RUNSTATS

Hur påverkar nya och gamla parametrar optimizern?

RUG Sandviken
23-24 januari 2007

TAN DATA AB

Frågeställningar

- Hur ska man köra sina runstats-jobb?
- I vilka situationer ska man specificera vad i runstats?
- Hur påverkas optimizern av de ”nya” parametrarna
  - **KEYCARD**
  - **FREQVAL NUMCOLS COUNT**
  - **COLGROUPS?**
- Hur hittar de kolumner jag behöver bygga upp statistik på?

Uppdatering av katalogen för optimizern

SYSTABLES

CARDF	Total number of rows in the table.
NPAGES	Total number of pages on which rows of this table are included.
NPAGESF	Total number of pages that are used by the table.
PCTROWCOMP	Percentage of rows compressed within the total number of active rows in the table.

SYSTABSTATS

CARDF	Total number of rows in the partition.
NPAGES	Total number of pages on which rows of this partition are included.

SYSCOLUMNS

COLCARDF	Estimated number of distinct values for the column. For an indicator column, this value is the number of LOBs that are not null and whose lengths are greater than zero. The value is -1 if statistics have not been gathered. The value is -2 for columns of an auxiliary table.
HIGH2KEY	Second highest value of the column. Blank if statistics have not been gathered or if the column is an indicator column or a column of an auxiliary table. If the column has a non-character data type, the data might not be printable. This column can be updated.
LOW2KEY	Second lowest value of the column. Blank if statistics have not been gathered or if the column is an indicator column or a column of an auxiliary table. If the column has a non-character data type, the data might not be printable. This column can be updated.

Observera att CARD inte blir uppdaterat för LARGE deklarerade tabeller

Uppdatering av katalogen för optimizern

SYSCOLDIST	
CARDF	The number of distinct values for the column group. This number is valid only for cardinality key column statistics. (A C in the TYPE column indicates that cardinality statistics were gathered.)
COLGROUPCOLNO	Identifies the set of columns that are associated with the key column statistics.
COLVALUE	Actual index column value that is being counted for distribution index statistics.
FREQUENCYF	Percentage of rows, multiplied by 100, that contain the values that are specified in COLVALUE.
NUMCOLUMNS	The number of columns that are associated with the key column statistics.

SYSTABLESPACE	
NACTIVE or NACTIVEF	Number of active pages in the table space; shows the number of pages that are accessed if a record cursor is used to scan the entire file. The value is -1 if statistics have not been gathered.

SYSINDEXES	
CLUSTERRATIOF	A number between 0 and 1 that, when multiplied by 100, gives the percentage of rows that are in clustering order. For example, a value of 1 indicates that all rows are in clustering order. A value of .87825 indicates that 87.825% of the rows are in clustering order.
CLUSTERING	Indication of whether CLUSTER was specified when the index was created.
FIRSTKEYCARDF	Number of distinct values of the first key column.
FULLKEYCARDF	Number of distinct values of the full key.
NLEAF	Number of leaf pages in the index.
NLEVELS	Number of levels in the index tree.

- Observera att FIRSTKEYCARD och FULLKEYCARD inte blir uppdaterat för LARGE deklarerade tabeller

Tabell och index definitioner

```
CREATE TABLE BG0TAN.TEST_RUNSTATS
(C1      CHAR(5)  NOT NULL,
 C2      CHAR(5) NOT NULL,
 C3      CHAR(5) NOT NULL,
 C4      CHAR(5) NOT NULL,
 C5      CHAR(5) NOT NULL,
 C6      CHAR(5) NOT NULL
IN ….
```

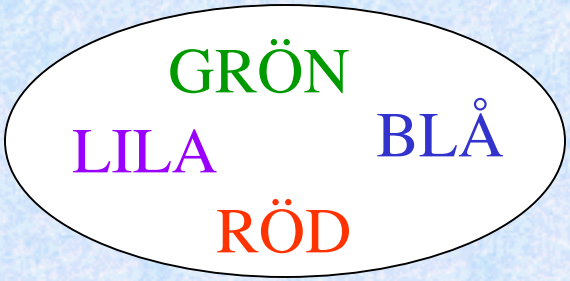
```
CREATE UNIQUE INDEX BG0TAN.ITEST_RUNSTATS_01
ON BG0TAN.TEST_RUNSTATS
(C1 ASC)
USING ….
```

```
CREATE INDEX BG0TAN.ITEST_RUNSTATS_01
ON BG0TAN.TEST_RUNSTATS
(C2 ASC,
 C3 ASC,
 C4 ASC)
USING ….
```

Kardinalitet- och frekvensstatisk

- **Kardinalitetstatistik (distribution, korrelation)**
  - Antalet unika värden för en kolumn eller grupp av kolumner
- **Frekvensstatistik**
  - Det procentuella antalet rader i tabellen som innehåller ett specifikt värde i en kolumn eller grupp av kolumner

Kardinalitet = 4
Frekvens = 25 %



Runstats på tabellnivå

```
RUNSTATS TABLESPACE BG0TANR0.BG0TANS0
   TABLE(BG0TAN.TEST_RUNSTATS)
```

- Defaultparameter för column är ALL
- Innebär att samtliga kolumner i tabellen får uppdaterad kardinalistetsstatistik
- Kardinalistesstatistik läggs in i antingen SYSCOLUMNS och SYSCOLDIST beroende på vilka parametrar som angivits
- I exemplet ovan läggs endast informationen in i SYSCOLLUMNS

SYSCOLUMNS

NAME	COLCARD	HIGH2KEY	LOW2KEY
C1	4	3	2
C2	4	RÖD	GUL
C3	4	LRÖD	LGUL
C4	4	MRÖD	MGUL
C5	4	C	B
C6	4	3	2

Runstats på tabellnivå

```
RUNSTATS TABLESPACE BG0TANR0.BG0TANS0
   TABLE(BG0TAN.TEST_RUNSTATS)
   COLUMN(C1)  COLGROUP(C3)
```

- Enstaka kolumner kan anges i **column** istället för att ange column(all) vilket ger en minskad cpu-förbrukning för stora tabeller
- Även **colgroup** samlar in kardinalitetsstatistik för icke indexerade kolumner
- Column och colgroup ger samma resultat om endast en kolumn anges
- Fortfarande finns endast kardinalitetsinformation i katalogen

SYSCOLUMNS

NAME	COLCARD	HIGH2KEY	LOW2KEY
C1	4	3	2
C2	-1	-1	-1
C3	4	LRÖD	LGUL
C4	-1	-1	-1
C5	-1	-1	-1
C6	-1	-1	-1

Runstats på tabellnivå

Hur får man då in frekvensstatistik i DB2-katalogen?

- Nya parametern **colgroup** i kombination med **freqval** måste användas för icke indexerade kolumner
- Colgroup huvudsyfte är att samla in statistik för icke indexerade kolumner, men kan med fördel även användas för att skapa förfinad statistik för indexerade kolumner
- **Keycard, numcols, count** används dock traditionellt för statistikinsamling för indexerade kolumner
- **Count** anger för hur många unika kardinalitetsvärden som det ska beräknas frekvensstatistik på. Count ska kombineras med **least**, **most** eller **both**

Runstats på tabellnivå

```
RUNSTATS TABLESPACE BG0TANR0.BG0TANS0
   TABLE(BG0TAN.TEST_RUNSTATS)
   COLUMN(C1) COLGROUP(C3) FREQVAL COUNT 25
```

- Via **freqval** erhålls frekvensstatistik för C3
- Frekvensstatistiken hittar man nu i SYSCOLDIST under TYPE=F
- Kardinalitetsstatistiken finns som vanligt i SYSCOLUMNS

SYSCOLUMNS

NAME	COLCARD	HIGH2KEY	LOW2KEY
C1	4	3	2
C2	-1	-1	-1
C3	4	LRÖD	LGUL
C4	-1	-1	-1
C5	-1	-1	-1
C6	-1	-1	-1

SYSCOLDIST

FREQ	NAME	COLVALUE	NUMCOLS	COLGROUPCOLNO	TYPE
2500	C3	LGUL	1	.. 0003	F
2500	C3	LBLÅ	1	.. 0003	F
2500	C3	LRÖD	1	.. 0003	F
2500	C3	LVIT	1	.. 0003	F

Runstats på tabellnivå

```
RUNSTATS TABLESPACE BG0TANR0.BG0TANS0
   TABLE(BG0TAN.TEST_RUNSTATS)
   COLUMN(C1)  COLGROUP(C5,C6)
```

- Möjlighet att skapa kardinalitetsstatistik på grupper av kolumner
- Statistiken finns nu både i SYSCOLUMNS och i SYSCOLDIST under TYPE=C
- I SYSCOLDIST heter ”gruppen” C5
- I SYSTABLES har C5 fått kardinalitetsstatistik men inte C6

SYSCOLUMNS

NAME	COLCARD	HIGH2KEY	LOW2KEY
C1	4	3	2
C2	-1	-1	-1
C3	4	LRÖD	LGUL
C4	-1	-1	-1
C5	4	C	B
C6	-1	-1	-1

SYSCOLDIST

FREQ	NAME	COLVALUE	NUMCOLS	COLGROUPCOLNO	TYPE
-1	C5		2	 00050006	C
2500	C3	LGUL	1	.. 0003	F
2500	C3	LBLÅ	1	.. 0003	F
2500	C3	LRÖD	1	.. 0003	F
2500	C3	LVIT	1	.. 0003	F

Runstats på tabellnivå

```
RUNSTATS TABLESPACE BG0TANR0.BG0TANS0
   TABLE(BG0TAN.TEST_RUNSTATS)
   COLUMN(C1) COLGROUP(C5,C6) FREQVAL COUNT (25)
```

- Genom att lägga till **freqval** för colgroup får man också frekvensstatistik för gruppen C5, C6

SYSCOLUMNS

NAME	COLCARD	HIGH2KEY	LOW2KEY
C1	4	3	2
C2	-1	-1	-1
C3	4	LRÖD	LGUL
C4	-1	-1	-1
C5	4	C	B
C6	-1	-1	-1

SYSCOLDIST

FREQ	NAME	COLVALUE	NUMCOLS	COLGROUPCOLNO	TYPE
-1	C5		2	 00050006	C
2500	C5	A 1	2	 00050006	F
2500	C5	B 2	2	 00050006	F
2500	C5	C 3	2	 00050006	F
2500	C5	D 4	2	 00050006	F
2500	C3	LGUL	1	.. 0003	F
2500	C3	LBLÅ	1	.. 0003	F
2500	C3	LRÖD	1	.. 0003	F
2500	C3	LVIT	1	.. 0003	F

Runstats på tabellnivå

Varför statistik för icke indexerade kolumner?

- Används av optimizern för att ta beslut om vilken tabell som ska accessas först vid join

Några iakttagelser runt colgroup?

- Ny möjlighet att bygga statistik på grupper av **både** indexerade och icke indexerade kolumner
- Kan vara problematiskt att ”hitta” rätt kolumn-kandidater till **colgroup**
- Colgroup på en kolumn utan att ange **freqval** tillför inget utöver vad som tidigare funnits
- Ger upphov till en stor SYSCOLDIST-tabell

Sample

- I sample specificeras hur många procent av tabellraderna som statistiken i kolumnen **colcardf** ska baseras på
- **Colcardf** innehåller det ***uppskattade*** antalet distinkta värden för en specifik kolumn om sample är < 100
- Tas ett värde under 100 % så baseras statistiken i colcardf på ett slumpmässigt utvalt antal rader i tabellen
- Defaultvärde för sample är 25%, ett värde under 10% rekommenderas ej
- Alla andra kolumner får korrekta värden vid sample < 100%
- Sample´s syfte är att minska CPU-förbrukningen för tabeller som innehåller stora volymer

Runstats på indexnivå

```
RUNSTATS TABLESPACE BG0TANR0.BG0TANS0
   TABLE(BG0TAN.TEST_RUNSTATS)
   INDEX(ALL)
```

- Även för index finns möjlighet att skapa både kardinalitets- och frekvensstatisk
- Statistiken återfinns i SYSCOLDIST och SYSINDEXES
- Önskat index kan specificeras i runstaten alternativt ges värdet INDEX(ALL) vilket skapar statistik för en tabells samtliga index
- Kardinalitetsstatistik skapas via **keycard**
- Frekvensstatistik skapas normalt via **freqval**, *men frekvensstatistik skapas ibland utan att freqval har angivits*

Runstats på indexnivå

- I SYSINDEXES har bl.a. kolumnerna **first-** och **fullkeycard** fått värden får bägge indexen
- För icke unika index finns också frekvensstatistik i SYSCOLDIST på första kolumnen i indexet (i exemplet kolumn C2), trots att ingen frekvensstatistik begärts via **freqval**
- För det unika indexet finns ingen frekvensstatistik, indexet är ju unikt

SYSINDEXES

NAME	UNIQUE	COLCO	FIRST KEYCARD	FULL KEYCARD
IX1	JA	1	4	4
IX2	NEJ	3	4	4

SYSCOLDIST

FREQ	NAME	COLVALUE	NUMCOLS	COLGROUPCOLNO	TYPE
2500	C2	VIT	1	 0002	F
2500	C2	RÖD	1	 0002	F
2500	C2	GUL	1	 0002	F
2500	C2	BLÅ	1	 0002	F

Vad vet DB2 om våra index nu?

Unika indexet

C1

SYSINDEXES

NAME	UNIQUE	COLCO	FIRST KEYCARD	FULL KEYCARD
IX1	JA	1	4	4

- DB2 känner till värdena för **FIRSTKEYCARD** och **FULLKEYCARD** (kardinalitet)
- Indirekt så vet också DB2 frekvensstatistiken för C1

Icke unika indexet

C2 C3 C4

SYSINDEXES

NAME	UNIQUE	COLCO	FIRST KEYCARD	FULL KEYCARD
IX2	NEJ	3	4	4

SYSCOLDIST

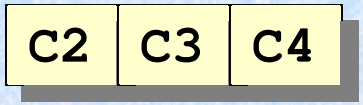
FREQ	NAME	COLVALUE	NUMCOLS	COLGROUPCOLNO	TYPE
2500	C2	VIT	1	 0002	F
2500	C2	RÖD	1	 0002	F
2500	C2	GUL	1	 0002	F
2500	C2	BLÅ	1	 0002	F

- DB2 känner till värdena för **FIRSTKEYCARD** och **FULLKEYCARD** (kardinalitet)
- DB2 har frekvensstatistik för C2

(*Statistiken för C3 ej med i detta exempel*)

Vad vet DB2 inte om vårt data?

Icke unika indexet



- Hur väl kolumnerna C3 respektive C4 filtrerar vårt data vet inte DB2
- Inte heller hur väl kolumnkombinationen C2 och C3 filtrerar våra villkor
- Både kardinalitet och frekvensstatistik saknas för dessa kombinationer
- Dessa uppgifter kan tas fram via parametrarna **keycard** och **freqval**

Runstats på indexnivå

- **Keycard** sammanställer kardinalitetsstatistik om alla unika värden i olika kombinationer i indexet enligt principen,
  - C2,
  - C2, C3
  - C2, C3, C4
- **Freqval** sammanställer frekvensstatistik för olika kombinationer i indexet, där **numcols** anger vilken kombination som ska sammanställas
- Hur många frekvensvärden som ska sammanställas styrs via parametern **count**
- Till count hör även här parametrarna **least, most** och **both**

Runstats på indexnivå

```
RUNSTATS TABLESPACE BG0TANR0.BG0TANS0
  TABLE(BG0TAN.TEST_RUNSTATS)
  INDEX(ALL) KEYCARD FREQVAL NUMCOLS 3 COUNT(10)
```

- Eftersom det icke unika indexet har tre kolumner anges 3 vid **numcols**
- Följande resultat erhålls:
  - Kardinalitetetsstatistik hittas i SYSCOLDIST för kombinationen C2, C3
  - Frekvensstatistik finns också i SYSCOLDIST för C2, C3, C4

SYSCOLDIST	COL	NUM		
FREQ	NAME	VALUE	COLS	COLGROUPCOLNO	TYPE
-1	C2		2	.. 00020003	C
2500	C2	VIT	1	.. 0002	F
2500	C2	RÖD	1	.. 0002	F
2500	C2	GUL	1	.. 0002	F
2500	C2	BLÅ	1	.. 0002	F
2500	C2	VIT LVIT MVIT	3	 000200030004	F
2500	C2	RÖD LRÖD MRÖD	3	 000200030004	F
2500	C2	GUL LGUL MGUL	3	 000200030004	F
2500	C2	BLÅ LBLÅ MBLÅ	3	 000200030004	F

Vart tog resten av kombinationerna vägen?

- Kardinaliteten för C1 finns i SYSINDEXES i **Firstkeycard** och kombinationen C2, C3, C4 finns i **Fullkeycard**
- Frekvensstatistiken för C2, C3 saknas dock!
- **Keycard** bygger upp kardinalitetsstatistik för alla indexets kolumner, men för frekvensstatistiken får man statistik exakt för det antal kolumner man skrivit i **freqval/numcols**, (3 i exemplet).
- För att få frekvensstatistik även för kombinationen C2, C3 måste följande specificeras

```
RUNSTATS TABLESPACE BG0TANR0.BG0TANS0
   TABLE(BG0TAN.TEST_RUNSTATS)
   INDEX(ALL)  KEYCARD
               FREQVAL NUMCOLS 1 COUNT(10),
               FREQVAL NUMCOLS 2 COUNT(10),
               FREQVAL NUMCOLS 3 COUNT(10)
```

Följande indexinformation finns nu i katalogen

SYSCOLDIST FREQ	NAME	COL VALUE	NUM COLS	COLGROUPCOLNO	TYPE
-1	C2		2	.. 00020003	C
2500	C2	VIT	1	.. 0002	F
2500	C2	RÖD	1	.. 0002	F
2500	C2	GUL	1	.. 0002	F
2500	C2	BLÅ	1	.. 0002	F
2500	**C2**	**VIT LVIT**	**2**	**.... 00020003**	**F**
2500	**C2**	**RÖD LRÖD**	**2**	**.... 00020003**	**F**
2500	**C2**	**GUL LGUL**	**2**	**.... 00020003**	**F**
2500	**C2**	**BLÅ LBLÅ**	**2**	**.... 00020003**	**F**
2500	C2	VIT LVIT MVIT	3	 000200030004	F
2500	C2	RÖD LRÖD MRÖD	3	 000200030004	F
2500	C2	GUL LGUL MGUL	3	 000200030004	F
2500	C2	BLÅ LBLÅ MBLÅ	3	 000200030004	F

- DB2 vet fortfarande inte hur kombinationen C2 och C4 i indexet filtrerar
- Här kan **colgroup** parametern utnyttjas även för indexen genom att gruppera ihop kolumnerna C2 och C4

SYSINDEXES NAME	UNIQUE	COLCO	FIRST KEYCARD	FULL KEYCARD
IX1	JA	1	4	4
IX2	NEJ	3	4	4

Frågeställningar

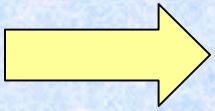
- Utnyttjar optimizern verkligen all denna information?
- Hur hittar jag alla kombinationer som är viktiga för optimizern?
- Hur många rader ska jag lägga in i katalogen (count)?
- Vad händer med SYSCOLDIST tabellen? Den förefaller kunna bli väldigt stor
- Är inte antalet matchande kolumner i en SQL-en mer betydande än dess filterfaktorer?
- Ska jag ändra alla mina runstatsjobb i produktion och i så fall hur gör jag det?

Varför frekvensstatistik - Skevt data

Nedanstående SQL har dålig performance

```
SELECT O.*, B.*
FROM   ORDER O INNER JOIN BIL B
ON     O.BILMÄRKE    = B.BILMÄRKE
WHERE  O.FNAMN       = 'BERGSTRÖM'
AND    O.LEV_STATUS  = 'BEST'
AND    B.MODELL      = 'SEDAN'
AND    B.LAND        = 'SE'
AND    B.FÄRG        = 'SILVER'
AND    B.MOTOR       = '2,8 TK'
```

Optimizern väljer att starta joinen i BIL-tabellen

BIL  ORDER

Varför frekvensstatistik - Skevt data

Följande frågor påvisar följande förhållande i tabellerna

```
SELECT COUNT(*)
FROM   ORDER O
WHERE  O.FNAMN       = 'BERGSTRÖM'
AND    O.LEV_STATUS  = 'BEST'
```

< 1% av raderna uppfyller villkoren

```
SELECT COUNT(*)
FROM   BIL B
WHERE  B.MODELL      = 'SEDAN'
AND    B.LAND        = 'SE'
AND    B.FÄRG        = 'SILVER'
AND    B.MOTOR       = '2,8 TK'
```

60% av raderna uppfyller villkoret

Optimizern borde ha valt att börja med ORDER-tabellen istället

Varför frekvensstatistik - Skevt data

Låt oss titta närmare på hur mycket varje villkor mot biltabellen filtrerar

```
B.MODELL      = 'SEDAN'
       9 olika modeller ger
       1/9 = 11,11%

B.LAND        = 'SE'
       227 olika länder ger
       1/227 = 0,44%

B.FÄRG        = 'SILVER'
       29 olika färger
       1/29 = 3,45%

B.MOTOR       = '2,8 TK'
       12 olika motorer ger
       1/12 = 8,33%
```

Antalet uppskattade rader tillbaka från biltabellen blir 0,00014 procent men vår SQL-sats gav 60 procent tillbaka

Vad är fel?

Varför frekvensstatistik - Skevt data

Ytterliggare SQL-frågor mot varje kolumn i sökvillkoret påvisar ett ojämnt fördelat data i tabellen

```
SELECT LAND, COUNT(*)
FROM BIL
GROUP BY LAND
ORDER BY 2 DESC
```

```
LAND
SE      4418514          (90,4%)
DE       160065
FR        28996
IT        19654
DK        15499
NO        15461
SF        14113
NL         8078
. .     . . . .
```

Jämn fördelning **1/227 = 0,44 %**

Faktisk fördelning **SE = 90,4 %**

Indexhantering - Skevt data

KOLUMN	UPPSKATTAD FÖRDELNING	AKTUELL FÖRDELNING
MODELL	11,11%	86,43%
LAND	0,44%	90,40%
FÄRG	3,45%	95,75%
MOTOR	8,33%	83,06%

Stor skillnad mellan uppskattad och faktiskt fördelning av värdena i vår fråga

Optimizern grundar sin beräkning på felaktiga förutsättningar

- Före korrekt information fanns i katalogen
  - Biltabellen valdes som första tabell : Svarstid > 30 minuter
- Efter uppdaterad information i katalogen
  - Ordertabellen valdes som första tabell : Svarstid < 1 minut

---

- Inget svar efter 10 timmar och 54 minuter
- Svar på 1 minut och 22 sekunder
- Skillnaden var rätt information om tabellens datastruktur i katalogen

Statistik vs dynamisk SQL

Kan all SQL utnyttja all statistik?

- Frekvensstatistik utnyttjas till fullo
  - för dynamisk SQL
- Frekvensstatistiken kan utnyttjas för statistik SQL om
  - literaler finns med i SQL-en
  - nullvärden finns med i statistiken
  - paketet är bundet med Reopt(always/vars)
- Kardinalitetsstatistiken utnyttjas för
  - dynamisk SQL
  - statisk SQL

Hur hittar man alla kombinationer?

- Manuella SQL-granskningar
- ”Long running queries”
- Visual Explain - Statistic Advisor
    - Runstats recommendation
    - Baserad på enskild SQL-fråga i V8
    - I V9 kommer utvärdering att ske på workload-basis

Hur många värden ska räknas?

- 10 är default
- Beror på hur ojämnt fördelat datat i tabellen är
- 10 räcker för en klar majoritet av alla frågor
- Finns färre värden än 10 byggs statistik på befintliga värden
- Läggs värden till efteråt och ingen ny runstats körs antas de nya värdena innehålla noll

Storleken på SYSCOLDIST

- Tabellen kan svälla om ”full statistik” tas på allt
- Kardinalitetsstatistiken i SYSCOLDIST uppdateras när ny runstats körs
- Frekvensstatistiken däremot läggs till i SYSCOLDIST och den gamla informationen finns kvar
- För att få bort rader kan
  - drop/create av index göras
  - runstats med freqval count(0) exekveras för de uppgifter som man vill ta bort ur katalogen
  - ”Runstat options” INDEX(ALL) KEYCARD exekveras vilket tar bort statistikrader där count > 10 för statistik på första kolumnen i indexet

Ska jag ändra mina runstatsjobb?

- Runstats med
  - TABLE(ALL)
  - COLUMN(ALL)
  - INDEX(ALL)
  - KEYCARD

  är tillräcklig i så gott som samtliga fall
- Det viktigaste är att runstats exekveras vid rätt tidpunkt och inte att alla tänkbara statistikkombinationer byggs upp

Om jag vill ändra mina runstatsjobb?

- Om man vill automatisera en mer förfinad statistik så kan det vara problematiskt att få till i sedvanliga verktyg
- CA och IBMs verktyg bygger ej upp alla kombinationer på freqval/numcols på ett index
- En ”nödlösning” för att automatiskt bygga upp frekvens-statistik på index är att ange numcols till 99

Några slutsatser

- Runstats med TABLE(ALL), COLUMN(ALL), INDEX(ALL), KEYCARD är tillräckligt ”bra” i så gott som samtliga fall
- Kardinalitetsstatistik används även för statistik SQL vilket gör den viktigare i många avseenden än frekvensstatistik
- Överväg att använda colgroup på kolumner som är med i SQL-villkor vid join. Syftet är att hjälpa optimizern att joina tabellerna i ”rätt ordning”
- Utnyttja även colgroup för frekvensstatistik som freqval/numcols inte klarar av att bygga upp
- Frekvensstatistik rekommenderas att användas då ”fel” sökväg har gjorts av optimizern, t.ex. på grund av skevt data

Några slutsatser

- Kardinlitetsstatistik och frekvensstatistik byggs upp olika av runstats. För frekvensstatistik är man tvungen att själv ange alla kombinationer man vill ha statistik på
- Defaultvärdena för sample och count är ”nöjaktiga”
- *Version 8 tycks vara betydligt mer ”intolerant” för bristfällig statistik i katalogen*

Till sist

- Statistikuppgifter kan även specificeras i LOAD, REBUILD INDEX och REORG
- Går dock inte att få med colgroup freqval count i dessa utilities
- Observera även att om en load eller rebuild index körning återstartas efter en abend så kommer ingen statistik att byggas upp överhuvudtaget
- För en reorg beror det på hur och var jobbet återstartas om statistiken byggs upp eller ej, *se Utility Guide 2.23.2.6.2 Restarting REORG TABLESPACE*
- För dynamisk SQL så sätts eventuella SQL-frågor som refererar till ett objekt som det körs runstats på till ”invalid”. Gäller även för en data sharing miljö.

Kontaktinfo

Torbjörn Andersson
TAN Data AB
torbjorn.andersson@tandata.se
www.tandata.se

www.tanproduction.com
www.kurspecialisterna.com

Materialet finns på
www.tandata.se | TAN Data | DB2-Länkar | DB2 Info