Universität Salzburg | Übungsleiter: Kwitt/Hütter/Resmerita

Datenbanken 1 – PS (501.073)

Projektabgabe – Teil 1 | Abzugeben bis **13.04.2018 (22:00)**

Im Folgenden finden Sie die Anwendungsbeschreibung in Deutsch als auch in Englisch. Der erste Teil des Projektes besteht darin, diese Anwendungsbeschreibung in Anforderungen an die Datenbank zu übersetzen (dies ist die sogenannte *Requirement Analyse*).

Anwendungsbeschreibung

Unsere Zielanwendung ist ein *soziales Netzwerk*. Den Personen/Teilnehmern in dem sozialen Netzwerk sollte es ermöglicht werden, kurze *Nachrichten* in Form von Texten zu veröffentlichen und, falls gewünscht, diese mit Multimedia Inhalt(en) zu versehen. Die Nachrichten sind immer öffentlich. Weiters sollte es möglich sein sich Kurznachrichten zu einem Thema oder einer Person anzeigen zu lassen, bzw. in diesen Kurznachrichten nach gewünschten Inhalten zu suchen. Personen/Teilnehmer sollten die Möglichkeit haben, Nachrichten zu kommentieren.

Application Description

Our target application is a *social network*. It should allow people to communicate what they find interesting by publishing short messages, potentially with some additional information such as multimedia content (e.g., pictures, videos). Such messages are always public. The information should be easily searchable. A person can specify what information to read, either (1) by topic or (2) by publisher of the message. Additionally, users should be able to post comments to published messages.

Abgabedetails

Die Abgabe erfolgt als **PDF** Datei (keine Ausnahmen).

Universität Salzburg | Übungsleiter: Kwitt/Hütter/Resmerita

Datenbanken 1 – PS (501.073)

Projektabgabe – Teil 2 | Abzugeben bis **27.04.2018 (22:00)**

Im Folgenden finden Sie die Anforderungsanalyse in Deutsch/Englisch. Textpassagen formatiert als '`Beispiel`' stellen Entitäten, Beziehungen oder Attribute dar. Die Aufgabe besteht darin, die Anforderungen in ein entsprechendes ER-Diagramm umzusetzen *(Anm.: Benutzen Sie dazu die angegebenen Namen für Entitäten, Attribute und Beziehungen)*.

Anforderungen

1. Ein Benutzer (`User`) hat einen Namen (`Name`), zusammengesetzt aus Vor- und Nachname (`Firstname` & `Lastname`). Weiters hat der Benutzer ein Geburtsdatum (`Birthdate`), einen eindeutigen Spitznamen (`Nickname`), sowie in Passwort (`Password`). Ein Benutzer kann mehreren anderen Benutzer folgen (`follows`). Die Gesamtanzahl der "Followers" (`#Followers`) wird berechnet.
2. Ein Benutzer kann mehrere Beiträge (`Posting`) verfassen (`posts`).
3. Ein Benutzer kann mehrere Beiträge gut finden (`likes`). Überdies kann ein Beitrag von mehreren Benutzern gut gefunden werden.
4. Ein Beitrag (`Posting`) beinhaltet die Erstellungszeit (`Time`) des Beitrags, sowie eine Ortsangabe (`Location`). Weiters hat ein Beitrag eine ID (`PostingID`), welche zusammen mit dem Spitznamen (`Nickname`) des Benutzers eindeutig ist. Zudem kann ein Beitrag mit mehreren Schlagwörtern (`Tags`) gekennzeichnet werden und die Anzahl wie oft ein Beitrag für gut befunden wurde wird berechnet (`#Likes`).
5. Ein Beitrag kann nur einem Benutzer zugeordnet sein.
6. Ein Beitrag muss Information(en) (`Information`) enthalten (`contains`). Dies kann nur eine Information, aber auch mehrere Informationen sind.
7. Jede Information (`Information`) ist entweder ein Text (`Text`) oder Daten (`Data`). Die Information ist eindeutig durch eine ID gekennzeichnet (`InformationID`) und beinhaltet zudem die Grösse der Information in Bytes (`Size`). Eine Information muss mit genau einem Beitrag assoziert sein.
8. Text (`Text`) enthält eine Nachricht (`Message`).
9. Daten (`Data`) haben einen Link (`Link`).
10. Ein Beitrag kann mehrere Kommentare (`Comment`) haben (`has`).
11. Ein Benutzer kann Kommentare verfassen (`comments`).
12. Ein Kommentar beinhaltet eine Erstellungszeit (`Time`), einen Text (`Text`), sowie eine Kommentar ID (`CommentID`), welche zusammen mit der `PostingID` und dem `Nickname` eindeutig ist.
13. Ein Kommentar kann nur einem Benutzer und einem Beitrag zugeordnet sein.

Requirements

1. A user (`User`) has a name (`Name`), composed of first- and lastname (`Firstname` & `Lastname`). Further, a user has a birthdate (`Birthdate`), a unique nickname (`Nickname`) and a password (`Password`). A user can follow multiple users (`follows`). The total number of followers (`#Followers`) is a calculated.
2. A user can create/post (`posts`) one or more postings (`Posting`).
3. A user can like (`like`) other postings and postings can be liked by multiple users.
4. A posting (`Posting`) contains the time (`Time`), a location (`Location`) and an identifier (`PostingID`). This posting ID, together with the unique nickname of the user uniquely identifies the posting. A post can also be marked with multiple keywords/tags (`Tags`).
5. A post must contain (`contains`) one ore more information (`Information`) entries.
6. Information (`Information`) is uniquely identified by an ID (`InformationID`), holds the size of the information in bytes (`Size`), and can either in the form of text (`Text`) or data (`Data`). Information needs to be associated with exactly one posting.
7. Text (`Text`) contains a message (`Message`).
8. Data (`Data`) contains a link (`Link`) to some external source.
9. A posting can have (`has`) associated comments (`Comment`).
10. A user can write (`comments`) multiple comments.
11. A comment has an associated time (`Time`), a text (`Text`), as well as a comment ID that is unique in combination with `Nickname` and `PostingID`.
12. A comment can only be associated with one user and one posting.

Abgabedetails

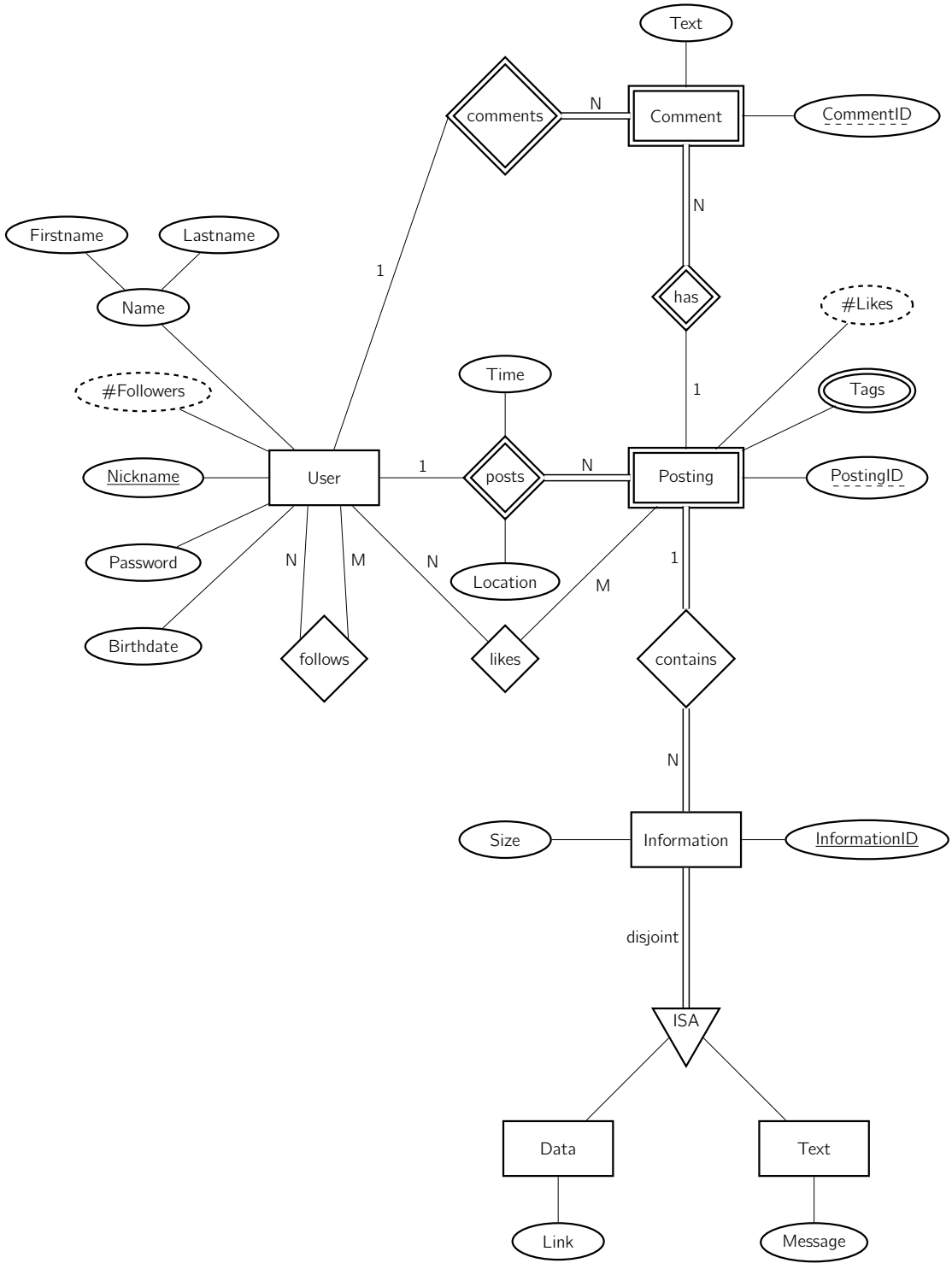
Die Abgabe erfolgt als **PDF** Datei (keine Ausnahmen). Das ER Diagramm *muss* die Symbole der Vorlesung verwenden und klar lesbar sein. Die Abgabe kann auch mit der Hand gezeichnet werden und danach eingescannt.

Universität Salzburg | Übungsleiter: Kwitt/Hütter/Resmerita

Datenbanken 1 – PS (501.073)

Projektabgabe – Teil 3 | Abzugeben bis **18.05.2018 (22:00)**

Im Folgenden finden Sie die Lösung zur letzten Abgabe. Die Aufgabe besteht nun darin, **dieses** ER-Diagramm in ein entsprechendes relationales Schema zu übersetzen. Primärschlüssel sind zu unterstreichen, Fremdschlüsselbeziehungen sind mit Projektion und Teilmengenoperator auszudrücken; die Notation der Vorlesung ist einzuhalten. Die Abgabe erfolgt als **PDF**-Datei.



Universität Salzburg — Übungsleiter: Kwitt/Hütter/Resmerita

Datenbanken 1 – PS (501.073)

Projektabgabe – Teil 4 — Abzugeben bis **8.6.2018 (22:00)**

Im Folgenden finden Sie das relationale Schema zum ER-Diagramm der letzten Abgabe. Die Primärschlüssel sind unterstrichen, die Fremdschlüsselbeziehungen sind in rot markiert.

Ihre Aufgabe besteht nun darin, dieses relationale Schema in SQL umzusetzen, also die Tabellen (mittels CREATE TABLE ... Anweisungen) zu generieren. Erstellen Sie dazu eine Datei `create.sql`, welche **alle** Anweisungen enthält. *Bitte achten Sie auf den korrekten Dateinamen.* Die `create.sql` Datei ist die einzige Datei die abgegeben werden soll.

Hinweis: Die `create.sql` Datei **muss** von PostgreSQL geladen werden können. Zum Testen könnten Sie auf Ihrem eigenen System beispielsweise eine Datenbank `mydb` anlegen und dann Ihre `create.sql` Datei laden:

```
createdb mydb
psql -d mydb -f create.sql
```

Wir werden die Abgaben automatisch überprüfen. Zunächst muss Ihr Skript auf PostgreSQL ausführbar sein, sonst wird Ihre Abgabe nicht weiter überprüft. Es gibt jeweils 0.4 Punkte für jede der folgenden Kategorien:

- jede Tabelle existiert mit richtigem Namen
- jedes Attribut existiert mit richtigem Namen und Datentyp
- jeder Primärschlüssel existiert
- jeder Fremdschlüssel existiert
- jede Beschränkung der erlaubten Datenwerte wurde angelegt

Die Kategorien werden in dieser Reihenfolge überprüft. Sobald in einer Kategorie ein Fehler gefunden wurde, bricht die Evaluation ab – diese und alle weiteren Kategorien werden dann mit 0 Punkten gewertet. Ein erneuter Upload ist jederzeit möglich.

Die automatische Evaluierung erfolgt mit leichter Verzögerung – es kann sein, dass Sie einige Minuten warten müssen, bis das Ergebnis vorliegt.

Abbildung von Entitäten

Anmerkung: Alle Fremdschlüsselattribute müssen NOT NULL sein.

Person[Nickname, Firstname, Lastname, Password, Birthdate]

Bitte um folgende Benennung + Datentypen im `create.sql` Skript: Tabellenname: Person

Nickname	TEXT
Firstname	TEXT
Lastname	TEXT
Password	TEXT
Birthdate	TIMESTAMP

Anmerkung: *Die Werte von Password müssen NOT NULL und länger als 0 Zeichen sein (Benutzen Sie dazu die Funktion `char_length`).*

Posting[PostingID, Nickname, Time, Location]
$\pi_{\text{Nickname}}(\text{Posting}) \subseteq \pi_{\text{Nickname}}(\text{Person})$

Bitte um folgende Benennung + Datentypen im `create.sql` Skript: Tabellenname: Posting

PostingID	BIGINT
Nickname	TEXT
Time	TIMESTAMP
Location	TEXT

Anmerkung: *Die Werte von Time müssen NOT NULL sein.*

Comment[CommentID, PostPostingID, PostNickname, CommenterNickname, CommentText]
$\pi_{\text{PostPostingID, PostNickname}}(\text{Comment}) \subseteq \pi_{\text{PostingID, Nickname}}(\text{Posting})$
$\pi_{\text{CommenterNickname}}(\text{Comment}) \subseteq \pi_{\text{Nickname}}(\text{Person})$

Bitte um folgende Benennung + Datentypen im `create.sql` Skript: Tabellenname: Comment

CommentID	BIGINT
PostPostingID	BIGINT
PostNickname	TEXT
CommenterNickname	TEXT
CommentText	TEXT

Information[InformationID, PostingID, Nickname, Size]
$\pi_{\text{PostingID, Nickname}}(\text{Information}) \subseteq \pi_{\text{PostingID, Nickname}}(\text{Posting})$

Bitte um folgende Benennung + Datentypen im `create.sql` Skript: Tabellenname: Information

InformationID	BIGINT
PostingID	BIGINT
Nickname	TEXT
Size	BIGINT

Data[InformationID, Link]
$\pi_{\text{InformationID}}(\text{Data}) \subseteq \pi_{\text{InformationID}}(\text{Information})$

Bitte um folgende Benennung + Datentypen im `create.sql` Skript: Tabellenname: Data

Link	TEXT

Anmerkung: *Die Werte von Link müssen NOT NULL und länger als 0 Zeichen sein. Die Datentypen und Namen der anderen Attribute entnehmen Sie den entsprechenden Tabellen.*

Text[InformationID, Message]
$\pi_{\text{InformationID}}(\text{Text}) \subseteq \pi_{\text{InformationID}}(\text{Information})$

Bitte um folgende Benennung + Datentypen im `create.sql` Skript: Tabellenname: Text

Message	TEXT

Anmerkung: *Die Werte von Message müssen NOT NULL und länger als 0 Zeichen sein. Die Datentypen und Namen der anderen Attribute entnehmen Sie den entsprechenden Tabellen.*

Abbildung von Relationen als Entitäten

Follows[FollowerNickname, FolloweeNickname]
$\pi_{\text{FollowerNickname}}(\text{Follows}) \subseteq \pi_{\text{Nickname}}(\text{Person})$
$\pi_{\text{FolloweeNickname}}(\text{Follows}) \subseteq \pi_{\text{Nickname}}(\text{Person})$

Bitte um folgende Benennung + Datentypen im `create.sql` Skript: Tabellenname: Follows, Attribute: FollowerNickname und FolloweeNickname.

Likes[PostPostingID, PostNickname, LikeeNickname]
$\pi_{\text{PostPostingID, PostNickname}}(\text{Likes}) \subseteq \pi_{\text{PostingID, Nickname}}(\text{Posting})$
$\pi_{\text{LikeeNickname}}(\text{Likes}) \subseteq \pi_{\text{Nickname}}(\text{Person})$

Bitte um folgende Benennung + Datentypen im `create.sql` Skript: Tabellenname: Likes, Attribute: PostNickname, PostPostingID, LikeeNickname.

Abbildung von mehrwertigen Attributen als Entitäten

Tags[PostingID, Nickname, Tag]
$\pi_{\text{PostingID, Nickname}}(\text{Tags}) \subseteq \pi_{\text{PostingID, Nickname}}(\text{Posting})$

Bitte um folgende Benennung + Datentypen im `create.sql` Skript: Tabellenname: Tags

Tag	TEXT

Anmerkung: *Die Werte von Tag müssen NOT NULL und länger als 0 Zeichen sein. Die Datentypen und Namen der anderen Attribute entnehmen Sie den entsprechenden Tabellen.*

Universität Salzburg | Übungsleiter: Kwitt/Resmerita/Hütter

Datenbanken 1 – PS (501.073)

Projektabgabe – Teil 5 | Abzugeben bis **29.06.2018 (22:00)**

Im letzen Teil des Projekts stellen wir Ihnen die Datenbank aus den letzten Projektabgaben, sowie ein Python Programm `query.py` zur Verfügung. Mittels

1. `create.sql`
2. `pop.sql`
3. `drop.sql`

kann die Datenbank (1) erstellt, (2) befüllt, sowie (3) gelöscht werden. Weitere Details finden Sie im Abschnitt *Testen Ihrer SQL Statements auf Ihrem lokalen Rechner*.

Das Python Programm `query.py` ließt eine Datei von SQL Statements. Die einzelnen Queries sind durch Kommentarzeilen (`-- QUERY n`) voneinander getrennt.

Hier ein **Beispiel**:

```
python query.py sql_queries.sql 3 count=5
```

Die SQL Datei `sql_queries.sql` sieht dabei beispielsweise so aus:

```
-- QUERY 1
select nickname
from person
where nickname=%(nickname)s;
-- QUERY 2
select * from text;
```

Die Reihenfolge der Queries ist unerheblich, Sie können also auch Query 2 vor Query 1 angeben.

Das Programm `query.py` ließt zuerst den Namen der Datei mit SQL Statements (hier: `sql_queries.sql`), dann einen Integer Wert zwischen 1 und 10, der das Statement in der Datei `sql_queries.sql` identifiziert. Danach werden die Parameter in der Form `parametername=parameterwert` angegeben – siehe Beispiel. In den Queries werden die Parameter dann in der Form `%(parametername)s` verwendet.

Ihre Aufgabe ist es, die SQL Statements in eine Datei `sql_queries.sql` zu schreiben.
(**Bitte genau diesen Dateinamen verwenden**)

Folgend sind die **10** (zehn) zu implementierenden SQL Anfragen aufgelistet. In den Kästchen finden Sie jeweils das geforderte Ausgabeformat.

1. Geben Sie den Nicknamen aller Personen aus, die mindestens einen Text gepostet haben. Jeder Nickname soll nur einmal ausgegeben werden. Ordnen Sie das Ergebnis nach Nickname.

```
nickname
```

2. Geben Sie die Anzahl an Personen aus, die nach dem '1.1.1995' geboren sind.

```
count
```

3. Geben Sie die Location aller Postings der Person "Bearbugar" aus, welche mehr als `%(count)s` Likes haben. Ordnen Sie das Ergebnis nach Location.

```
location
```

4. Geben Sie den am häufigsten verwendeten Tag der Person mit Nicknamen `%(nickname)s` aus. Ordnen Sie das Ergebnis nach Tag.

```
tag | count
```

5. Geben Sie alle Posts ohne Likes aus. Ordnen Sie das Ergebnis nach Nickname.

```
nickname | postingid
```

6. Geben Sie die `%(nth)s`-jüngste Person aus. Ordnen Sie das Ergebnis nach Nickname. Anmerkung: Falls es mehrere Personen mit dem gleichen Geburtsdatum gibt, haben diese die gleiche Position in der Sortierreihenfolge und sollen alle ausgegeben werden.

```
nickname | firstname | lastname | birthdate
```

7. Geben Sie von allen Personen, deren Nickname dem Muster `%(personpattern)s` (LIKE-Syntax) entspricht, den Nicknamen der Person mit der *geringsten* Anzahl an Followern aus. Sollten mehrere Personen dieses Kriterium erfüllen, sollen alle Nicknamen (alphabetisch sortiert) ausgegeben werden.

```
nickname | followercount
```

8. Geben Sie die Primärschlüsselattribute (PostingID, Nickname) aller Postings aus, die sowohl Data als auch Text beinhalten. Ordnen Sie das Ergebnis nach Nickname und PostingID.

```
postingid | nickname
```

9. Geben Sie die Primärschlüsselattribute (PostingID, Nickname) aller Postings aus, deren Informationen in Summe größer als `%(size)s` sind und mit dem Tag `%(tag)s` versehen sind. Ordnen Sie das Ergebnis nach Nickname und PostingID.

```
postingid | nickname | totalsize
```

10. Geben Sie alle Tags (aufsteigend sortiert) der Postings mit den meisten Likes aus. Sollten mehrere Postings die gleiche Anzahl an Likes haben, sollen alle Tags dieser Postings ausgegeben werden. Etwaige Duplikate sollen eliminiert werden.

```
tag
```

Testen Ihrer SQL Statements auf Ihrem lokalen Rechner

Hinweis: Die folgenden Statements und Pfade beziehen sich auf eine Beispielinstallation von Postgres. Im Programm `query.py` stellen wir die Verbindung zur Datenbank mit dem Python Modul `psycopg2` her. Unter Linux (bzw. Mac) können Sie dieses Modul beispielsweise mit

```
pip install psycopg2
```

oder (auf Debian-basierten Linuxdistributionen)

```
apt-get install python-psycopg2
```

installieren. Weitere Informationen finden Sie online unter:

- `https://wiki.postgresql.org/wiki/Using_psycopg2_with_PostgreSQL`
- `http://initd.org/psycopg/docs/`

Zum Testen ihrer SQL Statements installieren und starten Sie einen lokalen PostgreSQL Server. Entsprechende Anleitungen sind im Internet zu finden, für Debian-basierte Systeme z.B. `https://wiki.debian.org/PostgreSql`. Danach ist eine Datenbank zu erstellen, z.B. `mydb`. Weiters erstellen Sie die entsprechenden Tabellen und befüllen diese.

```
createdb mydb
psql -d mydb -f drop.sql
psql -d mydb -f create.sql
psql -d mydb -f pop.sql
```

Nun können Sie `query.py` wie folgt ausführen:

```
python query.py --connection-string "host='localhost' dbname='mydb'" sql_queries.sql 3 count=5
```

Alternativ, können Sie auch den Source Code von `query.py` editieren und die Variable

```
default_conn_string = "host='localhost' dbname='mydb'"
```

entsprechend setzen; dann brauchen Sie den Parameter `--connection-string` NICHT mehr anzugeben.

Evaluierung

Wir evaluieren Ihre Lösung gegen unsere Musterlösung, d.h., gegen die korrekte Anzahl (und Reihenfolge) der zurückgegebenen Tupel und die korrekte Anordnung der Attribute.