JDBC - RÜCKWÄRTSSALTO

INSY / A05

11. MÄRZ 2015

ERCEG, KRITZL
4AHITT

Inhalt

Github-Link: https://github.com/sgeyer-tgm/INSY05

Github-Tag: erceg_geyer_insy05_v1

1. Aufgabenstellung

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. Requirementanalyse

Arbeitspaket	Zuständige Person	Geschätzte Zeit	Erledigt
Library für das RM-Modell suchen	Erceg, Geyer	60 min	x
Library für das ERM-Modell suchen	Erceg, Geyer	60 min	x
Eingabe über Konsole	Erceg	240 min	x
Metadaten verarbeiten	Geyer	180 min	x
RM generieren	Erceg, Geyer	150 min	x
ERD generieren	Erceg, Geyer	200 min	x

3. detaillierte Arbeitsaufteilung mit Aufwandsabschätzung

Teilaufgabe	benötigte Gesamtzeit
UML-Diagramm erstellen	120 Minuten (2 Stunden)
Implementierung des Programms inkl. JavaDoc	720 Minuten (12 Stunden)
Testen des Programms	240 Minuten (4 Stunden)
Protokoll schreiben	120 Minuten (2 Stunden)
Gesamt	**1200 Minuten (20 Stunden)**

4. anschließende Endzeitaufteilung

4.1 Erceg

Arbeit	Datum	Zeit in Minuten
Diagramm erstellen	29.01.2015	80 Minuten
Libraries suchen	20.01.2015	120 Minuten
Diagramm fertiggestellt	21.01.2015	20 Minuten
Implementierung	21.01.2015	80 Minuten
Implementierung	29.01.2015	120 Minuten
Implementierung	02.02.2015	60 Minuten
Protokoll schreiben	15.02.2015	120 Minuten
Protokoll fertiggestellt	11.03.2015	120 Minuten
Gesamt	*11.03.2015*	**720 Minuten (12 h)**

4.2 Geyer

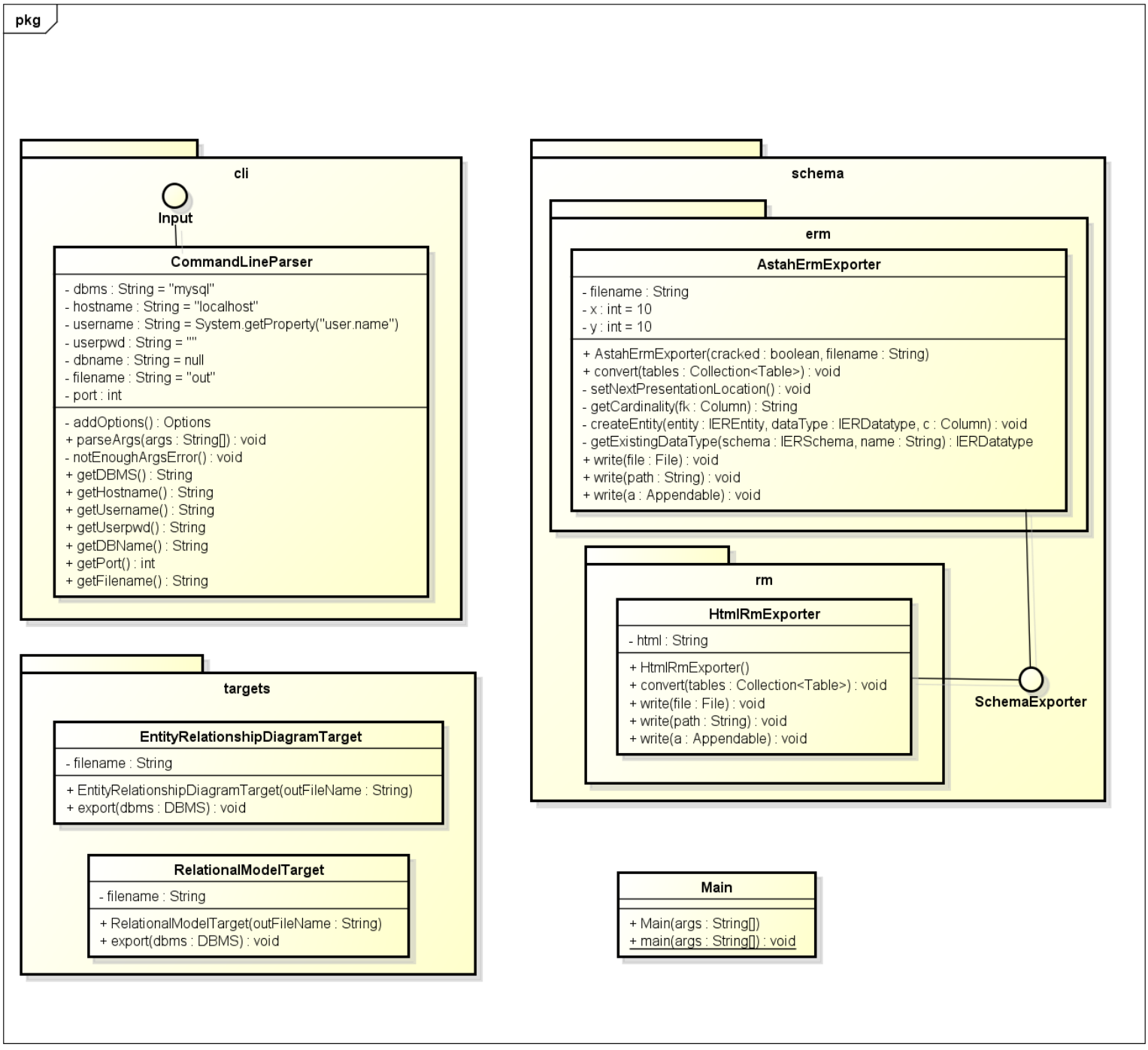
Arbeit	Datum	Zeit in Minuten
Libraries suchen	20.01.2015	180 Minuten
Implementierung	21.01.2015	100 Minuten
Implementierung	21.01.2015	80 Minuten
Implementierung	29.01.2015	150 Minuten
Implementierung	02.02.2015	120 Minuten
Protokoll schreiben	15.02.2015	60 Minuten
Protokoll fertiggestellt	11.03.2015	120 Minuten
Gesamt	*11.03.2015*	**810 Minuten (13 h 30 min)**

4.3 Gesamtsumme

Insgesamt haben wir für diese Übung **25 Stunden und 30 Minuten** benötigt, geschätzt wurden 20 Stunden. Das Testen des Programms, wofür 4 Stunden eingeschätzt worden sind, wurde nicht durchgeführt, daher lagen wir mit unserer Einschätzung (insg. 9 Stunden und 30 Minuten über der geschätzten Zeit) ein wenig daneben, jedoch wurde die Erstellung des Diagramms und die Implementierung des Programms von der Zeit her sehr genau eingeschätzt.

5. Designüberlegung

5.1 Abbildung



5.2 Überlegungen zur Struktur

Das UML-Diagramm wurde mit dem Programm „Astah“ erstellt.
Dazu wurde die mit dem Programm mitgelieferte API (astah-pro) verwendet.
Um das ERD generieren zu können wird die Vollversion des Programms benötigt. Es wurde eine Möglichkeit implementiert, um die Testversion von Astah zurückzusetzen. Dabei wird der .astah Orner im Home Verzeichnis des Users und somit auch die Informationen über den Ablauf der Lizenz gelöscht.

Um das Programm zu verwenden wurde der Code der Aufgabe AU04 „Exporter“ eingebunden (Code von Stefan Geyer, [1]).
Dieser übernimmt jegliche Anbindung an die Datenbank und kann sehr leicht erweitert werden. Für den Rückwärtssalto muss lediglich ein neues „Target“ erstellt und dieses am Exporter registriert werden. Der Exporter liefert den Targets ein Java SQL Connection Objekt zurück, welches vom Target weiter bearbeitet werden kann.
In der derzeitigen Version des Exporters werden nur MySQL und PostgreSQL unterstützt.

Die vom Exporter erhaltenen Daten werden über die SchemaCrawler Library aufgefasst und deren Metadaten ausgelesen. Diese können nun über POJOs abgerufen werden.

Für das Exportieren des RM-Diagramms wurde das HTML Format gewählt. Die Tabellen werden in einem div-Container verstaut und je nach Eigenschaften (PK oder FK) unterstrichen oder kursiv dargestellt.

Beim ERD werden die vom SchemaCrawler aufgefassten Daten einfach über die Astah API an ein Dokument gebunden. Datentypen werden auch übernommen. Das Diagramm wird in IDEF1X dargestellt. Diese Funktion ist „hardcoded“, kann aber selbstverständlich über Astah angepasst werden.

Falls Dateien mit dem selben Namen existieren, werden diese überschrieben.

6. Arbeitsdurchführung

6.1 Suchen nach Libraries

Es wurde eine nach einer Möglichkeit gesucht das Schema eines DBMS auszulesen, um Zeit und Aufwand zu sparen.
Dabei erschien das Tool „SchemaCrawler“ [2] als sehr hilfreich.
Der SchemaCrawler ist ein eigenständiges Java Programm, welches das Schema einer Datenbank ausgibt und diese Informationen weiterverarbeitet. So kann es die Daten z.B. als ERD im DOT Format ausgeben. Diese Funktion wurde nicht verwendet, da in unseren Augen ein wichtiger Punkt dieser Arbeit ein weiterverwendbares ERD ist. Astah bietet sich hierbei wesentlich besser an, da ein generiertes Astah File leicht vom User bearbeitet werden kann.
Der SchemaCrawler bietet unter anderem auch eine Java Library, welche die Daten auch ohne externem Programm verarbeiten kann.
Diese Library wurde verwendet, um die Metadaten in POJOs zu laden.

Da die Astah Library ziemlich ungenau dokumentiert ist und es wenige Examples gibt, hat das „Codedigging“ sehr lange gedauert und war sehr mühsam.

6.2 Erstellen eines Maven Projekts

Ein Maven Projekt wurde erstellt und die nötigsten Plugins wurden konfiguriert.
Um die Astah Library JAR ins lokale Maven Repository hinzuzufügen, wurde der folgende Befehl verwendet:

```
mvn install:install-file -Dfile=<Verzeichis_der_JAR> -DgroupId=jp.change-
vision.astah -DartifactId=astah-pro -Dversion=1.0 -Dpackaging=jar
```

Die Dependencies der Astah JAR und des Exporters wurden zum Projekt hinzugefügt.

6.3 Targets ausprogrammieren und beim Exporter registrieren

Ein ERD und ein RM Target wurden erstellt und zum Exporter hinzugefügt.

6.4 Verarbeiten der Programmargumente

Die Programmargumente werden über die Apache CLI Library geparst und dem Programm zur Verfügung gestellt.

7. Testbericht

Da leider keine Tests geschrieben worden sind, ist auch kein Testbericht vorhanden.

8. Lessons learned

- schnelle und effiziente Suche nach einem passenden Tool
- trotz schlecht dokumentierten APIs und keinen vorhandenen Examples einen Weg finden, das „Codedigging“ schneller auszuführen
- Erweiterung einer bestimmten Übung bzw. eines bestimmten Codes (in diesem Fall: Exporter-Aufgabe von Stefan Geyer)

9. Quellenangaben

[1] Stefan Geyer (2014). Privates Github-Repository „INSY” [Online]. Available at: https://github.com/sgeyer-tgm/INSY/tree/insys04/ [zuletzt abgerufen am 11.03.2015]

[2] Sualeh Fatehi (2003, 2015). SchemaCrawler [Online]. Available at: http://schemacrawler.sourceforge.net/ [zuletzt abgerufen am 11.03.2015]