NSA (Part 1)
SQL

Big Data Engineering
(formerly Informationssysteme)

Prof. Dr. Jens Dittrich

bigdata.uni-saarland.de

May 8, 2023

NSA (Part 1)

Planned structure for each two-week lecture:

1. Concrete application: NSA
2. What are the data management and analysis issues behind this?
3. Basics to be able to solve these problems
   - (a) Slides
   - (b) Jupyter/Python/SQL Hands-on
4. Transfer of the basics to the concrete application

NSA (Part 1)

1. Concrete application: the NSA

- short introduction
- Snowden
- global surveillance disclosures
- links for further reading

NSA: National Security Agency

- largest foreign intelligence agency in the US
- founded by Truman in 1945
- approx. 30,000–40,000 employees
- budget: 10.8 billion US dollars (estimated in 2013, exact numbers are a secret)
- https://en.wikipedia.org/wiki/National_Security_Agency

GCHQ, BND, ...

- comparable ‘agencies’ exist in other countries, e.g.:
- Government Communications Headquarters (GCHQ) in the United Kingdom
    - approx. 5,000 employees
    - ca. £2.6 billion budget
    - https://en.wikipedia.org/wiki/Government_Communications_Headquarters
- BND (Federal Intelligence Service) in Germany
    - approx. 6,000 employees
    - ca. 1 billion Euro budget
    - https://en.wikipedia.org/wiki/Bundesnachrichtendienst
    - “Internet surveillance of the BND is unconstitutional in its current form” (judgement of 19.5.2020): SPON BvG
- Stasi (Ministry for State Security) in the GDR
    - https://en.wikipedia.org/wiki/Ministerium_f%C3%BCr_Staatssicherheit

Tasks (among others)

- monitoring and deciphering of global communications
- industrial espionage
- early detection of potentially dangerous situations (however these are defined in individual cases)
- part of warfare
- in the USA, the NSA is under the supervision of the Department of Defence
- motivation: decoding of Enigma in the 2nd World War by Alan Turing

What exactly do they do?

What exactly the services do is kept secret.

Whistleblowers related to Mass Surveillance

- since the existence of secret services, there have always been whistleblowers.
- the best known is Edward Snowden, who worked as a sysadmin at the NSA until May 2013 (Film adaptation: Snowden, documentary: Citizenfour)
- from June 2013, he began to gradually publish secret NSA documents documenting mass surveillance by the intelligence services
- other important whistleblowers were Martin and Mitchell, William Binney, Russ Tice, Thomas Tamm,Thomas Drake, Katharine Gun (movie: Official Secrets), Julian Assange, Chelsea Manning, Reality Leigh Winner
- whistleblower on Wikipedia

Laura Poitras / Praxis Films

What is surveilled (short version)?

Simply everything!:

- phone calls: Audio recording of the last 30 days, worldwide!
- you want to know what person X discussed with person Y on the phone three weeks ago, no problem!
- E-mails, chats, bulletin boards
- cloud services
- ...
- see: NSA Files, The Guardian
- see: Global surveillance disclosures (2013–present), Wikipedia
- video from JD about it: Big Data is Watching You! But who is watching Big Data? (oder: Warum Daten wie Uran sind.), in German

Basic Law Article 10

This basic law is completely undermined by mass surveillance.

Metadata

Metadata

Metadata is data about other data.

Examples:

- who spoke to whom on the phone and when (not the actual content of the conversation)
- who bought which e-book and when (not the content of the book)
- who listened to which song/watched which film and when (not the content of the song, the film)
- who bought what and when
- ...

"We kill people based on metadata."

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

NSA: Nationales Sicherheitsamt, Andreas Eschbach

Book idea

Imagine if computer technology had developed 70 years earlier. In the Weimar Republic there were already computers, the ‘world network’ and later on mobile ‘people’s telephones’. And extensive data collections. This treasure trove of data fell into the hands of the Nazis when they seized power. What effects would this have had?

- the book idea is brilliant
- data analysis is described technically correct (except for minor technical things) and in detail up to examples in “Structured query language”
- link to the book (in German)

NSA: Nationales Sicherheitsamt, English Version

← **Tweet**

Jens Dittrich @jensdittrich · 22 Std.
@AndreasEschbach
Do you have plans to publish an English translation of your book "NSA"? (That would be great!)

1 3

Andreas Eschbach
@AndreasEschbach

An author's "plans" doesn't matter here; what would be needed is an English or American PUBLISHER who is willing to publish a translated version. So, should you happen to know one, don't hesitate ... 😇

Tweet übersetzen

6:26 nachm. · 2. Mai 2023 · **1** Mal angezeigt

Twittere deine Antwort

Antworten

Jens Dittrich @jensdittrich · 2 Min.
I see. Mabye some publisher already publishing books on mass surveillance could be interested? Your book is the best explanation on the risks of joining meta data out there...

Jens Dittrich @jensdittrich · 1 s
I am still using your book in my lecture "Big Data Engineering". This year it is in English. So, I have the problem that some students simply cannot read your book. Would it be possible to give my students access to like the first 50 pages translated by DeepL?

The Fundamental Problem with Data: The “Big Data”-Problem

“The real power lies in the possibility of linking seemingly innocuous data with the help of the computer in a way that leads to unexpected insights.”
[from the book, Adamek (Leiter der NSA) zu Himmler]

Structured Query Language (SAS) from the book

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 (Part 1)

2. What are the data management and analysis issues behind this?

today:

Question 1

How do we phrase more complex queries?

next week:

Question 2

... and what ethical problems arise from these kind of data collections?
How do we deal with them?

NSA (Part 1)

3. Basics to be able to solve these problems
   (a) Slides
   (b) Jupyter/Python/SQL Hands-on

- SQL (Structured Query Language), in Eschbach's book: "Strukturierte Anfragesprache"

SQL

Core idea of SQL (Structured Query Language)

SQL is a data **transformation** language. That is, a set of input relations is transformed into an output relation in a very diverse way.

- declarative: we describe with SQL **WHAT** the result is but **not HOW** it should be calculated
- very powerful, Turing Complete (with tricks)
- various "standards": SQL 92, 99, 2016, 2019, ...
- procedural extensions
- extensions for other data models: JSON, objects, etc.
- database connections/drivers for almost all programming languages

Most Common Mistakes when Dealing with SQL 1/3

"SQL is a language for writing and reading individual tuples."

⇒ "I use SQL mainly for reading and writing individual tuples: CRUD (Create, Read, Update, Delete), i.e. some sort of a tuple-like file system".

That's like using an entire factory production line just as a bottle opener.

- the true strengths of SQL thus remain unused.
- functionality that is actually available in SQL is re-implemented, with all the (hidden) costs: quality assurance, testing, bug fixes, ...
- a clear violation of the Laziness Principles (slide set 00)

Most Common Mistakes when Dealing with SQL 2/3

"SQL and especially joins are slow."

⇒ "I prefer to use NoSQL, Hadoop or implement it myself".

- SQL and the performance of a program generated from SQL are **two different dimensions**
- eventually, SQL is used to translate it into an executable program
- the performance of that program depends on many factors, but has **nothing** to do with claimed limitations of SQL!
- the most important influencing factors: Indexes, (query) optimisation algorithms, Physical design.
- more on this later in this and in the core lecture

Most Common Mistakes when Dealing with SQL 3/3

"SQL cannot handle more structured data such as JSON, objects, graphs"

⇒ "I prefer to use a key/value store".

- the relational model was already extended for SQL 1999
- basic idea: Domains can be of any type (especially structured!) and not just "atomic types"
- rich datatypes: arrays, nested tables, composite types, ..
- SQL 2016: JSON
- good overview video on this: Markus Winand, The Mother of all Query Languages: SQL in Modern Times

A lot has happened since SQL-92...

But in many projects only SQL-92 or little more is used. That means a lot of potential and money is often wasted.

Basic Structure of SQL-92 Queries

```
SELECT [DISTINCT] <List of attributes>
FROM              <List of tables>
WHERE             <Condition>
```

Here, the FROM corresponds to the relational cross product over the list of input tables, WHERE corresponds to the relational selection using the predicate and SELECT corresponds to the relational projection onto the list of columns.

Warning:

If SELECT is specified without DISTINCT, no duplicates are removed. Then, the result table is not necessarily a set (as in the relational model). If we want to remove all duplicates in the result, we must also specify DISTINCT.

Conceptual Execution Order

```
SELECT <List of attributes>     3. Projection to list of attributes
FROM   <List of tables>         1. Cross product over all tables
WHERE  <Condition>              2. Selection with condition
```

An SQL-92 statement can be read **conceptually** in such a way that the FROM is executed first, then the WHERE and then the SELECT. The database system does not have to execute these steps in this order. However, the result of the query must in any case be semantically identical to this conceptual order.

The SQL-statement:

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

corresponds to relational algebra expression $\pi_{\mathrm{A1},\ldots,\mathrm{An}}\Big(\sigma_{\mathrm{P}}\big(\mathrm{T1} \times \ldots \times \mathrm{Tm}\big)\Big)$.

Warning

Please do not confuse the SELECT from SQL with the selection operator σ of relational algebra!

SQL in Jupyter Notebook (using DuckDB)

SQL-92 Demo

This notebook introduces the basics of read-only SQL-92-queries.

In [1]:
```
import duckdb
```

Database Schema

All examples below are taken from the following scenario:

You are running a photo agency which has several types of employees: seniors, salespersons, and ph

Create schemas for all tables:

In [2]:
```
duckdb.sql("""
CREATE TABLE persons (
    id INTEGER PRIMARY KEY,
    lastname TEXT,
    firstname TEXT,
    birthday TEXT
);""")

duckdb.sql("""
CREATE TABLE employees (
    personId INTEGER PRIMARY KEY,
    salary INTEGER,
    experience INTEGER,
    FOREIGN KEY(personId) REFERENCES persons(id)
);""")

duckdb.sql("""
CREATE TABLE seniors (
    employeeId INTEGER PRIMARY KEY,
    numGreyHairs INTEGER,
    bonus INTEGER,
    FOREIGN KEY(employeeId) REFERENCES employees(personId)
);""")

duckdb.sql("""
CREATE TABLE salespersons (
    employeeId INTEGER PRIMARY KEY,
    areaOfExpertise TEXT,
    FOREIGN KEY(employeeId) REFERENCES employees(personId)
);""")

```

- Notebook: https://github.com/BigDataAnalyticsGroup/bigdataengineering/blob/master/SQL.ipynb
- DuckDB: https://duckdb.org

Joins in SQL

Basically, a join can be specified in two ways:

Implicit join:

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

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

Explicit join:

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

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

Examples

Implicit join:

```
SELECT          *
FROM            employees e, seniors s
WHERE           e.personid = s.employeeid
```

$\sigma_{\text{personid} = \text{employeeid}}(\text{employees} \times \text{seniors})$

Explicit Join:

```
SELECT          *
FROM            employees e JOIN seniors s
                ON e.personid = s.employeeid
```

$\text{employees} \bowtie_{\text{personid} = \text{employeeid}} \text{seniors}$

Attention

The way the join is formulated plays no role in the efficiency of query processing in the vast majority of database systems.

Conceptual Execution Order for Grouping

```
SELECT    [A], [Aggregate functions F]     5. Aggregation and projection
FROM      <Input tables>                   1. Cross product over all input tables
WHERE     <Condition P1>                   2. Selection of tuples with condition P1
GROUP BY  [B]                              3. Grouping
HAVING    <Condition P2 on agg-fct. H>     4. Selection of groups with condition P2
```

$$\begin{array}{ll}
\texttt{SELECT} & A_1,\ \ldots,\ A_n,\ \underbrace{f_1([G_1]),\ \ldots,\ f_{k_F}([G_{k_F}])}_{=:F} \\
\texttt{FROM} & T_1,\ \ldots,\ T_m \\
\texttt{WHERE} & P_1 \\
\texttt{GROUP BY} & B_1,\ \ldots,\ B_l \\
\texttt{HAVING} & P_2 \texttt{ with conditions on } \underbrace{h_1([G1]),\ \ldots,\ h_{k_H}([G_{k_H}])}_{=:H}
\end{array}$$

Rules

1. $[Q] = [T_1] \circ \ldots \circ [T_m]$ (common scheme of all occurring attributes of the request)
2. $[B] \subseteq [Q]$. ($[B]$ is a possibly empty subset of $[Q]$)
3. $[A] \subseteq [B]$. ($[A]$ is a possibly empty subset of $[B]$)
4. $[G_1], \ldots, [G_{k_G}], [H_1], \ldots, [H_{k_H}] \subseteq [Q]$. If $[G_i]$ or $[H_i]$ are empty, this is signalled by '*'..
5. P_2 may formulate conditions by means of aggregate functions H: even those that are not in the SELECT!

It must hold: $\{A1, \ldots, An\} \subseteq \{B1, \ldots, Bl\}$.

HAVING in Relational Algebra

$$
\begin{aligned}
\textit{from} &:= T_1 \times \ldots \times T_m \\
\textit{where} &:= \sigma_{P_1}(\textit{from}) \\
\textit{groupby} &:= \gamma_{[B];\; F \cup H}(\textit{where}) \\
\textit{having} &:= \sigma_{P_2}(\textit{groupby}) \\
\textit{select} &:= \pi_{A;F}(\textit{having})
\end{aligned}
$$

Alias (aka view)

Via <Identifier> := <Expression in relational algebra> we can arbitrarily roll out expressions in relational algebra and thus write them more clearly

HAVING vs WHERE

Please do not confuse WHERE with HAVING! WHERE is a condition on tuples, HAVING a condition on groups.

Example

```
SELECT      salary, count(*)
FROM        employees
GROUP BY    experience
```

(1) Grouping (aka horizontal partitioning):

	personid	salary	experience	group/HP
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) Grouping in Step 1 creates three HPs/groups (Which attribute value of 'salary' should be output?)

1

	personid	salary	experience
6	6	15000	1

all attribute values of 'salary' are the same in this group

(3) output 15000

2

	personid	salary	experience
3	3	50000	2
5	5	55000	2
7	7	50000	2

attribute values of 'salary' are **not** the same inside this group

what to output?

3

	personid	salary	experience
1	1	45000	3
2	2	37000	3
4	4	60000	3

attribute values of 'salary' are **not** the same inside this group

what to output?

Which Grouping Attributes are Allowed? (1/2)

Accordingly, the following SQL statement is not allowed:

```
SELECT        salary, count(*)
FROM          employees
GROUP BY      experience
```

Why?

Attribute “salary” is not mentioned in the GROUP BY-clause.
Therefore it must not be used in the SELECT-clause!

What about the following query?

```
SELECT        salary, count(*)
FROM          employees
GROUP BY      personid
```

Permitted or not?

Example

```
SELECT      salary, count(*)
FROM        employees
GROUP BY    personid
```

(1) Grouping (aka horizontal partitioning):

	personid	salary	experience	group/HP
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) Grouping in Step 1 creates seven HPs/groups (Which attribute value of 'salary' should be output?)

1

	personid	salary	experience
1	1	45000	3

2

	personid	salary	experience
2	2	37000	3

3

	personid	salary	experience
3	3	50000	2

4

	personid	salary	experience
4	4	60000	3

5

	personid	salary	experience
5	5	55000	2

6

	personid	salary	experience
6	6	15000	1

7

	personid	salary	experience
7	7	50000	2

in each HP/group all attribute values of 'salary' are the same

(3) output 'salary'

Which Grouping Attributes are Allowed? (2/2)

This is allowed (in some systems like PostgreSQL) because the grouping via the key guarantees that there is only one tuple in each group. This makes all other attribute values unique.

Rule for grouping attributes in SQL

Attributes not listed in GROUP BY must **not** be used in SELECT **without aggregation**!

Exception in some Systems: if we group on the key or any other functionally dependent attribute making sure that the attribute specified in SELECT is unique for all tuples in the HP/group.

Query Optimiser

- a query optimiser translates SQL into an executable programme
- similar to translation from C++ to binary code, here: SQL to binary code
- query optimiser tries to find best possible (fastest) programme
- but: the translation of SQL is much more domain-specific and declarative
- biggest challenges here:
  - correct join order
  - which data structures (called 'indexes') to use?
  - which algorithms to use?
  - need to estimate execution costs
  - make good use of hardware (CPUs and memory hierarchy)

The quality of its database optimiser has a huge influence on the performance of queries.

More about that later.

NSA (Part 1)

and thus back to:

2. What are the data management and analysis issues behind this?

Question 1

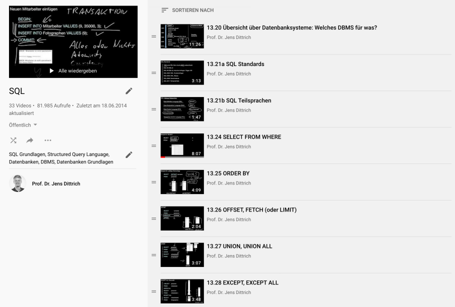
How do we make such complex requests?

With SQL!

Outlook to next week

More complex SQL,
Scenario from the NSA book,
ethical implications,
Countermeasures
other examples

Further material (in German and English)



Youtube Videos of Prof. Dittrich about SQL in German

- Chapter 4 in Kemper&Eickler in German
- Chapter 6 in Elmasri&Navathe in German or in English
- RelaX - relational algebra calculator contains the simplified IMDb schema used on the slides as well as the PhotoDB schema, allows you to query that data using either relational algebra or SQL
  - gist for IMDb_sample
  - gist for photodb