Acta Agraria Kaposváriensis (2011) Vol 15 No 3, 13-30
Kaposvári Egyetem, Állattudományi Kar, Kaposvár
Kaposvár University, Faculty of Animal Science, Kaposvár

Élelmiszer adatbázis szűrése mennyiségi megszorítások alapján logaritmikus indexeléssel

Kusper[1] G., Márien[2] Sz.
[1]Eszterházy Károly Főiskola, 3300 Eger, Eszterházy tér 1.
[2]Wit-Sys Zrt., 1036 Budapest, Árpád fejedelem útja 79.

ÖSSZEFOGLALÁS

Ebben a cikkben az IMEE (Index Maszkos Ehetem-E) algoritmust ismertetjük. Az algoritmust az eFilter projekt keretében fejlesztettük ki. A projekt célkitűzése egy olyan informatikai rendszer felállítása, amely egészségügyi adatok alapján szűri az élelmiszerek listáját. Az új algoritmus feladata eldönteni, hogy a felhasználó egy adott élelmiszert megehet-e, vagy sem. Ehhez ismerni kell a felhasználó étkezésre vonatkozó egészségügyi profilját. A profil megszorításokat tartalmaz, ezek alapján kell egy potenciálisan nagy élelmiszer adatbázisból szűrni a megehető tételeket. A cikkben közölt első algoritmus a természetes megközelítésből adódik. Ennek gyorsítására egy indexelési módszert írtunk le, amely a bitmap indexelési technika használatát teszi lehetővé. Bitmap indexet akkor érdemes használni, ha sok rekord van és az indexelt tulajdonság csak kevés értéket vehet fel. A javasolt módszer a megszorításokat alakítja át indexé, úgy, hogy csak minden a logaritmikus skálán lévő értéknél kisebb vagy egyenlő összetevő tartalmú étel lesz indexelve. Sajnos így egy megszorításnál csak megengedőbb vagy szigorúbb feltétel vizsgálható. Az általános alakú N < összetevő tartalom ≤ M megszorításhoz tartozó szigorúbb megszorítás 2^padlás(log_2(N)) < összetevő tartalom ≤ 2^padló(log_2(M)) tartozik, amit a legnagyobb tartalmazott megszorításnak nevezünk. Az IMEE algoritmusminden ugyanarra az összetevőre vonatkozó megszorítást összevon egy megszorítássá, majd ezekhez kiszámítja a legnagyobb tartalmazott megszorítást. Végül visszaadja azokat az élelmiszereket, amelyek minden megszorításnak eleget tesznek.
(Kulcsszavak: bitmap indexelés, egészségügyi profil, IMEE algoritmus)

ABSTRACT

Filtering Food Databases by Logaritmic Indeces on Quantity Constraints

G. Kusper[1], Sz. Márien[2]
[1]Eszterházy Károly College, H-3300 Eger, Eszterházy tér 1.
[2]Wit-Sys Zrt., H-1036 Budapest, Árpád fejedelem útja 79.

In this article we introduce the IMEE (May I Eat with Index Mask) algorithm, which we developed in the framework of the eFilter project. The goal of this project was to build up an information system, which can filter a food list by health data. The new algorithm has to decide whether the user may or may not eat a given food. To be able to do so we have to know the users health profile on eating. This profile contains constraints. We have to filter the potentially big food database according these constraints. The first algorithm in this article is the straightforward one. To speed up this algorithm we introduce an indexing mechanism, which enables us to use bitmap indexing. Bitmap indexing pays off if we have lots of records and the indexed entity has only a few possible values. The introduced

method can turn a constraint into an index by indexing only those foods which contain less or equal ingredients than a value on the logarithmic scale. Unfortunately, this was we can introduce only a less or more strict condition than the original one. We assign to the generic N < ingredient content ≤ M constraint the more strict one 2^ceiling(log_2(N)) < ingredient content ≤ 2^floor(log_2(M)), we call this the greatest specific mask. The IMEE algorithm works as follows: it closes up all constraints for each food ingredient, and then computes the greatest specific masks for each such constraint, and then filters those foods from the databases which fulfil all greatest contained constraints.
(Keywords: bitmap indexing, health profile, IMEE algorithm)

BEVEZETÉS

Ebben a cikkben olyan egyszerű, majd egyre kifinomultabb algoritmusokat közlünk, amelyek egy ételről, amelynek ismerjük az összetevőit, egy az összetevőkre vonatkozó megszorítás halmaz alapján eldöntik, hogy megfelelnek-e a megszorításoknak vagy sem, ehetem-e az ételt vagy sem.

A cikk fő motivációja az eFilter projekt, amelyet részletesen bemutatunk a következő fejezetben. Az eFilter projekt az EgerFood projekt továbbgondolása. Az EgerFood projektben létrehoztunk egy olyan információs rendszert (*Kusper et al*, 2007; 2008; *Liptai et al*, 2007; *Radványi et al*, 2007, 2008a, 2008b), amelynek segítségével a cégek le tudják tárolni, akár egyedi termékenként, hogy milyen összetevőkből készült az élelmiszer, az összetevőket ki szállította, mikor és hogyan használták azokat fel.

Az eFilter projekt szemszögéből ebből az a fontos, hogy néhány élelmiszerről egészen pontosan tudjuk, mit tartalmaz. Más adatforrást is használtunk az itt közölt algoritmusok vizsgálatához.

Fő forrásunk a http://www.ars.usda.gov/Services/docs.htm?docid=20959 linkről letölthető, az United States Department Of Agriculture szervezet által közölt élelmiszer adatbázis.

Az eFilter projekt bemutatása

Az eFilter projekt célja egy olyan informatikai rendszer felállítása, amely egészségügyi adatok alapján szűri az élelmiszerek listáját, amelyet a felhasználók szeretnének elfogyasztani. Itt az élelmiszer lehet egy nagyon egyszerű, pl. liszt, vagy akár egy nagyon összetett, pl. sajtos makaróni, is. Az egészségügyi adatokat egy egészségügyi profilban tároljuk. Ez tartalmazza az ételérzékenységeket, az allergiákat, diétákat és egyéb étkezésnél figyelembe veendő adatokat. Ugyanakkor ezeket az adatokat számszerűsítve tároljuk. Nem azt tároljuk, hogy a felhasználónak mogyoró allergiája van, hanem hogy a megengedett napi mogyoró bevitel 0.0 és 0.0 közt van, tehát megszorításként. Így az egészségügyi profil egy többdimenziós megszorítási mátrix.

Működési modell

Minden használati eset két alapvető működési módra vezethető vissza. A kétféle használati módot foglalja össze az *1. ábra.*

A szűrés során a bő lista minden egyes elemére az egészségügyi profilt alkalmazva eldönthető, hogy az adott termék megfelel-e a profilban megadott megszorításoknak vagy sem. Továbbá a fogyasztható termékekhez való hozzájutást segítő információk kérhetőek. Az ellenőrzés során egy termékről eldönthető, hogy megfelel-e az egészségügyi profilban megadott megszorításoknak vagy sem. Továbbá a termékkel kapcsolatos információk kérhetők le.

1. ábra

Az eFilter rendszer használati módjai

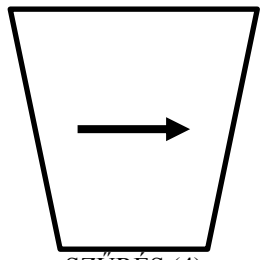
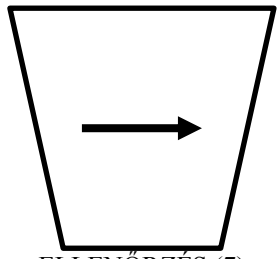
Bő lista (1)	**Egészségügyi profil(2)**	**Szűk lista (3)**
- Étkeztető szervezet menü listája - Bolt élelmiszer listája - Ételrendelésnél étlap - Hol kapható a fogyasztható termék?	 SZŰRÉS (4)	- Fogyasztható menük listája - Boltban kapható fogyasztható élelmiszerek listája - Fogyasztható ételek listája - Fogyasztható terméket áruló boltok listája
Kérdés (5)	**Egészségügyi profil**	**Válasz (6)**
- Megvásárolt élelmiszer fogyasztható-e? - Vásárlásnál megerősítés, hogy az adott termék fogyasztható-e?	 ELLENŐRZÉS (7)	- Igen / Nem + Megvásárolt élelmiszerről bő információ - Igen/Nem + Megvásárolt élelmiszerről bő információ

Figure 1: Possible usages of the eFilter system

Long list of foods (This is a list of foods. This list can be the list of available foods in a store, or the list of meals of a restaurant)(1), Health record (This record belongs to the user. It describes those constraints which limits what can the user eat)(2), Short list of foods(This is a list of foods. It contains those foods from the long list which match the constrains of the health record)(3), Filter(4), Question (The eFilter system can answer several questions: May I eat this food? Which foods may I eat from the list?)(5), Answer (The eFilter system can answer these questions by answering yes/no and by giving the list of foods the user might eat)(6), Control(7)

Adatmodell

Az egészségügyi profil tulajdonságai:

- Az egészségügyi profil lehet személyes, azaz fogyasztóhoz köthető vagy sablon, amely fogyókúra és egyéb betegségek esetén általános megkötések tételére használható;
- A személyes és a sablon egészségügyi profil is kor-, nem- és súlyfüggőek. Ezek leíró attribútumként az egészségügyi profil entitás részei;
- Az egészségügyi profilok strukturálhatók, azaz a már meglévő profilokból és azok szabályaiból építkezni lehet. A meghatározott egészségügyi profil sablonok felhasználhatók személyes profilok esetén. Például egy előre meghatározott betegséghez kötött egészségügyi profil beköthető egy személyes profilhoz, amelyet az adott beteg egyedi szabályaival tovább lehet finomítani.

Egészségügyi profil szabály:

- Megszorítások, amelyek irányai a következők lehetnek:
  - tiltás,
  - nem javasolt,
  - javasolt,
  - erősen javasolt;
- amelyek megadása mennyiségi korláttal történik. Egy megszorítás esetén, ha a mennyiség nem nulla, akkor csak egy megadott időszakra vonatkozhat, mivel akkor a fogyaszthatósággal kapcsolatos döntés csak az időszakban elfogyasztott mennyiséggel felhasználva születhet meg. (Nincs értelme egy konkrét döntésnél időszak és fogyasztás nélkül olyan szabályt felhasználni, ahol a mennyiség nem nulla, mivel ez felhasználás szempontjából értelmezhetetlen.);
- Betegséghez tartozó egészségügyi profil sablon esetén az adott betegség étkezési megkötései szerepelnek a szabályok között;
- Egy szabály típusai, azaz irányai és prioritása:
  - 0 – Tiltás,
  - 1 – Nem javasolt,
  - 2 – Erősen javasolt,
  - 3 – Javasolt;
- Idő-mennyiség mátrixot is tárolunk az egészségügyi profil szabályaiban, azaz lehessen megadni egy profilnál nem csak azt, hogy milyen terméket ehet a felhasználó, hanem azt is, hogy az adott megkötés milyen időszakra vonatkozik;
- Mennyiség és időszak megadásának csak együtt van értelme. Mennyiség megadása időszak nélkül értelmezhetetlen, hiszen nem adja meg a fogyasztás időtartományát. Időszak mennyiség nélküli megadása szintén értelmetlen, mivel mennyiség megadása nélkül nincs értelme az időszaki fogyasztás követésének;
- Egy szabály vonatkozhat termék kategóriára, termék elemre és termékre.

Biztosítani kell az egészségügyi szabályok integritását, azaz a következő ellenőrzéseket a szabályok deklarálásakor figyelembe kell venni:

- Nem lehet egy termékre két szabály: Ha egy termékre meghatároztunk egy szabályt, ami után egy másik szabályt is szeretnénk megszabni, akkor a két szabály típusának prioritása meghatározza, hogy melyiket van értelme megtartani, a „gyengébbet” el kell hagynunk. Ha például valaki nem javasolt, hogy túrót egyen, és jön egy új szabály, miszerint tilos neki túrót ennie, akkor az erősebb tiltó szabály alkalmazásának van értelme;
- Ha egy szabály egy termék kategóriára / termék összetevőre / termék elemre vonatkozik, akkor csak erősebb szabályt deklarálhatunk az adott kategóriában lévő termékre / terméket tartalmazó termékekre / termék elemet tartalmazó termékre.

Menü:

- Egy ütemezett, konkrét étrend. Annyival több, mint az egészségügyi profil, hogy itt nem csak megszorításokat teszünk egy adott időszakra, hanem konkrét étrend javaslatot teszünk.

AZ EHETEM-E ALGORITMUS

Ebben a fejezetben bevezetjük az EHETEM-E algoritmust, amely az inputként megadott ételről eldönti, hogy az tiltott, nem javasolt, javasolt, vagy nem tiltott az input egészségügyi profil alapján.

Egy konkrét példa egészségügyi profilra:

- Tiltások: dió > 0g,
- Nem javasolt: energiatartalom > 500 kcal, zsír > 20g, só > 2g,
- Erősen javasolt: üres,
- Javaslat: üres.

Ez a profil azt írja le, hogy diót nem szabad enni, olyan ételt, ami nagyon zsíros (zsír > 20g) vagy sós (só > 2g), azt nem javasolt enni. Megjegyzés, hogy ez a vagy a logikában szokásos megengedő vagy (or), nem a mindennapi nyelvben használt kizáró vagy (xor). Az erősen javasolt és a javaslat ételek listája üres. Látható, hogy a lista elemei közt vagy kapcsolat van. A tiltások általában ételérzékenységekből, allergiákból következnek. A nem javasolt ételek megszorításai hosszú távú kockázatok csökkentése miatt írhatja elő az orvos. Nagyon ritkán lehet olyan betegségünk, amikor valamilyen ételt ennünk kell. Ennek két fokozata az erősen javasolt és a javasolt.

A fenti profilt úgy kell értelmezni, hogy a szabályok az étel 100g-jára vonatkoznak. Ha nincs külön kikötve egy szabálynál, akkor az mindig az étel 100g-jára vonatkozik.

Az első szintű szabályellenőrzésnél nincs idő fogalmunk, tehát nem tudjuk azt mondani, hogy napi maximális zsír bevitel nem több 20g-nál, csak azt, hogy minden vizsgált tételnél a zsírtartalom nem több 20g-nál.

Nézzük az első szintű ellenőrzést végző algoritmust, ami egy egyszerű összegzésen alapszik:

Algoritmus: EHETEM-E

- Input: Étel, Egészségügyi profil.
- Output: Tiltott / nem javasolt / javasolt / nem tiltott.

1. lépés: Minden az egészségügyi profilban szereplő összetevőhöz készítsünk lokális változókat, kezdő értékük nulla. A fenti példát használva:
   $dió = 0, $energiatartalom = 0, $zsír = 0, $só = 0;
2. lépés: Ha az ételnek megvannak azon adatai, amiket gyűjtünk, akkor hozzáadjuk a változóhoz. Amelyek nincsenek meg a felső szinten, azokat az összetevők alapján számítjuk rekurzívan lefelé haladva, amíg nem találjuk az adott értéket. Ha nincs egyik összetevőben sem a kérdéses érték, akkor feltételezzük, hogy nincs az ételben. A figyelembe vett értéknél mindig 100g-ra kell visszaszámolni.

Pl.: Vegyük a diós kalács (zsír: 10g, fehérje: 1g, szénhidrát: 96g, liszt: 80g, dió: 10g, vaj: 10g) példáját. Ennek felső szinten nincs megadva az energiatartam és a só, de van dió és zsír. Így a $dió = 10 és a $zsír = 10g számítása készen is van. A többi érték számításához a diós kalács összetevőinél megadott értékeket kell összeadni. Tegyük fel, hogy ezekből áll a diós kalács: liszt (energiatartalom: 1500kJ, zsír: 0g, szénhidrát:98g, fehérje: 1g), dió (energiatartalom: 800kJ, zsír: 50g, szénhidrát: 0g, fehérje: 50g), vaj (energiatartalom: 2500kJ, zsír: 80g, szénhidrát: 0g, fehérje: 20g). Ebből számítható az energiatartalom:
$energiatartalom = 80/100 * 1500 + 10/100 * 800 + 10/100 * 2500.

Ez a szám úgy jön ki, hogy 100g diós kalácsban 80g liszt van és 100g lisztben 1500 kJ energia, stb… . Mivel só ezeknél sincs megadva, ezért ezeket a termékeket is rekurzívan tovább kell nézni, de csak a só tartalom után kutatva, hiszen a többi értéket már sikerült meghatározni magasabb szinten.

1. lépés: Ha van olyan feltétel a tiltó listán, amely igaz, akkor az algoritmus válasza „tiltott”, egyébként.
2. lépés: Ha van olyan feltétel a nem javasolt listán, amely igaz, akkor az algoritmus válasza „nem javasolt”, egyébként.

3. lépés: Ha javasolt lista minden feltétele igaz és maga a javasolt lista nem üres, akkor az algoritmus válasza „javasolt”, egyébként a válasz „nem tiltott”.

Vegyük észre, hogy a 3-5 lépés egy if – else if szerkezet ágai, ami meghatározza az algoritmus visszatérési értékét. Ahhoz, hogy az étel „javasolt” legyen, ehhez nem lehetett „tiltott” és „nem javasolt”, illetve a „javasolt” lista minden feltételét is ki kell elégítenie az ételhez tartozó értékeknek.

Könnyen észrevehető az algoritmus első javítása. A példában a második lépésben nem érdemes kiszámolni az étel energia tartalmat, hiszen már korábban tudjuk a dió tartalmat, ami pozitív, így kielégíti a $dió > 0 tiltó feltételt. Tehát elegendő addig számolni a változók tartalmát, amíg az első tiltó feltétel igazzá nem válik, illetve a tiltó listán kívüli változókat nem érdemes számolni, ha már valamely nem javasolt feltétel igaz lett. Ugyanekkor a tiltott listás változókat tovább kell számolni, hiszen lehet, hogy végül tiltott lesz az étel. A javított algoritmus:

Algoritmus: EHETEM-E Javított 1.

- Input: Étel, Egészségügyi profil.
- Output: Tiltott / nem javasolt / javasolt / nem tiltott.

1. lépés: Minden az egészségügyi profilban szereplő összetevőhöz készítsünk lokális változókat, kezdő értékük nulla.
2. lépés: A tiltott listán lévő változók értékét számoljuk. Ha az étel felső szintjén adott az érték, akkor azt másoljuk be a változóba, ha nem, akkor rekurzívan felbontjuk az ételt összetevőire és az ott megadott értékeket (visszaszámolva 100g ételre) adjuk hozzá a változóhoz. Minden rekurzív hívás előtt ellenőrizzük, hogy valamely tiltó feltétel igaz-e. Ha igaz, akkor az algoritmus válasza „tiltott”. Egyébként rekurzívan bontjuk az élelmiszert, amíg kell, vagy amíg lehet.
3. lépés: Hasonlóan kiszámítjuk minden nem javasolt listán lévő változó értékét (feltéve, hogy nem számoltuk már ki a tiltottak közt). Ha valamelyik nem javasolt feltétel igazzá válik a folyamat során, akkor az algoritmus válasza „nem javasolt”.
4. lépés: Hasonlóan kiszámítjuk minden javasolt listán lévő változó értékét (persze csak azokat, amiket még nem számoltunk ki). Miután minden változót kiszámoltunk megnézzük, hogy minden javasolt feltétel igaz-e. Ha igen és a javasolt lista nem üres, akkor az algoritmus válasza „javasolt”.
5. lépés: Az algoritmus válasza „nem tiltott”.

Látható, hogy a 3. lépésre csak akkor jutunk el, ha a második lépésben nem találtunk tiltó feltételt. Hasonló állítás fogalmazható meg a következő lépésekről. Tehát ez az algoritmus lusta abban az értelemben, hogy egy étel valamely értékét (pl. energiatartalom), csak akkor számolja ki, ha az szükséges.

Ugyanakkor a feltételeket intenzíven ellenőrzi, hiszen az étel minden felbontása előtt minden tiltó vagy nem javasolt, stb. feltételt ellenőriz (attól függően, melyik lépésben járunk). Ha sok, bonyolult feltétel van, akkor ez a megoldás rossz!

Egy egyszerű kompromisszum, hogy csak a teljesen kiszámolt értékekre épülő feltételeket értékeljük ki. Ekkor előfordulhat, hogy feleslegesen pontosan számítunk ki valamit, de a feltétel ellenőrzések száma nem az étel összetettségével arányos.

Másik megoldás, hogy a feltételek kiértékelésével bánunk lustán. Érdemes az egy összetevőre vonatkozó több feltételt összevonni és megnézni, hogy ellentmondásos-e. Lehet, hogy eleve ellentmondásos a feltétel rendszer, például: Tiltások: $energiatartalom > 2000kJ, $energiatartalom < 2500kJ. Azaz nem ehetünk semmit, amiben 2000kJ-nál több az energia; illetve nem ehetünk semmit, amiben nincs legalább 2500kJ energia. Ez

a feltétel rendszer önellentmondásos, hiszen vagy az egyik vagy a másik feltétel mindig igaz lesz.

SZŰRÉS INDEXELÉSSEL

Adatbázisok esetén bevett gyakorlat, hogy indexek segítségével gyorsítjuk a lekérdezéseket. Azokra az oszlopokra, amelyekre feltételek vonatkoznak, azokra érdemes indexet tenni, mert ez néha nagyságrendekkel gyorsíthatja a keresést. Ugyanakkor az indexek tárigényesek, illetve karbantartást igényelnek, de ezt a mai adatbázis-kezelők elrejtik előlünk.

Természetes ötlet, hogy az egészségügyi profil alapján végzett élelmiszerszűréshez is használjuk fel az indexeket. Ebben a fejezetben nem feltételezzük adatbázis használatát, de leírtak alapján könnyen létrehozható adatbázis-kezelőkben is a megfelelő indexek.

Az ilyen fajta indexelést a szakirodalom bitmap indexnek nevezi.

Elméleti háttér

Az adatkezelő rendszerek egyik hatékony indexelési technikája a bitmap indexek alkalmazása. A bitmap index struktúrát 1985-ben vezették be (*Spiegler et al*, 1985) az összetett feltételeknek megfelelő rekordok hatékony keresése céljából, néhány javítás után (*Chan et al*, 1998) széles körben elterjedt. Az alap bitmap index egy bittérképet jelent, ahol a kulcs minden értékének megfelel egy sor a mátrixban és minden rekord egy oszlophoz tartozik. A 1-es érték jelenti, hogy a rekord megfelel a sort jelentő feltételnek. A feltétel lehet egyezőség a kulccsal vagy az, hogy nagyobb, mint a kulcs (tartomány bitmap). A bitmap index előnyei:

- kis értéktartomány esetén tömör a tárolás,
- hatékonyság az összetett feltételek kezelése esetén (bitműveletekre visszavezethető feltételek esetén).

Egy K méretű értéktartományú és N rekordot tartalmazó minta esetén a bitmap indexben történő keresés blokk-költsége:

$$O(N/m) \quad (1)$$

ahol m az egy blokk tárolási egységen jegyzett elemek darabszáma. Ha egy klasszikus B-fa indexet veszünk, akkor ott

$$O(\log(kN)) \quad (2)$$

költséget kapunk. Ha m kicsi, pl. 0.001, akkor kb. 20000 rekordnál már a B-fa index hatékonyabban működne, hiszen a bitmap index mérete is lineárisan nő. A méret okozta probléma megoldáséra több módszer is kidolgozásra került. A legelterjedtebb megoldások:

- a bitmap tömörítése, például a WAH (*Wu et al*, 2006) módszer RLE kódolást használ,
- a kulcstartomány reprezentálás csökkentése (a kulcsérték közvetlen ábrázolása helyett a komponenseit ábrázolják).

Mivel a teljes bitmap mérete és a keresés is függ a kulcsok darabszámától, a bitmap index csak kis értékkészletű kulcsoknál célszerű használni. Egy másik szempontból pedig azt is figyelembe kell venni, hogy az indexen keresztüli elérés többlet költséget jelent a közvetlen SCAN művelettel szemben. Ezért kis méreteknél a SCAN eljárás hatékonyabb. A bitmap indexek fő ereje akkor jelenik meg, amikor

- nagy rekordszám, de kevés különböző kulcs,
- egy mezőre több feltétel is megjelenik.

Az Oracle RDBMS rendszerek implementált bitmap index (Oracle Bitmap Index Techniques: http://www.dba-oracle.com/oracle_tips_bitmapped_indexes.htm) is tömörített formában tárolja az index adatokat. A mérési eredmények alapján a keresési költség lineárisan arányos a kulcsok darabszámával. Kis táblaméreteknél a CBO optimalizáló motor a SCAN módszert alkalmazza.

Indexelés

A fejezet alap ötlete az, hogy egy mennyiségi megszorítás ellenőrzését vezessük vissza egyszerű ellenőrzésre, hogy az elfogyasztandó étel indexelt-e vagy sem.

Vegyünk egy egyszerű megszorítást: 1g < fehérje tartalom ≤ 4g, azaz legalább 1g-nál több fehérjét ennünk kell, de legfeljebb 4g-ot. Tegyük fel, hogy létezik 4 indexünk minden az adatbázisban meglévő élelmiszerhez. Egy élelmiszer indexelt a 0. index által, ha egyáltalán nem tartalmaz fehérjét. Az 1. index a 1g-nál kevesebb, vagy pont 1g fehérjét tartalmazó ételeket indexelje. A 2. index „≤ 2g” fehérje tartalomnak megfelelőeket indexeli. A 3. index a „≤ 4g” feltétel alapján indexel. Természetesen nem csak a „≤” bináris összehasonlító művelet alapján lehet indexelni, hanem bármilyen bináris összehasonlító művelettel. A különböző indexek megkülönböztetésére az index első helyére a használt összehasonlító műveletet írjuk.

Az az étel, amiben 1.1g fehérje van, az (≤,0,0,1,1) index négyessel bír, mert nem igaz, hogy nincs benne fehérje (0. index tehát 0), az se igaz, hogy 1g vagy kevesebb fehérje van benne (1. index tehát 0), de az már igaz, hogy 2g-nál, illetve 4g-nál kevesebb vagy egyenlő fehérje van benne (2. és 3 index tehát 1). Az az étel, amiben 4g fehérje van, az (≤,0,0,0,1) index négyessel bír.

Ebből a két megállapításból adódik az összetett feltétel (1g < fehérje tartalom ≤ 4g) leírása, 1g-nál nem kevesebb vagy egyenlő, de 4g-nál kevesebb, vagy egyenlő, ami így írható le: (≤,x,0,x,1), ahol x szimbólum azt jelöli, hogy a megszorítás vizsgálata szempontjából az x helyén lévő érték irreleváns. Azaz csak azt kell megnézni, hogy a 1. index 0 értékű-e, illetve a 3. index 1 értékű-e. Látható, hogy a megszorítás ellenőrzést így maximum 2 index vizsgálatra tudjuk redukálni.

A fenti példában bemutatott (≤,x,0,x,1) index n-est a továbbiakban **maszk**nak nevezzük. Nézzük néhány fehérje tartalom maszkját:

- 0 fehérje tartalom maszkja: (≤,1,1,1,1);
- 0.5 fehérje tartalom maszkja: (≤,0,1,1,1);
- 1 fehérje tartalom maszkja: (≤,0,1,1,1);
- 1.5 fehérje tartalom maszkja: (≤,0,0,1,1);
- 2 fehérje tartalom maszkja: (≤,0,0,1,1);
- 2.5 fehérje tartalom maszkja: (≤,0,0,0,1);
- 3 fehérje tartalom maszkja: (≤,0,0,0,1);
- 3.5 fehérje tartalom maszkja: (≤,0,0,0,1);
- 4 fehérje tartalom maszkja: (≤,0,0,0,1);
- 4.5 fehérje tartalom maszkja: (≤,0,0,0,0).

A fenti példából jó látszik, hogy balról jobbra nézve az első 1 után már mindig 1 jön, illetve jobbról balra nézve az első 0 után már mindig 0 jön. Így a következő maszkokat lehet megadni (a maszkban az x szimbólum azt jelöli, hogy a helyén lévő érték irreleváns, ha több x is van a maszkban, akkor a helyeiken lévő értékek különbözőek is lehetnek, tehát az x csak helyfoglaló):

- (≤,1,1,1,1) = (≤,1,x,x,x) – fehérje tartalom ≤ 0g, azaz, fehérje tartalom = 0g, azaz, nyomokban sem tartalmaz fehérjét;
- (≤,x,1,1,1) = (≤,x,1,x,x) – fehérje tartalom ≤ 1g;

- (≤,x,x,1,1) = (≤,x,x,1,x) – fehérje tartalom ≤ 2g;
- (≤,x,x,x,1) = (≤,x,x,x,1) – fehérje tartalom ≤ 4g;
- (≤,0,x,x,x) = (≤,0,x,x,x) – fehérje tartalom > 0g;
- (≤,0,0,x,x) = (≤,x,0,x,x) – fehérje tartalom > 1g;
- (≤,0,0,0,x) = (≤,x,x,0,x) – fehérje tartalom > 2g;
- (≤,0,0,0,0) = (≤,x,x,x,0) – fehérje tartalom > 4g;
- (≤,0,1,1,1) = (≤,0,1,x,x) – 0g < fehérje tartalom ≤ 1g;
- (≤,0,x,1,1) = (≤,0,x,1,x) – 0g < fehérje tartalom ≤ 2g;
- (≤,0,x,x,1) = (≤,0,x,x,1) – 0g < fehérje tartalom ≤ 4g;
- (≤,0,0,1,1) = (≤,x,0,1,x) – 1g < fehérje tartalom ≤ 2g;
- (≤,0,0,x,1) = (≤,x,0,x,1) – 1g < fehérje tartalom ≤ 4g;
- (≤,0,0,0,1) = (≤,x,x,0,1) – 2g < fehérje tartalom ≤ 4g.

Ebben a cikkben **pozitív index**nek nevezzük a ≤ összehasonlítást használó indexeket, és **negatív index**nek a > összehasonlítást használókat. Könnyen belátható, hogy a (≤,1,1,1,1) és a (>,0,0,0,0) maszk ugyanazt a megszorítást kódolja. Hasonlóan egyszerű belátni általánosan is, hogy a bitek és az összehasonlítás megfordításával a pozitív maszkból azt a negatív maszkot kapjuk, ami ugyanazt a megszorítást kódolja. Ez fordítva is igaz.

Ha egy maszkkal pontosan tudunk leírni egy megszorítást, az inkább véletlen, mert sokkal több olyan eset van, amikor ez nem lehetséges. Pl. a 2.8g >= fehérje tartalom > 2.5g feltételt csak közelíteni tudjuk a (≤,x,x,0,1) maszkkal, ami a 4g >= fehérje tartalom > 2g feltételnek felel meg. Ez a megszorítás tartalmazza az eredetit, ezért **tartalmazó megszorítás**nak nevezzük.

Ez alapján a **legkisebb tartalmazó maszk (LKTM)** egy olyan maszk, ami tartalmazó megszorítást ír le és nincs olyan maszk, amelyhez nála kisebb intervallumot és tartalmazó megszorítást írna le. Könnyen belátható, hogy minden „M >= összetevő tartalom > N” alakú megszorításhoz, amelynél M (azaz a felső határ) kisebb, mint a legnagyobb indexelt érték, adható legkisebb tartalmazó maszk.

Hasonlóan értelmezhető a **legnagyobb tartalmazott maszk (LNTM)** is, de az csak akkor adható meg, ha az eredeti megszorítás felső határánál létezik kisebb indexelt érték, az alsó határnál pedig nagyobb és a kettő nem ugyan az. Így például a 2.8g >= fehérje > 2.5g megszorításhoz nem létezik legnagyobb tartalmazott maszk. A következő példánál viszont létezik: A 4.8g >= fehérje tartalom > 1.5g megszorításhoz tartozó legnagyobb tartalmazott maszk a (≤,x,x,0,1), ami a 4g >= fehérje tartalom > 2g megszorítást kódolja. Ha a megszorítás „összetevő tartalom > N” alakú vagy „M >= összetevő tartalom” alakú, és M illetve N kisebb, mint a legnagyobb indexelt érték, akkor mindig megadható a legnagyobb tartalmazott maszk.

A legkisebb tartalmazó maszk az eredetinél megengedőbb vagy egyenlő, azaz több vagy ugyanannyi ételre igaz. A legnagyobb tartalmazott maszk az eredetinél szigorúbb vagy egyenlő megszorítást ír le, azaz kevesebb vagy ugyanannyi ételre igaz.

Indexelés logaritmikus index sűrűséggel

Már a fenti példában is logaritmikus indexelést használtunk, habár ez csak egy véletlen választásnak tűnt. A kérdés az, hogyan érdemes megválasztani az index határokat, úgy, hogy az eredeti megszorításnál csak kicsit megengedőbb, vagy csak kicsit szigorúbb megszorításokat tudjunk velük ellenőrizni, ugyanakkor ne legyen túl sok index se.

Jelen cikkben az a javaslatunk, hogy minden kettő hatványra tegyünk indexet. A mértékegység összetevőnként lehet más és más. A fehérjénél ez lehet gramm, de mogyorónál, ahol már kis mennyiség is túlzott reakciót válthat ki, a mértékegység lehet

akár milligramm. Tehát a következő pozitív vagy negatív indexek kerülnek ki minden összetevőre (a mértékegység helyette egység szerepel):

- 0. index: Az élelmiszer indexelt, ha az összetevő tartalma nagyobb, mint 0 egység;
- 1. index: Az élelmiszer indexelt, ha az összetevő tartalma nagyobb, mint 1 egység;
- 2. index: Az élelmiszer indexelt, ha az összetevő tartalma nagyobb, mint 2 egység;
- …
- n. index: Az élelmiszer indexelt, ha az összetevő tartalma nagyobb, mint 2^(n-1) egység.

Ha az egység milligramm, akkor a 20. index 0.524288 kg-nál nagyobb összetevő tartalmat indexel. Ennek az indexelésnek az az előnye, hogy milligrammos tartományban nagyon pontosan, ahol ez el is várt, ahol pedig nem szükséges a nagy pontosság, azaz a kg-os kategóriában, ott csak hozzávetőleges. Az indexelés pontosságát az indexelés hibája alfejezetben tárgyaljuk.

Sok összetevőre valószínűleg nem kell 20 index, hiszen, ha pozitív indexelés esetén az index feltételének egy élelmiszer sem felel meg, akkor az az index felesleges. Illetve negatív indexelés esetén, ha minden élelmiszer megfelel neki, akkor is felesleges.

Nézzük néhány szám pozitív és negatív maszkját:

- 0 negatív és pozitív maszkja: (>,0,0,0,0), (≤,1,1,1,1);
- 1 negatív és pozitív maszkja: (>,1,0,0,0), (≤,0,1,1,1);
- 2 negatív és pozitív maszkja: (>,1,1,0,0), (≤,0,0,1,1);
- 3 negatív és pozitív maszkja: (>,1,1,1,0), (≤,0,0,0,1);
- 4 negatív és pozitív maszkja: (>,1,1,1,0), (≤,0,0,0,1);
- 5 negatív és pozitív maszkja: (>,1,1,1,1), (≤,0,0,0,0);
- 6 negatív és pozitív maszkja: (>,1,1,1,1), (≤,0,0,0,0).

A következő példában a négyzetes lépésközt és negatív indexelést használva adjuk meg a 4 széles maszkok jelentését, feltéve, hogy az index a fehérjetartalmat indexeli:

- (>,0,0,0,0) = (>,0,x,x,x) = (≤,1,x,x,x) – fehérje tartalom = 0 egység;
- (>,x,0,0,0) = (>,x,0,x,x) = (≤,x,1,x,x) – fehérje tartalom ≤ 1 egység;
- (>,x,x,0,0) = (>,x,x,0,x) = (≤,x,x,1,x) – fehérje tartalom ≤ 2 egység;
- (>,x,x,x,0) = (>,x,x,x,0) = (≤,x,x,x,1) – fehérje tartalom ≤ 4 egység;
- (>,1,x,x,x) = (>,1,x,x,x) = (≤,0,x,x,x) – fehérje tartalom > 0 egység;
- (>,1,1,x,x) = (>,x,1,x,x) = (≤,x,0,x,x) – fehérje tartalom > 1 egység;
- (>,1,1,1,x) = (>,x,x,1,x) = (≤,x,x,0,x) – fehérje tartalom > 2 egység;
- (>,1,1,1,1) = (>,x,x,x,1) = (≤,x,x,x,0) – fehérje tartalom > 4 egység;
- (>,1,0,0,0) = (>,1,0,x,x) = (≤,0,1,x,x) – 0 egység < fehérje tartalom ≤ 1 egység;
- (>,1,x,0,0) = (>,1,x,0,x) = (≤,0,x,1,x) – 0 egység < fehérje tartalom ≤ 2 egység;
- (>,1,x,x,0) = (>,1,x,x,0) = (≤,0,x,x,1) – 0 egység < fehérje tartalom ≤ 4 egység;
- (>,1,1,0,0) = (>,x,1,0,x) = (≤,x,0,1,x) – 1 egység < fehérje tartalom ≤ 2 egység;
- (>,1,1,x,0) = (>,x,1,x,0) = (≤,x,0,x,1) – 1 egység < fehérje tartalom ≤ 4 egység;
- (>,1,1,1,0) = (>,x,x,1,0) = (≤,x,x,0,1) – 2 egység < fehérje tartalom ≤ 4 egység.

A fenti táblázatban megadtuk mind a negatív, mind a pozitív maszkot, ami leírja a megszorítást. Látható, hogy mindig két értéket kell csak vizsgálni a maszkból. A legelső negatív index maszk azt mutatja meg, hogy mi az x jelentése a második negatív alakban, azaz, a második negatív alakban lévő 1 előtt csak 1 állhat, illetve a 0 után csak 0, a kettő közt állhat 0 és 1 is.

Elképzelhető, hogy technikai megszorítások miatt csak azt tudjuk ellenőrizni, hogy a maszkban van-e 1, azt nem, hogy van-e 0. Ilyen esetben használni kell a negatív és a pozitív indexelést is. Mivel a kettőben a bitek pont átfordított helyzetben vannak, mint a

másikban, ezért például a negatív maszkban lévő 0 meglétét úgy tudom ellenőrizni, hogy megnézem, hogy ugyanazon a helyen a pozitív maszkban 1 van-e. Hasonlóan ellenőrzöm a pozitívban lévő 0 meglétét a negatív index segítségével.

A fenti trükköt kell alkalmaznunk akkor, ha a maszkot egy 32 bites (vagy ahány bites processzorral dolgozunk) bit sorozatként, azaz egy gépi szó típusú számként, tároljuk. Ilyenkor a K.-dik helyen lévő 1 meglétét úgy ellenőrizzük, hogy a szám >= 2^(K+1)-1 feltétel igaz-e, feltéve, hogy a számban a bitek sorrendje épp fordított, mint a maszkban, illetve negatív indexelést használunk. A tesztelés során pontosan ezt a megközelítést használtuk, annyi különbséggel, hogy nem fordítottuk meg a bitek sorrendjét, ami így utólag nézve okozott néhány nehézséget, de alapvetően nem befolyásolta a teszt hatékonyságát.

Az LNTM és az LKTM kiszámítása logaritmikus index sűrűség esetén

Már általánosan megadtuk a legnagyobb tartalmazott maszk (LNTM) és a legkisebb tartalmazó maszk (LKTM) definícióját. Ebben a részben azt mutatjuk meg, hogy logaritmikus index sűrűség esetén hogyan kell megadni ezt a két maszkot.

Az *1. táblázat* három példát és egy általános példát tartalmaz ezek kiszámítására.

1. táblázat

Példák a legnagyobb tartalmazott maszk (LNTM) és a legkisebb tartalmazó maszk (LKTM) kiszámítására

Megszorítás (1)	LNTM (2)	LKTM (3)
0.5g <fehérje tartalom (4) ≤3.5g	1g < fehérje tartalom ≤ 2g (≤,x,0,1,x)	0g < fehérje tartalom ≤ 4g (≤,0,x,x,1)
1.5g < fehérje tartalom ≤3.5g	2g < fehérje tartalom ≤ 2g Nem létezik LNTM (5)	1g <fehérje tartalom ≤ 4g (≤,x,0,x,1)
0.5g < fehérje tartalom ≤0.8g	1g < fehérje tartalom ≤ 0g Nem létezik LNTM	0g < fehérje tartalom ≤ 1g (≤,0,1,x,x)
N g < fehérje tartalom ≤ M g	N-nél nagyobb következő indexelt érték < fehérje tartalom ≤ M-nél kisebb indexelt érték (6)	N-nél kisebb következő indexelt érték < fehérje tartalom ≤ M-nél nagyobb indexelt érték (7)
N g < fehérje tartalom ≤ M g, és N>1 és M>1	2^padlás($\log_2$(N)) g < fehérje tartalom ≤ 2^padló($\log_2$(M)) g (8)	2^padló($\log_2$(N)) g < fehérje tartalom ≤ 2^padlás($\log_2$(M)) g (9)

Table 1: Examples how to calculate the greatest specific mask (GSM) and the least general mask (LGM)

Constraint(1), GSM(2), LGM(3), Protein content(4), There is no GSM(5), The next largest indexed value of N < protein content ≤ the next smallest indexed value of M (6), The next smallest indexed value of N < protein content ≤ the next largest indexed value of M (7), 2^ceiling(log2(N)) g < protein content ≤ 2^floor(log2(M)) g (8), 2^floor(log2(N)) g< protein content ≤2^ ceiling(log2(M)) g (9)

Ezek alapján az általános képlet a két maszk kiszámítására logaritmikus indexsűrűség esetén a következő, ahol N és M egy „N egység < összetevő tartalom ≤ M egység” alakú megszorítást írnak le:

- LNTM(N, M) = (≤,x,…,x,0,x,…x,1,x,…,x), ahol a maszkban lévő 0 a padlás($\log_2$(N))+1 vagy a 0. helyen van, ha N=0, vagy az 1. helyen van, ha 0 < N < 1, a maszkban lévő 1 pedig a padló($\log_2$(M))+1 vagy a 0. helyen van, ha 0 ≤ M < 1. Ha a 0 és az 1 azonos helyre esne, vagy az 1 megelőzi a 0-t, akkor nem létezik megoldás;
- LKTM(N, M) = (≤,x,…,x,0,x,…x,1,x,…,x), ahol a maszkban lévő 0 a padló($\log_2$(N))+1 vagy a 0. helyen van, ha 0 ≤ N < 1, a maszkban lévő 1 pedig a padlás($\log_2$(M))+1 vagy a 0. helyen van, ha M=0, vagy vagy az 1. helyen van, ha 0 < M < 1. Ha a 0 és az 1 azonos helyre esne, vagy az 1 megelőzi a 0-t, akkor nem létezik megoldás.

Ahol a padlás(x) függvény felfelé kerekít, a padló(x) pedig lefelé.

Algoritmus: Index Maszkos EHETEM-E (IMEE)

A fentieket figyelembe véve a következő algoritmus adható, amely már feltételezi, hogy a kettő hatványaira elhelyeztük a pozitív indexeket. Illetve az egészségügyi profilban lévő megszorítás mértékegységei megegyezik az indexelt érték mértékegységével minden összetevő esetén. Illetve minden megszorítás felső korlátja kisebb, mint a legnagyobb indexelt érték.

- Input: Egészségügyi profil, Élelmiszerek adatbázisa minden összetevő kettő hatványára indexelve.
- Output: Tiltott élelmiszerek listája.

1. lépés: Az egészségügyi profil tiltó megszorításait összetevőnként összevonom „N egység < összetevő tartalom ≤ M egység” alakú megszorítássá. Ez feltételezi, hogy minden megszorítás „X egység < összetevő tartalom”, „összetevő tartalom ≤ Y egység” vagy „X egység < összetevő tartalom ≤ Y egység” alakú. Ez a feltevés az eFilter projektben elfogadott.
2. lépés: Az ilyen fajta megszorítást pozitív maszkkal könnyen leírhatjuk. Ehhez először határozzuk meg az L és a H értéket. Mint látni fogjuk az L a 0, a H az 1 helyét határozza meg a „N egység < összetevő tartalom ≤ M egység” megszorításhoz tartozó maszkban.
   a. Ha N = 0, akkor L = 0, ha 0 < N < 1, akkor L = 1, egyébként L = padlás($\log_2$(N))+1, ahol padlás(x) felfelé kerekít. Ha N nem létezik, azaz az összevont megszorítás „összetevő tartalom ≤ M egység” alakú, akkor L se létezik.
   b. Ha M ≤ 1, akkor H = 0, egyébként H = padló($\log_2$(M))+1, ahol padló(x) lefelé kerekít. Ha M nem létezik, azaz az összevont megszorítás „ N egység < összetevő tartalom” alakú, akkor H se létezik.
3. lépés: Minden összetevőre kiszámolom az L és a H értéket az összevont feltétel alapján. Az így kapott L és H értékre szűkítem az adatbázist, úgy hogy csak azok az ételeket maradjanak, ahol az összetevő maszkja (≤,x,…,x,0,x,…x,1,x,…,x), ahol
   a. a 0 az L-dik indexen áll, illetve nincs 0 a maszkban, ha L nem létezik.
   b. az 1 a H-dik indexen áll, illetve nincs 1 a maszkban, ha H nem létezik.
   Megjegyzés: A maszkban 0-tól kezdődik az indexelés.
4. lépés: Vegyük észre, hogy ez az összevont feltétel legnagyobb tartalmazott maszkja (LNTM). Ha L >= H, akkor nem létezik a legnagyobb tartalmazott maszk. Ebben az esetben az algoritmus üres listát ad vissza.
5. lépés: A megmaradt ételek az összes összevont feltételnek megfelelnek, ezeket visszaadom.

A fenti algoritmust röviden IMEE algoritmusnak neveztük el. A legnagyobb gond vele, hogy nem mindig létezik a legnagyobb tartalmazott maszk. Szerencsére a gyakorlatban a tiltó megszorítások majdnem mindig „összetevő tartalom > X egység” alakúak. Ilyenkor mindig van legnagyobb tartalmazott maszk is, feltéve, hogy elég nagy a legnagyobb indexelt érték.

Az IMEE algoritmus elviekben más, mint a korábbiak, hiszen nem egy ételre kérdez rá, hogy az tiltott vagy sem, hanem étel listát ad vissza. A fenti verziójában a tiltott ételek listáját. Az IMEE algoritmussal nem csak a tiltott, hanem a tiltott, a nem javasolt, erősen javasolt, illetve a javasolt élelmiszerek listája is visszaadható. A nem tiltott élelmiszerek listája megkapható a tiltott élelmiszerek összes élelmiszerből történő kivonásával, vagy ha negáljuk a tiltó feltételeket és az alapján szűrünk.

Nézzünk egy példát az IMEE algoritmus futására. Tegyük fel, hogy a tiltásaink a következőek:

- zsír > 3dkg, zsír > 2dkg,
- fehérje > 1.5dkg, fehérje > 8 dkg,
- dió > 0g.

Ezek az összevonások után így néznek ki:

- zsír > 2dkg,
- fehérje > 1.5dkg,
- dió > 0g.

Az ezekhez tartozó legnagyobb tartalmazott maszkok:

- (≤,x,x,0,x) – zsír tartalom > 2 dkg,
- (≤,x,x,0,x) – fehérje tartalom > 2 dkg,
- (≤,0,x,x,x) – dió tartalom > 0 g.

Végül az algoritmus azokat az ételeket adja vissza, amelyek mind a három maszknak megfelelnek. Az első és a harmadik maszk pontosan írja le a megszorítást, a második egy erősebb megszorításnak felel meg, mint az eredeti. Így nem minden nem tiltott élelmiszert ad vissza, de amit visszaad, az biztosan nem tiltott.

Az IMEE algoritmus természetesen más indexeléssel is működőképes. Más indexelés esetén úgy kell számolni a H és az L értékét, hogy a 3. lépésben a legnagyobb tartalmazott maszkot kapjuk. Az indexek számításához annak a függvénynek az inverzét kell használni, amivel kijelöltük az indexelés érték határait. Mivel a példában erre a hatvány függvényt használtuk, ezért az indexek értékét a kettesalapú logaritmussal kellett számolni, ami a hatvány függvény inverze.

Az IMEE algoritmus úgy is átírható, hogy a legkisebb tartalmazó maszkot használjuk mindig. Ehhez a 2. lépést úgy kell átírni, hogy L számításánál lefelé, H számításánál felfelé kell kerekíteni. Ilyenkor az algoritmus visszaadhat tiltott ételeket is. Ilyenkor valamelyik korábbi EHETEM-E algoritmussal tovább szűrhetünk. Vagy az indexelést úgy kell beállítanom, hogy egy bizonyos hibahatáron belül adjon csak tiltott ételt vissza az algoritmus.

Az indexelés hibája

Az **indexelés hibája** az indexelt érték és a megszorításban lévő tényleges érték különbségének abszolút értékének és a tényleges érték hányadosa, amit százalékosan fejezünk ki. Példa:

Legyen a megszorítás a következő: fehérje tartalom > 1.5g. Az IMEE algoritmus ehhez az L = 2 értéket rendeli. Ez a 2^(L-1) = 2 index határnak felel meg. Így az indexelés hibája: ABS(2 – 1.5) / 1.5 = 33.33%.

Az **indexelés hibájának maximuma** akkor adódik, amikor a tényleges érték a legkisebb, amire még az IMEE algoritmus ugyanazt az index értéket adja. Ezek szerint a cikkben kifejtett indexelés maximális hibája 100%, mert minden L értékre

$$\lim_{N \to 2^{(L-2)}}((2^{(L-1)}-N)/N) = 100\%, \quad (3)$$

ahol 2^(L-1) a legkisebb tartalmazó maszk által leírt megszorítás alsó határa, a 2^(L-2) pedig az előző indexelhető érték.

Fokozatosan finomított lista

Az IMEE algoritmus legérdekesebb alkalmazása, amikor fokozatos finomításra használjuk az algoritmust. Itt az algoritmusnak azt a tulajdonságát használjuk ki, hogy nem a pontos listát, hanem egy attól szűkebbet ad vissza, ugyanakkor ezt a szűk listát nagyon gyorsan.

Az internet világában gyakori feladat, hogy egy nagy információs halmaz közelítését nagyon gyorsan le lehessen tölteni. Hasonló eredményt érhetünk el a következő algoritmus vázzal:

- Egy szűk halmazt gyorsan lekérdezni (LNTM);
- Ezt a szűk halmazt gyorsan eljuttatni a felhasználóhoz;
- Ha a felhasználó elidőzik a listán, akkor a lista finomítása:
  - LKTM – LNTM halmaz meghatározása;
  - Ezen halmazból leválogatni az eredeti feltételeknek megfelelő elemeket;
  - Leválogatás közben az új elemek folyamatos eljuttatása a felhasználóhoz.

Programozás technikailag ezt úgy érhetjük el, hogy a lista finomítását egy külön szál végzi a szerveren. Ezt a szálat csak akkor indítjuk, ha a felhasználó elidőz a listán. Ha a szál talál egy újabb elemet LKTM – LNTM halmazban, ami megfelel a megszorításoknak, akkor a szerver elküldi azt a felhasználó által használt klienshez.

Ez egyes elemek vizsgálatához felhasználhatom valamelyik EHETEM-E algoritmust.

TESZT EREDMÉNYEK

A tesztelés során az Oracle 11g adatbázis kezelő rendszert használtuk. Ezen belül tárolt eljárásokat, hiszen ezek gyorsabb adathozzáférést biztosítanak. Erre látunk példát ebben a cikkben (*Radványi*, 2004).

A teszt adatbázist http://www.ars.usda.gov/Services/docs.htm?docid=20959 linkről töltöttük le. A következő tárolt eljárással készítettük el az indexeket.

```
create or replace
FUNCTION GET_INDEX(ORG_VALUE IN NUMBER, COLNUM IN
NUMBER) RETURN NUMBER AS
vIND1 NUMBER; …; vIND17 NUMBER;
BEGIN
-- bemásoljuk az ABBREV_IND táblából az egyes tartományok értékeit
SELECT IND1, …, vIND17 FROM ABBREV_IND WHERE COL_NUM =
COLNUM;
-- if segítségével az index megállapítása
IF ORG_VALUE <= 0 THEN RETURN 262143; -- = 1111 1111 1111 1111 11
ELSIF ORG_VALUE <= vIND1 THEN RETURN 131071; -- = 0111 1111 1111
1111 11
…
```

```
ELSIF ORG_VALUE <= vIND17 THEN RETURN 1; -- = 0000 0000 0000 0000 01
ELSE RETURN 0;
END IF;
END GET_INDEX;
```

A GET_INDEX egy beérkező érték (ORG_VALUE) és oszlopszám (COLNUM) alapján egy indexet ad vissza. Egy segéd táblát használ, az ABBREV_IND táblát. Ez tartalmazza az index határokat. Itt az 1. index úgy lett meghatározva, hogy minden összetevőnél megkerestük a legkisebb értéket. A következő index mindig az előző kétszerese. Mivel a legnagyobb és a legkisebb érték aránya egyik összetevőnél sem volt nagyobb 2^17-nél, ezért 17 index értéket használtunk.

```
create or replace
PROCEDURE INDEX_LOOP AS
   var ABBREV%ROWTYPE; -- ABBREV típusú változó
   CURSOR curso IS SELECT * FROM ABBREV ORDER BY NDB_NO; --
kurzor
BEGIN
   FOR var IN curso LOOP -- végigmegy a tábla összes során
     var.WATER_IND:= GET_INDEX(var.Water,1); -- index lekérése
     UPDATE ABBREV SET WATER_IND = var.WATER_IND WHERE
NDB_NO = var.NDB_NO; -- az aktuálisan vizsgált sor első eleméhez tartozó index
frissítése
     var.Energ_Kcal_IND:= GET_INDEX(var.Energ_Kcal,2);
     UPDATE ABBREV SET Energ_Kcal_IND = var.Energ_Kcal_IND WHERE
NDB_NO = var.NDB_NO;
     …
   END LOOP;
END INDEX_LOOP;
```

Az INDEX_LOOP eljárás végigmegy az ABBREV tábla összes során, és frissíti a GET_INDEX függvény alapján az indexeiket. Itt csak az első két összetevő frissítését mutattuk meg. A többi összetevő frissítését töröltük, mert mindegyik nagyon hasonló, így könnyen megírhatóak a kihagyott sorok is.

Ezek után az egyes x_IND oszlopokra egy bitmap indexet helyeztünk el.

Ettől függetlenül a WATER és minden egyéb összetevő oszlopra is elhelyeztünk indexet. Nyilván ezekre felesleges lett volna bitmap indexet tenni, mert nagyon sok különböző érték van bennük.

A tesztelt SELECT utasítások közül közöljük a legbonyolultabbakat:

```
SELECT * FROM ABBREV WHERE WATER_IND >= POWER(2,9)-1 AND
ENERG_KCAL_IND >= POWER(2,9)-1 AND PROTEIN_IND >= POWER(2,9)-
1;
SELECT * FROM ABBREV WHERE WATER <= (SELECT IND9 FROM
ABBREV_IND WHERE COL_NUM = 1) AND ENERG_KCAL <= (SELECT
IND9 FROM ABBREV_IND WHERE COL_NUM = 2) AND PROTEIN <=
(SELECT IND9 FROM ABBREV_IND WHERE COL_NUM = 3);
```

Az első SELECT bitmap indexet használ, a másik sima megszorítást. A második SELECT utasításban lévő belső SELECT visszaadja azt az értéket, ami az első SELECT utasításban használt feltételnek megfelel. Így a két SELECT teljesen ugyanazt adja vissza.

A tesztek során a *2. ábrán* látható eredményre jutottunk.

2. ábra

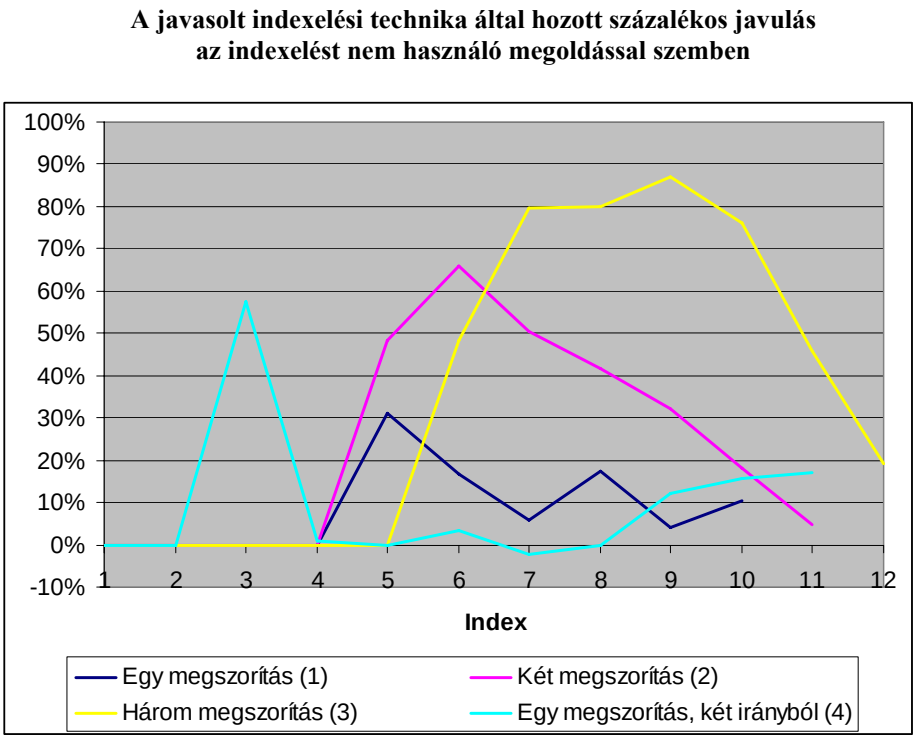


Figure 2: The improvements gained by the proposed indexing technique against the solution without indexing (given in percent)

The running time of the proposed solution in case of one constraint(1), One corresponds to the case with two constraints(2). One corresponds to the case with three constraints(3), One corresponds to the case with only one constraint, but this constraint also gives a lower and an upper bound(4)

Az X tengelyen értéke a POWER(2,X)-1 képletbe lett behelyettesítve az első SELECT utasításban.

A 0% a második SELECT futási ideje. Ehhez viszonyítottuk az első SELECT futási idejét relatíve. A pozitív érték jelentése, hogy gyorsabb az első SELECT, a negatív tartomány jelentése, hogy lassabb. Az egyes összehasonlítások abban különböznek, hogy a SELECT utasítás WHERE záradékában hány megszorítás van.

Látható, hogy X = 9 és 3 megszorítás használatának esetében a bitmap indexet használó SELECT közel 10-szer gyorsabb, mint az eredeti megszorítást használó „sima" SELECT. Nagyon érdekes, hogy ez a különbség a bitmap index javára csak sok százezer rekordot tartalmazó adatbázis esetén volt megfigyelhető. Néhány ezer rekordból álló adatbázisnál a bitmap indexelés nem gyorsabb (de legalább nem is lassabb), mint a nem bitmap indexet használó lekérdezés.

Hogy nagyobb X értékre miért jobb a bitmap index? Erre az lehet a magyarázat, hogy nagyobb X értékre több sort ad vissza a teszt során használt SELECT utasítás.

ÖSSZEFOGLALÁS

Ebben a cikkben nagyon alaposan átnéztük az EHETEM-E algoritmus legkülönbözőbb változatait. Ezek közül néhányat a Wit-Sys Zrt. munkatársai meg is valósítanak az eFilter pályázat keretében. Bizonyosan lehet még csiszolni ezeket az eljárásokat, miután a gyakorlatban megfigyeltük, milyen lekérdezések gyakoriak.

A bitmap indexelés ilyen fajta használata más élelmiszerekkel foglalkozó alkalmazásokban is hasznosíthatók, főleg, ha mennyiségi megszorításokat kell figyelni. Ilyen alkalmazás lehet az árulistába történő keresés, ahol egy adott összegnél olcsóbb árukat keresünk. Ezt például egy ár összehasonlító oldal keretében kitűnően lehetne hasznosítani.

A cikkben ismertetett fokozatos finomítás egyelőre nincs implementálva. Erre valószínűleg csak egy következő projekt keretében kerül sor.

IRODALOMJEGYZÉK

Chan, C.-Y., Ioannidis, Y.E. (1998): Bitmap index design and evaluation. Proceedings of the SIGMOD ’98 International Conference on Management of Data, 355-366. p.

Kusper G., Radványi T. (2007): Az EGERFOOD élelmiszerbiztonsági tudásközpont projekt információs rendszerének kialakítása. Networkshop 2007 konferencia, Eger, 8 p.

Kusper G., Radványi T. (2008): Az EGERFOOD élelmiszerbiztonsági nyomkövető rendszer – Hogyan modellezzük a cégek munkafolyamatait. Networkshop 2008 konferencia, Dunaújváros, 8 p.

Liptai K., Kusper G., Radványi T. (2007): Cryptographycal protocols in the Egerfood Information System. Annales Mathematicae et Informaticae 34, ISSN 1787-5021, 61-70. p.

Radványi T. (2004): Examination of the MSSQL server from the user's point view considering data insertion. Acta Academiae Pedagogicae Agriensis, 69-77. p.

Radványi T., Kusper G. (2007): Requirement analysis and a database model for the project EGERFOOD Food Safety Knowledge Center. Invited talk. Proceedings of ICAI-2007, 1. 15-23. p.

Radványi T., Kusper G., Kovács E. (2008a): Adatforgalom és mobilkommunikáció az Egerfood rendszerben. AgriaMédia 2008 konferencia, Kötet I. 100-107. p.

Radványi T., Kusper G., Kovács E. (2008b): Kommunikáció az EGERFOOD élelmiszerbiztonsági projekt információs rendszerében. IF2008 konferencia, CD-kiadvány, ISBN 978-963-473-129-0, 10 p.

Spiegler, I., Maayan, R. (1985): Storage and retrieval considerations of binary data bases. Information Processing and Management: an International Journal 21. 3. 233-254. p.

Wu, K., Otoo, E. J., Shoshani, A. (2006): Optimizing bitmap indices with efficient compression. ACM Transactions on Database Systems, 31. 1. 1-38. p.

Levelezési cím (*Corresponding author*):

Kusper Gábor
Eszterházy Károly Főiskola, Számítástudományi Tanszék
3300 Eger, Eszterházy tér 1.
Eszterházy Károly College, Department of Computer Science
H-3300 Eger, Eszterházy tér 1.
Tel.: 36-36-520-486
gkusper@aries.ektf.hu