Rückwärtssalto

Erstellen eines EER-Diagramm und RM einer RDBMS Datenbank mittels JDBC

08.01.2014

4BHIT
Nenad Gligorevic, Melanie Göbel

Inhaltsverzeichnis

Aufgabenstellung

Erstelle ein Java-Programm, dass 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

Aufwand und Zeit

Die Aufgabe scheint nicht sehr einfach zu sein aber machbar. Wichtig ist ein Tool zu erstellen von den EER-Diagrammen zu finden, die Verbindung mit der Datenbank via JDBC scheint einfacher zu sein

Schätzung und Realität

Arbeit	Geschätzte Zeit (in min)	Zuständig	Benötigte Zeit (in min)
Verbindung zur Datenbank	80	Melanie	30
Struktur zum Speichern der Daten	100	Nenad	120
Auslesen und Erstellen des RM	120	Melanie	100
Internetrecherche zu EER-Diagramm Tool	50	Melanie	30
Erstellen eines EER-Diagramm	200	Nenad	220
Beziehungen und Kardinalitäten im EERD	160	Nenad	120
Dokumentation	100	Melanie	100
Zusätzliche Arbeit (zB. Bug fixinig)	0	Melanie, Nenad	60
Insgesamt	13 Stunden 30 Min		13 Stunden

Arbeitsaufzeichung

Melanie

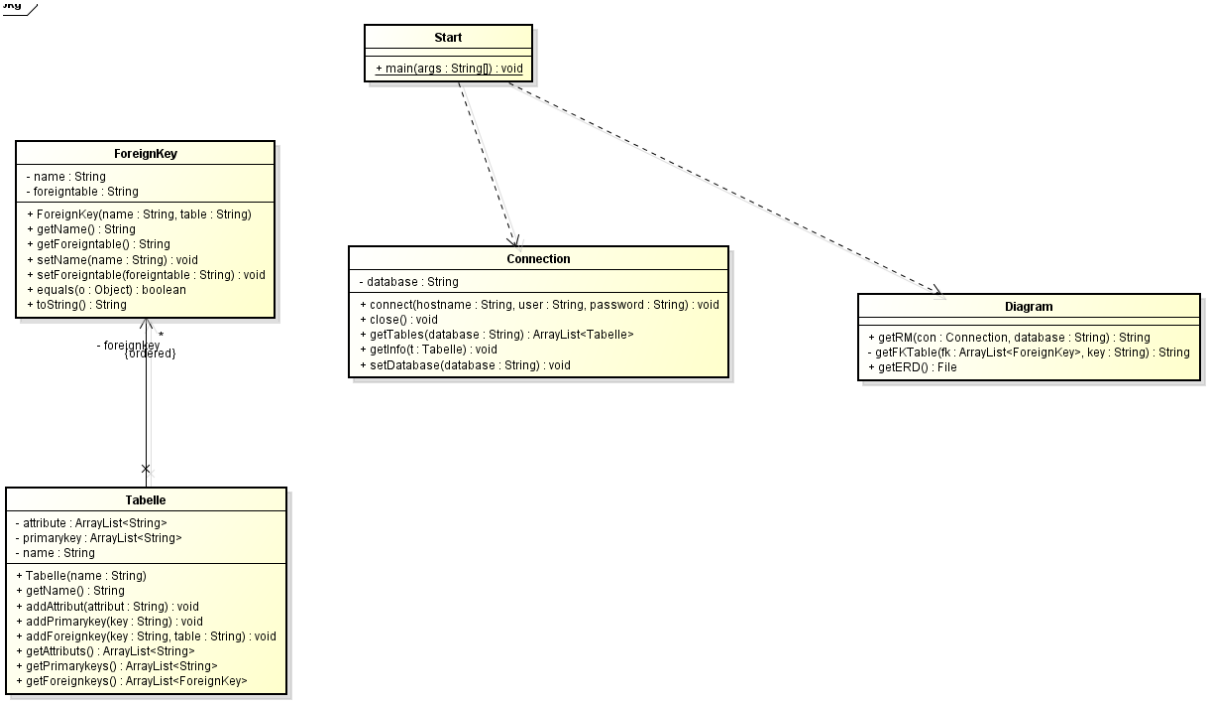
Arbeit	Datum, Uhrzeit	Benötigte Zeit in min
Verbindung zur Datenbank	14.1.2015 9:50-11:30	100
Auslesen und Erstellen des RM	20.1.2015 17:00-18:00	60
Designänderung (Zusätzliche Arbeit)	30.1.2015 22:00 – 22:40	40
Designänderung, RM	06.1.2015 23:00-00:00	60
ERD	07.02.2015 13:00-14:20	80
ERD	10.02.2015 18:00-20:20	140
Beziehungen EERD	17.02.2015 19:00-21:00	120

Nenad

Arbeit	Datum, Uhrzeit	Benötigte Zeit
Struktur zum Speichern der Daten	14.1.2015 9:50-11:30	100

Designüberlegung

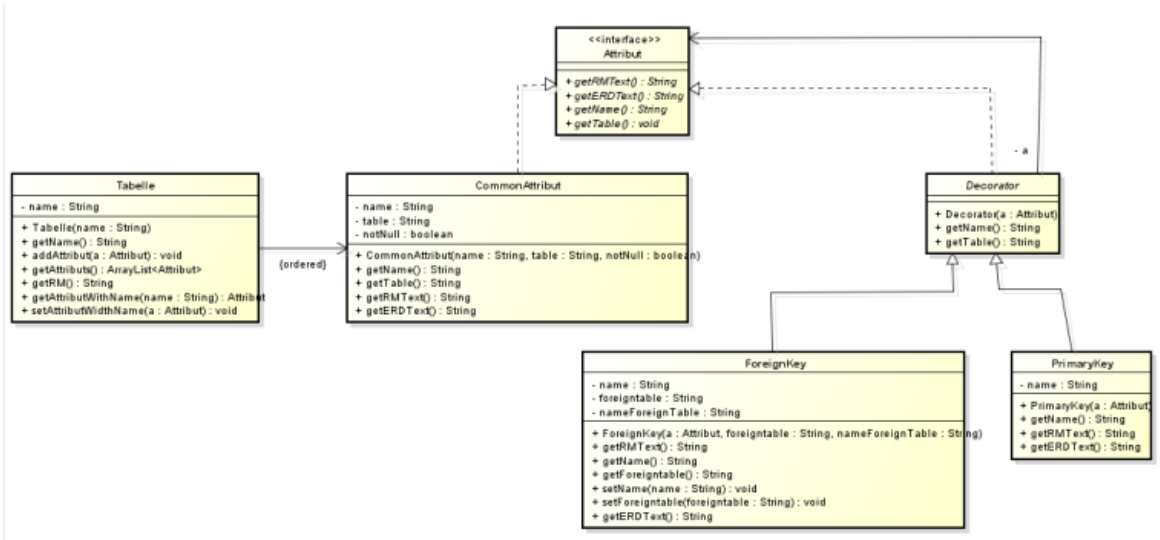
Erste Überlegung war, das Speichern der Information in die Klasse Tabelle auszulagern sowie die Verbindung zur Datenbank und auslesen der Datenbank in Connection. Diagram ist eine Klasse die sowohl ein Relationenmodel sowie ein Entity-Relationship-Diagram erstellen kann. In Start findet das Argumentparsing (Parameterüberprüfung) statt.



Probleme bei dem vorläufigen Design:

- Änderungen wie zB zusätzliche Infos bei einen Attribut in einer Tabelle (Datentyp) ist schwer zu implementieren (Änderungen am Code)
- Verwendung von mehreren Listen in Tabelle (kann zu doppelten Attributen führen)

Um Änderungen willkommen zu heißen, wird das Design-Pattern Decorator-Pattern angewendet. Dies hilft zur Abschaffung von mehreren Listen in Tabelle.



Arbeitsdurchführung

Connection

Erstellen der Verbindung zur RDBMS mit JDBC.

```
ds = new MysqlDataSource();
             ds.setServerName(hostname);
             ds.setUser(user);
             ds.setPassword(password);
             this.con = ds.getConnection();
```

Auslesen der Daten: Tabelle und deren Information (Attribute, PrimaryKeys, ForeignKeys und Tabellen von denen die Foreignkeys sind)

```
st = con.createStatement();
ResultSet rs1 = st.executeQuery("use "+database+";");
ResultSet rs = st.executeQuery("show tables;");
while(rs.next()){
      tables.add(new Tabelle(rs.getString(1)));
}
```

Auslesen der Attribute und PrimaryKeys mit Desc:

```
ResultSet rs = st.executeQuery("desc "+t.getName()+";");
while(rs.next()){
      t.addAttribut(rs.getString(1));
      if(rs.getString(4).equals("PRI"))
            t.addPrimarykey(rs.getString(1));
      }
```

Auslesen der ForeignKeys mit DatabaseMetaData:

```
DatabaseMetaData meta = con.getMetaData();
ResultSet rsK = meta.getImportedKeys(database, null, t.getName());

while(rsK.next()){
      t.addForeignkey(rsK.getString("FKCOLUMN_NAME"),rsK.getString("PKTABLE_NAME")
);

}
```

Diagram

Das RM wird mit `public String getRM(Connection con, String database){}` erstellt.

Das ERD wird mit `public boolean getDotFile(Connection con, String database, String filename){` und `public boolean Drawpng(String dotfile, String file){` erstellt.

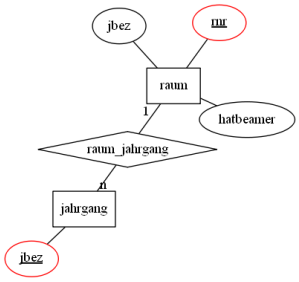
Jedes Attribut hat eine Methode getRMText() sowie getERDText(). Diese geben einen Text mit RM-Syntax oder Dot-Syntax mit ihren Daten zurück. Für das ERD braucht man ebenso Relationships, diese beinhalten ebenfalls eine getERDText() Methode.

Testbericht

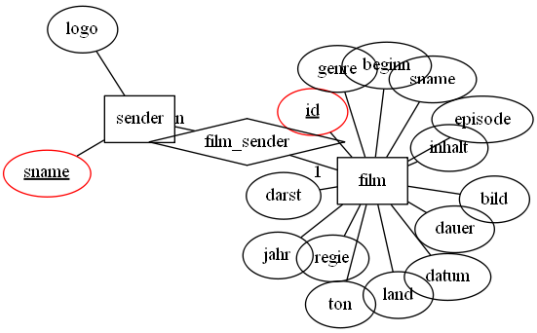
ERD

Ein ERD kann problemlos gezeichnet werden, wenn nicht viele Attribute pro Entität vorhanden sind. (Die Primarykeys sind nur rot gekennzeichnet damit man sie besser findet).

Hier sieht man ein ERD, das keine Makel hat:

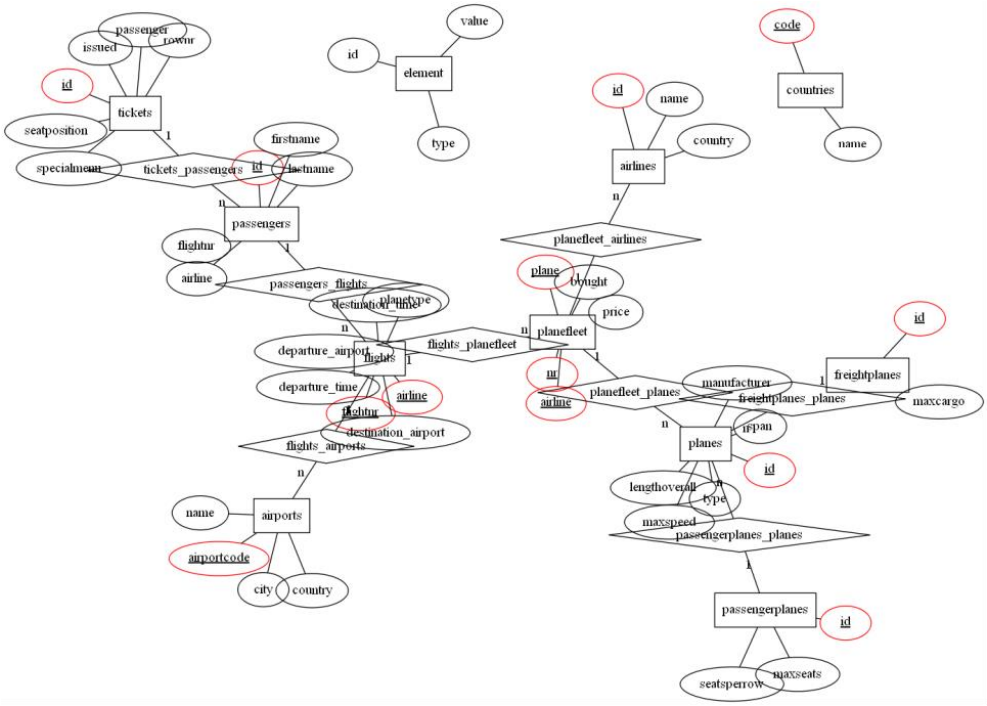


Wenn jedoch eine Entität viele Attribute besitzt passiert folgendes:



Attribute überlappen sich und sind schwer lesbar.

Das Testbeispiel hatte auch ein paar Probleme mit der Überlappung:



RM

Ebenso kann man hier dieselben 3 Beispiele wie beim Testbericht für das ERD.
Keine Probleme bei einer kleinen Datenbank:

jahrgang(jbez<PK>)
raum(rnr<PK>,hatbeamer,jahrgang.jbez:jbez)

Ebenso nicht bei dieser Datenbank mit einer Tabelle mit vielen Attributen:

film(id<PK>,datum,beginn,episode,genre,dauer,land,jahr,regie,bild,ton,darst,inhalt,
sender.sname:sname)
sender(sname<PK>,logo)

Auch die große Test-Datenbank weist auf keine Probleme hin, ebenso kann man hier gut die Namensänderung eines Fremdschlüssel beobachten:

airlines(id<PK>,name,country)
airports(airportcode<PK>,name,country,city)
countries(code<PK>,name)
element(id,type,value)
flights(
planefleet.airline:airline<PK>,flightnr<PK>,departure_time,destination_time,planefleet.plane:planetype,airport
s.airportcode:departure_airport,airports.airportcode:destination_airport)
freightplanes(planes.id:id<PK>,maxcargo)
passengerplanes(planes.id:id<PK>,maxseats,seatsperrow)
passengers(id<PK>,firstname,lastname,flights.flightnr:flightnr,flights.airline:airline)
planefleet(airlines.id:airline<PK>,planes.id:plane<PK>,nr<PK>,bought,price)
planes(id<PK>,manufacturer,type,lengthoverall,span,maxspeed)
tickets(id<PK>,issued,rownr,seatposition,specialmenu,passengers.id:passenger)

Fazit

Das RM funktioniert ohne Probleme und das ERD mit kleinen Einschränkungen.

Relationenmodell

Funktion	Status
Entitäten + Attribute anzeigen	✓
Primarykey mit <PK> gekennzeichnet	✓
Foreignkey: tabelle.attribut: attributname	✓
Primarykey + Foreignkey gemeinsam	✓

Entity-Relationship-Diagram

Funktion	Status
Entität + Attribute anzeigen	✓
Primarykey unterstrichen	✓
Speichern von notNull	✓
Speichern von Unique	✗
Beziehungen anzeigen	✓
Alles gut sichtbar	✗