Tranzakciók, nézettáblák, indexek

Párhuzamos folyamatok irányítása

Virtuális és materializált nézettáblák

Az adathozzáférés felgyorsítása

Miért van szükség tranzakciókra?

- Az adatbázis rendszereket általában több felhasználó és folyamat használja egyidőben.
  - Lekérdezések és módosítások egyaránt történhetnek.
- Az operációs rendszerektől eltérően, amelyek *támogatják* folyamatok interakcióját, az *ab* rendszereknek el kell különíteniük a folyamatokat.

Példa: rossz interakció

- Te és egy barátod egy időben töltötök 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).

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.

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.

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, utána viszont 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 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

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

- A döntésed csak azt mondja meg, hogy *te* hogy látod 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.

Read-Commited tranzakciók

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

Repeatable-Read tranzakciók

- Hasonló a read-commited 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.

Read Uncommitted

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

Nézettáblák

- A *nézettábla* olyan reláció, amit tárolt táblák (*alaptáblák*) és más nézettáblák felhasználásával definiálunk.
- Kétféle létezik:
  1. *virtuális* = nem tárolódik az adatbázisban; csak a relációt megadó lekérdezés.
  2. *Materializált* = kiszámítódik, majd tárolásra kerül.

Nézettáblák létrehozása

- Deklaráció:

  CREATE [MATERIALIZED] VIEW
  <név> AS <lekérdezés>;

- Alapesetben virtualizált nézettábla jön létre.

Példa: nézettábla definíció

- Ihatja(alkesz, sör) nézettáblában az alkeszek mellett azon söröket tároljuk, melyeket legalább egy kocsmában felszolgálnak az általa látogatottak közül:

```
CREATE VIEW Ihatja AS
    SELECT alkesz, sör
    FROM Látogat, Felszolgál
    WHERE Látogat.kocsma =
                        Felszolgál.kocsma;
```

Példa: nézettáblákhoz való hozzáférés

- A nézettáblák ugyanúgy kérdezhetők le, mint az alaptáblák.
  - A nézettáblákon keresztül az alaptáblák néhány esetben módosíthatóak is, ha a rendszer a módosításokat át tudja vezetni.
- Példa lekérdezés:

```
SELECT sör FROM Ihatja
WHERE alkesz = 'Sally';
```

Materializált nézettáblák

- Probléma: minden alkalommal, amikor az alaptáblák valamelyike változik, a materializált nézettábla frissítése is szükségessé válhat.
  - Ez viszont néha túl költséges.
- Megoldás: Periodikus frissítése a materializált nézettábláknak, amelyek egyébként „nem aktuálisak”.

Példa: levelezési lista

- A következő levelezési lista cs145-aut0708 valójában egy materializált nézettábla, ami a kurzusra beiratkozott hallgatókat tartalmazza.
- Ezt négyszer frissítik egy nap.
  - A feliratkozás után közvetlen még nem feltétlen kapja meg az ember az akkor küldött emaileket.

Példa: adattárház

- Wal-Mart minden áruházának minden eladását egy adatbázisban tárolja.
- Éjszaka az új adatokkal frissítik az áruház lánc *adattárházát*, ami itt az eladások materializált nézeteiből áll.
- Az adattárházat aztán elemzők használják, hogy trendeket figyeljenek meg és odamozgassák az árukat, ahol azok a legjobb áron értékesíthetők.

Indexek

- *Index* = olyan adatszerkezet, amivel egy-egy reláció sorait gyorsabban érhetjük el adott attribútumának, attribútumainak értéke, értékei alapján.
- Lehet hash tábla, de az *ab* rendszerekben a legtöbb esetben kiegyensúlyozott keresési fával valósítják meg (*B-fák*).

Indexek deklarálása

- Nincs standard megoldás!
- Tipikus szintaxis:

```
CREATE INDEX SörInd ON
  Sörök(gyártó);
CREATE INDEX EladásInd ON
  Felszolgál(kocsma, sör);
```

Indexek használata

- Adott *v* értékre az index azokhoz a sorokhoz irányít, ahol ez a *v* érték megjelenik a megfelelő attribútum(ok)nál.
- Példa: a SörInd és az EladásInd indexek segítségével megkeressük azokat a söröket, melyeket Pete gyárt és Joe árul. (következő dia)

Indexek használata --- (2)

```
SELECT ár FROM Sörök, Felszolgál
WHERE gyártó = 'Pete' AND
    Sör.név = Felszolgál.sör AND
    kocsma = 'Joe bárja';
```

1. A SörInd segítségével megkapjuk azokat a söröket, melyeket Pete gyárt.
2. Aztán a EladásInd használatával a Joe bárjában felszolgált sörök árait kapjuk meg.

Adatbázisok hangolása

- Az adatbázisok hangolásánál komoly kérdést jelent annak eldöntése, hogy milyen indexeket használjanak.
- Mellette: az index felgyorsíthatja a lekérdezések végrehajtását.
- Ellene: a módosítások lassabbak lesznek, hiszen az indexeket is módosítani kell.

Példa: hangolás

- Tegyük fel, hogy a sörös adatbázisunkban a következők történhetnek:
  1. Új tények kerülnek egy relációba (10%).
  2. Adott kocsmára és sörre keressük az ottani árat (90%).
- Ekkor az EladásInd a Felszolgál(kocsma, sör) fölött nagyszerű szolgálatot tesz, a SörInd a Sörök(gyártó) fölött pedig inkább a kárunkra van.

Hangolási szakértők

- Fontos kutatási feladat.
  - A kézi hangolás nagy szakértelmet kíván.
- A szakértő először egy *lekérdezés terhelési kimutatást* (query load) kap kézhez:
  1. véletlenszerűen lekérdezéseket választanak a korábban végrehajtottak közül.
  2. A tervező átad egy mintát.

Hangolási szakértők --- (2)

- A szakértő létrehozza a szerinte fontos indexeket, majd megvizsgálja azok hatását.
  - Minden minta lekérdezés esetén a lekérdezés optimalizálónak használnia kell az indexeket.
  - Így meg tudja mondani, hogy javult-e összességében a lekérdezések végrehajtási ideje.