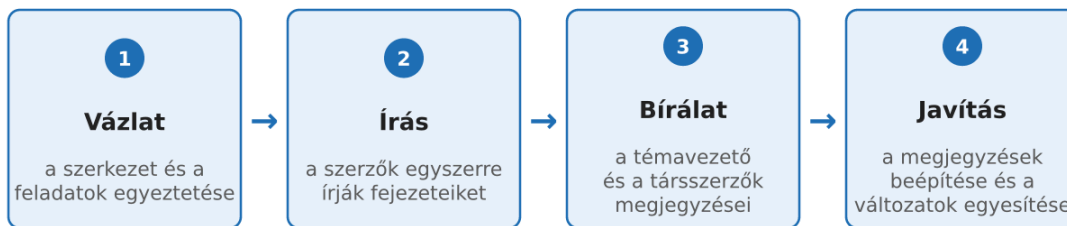
Dokumentumok közös szerkesztése

Közös szerkesztésen azt a helyzetet értjük, amikor több felhasználó módosítja ugyanazt a dokumentumot, és változtatásaik kézi összefésülés nélkül jutnak el egymáshoz. Aszerint, hogy a változtatások milyen gyorsan terjednek, három alapvető munkamódot különböztetünk meg:

- **aszinkron szerkesztés**, amelynek során a szerzők a dokumentum teljes változatait adják át egymásnak, és csak utólag fésülik össze őket,
- **zárolásos szerkesztés**, amikor a dokumentum egy adott részét egyszerre csak egy szerző módosíthatja,
- **valós idejű szerkesztés**, amelynek során minden változtatás azonnal megjelenik a többi felhasználónál.

Ez a dolgozat kizárólag az utolsó munkamóddal foglalkozik. Ez a felhasználók számára a legkényelmesebb, ugyanakkor a rendszer tervezésével szemben a legnagyobb követelményeket támasztja, mivel a változtatások sok helyen egyidejűleg keletkeznek, és eltérő sorrendben érkeznek meg.

A valós idejű szerkesztés ráadásul nemcsak a technikát változtatja meg, hanem az egész együttműködés menetét is. A közös írás jellemző folyamatát, ahogyan a megkérdezett felhasználók leírták, az 1.1. ábra szemlélteti.



a 2-4. lépés ismétlődik, amíg a bíráló lényeges megjegyzéseket hoz

1.1. ábra. A dokumentum közös írásának jellemző menete

1.1. A rendszerrel szemben támasztott követelmények

Meglévő szerkesztők felhasználóival készített interjúk és a szakirodalom alapján négy követelményt fogalmaztunk meg, amelyeknek egy közös szerkesztést támogató rendszernek meg kell felelnie:

1. konvergencia, vagyis az összes változtatás kézbesítése után a dokumentum minden másolatának azonos a tartalma,
2. a szándék megőrzése, vagyis a módosítás eredménye megfelel annak, amit a szerző a módosítás pillanatában látott,
3. válaszkészség, vagyis a helyi módosítás azonnal érvényesül, anélkül hogy a szerverre kellene várni,
4. hibatűrés, vagyis a kapcsolat rövid megszakadása nem vezet a félig megírt szöveg elvesztéséhez.

Az első két követelmény a helyességre, a másik kettő a használhatóságra vonatkozik. A gyakorlat azt mutatja, hogy éppen az azonnali válaszkészség és a garantált konvergencia együttes biztosítása a teljes tervezés legnehezebb része.

2. fejezet

A szinkronizáció megközelítései

A valós idejű közös szerkesztést támogató rendszerek minden változtatást először helyben hajtanak végre, és csak ezután küldik el a többi résztvevőnek. Elsősorban abban különböznek, hogyan kezelik azt a helyzetet, amikor két változtatás egyidejűleg keletkezik, és az egyes résztvevők eltérő sorrendben kapják meg őket.

2.1. Operációs transzformáció

Az operációs transzformáció Ellis és Gibbs munkáján alapul [1], akik elsőként írták le, hogyan lehet egy egyidejű műveletet úgy módosítani, hogy egy másik, a szerzője által nem ismert művelet után is végrehajtható legyen. Minden módosítást a szöveg egy adott pozícióján végzett beszúrási vagy törlési műveletként rögzítenek.

Ha például az egyik felhasználó egy szót szúr be egy bekezdés elejére, a másik pedig ugyanekkor töröl egy szót annak végéről, a törlés pozícióját a beszúrt szó hosszával el kell tolni. Ezt az eltolást a transzformációs függvény számítja ki.

2.1.1. Transzformációs függvény

A transzformációs függvény egyidejű műveletek egy párját kapja meg, és azok módosított változatait adja vissza. Ahhoz, hogy a dokumentum összes másolata konvergáljon, a függvénynek teljesítenie kell az úgynevezett transzformációs tulajdonságokat, amelyek teljesülését általános műveletekre nehéz bizonyítani. A legtöbb éles rendszer ezért egy központi szerverre támaszkodik, amely egyetlen műveleti sorrendet határoz meg.

2.2. Konfliktusmentes replikált adattípusok

A második megközelítést a konfliktusmentes replikált adattípusok jelentik, amelyeket összefoglalóan a CRDT rövidítéssel jelölnek [2]. A műveletek utólagos módosítása helyett

a dokumentum olyan reprezentációját választják, amelyben az egyidejű műveletek kommutálnak, és sorrendjük nem befolyásolja az eredményt.

Egy replikált adattípus konfliktusmentes, ha bármely két replika, amely ugyanazt a frissítéshalmazt kapta meg, a kézbesítés sorrendjétől függetlenül azonos állapotban van.

A dokumentum minden karaktere egyedi azonosítót kap, amely a karakter teljes élettartama alatt változatlan marad. A beszúrás pozícióját ezután nem egy szám, hanem a szomszédos karakter azonosítójára való hivatkozás határozza meg, így a szöveg más részein végzett egyidejű módosítások nem tolhatják el.

3. fejezet

Tervezés és megvalósítás

A szerkesztő prototípusa egy webes kliensből, egy szinkronizációs szerverből és egy adatbázisból áll, amelybe a szerver folyamatosan menti a dokumentumok állapotát. A kliens és a szerver úgy készült, hogy a szinkronizációs réteg cserélhető legyen, és mindkét ismertetett megközelítés ugyanazon a felhasználói felületen legyen összehasonlítható.

A kliens a dokumentum helyi replikáját tartja fenn, és a felhasználó minden módosítását a replikált adattípuson végzett műveletté alakítja. A műveleteket kötegekben küldi a szervernek, hogy gyors gépelés közben ne terhelje túl a hálózatot. A beérkező köteg szerveroldali feldolgozását egyszerűsítve a következő kódrészlet mutatja be:

```
def koteg_feldolgozasa(dokumentum, koteg):
    for muvelet in koteg.muveletek:
        if muvelet.id in dokumentum.alkalmazott:
            continue
        dokumentum.alkalmaz(muvelet)
        dokumentum.alkalmazott.add(muvelet.id)
    szetkuldes_a_tobbieknek(dokumentum.id, koteg)
```

Mivel a műveletek kommutálnak, a szervernek nem kell a kötegeket sorba rendeznie vagy transzformálnia. Elég, ha biztosítja, hogy egyetlen műveletet se hajtson végre kétszer, és minden köteg eljusson az összes csatlakozott klienshez.

3.1. Az előzmények tárolása

A szerver nemcsak a végleges szöveget tárolja, hanem a műveletek teljes naplóját is. A naplóból a dokumentum bármely korábbi változata visszaállítható, és újraépíthető egy hosszabb ideig lecsatlakozott kliens replikájának állapota. Ez a megközelítés a megváltoztathatatlan eseménynaplóra épülő architektúrának felel meg, ahogyan azt Kleppmann leírja [3]. Hogy a napló ne növekedjen korlátlanul, a szerver rendszeres időközönként pillanatképekbe tömöríti.

4. fejezet

Kiértékelés

A prototípust valós szerkesztési munkamenetek felvételeinek halmazán hasonlítottuk össze az operációs transzformáció egy referencia-implementációjával. Minden munkamenetben mértük a késleltetést, vagyis a szerzőnél végzett módosítás és a többi résztvevőnél való megjelenése között eltelt időt, a munkamenet átlagos késleltetését pedig a (4.1) szerint számítottuk ki:

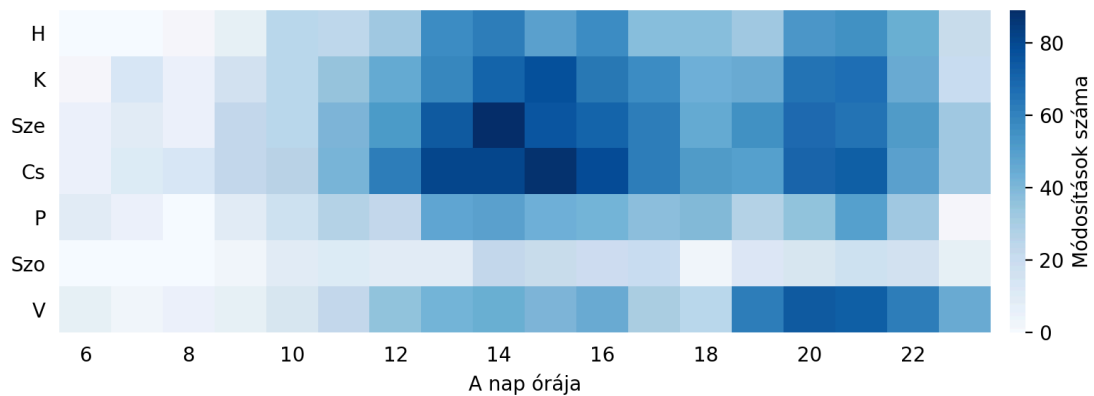
$$\bar{t} = \frac{1}{n} \sum_{i=1}^n (t_i^{\text{kézbetés}} - t_i^{\text{szerkesztés}}) \quad (4.1)$$

ahol n a munkamenet műveleteinek száma. Az egyidejűleg dolgozó felhasználók különböző számára mért értékeket a 4.1. táblázat foglalja össze.

Felhasználók	OT késleltetés (ms)	CRDT késleltetés (ms)	Adatok (kB)
2	38	35	12
5	52	41	19
10	87	48	31
20	164	63	58

4.1. táblázat. Átlagos késleltetés és a tárolt adatok mérete a felhasználók számának függvényében

A felhasználók számának növekedésével az operációs transzformáció késleltetése jóval gyorsabban nő, mivel a szervernek minden műveletet az összes egyidejű művelettel szemben transzformálnia kell. A CRDT alacsonyabb késleltetésének ára a tárolt adatok nagyobb mennyisége, hiszen minden karakter hordozza a saját azonosítóját. A módosítások időbeli eloszlását, amelyre a terheléses teszt épül, a 4.1. ábra mutatja.



4.1. ábra. A módosítások száma a hét napja és az óra szerint a vizsgált munkamenetekben

4.1. Az eredmények megvitatása

Az eredmények megerősítik, hogy kisebb csapatok esetén a két megközelítés egyenértékű, és inkább a megvalósítás egyszerűsége a döntő. Több egyidejű felhasználó esetén a CRDT egyértelmű előnyben van mind a késleltetés, mind a kapcsolat kiesésekkel szembeni ellenállóképesség tekintetében. A felhasználókkal folytatott kiegészítő interjúk, amelyeket a kvalitatív kutatás elvei szerint végeztünk [4], azt mutatták, hogy a száz milliszekundum alatti késleltetéskülönbséget a felhasználók nem érzékelik, a félig megírt szöveg kapcsolatkiesés utáni elvesztését viszont komoly problémának tartják.

Összegzés

A dolgozat a dokumentumok valós idejű közös szerkesztése során alkalmazott szinkronizáció két megközelítéseként hasonlította össze az operációs transzformációt és a konfliktusmentes replikált adattípusokat. Egy webes szerkesztő prototípusán megmutattuk, hogy a CRDT a műveletek központi sorba rendezése nélküli, egyszerűbb szerveret tesz lehetővé, és több felhasználó esetén alacsonyabb késleltetést nyújt.

A CRDT fő hátránya továbbra is a tárolt adatok nagyobb mennyisége. A következő lépés ezért annak vizsgálata lesz, hogyan befolyásolja az adاتمennyiség a napló rendszeres pillanatképekbe tömörítését, és hogyan viselkedik a prototípus hosszú távú üzemben, több tízezer dokumentummal.

Köszönetnyilvánítás

Itt mondhatasz köszönetet mindazoknak, akik a munkádat segítették. A köszönetnyilvánítás nem kötelező.

Irodalomjegyzék

- [1] C. A. Ellis és S. J. Gibbs, „Concurrency control in groupware systems”, in *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, New York: ACM, 1989, o. 399–407. doi: 10.1145/67544.66963.
- [2] M. Shapiro, N. Preguiça, C. Baquero, és M. Zawirski, „Conflict-free replicated data types”, in *Stabilization, Safety, and Security of Distributed Systems*, Berlin: Springer, 2011, o. 386–400. doi: 10.1007/978-3-642-24550-3_29.
- [3] M. Kleppmann, *Designing data-intensive applications*. Sebastopol: O’Reilly Media, 2017.
- [4] J. Hendl, *Kvalitativní výzkum: základní metody a aplikace*. Praha: Portál, 2005.

Függelék

F.1. Nyilatkozat generatív mesterséges intelligencia alkalmazásáról

- ☒ **Nem használtam** semmilyen generatív MI segédeszközt.
- ☐ **Használtam** generatív MI segédeszközt. Az MI-vel generált tartalmakat ellenőriztem, a generált kimenetek valóságtartalmáról meggyőződtem, az alábbi táblázatban megfelelően jelöltem minden használatot.

Felhasználási módok	Generatív MI eszköz(ök) neve	Érintett részek (fejezet, oldalszám, hivatkozás)	Használat becsült aránya (felhasználási módonként)
Irodalomkutatás			
Prompt lényegi része			
Programkód generálása			
Prompt lényegi része			
Új ötletek, megoldási javaslatok generálása			
Prompt lényegi része			
Vázlat létrehozása (szövegstruktúra, vázlatpontok)			
Prompt lényegi része			
Szövegblokkok létrehozása			
Prompt lényegi része			
Képek generálása illusztrációs célból			
Prompt lényegi része			
Adatvizualizáció, grafikonok generálása adatpontok alapján			
Prompt lényegi része			
Prezentáció készítése			
Prompt lényegi része			
Egyéb (nevezze meg)			
Prompt lényegi része			
Összesített százalékos érték (a feladat érdemi részére nézve):			
Összesített érték rövid, szöveges indoklása:			