13. előadás

Állományok

Összetett állományok

Adatszerkezetek és algoritmusok előadás
2018. május 8.

Kósa Márk, Pánovics János és Szathmáry László
Debreceni Egyetem
Informatikai Kar

Összetett állományszerkezetek

Kósa Márk
Pánovics János
Szathmáry László

Egy összetett állományszerkezet alapját egy egyszerű szerkezetű állomány, az alapállomány képezi, amely legtöbbször szeriális, néha szekvenciális. Az alapállományra épülnek rá a plusz információhordozó adatok.

Összetettállomány-kezelési technikák

- láncolás: az információhordozó adatok az állományon belül jelennek meg mutatómezők formájában, ekkor a rekordokat láncolt listába fűzzük föl
- indexelés: a plusz információk az állományon kívül, egy (vagy több) indextábla formájában jelennek meg

Mivel mind a két technika lemezcímeket kezel, ezért az összetett állományszerkezetek csak közvetlen elérésű háttértárolón alakíthatók ki.

Indexelés

Létezik az alapállomány, s ettől függetlenül jön létre a szerkezethordozó állomány.

Fölépül egy indextábla, amely annyi sorból áll, ahány rekordja van az alapállománynak. Az indextábla két oszlopa: **(1)** itt jelennek meg az alapállomány rekordjainak elsődleges kulcsai növekvő sorrendben; **(2)** az elsődleges kulcsú rekord lemezcíme (vagy olyan információ, amiből ez kiszámítható).

Lényeg: az alapállományra egy közvetlen elérésű lehetőséget raktunk rá.

Ha keresünk egy adott elsődleges kulcsú rekordot, akkor azt először az indextáblában keressük. Az indextábla általában kicsi, befér az elsődleges tárba. Bináris keresés is alkalmazható (gyors). Ezt a konstrukciót *elsődleges kulcsra épülő, egyszintű, teljes indexelés*nek nevezzük.

Kósa Márk
Pánovics János
Szathmáry László

Indexelés

Elképzelhető, hogy az indextáblában nem minden rekord elsődleges kulcsát tüntetem fel, hanem csak minden *X*. rekordét. Ez a *nem teljes (részleges) egyszintű indexelés*. Az indextábla mérete így csökken. Az indextábla alapján egy intervallumot tudunk behatárolni, mely intervallumot végig kell nézni.

Maga az indextábla egy szekvenciális állomány, vagyis indexelhető. Fölé létrehozhatok egy újabb indextáblát, amely most már ezt indexeli. (Itt már természetesen nem teljes indexelésről van szó. Ez egy 2. szintű indextábla. Például minden 100. index van benne feltüntetve.)

Először szekvenciálisan keresek a 2. indextáblában addig, amíg az adott kulcsot vagy egy nála nagyobbat nem találok. Ha nincs meg, akkor a behatárolt intervallumban keresek az 1. indextáblában. Ez akkor használatos, ha az indextábla nagyon nagy. Természetesen tetszőleges számú indextábla létrehozható. Ezt *többszintű indexelés*nek nevezzük.

Kósa Márk
Pánovics János
Szathmáry László

Indexelés

Az állományt bármely mezőre, mint másodlagos kulcsra indexelhetjük. Ekkor az indextábla kulcs oszlopába a rekordokban szereplő különböző másodlagoskulcs-értékeket írjuk, vagyis a táblának annyi sora lesz, ahány különböző másodlagoskulcs-érték szerepel az alapállományban. Az egyes kulcsértékek mellé az érték oszlopba több lemezcím is kerülhet, hiszen több olyan rekord is létezhet, amelynek azonos a másodlagos kulcsa.

Kósa Márk
Pánovics János
Szathmáry László

Összetett állományszerkezetek

Elnevezések

- Az elsődleges kulcsra felépített indextáblák és az alapállomány együttes neve: indexelt állomány. Fontos: közvetlen elérést teszünk az alapállományra (indexelt szeriális és indexelt szekvenciális állomány).
- Másodlagos kulccsal történő indexelés esetén az alapállomány és az indextáblák együttese: invertált állomány.

Az összetett állományszerkezetek összefoglaló táblázata:

technika / kulcs	láncolás	indexelés
elsődleges kulcs	láncolt szeriális állomány	indexelt szeriális állomány indexelt szekvenciális állomány
másodlagos kulcs	multilista állomány	invertált állomány

Kósa Márk
Pánovics János
Szathmáry László

Láncolt szeriális állomány

A szeriális alapállomány rekordjainak elsődleges kulcsa szerint felépítünk egy láncszerkezetet, amely a szeriális állomány szekvenciális feldolgozását teszi lehetővé. A láncot úgy alakítjuk ki, hogy a rekordok az elsődleges kulcs alapján növekvő sorrendben legyenek.

Műveletek

- Létrehozás: két lépésben. (1) szeriális állomány; (2) mutatómezők kitöltése, beláncoljuk a rekordokat a megfelelő sorrendben.
- Bővítés: a rekordot az állomány végére írjuk, majd beillesztjük a láncba.
- Törlés: az állományból logikailag töröljük, míg a láncból fizikailag is töröljük.
- Csere: az alapállománynak megfelelő módon.
- Elérés: soros (fizikai sorrend) vagy szekvenciális (láncolási sorrend).
- Keresés: teljes vagy lineáris.
- Feldolgozás: van egy történeti sorrend (ahogy jönnek a rekordok), ill. van egy szekvenciális sorrend.
- Újraszervezés: a szeriális állományból csak így tudunk fizikailag is törölni.

Kósa Márk
Pánovics János
Szathmáry László

Indexelt szeriális állomány

Adott egy szeriális alapállomány, s e mellé építek fel egy indextáblát az elsődleges kulcs alapján. Ez az indextábla teljes. Természetesen többszintű indexelés is alkalmazható. A szeriális állományra egy közvetlen elérést teszek.

Műveletek

- Létrehozás: két lépésben. (1) szeriális állomány; (2) indextábla vagy indextáblák.
- Bővítés: a rekordot az állomány végére írjuk. Az egyszintű indextáblát újra kell szervezni. Többszintű indexelés esetén a magasabb szinteken is keresztülvezetjük a változásokat. Bucket technika.
- Törlés: a szeriális állományból logikailag, míg az indextáblából fizikailag is töröljük.
- Csere: az alapállománynak megfelelő módon.
- Elérés: soros (fizikai sorrend), szekvenciális (az indexszerkezet alapján) vagy közvetlen (az indexszerkezet alapján).
- Újraszervezés: ha túl sok a logikailag törölt rekord.

Relációs adatbáziskezelők egyik kedvence. Könnyű megvalósítani, gyors.

Kósa Márk
Pánovics János
Szathmáry László

Indexelt szekvenciális állomány

Lényeg: a szekvenciális állományra rárakunk egy közvetlen elérési technikát.

Kérdés:

- Bővítés.
- Milyen indexszerkezetet tudok használni?

Létrehozás: két lépésben vagy egy lépésben. A rekordok fizikai és logikai sorrendje megegyezik, hiszen a rekordok az elsődleges kulcs szerint rendezettek. Az indextáblát párhuzamosan is létre tudom hozni.

Bővítés:

- újjászervezéssel (elvi lehetőség, lassú)
- független túlcsordulási terület alkalmazása
- osztott szabad helyek módszere

Kósa Márk
Pánovics János
Szathmáry László

Indexelt szekvenciális állomány

Bővítés: független túlcsordulási terület alkalmazása

Az alapállomány és indextáblák elfoglalnak valamilyen területet, s a bővítőrekordokat egy független túlcsordulási területre rakjuk.

Annak érdekében, hogy tudjuk, hogy honnan csordult túl, kombináljuk a láncolással. A bővítőrekordot az érkezés sorrendjében a független túlcsordulási területre rakjuk s beláncoljuk. A láncolást úgy végezzük el, hogy a logikai sorrendet tükrözze. Az indextábla újraszerveződik, de ezt valószínűleg a memóriában tudjuk kezelni.

Bővítés: osztott szabad helyek módszere

A logikai rekordokat csoportokba szervezzük, s az egyes csoportokban üres helyeket hagyunk az újonnan érkező rekordok számára. Két lehetőség:

- A bővítőrekordot betesszük a csoportba az érkezés sorrendjében. Ez nehezíti a visszakeresést, mivel a pótlólag bevitt rekordok megtalálásához végig kell keresni a csoportot.
- A rekord beillesztésekor a csoportot átrendezzük. Ez időigényes megoldás. Mivel az állományból való szekvenciális olvasás gyakoribb, mint az új rekordok beillesztése, ezért ez a megoldás jóval gazdaságosabb.

További előny: nincs szükség láncolásra, nincsenek túlcsordult rekordok az állományban. Tulajdonképpen dinamikus újraszervezést valósíthatunk meg mindaddig, amíg van szabad hely az újonnan érkező rekordok számára.

Kósa Márk
Pánovics János
Szathmáry László

Indexelt szekvenciális állomány

További műveletek

- Törlés: logikai.
- Csere: az alapállománynak megfelelő módon.
- Elérés: szekvenciális és közvetlen.
- Újraszervezés: ha túl sok a logikailag törölt állomány, vagy túl sok a túlcsordult rekord, vagy ha megváltozik az indexszerkezet.

Kósa Márk
Pánovics János
Szathmáry László

Multilista állomány

(Az elsődleges kulcsra épülő állományszerkezetek esetében a direkt és random állomány nem jön szóba. A másodlagos kulcsra való szervezéskor azonban mind a négy egyszerű állományszerkezet szóba jöhet.)

A multilista állomány a láncolást használja ki. Adott alapállomány esetén bármely másodlagos kulcsra felépíthető egy-egy multilista állomány.

Ötlet: az alapállományra ráhúzunk egy asszociatív szerkezetet. Az asszociációs csoportokat egy-egy lánc fogja képviselni. Annyi részlánc jön létre, ahány másodlagos kulcsa van az állománynak.

Minden multilista állomány mellé felépül egy-egy táblázat, melynek alaphelyzetben 3 oszlopa van.

Kósa Márk
Pánovics János
Szathmáry László

Műveletek

- Létrehozás: két lépésben. (1) alapállomány létrehozása; (2) láncszerkezet és táblázat kialakítása.

  Azok a rekordok, amelyek azonos másodlagos kulcsértékkel rendelkeznek összeláncolódnak. Sorrend: amit az alapállomány meghatároz. Ki kell tölteni a táblázatot: (a) abban a sorrendben, ahogy az állományban előfordulnak, vagy (b) rendezetten.

  A táblázat oszlopai:

  1. A másodlagos kulcsérték.
  2. Láncfejek. A lánc első elemének a címe.
  3. Hány elemű a lánc. Hány olyan rekord van, melynek a másodlagos kulcsértéke ilyen.

- Bővítés: a rekord fizikai helyét az alapállomány bővítési szabályai határozzák meg. A rekordot be kell tenni a láncszerkezetbe is. A bővítőrekordot a lánc elejére is betehetjük (a táblázatból kiderül az 1. elem címe), egyszerűen megoldható, viszont nagyon sok lesz a fejmozgás. A logikai és fizikai elhelyezkedés nagyon távol lesz. Ezért inkább a lánc végére érdemes betenni. Ekkor azonban végig kell menni a láncon. Ötlet: használjunk egy 4. oszlopot is, mely az utolsó láncelem címét tartalmazza. Nagyobb a táblázat, de kevesebb a fejmozgás.

  Ha meglévő részláncba pakolok be: módosítom a táblázatot a megfelelő helyen. Ha még nem volt ilyen másodlagos kulcsú: a táblázatban új sort hozunk létre. Ez a táblázat a tárban kezelhető, gyors.

Kósa Márk
Pánovics János
Szathmáry László

Multilista állomány

Műveletek

- Törlés: az alapállományból logikailag töröljük, míg a láncból fizikailag is töröljük s módosítjuk a táblázatot. Speciális eset: törlés egyelemű részláncból. Ekkor az adott sort a táblázatból el kell távolítani. Így az adott másodlagos kulcsérték eltűnik a multilistából.
- Csere: az alapállományban a csereszabálynak megfelelően cserélek. Ha módosul a másodlagos kulcs: törlés + bővítés. (Ha nem módosul, akkor nincs gond.)
- Elérés: adott másodlagos kulcsú rekordok keresése. Ha a táblázat rendezett, akkor alkalmazható bináris keresés is.

A táblázat számos kérdésre választ ad:

- Létezik-e ilyen másodlagos kulcsú rekord?
- Ha igen, hány darab van belőle?

Kósa Márk
Pánovics János
Szathmáry László

Invertált állomány bitmátrixszal

Felépítünk egy bitmátrixot, amelynek annyi oszlopa van, ahány rekordja az alapállománynak.

Az oszlopokat címkézhetjük a rekordok elsődleges kulcsával (kölcsönös és egyértelmű).

A sorokat a másodlagos kulcsértékek szerint címkézzük. A sorrend lényegtelen.

A mátrixban 1-est írunk oda, ahol az adott rekordban a másodlagos kulcs megjelenik (különben 0-t). Egy oszlopban legfeljebb egy darab 1-es szerepelhet. Egy sorban viszont tetszőleges számú 1-es lehet.

Az eredmény egy ritka (bit)mátrix lesz, mely kicsi és gyorsan kezelhető. A mátrix (táblázat) itt is számos kérdésre választ ad.

Kósa Márk
Pánovics János
Szathmáry László

Invertált állomány bitmátrixszal

Műveletek

- Létrehozás: két lépésben. (1) alapállomány létrehozása; (2) táblázat kialakítása.
- Bővítés: alapállomány bővítése az állomány szabályai szerint. A táblázatot egy oszloppal bővítem (új elsődleges kulcs). Új másodlagos kulcsérték esetén egy sorral bővül a táblázat.
- Törlés: alapállományból logikai törlés. Mátrixból fizikai törlés. Az oszlopok száma biztosan csökken. Ha a sorban csak ott volt 1-es, akkor azt is töröljük.
- Csere: az 1-es valamelyik sorból egy másikba kerül át. Ez lehet egy teljesen új sor is. Csökkenhet is a sorok száma (ha a sorban csak ott volt 1-es).
- Feldolgozás: lásd példa.

Nagyon népszerű, a relációs adatbáziskezelők egyik kedvence.

Kósa Márk
Pánovics János
Szathmáry László