Gyakorlati óravázlat 11 – Indexek, végrehajtási terv

1. Beágyazott lekérdezések
2. Nézetek
3. Módosítás nézeten keresztül

\_\_\_\_\_\_\_\_\_\_\_\_

**Koncepció**

Ezen a héten azt a táncot fogjuk járni, hogy mindenféle műveleteket (főleg lekérdezések, de lesz DML is) fogunk végrehajtani táblákon először index nélkül, aztán különböző indexek meglétével. A SET AUTOTRACE ON sql plus parancs segítségével minden műveletnek meg fogjuk nézni a végrehajtási tervét (execution plan). Így fogjuk – remélhetőleg – érthetővé tenni az indexes előadáson elhangzottakat. Ehhez azonban az kell, hogy egy izmosabb adatbázison kísérletezzünk, úgyhogy egy felturbózott kereskedős adatbázist hoztam létre a MINTA sémában, amit le kellene másolni az óra elején. Ahhoz pedig, hogy lássunk adatokat az indexekről a statisztikát is meg kell csinltatnuk a rendszerrel. A kézzel létrehozott statisztikákra azonnal készül elemézés, de a tábladefinícióval létrejövő indexek létrejöttükkor ugye üresek és a statisztikák nem készülnek el minden dml művelet után automatikusan (mert akkor több időt töltene a rendszer stat készítéssel, mint a felhasználók kiszolgálásával).

**Használandó adatábzis létrehozása**

Nem másolással hozzuk létre a táblákat, hogy a kényszerek is meglegyenek (nem mindegyiknek lesz jelentősége, de azért mégis). Az órai fájlban benne vannak a létrehozó állományok, és az adatok átmásolására szolgáló utasítások is.

**Indexek vizsgálata**

Ezután kérdezzük le az indexek adatait. Az egyik előadáson volt szó róla, hogy létezik egy katalógustábla a katalógustáblákról. Ebben meg tudjuk nézni, hogy milyen indexes katalógusaink vannak. Ebből érdekes lehet a USER\_INDEXES, a USER\_IND\_COLUMNS és a USER\_IND\_STATISTICS

Ha most megnézzük, akkor a pk és unique kényszerekkel automatikusan létrejött indexeket fogjuk látni. Az index neve megegyezik a kényszer nevével (ha csak nem adunk meg mást). Most szándékosan nevesített kényszereket készítettem, hogy könnyebb legyen követni. Index kézi létrehozásakor mindenféleképpen kell nevet adni. Én azt a névkonvenciót szoktam használni, hogy a kényszerek nevét akényszerre utaló előtaggal kezdem (PK\_, U\_, FK\_, CON\_, NN\_), az indexekét pedig IND\_ előtaggal. Mivel mindkettő konkrét táblához kapcsolódik, ezért a tábla nevét is bele szoktam venni az elnevezésbe és aztán a konkrét kényszerre, vagy az index oszlopaira vonatkozó névrész következik. Ettől persze el lehet térni, de nekem így logikus és követhető.

Rögtön létrehozáskor nem sok érdekességet látunk az indexekről, amit már magunktól is ne tudnánk. Az utolsó statisztika dátuma null érték. Annyi, hogy van egy CLOB oszlop termék táblában, arra is létrejött egy index, aminek a típusa LOB. Ez nem tananyag. Akit érdekel, olvasson utána.

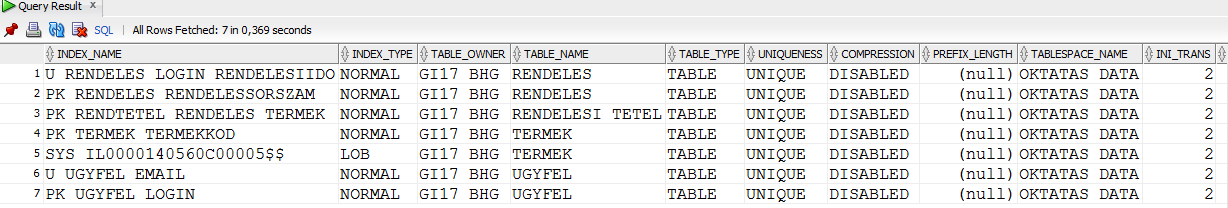
Egy kicsit át lehet ismételni az indextípusokról előadáson hallottakat. Egyik csoportosítás, hogy egyedi-e. Egyelőre csak ilyeneket látunk. Lehet egyedi indexet is kézileg is létrehozni, de nem javasolt, erre a célra célszerűbb a kényszerek használata.

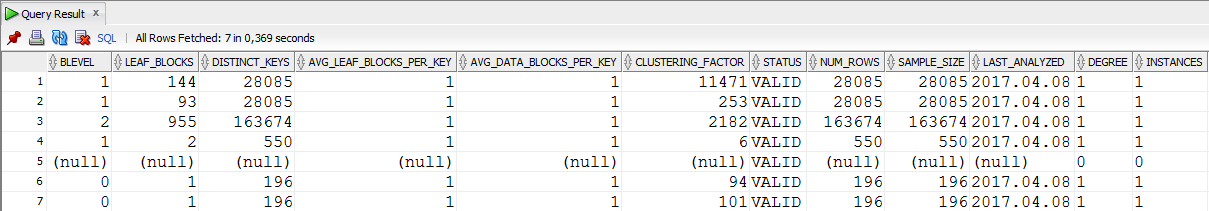
Ha nem egyedi, akkor elméleti síkon még mindig megkülönböztethetjük a szelektív és nem szelektív indexeket, ami annyit jelent, hogy előbbi esetben kevés az ismétlődés, az utóbbiban meg sok. Az OLTP adatbázisokban jellemzően B fa típusú indexeket használunk, ezeket NORMAL típusúként tűnteti fel a katalógus. A nem szelektív oszlopokra lehet létjogosultsága ehelyett a BITMAP indexnek, de OLTP rendszerekben nem hatékony. Azért fogunk ilyet is nézni, de majd az utolsó héten, amikor kicsit fogunk tranzakciókkal játszani, akkor fogjuk azt is demonstrálni, hogy milyen gondokat tud okozni egy bitmap index.

A másik csoportosítás aszerint lehet, hogy egyoszlopos (egyszerű), vagy többoszlopos (másnéven összetett vagy konkatenált) az index. A renedeles és a rendelesi\_tetel táblákban látunk egy-egy kétoszlopos kulcsot, amire értelemszerűen kétoszlopos index jött létre. Általában a gyakran együtt lekérdezett oszlopokra érdemes összetett indexet létrehozni. A sorrend nem mindegy. Általában a szelektívebbet érdemes előre venni, illetve azt, amelyikre önmagában is gyakran fogalmazunk meg feltételt. Azt, hogy az index milyen oszlopokra épül a USER\_IND\_COLUMNS katalógustáblából kérdezhetjük le (ezt spórolhatjuk meg, ha eleve az index nevébe beleírjuk az oszlopneveket is). Természetesen más infók is kiderülnek abból a táblából.

Ezután jön, hogy készítsen statisztikát, majd újra megnézzük az indexeink adatait. Ha a táblákra készítünk statisztikát, akkor automatikusan a tábla indexeire is. De kérhetünk csak index statisztikát is külön-külön. Ehhez kivételesen most menüt fogunk használni, mert sajnos máshogy nem működött, csak túl bonyolultan. (Ez szemléltetéshez kell, vizsgán nem kell statisztikát készíteniük, úgyhogy nem lesz gond, hogy nem szabad a menüt használni.) Tehát Connections ablak, táblák. Táblákon egyesével jobb egérgomb. Statistics/Gather statistics.

Most futassuk le újra a USER\_INDEXES lekérdezését. Idemásolom az eredmény jópár oszlopát. Amit ebből meg lehet nézni, tudják előadás alapján értelmezni, azt bekarikáztam.





Látni fogjuk pl. hogy hány különböző érték szerepel az indextáblákban (DISTINCT KEYS). Látjuk azt is, hogy a B-fa hány szintes (BLEVEL), illetve hogy hány szekvenciális indexlap van (levelek – leaves)(LEAF\_BLOCKS) Az ügyfél tábla login és e-mail indextáblája is elfér egy-egy blokkban, így ott még egyáltalán nem épül egyetlen b-fa szint sem. A termék tábla pk-ja már 2 blokkban fér csak el, így ott már megjelenik a B-fa első szintje, de a rendelés tábla pk 144 blokkjához is még elég egyetlen B-fa szint. A rendelési tétel tábla elsődleges kulcsának pl. a B-fája már 2 szintű. Alternatív kulcsoknál a különböző értékek száma megegyezik, az index mérete mégis lehet különböző az indexértékek méretkülönbsége miatt (pl. a rendelés táblában a pk egyoszlopos, 28085 különböző érték és a levelek 93 blokkban férnek el, míg az alternatív kulcsunk kétoszlopos: login és rendelési idő együtt, nyilván ez is 28085 különböző értéket tartalmaz, de 144 blokkban fér csak el.). A két átlagoszlopot csak azért karikáztam be, hogyha rákérdeznek, arra is tudjatok válaszolni, nem kell túlságosan belemenni. Az első (AVG\_LEAF\_BLOCKS\_PER\_KEY) a kulcsértékenként érintett átlagos indexblokk szám. Természetesen egyedi indexeknél ez értelemszerűen 1, ahogy a második is (AVG\_DATA\_BLOCKS\_PER\_KEY), azaz az adatrekordok átlagos száma kulcsértékenként is. Esetleg ezt a statisztikát majd vissza lehet nézni azután, hogy csináltunk nem egyedi indexet is, mert ott már érdekesek lehetnek ezek a számok. Azt mutatják, hogy mennyire szelktív az index. Az utolsó két bekarikázott oszlop egyértelmű, az indextábla sorainak száma, illetve a statisztika készítésének a dátuma.

Egy mezei felhasználó nem szokta ezeket a statisztikákat nézegetni, inkább csak azért nézzük meg ezeket, hogy kézzelfoghatóbb legyen az, amit az előadáson hallottak indexekről. A statisztikák a lekérdezés-optimalizáló számára fontosak. Amikor mi kiadunk egy select utasítást, akkor az Oracle a háttérben ezeknek a statisztikáknak az alapján dönti el, hogy milyen módon adja meg a választ. Melyik where feltételt értékelje ki először, join esetén melyik táblán menjen végig először, stb.

**Feladatok**

1. Dobjuk el az összes objektumunkat (fontos, mert kell a tárhely a mai tábláknak)
2. Hozzuk létre a kereskedelmi adatbázis 4 tábláját és másoljuk át bele az adatokat a MINTA sémából. Láthatjuk, hogy kicsit megnőtt az adatmennyiség.
3. Kérdezzük le az indexeink adatait a katlógustáblákból!
4. Készíttessünk a rendszerrel statisztikát!

**Indexek teljesítményének vizsgálata – végrehajtási terv**

Kapcsoljuk be a nyomkövetést (nem tudom, hogy hívják magyarul): sql plus parancs kell hozzá: set autotrace on

Az utasításainkat RUN SCRIPT-tel futtassuk. Most nem maga az eredmény lesz érdekes, hanem azok az infók, amiket utánaír a script outputban. Egyrészt a műveletek listája: innen látjuk, hogy milyen részműveletekre oszloptt a művelet (statement), és azok milyen sorrendben történtek. Ezen kívül a Cost oszlop szolgál információval a költségről, aminek nincs kézzelfogható mértékegysége, de összehasonlításra szolgál. Egyébként a cpu és az I/O költségből kalkulálja valamilyen módon. Látni fogjuk, hogy nem minden esetben használja fel az indexeinket, ilyenkor teljes táblaolvasás van. Nyilván a tábla méretétől is függ az, hogy milyen különbséget jelent az index scan a tábla scan-hez képest. Fogunk látni példát arra, hogy az összetett indexet csak az abban szereplő első oszlopra vonatkozó szűrésnél tudjuk felhasználni, a második oszlop szerint nem. Fogunk készíteni függvény alapú indexet is.

A terv alatti statisztikákat nem fogjuk nézni.

Az plan olvasási módja: a felső sorban látszik a művelet teljes költsége, illetve a ROWS oszlopban, hogy a statisztikák alapján mit gondol az Oracle, hogy hány eredményrekordunk lesz. A beljebb kezdett sorok az egyes műveletek részműveletei. Az olvasást tehát a részműveletektől kell kezdeni. A költségek pedig úgy alakulnak, hogy a részműveletek költségeinek összege adja ki a főbb műveletek költségét. Természetesen kerekítési hiba miatt ez nem mindig pontos.

**Feladatok**

1. Kapcsoljuk be a nyomkövetést és kérdezzük le az ugyfel táblából az azonosítót, a vezeték és keresztneveket! Ismételjük meg a lekérdezést különböző szűrőfeltételekkel, és vizsgáljuk meg az Execution plan információit!
   1. csak a Krisztina keresztnevűeket
   2. csak a krisztina login-hoz tartozót
   3. csak azokat, amelyeknek a loginja krisztina karaktersorral kezdődnek
   4. csak azokat, melyek loginjában szerepel a krisz karaktersor

Mivel az ügyfél tábla nagyon kicsi, nem lesz számottevő különbség a cost-ban, de annyit azért látunk, hogy melyikeknél használt indexet. (Meg lehet velük tippeltetni előre) A b) és c) megoldásához használ indexet. Az a) pontban a keresztnévre nincs index. A d pontban pedig a kirsztina nem feltétlenül a mező elején van, tehát nem megy sokra az abc sorrendű indexszel.

1. Kérdezzük le a rendeles táblából a rendelés sorszámát, idejét (másodperce pontosan) és az ügyfél azonosítóját (login)! Ismételjük meg a lekérdezést különböző szűrőfeltételekkel, és vizsgáljuk meg az Execution plan információit!
   1. csak a 100-as sorszámút
   2. csak a 100-nál kisebb sorszámúakat
   3. csak az 5000-nél kisebb sorszámúakat
   4. csak a 6000-nél kisebb sorszámúakat
   5. csak a megadott másodpercben leadott rendelést
   6. csak az INSTVANV loginú vásárló rendeléseit
   7. Ha gyakran kérdezünk le renedelési idő alapján, login nélkül, akkor érdemes lehet a rendelési időre külön is indexet készíteni. Vigyázat, ez már nem lesz unique! Hozzuk létre explicit módon és kérdezzünk le újra egy pontos időpontot és egy nap rendeléseit is.
   8. Dobjuk el az előbb létrehozott indexet és hozzunk helyette létre explicit módon egy függvény alapú indexet a rendelési időre az időadatok nélkül. Természetesen ez sem unique, majd ismételjük meg az utolsó két lekérdezést!

A d) pont lesz az első, amihez már nem fogja az indexet használni, mert nem elég szelektív a kérdés.

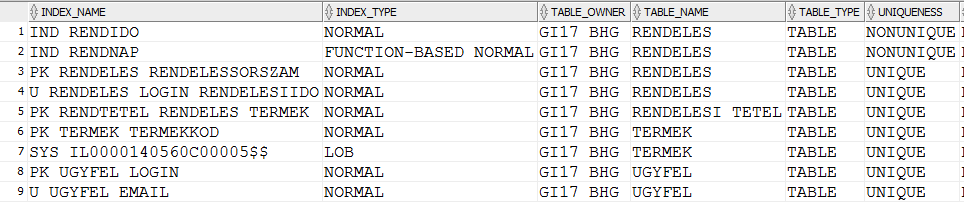
Az e) pontban azért nem tud indexet használni, mert a rendelési időre nincs indexünk, csak a loginnal párban, de az nem rendelési időre, hanem loginra rendezett, hiszen az van az összefűzésben elöl.

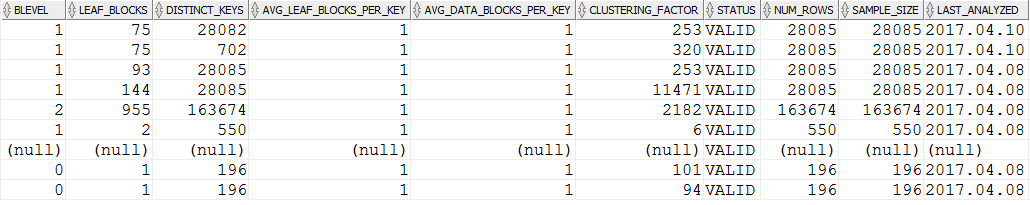
Az f) pontban látjuk, hogy valóban, ha olyat kérdezünk le, ami az összetett indexben elöl van, akkor azt fel tudja használni a kereséshez.

A g) pontban az előző pontok tanulságaira alapozva hozunk létre indexet külön csak a rendelési időre. Ha pontos időre kérdezünk, akkor nyilván segíteni fog, de ha napra, akkor már nem látja elég segítségnek az indexet.

Cseréljük le az előző indexet egy függvény alapú indexre, ahol csak a napokig vesszük az indexértékeket és ismételjük meg az előző két lekérdezést. Látjuk, hogy ez az index mindkét lekérdezéshez felhasználásra került.

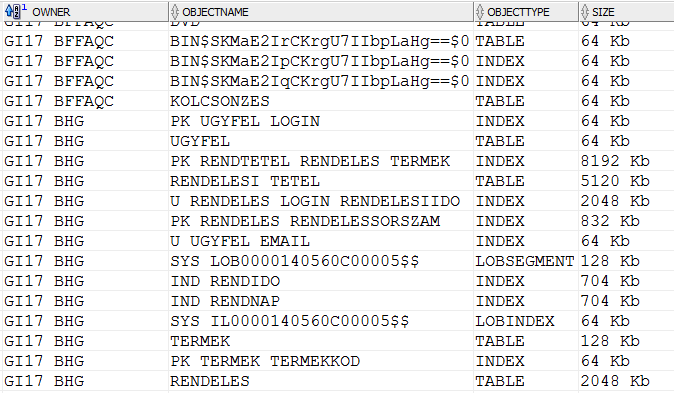
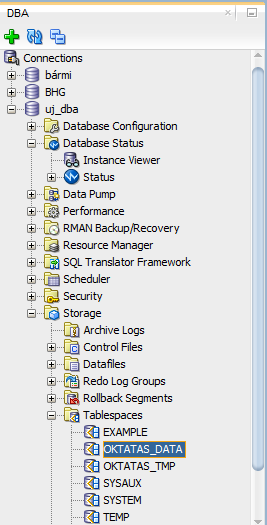
1. Hozzuk újra létre az eldobott indexet is és kérdezzük le újra az indexeink tulajdonságait!





Mivel összesen 20M limit van mindenkinek beállítva, takarékosan kell bánnunk az indexeinkkel (fel is emelem 30M-re). Akinek belefér az órába és van kedve ezzel foglalkozni, az nézzen rá a fizikai méretekre. Most kivételesen kapcsoljuk be újra az első órán látott DBA ablakott és ott nézzük be az OKTATAS\_DATA tablespace adatait. Az indexek mérete nagyon nagy. Azért meg lehet említeni, hogy a táblák sok esetben jóval több oszlopot tartalmaznak, és akkor a táblaméret-indexméret arány jobban néz ki, de azért jól látszik, hogy az indexek bizony helyigényesek.

Illetve akit a fizikai résszel kapcsolatos előadás megfogott, azoknak meg lehet mutatni ugyanitt az Instance Viewer-t.



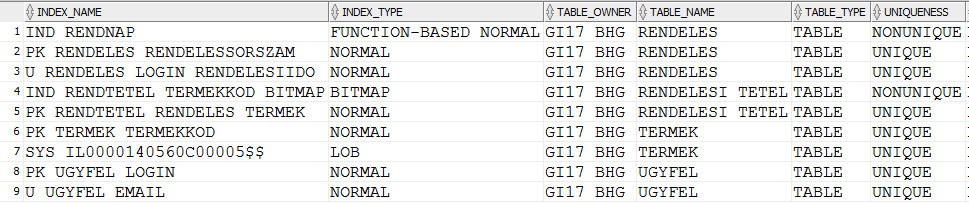
1. Kérdezzük le a termékeink azonosítóját, nevét és a belőlük leadott rendeléseket. Ismételjük meg a lekérdezést különböző szűrésekkel!
   1. csak a kabátok rendelése
   2. b. csak az F001-KB101 termékkódú termék rendelése
   3. készítsünk indexet a rendelési\_tétel táblában a termékkódra
   4. cseréljük le az előző indexet bitmap indexre! Nézzük is meg az összes indexünket!

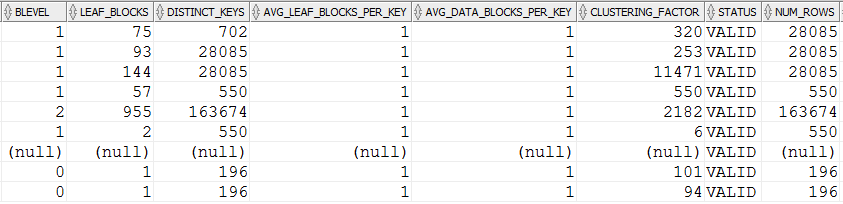
A HASH joint nem kell magyarázni, mondjunk csak annyit, hogy a join megvalósítására különböző algoritmusok léteznek, ezek egyike a hash. Akit érdekel, nyugodtan utánanézhet a többinek, illetve kikísérletezheti, hogy mikor melyik algoritmust választja az Oracle. Akár tdk dolgozatot is kihozhat ilyenek elemzéséből jövőre.

A b) megoldásnál természetesen igénybe veszi az pk-ra épülő indexet

A c)-nél felmerülhet, hogy érdemes indexet tenni az idegenkulcsra is (termékkód a rendelési tétel táblában), de látni fogjuk, hogy nem elég szelektív (570 termékhez kb. 170ezer rendelési tétel tartozik)

A d)-nél el kell dobnunk az indexet, hogy bitmap-ként létrehozzuk újra. Kb. harmadára fog csökkenni a lekérdezés költsége. Úgy tűnhet, hogy nagyon hatékony. Majd az utolsó órán tranzakciózunk kicsit és akkor fogunk rádöbbenni a bitmap index kárára (miszerint ha törlünk, módosítunk egy tranzakcióban, az túl nagy részét zárolja az adatbázisnak, mert a bitmap index a b-fával szemben nem soronként zárolódik, és így komoly fennakadást okozunk a többi tranzakcióban)





A második keretből az látszik, hogy bár ugyanarra a táblára készült, a bitmap indexnek nem 160ezer sora van, csak 550 és bár az indexértékek is kisebbek (hiszen a pk-ban is a termékkód van, csak nem első mezőként, hanem másodikként), a levélblokkok száma is nagyságrenddel kevesebb.

Óra végén dobassunk el a hallgatókkal mindent, hogy az össz tárhelyünk se fogyjon el! Illetve, ha gubanc lesz ezzel, akkor én központilag fogom törölni az összes hallgatói séma (GI17\_ kezdetű sémák) minden objektumát.

**Önálló feladatok:**

1. Kérdezzük le a 4 tábla joinjával, hogy az ügyfeleink összesen hány alakalommal rendeltek különböző termékkódú kabátokat és összesen mekkora mennyiségben!
2. Készítsünk olyan indexeket amelyek lecsökkentik a fenti lekérdezés költségét! Ha nincs elég tárhelyünk, akkor dobjunk el olyan indexeket, amelyek ehhez a lekérdezéshez nem kellenek!
3. Ismételjük az előző gyakorlatok gyakorló feladatait!