Freie Universität Berlin

Prof. Dr. Agn‘es Voisard, Nicolas Lehmann

Datenbanksysteme, SoSe 2017

Übungsblatt 5

Tutor: Nicolas Lehmann
Tutorium 10

Boyan Hristov, Julian Habib

8. Juni 2017

Link zum Git Repository: https://github.com/BoyanH/Freie-Universitaet-Berlin/tree/master/Datenbanksysteme/Solutions/homework5

1. Aufgabe

a)

$$
\begin{aligned}
\text{Minimal F} = \{ & \\
& \{B\} \to \{C\} \\
& \{B\} \to \{D\} \\
& \{C\} \to \{E\} \\
& \{D\} \to \{E\} \\
& \{AE\} \to \{D\} \\
\} & \\
& \Rightarrow \\
F+ = \{ & \\
& \{B\} \to \{C\} \\
& \{B\} \to \{D\} \\
& \{B\} \to \{CD\} \\
& \{C\} \to \{E\} \\
& \{D\} \to \{E\} \\
& \{AE\} \to \{D\} \\
& \{B\} \to \{C\} \\
& \{B\} \to \{D\} \\
& \{BC\} \to \{CDE\} \\
& \{BD\} \to \{CDE\} \\
& \{CD\} \to \{E\} \\
& \{AB\} \to \{CDE\} \\
& ... \\
\} &
\end{aligned}
$$

Damit ist AB ein Key / Schlüssel.

b) Pseudotransitivität wäre hier

$$
\begin{aligned}
& \{C\} \to \{E\} \wedge \{AE\} \to D \\
\Rightarrow & \{AC\} \to \{D\}
\end{aligned}
$$

c)

$$
\begin{aligned}
& \{BE\} \to \{B\}(\text{Reflexivity}) \\
\Rightarrow & \{A\} \to \{BE\} \wedge \{BE\} \to \{B\} \\
\Rightarrow & \{A\} \to \{B\}(Transitivity) \\
\Rightarrow & \{A\} \to \{B\} \wedge \{A\} \to \{A\}(\text{Reflexivity}) \\
\Rightarrow & \{A\} \to \{AB\}(\text{Union}) \wedge \{AB\} \to \{C\} \\
\Rightarrow & \{A\} \to \{C\}(\text{Transitivity}) \\
\Rightarrow & \{A\} \to \{C\} \wedge \{C\} \to \{D\} \\
\Rightarrow & \{A\} \to \{D\}(\text{Transitivity})
\end{aligned}
$$

2. Aufgabe

a) Attributhülle von {A}

Es gibt keine funktionale Abhängigkeiten für den Attribut A, deswegen ist seine Attributhülle {A} den Attribut selbst.

Attributhülle von {B}

$$\{B\} \to \{AD\} \Rightarrow \{B\} \to \{D\} \text{(Reflexivity)}$$
$$\Rightarrow \{B\} \to \{D\} \wedge \{D\} \to \{BC\}$$
$$\Rightarrow \{B\} \to \{C\} \text{(Transitivity)}$$

$$\Rightarrow \{B\} \to \{C\} \wedge \{B\} \to \{D\}$$
$$\Rightarrow \{B\} \to \{CD\} \text{(Union)}$$
$$\Rightarrow \{B\} \to \{CD\} \wedge \{CD\}\{AEF\}$$
$$\Rightarrow \{B\} \to \{AEF\} \text{(Transitivity)}$$

$$\{B\} \to \{B\} \text{(Reflexivity)}$$

$$\{B\} \to \{ABCDEF\}$$

Attributhülle von {B} = B^+ = {ABCDEF}. Deswegen ist B ein Schlüssel.

b) Wie wir bei Aufgabenteil a) gesehen haben ist B ein Schlüsel. Wegen $\{D\} \to \{BC\}$ ist D ein weiteres Schlüssel. Wegen $\{F\} \to \{ABD\}$ ist F auch ein Schlüssel. Von $\{E\} \to \{CD\}$ ist E auch Schlüssel. Weitere Kandidaten gibt es nicht, da per Definition ein Schlüssel das kleinste Superkey ist und alle andere Superkeys haben mindestens 2 Attributen.

Kandidaten für Schlüssel: B, D, E, F

c) Wir haben aus allen Kandidaten B für Schlüssel gewählt, da dieser am trivialsten zu sehen ist und auch alphabetisch der ersten Attribut ist. Da es keine feste Regeln gibt, welcher von den Kandidaten zu einem Schlüssel wird, war unsere Entscheidung nicht so schwer zu treffen.

3. Aufgabe

$$
\begin{aligned}
FD(R_2) = \{ & \\
& \{A\} \rightarrow \{D, E\} \\
& \{B\} \rightarrow \{C, D\} \\
& \{C\} \rightarrow \{A\} \\
& \{D, E\} \rightarrow \{A, C\} \\
& \} \\
& \\
& \{B\} \rightarrow \{CD\} \Rightarrow \{B\} \rightarrow \{C\} \text{(Reflexivity)} \\
& \Rightarrow \{B\} \rightarrow \{C\} \wedge \{C\} \rightarrow \{A\} \\
& \Rightarrow \{B\} \rightarrow \{A\} \text{(Transitivity)} \\
& \Rightarrow \{B\} \rightarrow \{A\} \wedge \{A\} \rightarrow \{DE\} \\
& \Rightarrow \{B\} \rightarrow \{DE\} \text{(Transitivity)} \\
& \Rightarrow \{B\} \rightarrow \{D\} \text{(Reflexivity)} \\
& \Rightarrow \text{Dependency } \{B\} \rightarrow \{CD\} \text{ kann ducrh } \{B\} \rightarrow \{C\} \text{ substituiert werden} \\
& \\
& \{C\} \rightarrow \{A\} \Rightarrow \text{ Dependency } \{DE\} \rightarrow \{A\} \text{ ist wegen dependency} \{DE\} \rightarrow \{C\} \text{ nicht notwendig} \\
& \\
\Leftrightarrow & \\
FD(R_2) = \{ & \\
& \{A\} \rightarrow \{D, E\} \\
& \{B\} \rightarrow \{C\} \\
& \{C\} \rightarrow \{A\} \\
& \{D, E\} \rightarrow \{C\} \\
& \} \\
& \\
\Leftrightarrow & \text{(Decomposition)} \\
& \\
FD(R_2) = \{ & \\
& \{A\} \rightarrow \{D\} \\
& \{A\} \rightarrow \{E\} \\
& \{B\} \rightarrow \{C\} \\
& \{C\} \rightarrow \{A\} \\
& \{D, E\} \rightarrow \{C\} \\
& \}
\end{aligned}
$$

Damit sind alle Anforderungen aus der Vorlesung erfüllt →

1. Every right side of a dependency in F is a single attribute. (apply decomposition)

2. For no $X \to A$ in F is the set $F - \{X \to A\}$ equivalent to F.
3. For no $X \to A$ in F and subset Z of X is $F - \{X \to A\} \cup \{Z \to A\}$ equivalent to F.

Aufgabe 5

a) Der Scraper liest mehrere Seiten von einem Web Application, und zwar auf mehrere Sprachen über die Jahren 2000 bis 2015 inklusive. Von den Seiten nimmt er immer die erste Tabelle, davon alle Zeilen mit rechtsbündigen Text. Für jede Zeile in der Tabelle speichert der Scraper eine Zeile in .csv Datei, wo Land, Jahr und die Zellen in der Tabelle nacheinander stehen.

Leider könnten wir den Scraper nicht ausführen, da diese Seiten nicht erreichbar waren. Laut Julian aber speichtert der Scraper alle Informationen von Hunderassen der Jahre 2000-2016 in verschiedene Sprachen in einer CSV Datei.

Nach mehrere versuche und VPN (war es überhaupt notwendig?) und nachdem es spät genug war, damit nicht alle wild die Seite scrapen, haben wir gesehen, dass auf der Seite eine Tabelle von den besten Hunden in Wettbewerben steht, je mit Name, Geschlecht, abgeschlossene Wettbewerbe usw. Jede Zeile aus alle solchen Tabellen wird eine Zeile in unsere CSV Datei und dazu werden noch zwei Spalten für Land und Jahr eingefügt. Im großen und ganzen wird eine perfekte CSV Datei für das DBS Projekt im nächsten Semester :)

b)

```python
from bs4 import BeautifulSoup
import requests
import re
from collections import Counter

PER_PAGE_NAVIGATION_CLASS = 'seitenweise_navigation'
PAGINATION_CLASS = 'pagination'
MAIN_PAGE_URL = 'https://www.heise.de/thema/https'
SITE_SUFFIX = '?seite='

# this function returns a soup page object
def getPage(url):
    r = requests.get(url)
    data = r.text
    spobj = BeautifulSoup(data, "lxml")
    return spobj

# pushes article soup objects into the passed articles array
def parsePage(soup, articles):
    # Summary:
    # In each page, get the <aside class="recommendations"> elements
    # Inside, search for a <div class="recommendation">
    # Every div of that kind is an article, add it to the articles array passed to the function

    recommendationsAside = soup.find('aside', class_ = 'recommendations')
    recommendations = recommendationsAside.find_all('div', class_ = 'recommendation')

    for rec in recommendations:
        articles.append(rec)

# returns array
def getHeadings(articles):
    # Summary:
    # get all <header> tags in an article,
    # remove new lines and extra spaces and return the result
```

```
    headings = []

    for article in articles:
        heading = article.find('header')
        headingText = heading.text.replace('\n', '').strip()
        headings.append(headingText)

    return headings

# returns Dictionary<Word, Count>
def getPerWordCountInHeadings(headings):
    # Summary:
    # All symbol are removed from the article except for the one in the german language and dashes
    # Words are then separated by dashes and spaces and counted

    perWordCount = {}

    for heading in headings:
        #re.sub...
        words = re.sub('[^a-zA-Z äÄöÖüÜß\-\'`]', '', heading).replace('-', ' ').lower().split(' ')

        for word in words:
            if len(word) < 3:
                continue
            if not word in perWordCount:
                perWordCount[word] = 1
            else:
                perWordCount[word] += 1

    return perWordCount

# returns array
def getTopNWords(wordCountDict, n):
    # Summary:
    # Using the functionality from collections.Counter get the top 3 keys
    # in the dictionary sorted by their value and push them to the returned array

    counter = Counter(wordCountDict)
    topNWords = []

    for word in counter.most_common(n):
        topNWords.append(word)

    return topNWords


# scraper website: greyhound-data.com
def main():

    articles = []

    mainPage = getPage(MAIN_PAGE_URL)
    navigation = mainPage.find_all('nav', class_ = PER_PAGE_NAVIGATION_CLASS)[0]
    navItemsContainer = mainPage.find('span', class_ = PAGINATION_CLASS)
    nonActivePages = navItemsContainer.find_all('a')
    furtherPages = len(nonActivePages)

    parsePage(mainPage, articles);

    for page in range(1, furtherPages+1):
        furtherPage = getPage(MAIN_PAGE_URL + SITE_SUFFIX + str(page))
        parsePage(furtherPage, articles)

    headings = getHeadings(articles)
    perWordCount = getPerWordCountInHeadings(headings)
```

```
102     top3Words = getTopNWords(perWordCount, 3)
104     print("\nDie meist benutzten 3 Wörter mit mehr als 3 Buchstaben inklusive sind: \n
        ")
105     for word in top3Words:
106         print("{0}: {1} mal".format(word[0], word[1]))
107     print("\n\nWichtig: Nur die Artikeln link die zu HTTPS releavnt sind wurden
        analyziert!\n")


110 # main program

112 if __name__ == '__main__':
113     main()
```