Adatbázisok 1. Tranzakciók, nézettáblák, indexek – 1. rész

Párhuzamos folyamatok irányítása

Virtuális és materializált nézettáblák

Az adathozzáférés felgyorsítása

Tranzakciók

- *Tranzakció* = olyan folyamat, ami adatbázis lekérdezéseket, módosításokat tartalmaz.
- Az utasítások egy „értelmes egészt” alkotnak.
- Egyetlen utasítást tartalmaznak, vagy az SQL-ben explicit módon megadhatóak.
- Kell: végrehajtásuk tartós legyen

Tranzakciók feldolgozása

- Tranzakciófeldolgozó két nagyobb részből áll:

1. Konkurenciakezelő: tranzakciók oszthatatlanságára, elkülönítésére
   - Egy táblában ugyanazt a rekordot egyidőben ketten változtatják. Hogyan tudjuk elkerülni a versenyhelyzetet?
2. Naplózás- és helyreállítás-kezelő: tranzakciók tartósságára
   - Egymillió forintot utalunk bankfiókok között, de áramkimaradás van. Mi a korrekt adatbázis állapot?

Miért van szükség tranzakciókra?

- Általában több felhasználó, folyamat egyidőben
  - Lekérdezések és módosítások egyaránt történhetnek
- Az op. rsz.-ektől eltérően, amelyek *támogatják* folyamatok interakcióját, az *ab* rendszereknek el kell különíteniük a folyamatokat

Példa: rossz interakció

- Ön és egy barátja egy időben tölt fel 100 dollárt ugyanarra a számlára ATM-en keresztül.
  - Az ab rendszernek biztosítania kell, hogy egyik művelet se vesszen el.
- Ezzel szemben az operációs rendszerek megengedik, hogy egy dokumentumot ketten szerkesszenek egy időben. Ha mind a ketten írnak, akkor az egyik változtatás elvész (elveszhet).

Sorbarendezhetőség

- A műveletek végrehajtási sorrendje fontos
- Példa: repülőgép-helyfoglalási alkalmazás
- Tábla

```
Légijáratok(járat_szám, dátum, ülés_szám, ülés_státusz)
```

- Lekérdezés:

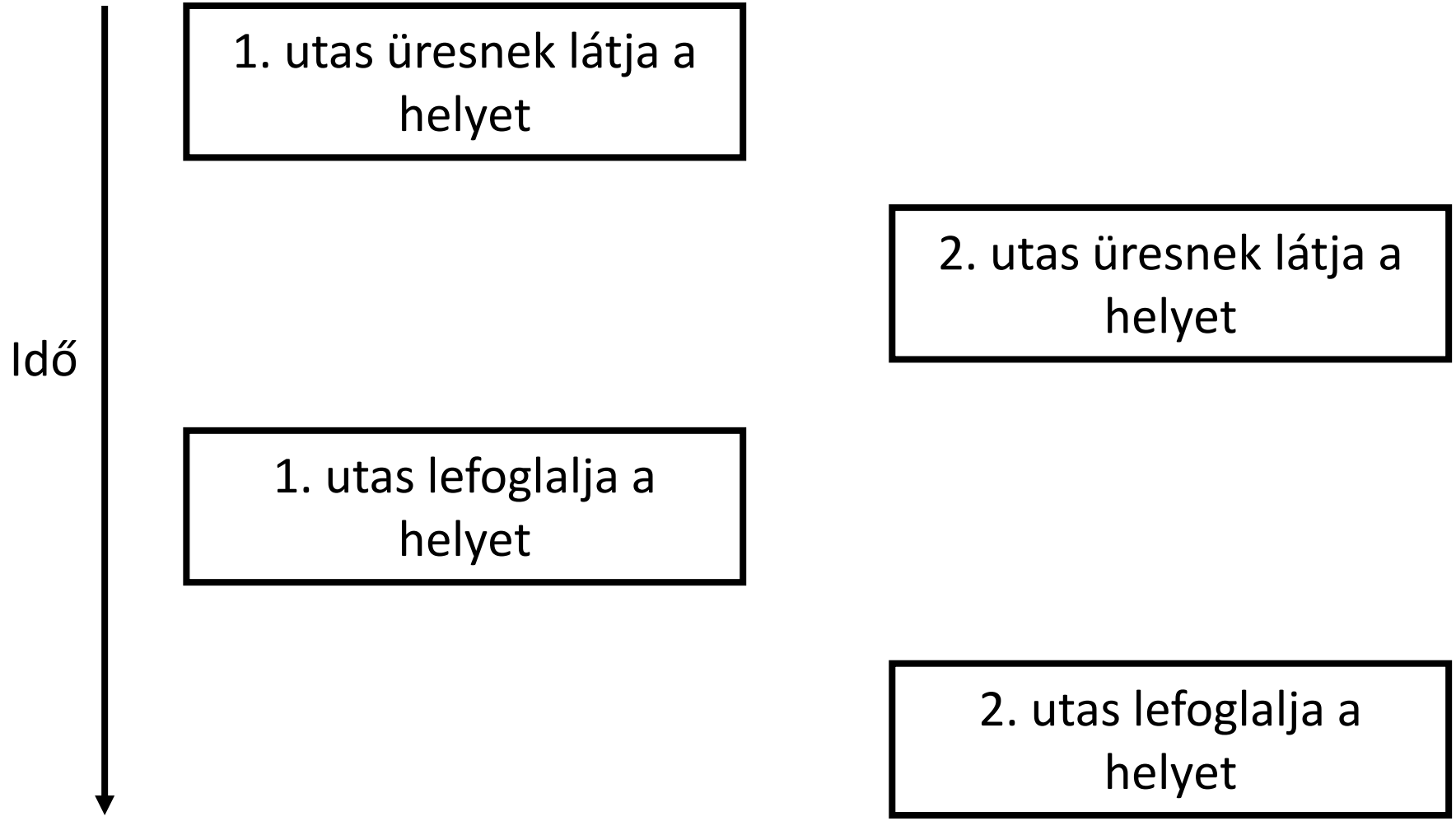
```
SELECT ülés_szám FROM Légijáratok
WHERE járat_szám = 11 AND dátum = DATE '2020-09-25'
AND ülés_státusz ='szabad';
```

- Lefoglalás:

```
UPDATE Légijáratok SET ülés_státusz ='foglalt'
WHERE járat_szám = 11 AND dátum = DATE '2020-09-25'
AND ülés_szám = '6C';
```

- Másik utas is pont ugyanekkor, ugyanazt az ülést

Sorbarendezhetőség



ACID tranzakciók

- *Az ACID tranzakciók*:
  - *Atomiság (atomicity)*: vagy az összes vagy egy utasítás sem hajtódik végre.
  - *Konzisztencia (consistency)*: az adatbázis megszorítások megőrződnek.
  - *Elkülönítés (isolation)*: a felhasználók számára úgy tűnik, mintha folyamatok, elkülönítve, egymás után futnának le.
  - *Tartósság (durability)*: egy befejeződött tranzakció módosításai nem vesznek el.
- Opcionálisan: gyengébb feltételek is megadhatóak.

ACID tranzakciók

- *Az ACID tranzakciók röviden*:
  - *Atomiság (atomicity)*: „minden vagy semmi”
  - *Konzisztencia (consistency)*: „helyesnek tűnik”
  - *Elkülönítés (isolation)*: „mintha egyedül lenne”
  - *Tartósság (durability)*: „túléli a hibákat”

COMMIT

- A COMMIT SQL utasítás végrehajtása után a tranzakció véglegesnek tekinthető.
  - A tranzakció módosításai véglegesítődnek.

ROLLBACK

- A ROLLBACK SQL utasítás esetén a tranzakció *abortál*.
  - Azaz az összes utasítás visszagörgetésre kerül.
- A 0-val való osztás vagy egyéb hibák, szintén visszagörgetést okozhatnak, akkor is, ha a programozó erre nem adott explicit utasítást.

Példa: egymásra ható folyamatok

- A Felszolgál(kocsma, sör, ár) táblánál tegyük fel, hogy Joe bárjában csak Bud és Miller sörök kaphatók 2.50 és 3.00 dollárért.
- Sally a Felszolgál táblából Joe legolcsóbb és legdrágább sörét kérdezi le.
- Joe viszont úgy dönt, hogy a Bud és Miller sörök helyett ezentúl Heinekent árul 3.50 dollárért.

Sally utasításai

(max) SELECT MAX(ár) FROM Felszolgál
WHERE kocsma = ’Joe bárja’;

(min) SELECT MIN(ár) FROM Felszolgál
WHERE kocsma = ’Joe bárja’;

Joe utasításai

- Ugyanabban a pillanatban Joe a következő utasításokat adja ki:

(del) DELETE FROM Felszolgál

WHERE kocsma = ’Joe bárja’;

(ins) INSERT INTO Felszolgál

VALUES(’Joe bárja’, ’Heineken’, 3.50);

Átfedésben álló utasítások

- A (max) utasításnak a (min) előtt kell végrehajtódnia, hasonlóan (del) utasításnak az (ins) előtt, ettől eltekintve viszont nincsenek megszorítások a sorrendre vonatkozóan, ha Sally és Joe utasításait nem gyűjtjük egy-egy tranzakcióba.

Példa: egy furcsa átfedés

- Tételezzük fel a következő végrehajtási sorrendet: (max)(del)(ins)(min).

Joe árai:	{2.50,3.00}	{2.50,3.00}		{3.50}
Utasítás:	(max)	(del)	(ins)	(min)
Eredmény:	3.00			3.50

- Mit lát Sally? MAX < MIN!

A probléma megoldása tranzakciókkal

- Ha Sally utasításait, (max)(min), egy tranzakcióba gyűjtjük, akkor az előbbi inkonzisztencia nem történhet meg.
- Joe árait ekkor egy adott időpontban látja.
  - Vagy a változtatások előtt vagy utánuk, vagy közben, de a MAX és a MIN ugyanazokból az árakból számolódik.

Egy másik hibaforrás: a visszagörgetés

- Tegyük fel, hogy Joe a (del)(ins) és utasításokat nem, mint tranzakció hajtja végre, viszont utána úgy dönt, jobb ha visszagörgeti a módosításokat.
- Ha Sally az (ins) után, de visszagörgetés előtt hajtatja végre a tranzakciót, olyan értéket kap, 3.50, ami nincs is benne az adatbázisban végül.

Megoldás

- A (del)(ins) és utasításokat Joe-nak is, mint tranzakciót kell végrehajtatnia, így a változtatások akkor válnak láthatóvá, ha a tranzakció egy COMMIT utasítást hajt végre.
  - Ha a tranzakció ehelyett visszagörgetődik, akkor a hatásai sohasem válnak láthatóvá.

Elkülönítési szintek

- Az SQL négy *elkülönítési szintet* definiál, amelyek megmondják, hogy milyen interakciók engedélyezettek az egy időben végrehajtódó tranzakciók közt.
- Ezek közül egy szint (“sorbarendezhető”) = ACID tranzakciók.
- Minden *ab* rendszer a saját tetszése szerint implementálhatja a tranzakciókat.

Az elkülönítési szint megválasztása

- Az utasítás:

SET TRANSACTION ISOLATION LEVEL *X*

ahol *X* =

1. SERIALIZABLE
2. REPEATABLE READ
3. READ COMMITTED
4. READ UNCOMMITTED

Nem-véglegesített olvasás (Read Uncommitted)

- A READ UNCOMMITTED elkülönítési szinten futó tranzakció akkor is láthatja az adatokat, amikor a változtatások még nem lettek véglegesítve („kommitolva”).
  - Piszkos adatok
- Példa: Ha Sally a READ UNCOMMITTED szintet választja, akkor is láthatja a 3.50 dollárt, mint árat, ha később Joe abortálja a tranzakcióját.

Véglegesített olvasás (Read-Committed) tranzakciók

- Ha Sally READ COMMITTED elkülönítési szintet választ, akkor csak kommitálás utáni adatot láthat, de nem feltétlenül mindig ugyanazt az adatot.
  - Nem ismételhető olvasás
- Példa: READ COMMITTED mellett megengedett a (max)(del)(ins)(min) átfedés amennyiben Joe kommitál.
  - Sally legnagyobb megdöbbenésére: MAX < MIN.

Ismételhető olvasás (Repeatable-Read) tranzakciók

- Hasonló a read-committed megszorításhoz. Itt, ha az adatot újra beolvassuk, akkor amit először láttunk, másodszor is látni fogjuk.
- De második és az azt követő beolvasások után akár *több* sort is láthatunk.

Példa: ismételhető olvasás

- Tegyük fel, hogy Sally REPEATABLE READ elkülönítési szintet választ, a végrehajtás sorrendje: (max)(del)(ins)(min).
  - (max) a 2.50 és 3.00 dollár árakat látja.
  - (min) látja a 3.50 dollárt, de 2.50 és 3.00 árakat is látja, mert egy korábbi olvasáskor (max) már látta azokat.

Sörök(<u>név</u>, gyártó)
Kocsmák(<u>név</u>, cím, engedélySzám)
Alkeszek(<u>név</u>, cím, telefon)
Szeret(<u>alkesz</u>, <u>sör</u>)
Felszolgál(<u>kocsma</u>, <u>sör</u>, ár)
Látogat(<u>alkesz</u>, <u>kocsma</u>)

Példa: ismételhető olvasás

- Vegyünk még egy példát:

Idő

1. tranzakció
```
SELECT * FROM Felszolgál WHERE ár BETWEEN 2 AND 5;
```

2. tranzakció
```
INSERT INTO Felszolgál(kocsma, sör, ár)
       VALUES ('Joe kocsmája','Soproni',2.5);
COMMIT;
```

1. tranzakció
```
SELECT * FROM Felszolgál WHERE ár BETWEEN 2 AND 5;
COMMIT;
```

Sörök(<u>név</u>, gyártó)
Kocsmák(<u>név</u>, cím, engedélySzám)
Alkeszek(<u>név</u>, cím, telefon)
Szeret(<u>alkesz</u>, <u>sör</u>)
Felszolgál(<u>kocsma</u>, <u>sör</u>, ár)
Látogat(<u>alkesz</u>, <u>kocsma</u>)

Példa: ismételhető olvasás

- Vegyünk még egy példát:

Idő

1. tranzakció
```
SELECT * FROM Felszolgál WHERE ár BETWEEN 2 AND 5;
```

2. tranzakció
```
INSERT INTO Felszolgál(kocsma, sör, ár)
       VALUES ('Joe kocsmája','Soproni',2.5);
COMMIT;
```

1. tranzakció
```
SELECT * FROM Felszolgál WHERE ár BETWEEN 2 AND 5;
COMMIT;
```

→ Fantomadat megjelenik

- A fantomolvasás ellen nem véd

Sorbarendezhető (serializable) tranzakciók

- Ha Sally a (max)(min), Joe a (del)(ins) tranzakciót hajtatja végre, és Sally tranzakciója SERIALIZABLE elkülönítési szinten fut, akkor az adatbázist vagy Joe módosításai előtt vagy után látja, a (del) és (ins) közötti állapotban sohasem.

Az elkülönítési szint személyes választás

- Az Ön döntése csak azt mondja meg, hogy *Ön* hogy látja az adatbázist, és nem azt, hogy mások hogy látják azt.
- Példa: Ha Joe sorbarendezhető elkülönítési szintet használ, de Sally nem, akkor lehet, hogy Sally nem talál árakat Joe bárja mellett.
  - azaz, mintha Sally Joe tranzakciójának közepén futtatná a sajátját.

Tranzakciók

- Az SQL elkülönítési szintjeinek tulajdonságai

Elkülönítési szint	Piszkos adat olvasása	Nem ismételhető olvasás	Fantomadatok
READ UNCOMMITTED	Megengedett	Megengedett	Megengedett
READ COMMITTED	Nem Megengedett	Megengedett	Megengedett
REPEATABLE READ	Nem Megengedett	Nem Megengedett	Megengedett
SERIALIZABLE	Nem Megengedett	Nem Megengedett	Nem Megengedett