

Tag 3: Auf dem Weg zur datenbewussten Organisation

Session 9: Daten-Workflows und Open Government Data

Simon Munzert
Hertie School

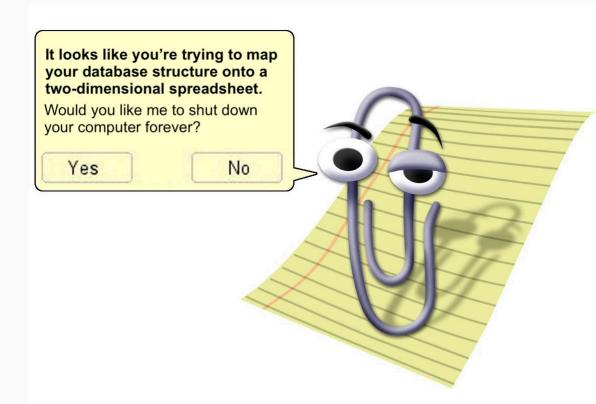
1. Relationale Datenstrukturen
2. Open Government Data
3. Daten-Workflow in der Behörde

Relationale Datenstrukturen

Die Allgegenwärtigkeit von relationalen Datenstrukturen

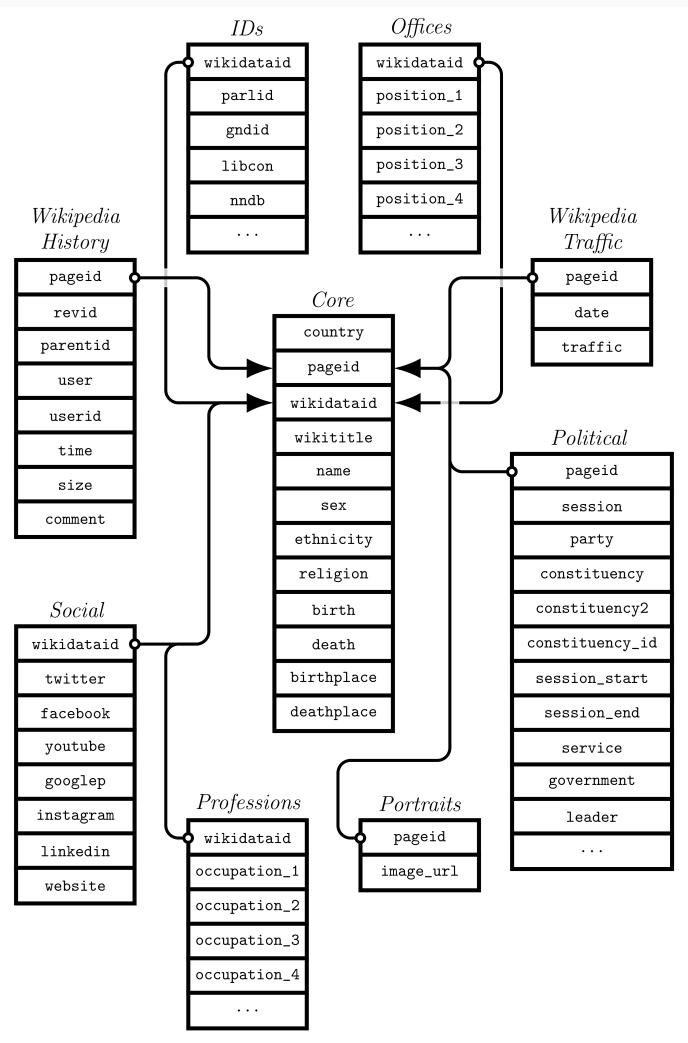
Von Tabellen...

- Wenn Sie an Daten auf Ihrem Laptop denken, haben Sie möglicherweise ein **Spreadsheet** im Kopf.
- In der Tat arbeitet ein Großteil der klassischen "statistischen" Software (SPSS, Stata, MS Excel) standardmäßig mit Spreadsheets als Dateninput.
- Gleichzeitig ist einer Ihrer ersten Gedanken möglicherweise auch **dateibasiert**. Die Daten werden in Dateien gespeichert, und diese Dateien werden von unserer Datenverwaltungssoftware gelesen (und erstellt).
- In vielen Fällen ist eine **zweidimensionale Datenstruktur** sinnvoll. Wir nutzen zum Beispiel
 - Befragte x Einstellungen
 - Bezirke x Merkmale
 - Beiträge in sozialen Medien x Textmerkmale



Die Allgegenwärtigkeit von mehrdimensionalen Datenstrukturen

Hertie School



... zu komplexen Datenstrukturen

- Je länger man jedoch darüber nachdenkt, desto problematischer wird es, Daten systematisch in zweidimensionalen Strukturen abzulegen.
- **Beispiele:**
 - Befragte x Einstellungen x Länder x Zeit
 - Bezirke x Unterbezirke Merkmale x Zeit
 - Beiträge in sozialen Medien x Retweets x Benutzer x Benutzereigenschaften x Netzwerkmerkmale x Metadaten
- Die Abbildung von dreidimensionalen auf zweidimensionale Strukturen ist technisch einfach machbar.
- Bei **vielfachen heterogenen Datenquellen** wird es unübersichtlich.
- Die Verwaltung komplexer Datenstrukturen ist nur ein Vorteil der Verwendung von **Datenbanken**.

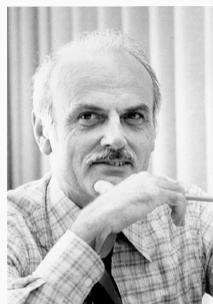
Größe und Struktur

- Sie haben **Datenmengen, die den Arbeitsspeicher** auf Ihrem Computer übersteigen. Datenbanken sind nur durch die verfügbare Festplattengröße begrenzt (oder können auf mehrere Festplatten/Rechner verteilt werden).
- Ihre **Datenstruktur ist komplex**. Datenbanken erlauben es, Daten mit komplexen Datenstrukturen zu speichern, abzurufen und zu untergliedern.
- Ihre Daten sind groß und Sie müssen **häufig darauf zugreifen/daraus filtern**. Die Abfrage von Datenbanken ist schnell.
- Sie legen Wert auf **Datenqualität** und haben klare Vorstellungen, wie Daten aussehen sollen. Mit Hilfe von Datenbanken können Sie spezifische Regeln für die Erweiterung und Aktualisierung Ihrer Datenbank definieren.

Zugänglichkeit und Gleichzeitigkeit

- Sie **arbeiten gemeinsam mit anderen** an einem Datenprojekt. Mit einer Datenbank haben Sie eine gemeinsame, zuverlässige Infrastruktur zur Verfügung, auf die mehrere Benutzer gleichzeitig zugreifen können.
- Wenn mehrere Parteien beteiligt sind, können Nutzerprivilegien wichtig sein (z. B. nur Lesezugriff, Zugriff auf Teile der Daten, eingeschränkte Verwaltungsrechte usw.). Die meisten Datenbanken erlauben die **Definition unterschiedlicher Nutzungsrechte für verschiedene Benutzer**.

- Das Konzept relationaler Datenbanken baut auf dem **relational model (RM)** auf, das 1969/1970 von **Edgar F. "Ted" Codd** vorgeschlagen wurde.
- Codd beschrieb das RM formal, führte es aber auch mit Konzepten ein, die auch heute noch verwendet werden (Normalisierung, Schlüssel, Joins, usw.).
- Die Grundannahme des relationalen Modells ist, dass alle Daten als Relationen (Tabellen) dargestellt werden können.
- Informationen werden dann durch in Beziehung stehende Datenwerte dargestellt.



Information Retrieval

P. BAXENDALE, Editor

A Relational Model of Data for Large Shared Data Banks

E. F. CODD
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. INTRODUCTION

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the “connection trap”).

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. DATA DEPENDENCIES IN PRESENT SYSTEMS

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed without logically impairing some application programs is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.2.1. *Ordering Dependence.* Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is

Credit Communications of the ACM 13(6), 1970

Speicherung von Daten in Tabellen

- Eine einzige Tabelle stellt bereits Beziehungen her:
Alle Daten in einer Zeile gehören zu demselben Datensatz.
- Wenn wir komplexere Beziehungen darstellen wollen (z. B. das Gewicht einer Person zweimal messen oder das Gewicht ihrer Kinder ebenfalls messen), können wir Daten von einer Tabelle zur anderen in Beziehung setzen.

Beispiel

- Wir haben Daten über Peter, Paul und Mary gesammelt.
- Wir haben Informationen über Geburtstage, Telefonnummern und Lieblingsspeisen.
- Wie können wir diese Informationen in Tabellen darstellen?

Table 7.1 Our friends' birthdays and favorite foods

nameid	name	birthday	favoritefood1	favoritefood2	favoritefood3
1	Peter Pascal	01/02/1991	spaghetti	hamburger	
2	Paul Panini	02/03/1992	fruit salad		
3	Mary Meyer	03/04/1993	chocolate	fish fingers	hamburger

Table 7.2 Our friends' telephone numbers

telephoneid	nameid	telephonenumber
1	1	001665443
2	2	002555555
3	1	001878345

- Wir beginnen mit der Darstellung der Daten in zwei Tabellen.
- Sie sind über den Schlüssel `nameid` verknüpft, so dass wir die vollständigen Namen nicht in die Tabelle der Telefonnummern aufnehmen müssen.
- Beachten Sie, dass wir hier eine 1:m (one-to-many) Beziehung haben, da Peter zwei Telefonnummern hat.

Speicherung von Daten in Tabellen

- Eine einzige Tabelle stellt bereits Beziehungen her:
Alle Daten in einer Zeile gehören zu demselben Datensatz.
- Wenn wir komplexere Beziehungen darstellen wollen (z. B. das Gewicht einer Person zweimal messen oder das Gewicht ihrer Kinder ebenfalls messen), können wir Daten von einer Tabelle zur anderen in Beziehung setzen.

Beispiel

- Wir haben Daten über Peter, Paul und Mary gesammelt.
- Wir haben Informationen über Geburtstage, Telefonnummern und Lieblingsspeisen.
- Wie können wir diese Informationen in Tabellen darstellen?

Table 7.1 Our friends' birthdays and favorite foods

nameid	name	birthday	favoritefood1	favoritefood2	favoritefood3
1	Peter Pascal	01/02/1991	spaghetti	hamburger	
2	Paul Panini	02/03/1992	fruit salad		
3	Mary Meyer	03/04/1993	chocolate	fish fingers	hamburger

Table 7.2 Our friends' telephone numbers

telephoneid	nameid	telephonenumber
1	1	001665443
2	2	002555555
3	1	001878345

- Die Art und Weise, wie wir die Daten speichern, ist jedoch nicht ideal. In der ersten Tabelle haben wir drei Spalten, die praktisch das Gleiche messen. Und was, wenn es noch mehr Lieblingsessen gibt? Das Hinzufügen von Informationen auf diese Weise führt zu einer Menge redundanter Informationen.
- Wir können die Daten normalisieren, um Redundanzen zu vermeiden.

Tabellarische Daten: Beispiel

Speicherung von Daten in Tabellen

- Eine einzige Tabelle stellt bereits Beziehungen her:
Alle Daten in einer Zeile gehören zu demselben Datensatz.
- Wenn wir komplexere Beziehungen darstellen wollen (z. B. das Gewicht einer Person zweimal messen oder das Gewicht ihrer Kinder ebenfalls messen), können wir Daten von einer Tabelle zur anderen in Beziehung setzen.

Beispiel

- Wir haben Daten über Peter, Paul und Mary gesammelt.
- Wir haben Informationen über Geburtstage, Telefonnummern und Lieblingsspeisen.
- Wie können wir diese Informationen in Tabellen darstellen?

Table 7.3 Our friends' birthdays—revised

name id	first name	last name	birthday
1	Peter	Pascal	01/02/1991
2	Paul	Panini	02/03/1992
3	Mary	Meyer	03/04/1993

Table 7.4 Our friends' food preferences

nameid	foodname	rank
1	spaghetti	1
1	hamburger	2
2	fruit salad	1
3	chocolate	1
3	fish fingers	2
3	hamburger	3

- Besser ist es, die Informationen aufzuteilen und eine weitere Tabelle für die Essenspräferenzen zu erstellen.
- Es bleibt immer noch etwas Redundanz übrig. Ist es wirklich notwendig, "Hamburger" zweimal in der Tabelle zu haben?

Tabellarische Daten: Beispiel

Speicherung von Daten in Tabellen

- Eine einzige Tabelle stellt bereits Beziehungen her:
Alle Daten in einer Zeile gehören zu demselben Datensatz.
- Wenn wir komplexere Beziehungen darstellen wollen (z. B. das Gewicht einer Person zweimal messen oder das Gewicht ihrer Kinder ebenfalls messen), können wir Daten von einer Tabelle zur anderen in Beziehung setzen.

Beispiel

- Wir haben Daten über Peter, Paul und Mary gesammelt.
- Wir haben Informationen über Geburtstage, Telefonnummern und Lieblingsspeisen.
- Wie können wir diese Informationen in Tabellen darstellen?

Table 7.5 Our friend's food preferences—revised

rankid	nameid	foodid	rank
1	1	1	1
2	1	2	2
3	2	3	1
4	3	4	1
5	3	5	2
6	3	2	3

Table 7.6 Food types

foodid	food name	healthy	kcalp100g
1	spaghetti	no	0.158
2	hamburger	no	0.295
3	fruit salad	yes	0.043
4	chocolate	no	0.546
5	fish fingers	no	0.290

- Jetzt ist es besser.
- Bei der Umstrukturierung der Informationen in unserer Datenbank haben wir **Redundanz (Duplikation)** vermieden.
- Dies ist der Prozess der **Datenbanknormalisierung**.

Hintergrund und Geschichte

- SQL (ausgesprochen [es,kju:'el] wie in S-Q-L, oder [si:kwəl] wie in sequel) steht für **Structured Query Language**. Ursprünglich hieß sie SEQUEL (Structured English Query Language), wurde aber aufgrund von Markenrechtsproblemen fallen gelassen.
- Es handelt sich um eine domänenspezifische Sprache, die für die Abfrage von Daten in relationalen Datenbanken entwickelt wurde.
- Ursprünglich wurde sie 1974 bei IBM von **Donald D. Chamberlin** und **Raymond F. Boyce** entwickelt.

Warum SQL?

- Es gibt zwar unterschiedliche Datenbanktypen, aber die meisten relationalen Datenbanken, denen Sie begegnen werden, sprechen SQL.
- Die wichtigste Fähigkeit für die Arbeit mit Datenbanken ist das Erlernen von SQL.
- SQL wird in vielen (den meisten?) Stellenausschreibungen für Datenwissenschaftler als erforderliche Fähigkeit aufgeführt.

Was sind Datenbanken?

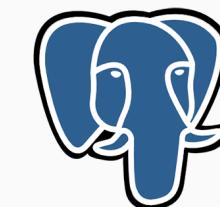
- Datenbanken sind eine organisierte Sammlung von Daten.
- Sie sind so organisiert, dass sie ein effizientes Abrufen von Daten(-selektionen) ermöglichen.
- Sie enthalten Daten + Metadaten über Struktur und Organisation.
- Der Zugriff auf sie erfolgt im Allgemeinen über ein Datenbankmanagementsystem.

Wo sind Datenbanken?

- Datenbanken können lokal oder dezentral, im Arbeitsspeicher oder auf der Festplatte existieren.
- Wenn sie lokal gespeichert sind, werden sie als Binärdatei (nicht als Textdatei) gespeichert.
- Im Allgemeinen denken wir an ein **Client-Server-Modell**:
 - Datenbanken befinden sich auf einem **Server**, der sie verwaltet.
 - Die Benutzer interagieren mit dem Server über ein **Client**-Programm.
 - Mehrere Benutzer können auf dieselbe Datenbank **gleichzeitig zugreifen**.

Was sind DBMS?

- Datenbankmanagementsysteme (DBMS) bieten **effiziente**, **zuverlässige**, **bequeme**, **sichere** Speicherung von und Zugriff auf **massive** Daten.
 - **Effizient**: Einfach schnell.
 - **Zuverlässig**: Hohe Betriebszeit.
 - **Komfortabel**: Hochwertige Abfragesprachen.
 - **Sicher**: Robust gegenüber Stromausfällen, Knotenausfällen, usw.
 - **Mehrbenutzer**: Gleichzeitigkeitskontrolle. Nicht nur ein Benutzer, sondern mehrere.
 - **Massiv**: Denken Sie an Terabytes, nicht an Gigabytes. Verarbeitung von Daten, die sich außerhalb des Speichers befinden.
- Es gibt **etliche DBMS** allein für relationale Datenbankstrukturen.
- RDBMS unterscheiden sich in Bezug auf Fähigkeiten, implementierte Funktionen, Betriebssystemunterstützung und vieles mehr. Siehe [The Database Database](#) für einen Überblick.



PostgreSQL



Drei Grundformen von DBMS

- **Client-Server-DBMS:** Läuft auf einem leistungsstarken zentralen Server, den Sie von Ihrem Computer (dem Client) aus ansteuern. Ideal für die gemeinsame Nutzung von Daten durch mehrere Personen in einer Organisation. Beliebte Client-Server-DBMS sind PostgreSQL, MariaDB, SQL Server und Oracle.
- **Cloud DBMS:** Ähnlich wie Client-Server-DBMS, aber sie werden in der Cloud ausgeführt. Das bedeutet, dass sie problemlos extrem große Datenmengen verarbeiten und bei Bedarf automatisch mehr Rechenressourcen bereitstellen können. Beliebte Beispiele: Snowflake, RedShift von Amazon und BigQuery von Google.
- **Eingebettetes (prozessinternes) DBMS:** Werden vollständig auf Ihrem Computer oder innerhalb einer Anwendung ausgeführt. Sie eignen sich hervorragend für die Arbeit mit großen Datensätzen, bei denen Sie der Hauptnutzer sind. Beispiele: SQLite oder [DuckDB](#).

Weitere nützliche Ressourcen für den Einstieg in das Thema Datenbanken

Einen guten Einstieg bietet [The Beginner's Guide to Databases](#).

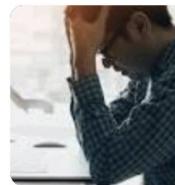
Außerdem [Awesome MySQL](#), eine Goldgrube für die Arbeit mit MySQL.



Microsoft Excel blunder: Developers blamed for loss of thousands of COVID-19 test results

The error has hampered the UK's contact-tracing program at a time when the country is undergoing a second wave of coronavirus infections. By using the XLS ...

1 month ago



Excel glitch leads to nearly 16,000 confirmed coronavirus cases going unreported in United Kingdom

For the test and trace program to work well, contacts should be notified as soon as possible. Health Secretary Matt Hancock told MPs that the problem related to ...

1 month ago



How were 16,000 Test and Trace coronavirus cases lost on Excel?

The cases were lost due to a technical error on a Microsoft Excel spreadsheet. ... Trace 'immediately' after the issue was resolved and thanked contact tracers for ...

1 month ago



Does Contact Tracing Work? Quasi-Experimental Evidence from an Excel Error in England*

Thiemo Fetzer[†] Thomas Graeber[‡]

November 24, 2020

Abstract

Contact tracing has been a central pillar of the public health response to the COVID-19 pandemic. Yet, contact tracing measures face substantive challenges in practice and well-identified evidence about their effectiveness remains scarce. This paper exploits quasi-random variation in COVID-19 contact tracing. Between September 25 and October 2, 2020, a total of 15,841 COVID-19 cases in England (around 15 to 20% of all cases) were not immediately referred to the contact tracing system due to a data processing error. Case information was truncated from an Excel spreadsheet after the row limit had been reached, which was discovered on October 3. There is substantial variation in the degree to which different parts of England areas were exposed – by chance – to delayed referrals of COVID-19 cases to the contact tracing system. We show that more affected areas subsequently experienced a drastic rise in new COVID-19 infections and deaths alongside an increase in the positivity rate and the number of test performed, as well as a decline in the performance of the contact tracing system. Conservative estimates suggest that the failure of timely contact tracing due to the data glitch is associated with more than 125,000 additional infections and over 1,500 additional COVID-19-related deaths. Our findings provide strong quasi-experimental evidence for the effectiveness of contact tracing.

Keywords: HEALTH, CORONAVIRUS

JEL Classification: I31, Z18

Weshalb SQL-Skills wichtig sind

Technology

Facebook made big mistake in data it provided to researchers, undermining academic work

Company accidentally left out half of all of its U.S. users in providing data to a research consortium

The error resulted from Facebook accidentally excluding data from U.S. users who had no detectable political leanings — a group that amounted to roughly half of all of Facebook's users in the United States. Data from users in other countries was not affected.

"It's data. Of course, there are errors," said Gary King, a Harvard professor who co-chairs Social Science One. "This, of course, was a big error."

King, director of the university's Institute for Quantitative Social Science, said dozens of papers from researchers affiliated with Social Science One had relied on the data since Facebook shared the flawed set in February 2020, but he said the impact could be determined only after Facebook provided corrected data that could be reanalyzed. He said some of the errors may cause little or no problems, but others could be serious.

Social Science One shared the flawed data with at least 110 researchers, King said.

An Italian researcher, Fabio Giglietto, discovered data anomalies last month and brought them to Facebook's attention. The company contacted researchers in recent days with news that they had failed to include roughly half of its U.S. users — a group that likely is less politically polarized than Facebook's overall user base. The New York Times first reported Facebook's error.

Source Washington Post

 **Sol Messing** @SolomonMg · Sep 11 ...
What happened that generated the error: TBD. I'd bet that U.S. user-political affinity was joined to the rest of the data using a LEFT JOIN instead of a LEFT OUTER JOIN. Again FB folks are likely working to fix this ASAP.
3 10 35 ↑

 **Sol Messing** @SolomonMg · Sep 11 ...
What was the likely consequence: people in the U.S. with no interest in political information were excluded. Substantively this would make FB look more hyper-partisan, as per [@deaneckles](#)' tweet here:

 Dean Eckles @deaneckles · Sep 11
That is, contra some reactions that somehow this error "helped" Facebook, I would expect this made FB look more filled with misinfo & polarizing content than it was.
Obviously, this error will have lasting consequences...
twitter.com/daveyalba/stat...
[Show this thread](#)
1 8 31 ↑

 **Sol Messing** @SolomonMg · Sep 11 ...
What are the broader systematic issues in play here: researchers didn't have access to the raw data or pipeline code. That's a huge deal and makes it nearly impossible to do the usual, focused deep dive data forensics that research often entails.

Source Solomon Messing / Twitter

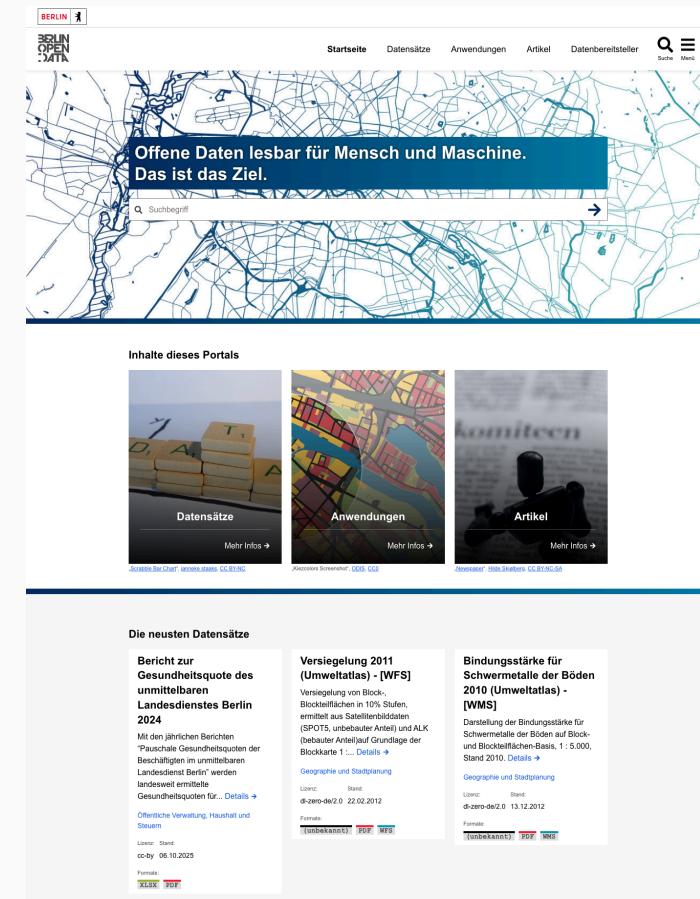
Open Government Data

Was sind Open Government Data (OGD)?

- **Definition:** Daten, die von öffentlichen Stellen bereitgestellt und zur freien Nutzung offen zugänglich sind.
- **Kernprinzipien:**
 - Kostenfreier Zugang, Maschinenlesbarkeit
 - Offene Formate (z. B. CSV, JSON, GeoJSON)
 - Offene Lizenzen (z. B. [Datenlizenz Deutschland](#), CC BY)

Historische Entwicklung

- International:
 - 2009: Start von [data.gov](#) (USA)
 - 2010: [data.gov.uk](#) (UK)
- Deutschland:
 - 2013: Start von [GovData.de](#)
 - 2017/2021: [Open-Data-Gesetz](#) (§ 12a Open Data Gesetz)
 - 2023: Ausbau durch [Data Act](#) und Datenstrategie der Bundesregierung



Was ist DCAT-AP.de?

- DCAT-AP.de ist das **deutsche Anwendungsprofil** des europäischen Metadatenstandards **DCAT-AP (Data Catalog Vocabulary – Application Profile)**.
- Ziel: Einheitliche **Beschreibung, Austausch und Auffindbarkeit** von offenen Verwaltungsdaten.
- Entwickelt von **GovData, BMI, FITKO und KoSIT**.

Warum wichtig?

- Schafft **Interoperabilität** zwischen Datenportalen in Bund, Ländern und Kommunen.
- Ermöglicht **automatische Aggregation** von Datensätzen auf Portalen wie *GovData.de* oder *European Data Portal*.
- Grundlage für **maschinenlesbare Metadaten**

Beispiel: DCAT-AP.de-Metadatenstruktur (vereinfacht)

```
dcat:Dataset:  
dct:title: "Baustellen Hamburg 2025"  
dct:description: "Offene Daten zu aktuellen Baustellen im Stadtgebiet Hamburg"  
dct:publisher: "Freie und Hansestadt Hamburg, Behörde für Verkehr"  
dct:license: "Datenlizenz Deutschland – Namensnennung 2.0"  
dcat:distribution:  
  - dcat:accessURL: "https://suche.transparenz.hamburg.de/dataset/baustellen-2025"  
    dct:format: "CSV"  
    dcat:downloadURL: "https://daten.hamburg.de/baustellen.csv"
```

Chancen

- **Transparenz & Vertrauen:** Nachvollziehbare Entscheidungsprozesse.
- **Innovation:** Grundlage für datenbasierte Dienstleistungen (z. B. Chatbots, Analysen).
- **Kollaboration:** Kooperation zwischen Verwaltung, Forschung, Zivilgesellschaft.

Herausforderungen

- **Qualität & Aktualität:** Viele Datensätze sind veraltet oder unvollständig.
- **Fragmentierung:** Unterschiedliche Standards, viele kleine Portale.
- **Nachnutzbarkeit:** Fehlende Dokumentation, unklare Lizenzierung.
- **Personelle Kapazitäten:** Open Data oft „Nebenprojekt“ ohne langfristige Betreuung.

OGD im Daten-Workflow denken

1. **Erhebung:** Welche Verwaltungsdaten können proaktiv offen bereitgestellt werden?
2. **Verarbeitung:** Wie lassen sich OGD mit internen Daten kombinieren (z. B. über APIs)?
3. **Analyse:** Welche Mehrwerte entstehen durch Verknüpfung (z. B. Lagebilder, Prognosen)?
4. **Kommunikation:** Ergebnisse öffentlich rückspiegeln – *Open by Default, Secure by Design.*

Daten-Workflow in der Behörde

Ziel

Mitarbeitende sollen **schnell, verständlich und transparent** Antworten aus Akten erhalten. Der Chatbot:

- durchsucht **Metadaten** und **Dokumente** (PDF, Word).
- fasst Inhalte in **Klartext** zusammen.
- zeigt **genau an, woher** die Information stammt (Akte/Seite/Abschnitt, Link).
- respektiert **Zugriffsrechte** (nur sehen, was erlaubt ist).

Typische Fragen

- „Wie ist der aktuelle Genehmigungsstand Fall X?“
- „Welche Auflagen wurden im Bescheid vom 12.04.2024 gemacht?“
- „Bitte die wichtigsten Änderungen der letzten 30 Tage zu Vorgang Y.“



1) Bedarf klären

Welche Fragen soll der Chatbot beantworten? Welche Akten & Felder sind relevant?

2) Daten gewinnen

Metadaten aus Fachverfahren, Dokumente aus Dateiablagen, optional OGD-Ressourcen.

3) Qualität sichern

Plausibilitäten, Duplikate, Dateiformate, fehlende Felder
– automatisiert prüfen.

4) Speichern & Zugreifen

Rohdaten im Staging, kurierte Sichten für Suche & Antworten.

5) Analysieren & aufbereiten

Indizes/Embeddings für Suche, SQL-Views für Kennzahlen, Regeln für Zitate.

6) Bereitstellen

Interne API für den Chatbot; optional Veröffentlichung ausgewählter Daten als Open Data.

7) Betreiben & verbessern

Monitoring, Nutzungsfeedback, Änderungslog, regelmäßige DMP-Updates.

Wozu?

Der DMP ist die **Betriebsanleitung** für Daten: Was liegt wo, warum, wer darf was, welche Regeln gelten?

Nutzen: Nachvollziehbarkeit, Rechtssicherheit (DSGVO), Qualität, Wiederverwendung.

Was muss rein? (Chatbot-Kontext)

- **Bestand:** Akten-Metadaten, Dokumente, Zugriffsrechte.
- **Rechtslage:** Zweckbindung, Löschfristen, Veröffentlichung.
- **Struktur:** Tabellenfelder, Dateiformate, Versionierung.
- **Metadaten:** Titel, Beschreibung, Lizenz, Kontakt (DCAT-AP.de, wo sinnvoll).
- **Veröffentlichung:** Was bleibt intern? Was kann auf's Portal?

DMP-Checkliste für den Chatbot

- **Rollen:** Data Owner (Fachbereich), Data Steward (Qualität/Metadaten), Plattform-Team (Betrieb), Datenschutz.
- **Datenfluss:** Wie kommen neue Akten/Dokumente ins System? (Rhythmus, Verantwortliche)
- **Qualität:** Mindestfelder, Pflicht-Metadaten, Prüfregeln (z.B. Datum, Aktenzeichen).
- **Rechte:** Wer sieht welche Akten? (RBAC/ABAC)
- **Transparenz:** Wie werden **Quellenangaben** im Chatbot erzeugt und getestet?
- **Schnittstellen:** Interne API, optional Open-Data-Publikation.

Speichern

- PostgreSQL (Metadaten, Rechte, kuratierte Sichten)
- Objektspeicher (PDF/Scans)
- Suchindex/Embeddings (schnelle Textsuche)

Verarbeiten

- Pipelines (Airflow/Prefect/GitHub Actions) für Import, Qualität, Aktualisierung
- Transformationen in DB (SQL/dbt) – nachvollziehbar & versioniert
- API (REST) für den Chatbot + Audit-Logs

ETL (Extract → Transform → Load)

- Vor dem Laden wird stark bereinigt/vereinfacht.
- **Gut**, wenn Quellsysteme sehr uneinheitlich sind und zuerst „aufgeräumt“ werden muss.
- **Risiko**: Details gehen verloren, Nachvollziehbarkeit sinkt.

ELT (Extract → Load → Transform)

- **Alles erst laden**, dann **im Zielsystem** aufbereiten (SQL, dbt).
- **Vorteil**: Rohdaten bleiben erhalten, bessere Reproduzierbarkeit.
- **Für den Chatbot** ideal: verschiedene Sichten für Suche, Antworten, Open Data.

1. **Extract:** Neue/aktualisierte Akten-Metadaten + Dokument-Verweise abholen (APIs, Dateifreigaben).
2. **Load:** Rohdaten unverändert ins Staging (DB + Objektspeicher).
3. **Transform:**
 - Kuratierte Tabellen/Views für Suche & Antworten (z.B. `akten_kern`, `dokumente_index`).
 - Qualitätsprüfungen (Plausibilitäten, Pflichtfelder).
 - Provenienzfelder für transparente Quellenangabe.
4. **Serve:** Interne API für den Chatbot (Fragen rein → Antwort + Quellnachweise raus).
5. **Publish (optional):** Freigegebene Tabellen automatisiert ans Open-Data-Portal.

Prinzip

Jede Antwort enthält **Zitate** mit: *Akte, Dokument, Seite/Abschnitt, Zeitstempel, Link* (oder Pfad).

So können Mitarbeitende die Antwort **prüfen** und im Original **nachlesen**.

Umsetzung

- Beim Zusammenstellen der Antwort werden die **Trefferstellen** (IDs, Seiten) mitgegeben.
- In der DB gibt es **Provenienz-Spalten** (z. B. `quelle_dokument_id`, `quelle_seite`).
- Die API liefert Antwort + **Liste der Quellen**; der Chatbot rendert diese als Fußnoten/Box.

- **Tests:** Pflichtfelder, Wertebereiche, eindeutige Aktenzeichen; automatisiert in der Pipeline.
- **Versionierung:** SemVer für Sichten/Modelle, **Änderungslog** (Was hat sich geändert? Warum?).
- **Monitoring:** Ladezeiten, Fehlerraten, API-Verfügbarkeit, Suchqualität (Treffergenauigkeit).
- **Support:** Kontaktadresse, Ticket-Prozess, Reaktionszeiten.
- **Datenschutz:** Zugriffsebenen prüfbar (Audit-Logs), Löschfristen automatisiert.

Rollout-Plan in kleinen, sicheren Schritten

1. **Pilot** mit einem Fachbereich (begrenzter Aktenbestand).
2. **Such-Chat** nur auf **Metadaten** → frühes Feedback zur Treffqualität.
3. **Dokumenteinbindung** (Auszüge) + **Quellenbox** in Antworten.
4. **Rechteprüfung** vor Anzeige; Schulung & Leitfaden.
5. **Skalierung** auf weitere Bereiche; OGD-Teilmengen veröffentlichen.

Beispiele aus Eurer Praxis!