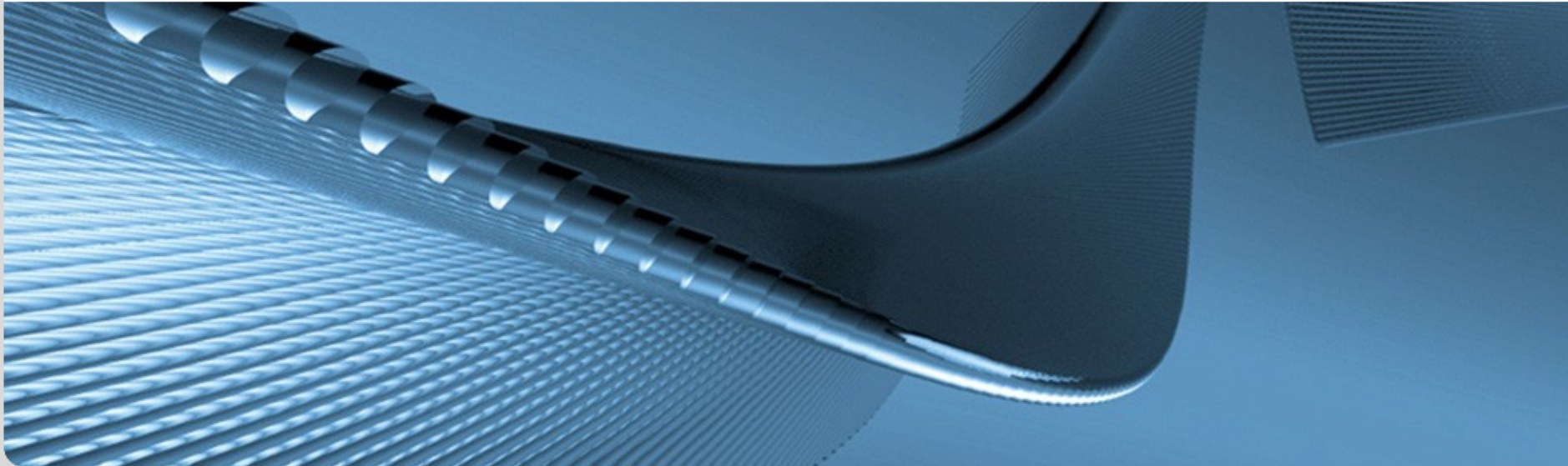


Datenschutz und Privatheit in vernetzten Informationssystemen

Kapitel 10: Hippokratische Datenbanken

Erik Buchmann (buchmann@kit.edu)

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



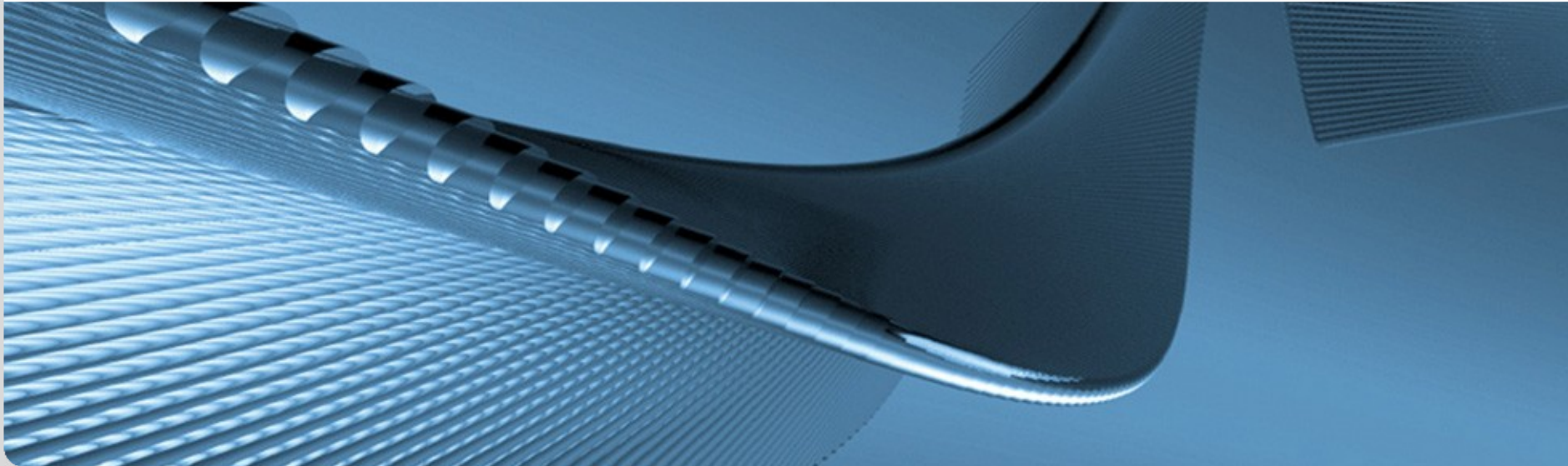
Inhalte und Lernziele dieses Kapitels

- Einführung
- Hippokratische Datenbanken
 - Aufbau
 - Das Inferenz-Problem
 - Auditing
- Abschluss

- Lernziele
 - Sie können die Zielsetzung und den Aufbau Hippokratischer Datenbanken beschreiben.
 - Ihnen sind die theoretischen Grenzen dieses Konzepts bewußt.
 - Sie können erklären, wie mittels Auditing verdächtige Anfragen identifiziert werden, über die bestimmte Informationen aus der Datenbank herausgelangt sein könnten.

Einführung

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



Wiederholung DBMS (1/3)

- Begriffsklärung
 - Datenbank (DB): Menge von Daten
 - Datenbankmanagementsystem (DBMS):
Software zur Datenverwaltung
 - Datenbanksystem: DBMS + Daten
 - Tupel: ein Datensatz in der DB
 - Relation: Menge von Tupeln (anschaulich: Tabelle)

- Verwaltung **großer Datenmengen** und Entwicklung entsprechender Anwendungen.
- Warum Datenbanken?
 - Standardsoftware reduziert Komplexität bei der Anwendungsentwicklung
 - 50% weniger Aufwand bei Anwendungsentwicklung mit Datenbanken, gesamthaft betrachtet, bei überschaubaren Projekten.
 - Größere Differenz bei größeren Projekten.
Verwendung von Datenbank-Technologie ermöglicht neue Anwendungen; ihre Entwicklung ohne Datenbank-Technologie wäre zu komplex.

- Die 9 Codd'schen Regeln:
 - **Integration:** nichtredundante Datenverwaltung,
 - **Operationen:** Speichern, Suchen, Ändern,
 - **Katalog:** Metadaten im Data Dictionary,
 - **Benutzersichten,**
 - **Integritätssicherung:** Korrektheit der Datenbank,
 - **Datenschutz:** Ausschluß unautorisierter Zugriffe,
 - **Transaktionen:** mehrere atomare DB-Operationen als Funktionseinheit,
 - **Synchronisation:** parallele Transaktionen koordinieren,
 - **Datensicherung:** Wiederherstellung von Daten nach Systemfehlern.

- Der Betreiber (zumindest der Administrator) darf *alles*
 - beliebige Abfragen
 - beliebige Arten der Datenmanipulation
 - laufende Abfragen einsehen
 - Zugriffsrechte festlegen
 - beliebige Sichten auf die Daten definieren

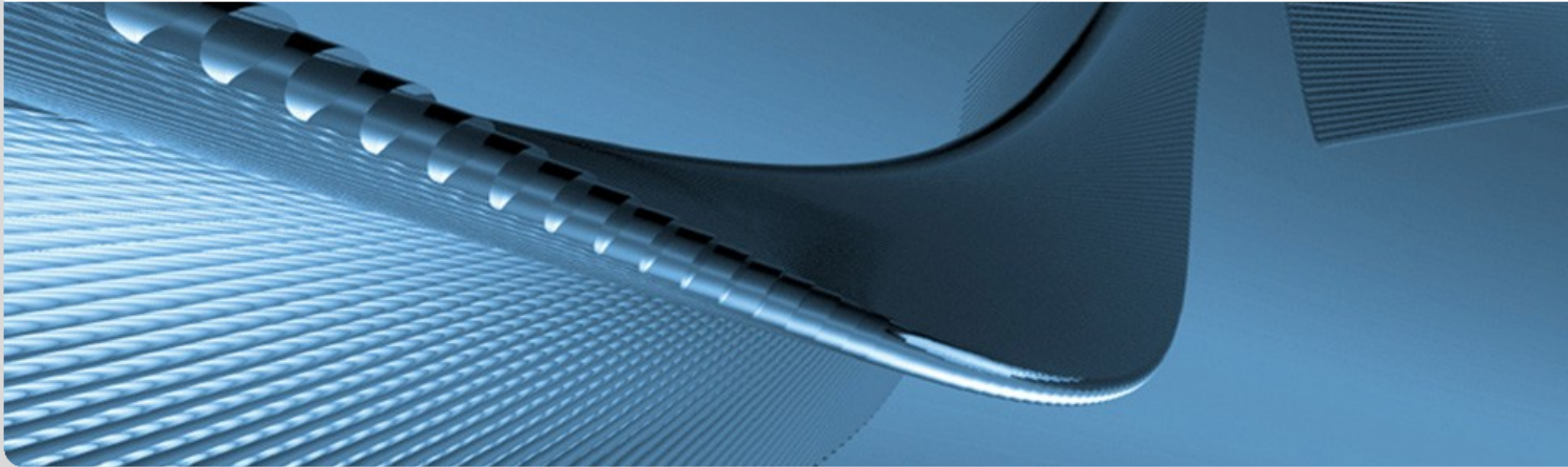
- Keine Garantien, dass sich der Betreiber *tatsächlich* an seine eigenen Datenschutzrichtlinien hält

- Problem bisher: ***Vertrauen in Anbieter erforderlich***
 - vertrauenswürdige Dritte Partei als Betreiber von Anonymisierungsdiensten
 - Anbieter setzt die Datenschutzansätze effektiv ein, die er verspricht
 - Anbieter hält sich an seine Datenschutzerklärung
 - Ausnahmen: Methoden zum Datenselbstschutz; aber die sind oft schwierig *sicher* anzuwenden

- Ansatz dieser Vorlesung: ***vertrauenswürdiges System***
 - selbst Systemadministrator kann nur Daten sehen, wenn vom Betroffenen erlaubt
 - technische Maßnahmen *verhindern* Datenmißbrauch

Hippokratische DB am Beispiel

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



Der Hippokratische Eid

- Benannt nach dem griechischen Arzt Hippokrates, um 400 v. Chr.

- Ausschnitt:

“...Was ich bei der Behandlung oder auch außerhalb meiner Praxis im Umgange mit Menschen sehe und höre, das man nicht weiterreden darf, werde ich verschweigen und als Geheimnis bewahren...”

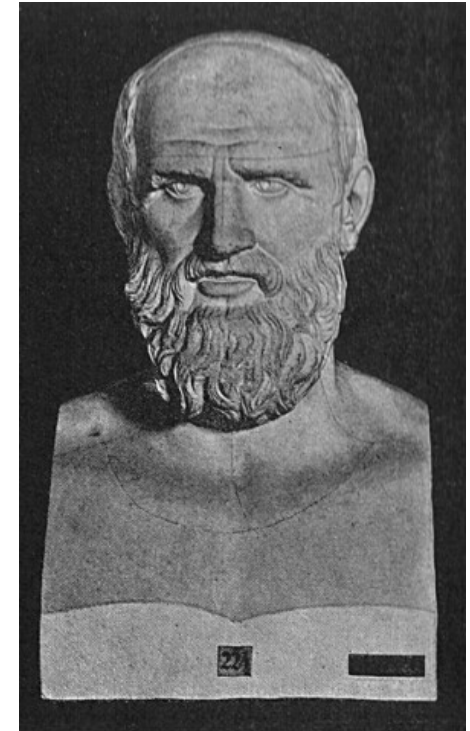


Bild: Wikimedia

10 Prinzipien Hippokratischer DBs (1/2)

■ Purpose Specification

- Erhebungszweck mit erhobenen Daten verknüpfen

■ Consent

- Erhebung mit Zustimmung oder Einwilligung des Betroffenen; an Erhebungszweck gebunden

■ Limited Collection

- nur Daten erheben, die für Anfrage erforderlich sind

■ Limited Use

- nur Anfragen gemäß Erhebungszweck beantworten

■ Limited Disclosure

- keine Informationen herausgeben, wenn nicht für Erhebungszweck erforderlich oder ohne Zustimmung

10 Prinzipien Hippokratischer DBs (2/2)

■ Limited Retention

- Daten dürfen nur so lange gespeichert bleiben, bis der Erhebungszweck erfüllt ist

■ Accuracy

- Daten müssen korrekt sein

■ Safety

- Datensicherheit gewährleisten

■ Openness

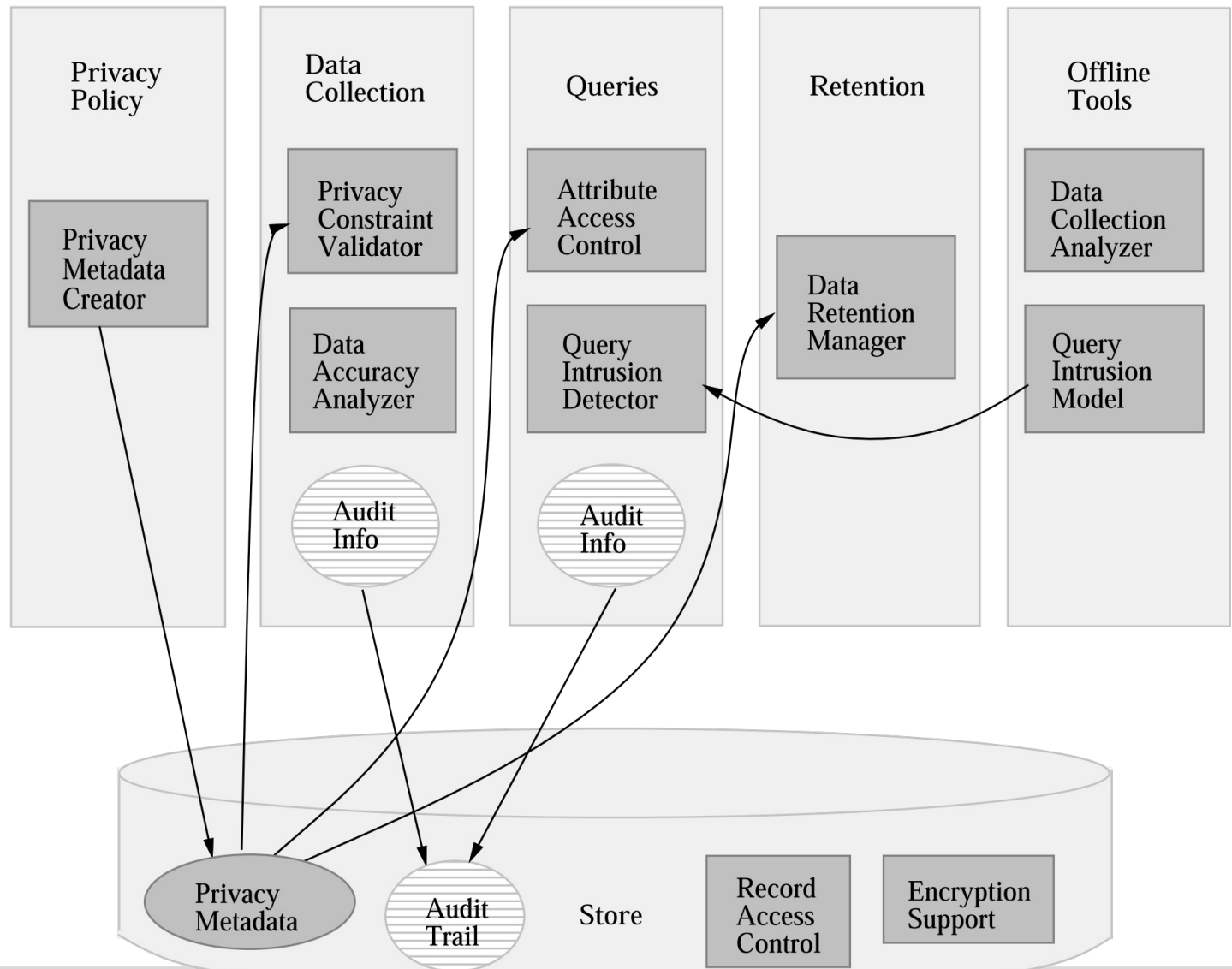
- Betroffene sollen Zugang zu ihren Daten haben

■ Compliance

- Betroffene sollen nachprüfen können, ob ihre Daten gemäß den o.g. Prinzipien behandelt wurden

- Online-Buchladen mit Hippokratischer Datenbank
 - Verkauf von Büchern (Lieferadresse, Kreditkarte)
 - Berechnung von Buchempfehlungen (Verkäufe)
 - Telemarketing (Kontaktadresse)
- Kunde A:
 - um Datenschutz besorgt,
alle Daten schnellstmöglich löschen
- Kunde B:
 - komfortbewußt,
möchte Empfehlungen, Registrierung etc. nutzen, aber keine Werbung erhalten

Architekturmodell



Quelle: [1]

■ Schema für Auftragsdaten

table	attributes
customer	purpose, customer-id, name, shipping-address, email, credit-card-info
order	purpose, customer-id, transaction-id, book-info, status

Bezahlung, Versand, Kontaktdaten
Auftrag, Bestellstatus

■ Schema für Datenschutz-Metadaten

table	attributes
privacy-policies	purpose, table, attribute, { external-recipients }, retention
privacy-authorizations	purpose, table, attribute, { authorized-users }

Zweckbindung, externe Datenempfänger, Löschfristen
autorisierte Nutzer

Datenschutzerklärung des Buchladens

purpose	table	attribute	external-recipients	retention
purchase	customer	name	{ delivery-company, credit-card-company }	1 month
purchase	customer	shipping-address	{ delivery-company }	1 month
purchase	customer	email	<i>empty</i>	1 month
purchase	customer	credit-card-info	{ credit-card-company }	1 month
purchase	order	book-info	<i>empty</i>	1 month
registration	customer	name	<i>empty</i>	3 years
registration	customer	shipping-address	<i>empty</i>	3 years
registration	customer	email	<i>empty</i>	3 years
recommendations	order	book-info	<i>empty</i>	10 years
purchase-circles	customer	shipping-address	<i>empty</i>	1 year
purchase-circles	order	book-info	{ aggregated-all }	1 year

- Zwecke: Kauf, Anmeldung, Empfehlungen, Werbung
- je nach Zweck können Daten 1 Monat bis 10 Jahre nach Dienstonutzung gelöscht werden
- einige Daten werden an externe Dienstleister übermittelt

Anm.: Datenschutzerklärung kann automatisch generiert werden, z.B. aus P3P-Policy

Autorisierte Nutzer im Buchladen

purpose	table	attribute	authorized-users
purchase	customer	customer-id	<i>all</i>
purchase	customer	name	{ shipping, charge, customer-service }
purchase	customer	shipping-address	{ shipping }
purchase	customer	email	{ shipping, customer-service }
purchase	customer	credit-card-info	{ charge }
purchase	order	customer-id	<i>all</i>
purchase	order	transaction-id	<i>all</i>
purchase	order	book-info	{ shipping }
purchase	order	status	{ shipping, customer-service }
registration	customer	customer-id	<i>all</i>
registration	customer	name	{ registration, customer-service }
registration	customer	shipping-address	{ registration }
registration	customer	email	{ registration, customer-service }
recommendations	order	customer-id	{ mining }
recommendations	order	transaction-id	{ mining }
recommendations	order	book-info	{ mining }
purchase-circles	customer	customer-id	{ olap }
purchase-circles	customer	shipping-address	{ olap }
purchase-circles	order	customer-id	{ olap }
purchase-circles	order	book-info	{ olap }

Beispiele für autorisierte Nutzer

- Versandabteilung (shipping) darf zugreifen auf
 - Name
 - Versandadresse
 - Bestellung
 - Buchungsstatus, Email-Adresse
- darf nicht zugreifen auf
 - Kreditkartendaten

→ bis hierher lässt sich alles noch mit klassischen DBMS realisieren

- Nutzerpräferenzen mit Datenschutzerklärung abgleichen
 - Nutzer A: (Datenschutz)
 - opt-out auf alles, das nicht explizit für Kauf erforderlich
 - falls Präferenzen eine Löschfrist <1 Monat erfordern, würde DB die Transaktion zurückweisen
 - Nutzer B: (Komfort)
 - opt-in für dauerhafte Registrierung beim Händler, Berechnung von Buch-Empfehlungen
- Nutzerdaten in DB speichern
 - Tupel mit *erlaubtem Verwendungszweck* annotieren
 - *Audit Trail* starten um Einhaltung der Vorgaben zu kontrollieren
 - *Korrektheit* der Daten kontrollieren

Anfragen an die Datenbank (1/2)

- Jede Anfrage wird mit einem *Zweck* an DBMS geschickt;
in unserem Beispiel:
 - Queries für Data Mining hätten den Zweck “recommendations”
 - Queries für Versandabwicklung hätten den Zweck “purchase”
- Anfrage nur dann ausführen,
 - wenn Anfragesteller in der Menge der autorisierten Nutzer für diesen Zweck enthalten
 - wenn alle abgefragten Attribute für diesen Zweck vom Betroffenen freigegeben sind

Anm: klassische DBMS binden Zugriffsrechte an Datensätze

- Während der Anfrageausführung
 - DBMS blendet alle Attribute aus, für die Zweck der Anfrage nicht im erlaubten Verwendungszweck
 - z.B.: Empfehlungen werden nur aus den Käufen berechnet, die mit Zweck “recommendation” annotiert, selbst wenn noch für andere Zwecke in DB enthalten
- Nach Anfrageausführung
 - *Audit Trail* abschließen
 - *Query Intrusion Detector* sucht nach unüblichen (aber erlaubten) Anfragemustern
 - z.B. wenn Angestellter aus der Versandabteilung viele Kundendaten auf einmal kopiert um sie zu verkaufen

- *Data Retention Manager*
 - löscht Daten nach Ablauf des Verfallsdatums

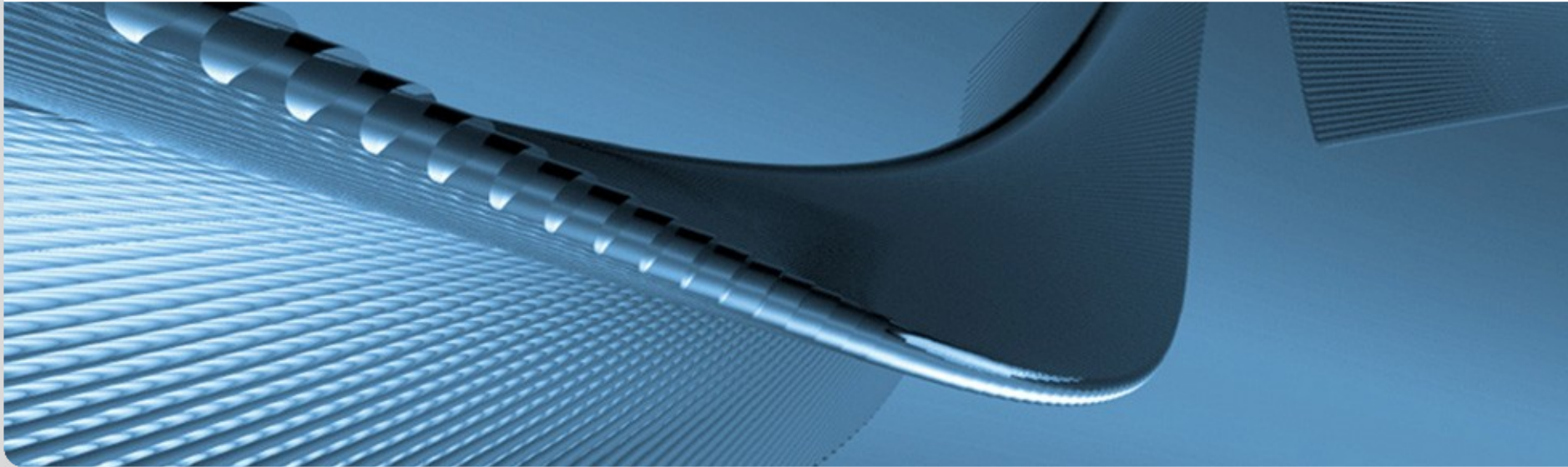
- *Data Collection Manager*
 - untersucht Anfrageset
 - Gemäß Anwendungszweck soll
 - DB so wenig Daten wie möglich speichern
 - Anfrage auf so wenig Daten wie möglich zugreifen

- Eingabe: Zweck der Anfrage, Attribute der Anfrage, Zweckbindung der erhobenen Daten
 - Welche Daten werden für eine Aufgabe nicht benötigt?
→ *Daten mit einem Verwendungszweck annotiert, werden aber niemals für diesen Zweck abgerufen*

- Drei große Fragestellungen:
 - **Zugriff:** identifiziere Attribute, die für gegebenen Zweck erhoben, aber nicht genutzt werden
 - **Granularität:** analysiere für jeden Zweck und jedes numerische Attribut, ob es in der Form benötigt wird
 - **Minimalität:** was ist die minimale Anfrage für einen gegebenen Zweck?

Sicherheit gegen Inferenz

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



- Verhindern, dass durch geschickten Anfragemix private Tupel bekannt werden
 - Beweis dass Datenbank kompromittiert werden kann: zeigen, dass eine Sequenz von Anfragen konstruiert werden kann, welche den gesuchten Wert mit einem eindeutigen Schlüssel verknüpft
 - *einzelnes Tupel mit bestimmter Merkmalskombination ist Quasi-Identifizier*
 - Beweis des Gegenteils schwierig:
zeigen dass keine solche Anfragefolge konstruierbar
 - Was ist mit Vorkenntnissen oder Hintergrundinformationen?

- Selektiert einen *exakten* Wert vom Query Set, z.B. Maximum, Minimum, Median, etc.
 - **select** min(age) **from** persons **where** sex = female **and** job = scientist
- Trivial: **select** min(age) **from** persons **where** name = X
- Problematisch:
 - Überlappung in der Selektion mehrerer Queries
 - 1. Median von {x, a, b, c}
 - 2. Median von {y, a, b, c}
 - 3. ...
 - Anm: selbst wenn alles bis auf eine Überlappung von 1 verboten wird, kann die DB kompromittiert werden
→ zwei gleiche Ergebnisse := gleiches Tupel

- Gib die Zahl oder ein berechnetes Aggregat über die Tupel zurück, auf die ein gegebenes Kriterium passt
 - **select** count(*) **from** persons **where** sex = female **and** age > 40 **and** job = scientist
 - Problematisch:
 - zu kleine Ergebnismengen für die Selektion, oder
 - aufeinanderfolgende Anfragen mit kleinen Schnittmengen in den where-Bedingungen
 - 1. Durchschnitt von $\{x, v_1, v_2, \dots, v_n\}$
 - 2. Durchschnitt von $\{y, v_1, v_2, \dots, v_n\}$
- wenn x bekannt kann y berechnet werden

- Gewichtete Summe der Elemente von der selektierten Tupelmeng
 - **select** sum(A.a*B.b) **from** A, B **where**...

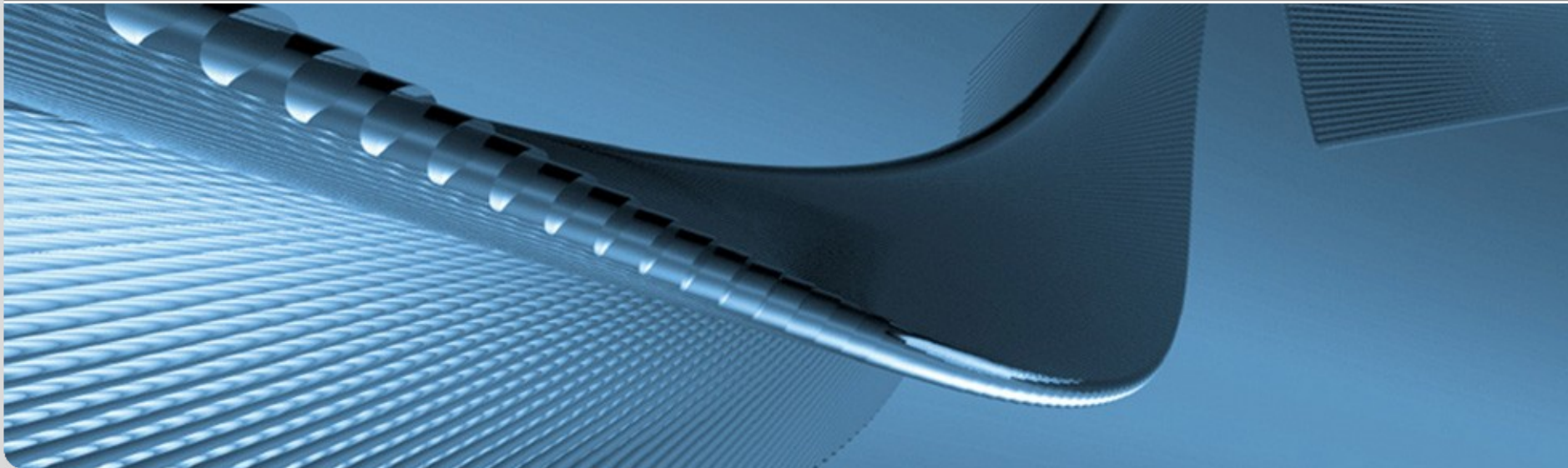
$$q(z_1, \dots, z_k) = \sum_{j=1}^k a_j z_j$$

- Problem auch hier: Überlappung in Selektion;
für $q_1 = \{v_1, v_2, v_{j-1}, x \dots v_k\}$, $q_2 = \{v_1, v_2, v_{j-1}, y \dots v_k\}$
gilt: $(q_1 - q_2) = a_j^* (x - y)$

- Es ist sehr schwierig, **sichere** Hippokratische Datenbanken zu konstruieren
 - Berechnungsvorschriften wie Aggregate, Summen, gewichtete Summen lassen sich oft angreifen, wenn ein Element in der Selektionsmenge dem Angreifer bekannt
 - Selektionsanfragen wie Min oder Max lassen sich oft angreifen, wenn gleiche Tupel in der Selektionsmenge mehrerer Anfragen enthalten sein dürfen
- Einfache Strategien wie “verbiere Schnittmengen von Prädikaten” führen zu einem unbenutzbaren DBS

Auditing

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“

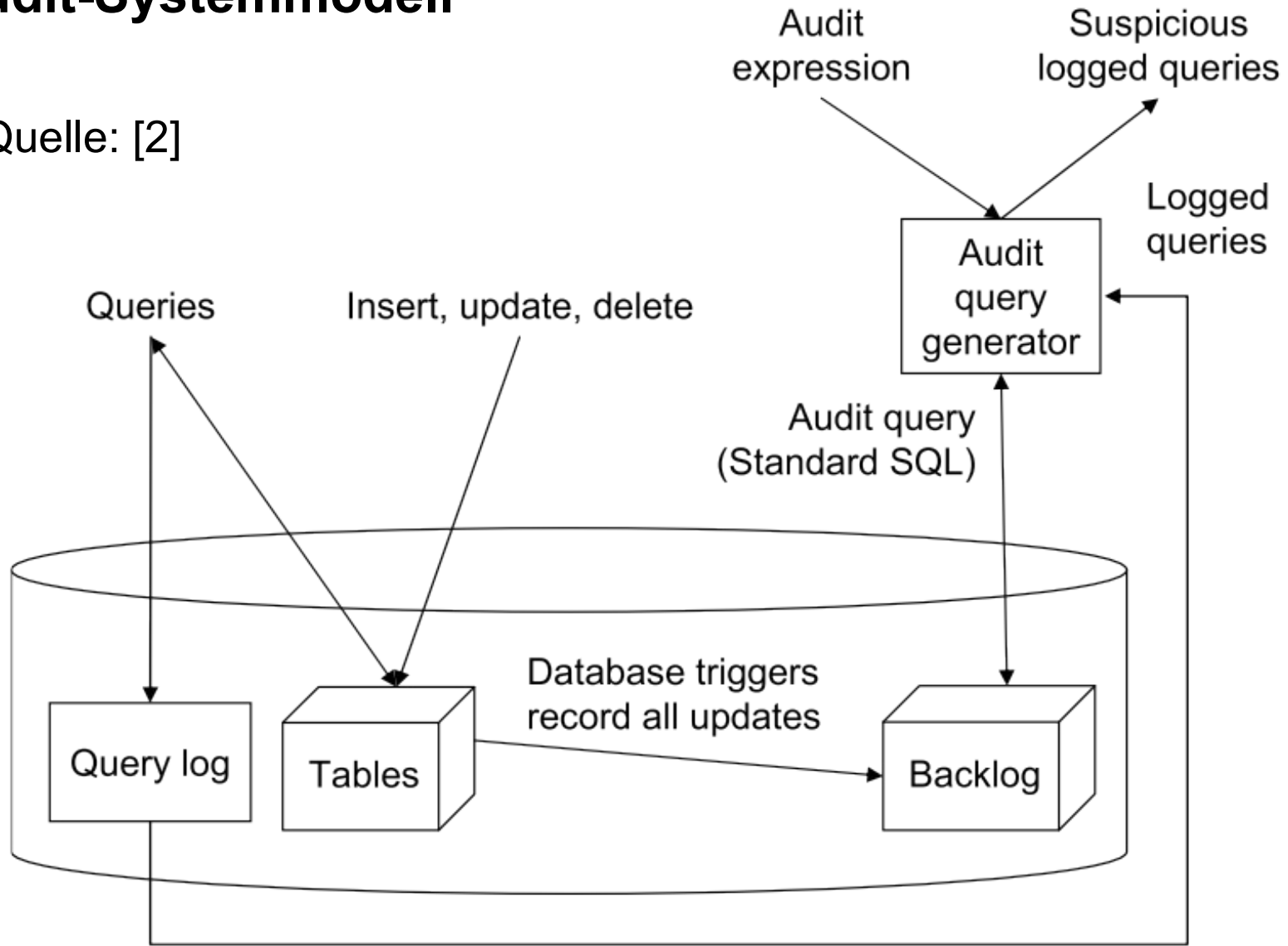


Auditing in Hippokratischen DB

- Ziel: Compliance
 - Nachweis, dass sich das DBMS an die Datenschutzerklärung gehalten hat
- Beispiel
 - Nutzer lässt einen Bluttest bei Firma A machen, und erhält kurz darauf Werbung für Diabetes-Tests von Firma B. Nun möchte der Nutzer wissen, *an wen A welche* Daten weitergegeben hat.

Audit-Systemmodell

■ Quelle: [2]



■ *Backlog*

- speichert insert, delete, update
- wird ohnehin von jeder Datenbank geführt
(nur möglicherweise nach dem Ende der Transaktion gelöscht)

■ *Query Log*

- speichert Anfragen zusammen mit Metadaten (Anfrager, Anfragezeit, Anfragezweck)

■ *Audit Query Generator*

- generiert aus den Audit-Termen und Queries im Log Anfragen, um herauszufinden ob und welche Tupel betroffen sind

- Sprache zum Auditing, in Anlehnung an SQL

during *start-time to end-time*

audit *audit list*

from *table list*

where *condition list*

- Beispiel: Relation Customer: {cid, plz,}

Relation Treatment: {tid, pcid, disease ...}

- Wurde 'disease' Attribut von jemandem weitergegeben, der im PLZ-Bezirk 96120 wohnt?

audit disease

from Customer c, Treatment t

where c.cid = t.pcid **and** c.plz = 95120

- verdächtige Anfrage, wenn Adresse 96120 enthält

select address **from** Customer c, Treatment t

where c.cid = t.pcid **and** t.disease = 'diabetes'

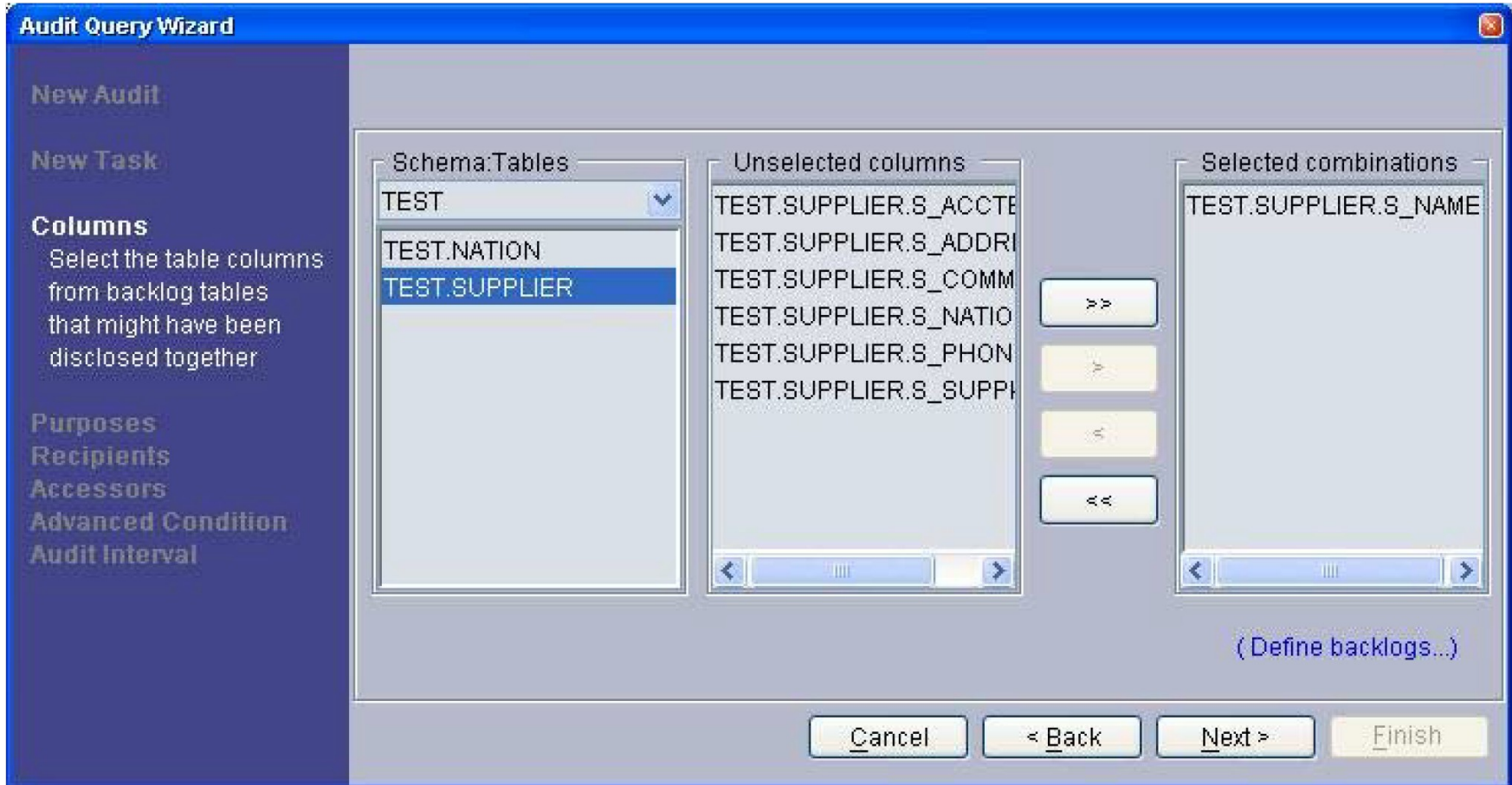
Was sind 'verdächtige' Anfragen?

- *Intuitiv*: Anfragen, die auf Audit-Attribute zugreifen UND Tupel existieren in der DB, bei denen Audit-Attribute das Ergebnis beeinflussen
- **audit** disease
from Customer c, Treatment t
where c.cid = t.pcid **and** c.plz = 95120
 - disease hat Einfluss auf Ergebnismenge wenn c.plz = 95120 in DB vorhanden
select address
from Customer c, Treatment t
where c.cid = t.pcid **and** t.disease = 'diabetes'
 - disease hat keinen Einfluss auf Ergebnismenge
select address
from Customer c, Treatment t, Medication m
where c.cid = t.pcid **and** t.disease = m.medication

- Tables T, R mit Columns $C_{1..n}$
Selektion σ , Multi-Set Projektion π
SPJ-Query $Q = \pi_{COQ}(\sigma_{PQ}(TxR))$
- Unentbehrlichkeit (indispensability) eines Tupels v für Q :
 $ind(v, Q) \Leftrightarrow \pi_{COQ}(\sigma_{PQ}(TxR)) \neq \pi_{COQ}(\sigma_{PQ}((T-\{v\})xR))$
- Candidate Query Q für Audit Expression A mit Output O :
 $cand(Q, A) \Leftrightarrow C_Q \subseteq C_{OA}$
- Verdächtige (suspicious) Query Q bzgl. A :
 $susp(Q, A) \Leftrightarrow \exists v \in T \text{ s.t. } ind(v, Q) \wedge ind(v, A)$

- Für Beweise und komplexere Anfragen als einfache Select-Projektion-Join siehe [2]
- Vergleichbare formale Frameworks lassen sich auch für die Identifikation ungenutzter bzw. zu feingranularer Attribute aufstellen

IBM Hippocratic Database Auditing (1/3)



The screenshot shows the 'Audit Query Wizard' dialog box. On the left is a dark blue sidebar with navigation options: 'New Audit', 'New Task', 'Columns', 'Purposes', 'Recipients', 'Accessors', 'Advanced Condition', and 'Audit Interval'. The 'Columns' option is selected, and its description reads: 'Select the table columns from backlog tables that might have been disclosed together'. The main area contains three panels: 'Schema:Tables' with a dropdown set to 'TEST' and a list containing 'TEST.NATION' and 'TEST.SUPPLIER'; 'Unselected columns' with a list of columns including 'TEST.SUPPLIER.S_ACCTE', 'TEST.SUPPLIER.S_ADDRI', 'TEST.SUPPLIER.S_COMM', 'TEST.SUPPLIER.S_NATIO', 'TEST.SUPPLIER.S_PHON', and 'TEST.SUPPLIER.S_SUPPI'; and 'Selected combinations' with a list containing 'TEST.SUPPLIER.S_NAME'. Between the panels are four buttons: '>>', '>', '<', and '<<'. At the bottom right of the main area is a link '(Define backlogs...)'. At the bottom of the dialog are four buttons: 'Cancel', '< Back', 'Next >', and 'Finish'.

IBM Hippocratic Database Auditing (2/3)

Audit Query Wizard

New Audit

New Task

Columns

Purposes
Select the purposes

Recipients

Accessors

Advanced Condition

Audit Interval

Filter purposes
*

Use Predicate

SELECT ALL

Resulted predicate: "%"

Available purposes

- Accounting
- Marketing
- Purpose 1
- Purpose 2
- Purpose 3

Selected purposes

Add new purpose: Add

Cancel < Back Next > Finish

IBM Hippocratic Database Auditing (3/3)

Audit Query Wizard

New Audit

New Task

Columns

Purposes

Recipients
Select the recipients

Accessors

Advanced Condition

Audit Interval

Filter recipients
*

Use Predicate
STARTS WITH
Chief
Resulted predicate:
"Chief%"

Available recipients
Chief Executive Officer
Chief Finance Officer
Chief Information Officer
Chief Marketing Officer
Recipient 1
Recipient 2
Recipient 3

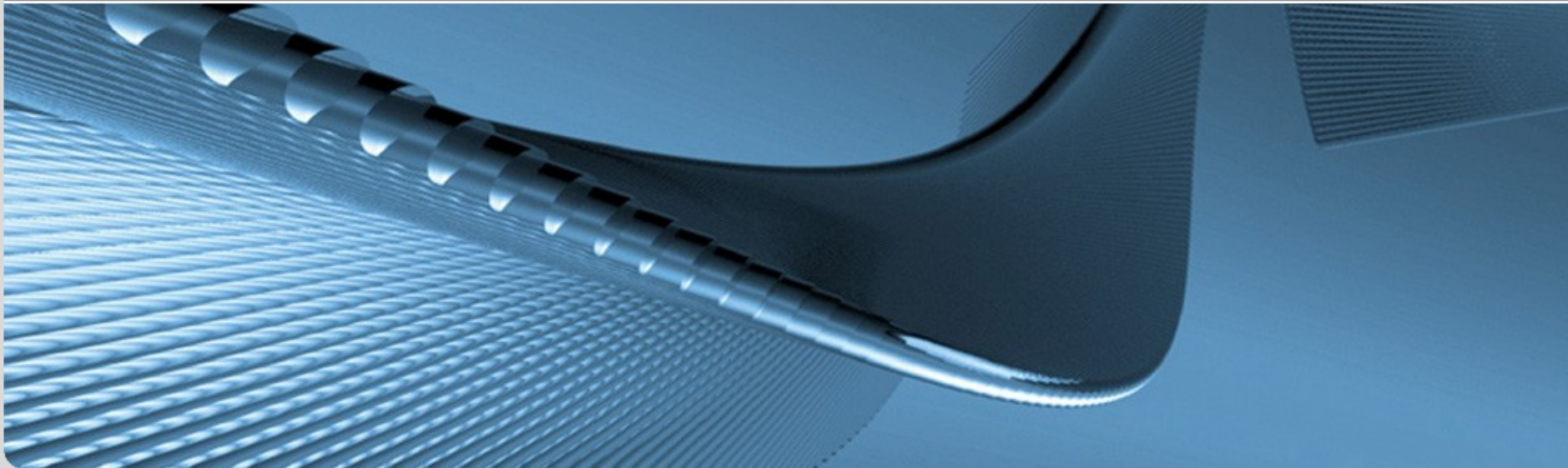
Selected recipients

Add new recipient: Add

Cancel < Back Next > Finish

Abschluss

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



- Hippokratische Datenbanken sind ein vielversprechendes Konzept
 - Informationssystem garantiert Einhaltung von Datenschutzregeln selbst gegen seinen Betreiber
 - Auditing, Transparenz, Abgleich mit Nutzerpräferenzen
- Aber: zahlreiche Schwierigkeiten bei der Umsetzung
 - Inferenz: einige Arten von Anfragen sind systembedingt unsicher, wenn Hintergrundwissen beim Angreifer verfügbar
 - noch nicht angesprochen: Performanz, Sprache für Präferenzen und Datenschutz, etc.

- [1] R. Agrawal et al.: *Hippocratic Databases*. VLDB 2002
- [2] R. Agrawal et al.: *Auditing Compliance with a Hippocratic Database*. VLDB 2004