Aufgabenblatt 5
JDBC

- Abgabetermin: **Dienstag, 21.06.11**
- Zur Prüfungszulassung muss ein Aufgabenblatt mit mind. 25% der Punkte bewertet werden und alle weiteren Aufgabenblätter mit mindestens 50% der Punkte.
- Die Aufgaben sollen **in Zweiergruppen** bearbeitet werden.

- **Abgabe:**
  - per E-Mail an dbs1-2011@hpi.uni-potsdam.de
  - eine Java-Datei mit folgendem Dateinamen und entsprechendem Klassennamen:
    blatt<aufgabenblattNr>aufgabe<aufgabenNr><Nachnamen>.java
    Bitte **keine Leerzeichen, Unterstriche, Umlaute, Sonderzeichen**, . . . im Dateinamen!

Aufgabe 1: Analysewerkzeug für DB2-Tabellenspalten

Ziel der Aufgabe ist es ein Java-Programm zu schreiben, das statistische Informationen für bestimmte Spalten in DB2-Tabellen sammelt und dem Benutzer ausgibt. Hierfür wird für jeden verschiedenen Wert in der Spalte ein Eintrag in einer selbst definierten Analyse-Tabelle erstellt. Neben dem eigentlichen Wert werden dort Informationen bezüglich Länge, Genauigkeit (bei Zahlenwerten), Häufigkeit, ermitteltem Datentyp und weitere Informationen gespeichert. Anschließend können durch SQL-Anfragen auf dieser Tabelle dann Aussagen gemacht werden, wie z.B. häufigste Datentypen, längster Wert oder ob die Spalte ein Schlüsselkandidat ist.

Programmablauf

Der Benutzer soll das Programm mit folgenden Parametern starten können:

- -d <Datenbankname>
- -s <IP-Adresse des Datenbank-Servers> [in der VM mittels *ip addr* ermitteln (Eintrag *eth0*), falls im Gastsystem entwickelt wird]
- -p <Port der Datenbank-Instanz> [in der VM *50001*]
- -u <Username> [in der VM *db2inst1*]
- -pw <Passwort> [in der VM *pefahe22*]
- -t <Tabellenname>
- -c <Spaltenname>

Anschließend soll das Programm die Werte des Attributs lesen und die Metainformationen in der Analysetabelle speichern. Diese soll folgendes Schema und Namen haben:

<Tabellenname>_<Spaltenname>(<u>value</u>, datatype, length, precision, scale, generalFormat, count)

Die vorgegebenen <Datentypen> der Spalten und ihre *Bedeutung* im einzelnen:

- value <varchar(255)> *einer der verschiedenen Werte der Spalte*
- datatype <varchar(10)> *ermittelter Datentyp*

- length <int> *Länge des Wertes*
- precision <int> *Präzision des Wertes bei Zahlenwerten (sonst 0)*
- scale <int> *Nachkommastellen, falls Gleitkomma-Datentyp*
- generalFormat *normalisierte Repräsentation [aus "059.aBCd+" wird z.B. "999.aAAa+"]*
- count <int> *absolute Häufigkeit des Wertes*

Die Informationen, die in der Analysetabelle gespeichert werden sollen, können mittels der Hilfsklasse *DistinctValue* bestimmt werden. Hierfür muss lediglich ein Objekt der Klasse für jeden verschiedenen Wert erzeugt werden:

```
DistinctValue dv = new DistinctValue(Wert, Häufigkeit);
```

Anschließend kann über *Getter*-Methoden auf jedes Feld dieses Objekts zugegriffen werden, um die Metadaten in der Datenbank zu speichern.

Nachdem alle Statistikinformationen gesammelt und gespeichert in der Datenbank vorliegen, können die eigentlichen Untersuchungen auf der Statistiktabelle durchgeführt werden. Hierfür sollen per JDBC SQL-Anfragen auf der Analysetabelle ausgeführt und anschließend deren Ergebnis ausgegeben werden. Zum Beispiel könnte der häufigste Wert in der Spalte mittels folgendem Query ermittelt werden:

```
select value
  from <Tabellenname>_<Spaltennname>
  order by count desc
  fetch first 1 row only;
```

Die zu implementierenden Funktionen sollen folgende Fragen beantworten:

- Welches ist die größte Länge innerhalb der Spalte?
- Welches sind die 3 häufigsten Datentypen in der Spalte?
- Ist die Spalte prinzipiell ein Schlüssel? (also: Ist die Spalte Unique?)
- Welches ist der empfohlene Datentyp, um alle Werte speichern zu können? Hinweis: Falls beispielsweise nur die Datentypen *INT16* und *INT32* gefunden werden, ist *INT32* der richtige Datentyp. Befindet sich allerdings zusätzlich ein Wert mit einem Gleitkomma-Datentypen in der Spalte, so ist dieser *FLOAT*-Typ auszugeben. Falls zusätzlich normale Zeichenketten vorkommen, so ist *STRING* der richtige Datentyp für die Spalte.

Programmaufruf und Ausgabeformat

Der Aufruf des Programms sollte folgendermaßen aussehen:

```
db2inst1@db2expc:~> java blatt5aufgabe1Nachnamen -d db2db1 -s 127.0.0.1
                    -p 50001 -u db2inst1 -pw pefahe22 -t actor -c name
```

Die Ausgabe des Programms sollte folgendermaßen aussehen (Hinweis: Werte entsprechen nicht der Ausgabe auf der vorgegebenen Instanz):

```
Maximale Länge des Attributs name in Tabelle actor:
  12
Die drei häufigsten Datentypen des Attributs name in Tabelle actor:
  INT32
  INT8
  DFLOAT
Der empfohlene Datentyp des Attributs name in Tabelle actor:
  STRING
Attribut name in Tabelle actor ist kein Schlüsselkandidat.
```

Es sollen grundsätzlich immer alle vier Fragen beantwortet werden.

Allgemeine Hinweise zur Entwicklung

Grundsätzlich wird empfohlen, außerhalb der VM auf den Poolrechnern mit Eclipse zu entwickeln und von dort auf die laufende VM zuzugreifen. Die individuelle IP-Adresse der VM lässt sich auf der Konsole mit dem Befehl *ip addr* ermitteln, wessen Ausgabe folgendermaßen aussieht:

```
db2inst1@db2expc:~> ip addr
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,NOTRAILERS,UP> mtu 1500 qdisc pfifo_fast ...
    inet 192.168.163.135/24 brd 192.168.163.255 scope global ...
    inet6 fe80::20c:29ff:fe80:9c99/64 scope link
       valid_lft forever preferred_lft forever
3: sit0: <NOARP> mtu 1480 qdisc noop
    link/sit 0.0.0.0 brd 0.0.0.0
```

Die fett gedruckte IP-Adresse ist diejenige unter welcher die VM erreichbar ist.

Der Standard-Port der DB2-Instanz lautet *50001*.

Zum Arbeiten wird weiterhin der *DB2-JDBC*-Treiber benötigt, welcher zusammen mit der bereitgestellten Hilfsklasse *DistinctValue*.java im Lehrveranstaltungs-Share bereitsteht.