CURSOR Software AG
Ihr IBM Distributionspartner

Tech-Tipp:
Auto-Update-Statistics – Functional Indices

Im Tech-Tipp des Newsletters 2009/07 wurde auf zwei Restriktionen des Automatic Update Statistics Feature (AUS) der IDS-Version 11.50 hingewiesen. Eine der Restriktionen ist, dass AUS Spalten in funktionalen Indices nicht berücksichtigt.

Der folgende Beitrag beschreibt die Hintergründe und einen Workaround, diese Restriktion aufzuheben.

Funktionsbasierte Indices

Mit der Version 9.x wurde die Systemtabelle *sysindexes* durch die Tabelle *sysindices* ersetzt. Wurden die Indexspalten einer Tabelle bis Version 7.x durch die Spalten *sysindexes.part1* bis *sysindexes.part16* beschrieben, so besitzt die Tabelle *sysindices* hierzu eine Spalte *indexkeys*.
Da diese Spalte in einem benutzerdefinierten Format (Datentyp *indexkeyarray*) vorliegt, kann auf ihren Inhalt lediglich über hierzu vorgesehene Schnittstellen zugegriffen werden. Der benutzerdefinierte Datentyp stellt hierfür drei Funktionen zur Verfügung:

- indexkeyarray_out(indexkeyarray) returning lvarchar
  - gibt eine String-Repräsentation von indexkeyarray zurück
- indexkeyarray_send(indexkeyarray) returning sendrecv
  - liefert einen benutzerdefinierten Datentyp zurück, für den keine Zugriffsmethode veröffentlicht ist.
- ikeyextractcolno (indexkeyarray, integer) returning smallint
  - liefert den Spaltenindex aus indexkeyarray an der über das 2. Argument beschriebenen Position.

Zur Verdeutlichung werden die Funktionen im Folgenden auf ein Testbeispiel angewandt.

Hierzu dient folgende Tabelle *testtab*:

```
create table testtab(f1 integer, f2 varchar(12), f3 integer)
```

mit den Indices:

```
create index testtab_i1 on testtab (f1, f3);
create index testtab_i2 on testtab (mylower(f2), f1);
```

Die Funktion *mylower* ist ein Wrapper um die eingebaute Funktion *lower*, da funktionale Indices keine eingebauten Funktionen erlauben:

```
create function mylower(arg1 varchar(255))
returning varchar(255)
with (NOT VARIANT)
        return lower(arg1);
end function;
```

Anhand der Ausgabe des folgenden Selects auf Tabelle *sysindices* ist zu erkennen, dass die Funktion *ikeyextractcolno* keine Spaltenindices zu Index *testtab_i2* ausgibt:

```
select
        idxname,
        indexkeyarray_out(indexkeys) keyarray,
        ikeyextractcolno(indexkeys,0) colno1,
        ikeyextractcolno(indexkeys,1) colno2
from sysindices s, systables t
where s.tabid = t.tabid
and t.tabname = 'testtab'
order by 1
```

Ausgabe:

```
idxname      keyarray             colno1  colno2
------------ -------------------- ------- -------
testtab_i1   1 [1], 3 [1]         1       3
testtab_i2   <447>(2) [1], 1 [1]  0       0
```

Funktionale Indexspalten werden in folgender Form beschrieben:
<sysprocedures.procid> (syscolumns.colno)

Aus Gründen der Abwärtskompatibilität steht anstelle der Tabelle *sysindexes* eine gleichnamige View auf Basis der Tabelle *sysindices* zur Verfügung. Im Fall eines funktionsbasierten Indexes liefern die Spalten *sysindexes.part1* bis *sysindexes.part16* den Wert 0.

Automatic Update Statistics

Auch Spalten in funktionsbasierten Indices sollten in einer Strategie zur Aktualisierung der Statistiken berücksichtigt werden, da davon ausgegangen werden darf, dass das Zugriffsverfahren über den Index von der Werteverteilung der Spalten abhängig ist. Ursache für die fehlende Unterstützung funktionaler Indices ist die Nutzung der Funktion *ikeyextractcolno.*

Diese Funktion wird in zwei Funktionen innerhalb Automatic Update Statistics verwendet:
Funktion *aus_setup_mon_table_profile* testet damit, ob ein geforderter Index auf Tabelle *mon_table_profile* existiert. Die Funktion muss daher in diesem Zusammenhang nicht betrachtet werden.

Wesentlich ist jedoch die Funktion *aus_load_dbs_data*, die eine eindeutige Liste aller Indexspalten liefern soll. Das Ergebnis wird von folgendem (vereinfachten) Select geliefert:

```
SELECT
      C.tabid, C.colno, MAX(C.colname), 'N'
FROM
      syscolumns AS C,
      sysindices AS I
WHERE C.tabid = I.tabid
AND C.colno IN (
ABS(ikeyextractcolno(indexkeys,0)),
ABS(ikeyextractcolno(indexkeys,1)),
ABS(ikeyextractcolno(indexkeys,2)),
ABS(ikeyextractcolno(indexkeys,3)),
ABS(ikeyextractcolno(indexkeys,4)),
ABS(ikeyextractcolno(indexkeys,5)),
ABS(ikeyextractcolno(indexkeys,6)),
ABS(ikeyextractcolno(indexkeys,7)),
ABS(ikeyextractcolno(indexkeys,8)),
ABS(ikeyextractcolno(indexkeys,9)),
ABS(ikeyextractcolno(indexkeys,10)),
ABS(ikeyextractcolno(indexkeys,11)),
ABS(ikeyextractcolno(indexkeys,12)),
ABS(ikeyextractcolno(indexkeys,13)),
ABS(ikeyextractcolno(indexkeys,14)),
ABS(ikeyextractcolno(indexkeys,15)))
GROUP BY 1,2,4
```

In Funktion *aus_load_dbs_data* liefert die View zusätzlich per Union-Operator Treffer aus Tabelle *sysdistrib*, die jedoch hier außer Betracht bleiben können.

Für die Tabelle *testtab* liefert diese Select-Anweisung folgendes Resultat:

```
tabid        colno
------------ ------- ------- ----
3853         1       f1      N
3853         3       f3      N
```

Spalte *f2* wird hierbei nicht gefunden.

Da keine intelligentere Schnittstelle zu dem Datentyp *indexkeyarray* existiert, kann alternativ die Ausgabe der Funktion *indexkeyarray_out* ausgewertet werden.

Eine Indexspalten kann entweder als einfache Zahl oder als Funktionsargument in runden Klammern im Rückgabewert von *indexkeyarray_out* erscheinen (siehe oben beschriebene Ausgabe der Select-Anweisung auf sysindices).

Daher liefert die folgende Select-Anweisung das gleiche Ergebnis wie oben, ist jedoch vollständig:

```
SELECT
      C.tabid, C.colno, MAX(C.colname), 'N'
FROM
      syscolumns AS C,
      sysindices AS I
WHERE C.tabid = I.tabid
AND (' ' || indexkeyarray_out(indexkeys)
    MATCHES '*[( ]' || C.colno || '[), ]*')
GROUP BY 1,2,4
```

Ausgabe:

```
tabid        colno
------------ ------- -------- -----
3853         1       f1       N
3853         2       f2       N
3853         3       f3       N
```

Mit der folgenden Insert-Anweisung werden die Treffer der View in eine Tabelle der Sysadmin-Datenbank übertragen:

```
INSERT INTO aus_work_icols
    (aus_icols_tabid, aus_icols_colno,aus_icols_colname,aus_icols_lkey)
    SELECT tabid, colno, colname,lkey FROM aus_view
```

Für die vierte Spalte *aus_icols_lkey*, die beschreibt, ob die Tabellenspalte als führende Indexspalte vorkommt, liefert die View die Konstante ‚N'.
Dieser Wert wird in einer nachfolgenden Update-Anweisung geändert:

```
UPDATE aus_work_icols SET aus_icols_lkey = 'Y'
WHERE aus_icols_tabid*65536 + aus_icols_colno
IN (
      SELECT tabid*65536 + ABS(ikeyextractcolno(indexkeys,0))
      FROM sysindices
)

1 Row(s) affected
```

Das Update trifft erwartungsgemäß nur die Spalte *f1*.

Diese SQL-Anweisung ist folgendermaßen zu ändern:

```
UPDATE aus_work_icols SET aus_icols_lkey = 'Y'
WHERE EXISTS (
  SELECT * FROM sysindices i
  WHERE aus_work_icols.aus_icols_tabid = i.tabid
  AND (indexkeyarray_out(indexkeys) LIKE
      aus_work_icols.aus_icols_colno || ' %'
   OR substr(indexkeyarray_out(indexkeys), 1,
    instr(indexkeyarray_out(indexkeys), ')'))
   matches '<*[( ]' || aus_icols_colno || '[,)]*'
  )
)

2 Row(s) affected
```

Hierdurch werden als führende Indexspalten *f1* und *f2* gekennzeichnet.
Die erste Bedingung der mit OR verknüpften Bedingungen trifft führende Indexspalten, während die zweite Bedingung Spalten in führenden funktionalen Indices findet.
Für die zweite Bedingung wird eine Funktion *instr* verwendet, die die Position des ersten Zeichens ')' in dem Index-String zurückgibt. Sie ist funktional an die Oracle-Funktion *instr* angelehnt:

```
CREATE FUNCTION instr(
    p_src     VARCHAR(254),
    p_delim   CHAR(1))

RETURNING SMALLINT

    DEFINE v_pos                    SMALLINT;
    DEFINE v_error_code             INTEGER;
    DEFINE v_isam_code              INTEGER;
    DEFINE v_error_message          VARCHAR(254);

    ON EXCEPTION SET v_error_code, v_isam_code, v_error_message
        RAISE EXCEPTION -746, 0, 'Fehler in instr: ' ||
             v_error_message || ' [' || v_error_code || '] ' ||
             length(p_src) || ' ' || p_src;
    END EXCEPTION

    IF p_src IS NULL THEN
        RETURN 0;
    END IF

    FOR v_pos = 1 TO length(p_src)
        IF substr(p_src, v_pos, 1) = p_delim THEN
            RETURN v_pos;
        END IF
    END FOR

    RETURN 0;

END FUNCTION
```

Es ist zu empfehlen, die Funktion *aus_load_dbs_data* aus der Datei *$INFORMIXDIR/etc/sysadmin/sch_aus.sql* in eine neue SQL-Datei auszulagern und dort zu ändern.

Sobald die Funktion in der Datenbank *sysadmin* neu erstellt wird, wird ein erneuter Start des Tasks *Auto Update Statistics Evaluation* die Änderung berücksichtigen.

Die erforderlichen Quellen können über ibm-support@cursor.de angefordert werden. Sie werden Ihnen kostenlos zum Download bereitgestellt.

Ihr Ansprechpartner:

Rüdiger Teves | CURSOR-Support | ibm-support@cursor.de | www.cursor-distribution.de

---