NSA (Teil 2)
Analytisches SQL und Big Data

VL Big Data Engineering
(vormals Informationssysteme)

Prof. Dr. Jens Dittrich

bigdata.uni-saarland.de

12. Mai 2022

Letzte Woche:

1. Konkrete Anwendung: NSA

- Snowden, Spionageaffäre, kurze Einführung, Links zum Weiterlesen
- Daten vs Metadaten
- das Buch von Eschbach

NSA (Teil 2)

2. Was sind die Datenmanagement und -analyseprobleme dahinter?

Frage 1

Wie stellen wir so komplexe Anfragen?

- SQL!
- Einfache Beispiele in Jupyter mit Sqlite (Sqlite ist von den Fähigkeiten/der Performance vergleichbar mit einer sehr einfachen Jolle)

NSA (Teil 2)

Jetzt:

4. Transfer der Grundlagen auf die konkrete Anwendung

Danach:

Frage 2

... und welche ethischen Probleme entstehen durch diese Anfragen? Wie gehen wir damit um?

nächste Woche:

Frage 3

... und was ist, wenn die Daten größer werden? Wie kommen wir eigentlich von SQL zu einem effizienten Programm?

Die fatale Grundidee einer der Anfragen im Buch „NSA“

Anfrageidee 1

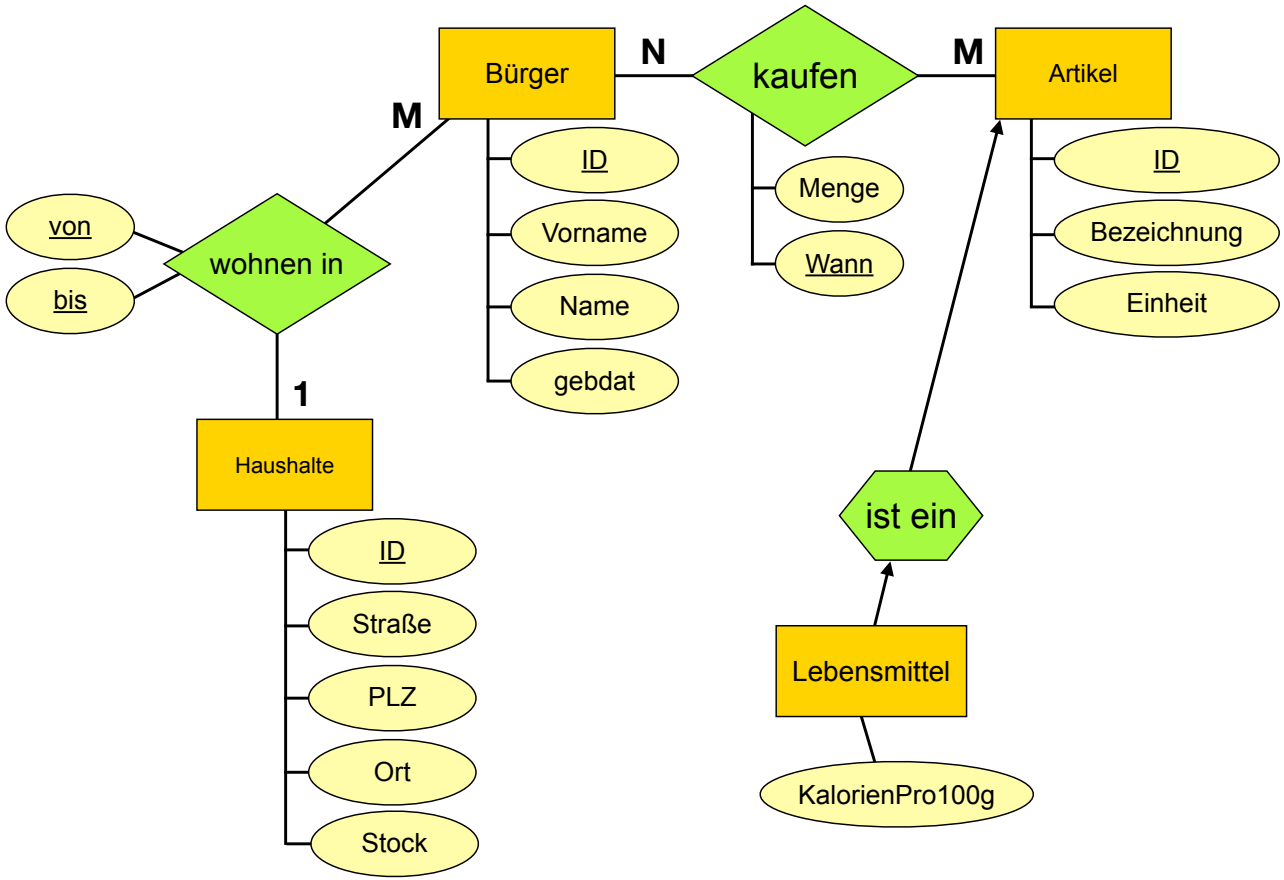
Jeder Mensch nimmt im Schnitt ungefähr gleich viele Kalorien zu sich. Kauft jemand im Schnitt viel mehr Kalorien als der Durchschnitt, kann das darauf hinweisen, dass sie/er heimlich jemanden mitversorgt (mit anderen Worten: versteckt!).

Um diese Anfrage zu beantworten müssen wir also:

1. den durchschnittlichen Kalorienverbrauch eines Menschen abschätzen
2. pro Haushalt ermitteln, wieviele Kalorien im Schnitt eingekauft werden
3. den durchschnittlichen Kalorienverbrauch pro Person im Haushalt ermitteln
4. die Ergebnisliste absteigend sortiert ausgeben: höchster Verbrauch pro Person zuerst, nur die auffälligsten Haushalte
5. im Buch wird der nächste Schritt von der SS ausgeführt...

Bei welchem Schritt beginnt hier die ethische Verantwortung von uns Informatikern und Data Scientists?

Teil des ER-Modells aus dem Buch (bereits verbessert)

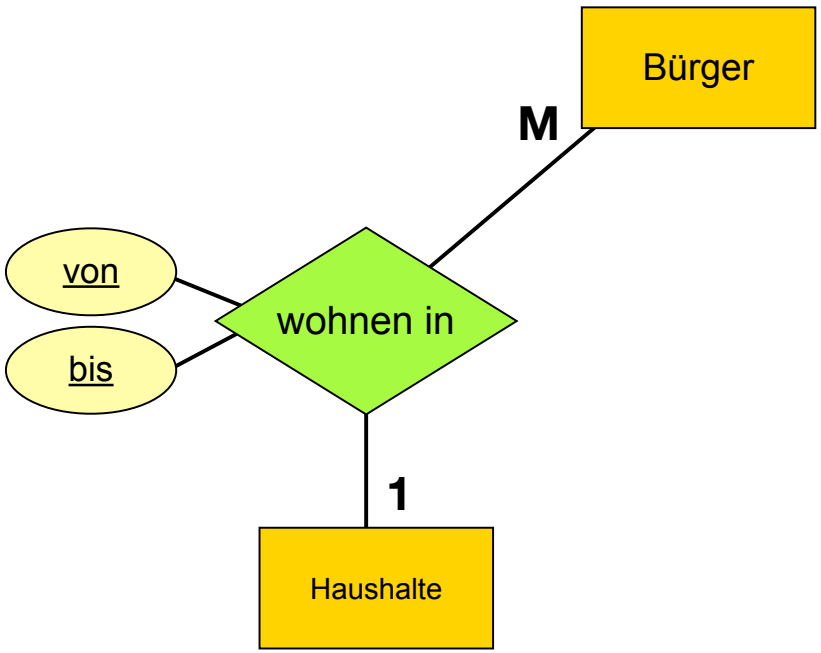


Achtung

Zeit ist in „wohnen in“ als zusätzlicher Teil des Schlüssels modelliert!
D.h. wir erlauben ab sofort, dass Beziehungsattribute, die zeitliche Zusammenhänge modellieren, Teil des Schlüssels sind!

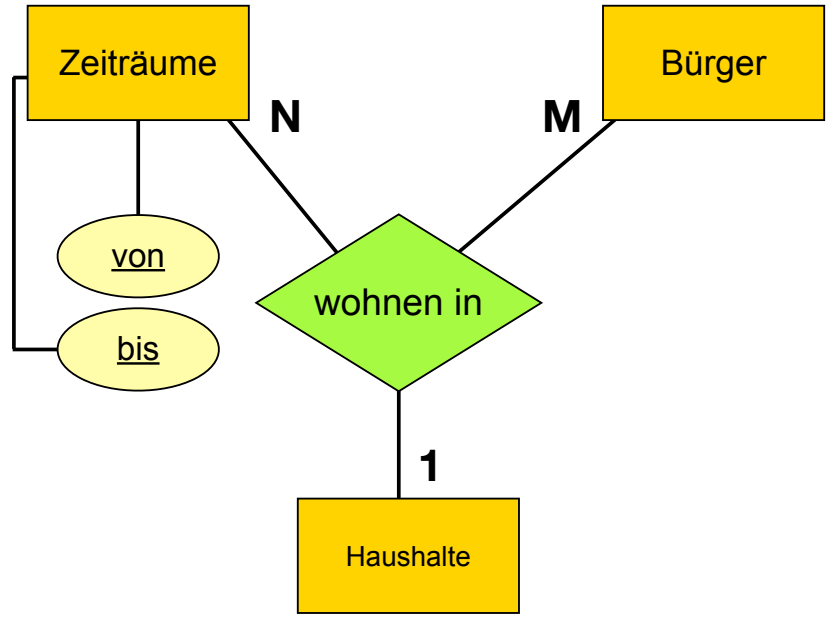
Warum? Attribut vs Entitätstyp

(1)



Entweder: Zeiträume als Schlüsselattribute des Beziehungstyps modelliert (verkürzte Modellierung, nicht ganz korrekt im Sinne der Definition im Kapitel zu ER, aber lesbarer)

(2)



Oder: Zeiträume als separater Entitätstyp modelliert

Umsetzung ins Relationale Modell

Achtung

Bei der Umsetzung ins relationale Modell kommt in beiden Fällen (fast) dasselbe Ergebnis raus! Hier die direkte Umsetzung von (1):

[wohnen in] : {[
bürger_id:(Bürger→ID),
von:timestamp,
bis:timestamp,
haushalt_id:(Haushalte→ID)
]}

Notation für zusammengesetzte Fremdschlüssel

Umsetzung von (2):

[wohnen in] : {[
...
[von,bis]:(Zeiträume→[von,bis])
...
]}

Leseweise mehrstelliger Beziehungstypen

- Zeiträume $\times$ Bürger $\mapsto$ Haushalte

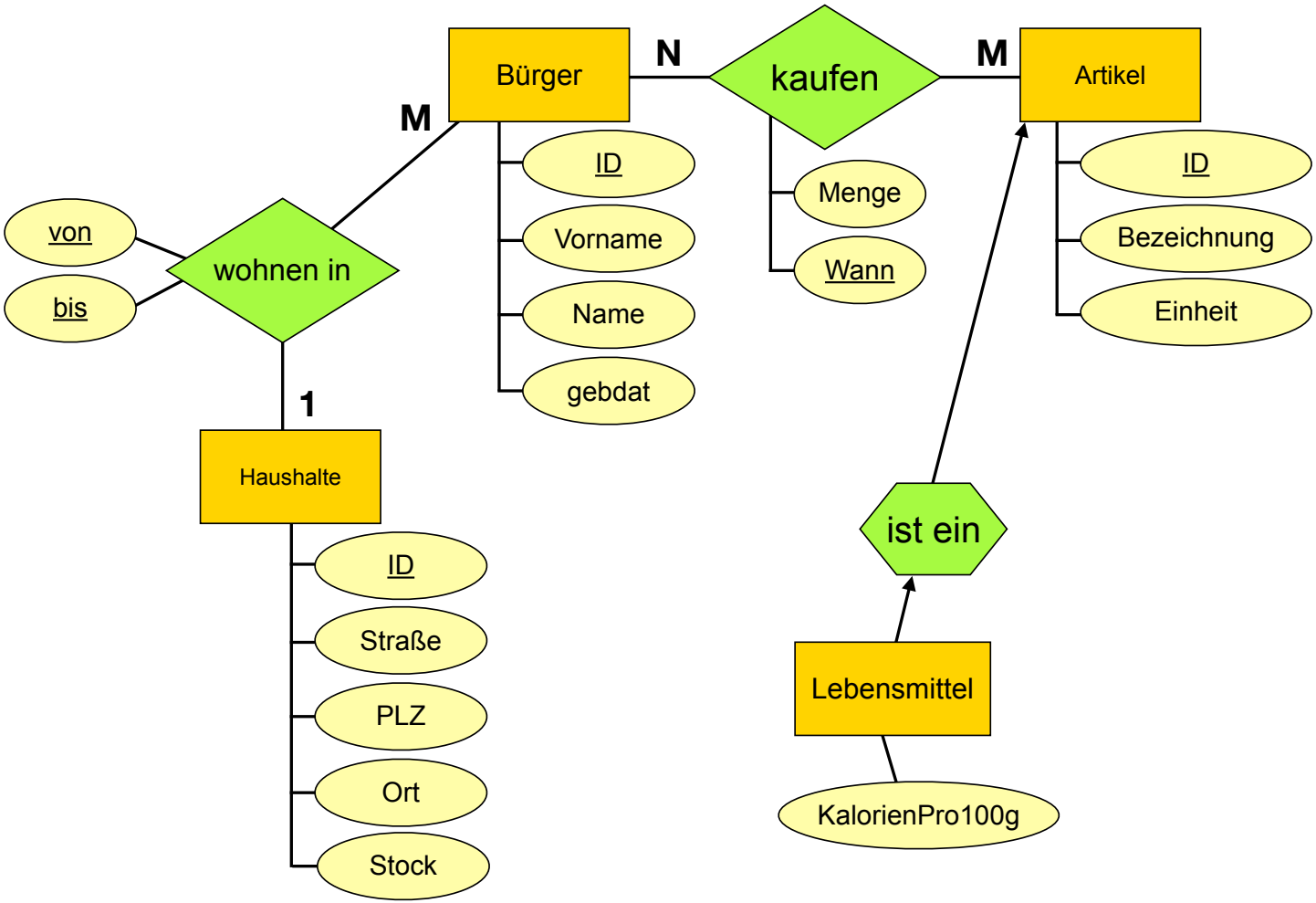
  Ein Zeitraum und ein Bürger **bestimmen zusammen** den Haushalt: zu einem gegebenen Zeitraum und Bürger gibt es entweder keinen oder einen Haushalt aber nicht mehrere. Ein Bürger kann zu einem Zeitraum nur in einem Haushalt gemeldet sein.
- Haushalte $\times$ Zeiträume $\nmapsto$ Bürger

  Ein Haushalt und ein Zeitraum **bestimmen nicht zusammen** den Bürger: zu einem gegebenen Haushalt und Zeitraum gibt es beliebig viele Bürger. Im selben Zeitraum können im selben Haushalt mehrere Bürger gemeldet sein.
- Bürger $\times$ Haushalte $\nmapsto$ Zeiträume.

  Ein Bürger und ein Haushalt **bestimmen nicht zusammen** den Zeitraum: zu einem gegebenen Bürger und Haushalt gibt es beliebig viele Zeiträume. Derselbe Bürger kann im selben Haushalt zu mehreren Zeiträumen gemeldet sein.

P.S.: Aber was passiert eigentlich, wenn [von;bis]-Zeiträume überlappen? Upps!

Teil des ER-Modells aus dem Buch (bereits verbessert)



Relationales Modell hierfür

[Haushalte] : {[
    ID:int,
    Straße:str,
    PLZ:int,
    Ort:str,
    Stock:int
]}

[Bürger] : {[
    ID:int,
    Vorname:str,
    Name:str,
    Gebdat:timestamp
]}

[Artikel] : {[
    ID:int,
    Bezeichnung:str,
    Einheit:str
]}

[Lebensmittel] : {[
    ID:(Artikel→ID),
    KalorienPro100g:int
]}

[wohnen in] : {[
    Bürger_ID:(Bürger→ID),
    von:timestamp,
    bis:timestamp
    Haushalt_ID:(Haushalte→ID)
]}

[kaufen] : {[
    Artikel_ID:(Artikel→ID),
    Bürger_ID:(Bürger→ID),
    Wann:timestamp,
    Menge:int
]}

Das NSA-Szenario aus dem Buch in Python

Citizens & Households

In a first step, we want to show the citizens and the households they currently live in. The current household can be determined by looking at the "until" attribute. If it is the empty string (remember: the fields are generated with datatype "Text"), the citizen currently lives in this household. If it is non-empty, the citizen lived here in the past.

In [ ]:
```
SELECT citizens.firstname, citizens.lastname, households.street, households.postcode, households
FROM citizens
    JOIN livingIn ON citizens.id = livingIn.citizen_id
    JOIN households ON households.id = livingIn.household_id
WHERE livingIn.until IS ""
LIMIT 10;
```

Count Inhabitants

To search for hidden persons, we first need to know the number of (official) inhabitants of each household. This can be achieved by grouping over the id of the household and counting the number of citizen_id within each group.

In [ ]:
```
DROP VIEW IF EXISTS inhabitantsPerHousehold;
CREATE VIEW inhabitantsPerHousehold AS
SELECT livingIn.household_id AS household_id, COUNT(livingIn.citizen_id) AS numInhabitants
FROM livingIn
WHERE livingIn.until is ""
GROUP BY livingIn.household_id;
```

https://github.com/BigDataAnalyticsGroup/bigdataengineering/blob/master/NSA.ipynb

NSA (Teil 1&2)

Frage 2

... und welche ethischen Probleme entstehen durch diese Anfragen? Wie gehen wir damit um?

Ethische Probleme

Im Buch von Eschbach konzipiert die Hauptfigur Helene, Mitarbeiterin des NSA, diverse „SAS“-Anfragen und hilft so, versteckte Menschen zu finden.

Auswirkungen von Helenes SQL-Anfragen

- Ohne Helenes Arbeit wären viele dieser Menschen vermutlich niemals gefunden worden.
- Was bedeutet dies für die Arbeit von Helene?

Zum großen Teil wird dies bereits im Buch von Eschbach erläutert. Helene findet mit ihren diversen Anfragen auch ihren Geliebten, der sich ebenfalls versteckt hält.

Kernfrage

Kernfrage

- Was hätte in diesem Szenario verhindert, dass diese Menschen gefunden werden?
- Was hätte Helene tun können?

Achtung

Keine der folgenden Hinweise ist als Handlungsanweisung zu verstehen. Einige dieser Aktionen sind strafbar. Wir diskutieren hier prinzipiell, was für Aktionsmöglichkeiten Helene oder irgendjemand anders im Rahmen zivilen Ungehorsams/zivilen Widerstands in einer vergleichbaren Situation gehabt hätte.

Technische Ansätze 1/6

Datensparsamkeit

Beispiel: Bargeld vs elektronischer Zahlungsverkehr

Problem: insgesamt inzwischen sehr unrealistisch, aber grundsätzlich hilfreich; im Buch wird Bargeld verboten, ob man damit einen Schwarzmarkt verhindern kann, ist fraglich

Nur wenige Attribute speichern

Beispiel: um zu zeigen, dass ich etwas gekauft habe, brauche ich nicht preiszugeben, was ich gekauft habe

Problem: kann ich das Schema vorher überhaupt einschränken? Das machen wir bei der Datenmodellierung ja sowieso! Sind gewisse Attribute später nicht doch problematisch?

Technische Ansätze 2/6

Anonymisieren von Daten

Beispiel: Anonymisieren von Suchanfragen in einer Suchmaschine

Problem: oft wieder deanonymisierbar durch Joins und meist nur **pseudoanonym**, z.B. AOL-Suchanfragen, [NYT-Artikel]

Verschlüsseln von Teilen der Daten

auf Attribut-, Tupel-, Tabellen- oder Datenbankebene

Problem: Join mit (unbekannter) Klartextdatenbank bricht diese Verschlüsselung möglicherweise, aber sicherlich besser als gar nicht zu verschlüsseln (Aufwand für den Angreifer steigt)

Technische Ansätze 3/6

Joins schwierig machen, Entfernen möglicher Join-Schlüssel

Beispiel: keine einheitliche Bürger-ID, temporäre Schlüssel

Problem: eventuell trotzdem rekonstruierbar, allerdings nur mit massiv höherem Aufwand

Beispiel für die Verwendung temporärer Schlüssel: Die Corona-Warn-App

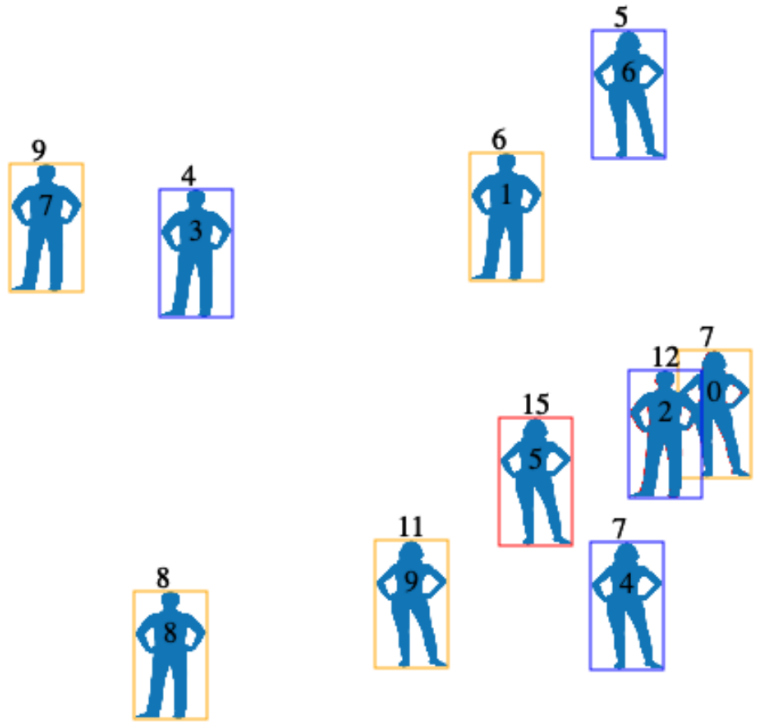
Die Corona-Warn-App

- die Corona-Warn-App (offiziell, wiki) ist eine seit 2020 verfügbare App zur Kontaktpersonennachverfolgung
- entwickelt von SAP SE und Deutsche Telekom
- dezentraler Ansatz, entwickelt nachdem ein zunächst zentraler Ansatz nach heftiger Kritik verworfen wurde

Die Corona-Warn-App: Idee (vereinfacht)

- Smartphones messen Abstand zu anderen Smartphones mittels Bluetooth
- unterschreiten zwei Smartphones für einen festlegten Zeitraum eine Distanz, zeichnen beide Smartphones dies als Begegnung auf
- es gibt dabei keine persistenen, personenenbezogenen Schlüssel
- jedes Smartphone erzeugt alle 15 Minuten einen neuen pseudo-zufälligen Schlüssel, nur diese werden ausgetauscht
- jeder Nutzer, der positiv getestet wurde, meldet dies an einen zentralene Server und sendet seine Schlüssel der letzten zwei Wochen an den Server
- alle Nutzer laden regelmäßig die Schlüssel aller positiv getesteten herunter und schneiden sie **lokal** gegen die eigenen Schlüssel zur Risikoermittlung (Details)

Das Prinzip der Corona-Warn-App



https://github.com/BigDataAnalyticsGroup/bigdataengineering/blob/master/CoronaWarnAppPrinciple.ipynb

Technische Ansätze 4/6

Differential Privacy, Verrauschen der Daten

kleinen Fehler zu den Daten hinzuaddieren, so dass Analysen nur teilweise verfälscht werden; Analysen funktionieren dann (hoffentlich) nur noch für Gruppen von Tupeln (k-Anonymity) aber nicht mehr zur Identifikation einzelner Tupel

Problem: wie stark verrauschen, um Analysen nicht zu verfälschen. Ist das wirklich sicher? (siehe Security-Vorlesungen)

Ändern/Manipulieren der Daten

Daten so ändern, dass Anfragen bestimmte Ergebnisse nicht (mehr) anzeigen

Problem: eigentlich technisch leicht zu entdecken (auch im Buch)

Technische Ansätze 5/6

Anti-Hardware-Administration (Hardware manipulieren/zerstören/verlangsamen)

„der Server ist leider schon wieder kaputt, ich weiß auch nicht, woran das liegt!“

Beispiele:

- im Buch gibt es immer das eine „Datensilo“, das langsam ist
- Stromspitzen
- Wasserschäden
- ...

Problem: mit etwas Nachdenken auch leicht zu entdecken...

Technische Ansätze 6/6

Anti-Software-Engineering (Software manipulieren/zerstören/verlangsamen)

„da kann man nichts machen, das dauert halt so lange!“:

Beispiele:

- unwartbaren Spaghetti-Code produzieren (machen die meisten sowieso von ganz alleine...)
- komplexe Anfragen schreiben, die den Anfrageoptimierer überfordern
- falsches Datenbanksystem benutzen
- Datenbanksystem falsch konfigurieren
- NoSQL, JSON oder XML nutzen
- Code schreiben, der quadratische oder schlimmere Laufzeit hat: „Ein Join hat leider immer quadratische Komplexität.“
- Code garnicht oder falsch dokumentieren
- schwer zu findende Fehler einbauen (concurrency is your friend)

Soziale Ansätze

- „diese Anfrage ist komplizierter als gedacht, ich brauche mehr Zeit“
  (im Buch genutzt)
- „das geht nicht so einfach, wir sollten noch Aspekt XY berücksichtigen...“
  (im Buch genutzt)
- „wir haben nicht die richtige Software/Hardware dafür, wir müssen erst XY kaufen und Schulungen organisieren...“
  (XY ist aber das falsche Werkzeug)
- „dafür sollten wir Herrn X hinzuziehen, der ist Experte auf diesem Gebiet...“
  (in Wahrheit ist Herr X ein Idiot und bremst das Projekt)
- „wir sollten das an die Abteilung Y delegieren...“
  (die haben in der Vergangenheit auch nichts hinbekommen...)
- „das Projekt ist zu groß, wir sollten es aufteilen in Teilprojekte...“
  (so dass hinterher niemand mehr den Überblick hat und wir in sinnlosen Meetings Zeit verlieren...)

Rechtliche Ansätze

Joins verhindern, Zugriff auf mehrere Datenquellen erschweren, keine Datenkartelle

Grundsätzlich:

- Zugriff auf „sensible“ Datenquellen stark beschränken
- verhindern, dass bestimmte Arten von Datenquellen zusammengebracht werden, vgl. Technische Ansätze 3/6

Problem:

Was sind denn sensible Datenquellen und was nicht?

- Sind zeitliche Daten sensibel?
  Sehr wahrscheinlich.
- Sind räumliche Daten sensibel?
  Sehr wahrscheinlich.
- und andere Daten weniger?
  Hmmmm.

Der Kriminalfall

Die Ermittlung

Am 27.4.2019 wurde in der Stadt X eine Straftat verübt. Ein Zeuge hat jemanden beobachtet, für den ein Phantombild angefertigt wurde. Eine Anfrage des Phantombildes an eine Bilddatenbank zeigt Herrn M als möglichen Kandidaten. Herr M ist in X gemeldet. Aber war Herr M am 27.4. überhaupt in X?

- Die GPS-Daten von seinem (inzwischen abgehörten) Handy sagen nein.
- Ms Stimme wurde von einem anderen Handy aufgezeichnet, dieses Handy befand sich 200 km von der Stadt X entfernt.
- Der Strom- und Wasserzähler von Ms Wohnung in X zeigen eindeutig, dass morgens bis 10 Uhr und abends ab 18 Uhr jemand zuhause war. Die Stromprofile sehen genauso aus wie für fast jeden anderen Tag.
- War M es?

Kommissar Equi-Join beginnt mit der Ermittlung

Was wissen wir über Herrn M, das sich als Join-Prädikat eignet?
Mit anderen Worten: wie können wir die Informationen zu Herrn M anreichern?

Herr M sammelt Turnschuhe

Herr M pflegt eine private kleine Relation, in der alle seine Turnschuhe verzeichnet sind.

[Turnschuhe] : {[
    ID:int,
    Modell:str,
    Größe:float
]}

Hinweis:

Es gibt in diesem Beispiel keine Zeit und/oder Ortsattribute.

Was soll da schon groß passieren? Das ist völlig harmlos, oder?
„Herr M hat nichts zu verbergen.“

Ein Kaufhaus

[Schuhe] : {[
    ID:int,
    Modell:str,
    Größe:float
]}

[Verkäufe] : {[
    Schuhe_ID:(Schuhe→ID),
    Menge:int,
    Wann:timestamp
]}

Das ist nur ein kleiner vereinfachter Ausschnitt des Schemas. Natürlich hat das Kaufhaus Infos zu allen verkauften Artikeln.

Das Sneakers-Szenario in Python

Tracking wihout space and time attributes

This notebook showcases an example where a personal database does not contain time or space attributes. Still that data may be critical when combined with other data sources that have spatial and/or temporal attributes.

Define the schema and load the data of Mr. M's personal database:

In [1]:
```
-- personal sneakers "database" (in fact just one tiny relation) of Mr. M:
PRAGMA foreign_keys = OFF;

DROP TABLE IF EXISTS sneakers;

PRAGMA foreign_keys = ON;

CREATE TABLE sneakers (
    id INTEGER PRIMARY KEY,
    label TEXT,
    size FLOAT
);
```

In [2]:
```
-- load csv data into tables:
-- enable csv mode:
.mode csv

-- import the necessary files:
.import data/sneakers/MrM_sneakers_no_header.csv sneakers

-- enable pretty formatting:
```

https://github.com/BigDataAnalyticsGroup/bigdataengineering/blob/master/Sneakers.ipynb

NSA (Teil 2)

Frage 2

... und welche ethischen Probleme entstehen durch diese Anfragen? Wie gehen wir damit um?

1. Es gibt zahlreiche mögliche Gegenmaßnahmen sowohl rechtlich als auch im Rahmen des zivilen Ungehorsams.
2. Grundsätzlich sollte das gemeinsame Auswerten von mehreren Datenquellen sehr viel stärker reglementiert werden.
3. Die Überwachung durch Metadaten — die auch oft aufbereitete Video-/Audio etc Daten sind — kann überraschende Zusammenhänge offenlegen.
4. Diese Effekte können im Negativen (siehe Beispiele oben) als auch im Positiven (z.B. Datenjournalismus, Panama-Papers, Sneakers-Beispiel, Freispruch durch Daten) genutzt werden.
5. Die Grundproblematik ist nur sehr schwierig zu lösen: was, wenn Daten in falsche Hände geraten?

Ausblick auf nächste Woche: Anfrageoptimierung

nächste Woche:

Frage 3

... und was ist, wenn die Daten größer werden? Wie kommen wir eigentlich von SQL zu einem effizienten Programm?

Weiterführendes Material

- siehe diverse Links in 03 NSA (Teil 1)
- Edward Snowden. Permanent Record: Meine Geschichte. amz
- Glenn Greenwald. Die globale Überwachung: Der Fall Snowden, die amerikanischen Geheimdienste und die Folge amz
- Josef Foschepoth. Überwachtes Deutschland: Post- und Telefonüberwachung in der alten Bundesrepublik. amz
- Tom Hillenbrand. Des Königs NSA: 1684 statt 1984. tomhillenbrand.de, youtube

Literaturverzeichnis Allgemein:
`https://www.infomath-bib.de/tmp/vorlesungen/info-basic_big-data-engineering.html`