Johdanto

- Georgios Evangelidis (Technological Educational Institute of Thessaloniki), David Lomet (Microsoft), Betty Salzberg (Northeastern University, Boston): The hB^{Π} -tree: A Multi-attribute Index Supporting Concurrency, Recovery and Node Consolidation
- hB^{Π} -puut mahdollistavat usean attribuutin indexoinnin.
- Tarkoituksena yhdistää hB -puu, joka mahdollistaa usean attribuutin indexoinnin ja Π -puu, joka taas tarjoaa tehokkaat rinnakkaisuus- ja toipumismetodit.
- hB^{Π} -puusta on olemassa useita eri versioita, joilla kullakin on erilainen tapa solmujen pilkkomiseen ja indexivälin kiinnittämiseen.
- Tehokkuustestit osoittavat hB^{Π} -puut erittäin tehokkaiksi hakurakenteiksi.
- Tässä esityksessä käydään läpi ensin Π -puiden rakenne ja toiminnot, sitten hB -puiden rakenne ja toiminnot, ja näiden jälkeen hB^{Π} -puiden rakenne ja toiminnot, sekä muutamia suorituskykytestausten tuloksia.

Moniattribuuttinen data

- Moniattribuuttinen data tarkoittaa dataa, joka järjestetään useamman kuin yhden attribuutin mukaan. Esimerkiksi leveysasteet, pituusasteet ja korkeus maanpinnasta.

- Suoraviivainen tapa moniattribuuttisen datan käsittelyyn on käyttää useita yksiattribuuttisia indeksejä. Valitettavasti se on yleensä erittäin tehotonta, sillä vain yksi näistä indekseistä voi olla klusteroiva indeksi ja sen lisäksi aina insertointien ja deletointien yhteydessä kaikki indeksit pitää päivittää.

- Jotta multiattribuuttista dataa voitaisiin käsitellä tehokkaasti, pitää määritellä multiattribuutti-indeksi.

- Multiattribuuttinen data jakautuu kahteen päätyyppiin:

- **Pistedata**, joka voidaan mieltää geometrisiksi pisteiksi k-ulotteisessa avaruudessa. Esimerkiksi (x, y, z) -kolmikot kolmiulotteisessa Euclidean avaruudessa.

- **Avaruusdata**, joka voidaan mieltää k-ulotteisen avaruuden aliavaruuksiksi. Esimerkiksi monikulmiot kaksiulotteisessa Euclidean avaruudessa. Nämä ovat yleensä aina maantieteellistä dataa.

Π -Puun rakenne

- Π -puu on yleistetty B^{link} -puu. B^{link} -puussa on sivuosoittimia (side-pointers) tai linkkejä osoittamassa indeksisolmusta samalla tasolla olevaan viereiseen indeksisolmuun. Linkit mahdollistavat sen että indeksivälin kiinnittäminen voi olla erillinen toiminto indeksisolmun puolittamisesta, sillä vaikkei tieto solmun puolittamisesta vielä olekaan ehtinyt vanhemmalle saakka, voidaan haku kuitenkin suorittaa oikein linkkien avulla.

- Π -puussa rakenteen vaihdos tapahtuu sarjana atomisia toimenpiteitä, jolloin muissa kuin lehtisolmuissa tarvitsee käyttää vain lyhytaikaisia lukkoja.

- Π -puu on rakenteeltaan DAG (directed asyclic graph). Se koostuu indeksi- ja datasolmuista. Kukin solmu on vastuussa tietystä osasta avainavaruutta. Π -puun solmu voi joko:

- **Olla suoraan vastuussa** osasta avainavaruutta. Indeksisolmulle tämä on se avaruus, jonka solmu jakaa sen lapsisolmujen kesken ja datasolmulle tämä on se avaruus, jossa olemassa olevat ja potentiaaliset datapisteet sijaitsevat. Tätä avaruutta kutsutaan indeksiväliksi (index terms).

- **Delegoida vastuun** osasta avaruudesta sisarussolmuilleen. Tätä avaruutta kutsutaan sisarväliksi (sibling terms).

- Kukin puun taso alkaa yhdestä juurisolmusta. Solmun jakaminen luo aina uuden solmun samalle tasolle. Kaikki datasolmut sijaitsevat puun alimmalla tasolla.

- Π -puussa yhteen solmuun voi viitata useampi vanhempi.Tällaisia solmuja kutsutaan monivanhempisiksi solmuiksi (multi-parent node). Solmua, jolla on vain yksi vanhempi kutsutaan vastaavasti yksivanhempiseksi (single-parent node).

Haku Π -Puussa

- Haut lähtevät liikkeelle Π -puun juurisolmusta. Tullessaan solmuun haku tutkii sisältyykö haettava piste solmun sisältämään avaruuteen. Jos sisältyy jatketaan hakua solmun lapsisolmusta ja jollei niin siirrytään sivuosoittimien kanssa sisarsolmuun niin kauan kunnes sopiva avaruus löytyy.

- Koska Π -puussa voidaan vapauttaa solmuja, käyttää haku lukkojen kytkentää. Solmun pitämää lukkoa ei vapauteta ennen kuin seuraava solmu (lapsi tai sisar) pyytää lukkoa.

- Jos puun jollain tasolla tehdään päivitystä päivityksen kanssa samalla tasolla olevat solmut lukitaan eksklusiivisella lukolla. Muilla puun tasoilla, joilla ei tehdä päivitystä lukot ovat jaettuja lukkoja.

- Kun haetaan yhtäsuuruusehdon täyttävää solmua, löytyy yksikäsitteinen polku juuresta lehtisolmuun. Haku muistuttaa tällöin B^{link} -puussa tehtyä hakua.

- Kun haetaan arvovälejä, kullakin tasolla käydään useissa lapsisolmuissa. Näitä solmuja ovat ne, joiden sisältämä avaruus leikkaa annettua hakuavaruutta. Lopulta kun ollaan seurattu useita hakupolkuja löydetään kaikki datasolmut, jotka kuuluvat haettuun arvoväliin.

Solmujen jako ja indeksivälin ilmoitus Π -Puussa

- Π -puun solmu jaetaan kun insertointi aikaansaa sen täyttymisen. Osa solmun sisällöstä siirtyy uuteen sisarussolmuun ja indeksiväli joka kuvaa syntyneiden avaruuksien hajaantumisen lähetetään jakaantuneen solmun vanhemmalle. Π -puun solmun jako on samanlainen kuin B-puun solmun jako.

- Π -puussa solmun jako ja indeksivälin ilmoitus suoritetaan kahtena erillisenä atomisena ja toipuvana toimintona seuraavasti:

- **Solmun jako:** Jako tapahtuu kun Π -puun solmu on täynnä eikä voi enää ottaa vastaan lisää tietoa. Tällöin solmu, jota kutsutaan kontaineriksi (container) jaetaan ja osa sen sisällöstä siirretään uuteen, erotettuun solmuun (extracted). Solmun jako päättyy siihen kun sisaruksen arvo talletetaan kontaineriin. Vain kontainerisolmu tarvitsee lukita (eksklusiivisella lukolla) solmun jaon ajaksi.

- **Indeksivälin ilmoittaminen:** Indeksiväli, joka kuvaa sitä avaruutta joka erotettiin kontainerista, lähetetään kontainerin vanhemmalle. Indeksiväli ilmoitetaan aina vain yhdelle vanhemmalle kerrallaan ja se on atominen toimenpide. Vain vanhemman solmun täytyy olla suojattu kirjoituslukolla.

- Solmun jako toimii samalla tavalla sekä indeksisolmuille että datasolmuille.

- (esimerkki, kuva 1)

Solmujen yhdistäminen Π -Puussa

- Π -puun solmu, jonka tilankäyttö putoaa alle tietyn ennaltamääritellyn rajan pitää yhdistää toisen solmun kanssa, jotta tilankäyttö pysyy tehokkaana.
- Π -puussa erotetun solmun sisältö siirretään aina kontainersolmuun ja erotettu solmu vapautetaan. Jotta siirto voidaan tehdä pitää kontainerissa olla tarpeeksi tyhjää tilaa ja seuraavat kaksi muuta lisäehtoa olla voimassa:
- a.) Sekä kontainerin että erotetun solmun pitää olla saman vanhemman lapsia.
- b.) Erotetulla solmulla ei saa olla kuin yksi vanhempi.
- Nämä ehdot yksinkertaistavat solmjen yhdistämistä ja lisäävät rinnakkaisuutta, sillä vain yhtä vanhempisolmua tarvitsee päivittää yhdistämisen aikana.
- Yhdistämisen aikana vain vanhempi ja kaksi lasta ovat lukittuina.
- (esimerkki, kuva 1)

Π -Puiden rinnakkaisuus ja toipuminen

- Atominen toiminto Π -puussa pitää enintään kolmea solmua lukittuna eksklusiivisella lukolla kerrallaan. Koska kerralla tarvitaan vain pieni määrä lukkoja ja koska atomiset toiminnot eivät riipu tietokannan transaktioista on Π -puun rinnakkaisuus tehokasta.

- Atomiset toiminnot logitetaan. Jos ennen atomisen toiminnon valmistumista tapahtuu virhe, se peruutetaan. Kokonaiset atomiset toiminnot (toiminnot joiden kaikki logitiedostot ovat levyllä) suoritetaan uudelleen toipumisen yhteydessä jos niiden tuloksia ei vielä ole kirjoitettu levylle.

- Kun Π -puun uudelleenrakennus joudutaan keskeyttämään järjestelmän virheiden takia, jää Π -puu konsistenttiin tilaan. Haut voidaan aina suorittaa käymällä läpi puuta seuraamalla sivuosoittimia.

- Π -puun kaksi instanssia voivat olla rakenteellisesti erilaisia, mutta semanttisesti samanlaisia.

hB -puun rakenne

- hB -puu (holey Brick puu) koostuu indeksi- ja datasolmuista.

- hB -puun solmut ovat 'palikoita' eli moniulotteisia suorakulmioita, tai oikeastaan reiällisiä palikoita, eli palikoita, joista pienempiä palikoita on poistettu. hB -puun solmu sisältää kd-puun, jota se käyttää indeksin ja sisarusvälin säilyttämiseen.

- Indeksisolmut ovat vastuussa k-ulotteisista aliavaruuksista. Ne sisältävät kd-puun, jota käytetään järjestämään tietoa seuraavaksi alimman tason lapsisolmuista ja niistä alueista, jotka on erotettu ja siirretty sisarsolmuille samalla indeksitasolla.

- Datasolmut sisältävät datatietueita. Datasolmut voivat sisältää myös kd-puita, jotka mahdollistavat datasolmujen omien sisärajojen määritykset.

- Jokainen solmu säilyttää kuvausta siitä avaruudesta, josta se on vastuussa. Kullakin attribuutilla on kaksi arvoa (low, high). Näitä arvoja kutsutaan hB -puun solmun rajoiksi. (esimerkki, kuva 2)

- Toisin kuin muut moniattribuutti -indeksit, jotka jakavat solmut hypertasoihin, hB -puu voi käyttää useampaa kuin yhtä attribuuttia kuvaamaan erotettuja alueita. Tämän vuoksi solmut voivat olla palikoita, joissa on reikiä jotka kuvaavat erotettuja alueita. Tämän vuoksi sitä kutsutaankin nimellä holey Brick.

- Jotta hB-puu voidaan muuttaa Π -puuksi, pitää siihen lisätä osoittimet erotettuihin solmuihin. Sen lisäksi, jotta voisimme löytää kandidaatteja solmujen yhdistämiseen tarvitsee määritellä järjestys, jossa indeksisolmut sisältävät lapsensa. Sen lisäksi pitää pystyä kertomaan onko jokin solmu yksi- vai monivanhempinen.

- (esimerkki, kuva 3)

- Tarvitaan neljä rakenteellista muutosta hB-puuhun, jotta siitä voidaan saada hB^{Π} -puu. Nämä ovat:
  - Sivuosoittimet
  - Solmun jako sen kd -puun juuressa
  - Dekoraatiot
  - Jatkuu -arvot
- **Sivuosoittimet:** (esimerkki, kuva 3)
- **Solmun jako sen kd -puun juuressa:** hB-puussa toinen alipuista jää jaettuun solmuun ja toisesta tulee uuden solmun kd -puu. Alkuperäinen kd -puun juuri katoaa eikä erotettuun solmuun ole olemassa osoitinta.
- hB^{Π} -puussa kd -puun juuri säilytetään alkuperäisessä solmussa. Uusi solmu on nyt erotettu solmu (extracted node) ja alkuperäinen solmu on kontainerisolmu (container node). Juurisolmun säilyttäminen on tärkeää, koska Π -puussa solmu joka jaetaan (kontainersolmu) jatkaa sen oman avainavaruuden säilyttämistä.
- (esimerkki, kuva 3)
- **Dekoraatiot:** Kolmas muutos koskee sitä miten hB^{Π} -puun lasten osoitteet talletetaan niiden vanhempien kd -puihin.
- Tämä muutos auttaa määrittelemään löytyykö sopivaa solmua yhdistämään vajaa solmu.
- hB -puun solmussa voi kd -puussa olla useita viittauksia samaan lapseen (esimerkiksi kuvassa 3a K:n osoite on talletettu kahdesti). Jokainen vanhemman kd-puun lehtisolmu joka viittaa suoraan sellaiseen dataan, jonka lapsisolmu sisältää, sisältää osoittimen lapseen.

- hB^{Π} -puussa taas identifioimme sen kd -puun solmun, joka on sen alipuun juuri, joka sisältää sen avaruuden josta lapsisolmu on vastuussa. Tämän jälkeen alipuun juuri merkitään lapsisolmun osoitteella. Tästä lapsisolmun osoitteesta käytetään nimeä dekoraatio ja vastaavaa alipuuta kutsutaan dekoroiduksi alipuuksi.

- Lehtisolmut sisältävät nyt jotain seuraavista kolmesta informaatiosta:
  - 1. Lapsisolmun osoite
  - 2. Sisarsolmun osoite
  - 3. Null -osoite

- Vain indeksisolmuilla voi olla dekoraatioita, koska datasolmuilla ei ole lapsia.

- Se missä järjestyksessä solmut sisältävät toisiaan (containment order) voidaan päätellä tutkimalla solmun kd -puuta.

- (esimerkki, kuva 3)

- **Jatkuu -arvot:** Jotta erotettu solmu voitaisiin tuhota sen jälkeen kun se on yhdistetty kontaineriinsa, tarvitsee tietää että se on yksivanhempinen solmu. Solmun vanhemmassa olevaa Jatkuu -arvoa käytetään tämän asian päättelyyn.

- Monivanhempisia solmuja luodaan kun hB^{Π} -puun indeksisolmu jaetaan ja dekoroimaton kd -alipuu erotetaan (kd -alipuun juuri ei siis sisällä dekoraatiota). Erotettu kd -alipuu dekoroidaan samalla dekoroinnilla kuin jaettu kd -alipuu. Jakamisen jälkeen se hB^{Π} -puun lapsisolmu joka esiintyy dekoraatiossa on monivanhempinen solmu, johon osoittavat sekä alkuperäinen solmu että uusi solmu.

- Solmujen yhdistäminen Π -puussa vaati sen, että poistettava solmu on yksivanhempinen.

- Jotta saadaan tietää onko solmu yksivanhempinen vai ei, käytetään Jatkuu -arvoa jokaisessa sivuosoittimessa. Sivuosoittimen Jatkuu -arvo voi olla joko tosi (true) tai epätosi (false) riippuen siitä onko siitä kd -puun osasta, jossa kyseinen sivuosoitin on, sivuosoitinta sisarsolmuun.

- (esimerkki, kuva 3)

Haku hB -puussa

- Haku aloitetaan hB -puun juuresta. Juuri käydään läpi kulkemalla sen kd-puun läpi. Jokaisella kd-puun solmulla on on informaatiota: atribuutti ja sen arvo. Vertaamalla tätä arvoa hakupisteen vastaavaan arvoon tiedetään kumpaan kd-puun lapsisolmuun tulee siirtyä.

- Näin päästään lopulta hB-puun datasolmuun. Datasolmu sisältää alueen, johon hakupiste kuuluu, jos sellaista löytyy koko puusta.

Solmujen jako hB -puussa

- Kun insertio aiheuttaa hB-puun data- tai indeksisolmun täyttymisen, sen kd-puuta käytetään etsimään kd-alipuu jonka koko on välillä 1/3 ja 2/3 koko solmun koosta. Uusi hB -puun solmu luodaan ja osa täyttyneen solmun sisällöstä siirretään siihen.

- Jos indeksisolmun jakoraja leikkaa sen lapsen avaruutta, tulee lapsisolmusta monivanhempinen solmu. Sen osoite talletetaan sekä originaaliin, että sisarsolmuun. Kaikki monivanhempisen solmun vanhemmat ovat samalla tasolla puussa, sillä solmun jako luo uuden solmun aina samalle tasolla puuta, jossa originaali solmu on.

hB^{Π} -puun rakenne

- hB^{Π} -puu on yhdistelmä hB -puusta ja Π -puusta. Se korjaa muutamia hB -puun puutteita ja sopivilla muutoksilla perii rinnakkaisuuden, toipumisen ja solmujen poistamisalgoritmit Π -puusta.

- hB^{Π} -puu toimii moniulotteisena ja sopii erinomaisesti moniulotteisiin sovelluksiin.

- hB^{Π} -puu on oikea puu, ei DAG. Tämän ansiosta indeksivälin ilmoitus -algoritmista saadaan yksinkertaisempi.

- hB^{Π} -puun solmun sisältämä avaruus, eli se avaruus mikä ei sisällä aliavaruuksia jotka on delegoitu sisarussolmuille, voidaan esittää erillisten suorakulmaisten alueiden unionina, jotka vastaavat niitä tietuelistoja jotka sijaitsevat solmussa. Näiden erillisten avaruuksien asettamia rajoja kutsutaa data-avaruuden rajoiksi (Data Space Boundaries, DSB).

- hB -puun ongelmana on se, että jos indeksisolmut jaetaan siten, että erotettu avaruus ja jäljelle jäävä avaruus ei vastaa datasolmujen avaruuksien unionia ilmenee virheellisiä hakutuloksia.

- Jotta hB -puussa ilmenevä vika saadaan korjattua suoritetaan hB -puun indeksisolmujen jako vain DSB:ssä.

- Kun jako tehdään vain DSB:ssä saavutetaan myös se etu, että kaikki solmut ovat silloin vain yksivanhempisia. Tämän vuoksi hB^{Π} -puu onkin oikea puu, ei DAG.

Solmujen jako hB^{Π} -puussa

- Jako dekoraatiosolmuissa (D): Jako dekoraatiosolmuissa -algoritmilla on sellainen hyvä ominaisuus, että se tekee kaikista solmuista yksivanhempisia. Algoritmi pääpiirteissään:
  - 1. Etsi dekoroitu alipuu C:stä, jonka koko on lähinnä puolikkaan solmun kokoa. Jollei löydy EXIT.
  - 2. Luo uusi solmu X, erota dekoroitu alipuu C:stä ja siirrä se X:ään.
  - 3. Korvaa erotettu alipuu C:ssä osoittimella X:ään (sivuosoitin).
- Posting the full path (fp): Menetelmä jossa indeksivälin ilmoittamista yksinkertaistetaan siten, että käydään läpi koko reitti juuresta solmuun ja ilmoitetaan kaikkien niiden kd-puun solmujen arvot vanhemmille, joita ei vielä aikaisemmin ole ilmoitettu,
- D/fp -alkoritmi, jossa kd-puu jaetaan jostain mielivaltaisista paikoista.
- A/fp -alkoritmi, jossa kd-puu jaetaan vain decorated alipuiksi.
- fp = Posting the full path, A = Arbitrary, D = Dekoroitu
- Eri alkoritmien ominaisuuksia:

Algoritmi	Puolitus	Indeksivälin puolitus	Monta vanhempaa
D/fp	Rajoittava	Vinoutunut	Ei
A/fp	Joustava	Tasapainoinen	Kyllä

Solmujen yhdistäminen hB^{Π} -puussa

- Kuten Π -puussa, vajaa hB^{Π} -puun solmu yhdistetään sisarussolmuun ja poistetun solmun vanhempaa muutetaan vastaamaan uutta tilannetta.
- Yksinkertaisuuden ja tehokkuuden vuoksi vajaa hB^{Π} -puun solmu yhdistetään aina sen kontaineriin.
- hB^{Π} -puun solmujen yhdistämiselle on kaksi ehtoa:
- 1. Vajaalla, erotetulla solmulla on sama vanhempi kuin kontainerilla.
- 2. Sekä erotettu solmu, että kontainerisolmu ovat yksivanhempisia.
- Jos nämä ehdot eivät täyty tai solmujen sisällöt eivät mahdu yhteen solmuun ei yhdistämistä tehdä.

hB^{Π} -puun suorituskyky

- Suorituskykytestauksissa on käytetty sekä tietokoneella generoitua dataa, että maantieteellistä dataa Sequoia -projektista.
- Testauksessa tutkittiin solmujen tilanhallintaa sekä hakujen tehokkuutta.
- Testitulokset osoittivat että kun hB^{Π} -puuta käytettiin yksiattribuuttisena indeksinä se oli yhtä tehokas kuin B^{+} -puu.
- hB^{Π} -puu on tehokas myös moniattribuuttisena indeksinä, koska kd-puun solmu tallettaa aina vain yhden attribuutin arvon. Siksi kd-puun solmun koko (ja samalla siis kd-puiden koko jotka ovat hB^{Π} -puussa) ei riipu indeksoitavien attribuuttien määrästä.
- Kukin hB^{Π} -puun solmuista tallettaa omat rajansa (low ja high -arvot kullekin attribuutille, jotka kuvaavat sen avaruuden, josta solmu on vastuussa). Näitä atribuuttien arvoja on 2k kappaletta k-ulotteisessa hB^{Π} -puussa. Eli kun ulottuvuuksien määrää kasvatetaan, kasvaa myös rajojen viemä tila solmussa. Tämä tila ei kuitenkaan merkityksellinen kun käytetään isoja levysivuja.