CURSOR Software AG
Ihr IBM Distributionspartner

Tech-Tipp: Auto-Update-Statistics - LOCALE-Settings

Mit der Version 11.50 lieferte IBM-Informix erstmals ein Verfahren zum automatischen Aktualisieren der Statistiken aus: **Automatic Update Statistics (AUS).**

AUS besteht aus einer Anzahl von SPL-Prozeduren und Tabellen in der Sysadmin-Datenbank.
Start und Kontrolle von AUS sind in den bereits mit Version 11.10 erschienenen Scheduler integriert.
AUS hat noch einige Restriktionen, die dessen Verwendung auf folgende Bedingungen einschränkt:

- DB_LOCALE der User-Datenbanken muss der DB_LOCALE der Sysadmin-Datenbank entsprechen.
- AUS unterstützt keine funktionalen Indizes.

Wir beschreiben Ihnen hier einen Lösungsweg, AUS auch mit einer anderen DB_LOCALE als der Standard-Locale der Sysadmin-Datenbank zu betreiben. Am Ende des Dokuments finden Sie einen Java-Programmcode, mit dem Sie eine funktionsfähige Test-/Demo-Version erstellen können.

Eine Unterstützung funktionaler Indizes wird in einer der nächsten Ausgaben unserer Tech-Tipps vorgestellt.

Problemstellung
Die Problematik bei abweichenden DB-Locales entsteht dadurch, dass die AUS-Prozeduren in einer Connection an der Sysadmin-Datenbank SQL-Anweisungen zu Tabellen von User-Datenbanken ausführen. So konstruiert die Prozedur *aus_load_dbs_data* beispielsweise eine dynamische SQL-Anweisung unter Beteiligung von Datenbanktabellen mehrerer Datenbanken und führt diese per "EXECUTE IMMEDIATE" aus:

```
INSERT INTO sysadmin:aus_work_info (<columnlist>)
SELECT
col1, ...
(SELECT ... FROM sysmaster:systabnames)
FROM <user-database>:systables
```

wobei <user-database> der Datenbankname jeder Datenbank sein kann, die für AUS zugelassen ist.

Die Funktion *aus_refresh_stats* führt dynamisch konstruierte Update Statistics Anweisungen zu Tabellen von User-Datenbanken aus. Werden User-Datenbanken in einer anderen Locale als en_us.819 erstellt, so tritt beim Start der AUS-Tasks der Fehler -23197 "Database locale information mismatch" bei einer der oben genannten AUS-Funktionen auf.

Lösungsweg
Zur Beseitigung dieses Problems müssen die Datenbanken sysadmin, sysmaster und die von AUS zu berücksichtigenden User-Datenbanken in einer einheitlichen Locale erstellt werden.

Folgende Schritte sind hierzu notwendig:

- Sysmaster
  - Änderung der Zuweisung an Variable DB_LOCALE im Skript etc/buildsmi
  - Neu Erstellen der Sysmaster-Datenbank durch Aufruf des Skripts buildsmi

- Sysadmin
  - Setzen der Umgebungsvariable DB_LOCALE
  - Aufruf folgender SQL-Dateien in etc/sysadmin mit *dbaccess*:

```
dbaccess - db_uninstall
dbaccess - db_create
dbaccess sysadmin db_install
dbaccess sysadmin start
dbaccess sysadmin sch_tasks
dbaccess sysadmin sch_aus
```

- User-Datenbanken
  - Setzen der Umgebungsvariable DB_LOCALE
  - Erstellen der Datenbanken

Datenbanken mit einer weiterhin abweichenden DB-Locale können durch den Start der folgenden SQL-Anweisung von der Behandlung durch AUS ausgeschlossen werden:

```
INSERT INTO ph_threshold (
      id,name,task_name,value,value_type,description)
SELECT
      0,
      "AUS_DATABASE_DISABLED",
      "Auto Update Statistics Evaluation",
      dbs_dbsname,
      "STRING",
      "disable database for AUS"
FROM sysmaster:sysdbslocale
WHERE dbs_collate  <> (
      SELECT dbs_collate from sysmaster:sysdbslocale
      WHERE dbs_dbsname = "sysadmin")
```

Anschließend gelingt ein manueller Start der AUS-Evaluator-Funktion per *dbaccess sysadmin*, sofern die Session mit der korrekten DB_LOCALE gestartet wird:

```
execute procedure sysadmin:aus_evaluator('F')
```

Wird der Task „Auto Update Statistics Evaluation“ jedoch über den Scheduler gestartet, scheitert er wiederum mit dem Fehler-23197 „Database locale information mismatch".
Ursache hierfür ist, dass die Session vom Scheduler mit DB_LOCALE=en_us.819 gestartet wird.
Dieses Verhalten ist nicht über eine dokumentierte Methode zu steuern.

Eine Möglichkeit AUS unter diesen Bedingungen nutzen zu können, ist der Aufruf aus einer Umgebung mit einer selbst definierten DB_LOCALE.

Die folgende Vorgehensweise versucht sowohl den Schedule-Mechanismus als auch die eingebauten AUS-Routinen zu nutzen. Im Wesentlichen sind folgende Bedingungen einzuhalten:

- Start einer UDR durch den Scheduler der Datenbankserverinstanz
- Aufruf der AUS-Routinen mit gesetzter DB_LOCALE.

Erreicht wird dies durch eine Java-UDR, die vom Scheduler aufgerufen wird, eine separate Connection mit gesetzter DB_LOCALE zur Sysadmin-Datenbank aufbaut und die AUS-Routinen mit dieser Datenbankverbindung ausführt.

Folgende Schritte sind hierzu erforderlich:

- Erstellen von den AUS-UDRs entsprechenden Java-Methoden
  und den zu deren Aufruf erforderlichen Java-UDRs

  Jede vom Scheduler aufgerufene AUS-Funktion wird durch eine Java-UDRs ersetzt, die eine Java-Methode aufruft. Die Signatur der Java-Methode ist analog zu der Signatur der AUS-Funktion bzw. Java-UDR:

AUS-Function	**Java-UDR**	**Java-Methode**	**Signaturen**
aus_evaluator	udr_evaluator	ausEvaluate	(boolean)
			(int, int)
aus_refresh_stats	udr_refresh_stats	ausRefresh	()
			(int, int)

Die Java-UDRs können per *dbaccess sysadmin* durch folgende SQL-Anweisungen erstellt werden:

```
create function udr_evaluator(eval_only BOOLEAN)
      returning int
      with (class='jvp')
      external name 'ausudr:aus.ExecAUS.ausEvaluate(boolean)'
      language java;

create function udr_evaluator(task_id INTEGER, task_seq INTEGER)
      returning int
      with (class='jvp')
      external name 'ausudr:aus.ExecAUS.ausEvaluate(int, int)'
      language java;

create function udr_refresh_stats()
      returning int
      with (class='jvp')
      external name 'ausudr:aus.ExecAUS.ausRefresh()'
      language java;

create function udr_refresh_stats(task_id INTEGER, task_seq INTEGER)
      returning int
      with (class='jvp')
      external name 'ausudr:aus.ExecAUS.ausRefresh(int, int)'
      language java;
```

Das Verfahren zum Erstellen von Java-UDRs ist im *J/Foundation Developer's Guide* beschrieben.

- Erstellen der Java-Klasse
  Nach Anpassung der Connection-URL kann die im Anhang beschriebene Java-Quelle mit einem Java-Compiler der Version 1.5 übersetzt werden:

```
javac ExecUDR.java
```

Die Variable *CLASSPATH* muss das Jar-Archiv *$INFORMIXDIR/extend/krakatoa/krakatoa.jar* enthalten.

- Erstellen eines Jar-Archivs
  Die übersetzte Class-Datei ExecUDR.class ist in ein Verzeichnis *aus* zu verschieben.
  Anschließend kann das Jar-Archiv *ausudr.jar* erstellt werden:

  ```
  jar cvf ausudr.jar aus/ExecAUS.class
  ```

- Registrieren des Jar-Archivs in Datenbank sysadmin
  Das Jar-Archiv *ausudr.jar* ist in einen neuen Ordner *$INFORMIXDIR/extend/ausudr* zu verschieben.
  Anschließend kann es durch die folgende SQL-Anweisung per *dbaccess sysadmin* registriert werden:

  ```
  execute procedure sqlj.install_jar(
      'file:$INFORMIXDIR/extend/ausudr/ausudr.jar', 'ausudr');
  ```

- Austausch des Funktionsnamens der AUS-Tasks per dbaccess sysadmin
  Letztlich ist noch dafür zu sorgen, dass der Scheduler die alternativen Funktionen verwendet. Hierzu muss der auszuführende Befehl in der Spalte *ph_task.tk_execute* ausgetauscht werden:

  ```
  update sysadmin:ph_task set
       tk_execute = "udr_evaluator"
       where tk_name = "Auto Update Statistics Evaluation";
  update sysadmin:ph_task set
       tk_execute = "udr_refresh_stats"
       where tk_name = "Auto Update Statistics Refresh";
  ```

- Ändern OpenAdmin PHP-Skript *updstats.php*
  Das OpenAdmin Tool besitzt die Möglichkeit die Statistiken unmittelbar zu aktualisieren. Dies wird über eine Schaltfläche "Refresh Evaluation" auf der Seite „Auto Update Statistics" ausgelöst.

  Die Namen der hierbei aufgerufenen Prozeduren sind im folgenden Verzeichnis anzupassen:
  Apache<Version>/htdocs/openadmin/modules in dem PHP-Skript *updstats.php*

  ```
  $qry = "EXECUTE FUNCTION aus_evaluator('t')";
  --> $qry = "EXECUTE FUNCTION udr_evaluator('t')";

  $qry = "EXECUTE FUNCTION aus_evaluator('f')";
  --> $qry = "EXECUTE FUNCTION udr_evaluator('f')";
  ```

Nach Abschluss der beschrieben Vorgänge ist AUS sowohl im Zusammenwirken mit dem Scheduler als auch mit dem OpenAdmin Tool funktionsfähig.

Das geschilderte Verfahren wurde unter folgender Testumgebung verifiziert:
- Windows XP SP3
- IDS 11.50.TC4
- JDK 1.5

Ihr Ansprechpartner:
Erik Stahlhut | Leiter IBM Distribution | erik.stahlhut@cursor.de | www.cursor-distribution.de

---

Appendix: Demoversion *ExecAus.java*

Der folgende Java-Programmcode liefert eine funktionsfähige Demo-Version zu Testzwecken. Er ist nicht für den Einsatz unter Produktionsbedingungen gedacht. Wichtig: Der Parameter URL der Methode *DriverManager.getConnection(URL)* ist an die reale Umgebung anzupassen.

```
package aus;

import java.sql.*;
import com.informix.udr.*;

public class ExecAUS
{
   static Connection ausConn = null;
   static PreparedStatement evalStmt = null;
   static PreparedStatement refreshStmt = null;

   static void connect()
         throws SQLException
   {
         if (ausConn == null) {

               try {

                     Class.forName("com.informix.jdbc.IfxDriver");

                     ausConn = DriverManager.getConnection(
                                 "jdbc:informix-sqli:" +
                                 "//acer1350:1527/sysadmin:" +
                                 "informixserver=ol_acer1350;" +
                                 "CLIENT_LOCALE=en_us.CP1252;" +
                                 "DB_LOCALE=en_us.CP1252");

               } catch (Exception e) {
                     throw new SQLException(e.toString());
               }

               evalStmt = ausConn.prepareStatement(
                                 "execute function aus_evaluator(?, ?, ?)");
               refreshStmt = ausConn.prepareStatement(
                                 "execute function aus_refresh_stats(?, ?)");
         }
   }

   public static Integer ausEvaluate(boolean evalOnly)
         throws SQLException
   {
         return ausEvaluate(-1, 1, (evalOnly ? 0 : 1));
   }

   public static Integer ausEvaluate(int taskId, int taskSeq)
         throws SQLException
   {
```


```
        return ausEvaluate(taskId, taskSeq, 1);
    }

    public static Integer ausEvaluate(int taskId, int taskSeq, int evalOnly)
        throws SQLException
    {
        Integer res = null;

        connect();

        evalStmt.setInt(1, new Integer(taskId));
        evalStmt.setInt(2, new Integer(taskSeq));
        evalStmt.setInt(3, new Integer(evalOnly));

        ResultSet rs = evalStmt.executeQuery();

        while(rs.next()) {
            res = rs.getInt(1);
        }
        rs.close();

        return res;
    }

    public static Integer ausRefresh()
        throws SQLException
    {
        return ausEvaluate(-1, 1);
    }

    public static Integer ausRefresh(int taskId, int taskSeq)
        throws SQLException
    {
        Integer res = null;

        connect();

        evalStmt.setInt(1, new Integer(taskId));
        evalStmt.setInt(2, new Integer(taskSeq));

        ResultSet rs = evalStmt.executeQuery();

        while(rs.next()) {
            res = rs.getInt(1);
        }
        rs.close();

        return res;
    }

}
```