SQL ohjelmoijan muistilista
DB2 v7

- Työryhmä

  Tuovi Henttu, Osuuspankki

  Ville Hurmalainen, Pohjola

  Anne Lesell, IBM

  Sinikka Mononen, Octel

  Minna Nurmela, Kela

  Paavo Tukia, Octel

SQL ohjelmoijan muistilista
DB2 v7

1. *Yllättävät saantipolut*
2. *Turhat lajittelut*
3. *CPU-ajan säästö*
4. *Turvallisuus*
5. *Selkeys ja ylläpidettävyys*
6. *Lukitukset*
7. *Yllättävät tulokset*
8. *Hajautettu ympäristö*
9. *Data Sharing*
10. *Stored Procedures*
11. *LOBit*
12. *Runstats-ohje /Joinit*
13. *XML*

1. Yllättävät saantipolut (1)

1. Muuttujien ja sarakkeiden tietomuodon tulee vastata sarakkeen tietomuotoa
2. Vältä skalaarifunktioita WHERE ehdossa kentissä joita käytetään haussa
3. Vältä laskemista WHERE lausekkeessa
4. Ehtopari >=, <= verrattaessa muuttujaa kahteen sarakkeeseen
5. Vältä negaatiota haussa käytettäville sarakkeille
6. Alikyselyssä kirjoitustapa määrää suoritustavan ja -järjestyksen
7. FETCH FIRST n ROWS ja OPTIMIZE FOR n ROWS tehostavat hakupolkua pienillä aineistoilla

1. Yllättävät saantipolut (2)

8. OR, jota ei voi esittää IN listana, aiheuttaa aina vähintään MIA hakupolun.
9. Yhden rivin haussa INDEX ONLY voidaan saada laittamalla hakuun sarakefunktio.
10. Tarkista MIN ja MAX funktioiden hakupolku EXPLAINilla.
11. NOT NULL-saraketta verrattaessa sarakkeeseen, jossa NULL-sallittu, käytä COALESCE funktiota.
    - löytyy myös IFNULL funktio…
12. *Älä tutki isäntämuuttujien arvoja WHERE lausekkeella.*
13. Älä turhaan käytä COALESCE funktiota, jos NULL ei sallittu
14. Jos taulujärjestys JOIN lausekkeessa vaihtelee, tarkempi statistiikka indeksisarakkeille voi auttaa.

Yllättävät saantipolut / Isäntämuuttujan arvon tutkiminen WHERE lausekkeella?

- Miksi ihmeessä joku tutkisi SQL:llä muuttujaa?
  - "kätevä" tapa rakentaa "yleiskäyttöisiä" SQL-lauseita
  - vastaa Javan ns. NULL ehtolauseketta eli ehto/ehdon osa voidaan kääntää pois käytöstä.
  - Tuottaa takuuvarmasti huonon hakupolun

```
SELECT ...
FROM    AGREEMENT
WHERE   START_DATE < :hv3
 AND   (   APPR_FLAG = 'VOIM'
       AND AMT     > 15000
       AND CLAIMS > 10
       OR  :hv2 = 'RISKIVAK')
 AND   (   AMT < 1000
       OR  :hv2 = 'PIENVAK') ;
```

2. Turhat lajittelut

1. Käytä ORDER BY –listassa hakemiston järjestystä
2. Jos indeksit EIVÄT tue lajittelujärjestystä, rivit lajitellaan tilapäistaulukkoon.
3. Vältä DISTINCT –määreen käyttöä. Kokeile, saatko GROUP BY:lla paremmat hakupolut.
4. Käytä UNION ALL lauseketta, ellet tarvitse duplikaattien poistoa.
5. Lajiteltaessa JOIN tulosta, jossa lajittelutekijät kohdistuvat useaan tauluun, voi syntyä ylimääräinen lajittelu.

3. CPU ajan säästö

1. *LOAD apuohjelman käyttö säästää…*
2. Jokerimerkit (%, _) LIKE ehdon ensimmäisenä merkkinä sulkee pois MIS saantipolun.
3. Määrittele kohdistin eräajoissa WITH HOLD –optiolla.
4. Olemassaolon tarkistus: FETCH FIRST ROW ONLY määreellä.
5. SELECT –lauseeseen vain ne sarakkeet, joiden tietoa tarvitset. WHERE –ehtojen sarakkeita ei pidä turhaan toistaa.
6. Vältä UPDATE –lauseessa tietoarvoltaan muuttumattomien sarakkeiden luetteleminen.
7. Vältä turhien SQL-lauseiden suorittamista.
8. CPU:n käyttö oikeaan paikkaan, ohjelma / SQL.

CPU ajan säästö / LOAD apuohjelma

- Jos tarve 24/7 käytölle, niin ratkaisu on LOAD … RESUME YES SHRLEVEL CHANGE.
- Tällöin
  - Taulua voi käyttää samalla (*ei estä muiden sovellusohjelmien suoritusta kuten normaali LOAD*)
  - toimii kuten SQL INSERT –lausein tehty ohjelma
  - säilyttää cluster-järjestyksen
  - lisäykset menevät lokille (*Log no mahdoton*)
  - viite-eheys tarkastetaan samalla (*enforce no mahdoton*)
  - lisäksi INSERT-triggerit laukeavat

4. Turvallisuus

1. Käytä rivien päivityksessä ja poistossa kohdistinta. Suora käyttö vain jos perusavain tunnetaan.

2. Jos käytät näkymän määrittelyssä WHERE –ehtoja ja näkymää käytetään rivien päivitykseen / lisäykseen, määrittele näkymä WITH CHECK OPTION määreellä.

3. Tarkasta aina SQLCODE SQL-lauseen jälkeen.

5. Selkeys ja ylläpidettävyys

1. Älä käytä SELECT * lausetta.
2. SELECT lauseeseen vain ne sarakkeet, joita todella tarvitset.
3. INSERT lauseeseen näkymän sarakkeet, joihin viet tietoa.
4. Määrittele FOR UPDATE OF –sarakelistaan vain ne sarakkeet, joita aiot päivittää. Jos rivejä poistetaan, luettele perusavaimen sarakkeet.
5. Käytä mieluummin IN listaa usean OR ehdon sijaan.
6. Käytä isäntämuuttujina DCLGEN -struktuureja.
7. Luettele ORDER BY –sarakelistassa sarakkeet nimeltä.
8. Sarakkeen nimen on hyvä olla kuvaava, kommentoi sekä anna otsakenimi (LABEL) kullekin sarakkeelle. Kun käytät samasta tiedosta aina samaa nimeä, DB2-katalogi toimii tietohakemistona.

6. Lukitukset

1. LUKEVAN kursorin kohteena olevien taulujen suora päivittäminen samassa yhteydessä (UOW) voi häiritä kursorin etenemistä. [*eli ns. kengurukursoori*]
2. Päiväeräohjelmissa voi deadlock- ja timeout-paluukoodin jälkeen (-911) harkita kesken jääneen LUW:n suoritusta uudelleen muutamia kertoja.
3. Määrittele kohdistin FOR FETCH ONLY, jos rivejä luetaan.
4. Lukevissa ohjelmissa harkitse UR-optiota (riski: epäyhtenäinen tulos).
5. Tee commit eräohjelmassa 2-5 sekunnin välein.
6. Sensitiivinen scrollable kursori pitää lukot koko suoritusajan ohjelman commitista huolimatta.

7. Yllättävät tulokset

1. Käytä ORDER BY –määrettä aina, ja vain silloin, kun tulosjoukko halutaan tietyssä järjestyksessä.
2. Varo väärää päätelmää sarake-funktioiden luvusta.
   - jos ehto jää toteutumatta, SQLCODE = 0 ja arvo on NULL
3. NULL arvoon löytyy vain ehdolla IS NULL, eikä millään muulla ehdolla. Kaikista muista kyselyistä rivi jää huomioimatta.
4. Ole varovainen WHERE –ehtojen kanssa ulkoliitoksen yhteydessä.

8. Hajautettu ympäristö

1. OPTIMIZE FOR N ROWS vaikuttaa tietoliikenneblokin kokoon.

2. Käytä hajautetussa ympäristössä kohdistimen määrityslauseessa FOR FETCH ONLY –määrettä, jos rivejä aiotaan vain lukea.

9. Data Sharing

1. Commit-väli & LOCK AVOIDANCE –optio.

2. Ohjelman suunnittelu: hyvä ei riitä, pitää olla loistava. Tiiviit tietokantakäsittelylohkot (compound block).

3. Käytä ehdottomasti LOCK AVOIDANCE –optiota, vaikutus tuntuu paljon laajemmin käydessään kaikkien koneiden läpi.

10. Stored Procedures

1. Huomioi, että kutsuttaessa CICS tapahtumasta stored procedurea, tapahtuu osoitetilan vaihto.

2. Harkitse, kannattaako sallia stored procedurien kutsu eräohjelmista.

3. Ohjelman kutsun (CALL) ja stored proc –kutsun CPU kulutuksen ero suuri (jopa 10-kertainen).

Muut

1. Perustauluun ei määritellä lob-saraketta. LOBista tehdään 1:1-liitos perustauluun
   - huoltotoimet hoituvat
   - selailussa LOB esille vain tarvittaessa
   - varchar vs LOB
     - LOB ei kirjaannu lokille
     - varchar ei tehokas, eikä kuultu mitään hyvää
2. Runstats-ohje /Joinit - lähinnä V8 teho-ongelmia
3. XML - vasta V8 ohjeisiin