NSA (Teil 1)
SQL

VL Big Data Engineering
(vormals Informationssysteme)

Prof. Dr. Jens Dittrich

bigdata.uni-saarland.de

5. Mai 2022

NSA (Teil 1)

Geplante Struktur für jeweils zwei Wochen Vorlesung:

1. Konkrete Anwendung: NSA
2. Was sind die Datenmanagement und -analyseprobleme dahinter?
3. Grundlagen, um diese Probleme lösen zu können
   (a) Folien
   (b) Jupyter/Python/SQL Hands-on
4. Transfer der Grundlagen auf die konkrete Anwendung

NSA (Teil 1)

1. Konkrete Anwendung: die NSA

- Snowden, Spionageaffäre, kurze Einführung, Links zum Weiterlesen

NSA: National Security Agency

- größter Auslandsgeheimdienst der USA
- 1945 von Truman gegründet
- ca. 40.000 Mitarbeiter
- Budget: 10,8 Milliarden US-Dollar (geschätzt, genaue Angaben geheim)
- https://de.wikipedia.org/wiki/National_Security_Agency

GCHQ, BND, ...

- vergleichbare „Dienste“ existieren in anderen Ländern, z.B.:
- Government Communications Headquarters (GCHQ) in Großbritannien
  - ca. 5.000 Mitarbeiter
  - ca. 2,6 Milliarden Pfund Budget
  - https://de.wikipedia.org/wiki/Government_Communications_Headquarters
- BND (Bundesnachrichtendienst) in Deutschland
  - ca. 6.000 Mitarbeiter
  - ca. 1 Milliarde Euro Budget
  - https://de.wikipedia.org/wiki/Bundesnachrichtendienst
  - “Internetüberwachung des BND ist in heutiger Form verfassungswidrig” (Urteil vom 19.5.2020): SPON BvG
- Stasi (Ministerium für Staatssicherheit) in der DDR
  - https://de.wikipedia.org/wiki/Ministerium_f%C3%BCr_Staatssicherheit

Aufgaben (u.a.)

- Überwachung und Dechiffrierung der weltweiten Kommunikation
- Wirtschaftsspionage
- frühzeitiges Erkennen von Gefahrenlagen
  (wie auch immer diese im Einzelfall definiert sind)
- Teil der Kriegsführung
- in den USA ist die NSA unter der Aufsicht des Verteidigungsministeriums
- Motivation: Entschlüsselung von Enigma im 2. Weltkrieg durch Alan Turing

Was genau machen die?

Was die Dienste genau machen, wird geheim gehalten.

Die Spionageaffären

- seit Bestehen von Geheimdiensten gab es immer wieder Whistleblower (Dt.: Enthüller, Aufdecker, Informant)
- der bekannteste ist Edward Snowden, der bis Mai 2013 als Sysadmin bei der NSA arbeitete (Verfilmung: Snowden, Doku: Citizenfour)
- ab Juni 2013 hat er angefangen, nach und nach geheime Dokumente der NSA zu veröffentlichen, die die Massenüberwachung durch die Geheimdienste dokumentieren
- andere wichtige Whistleblower waren Martin und Mitchell, William Binney, Russ Tice, Thomas Tamm,Thomas Drake, Katharine Gun (Verfilmung: Official Secrets), Julian Assange, Chelsea Manning, Reality Leigh Winner
- Whistleblower auf Wikipedia

Laura Poitras / Praxis Films

Was überwacht wird (stark verkürzt)

Einfach alles!:

- Telefongespräche: Audioaufzeichnung der letzten 30 Tage, weltweit!
- Sie wollen wissen, was Person X mit Person Y vor drei Wochen am Telefon besprochen hat, kein Problem!
- E-Mails, Chats, Bulletin-Boards
- Clouddienste
- ...
- siehe: NSA Files, The Guardian
- siehe: Globale Überwachungs- und Spionageaffäre, Wikipedia
- Video von JD dazu: Big Data is Watching You! But who is watching Big Data? (oder: Warum Daten wie Uran sind.)

Grundgesetz Artikel 10

Dieser Artikel verbürgt das Brief-, das Post- sowie das Fernmeldegeheimnis.
Dieser Artikel wird durch die Überwachung faktisch außer Kraft gesetzt.

Metadaten

Metadaten

Metadaten sind Daten, die Merkmale über andere Daten enthalten.

Beispiele:

- wer hat wann mit wem telefoniert (nicht den Inhalt des Gesprächs)
- wer hat wann welches E-Book gekauft (nicht den Inhalt des Buches)
- wer hat wann welchen Song gehört/Film gesehen (nicht den Inhalt des Songs, des Films)
- wer hat wann was gekauft
- ...

"We kill people based on metadata."

[Michael Hayden, former NSA-director], Quelle: Heise/Youtube

NSA: Nationales Sicherheitsamt, Andreas Eschbach

Buchidee

Stellen Sie sich vor, die Computertechnologie hätte sich 70 Jahre eher entwickelt. In der Weimarer Republik gab es bereits Komputer, das Weltnetz und später mobile Volkstelefone. Und umfangreiche Datensammlungen. Dieser Datenschatz fällt bei der Machtergreifung den Nazis in die Hände. Was hätte dies für Auswirkungen gehabt?

- brilliante Buchidee
- Datenanalyse wird in dem Buch zu 95% technisch korrekt beschrieben bis hin zu Beispielen in „Strukturierter Abfrage-Sprache“
- Link zum Buch

„Die eigentliche Macht liegt in der Möglichkeit, für sich genommen scheinbar harmlose Daten mithilfe des Komputers auf eine Weise zu verknüpfen, die zu ungeahnten Einsichten führt.“
[aus dem Buch, Adamek (Leiter der NSA) zu Himmler]

Strukturierte Abfrage-Sprache (SAS) aus dem Buch

```
SELEKTIERE AUS Einwohner
ALLE ( Vorname, Name, Straße, Ort, GebDat )
FÜR (
GebDat:Jahr >= 1913
UND
GebDat:Jahr <= 1917
UND
GebOrt = »Berlin«
UND
Vorname = »Cäcilia« )
```

Dann drückte sie eine Taste, und der Text verschwand wieder. Auf dem Schirm erschien die Nachricht: *SAS – Ausführung läuft*.

»Was heißt SAS?«, fragte Lettke mit dem unguten Gefühl, an Dinge zu rühren, die ihn nichts angingen.

»Das ist die Abkürzung für ›Strukturierte Abfrage-Sprache‹«, sagte sie und sah

NSA (Teil 1)

2. Was sind die Datenmanagement und -analyseprobleme dahinter?

heute:

Frage 1

Wie stellen wir so komplexe Anfragen?

nächste Woche:

Frage 2

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

NSA (Teil 1)

3. Grundlagen, um diese Probleme lösen zu können
   (a) Folien
   (b) Jupyter/Python/SQL Hands-on

- SQL (Structured Query Language), im Buch: Strukturierte Anfragesprache

SQL

Kernidee von SQL (Structured Query Language)

SQL ist eine Daten**transformations**sprache. D.h. eine Menge von Eingaberelationen wird auf sehr vielfältige Weise in eine Ausgaberelation transformiert.

- deklarativ: wir beschreiben mit SQL **WAS** das Ergebnis ist aber **nicht, WIE** es berechnet werden soll
- sehr mächtig, Turing Complete (mit Tricks)
- verschiedene "Standards": SQL 92, 99, 2016, 2019, ...
- prozedurale Erweiterungen
- Erweiterungen für andere Datenmodelle: JSON, Objekte, etc.
- Anbindung/Treiber für nahezu jede Programmiersprache

Häufigste Fehler im Umgang mit SQL 1/3

„SQL ist eine Sprache für das Schreiben und Lesen einzelner Tupel.“

⇒ „Ich nehme SQL hauptsächlich für das Lesen und Schreiben einzelner Tupel: CRUD (Create, Read, Update, Delete). D.h. eine Art tupelartiges Dateisystem“.

Das ist ungefähr so, als würde ich eine komplette Fabrikproduktionsstraße nur als Flaschenöffner benutzen.

- Die wahren Stärken von SQL bleiben so ungenutzt.
- Funktionalität, die eigentlich in SQL vorhanden ist, wird nachimplementiert, mit allen (versteckten) Kosten: Qualitätssicherung, Testen, Bug Fixes, ...

Häufigste Fehler im Umgang mit SQL 2/3

„SQL und insbesondere Joins sind langsam.“

⇒ „Ich nehme lieber NoSQL, Hadoop oder implementiere es selbst“.

- Bitte geben Sie Ihren Bachelor bei unserer Studienkoordinatorin zurück.
- SQL und die Performance von SQL-Statements sind **zwei verschiedene Dimensionen**
- die Performance von SQL hängt von sehr vielen Faktoren ab, aber nicht von prinzipiellen Limitierungen von SQL!
- die wichtigsten Einflussfaktoren: Indexe, Art der (Anfrage-)Optimierung von SQL, Physisches Design
- dazu später und in der Stammvorlesung mehr

Häufigste Fehler im Umgang mit SQL 3/3

„SQL kann nicht mit stärker strukturierten Daten wie JSON, Objekten, Graphen umgehen“

⇒ „Ich nehme lieber einen Key/Value-Store“.

- bereits für SQL 1999 wurde das relationale Modell erweitert
- Grundidee: Domänen können beliebigen Typs (insbesondere strukturiert!) sein und nicht nur „atomare Typen“
- rich datatypes: arrays, nested tables, composite types, ..
- SQL 2016: JSON
- gutes Übersichtsvideo hierzu: Markus Winand, The Mother of all Query Languages: SQL in Modern Times

Seit SQL-92 ist sehr viel passiert...

Aber in vielen Projekten wird nur SQL-92 oder wenig mehr benutzt.
D.h. es wird oft viel Potential und Geld verschwendet.

Grundstruktur von SQL-92-Anfragen

```
SELECT [DISTINCT] <Liste von Attributen>
FROM              <Liste von Tabellen>
WHERE             <Bedingung>
```

Hierbei entspricht das FROM dem relationalen Kreuzprodukt über die Liste von Tabellen, WHERE entspricht der relationalen Selektion mit Hilfe des Prädikats und SELECT entspricht der relationalen Projektion auf die Liste von Spalten.

Achtung:

Wird SELECT ohne DISTINCT angegeben, werden keine Duplikate entfernt. Die Ergebnisrelation ist dann nicht unbedingt eine Menge (wie im relationalen Modell).
Wenn wir alle Duplikate im Ergebnis entfernen wollen, müssen wir zusätzlich DISTINCT angeben.

Konzeptuelle Ausführungsreihenfolge

```
SELECT <Liste von Attributen>      3. Projektion zu Liste von Attributen
FROM   <Liste von Tabellen>        1. Kreuzprodukt über alle Tabellen
WHERE  <Bedingung>                 2. Selektion mit Bedingung
```

Ein SQL-92-Statement kann **konzeptuell** so gelesen werden, dass zuerst das FROM ausgeführt wird, dann das WHERE und dann das SELECT. Das Datenbanksystem muss die Schritte **nicht** in dieser Reihenfolge ausführen. Das Ergebnis der Anfrage muss aber in jedem Fall identisch sein zur dieser konzeptuellen Reihenfolge.

Somit entspricht:

```
SELECT A1,...,An
FROM   T1,...,Tm
WHERE  P
```

in relationaler Algebra dem Ausdruck $\pi_{A1,\ldots,An}\Big(\sigma_P\big(T1 \times \ldots \times Tm\big)\Big)$.

Achtung

Bitte nicht das SELECT aus SQL mit der Selektion σ der relationalen Algebra verwechseln!

SQL im Jupyter Notebook

In [1]:
```
-- CSV-Modus einschalten:
.mode csv
-- ein paar CSV-Dateien als Tabellen importieren:
.import data/photodb/mitarbeiter.csv mitarbeiter
.import data/photodb/personen.csv personen
.import data/photodb/seniors.csv seniors
.import data/photodb/verkaeufer.csv verkaeufer
.import data/photodb/fotographen.csv fotographen
```

In [2]:
```
-- Tabellen bei der Ausgabe hübscher formatieren:
.mode columns
.headers on
```

In [3]:
```
-- ganze Tabelle anzeigen lassen:
SELECT *
FROM mitarbeiter;
```

personid	gehalt	erfahrung
1	45000	3
2	37000	3
3	50000	2
4	60000	3
5	55000	2
6	15000	1
7	50000	2

In [4]:
```
-- ganze Tabelle anzeigen lassen:
SELECT *
FROM seniors;
```

mitarbeiterid	anzahlgrauehaare	bonus
1	45	34000
2	457	40000

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

Joins in SQL

Grundsätzlich kann der Verbund (im Folgenden mit dem englischen **Join** benannt; das deutsche Wort wird von fast niemandem benutzt) auf zwei Arten spezifiziert werden.

Impliziter Join:

```
SELECT          A1,...,An
FROM            T1,...,Tm
WHERE           P
```

$$\pi_{A1,\ldots,An}\big(\sigma_{P}(T1 \times \ldots \times Tm)\big)$$

Expliziter Join:

```
SELECT          A1,...,An
FROM            <T1> JOIN <T2> ON P
```

$$\pi_{A1,\ldots,An}(T1 \bowtie_{P} T2)$$

Beispiele

Impliziter Join:

```
SELECT          *
FROM            mitarbeiter m, seniors s
WHERE           m.personid = s.mitarbeiterid
```

$\sigma_{\text{personid} = \text{mitarbeiterid}}(\text{mitarbeiter} \times \text{seniors})$

Expliziter Join:

```
SELECT          *
FROM            mitarbeiter m JOIN seniors s
                ON m.personid = s.mitarbeiterid
```

$\text{mitarbeiter} \bowtie_{\text{personid} = \text{mitarbeiterid}} \text{seniors}$

Achtung

Die Art der Formulierung des Joins spielt für die Effizienz der Anfrageverarbeitung in den allermeisten Datenbanksystemen **keine** Rolle.

Konzeptuelle Ausführungsreihenfolge bei Gruppierung

```
SELECT   [A], [Aggregatfunktionen F]     5. Aggregation und Projektion
FROM     <Tabellen>                      1. Kreuzprodukt über alle Tabellen
WHERE    <Bedingung P1>                  2. Selektion von Tupeln mit Bedingung P1
GROUP BY [B]                             3. Gruppierung
HAVING   <Bedingung P2 an Aggfkt. H>     4. Selektion von Gruppen mit Bedingung P2
```

SELECT $A_1, \ldots, A_n, \underbrace{f_1([G_1]), \ldots, f_{k_F}([G_{k_F}])}_{=:F}$

FROM $T_1, \ldots, T_m$

WHERE P_1

GROUP BY $B_1, \ldots, B_l$

HAVING P_2 mit Bedingungen an $\underbrace{h_1([G1]), \ldots, h_{k_H}([G_{k_H}])}_{=:H}$

Regeln

1. $[Q] = [T_1] \circ \ldots \circ [T_m]$ (gemeinsames Schema aller vorkommenden Attribute der Anfrage)
2. $[B] \subseteq [Q]$. ($[B]$ ist eine möglicherweise leere Teilmengen von $[Q]$)
3. $[A] \subseteq [B]$. ($[A]$ ist eine möglicherweise leere Teilmengen von $[B]$)
4. $[G_1], \ldots, [G_{k_G}], [H_1], \ldots, [H_{k_H}] \subseteq [Q]$. Falls $[G_i]$ oder $[H_i]$ leer sind, wird dies durch '*' signalisiert.
5. P_2 darf Bedingungen mittels Aggregatfunktionen H formulieren: auch solche, die nicht im SELECT stehen!

HAVING in Relationaler Algebra

$$from := T_1 \times \ldots \times T_m$$

$$where := \sigma_{P_1}(from)$$

$$groupby := \gamma_{[B];\ F \cup H}(where)$$

$$having := \sigma_{P_2}(groupby)$$

$$select := \pi_{A;F}(having)$$

Alias (aka Sicht, View)

Mittels <Bezeichner> := <Ausdruck in relationaler Algebra> können wir Ausdrücke in relationaler Algebra beliebig ausrollen und somit übersichtlicher schreiben

HAVING vs WHERE

Bitte nicht WHERE mit HAVING verwechseln! WHERE ist eine Bedingung auf Tupeln, HAVING eine Bedingung auf Gruppen.

Beispiel

```
SELECT      gehalt, count(*)
FROM        mitarbeiter
GROUP BY    erfahrung
```

(1) Gruppierung:

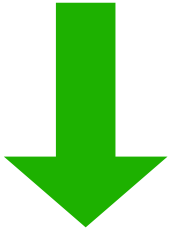
	personid integer	gehalt numeric	erfahrung integer	Gruppe
1	1	45000	3	3
2	2	37000	3	3
3	3	50000	2	2
4	4	60000	3	3
5	5	55000	2	2
6	6	15000	1	1
7	7	50000	2	2

(2) Durch das group by entstehen drei horizontale Partitionen (aka Gruppen):

1

	personid integer	gehalt numeric	erfahrung integer
6	6	15000	1

Gehalt ist eindeutig innerhalb dieser Gruppe



(3) 15000 ausgeben

2

	personid integer	gehalt numeric	erfahrung integer
3	3	50000	2
5	5	55000	2
7	7	50000	2

Gehalt ist **nicht** eindeutig innerhalb dieser Gruppe

was ausgeben?

3

	personid integer	gehalt numeric	erfahrung integer
1	1	45000	3
2	2	37000	3
4	4	60000	3

Gehalt ist **nicht** eindeutig innerhalb dieser Gruppe

was ausgeben?

Welche Gruppierungsattribute sind erlaubt?

Demnach ist das folgende SQL-Statement nicht erlaubt:

```
SELECT      gehalt, count(*)
FROM        mitarbeiter
GROUP BY    erfahrung
```

Warum?
"gehalt" steht nicht im GROUP BY.
Dann darf es nicht im SELECT stehen!

Was ist mit folgender Anfrage?

```
SELECT      gehalt, count(*)
FROM        mitarbeiter
GROUP BY    personid
```

Erlaubt oder nicht?

Beispiel

```
SELECT      gehalt, count(*)
FROM        mitarbeiter
GROUP BY    personid
```

(1) Gruppierung:

	personid integer	gehalt numeric	erfahrung integer	Gruppe
1	1	45000	3	1
2	2	37000	3	2
3	3	50000	2	3
4	4	60000	3	4
5	5	55000	2	5
6	6	15000	1	6
7	7	50000	2	7

(2) Durch das group by entstehen sieben horizontale Partitionen (aka Gruppen):

1

	personid integer	gehalt numeric	erfahrung integer
1	1	45000	3

2

	personid integer	gehalt numeric	erfahrung integer
2	2	37000	3

3

	personid integer	gehalt numeric	erfahrung integer
3	3	50000	2

4

	personid integer	gehalt numeric	erfahrung integer
4	4	60000	3

5

	personid integer	gehalt numeric	erfahrung integer
5	5	55000	2

6

	personid integer	gehalt numeric	erfahrung integer
6	6	15000	1

7

	personid integer	gehalt numeric	erfahrung integer
7	7	50000	2

Gehalt ist eindeutig innerhalb jeder Gruppe

(3) eindeutiges Gehalt ausgeben

Welche Gruppierungsattribute sind erlaubt?

Das ist erlaubt, da durch die Gruppierung über den Schlüssel garantiert ist, dass in jeder Gruppe nur ein Tupel ist. Dadurch sind alle anderen Attributwerte eindeutig.

Regel für Gruppierungsattribute in SQL

Attribute, die nicht im GROUP BY stehen, dürfen **nicht** ohne Aggregation im SELECT verwendet werden!

Außer es wird über den Schlüssel gruppiert, dann dürfen alle Attribute im SELECT verwendet werden.

Anfrageoptimierer

- ein Anfrageoptimierer übersetzt SQL in ein ausführbares Programm
- ähnlich wie Übersetzung von C++ zu Binärcode, hier: SQL zu Binärcode
- Anfrageoptimierer versucht bestmögliches (schnellstes) Programm zu finden
- aber: die Übersetzung von SQL ist viel domainspezifischer und deklarativ
- größte Herausforderungen hierbei:
  - richtige Joinreihenfolge
  - welche Datenstrukturen (genannt 'Indexe') benutzen?
  - welche Algorithmen benutzen?
  - Ausführungskosten sinnvoll schätzen
  - Hardware (CPUs und Speicherhierarchie) gut nutzen

Die Qualität eines Datenbanksystems wird wesentlich durch die Qualität seines Datenbankoptimierers bestimmt.

Dazu später mehr.

NSA (Teil 1)

und damit zurück zu:

2. Was sind die Datenmanagement und -analyseprobleme dahinter?

Frage 1

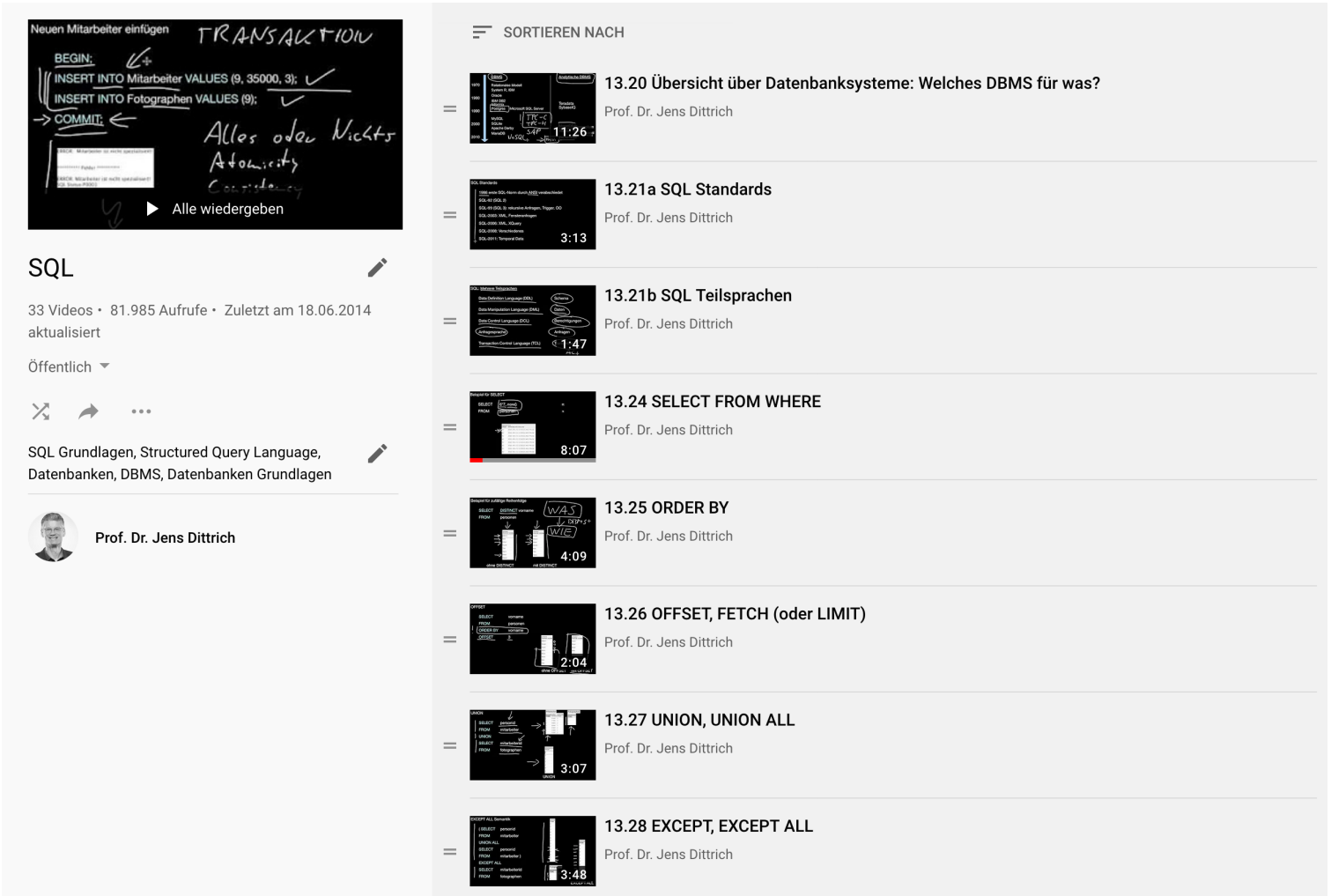
Wie stellen wir so komplexe Anfragen?

Mit SQL!

Ausblick auf nächste Woche

Komplexeres SQL,
Szenario aus dem NSA-Buch,
weitere Beispiele,
Gegenmaßnahmen

Weiterführendes Material



Youtube Videos von Prof. Dittrich zu SQL

sowie Kapitel in Kemper&Eickler