Brinnich Selina & Kopec Jakub

Rückwärtssalto

Protokoll 17.02.2015

Inhalt

1 Aufgabenstellung[1]

Erstelle ein Java-Programm, das Connection-Parameter und einen Datenbanknamen auf der Kommandozeile entgegennimmt und die Struktur der Datenbank als EER-Diagramm und Relationenmodell ausgibt (in Dateien geeigneten Formats, also z.B. PNG für das EER und TXT für das RM).

Verwende dazu u.A. das ResultSetMetaData-Interface, das Methoden zur Bestimmung von Metadaten zur Verfügung stellt.

Zum Zeichnen des EER-Diagramms kann eine beliebige Technik eingesetzt werden für die Java-Bibliotheken zur Verfügung stehen: Swing, HTML5, eine WebAPI, ... Externe Programme dürfen nur soweit verwendet werden, als sich diese plattformunabhängig auf gleiche Weise ohne Aufwand (sowohl technisch als auch lizenzrechtlich!) einfach nutzen lassen. (also z.B. ein Visio-File generieren ist nicht ok, SVG ist ok, da für alle Plattformen geeignete Werkzeuge zur Verfügung stehen)

Recherchiere dafür im Internet nach geeigneten Werkzeugen.

Die Extraktion der Metadaten aus der DB muss mit Java und JDBC erfolgen.

Im EER müssen zumindest vorhanden sein:

- korrekte Syntax nach Chen, MinMax oder IDEFIX
- alle Tabellen der Datenbank als Entitäten
- alle Datenfelder der Tabellen als Attribute
- Primärschlüssel der Datenbanken entsprechend gekennzeichnet
- Beziehungen zwischen den Tabellen inklusive Kardinalitäten soweit durch Fremdschlüssel nachvollziehbar. Sind mehrere Interpretationen möglich, so ist nur ein (beliebiger) Fall umzusetzen: 1:n, 1:n schwach, 1:1
- Kardinalitäten

Fortgeschritten (auch einzelne Punkte davon für Bonuspunkte umsetzbar):

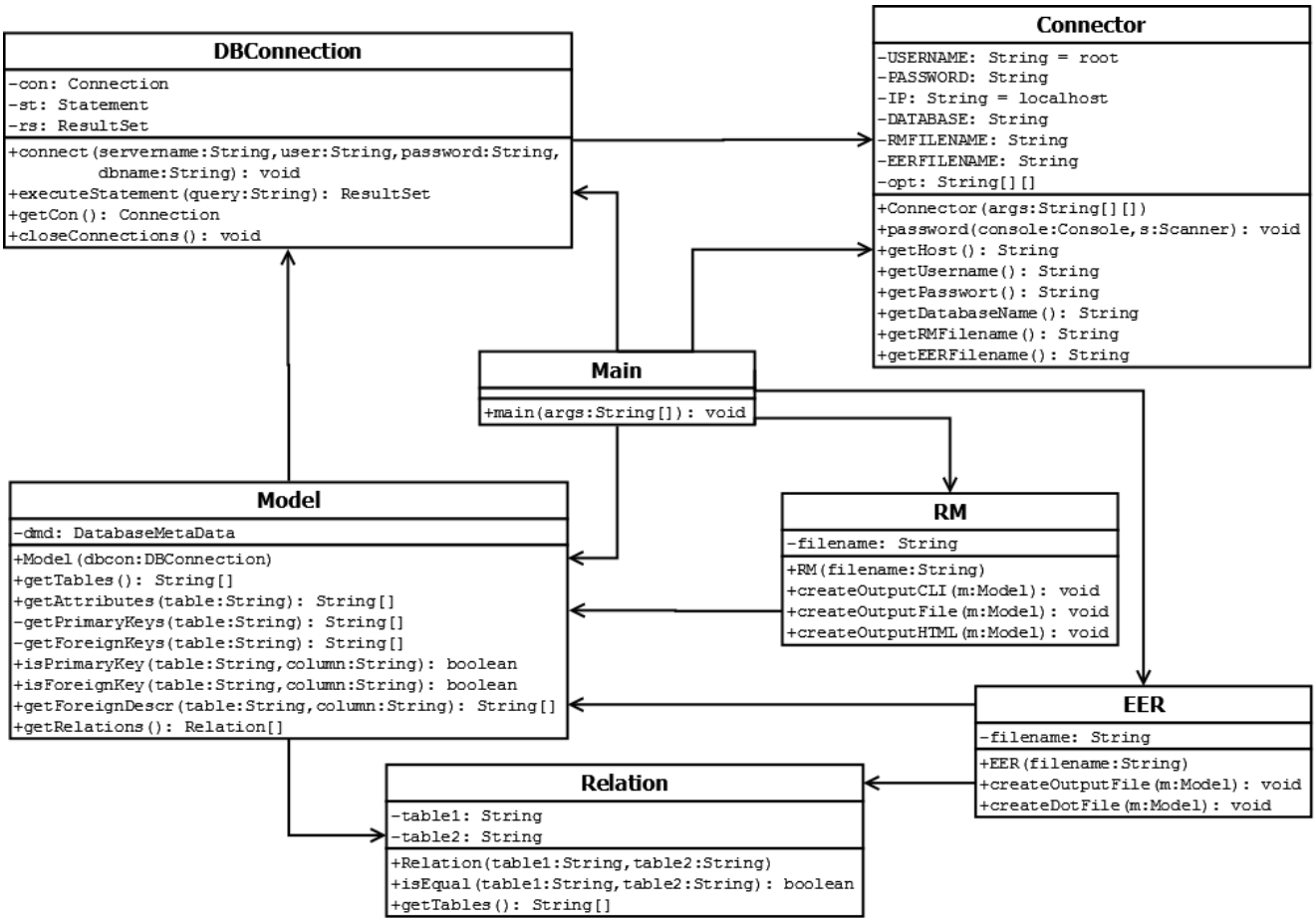
- Zusatzattribute wie UNIQUE oder NOT NULL werden beim Attributnamen dazugeschrieben, sofern diese nicht schon durch eine andere Darstellung ableitbar sind (1:1 resultiert ja in einem UNIQUE)
- optimierte Beziehungen z.B. zwei schwache Beziehungen zu einer m:n zusammenfassen (ev. mit Attributen)
- Erkennung von Sub/Supertyp-Beziehungen

2 Designüberlegung

2.1 Arbeitspakete

Requirements		Implementiert	Getestet
Allgemein	Eingabeparameter verarbeiten	✔	✔
	Connection zur DB herstellen	✔	✔
	Fehlerfälle richtig bearbeiten	✔	✔
RM	Tabellenname, Spaltenname auslesen	✔	✔
	Primärschlüssel und Fremdschlüssel feststellen und kennzeichnen	✔	✔
	In ein Textfile schreiben (txt & html)	✔	✔
EER	Zeichen-Tool evaluieren und einbinden	✔	✔
	Entitäten feststellen und zeichnen	✔	✔
	Attribute feststellen und zeichnen	✔	✔
	Beziehungen feststellen und zeichnen	✔	✔
	Kardinalitäten von Beziehungen feststellen und zeichnen		

2.2 UML - Diagramm



3 Arbeitsaufteilung & Aufwandsabschätzung

Requirements		Geschätzt	Benötigt
Allgemein	Eingabeparameter verarbeiten	15 min	5 min
	Connection zur DB herstellen	10 min	2 min
	Fehlerfälle richtig bearbeiten	30 min	20 min
RM	Tabellenname, Spaltenname auslesen	15 min	10 min
	Primärschlüssel und Fremdschlüssel feststellen und kennzeichnen	1 h	30 min
	In ein Textfile schreiben (txt & html)	30 min	10 min
EER	Zeichen-Tool evaluieren und einbinden	4 h	2 h
	Entitäten feststellen und zeichnen	30 min	20 min
	Attribute feststellen und zeichnen	1 h	40 min
	Beziehungen feststellen und zeichnen	1 h	1 h
	Kardinalitäten von Beziehungen feststellen und zeichnen	3 h	

4 Arbeitsdurchführung[2]

4.1 Relationenmodell

Die Connection zur Datenbank, sowie die Verarbeitung der Eingabeparameter wurden von der vorherigen Aufgabe "The Exporter" übernommen und entsprechend der Anforderungen dieser Aufgabenstellung angepasst.

Das Auslesen der Tabellen und Spalten wurde mithilfe des Interfaces DatabaseMetaData umgesetzt.

DatabaseMetaData stellt Methoden wie `getTables()` und `getColumns()` bereit, die jeweils ein ResultSet zurück liefern, aus dem man die entsprechenden Meta-Daten auslesen kann (mittels `getString([Attribute_Name])`; "TABLE_NAME" zum Auslesen des Tabellennamens, "COLUMN_NAME" zum Auslesen des Spaltennamens)

```
try {
    ResultSet t = dmd.getTables(null, null, null, null);
    while(t.next()){
        tables.add(t.getString("TABLE_NAME"));
    }
} catch (SQLException e) {
    System.out.println(e.getMessage());
}
```

```
try {
    ResultSet t = dmd.getColumns(null, null, table, null);
    while(t.next()){
        columns.add(t.getString("COLUMN_NAME"));
    }
} catch (SQLException e) {
    System.out.println(e.getMessage());
}
```

Die Primärschlüssel wurden mit der Methode `getPrimaryKeys()` ausgelesen. Dabei wird ebenfalls ein ResultSet zurückgegeben.

```
try {
    ResultSet t = dmd.getPrimaryKeys(null, null, table);
    while(t.next()){
        columns.add(t.getString("COLUMN_NAME"));
    }
} catch (SQLException e) {
    System.out.println(e.getMessage());
}
```

Die Fremdschlüssel wurden mit der Methode `getImportedKeys()` ausgelesen.

```
try {
    ResultSet t = dmd.getImportedKeys(null, null, table);
    while(t.next()){
        columns.add(t.getString("FKCOLUMN_NAME"));
    }
} catch (SQLException e) {
    System.out.println(e.getMessage());
}
```

Die Informationen über die Referenz des Fremdschlüssels werden mit derselben Methode ausgelesen. Statt "FKCOLUMN_NAME" wird allerdings nun "PKTABLE_NAME" für den Tabellennamen der Referenztabelle und "PKCOLUMN_NAME" für den Spaltennamen der Referenzspalte.

```
try {
    ResultSet t = dmd.getImportedKeys(null, null, table);
    while(t.next()){
        if(t.getString("FKCOLUMN_NAME").equals(column)){
            s[0] = t.getString("PKTABLE_NAME");
            s[1] = t.getString("PKCOLUMN_NAME");
            return s;
        }
    }
} catch (SQLException e) {
    System.out.println(e.getMessage());
}
```

Die Ausgabe des RM wird sowohl in Console, als auch in einem .txt & .html File implementiert. Dazu werden zunächst alle Tabellen einzeln durchgegangen und in jeder Tabelle nochmal alle Spalten aufgeschrieben. Pro Spalte wird überprüft, ob sie ein Primärschlüssel und/oder Fremdschlüssel ist. Bei Primärschlüssel wird bei .txt bzw. in der Konsole <PK> davor geschrieben und bei .html wird der Text mit <u> unterstrichen. Bei Fremdschlüssel wird bei .txt bzw. in der Konsole <FK> davor geschrieben und bei .html wird der Text mit <i> kursiv geschrieben.

4.2 EER

Zum Zeichnen des EER-Diagramms musste zunächst ein passendes Zeichentool evaluiert werden. Hierbei haben wir uns für Graphviz[3] entschieden, da es zur verwendeten Sprache *dot* ausreichend Dokumentation gibt und die Sprache alles zur Verfügung stellt, das zum Zeichnen des EER-Diagrammes benötigt wird. Außerdem ist Graphviz plattformunabhängig, so wie es der Aufgabenstellung entspricht.

Als Erstes musste nun das Tool installiert werden. Unter Windows-Betriebssystemen muss dabei der bin-Ordner von Graphviz zusätzlich in die Path-Variable eingetragen werden.

4.2.1 dot-File erstellen[4]

Um ein .dot File zu erstellen, muss zunächst ein neuer Graph definiert werden. Das erreicht man mit der Syntax *graph [name]{ … }*.

Innerhalb der Klammern können nun in dot-Syntax Objekte und Verbindungen zwischen diesen Objekten erstellt werden. Ein neues Objekt benötigt lediglich einen Namen (mit Semikolon als Zeilenende). Da wir für das EER-Diagramm Rechtecke anstatt Ellipsen benötigen, schreiben wir nach dem Namen noch *[shape=box]*.

Die einzelnen Attribute der Tabellen werden ebenfalls mit Namen angeschrieben, jedoch mussten wir hier als Namen [tabellenname][attributname] nehmen und den Text als *[label=[attributname]]* definieren, da sonst alle Attribute, die gleich heißen (z.B. mehrere id's in unterschiedlichen Tabellen), nur ein einziges Objekt verwenden. Für Primärschlüssel wird beim Label HTML-Syntax angewandt: *[label=<<u>[attributname]</u>>]*. Um die Attribute mit den Entitäten zu verbinden, wird *[tabellenname]--[tabellenname][attributname]* verwendet. Das -- definiert eine Linie zwischen den beiden Objekten. Für Pfeile wird -> verwendet.

Beziehungen benötigen ebenfalls Objekte. Diese werden bei uns *bez[nr.]* genannt, wobei *nr.* eine fortlaufende Nummer ist. Als Shape wird eine Raute benötigt und in der Raute soll kein Text stehen: *[shape=diamond label=""]*. Nun kann diese Beziehung folgendermaßen verwendet werden: *[tabellenname1]--[bez][nr.]--[tabellenname2]*.

4.2.2 dot zu png

Ein fertig erstelltes .dot File kann mit folgendem Befehl in Java zu einem .png umgewandelt werden, sofern Graphviz bereits installiert wurde:

```
try {
    Process proc = Runtime.getRuntime().exec("neato -Tpng " + filename + ".dot -o " + filename + ".png");
} catch (IOException e) {
    e.printStackTrace();
}
```

5 Quellenangaben

[1] Erhard List, "A05 – RÜCKWÄRTSSALTO", https://elearning.tgm.ac.at/mod/assign/view.php?id=31205, last seen: 21.01.2015
[2] Oracle, "Java Doc – Interface DatabaseMetaData", http://docs.oracle.com/javase/7/docs/api/java/sql/DatabaseMetaData.html, last seen: 19.01.2015
[3] Graphviz, "Graphviz", http://www.graphviz.org/, last seen: 11.02.2015
[4] Tony Ballantyne, "Drawing Graphs using Dot and Graphviz", http://www.tonyballantyne.com/graphs.html, last seen: 21.01.2015