

1. Személyes adatok

Név: Nyárády Péter
Neptun kód: QJA31E
E-mail: pnyarady@gmail.com
Blog oldal: <http://oraoptimization.blogspot.com/>

Konzulensek: Kardkovács Zsolt – BME
kardkovacs@tmit.bme.hu

Marton József – BME
marton@db.bme.hu

Sárecz Lajos – Oracle Hungary kft.
lajos.sarecz@oracle.com

Feladat címe: Oracle Adatbázisok Optimalizálása
Feladat leírása: Irodalomfeldolgozás

Tanulmány:

http://members.chello.hu/m.nyarady/blog/beszamolok/7felev/tanulmany_qja31e.pdf

Tartalomjegyzék

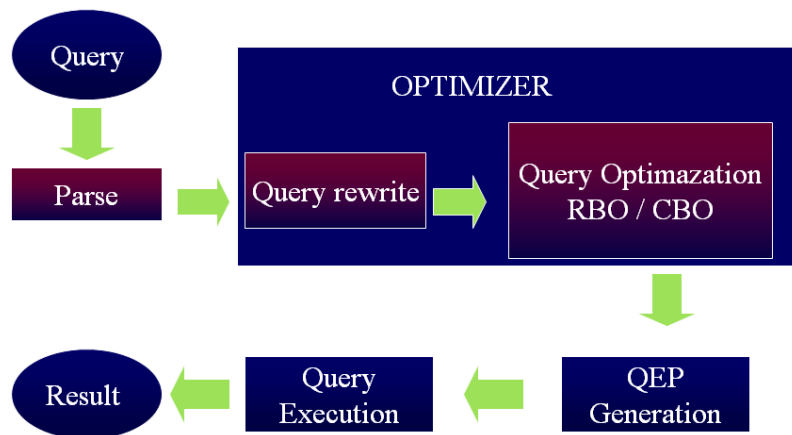
1. Személyes adatok	1
2. Bevezetés	2
2.1. Elméleti bevezető	2
2.2. A munka előzményei és kiindulási állapota	3
3. Az elvégzett munka	3
3.1. Struktúra	3
3.2. Sémaobjektumok	4
3.3. Adatszótár	4
3.4. Memória	5
3.5. Processzek	6
4. A végállapot ismertetése	6

2. Bevezetés

2.1. Elméleti bevezető

Az optimalizálás témaköre egy biztos, alapos háttértudást igénylő szakterület, melyhez számos területen szerzett elméleti alapok szükségesek. Óriási mennyiségű szakirodalom található az Interneten, illetve az Oracle oldaláról¹ is rengeteg fórum és blog oldal érhető el, melyek a téma különböző területeit részletezik.

Az adatbázis teljesítményét növelő technikák megismerése előtt feltétlen fontos megérteni, hogy egy lekérdezés végrehajtása során mi játszódik le a háttérben. Ehhez az 1. ábra nyújt segítséget.



1. ábra. Lekérdezés végrehajtásának lépései [1]. A nyilak a transzformált lekérdezés áramlását jelölik az egyes lépések között.

Az utasítás először egy szintaktikai elemzésen (parse) esik át, majd egy végrehajtási tervet (QEP – query execution plan) készítünk a lekérdezéshez. Végül végrehajtjuk (execution) magát a lekérdezést, s visszaadjuk az eredményét. Az Oracle rendelkezik egy beépített optimalizérral, amely a végrehajtási terv kiszámításánál játszik fontos szerepet. Az optimalizier kezdeti verzióiban a végrehajtási terv számítását szabály-alapú optimalizáció (Rule Based Optimization, RBO [2]) segítette (Oracle 6-os verziójától). Ez gyakorlatilag egy statikus szabályrendszer, ami alapján az optimizer eldönti, hogy a táblákat milyen sorrendben illessze össze, milyen indexeket használjon, hogyan kérdezze le az adatokat, stb.. A szabályrendszer statikus mivolta miatt azonban nem működhet dinamikusán jól egy rendszer. Ennek folyamatos orvoslása miatt a szabályok száma az évek során hatalmasra nőtt, ezért az RBO-t fokozatosan felváltotta a költség-alapú optimalizáció (Cost Based Optimization, CBO) – az Oracle 10g-tól az RBO mód használata már nem is engedélyezett. A CBO már egy jóval dinamikusabb módszer. Az összes rendelkezésre álló adat (könyvtárak, statisztikák, hisztogramok, paraméter beállítások) alapján próbálja a leoptimalisabb végrehajtási tervet kiszámítani.

Gyakran előfordulhat azonban olyan eset, hogy az optimizer sem a legjobb végrehajtási tervet készíti el. Ekkor kerülnek előtérbe az ún. SQL hintek, amelyekkel rákényszeríthetjük az adatbázist arra, hogy az általunk javasolt optimalizálási eljárást, tábla hozzáférést és/vagy illesztési sorrendet és metódust használja.

¹<http://otn.oracle.com/>

Az SQL hintek által biztosított beavatkozásán kívül azonban számos egyéb lehetőség van arra, hogy növeljük az adatbázisunk teljesítményét. Ilyenek lehet például az adatbázis sémájának átkonfigurálása, indexek és materializált nézetek létrehozása a frekvenciánál magasabb adatokra, valamint az Oracle összetett memóriarendszerének a finomhangolása.

2.2. A munka előzményei és kiindulási állapota

A félév elején még gyakorlatilag semmilyen, a témához közel álló tudással nem rendelkezem, ebből kifolyólag nagy mennyiségű irodalomfeldolgozás várt rám. Az adatbázis hatékony hangolásához elengedhetetlen, hogy az ember mélységében ismerje az adatbázis felépítését és működését. A féléves irodalomfeldolgozásom ezért főként az Oracle Library - Concepts fejezetére [3] „korlátozódott”. Ezen felül még több, adatbázis programozással és SQL tuningolással foglalkozó, igencsak terjedelmes ebookokat dolgoztam fel.

A kiindulási állapothoz hozzátartozik továbbá, hogy a (2007 őszi) félév kezdetén jött ki az Oracle Database legújabb verziója, a 11g. Kezdetben ehhez csak Linux-os változat létezett, így – Windows-os felhasználó lévén – komoly időt és munkát kellett fordítanom a rendszer és az adatbázis telepítésére is. Ez utóbbi egyébként koránt sem olyan egyszerű feladat, mint amire az ember számítana. Számos apró kis buktatója lehet a telepítésnek, ami hosszas keresést és utána olvasást igényelhet. Ennek megkönnyítése érdekében elérhető a neten néhány színvonalas video is², amelyek Oracle adatbázis telepítést mutatnak be lépésről lépésre, s megválaszolják a felmerülő kérdéseket, döntéshelyzeteket.

3. Az elvégzett munka

A 7. szemeszteres önálló laboratóriumban előírt feladat az irodalomfeldolgozás, így az én félévemet is főként ez képezte. Fontos volt alaposan megismernem az Oracle adatbázisok felépítését és működését, amit az Oracle Library - Concepts fejezete [3] alapján végeztem. Az elvégzett munkáról egy 50 oldalas tanulmány [4] készült, melynek elérhetősége a borítón is fel van tüntetve – itt csak egy rövid áttekintést adnék az érintett területekről.

3.1. Struktúra

Az Oracle Database logikai felépítése alapvetően három szintből áll. A legkisebb egységet az **adatblokkok** képezik, amik egy előre meghatározott, fix pár bytesnyi részt jelentenek. A fizikailag folytonosan elhelyezkedő, valamilyen tárolási célból előre lefoglalt adatblokkok képezik a következő szintet, az **extenteket**. Ha egy extent betelik, de szükség van további szabad helyre, akkor új extentet kell foglalnunk. Az így keletkező, azonos célra foglalt, s azonos táblateren belül elhelyezkedő extentek alkotják a harmadik hierarchia szintet, a **szegmenseket**. A tanulmány részletesen kifejti, hogy hogyan lehet az adatblokkok írásának engedélyezését szabályozni (pl. PCTFREE, PCTUSED paraméterekkel), valamint hogy hogyan működik az extentek és szegmensek területfoglalásai és -felszabadításai.

A logikai adattárolás legnagyobb egységei a táblateretek, míg a fizikai tárolás adatfájlok formájában történik. A táblateretek menedzselésére két mód kínálkozik: a *lokálisan vezérelt* táblateretek egy bitmap alapján működnek, míg a *könyvtár vezérelt* táblateretek az adatszótárban tartják nyilván a lefoglalt területeket. Egy adatfájl egyszerre csak egy táblateret tartozhat,

²<http://www.youtube.com/watch?v=CHzV4LZnvHc>

azonban egy táblatér akár több adatfájlban is tárolhatja az adatait. Illetve léteznek ún. ideiglenes táblaterek és adatfájlok is, melyek a számítások gyorsítását hivatottak elősegíteni.

3.2. Sémaobjektumok

Sémának nevezzük az egy felhasználóhoz tartozó logikai adatstruktúrák (séma objektumok) összességét. Ezek az objektumok nem feleltethetők meg egy az egyben fizikai diszken tárolt fájloknak. Logikailag egy objektum egy tablespacen belül helyezkedik el, fizikailag azonban tárolódhat akár több datafileban is. A sémák és tablespacek között nincs semmilyen összefüggés: egy tablespace tartalmazhat objektumokat több különböző sémából, illetve egy séma objektumai is tárolódhatnak különböző tablespacekben. Számos sémaobjektum létezik, melyek közül optimalizálási szempontból az indexek és a materializált nézetek a legfontosabbak, így itt csak ezekről írnék röviden.

Az indexek opcionális, táblákhoz és klaszterekhez rendelhető struktúrák. Indexeket lehet egy vagy több oszlophoz rendelni, melyek segítségével egy SQL utasítás esetén gyorsabban megtalálja a keresett információt az adatbázis - ezáltal a helyesen használt indexek jelentik az I/O műveletek csökkentésének fő forrását. A következő index-sémák használhatóak: B-fa index, B-fa klaszter index, Hash klaszter index, Reverse key (inverz kulcs) index, Bitmap index, Bitmap join index. Az indexek használata automatikus, csak a létrehozásukkal és az esetleges törlésükkel kell foglalkozni. Továbbá az optimalizáló (optimizer) akár felhasználhat egy már létező indexet egy új index létrehozására, ami jóval gyorsabb index-építést eredményezhet. Az Oracle adatbázis B*-fákat használ az indexek tárolására. Ezáltal a szekvenciális keresés átlagos $n/2$ idejét lecsökkenti $O(\log(n))$ -re.

A materializált nézeteket adatok összegzésére, számítására, replikálására és szétosztására használhatjuk. Ebből kifolyólag főként adattárházaknál, döntéstámogató rendszereknél és elosztott vagy mobil számításoknál használjuk őket. Az optimizer automatikusan felismeri, hogy mikor lehet egy kérést materializált nézet segítségével kielégíteni, s automatikusan behelyettesíti azt a lekérdezésbe. Így nem szükséges közvetlen a táblákból vagy nézetekből kinyerni a kívánt adatokat, amivel növelhetjük a teljesítményt.

3.3. Adatszótár

Az adatszótár (**data dictionary**) az Oracle adatbázis egyik legfontosabb részét képezi. A központi, csak olvasható referencia táblák és nézetek tartoznak hozzá, melyekben az adatbázisról tároljuk a következő lényeges információkat:

- séma objektumok definíciója
- séma objektumok számára allokált és felhasznált területek
- oszlopok alapértelmezett értékei
- integritás kényszerekről információk
- az adatbázis felhasználóinak nevei
- az egyes felhasználókhoz tartozó jogok és szerepek
- naplózási információk

- egyéb általános adatbázis információk

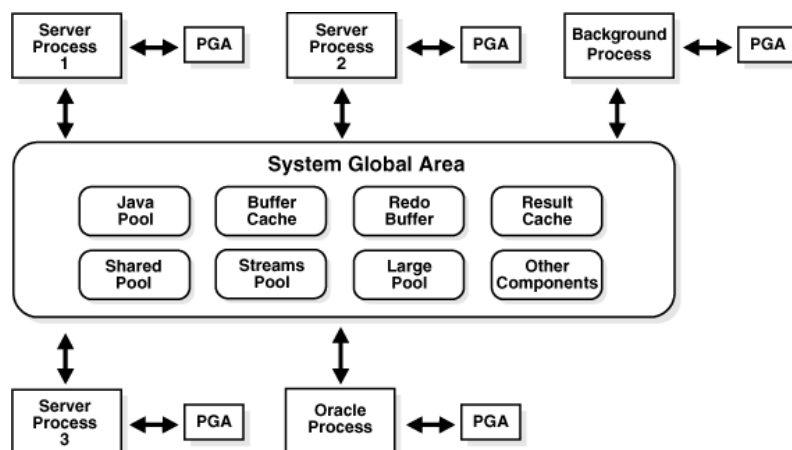
A SYSTEM tabletérben tároljuk, ami mindig online, ezért ezek az információk mindig elérhetőek. Szerkezetileg kétféle objektumot tartalmaz:

1. *Alap táblák (Base tables)*: az adatbázisról tartalmazznak információkat. Csak az Oracle adatbázis írhatja és olvashatja őket. Az adatokat titkosított formában tárolják.
2. *Felhasználói nézetek (User-Accessible Views)*: nézetek, melyek összegzik és a felhasználók számára emészthető formában megjelenítik az Alap táblákban tárolt információkat. A legtöbb felhasználó ezekhez a nézetekhez férhet hozzá.

3.4. Memória

Az Oracle adatbázis memóriájában a következő komponenseket tároljuk:

- programkód
- információ a csatlakozott active és inactive sessionökről
- programvégrehajtás közben szükséges információk, állapotok
- az adatbázis processzei között megosztott információk (pl. zárak)
- cachelt adatok, mint pl. adatblokkok és redo log bejegyzések, amik ugyanakkor természetesen tárolva vannak a merevlemezen is



2. ábra. Az Oracle Adatbázis memóriájának struktúrája [5].

A memória struktúrája (2. ábra) alapvetően három része osztható:

- *Szoftver kód területek*: az éppen futó vagy futtatható kódokat tartalmazza. Általában a felhasználói programoktól eltérő helyen van.

- *System Global Area (SGA)*: az összes szerver- és háttér folyamat (processz) között megosztott memória struktúrák (SGA komponensek – lásd fenti ábra) csoportja. Adat és vezérlési információkat tartalmaz.
- *Program Global Area (PGA)*: egy bizonyos szerver processzhez tartozó adat és vezérlési információkat tartalmazó memóriaterület, mely a szerverprocessz indításakor jön létre, s csak ő fér hozzá. Minden egyes szerverprocesszhez (és háttér folyamatához) tartozik egy-egy PGA. Az adatbázis inicializálási paramétereinél megadott PGA méret az összes PGA együttes méretére, s nem az egyes példányokra vonatkozik.

3.5. Processzek

Minden Oracle adatbázishoz csatlakozott felhasználónak két kód-modult kell futtatnia ahhoz, hogy hozzáférhessen egy adatbázis példányhoz. Ezek:

- Valamilyen adatbázis **alkalmazás** (pl. előfordító program) vagy **Oracle eszköz** (pl. SQL*Plus), amely SQL utasításokat továbbít egy Oracle adatbázishoz.
- Adatbázis **szerver kód**, amely értelmezi és végrehajtja az SQL utasításokat.

Ezeket a kód-modulokat futtatják a processzek. Ennek megfelelően a processzeket is két fő csoportra oszthatjuk: *felhasználói processzek*, melyek az alkalmazás kódját futtatják, illetve *Oracle adatbázis processzek*, ahová a szerver- és háttér folyamatok tartoznak.

A processzek struktúrája függ az operációs rendszertől és az adatbázis beállításaitól is. Dedikált vagy osztott szervertes megoldások közül választhatunk. Dedikált szervertes kapcsolat esetén minden felhasználóhoz tartozik egy felhasználói (user) és egy dedikált szerver processz, míg osztott szervertes kapcsolatnál egy szerver processz akár több felhasználót is kiszolgálhat egyszerre.

4. A végállapot ismertetése

A féléves munka eredményét tulajdonképpen a már említett tanulmány [4] jelenti, amelyből itt egy kis ízelítőt próbáltam adni. Az optimalizálási dokumentációk alapján elsajátított technikákat majd a 8. szemeszteres rendszerterv elkészítésénél fogom alkalmazni.

Hivatkozások

- [1] http://www.dsvolk.ru/oracle/papers/understanding_the_oracle_optimizer.ppt, 2008. május 5., 12:56
- [2] http://en.wikipedia.org/wiki/Query_plan, 2008. május 5., 13:02
- [3] http://download.oracle.com/docs/cd/B28359_01/server.111/b28318.pdf, 2008. május 5., 13:54
- [4] http://members.chello.hu/m.nyarady/blog/beszamolok/7felev/tanulmany_qja31e.pdf, 2008. május 5., 14:09
- [5] http://download.oracle.com/docs/cd/B28359_01/server.111/b28318/memory.htm#CHDHAHIJ, 2008. május 6., 16:34