# SQL tuning

Lassan futó lekérdezések „hangolása”, teljesítménybeli elvárásoknak megfelelelésének érdekében.

Oracle-ben nagyrész automatikus, de rá lehet segíteni.

# Szintaktikai / Szemantikai ellenőrzés:

* Sintaktikai: jó lekérdezésű SQL, batartottuk e a szabályait
* Szemnatikai: a lekérdezés értelmes e, van e érelme vagy nem

# Shared Pool Check:

Az Oracle a lekérdezéshez egy hash # értéket generál és ezt kéri / ellenőrzi le. Ha van találat (korábban valaki ezt kérte), akkor is meg kell bizonyosodnia a rendszernek, hogy a felhasználónak van jogosultsága hozzáférni, lekérdeznie.

# Optimalizálás

## CBO (Cost-Based Optimization):

* Cos – Az Oracle számolja ki
* Lekérdezés teljes költségét számolja ki (megbecsüli, mert nem feltétlenül a legfirsebb a statisztika) - az Oracle részletes statisztikát vezet a táblázat objektumairól és ezekből becsüli a költségértéket
* A legkiseb költségűt választja

## Ruled based:

* Lefektettek alap szabályokat pl, ha az oszlopon van index, akkor egy where esetén előbb azt vizsgálja meg

# Row Source Generation

Miután kiválaszotta a legkisebb költségű tervet utána lefordításra kerül elemi utasításokra.

Row Source – Rekordok halmaza

A végrehajtható lépéseknek mindig van?

* egy inputja, ami lehet egy vagy több row source
* és egy outputja, ami egy row source

ID 0 (Select, legelső sor)-ban látszódik a Cost-nál az össz költséget

# Row Source Tree

Megmutatja az általa legenerált fastuktúra:

* a táblák sorrendjét
* a hozzáférés módját az egyes táblákhoz
* a táblák összekapcsolásának módját
* az egyéb műveleteket

Post order módszer szerint kell értelmezni. A szülőelem bemeneti adatai alatta lévő műveletekben lesz. (Ahhoz, hogy a szülő műveletet el tudjuk végezni, ahhoz el kell végezni az alatta lévő műveleteket.)

# Access path

A hozzáférés módját: hogyan olvasunk ki egy row source-t

## Típusai:

* **Full Table Scan**
* **Table Access by Rowid**
* **B-Fa indexek:**
  + **Index Unique Scan**
  + **Index Range Scan**
  + **Index Full Scan**
  + ...

## Full Tabel Scan:

Minen sort beolvas a táblából pl, ha:

* nincs benne index
* hint van beleírva -> kényszerítjük vele
* az egyéb módszerek költségesebbek lennének (az oszlop túl kicsi szelektivitású, ...)

## Table Acces by Rowid:

Csak a rowid sorokat akarja kiolvasni.

A rowid megmutatja hol van fizikailag a rekod.

## Index

Opcionális adatstruktúra, amely bizonyos esetekben felgyorsíthatja a rekordokhoz való hozzáférést.

## Index unique scan

Legfeljebb 1 rowid-t ad vissza (lehet, hogy nem ad vissza és azért csak max 1-et, mert unique).

## Index range scan

Egy értéktartományhoz tartozó rowid-kat ad vissza.

## Index full scan:

Kiolvassa a levél elemeket a b fából elejétől a végéig, sorrendben.

Indexelt, nem null mezőre rendezést kérünk. Ha pl rendezni akarunk akkor mégtöbb értelme van ennek.

## Index fast full scan

Nem veszi figyelembe a sorrendet, csak megkeresi a levél elemeket és különböző sorrendben kilistázza (nem kell az ’ugrálást megtartani’).

## Index skip scan

Nem az első mezőre, hanem a második mezőre keresünk. (Át lehet ugrani részt.)

Összetett indexünk van és a keresett feltétel a második oszlopára vonatkozik.

## Index join scan

Két indexből kiszedi az oszlopértékeket a rowid-kalés utána a rowid alapján összekapcsolja őket.

# Join

3 különböző algoritmussal tudja végrehajtani (meg kell határozni, hogy a 3 közül melyiket válassza + meg kell kapnia a 2 row source-t és kitalálja a módszerét, ami összefügg azzal, hogy milyen típúsú joint futtatunk és ha több táblát join-olunk össze, akkor eldönti, hogy azokat milyen sorrendben join-olja)

Cél: minél kevesebb rekorddal kelljen dolgozni

3 módszer:

* Nested loop join
* Sort-merge join
* Hash join
* (Cartesian join)

## Nested loop join

A külső row source minden sorához megkeresi a belső row source-ból a hozzá tartozót.

A kicsi row source-t választja külsőnek, ha egy nagy és kicsit szeretnénk össze join-olni.

Arra optimalizál, hogy az első néhány sort adja vissza nagyon gyorsan.

## Sort-merge join

A 2 row source-t először rendezi aztán összefésüli őket, úgy, hogy elindul először az egyikből, veszi az első sort és megnézni, hogy a másiknak a 2. sorában összepasszol e.

Ha összepasszolnak, akkor a következőt veszi még mindig a 2. táblából.

Ha nem passzolnak, akkor lép 1-et az első táblából.

Akkor használják, ha amúgy is szükséges rendezni, vagy nagy tábláink vannak.

(A rendezés dága művelet.)

## Hash join

Az egyik row source-hoz egy hash táblát épít fel.

A kisebb táblához az összekötő oszlop értékeit hash-eli és készít belőle egy hash táblát.

A második táblán az összekötő részt hash-eli és megnézi, hogy egyezik e a másik hash értékével.

Akkor tudja használni, hogy ha =-jel van a kettő között (egyezést keresünk).

## Caresian join

Minden sort minden sorral párosít.

Ha ilyen van, akkor elírás vagy hiba van valószínűleg. (Nagyon ritkán kell ténylegesen.)