Datenbanksysteme

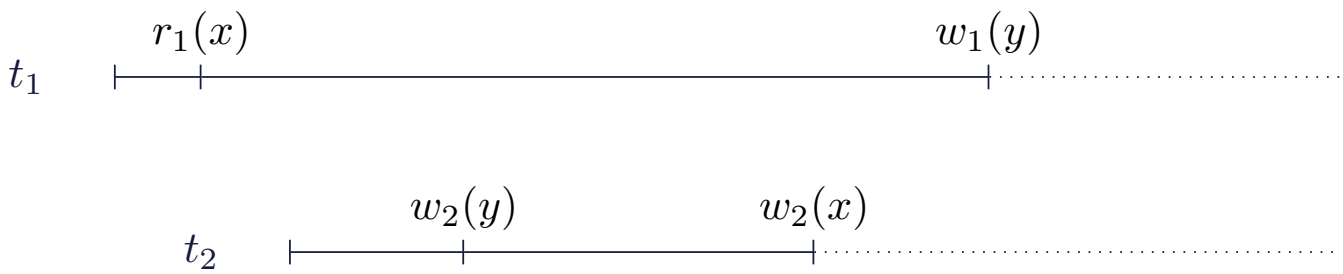
Wintersemester 2016/17

Prof. Dr.-Ing. Sebastian Michel
TU Kaiserslautern

smichel@cs.uni-kl.de

Deadlocks

- werden verursacht durch zyklisches Warten auf Sperren
- Beispiel



Deadlock-Erkennung

Aufbau eines dynamischen Wait-for-Graph (WfG) mit aktiven TAs als Knoten und Wartebeziehungen als Kanten: Eine Kante von t_i nach t_j ergibt sich, wenn t_i auf eine von t_j gehaltene Sperre wartet.

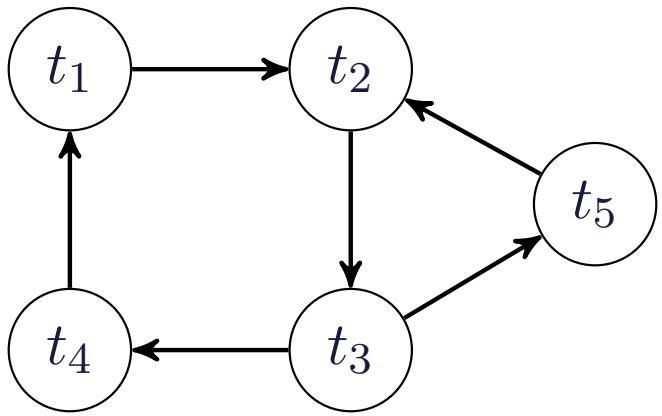
Testen des WfG zur Zyklenerkennung

- kontinuierlich (bei jedem Blockieren)
- periodisch (z.B. einmal pro Sekunde)

Erkennung von Verklemmung

Wartegraph mit zwei Zyklen

- $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4 \rightarrow t_1$
- $t_2 \rightarrow t_3 \rightarrow t_5 \rightarrow t_2$



- Beide Zyklen können durch Rücksetzen von t_3 "gelöst" werden.
- Zyklenerkennung durch Tiefensuche im Wartegraphen.
- Verschiedene Strategien welche TA zurückgesetzt werden soll (jüngste, älteste, nach # Sperren etc.)

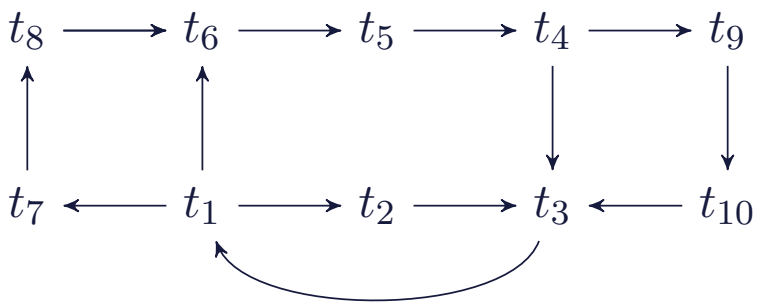
Deadlock Auflösung

- Wähle eine TA in einem WfG-Zyklus aus
- Setze diese TA zurück
- Wiederhole diese Schritte, bis keine Zyklen mehr gefunden werden

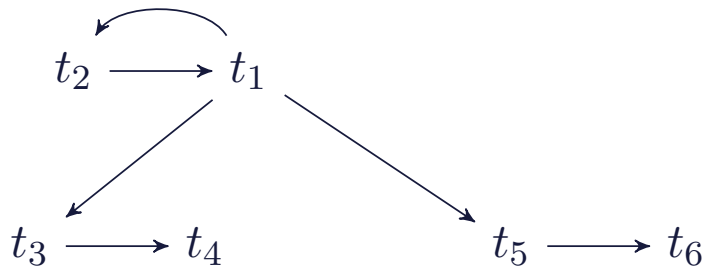
Mögliche Strategien zur Bestimmung von "Opfern"

- Zuletzt blockierte TA
- Zufällige TA
- Jüngste TA
- Minimale Anzahl von Sperren
- Minimale Arbeit (geringster Ressourcen-Verbrauch, z.B. CPU-Zeit)
- Meiste Zyklen
- Meiste Kanten

Deadlock Auflösung - Beispiele



Meiste-Zyklen-Strategie würde t_1 (oder t_3) auswählen, um alle 5 Zyklen aufzubrechen.



Meiste-Kanten-Strategie würde t_1 auswählen, um 4 Kanten zu entfernen.

Strategien zur Deadlock-Vermeidung

Idee: Blockierungen (lock waits) werden eingeschränkt, so dass **stets** ein azyklischer WfG gewährleistet werden kann.

Strategien

- **Wait-Die:** Sobald Sperranforderung von t_i zu Konflikt mit t_j führt: Wenn t_i vor t_j gestartet ist (also älter ist), dann **wait**(t_i), sonst **restart**(t_i).
  "TA wird nur von jüngeren blockiert"
- **Wound-Wait:** Sobald t_i mit t_j in Konflikt gerät: Wenn t_i vor t_j, dann **restart**(t_j) sonst **wait**(t_i)
  "TA kann nur von älteren blockiert werden und TA kann den Abbruch von jüngeren, mit denen sie in Konflikt gerät, verursachen"

Weitere Strategien zur Deadlock-Vermeidung

- **Immediate Restart:** Sobald t_i mit t_j in Konflikt gerät: **restart**(t_i)
- **Running Priority:** Sobald t_i mit t_j in Konflikt gerät: wenn t_j selbst blockiert ist, dann **restart**(t_j) sonst **wait**(t_i)
- **Konservative Ansätze:** TA, die zurückgesetzt wird, ist nicht notwendigerweise in einen Deadlock involviert.
- **Timeout:** Wenn Timer ausläuft, wird t zurückgesetzt in der Annahme, dass t in einen Deadlock involviert ist.

Zeitstempelverfahren

Grundsätzliche Idee

- Transaktion (TA) bekommt bei BOT einen systemweit eindeutigen Zeitstempel; er legt die Serialisierbarkeitsreihenfolge fest.
- TA hinterlässt den Wert ihres Zeitstempels auf jedem Objekt o_i auf das sie zugreift

Timestamp Ordering (TO)

- Jeder TA t_i wird ein eindeutiger Zeitstempel $ts(t_i)$ zugewiesen
- Zentrale TO-Regel: Wenn $p_i(x)$ und $q_j(x)$ in Konflikt stehen, dann muss für jeden Schedule s folgendes gelten:

$$p_i(x) <_s q_j(x) \text{ gdw } ts(t_i) < ts(t_j)$$

Zeitstempelverfahren (2)

Theorem: $Gen(TO) \subseteq CSR$

Prinzipielle Arbeitsweise

- Vergabe von aufsteigend eindeutigen TA IDs (Zeitstempel ts der TA)
- "Stempeln" des Objektes x bei Zugriffen t_i: $TS(x) := ts(t_i)$

Read und Write Timestamps

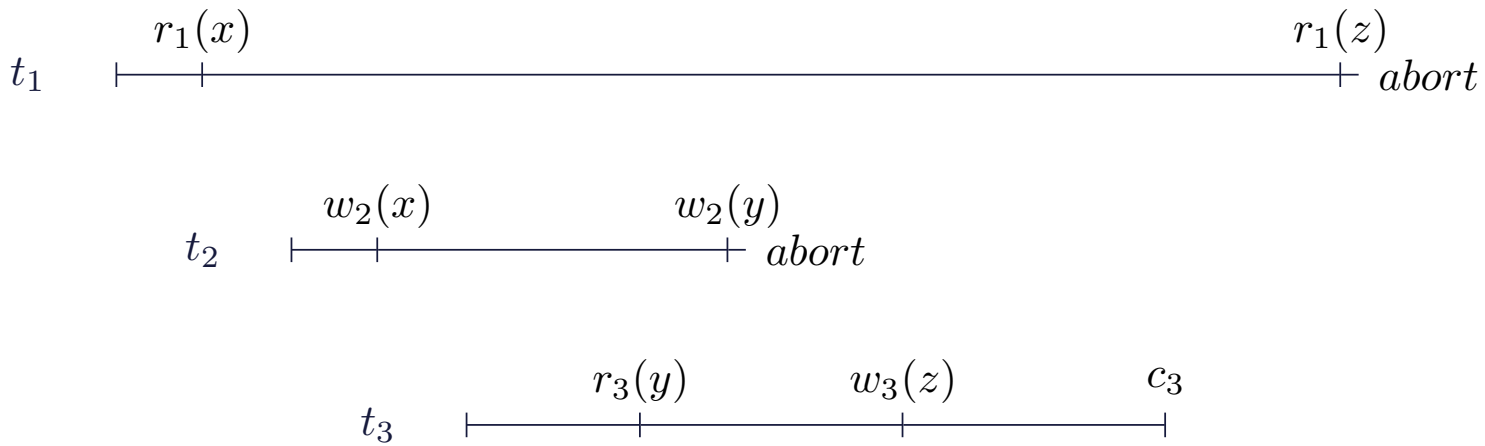
Der TO Scheduler muss für jedes Objekt x die beiden folgenden Zeitstempel speichern:

1. **max-r-scheduled(x)**: der Wert des größten Zeitstempels einer Leseoperation auf x
2. **max-w-scheduled(x)**: den Wert des größten Zeitstempels einer Schreiboperation auf x

Vorgehensweise TO Scheduler

Wenn eine Operation $p_i(x)$ im Scheduler ankommt, wird $ts(t_i)$ verglichen mit max-q-scheduled(x) für jede Operation q, die mit p in Konflikt ist.

- **Falls $ts(t_i) <$ max-q-scheduled(x): $p_i(x)$ wird zurückgewiesen (rejected)**
- **Ansonsten:** $p_i(x)$ wird erlaubt (und an Datenmanager geschickt) und max-p-scheduled(x) wird auf den Zeitstempel $ts(t_i)$ gesetzt falls $ts(t_i) >$ max-p-scheduled(x)



Regeln und Thomas' Write Rule

Abk. $i := ts(t_i)$

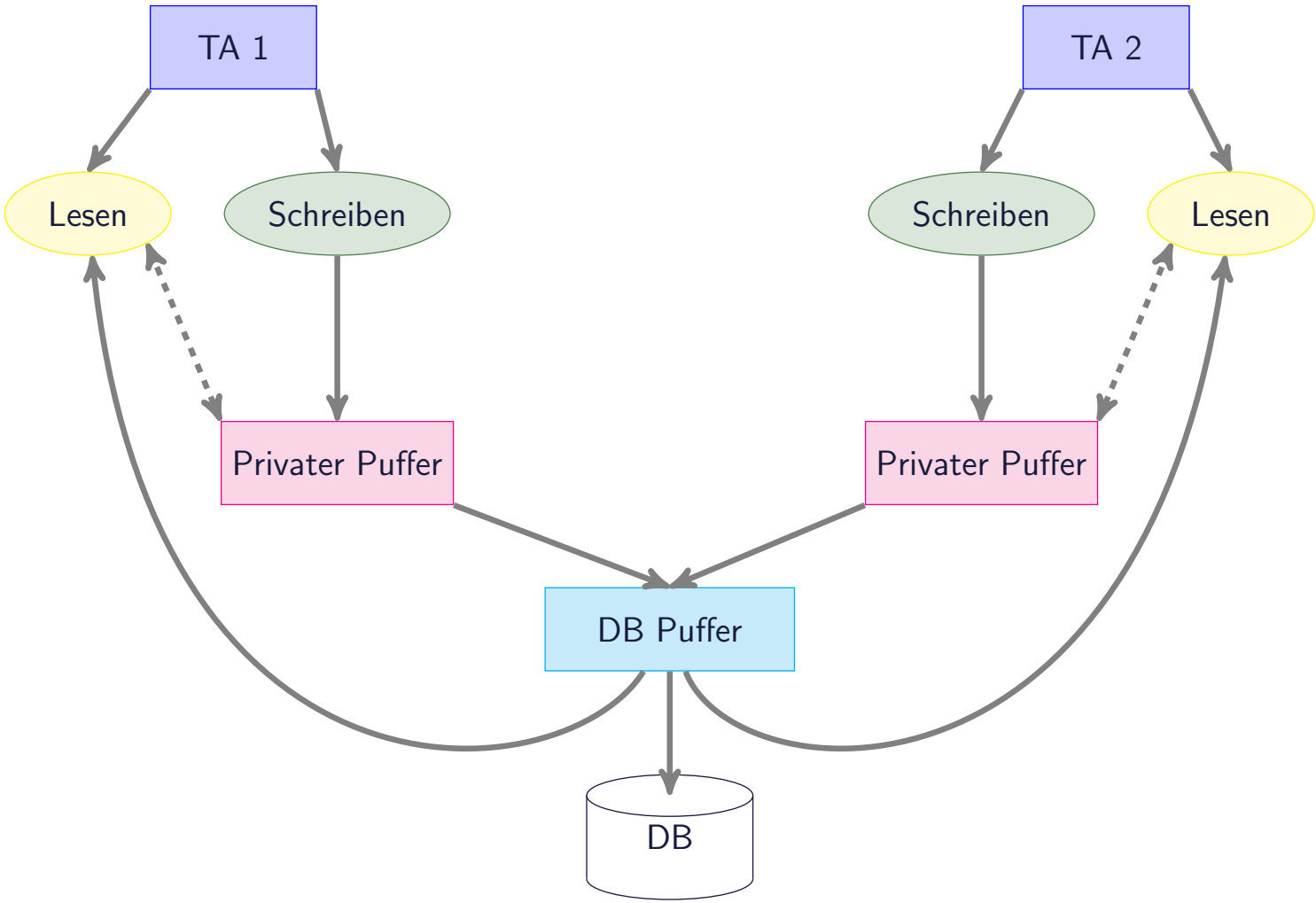
R1	$r_i \wedge (i \geq$ max-w$(x))$	if max-r$(x) < i$ then max-r$(x) := i$; Lesen
R2	$w_i \wedge (i \geq$ max-r$(x)) \wedge (i \geq$ max-w$(x))$	max-w$(x) := i$; Schreiben
R3	$w_i \wedge (i \geq$ max-r$(x)) \wedge (i <$ max-w$(x))$	Ignoriere Schreiben(*)
R4	$w_i \wedge (i <$ max-r$(x))$	Zurücksetzen!
R5	$r_i \wedge (i <$ max-w$(x))$	Zurücksetzen!

(*) Thomas' Write Rule

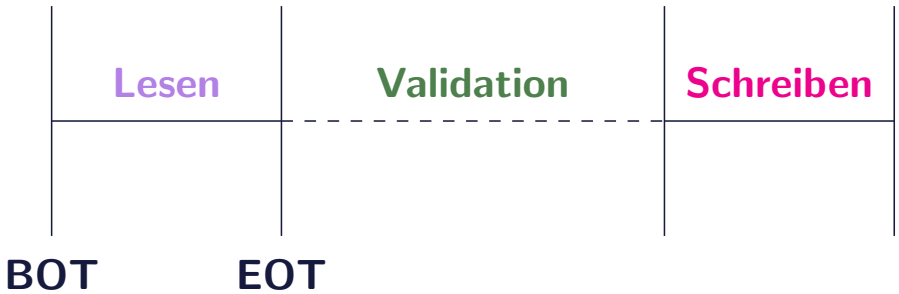
- Optimierung/Reduzierung von Schreibkonflikten
- Idee: Last-write-wins
- Gegeben TA t_i mit Zeitstempel $ts(t_i)$
- Falls $ts(t_i) \geq$ max-r-scheduled(x) und $ts(t_i) <$ max-w-scheduled(x), dann ignoriere Schreiben $w_i(x)$.

Optimistic Concurrency Control (OCC)

Idee: Transaktionen laufen erstmal getrennt voneinander mittels privatem Puffer ab. Anschließend wird validiert ob es Konflikte zwischen den TAs gibt.



Optimistic Concurrency Control (OCC)



Lesephase

- eigentliche TA-Verarbeitung
- Änderungen einer Transaktion werden in privatem Puffer durchgeführt

Optimistic Concurrency Control (OCC)

Validierungsphase

- Überprüfung, ob ein Lese-/Schreib- oder Schreib-/Schreib-Konflikt mit einer parallel laufenden Transaktionen passiert.
- Konfliktauflösung durch Zurücksetzen von Transaktionen

Schreibphase

- nur bei positiver Validierung
- Lese-Transaktion ist ohne Zusatzaufwand beendet
- Schreib-Transaktionen schreiben hinreichende Log-Information und propagieren Änderungen

OCC: Validierungs Prinzipien BOCC und FOCC

Backward Oriented (BOCC)

Validierung gegenüber bereits beendeten (Änderungs-) TAs

Forward Oriented (FOCC)

Validierung gegenüber laufenden TA

Details: Theo Härder: Observations on optimistic concurrency control schemes. Inf. Syst. 9(2): 111-120 (1984)
`http://wwwlgis.informatik.uni-kl.de/cms/fileadmin/publications/1984/Hae84.InformationSystems.pdf`

Sonstiges: Ausblick

Weitere Ansätze/Prinzipien

- **RUX**
- **Hierarchische Sperrverfahren**
- **MVCC: Multi-Version-Concurrency-Control**
- **Snapshot Isolation**

Es gibt auch noch weitere Anomalien, die wir bislang nicht betrachtet haben, z.B. das Phantom Problem, oder Write Skew.

Transaktionsverwaltung in SQL92

Problem:

Serialisierbarkeit kann zu Performanzproblemen führen.

Idee:

Isolationsgarantien abschwächen, um höhere Performanz zu erreichen.

Vier sogenannte Isolationslevel:

set transaction
[read only, | read write,]
[**isolation level**
read uncommitted, |
read committed, |
repeatable read, |
serializable,] |
[**diagnostics size ...,**]

Isolationsstufen

read uncommitted:

- Schwächste Konsistenzstufe
- Liest möglicherweise inkonsistente/uncommitted data
- Nur für read-only TAs
- Geeignet für "browsing"
- Keine locks für read-Operationen

read committed:

- Nur committed Werte dürfen gelesen werden
- Auch für Schreib-TAs erlaubt
- Locks zum Schreiben im strikten 2PL
- Locks zum Lesen nur kurzzeitig gehalten
- Non-repeatable read möglich
- Phantom-Problem möglich

Isolationsstufen (2)

repeatable read:

- Locks zum Schreiben mit striktem 2PL
- Locks zum Lesen mit striktem 2PL
- Non-repeatable read ausgeschlossen
- Phantom-Problem möglich

serializable:

- Volle Serialisierbarkeit
- Nicht immer default
- Locks zum Schreiben mit striktem 2PL
- Locks zum Lesen mit striktem 2PL
- Zusätzlich wird der Zugriff über Indexe/Prädikate gelockt

Snapshot Isolation

- Transaktion t_1 sieht die Datenbank in der Version wie sie zu Beginn von t_1 existiert.
- Werden Daten zwischenzeitlich von anderen Transaktionen geändert, sieht t_1 trotzdem nur die alte Version
- t_1 sieht natürlich alle eigenen Änderungen

Vorteile:

- Lesen sieht keine Daten nicht-abgeschlossener Transaktionen
- Lesen blockiert kein Schreiben
- Schreiben blockiert kein Lesen
- Die Transaktionen arbeiten nebenläufig auf verschiedenen "Snapshots" der Datenbank
- Bessere Performance für die meisten Workloads

Implementierungen: Oracle und Postgres

Snapshot Isolation: Eigenschaften

- Snapshot Isolation sorgt für ein hohes Maß an Isolation
- Snapshot Isolation hat eine bessere Performance als Systeme, die nur striktes 2PL implementieren
- aber:
  - Snapshot Isolation löst nicht vollständig das Phantom-Problem
  - Snapshot Isolation verhindert nicht, dass zwei Transaktionen dasselbe Datum überschreiben (**write skew**)
- Deswegen ist Snapshot Isolation zwischen read committed und repeatable read anzusiedeln.
- Allerdings implementieren die DMBS nicht einfach nur Snapshot Isolation sondern verhindern natürlich write skew.
- Phantom Problem muss man selbst lösen.

  in Postgres z.B. durch explizites zusätzliches Locking (`https://www.postgresql.org/docs/9.6/static/explicit-locking.html`)

Serialisierung in Postgres

Postgres bietet alle 4 Isolationsstufen an, aber:

- **read uncommitted** hat denselben Effekt wie read committed
- **repeatable read** hat denselben Effekt wie serializable
- **serializable** bedeutet **nicht** serialisierbar!
  Warum? Wegen snapshot isolation.

`http://www.postgresql.org/docs/9.1/static/sql-set-transaction.html`

Übersicht

Vorschau auf kommende Inhalte:

- Probabilistic Databases
- Column-Stores und In-Memory Datenbanken
- MapReduce

Probabilistic Databases

Literatur:

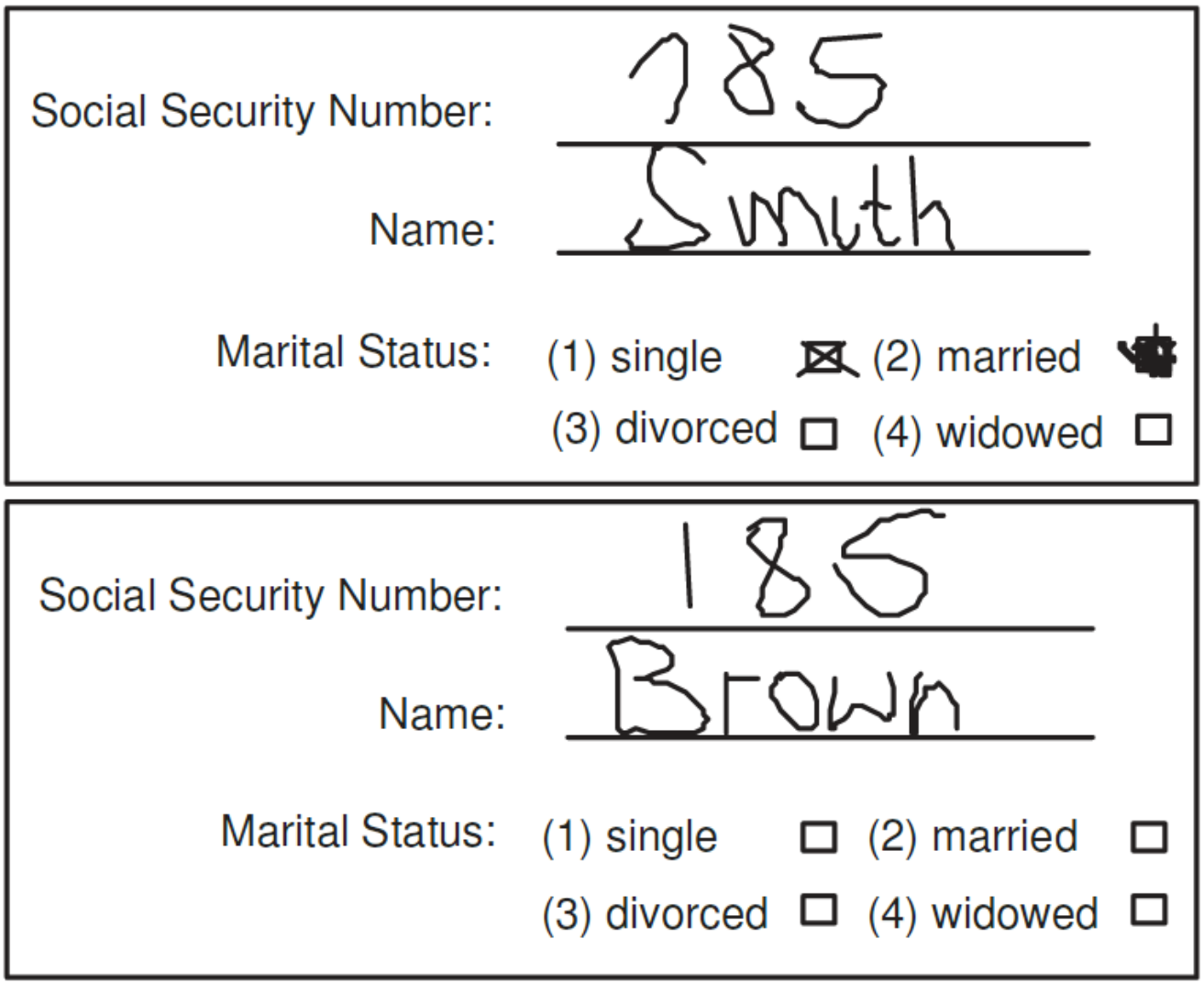
- Dan Suciu, Dan Olteanu, Christopher Ré, Christoph Koch: Probabilistic Databases. Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2011.
- Nilesh Dalvi and Dan Suciu. Efficient Query Evaluation on Probabilistic Databases. VLDB conference, 2004.http://www.vldb.org/conf/2004/RS22P1.PDF

Weitere Quellen:

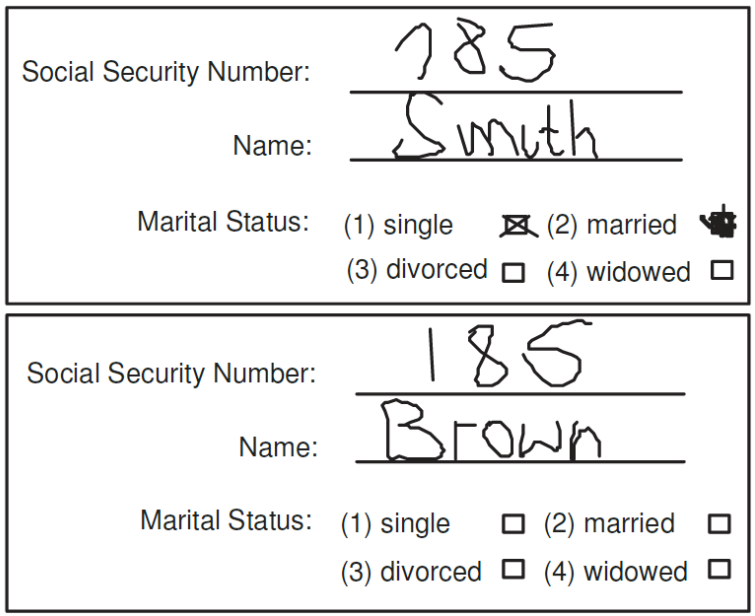
- http://www.cs.ox.ac.uk/dan.olteanu/tutorials/olteanu-pdb-kr16.pdf
- https://homes.cs.washington.edu/~suciu/talk-msra-db-summerschool-2013.pdf
- http://resources.mpi-inf.mpg.de/departments/d5/teaching/ss12/sum/slides/04-probabilistic-databases-handout.pdf

Beispiel

Zwei ausgefüllte Formulare



Beispiel



Mögliche Interpretation in einer Datenbank:

RID	SSN	Name	M
t_1	NULL	Smith	NULL
t_2	NULL	Brown	NULL

NULL Werte enthalten wenig/keine Informationen.

Idee: Wir könnten alle möglichen Ausprägungen in die Datenbank aufnehmen.

- Die SSN von Smith ist entweder 185 oder 785. Die SSN von Brown ist entweder 185 oder 186.
- Smiths Maritial Status (M) ist entweder 1 oder 2. Browns M ist entweder 1, 2, 3 oder 4.

Wir haben insgesamt $2 \times 2 \times 2 \times 4 = 32$ mögliche Interpretationen für die beiden Formulare:

SSN	Name	M
185	Smith	1
185	Brown	1

SSN	Name	M
185	Smith	1
185	Brown	2

SSN	Name	M
185	Smith	1
185	Brown	3

SSN	Name	M
185	Smith	1
185	Brown	4

SSN	Name	M
185	Smith	1
186	Brown	1

SSN	Name	M
185	Smith	1
186	Brown	2

SSN	Name	M
185	Smith	1
186	Brown	3

SSN	Name	M
185	Smith	1
186	Brown	4

und so weiter ....

Unvollständige Datenbanken

Eine **unvollständige Datenbank** ist eine endliche Menge von Datenbankinstanzen $\mathbf{W} = (W_1, \ldots, W_n)$.

W_1

SSN	Name	M
185	Smith	1
185	Brown	1

W_2

SSN	Name	M
185	Smith	1
185	Brown	2

W_3

SSN	Name	M
185	Smith	1
185	Brown	3

W_4

SSN	Name	M
185	Smith	1
185	Brown	4

W_5

SSN	Name	M
185	Smith	1
186	Brown	1

W_6

SSN	Name	M
185	Smith	1
186	Brown	2

Jedes W_i ist eine **mögliche Welt** (**possible world**).

Unvollständige Datenbanken

Eine **unvollständige Datenbank** ist eine endliche Menge von Datenbankinstanzen $\mathbf{W} = (W_1, \ldots, W_n)$.

W_1

SSN	Name	M
185	Smith	1
185	Brown	1

W_2

SSN	Name	M
185	Smith	1
185	Brown	2

W_3

SSN	Name	M
185	Smith	1
185	Brown	3

W_4

SSN	Name	M
185	Smith	1
185	Brown	4

W_5

SSN	Name	M
185	Smith	1
186	Brown	1

W_6

SSN	Name	M
185	Smith	1
186	Brown	2

Jedes W_i ist eine mögliche Welt (possible world).

Typisches Szenario: 200M Personen, 50 Fragen, eine aus 10 000 Antworten unklar (2 Optionen)

- 2^{10^6} possible worlds
- Eine Welt ist eine Tabelle mit 50 Spalten und 200 Millionen Zeilen!

Probabilistische Datenbank

Eine **Probabilistische Datenbank** (probabilistic database) ist $(\mathbf{W}, P)$, mit $\mathbf{W}$ ist eine unvollständige Datenbank und $P : \mathbf{W} \rightarrow [0, 1]$ ist eine Wahrscheinlichkeitsverteilung und $\sum_{W_i \in \mathbf{W}} P(W_i) = 1$

W_1: $P(W_1) = 0.1$

SSN	Name	M
185	Smith	1
185	Brown	1

W_2: $P(W_2) = 0.1$

SSN	Name	M
185	Smith	1
185	Brown	2

W_3: $P(W_3) = 0.1$

SSN	Name	M
185	Smith	1
185	Brown	3

W_4: $P(W_4) = 0.1$

SSN	Name	M
185	Smith	1
185	Brown	4

W_5: $P(W_5) = 0.3$

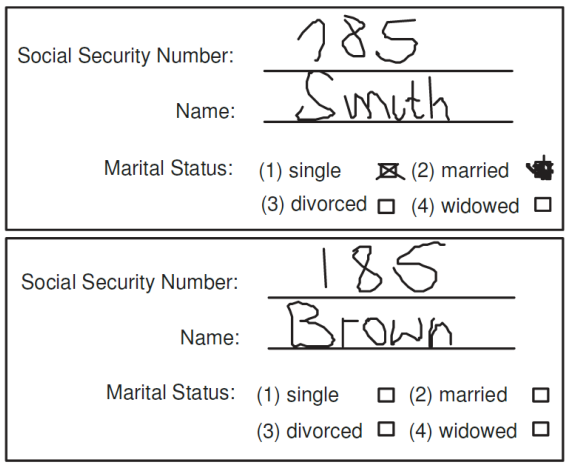
SSN	Name	M
185	Smith	1
186	Brown	1

W_6: $P(W_6) = 0.3$

SSN	Name	M
185	Smith	1
186	Brown	2

Für $\mathbf{W} = \{W_1, \ldots, W_6\}$, $\sum_{W_i \in \mathbf{W}} P(W_i) = 1.$

Kompakte Darstellung von Unvollständigen/Probabilistischen Datenbanken



Kompakte Darstellung durch Oder-Mengen (or-set representation)

SSN	Name	M
{185, 785}	Smith	{1, 2}
{185, 186}	Brown	{1, 2, 3, 4}

Darstellung nutzt die Unabhängigkeit der möglichen Werte der verschiedenen Felder aus (hier, ohne Angabe von Wahrscheinlichkeiten).

Weiteres Beispiel

Name	Vogel	Art
Mary	Vogel-1	{Fink: 0.8, Tukan: 0.2}
Susan	Vogel-2	{Nachtigall: 0.65, Tukan: 0.35}
Paul	Vogel-3	{Kolibri: 0.55, Tukan: 0.45}

Mögliche Welten:

N	V	A
M	1	F
S	2	N
P	3	K

0.286

N	V	A
M	1	F
S	2	N
P	3	T

0.234

N	V	A
M	1	F
S	2	T
P	3	K

0.154

N	V	A
M	1	F
S	2	T
P	3	T

0.126

N	V	A
M	1	T
S	2	N
P	3	K

0.0715

N	V	A
M	1	T
S	2	N
P	3	T

0.0585

N	V	A
M	1	T
S	2	T
P	3	K

0.0385

N	V	A
M	1	T
S	2	T
P	3	T

0.0315

Berechnung der **Wahrscheinlichkeiten** durch Produkt der Wahrscheinlichkeiten der einzelnen Ausprägungen (Möglichkeiten). Z.B. Für erste Welt (FNK): $0.8 \times 0.65 \times 0.55 = 0.286$

Alternative Repräsentation: Block-Independent Disjoint (BID) Relationen

RID	SSN	P
t_1	185	0.7
t_1	785	0.3
t_2	185	0.8
t_2	186	0.2

RID	N	P
t_1	Smith	1
t_2	Brown	1

RID	M	P
t_1	1	0.9
t_1	2	0.1
t_2	1	0.25
t_2	2	0.25
t_2	3	0.25
t_2	4	0.25

Interpretation:

- Die Tupel in einem Block mit gleichem Schlüssel RID sind **disjoint**. Jede Welt enthält ein Tupel pro Block, d.h. die Tupel aus einem Block schließen sich gegenseitig aus.
- Blöcke sind unabhängig voneinander.
  Die Wahlmöglichkeiten von Tupeln aus verschiedenen Blöcken sind unabhängig voneinander. Die aggregierte Wahrscheinlichkeit für eine Welt das erste Tupel aus dem ersten Block der Relationen zu enthalten ist $0.7 \times 1 \times 0.9 = 0.63$.

Tuple-Independent Datenbank

TI-Datenbanken sind BID-Datenbanken, in denen jeder Block genau ein Tupel hat.
TI-Datenbanken sind die einfachsten und am häufigsten anzutreffendes Modell für probabilistische Daten.

RID	SSN	P
t_1	185	0.7
t_2	185	0.8

RID	N	P
t_1	Smith	1
t_1	Brown	1

RID	M	P
t_1	1	0.9
t_2	2	0.25

Interpretation:

- Jedes Tupel befindet sich in einer beliebigen Welt mit Wahrscheinlichkeit $P(t)$
- Eine Relation mit n Tupeln, wobei jede Wahrscheinlichkeit < 1 ist, hat 2^n mögliche Welten, da ein Tupel in einer Welt enthalten sein kann oder nicht.
- Das Beispiel oben hat 2^4 mögliche Welten

Sind BID-Datenbanken ausreichend?

Was ist mit **Abhängigkeiten** zwischen Blöcken?

Im Beispiel: SSN sollte nicht mehrfach vergeben werden, d.h. Welten in denen sowohl t_1 als auch t_2 die SSN 185 haben sollten nicht erlaubt sein.

RID	SSN	P
t_1	185	0.7
t_1	785	0.3
t_2	185	0.8
t_2	186	0.2

RID	SSN	ϕ
t_1	185	$X = 1$
t_1	785	$X = 2$
t_2	185	$Y = 1 \wedge X \neq 1$
t_2	186	$Y = 2$

Idee: Benutze **Zufallsvariablen**, um Abhängigkeiten zwischen Tupeln zu modellieren.

- Vermeide Welten, in denen t_1 und t_2 die SSN 185 besitzen, durch sich ausschließende Zuweisungen für Variable X.
- Die Wahrscheinlichkeit von Tupeln wird nun durch die Wahrscheinlichkeit der Variablen modelliert (nächste Folie).

Probabilistic-Conditional-Datenbanken (PC-Datenbanken)

RID	SSN	ϕ
t_1	185	$X = 1$
t_1	785	$X = 2$
t_2	185	$Y = 1 \wedge X \neq 1$
t_2	186	$Y = 2$

RID	SSN	P
X	1	0.7
X	2	0.3
Y	1	0.8
Y	2	0.2

Interpretation:

- Die Tabelle rechts enthält die Wahrscheinlichkeitsverteilungen der einzelnen Zufallsvariablen.
- Jede Zuweisung (Assignment/Valuation) der Variablen definiert eine Welt, deren Wahrscheinlichkeit durch das Produkt der Wahrscheinlichkeiten der Variablen-Zuweisungen gegeben ist.
- Jedes Tupel t ist abhängig von der Erfüllbarkeit der Formel $\phi(t)$ und ist in den Welten vertreten mit Zuweisungen die $\phi(t)$ erfüllen.

TIs und BIDs als Spezialfälle von PCs

Wir betrachten wieder Beispiel TI-Datenbank von zuvor:

RID	SSN	P
t_1	185	0.7
t_2	185	0.8

RID	N	P
t_1	Smith	1
t_2	Brown	1

RID	M	P
t_1	1	0.9
t_2	2	0.25

Diese Datenbank kann als PC-Datenbank wie folgt beschrieben werden:

RID	SSN	ϕ	P
t_1	185	s_1	0.7
t_2	185	s_2	0.8

RID	N	ϕ	P
t_1	Smith	n_1	1
t_2	Brown	n_2	1

RID	M	ϕ	P
t_1	1	m_1	0.9
t_2	2	m_2	0.25

Possible-World-Semantik

Possible-World-Semantik: Gegeben eine Datenbank $\mathbf{W} = \{W_1, \ldots, W_n\}$ und eine Anfrage q, das Ergebnis der Anfrage ist $q(\mathbf{W}) = \{q(W_1), \ldots, q(W_n)\}$.

Anfragen auf unvollständigen Datenbanken

Gegeben eine Anfrage q und eine unvollständige Datenbank $\mathbf{W}$:

- Eine Antwort t heißt **sicher (certain)**, falls $\forall\, W_i \in \mathbf{W} : t \in q(W_i)$
- Eine Antwort t heißt **möglich (possible)**, falls $\exists W_i \in \mathbf{W} : t \in q(W_i)$

W_1		
SSN	Name	M
185	Smith	1
185	Brown	1

W_2		
SSN	Name	M
185	Smith	1
185	Brown	2

W_3		
SSN	Name	M
185	Smith	1
185	Brown	3

W_4		
SSN	Name	M
185	Smith	1
185	Brown	4

W_5		
SSN	Name	M
185	Smith	1
186	Brown	1

W_6		
SSN	Name	M
185	Smith	1
186	Brown	2

Sei $\mathbf{W} = \{W_1, \ldots, W_6\}$

- Die Anfrage $\exists N\ \exists M$: Census(S,N,M) hat die sichere Antwort (185) und mögliche Antworten (185) und (186).
- Die Anfrage $\exists S\ \exists M$: Census(S,N,M) hat (Smith) und (Brown) als sichere und mögliche Antworten.

Anfragen auf unvollständigen Datenbanken

Gegeben eine Anfrage q und eine probabilistische Datenbank $(\mathbf{W},P)$: Die **Wahrscheinlichkeit eines Antwort-Tupels** t ist:

$$P(t) = \sum \{P(W_i) \mid W_i \in \mathbf{W}, t \in q(W_i)\}$$

W_1: $P(W_1) = 0.1$

SSN	Name	M
185	Smith	1
185	Brown	1

W_2: $P(W_2) = 0.1$

SSN	Name	M
185	Smith	1
185	Brown	2

W_3: $P(W_3) = 0.1$

SSN	Name	M
185	Smith	1
185	Brown	3

W_4: $P(W_4) = 0.1$

SSN	Name	M
185	Smith	1
185	Brown	4

W_5: $P(W_5) = 0.3$

SSN	Name	M
185	Smith	1
186	Brown	1

W_6: $P(W_6) = 0.3$

SSN	Name	M
185	Smith	1
186	Brown	2

Sei $\mathbf{W} = \{W_1, \ldots, W_6\}$

- $\exists N\ \exists M$: Census(S, N, M): P(185) = 1 und P(186)=0.6
- $\exists S\ \exists M$: Census(S, N, M): P(Smith)= P(Brown) = 1

Possible-Answer-Set vs. Possible-Tuple Semantik

Possible-Tuple Semantik (Wird hauptsächlich betrachtet)

Im Sinne der Definition auf vorheriger Folie. Ein Tupel ist eine mögliche Antwort, falls es eine mögliche Welt gibt, in der das Tupel auftritt. Wahrscheinlichkeit des Tupels gegeben durch Summe der Wahrscheinlichkeiten der Welten, für die es im Ergebnis liegt. Repräsentation kompakt und angenehm für Anwender: Liste oder Tabelle der Tupel geordnet nach ihrer Wahrscheinlichkeit:

$$q^{rank} = [(t_i, P(t_i)), (t_j, P(t_j)), \ldots] \quad \text{mit } P(t_i) \geq P(t_j)$$

Possible-Answer-Set Semantik

Hier werden Ergebnismengen betrachtet, keine einzelnen Tupel. Wahrscheinlichkeit einer Ergebnismenge ist die Summe der Wahrscheinlichkeiten der Welten, die genau diese Menge als Ergebnis haben: $P(A) = \sum\{P(W)|W \in \mathbf{W} \wedge A = q(W)\}$. Gesamtergebnis wird auch als q^{pwd} bezeichnet.

N	V	A
M	1	F
S	2	N
P	3	K

0.286

N	V	A
M	1	F
S	2	N
P	3	T

0.234

N	V	A
M	1	F
S	2	T
P	3	K

0.154

N	V	A
M	1	F
S	2	T
P	3	T

0.126

N	V	A
M	1	T
S	2	N
P	3	K

0.0715

N	V	A
M	1	T
S	2	N
P	3	T

0.0585

N	V	A
M	1	T
S	2	T
P	3	K

0.0385

N	V	A
M	1	T
S	2	T
P	3	T

0.0315

Welche Arten wurden beobachtet?

$\exists N \exists V : Beobachtungen(N,V,A)$

A	Wahrscheinlichkeit P
F	0.80
T	0.714
N	0.65
K	0.55

Für Ergebnis T(ukan):

$0.234 + 0.154 + 0.126 + 0.0715 + 0.0585 + 0.0385 + 0.0315 = 0.714$

Anfrageevaluierung

Problemstellung:
Für eine gegebene Anfrage und probabilistische Datenbank (D,P), berechne für jedes mögliches Anfragetupel t dessen Wahrscheinlichkeit $P(t)$, d.h. $q^{rank}(D)$.

Wir betrachten im Folgenden nur Tuple-Independent-Datenbanken und einfache konjunktive Anfragen.

Anmerkung zur Notation: Wenn die Tupel einer Datenbank unabhängig voneinander sind, so nennt man in Literatur die Datenbank auch extensional.

Beispiel

Gegeben eine Datenbank D mit zwei Tabellen $S(A,B)$ und $T(C,D)$.

$S =$

	A	B	
s_1	'm'	1	0.8
s_2	'n'	1	0.5

$T =$

	C	D	
t_1	1	'p'	0.6

Welt	Wahrscheinlichkeit
$D_1 = \{s_1, s_2, t_1\}$	0.24
$D_2 = \{s_1, t_1\}$	0.24
$D_3 = \{s_2, t_1\}$	0.06
$D_4 = \{t_1\}$	0.06
$D_5 = \{s_1, s_2\}$	0.16
$D_6 = \{s_1\}$	0.16
$D_7 = \{s_2\}$	0.04
$D_8 = \emptyset$	0.04

Anfrage: $q(u) \leftarrow S(x,y),T(z,u),y = z$

$q^{pwd}(D) =$

Antwort	P
{'p'}	0.54
$\emptyset$	0.46

$q^{rank}(D) =$

Antwort	P
'p'	0.54

Weitere Anfrage

$$q_1(x) \leftarrow S(x,y), T(z,y), y = z$$

Im Sinne der Possible-Answer-Set-Semantik:

$q_1^{pwd}(D) =$

Antwort	Wahrscheinlichkeit
{'m', 'n'}	0.24
{'m'}	0.24
{'n'}	0.06
$\emptyset$	0.46

Im Sinne der Possible-Tuple-Semantik:

$q_1^{rank}(D) =$

Antwort	Wahrscheinlichkeit
{'m'}	0.48
{'n'}	0.30

Betrachten wir wieder Beispiel von zuvor. Die Tupelvariablen s_i und t_i werden nun als **Zufallsvariablen** in **einem dem Tupel t angehefteten Ausdruck $e(t)$** in einem "besonderen" Attribut E interpretiert.

$S =$

A	B	E
'm'	1	s_1
'n'	1	s_2

$T =$

C	D	E
1	'p'	t_1

Bei einem Join bzw. Kreuzprodukt, werden die Ausdrücke e_i der beteiligten Tupel kombiniert, wie folgt:

Auswertung von Join $S \bowtie_{B=C} T$ ergibt:

A	B	C	D	E
'm'	1	1	'p'	$s_1 \wedge t_1$
'n'	1	1	'p'	$s_2 \wedge t_1$

Bei einer Projektion (Achtung: Mengensemantik):

$\pi_D(S \bowtie_{B=C} T)$ ergibt:

D	E
'p'	$(s_1 \wedge t_1) \vee (s_2 \wedge t_1)$

Intensionale Anfrageauswertung

Wie die Ausdrücke in während einer Anfrageverarbeitung "mitgeführt" werden, wird durch folgende Definition beschrieben.

Selektion:

$$e_{\sigma_c}(t) = \begin{cases} e(t) & \text{falls } c(t)\text{true} \\ \bot & \text{sonst} \end{cases}$$

Projektion:

$$e_{\pi_A}(t) = \bigvee_{t':\pi_A(t')=t} e(t')$$

Kreuzprodukt:

$$e(t,t') = e(t) \wedge e(t')$$

$\bot$ bezeichnet hier das unmögliche Ereignis, d.h. $P(\bot) = 0$

Intensionale Anfrageauswertung

Ausdruck

$$e = (s_1 \wedge t_1) \vee (s_2 \wedge t_1)$$

Wahrheitstabelle zu Ausdruck e:

s_1	s_2	t_1	e
0	0	0	0
0	0	1	0
0	1	0	0
0	**1**	**1**	**1**
1	0	0	0
1	**0**	**1**	**1**
1	1	0	0
1	**1**	**1**	**1**

Betrachte Wahrheitstabelle. Ausdruck e evaluiert also zu 1 für die Einträge $(0,1,1)$, $(1,0,1)$ und $(1,1,1)$.

$$P((0,1,1)) = (1-P(s_1)) \times P(s_2) \times P(t_1) = 0.2 \times 0.5 \times 0.6 = 0.06$$

$$P((1,0,1)) = P(s_1) \times (1-P(s_2)) \times P(t_1) = 0.8 \times 0.5 \times 0.6 = 0.24$$

$$P((1,1,1)) = P(s_1) \times P(s_2) \times P(t_1) = 0.8 \times 0.5 \times 0.6 = 0.24$$

Insgesamt (d.h. Summe) also: $P(e) = 0.54$

Komplexität des Problems

Für eine Boolesche Formel mit n Variablen, gibt es zwischen 0 und 2^n Belegungen, die die Formel erfüllen.
Manche Anfragen (mehr dazu später) können in PTIME berechnet werden, für andere Anfragen ist das Problem #P-vollständig.

Klasse #P ("sharp P"): Klasse von "Zählproblemen"

- NP = Klasse von Entscheidungsproblemen, hier, bzgl. SAT, "Gibt es eine Belegung, die die Formel wahr macht?"
- #P = Klasse von Problemen der Form, "Wie viele Belegungen erfüllen die Formel?" (#SAT)

Ein Problem aus #P ist mindestens so schwierig, wie das entsprechende Entscheidungsproblem aus NP.

Theorem [Leslie Valiant, 1979]: **#SAT ist #P-vollständig.**

Beobachtungen

Idee: Extensionale Evaluierung: Wir betrachten einfache Wahrscheinlichkeiten der Tupel während der Anfrageausführung und keine (komplexen) Formeln.

- Jedem Tupel, insbesondere in den Zwischenschritten, wird eine Wahrscheinlichkeit zugewiesen, d.h. eine "Zahl" aus $[0, 1]$.
- Wie mit diesen Wahrscheinlichkeiten P umgegangen wird, wird wieder wie vorhin bei der intensionalen Auswertung via den Operatoren der relationalen Algebra definiert.

Extensionale Operatoren

Selektion:

$$P_{\sigma_c}(t) = \begin{cases} P(t) & \text{falls } c(t)\text{true} \\ 0 & \text{sonst} \end{cases}$$

Projektion:

$$P_{\pi_A}(t) = 1 - \prod_{t' : \pi_A(t') = t} (1 - P(t'))$$

Kreuzprodukt:

$$P(t, t') = P(t) \times P(t')$$

Extensionale Operatoren: Join (bzw. Kreuzprodukt ähnlich)

A	P
a_1	p_1
a_2	p_2
a_3	p_3

$\bowtie$

A	B	P
a_1	b_1	q_1
a_1	b_2	q_2
a_2	b_3	q_3
a_2	b_4	q_4
a_2	b_5	q_5

$=$

A	B	P
a_1	b_1	$p_1 \times q_1$
a_1	b_2	$p_1 \times q_2$
a_2	b_3	$p_2 \times q_3$
a_2	b_4	$p_2 \times q_4$
a_2	b_5	$p_2 \times q_5$

Extensionale Operatoren: Projektion

Relation $S(A,B)$:

A	B	P
a_1	b_1	q_1
a_1	b_2	q_2
a_2	b_3	q_3
a_2	b_4	q_4
a_2	b_5	q_5

Projektion auf Attribut A: $\pi_A(S)$

A	P
a_1	$1-(1-q_1)\times(1-q_2)$
a_2	$1-(1-q_3)\times(1-q_4)\times(1-q_5)$

Extensionale Operatoren: Selektion

Relation $S(A,B)$:

A	B	P
a_1	b_1	q_1
a_1	b_2	q_2
a_2	b_3	q_3
a_2	b_4	q_4
a_2	b_5	q_5

Selektion $\sigma_{A=a_2}(S)$:

A	B	P
a_2	b_3	q_3
a_2	b_4	q_4
a_2	b_5	q_5

Beispiel: Extensionale Evaluierung - Problem

Betrachten wir die die Anfrageausführung zu $\pi_D(S \bowtie_{B=C} T)$ über folgende Tabellen (wie weiter vorne):

$S =$

A	B	P
'm'	1	0.8
'n'	1	0.5

$T =$

C	D	P
1	'p'	0.6

Zuerst der innere Teil, also der Join $S \bowtie_{B=C} T$, ergibt:

A	B	C	D	P
'm'	1	1	'p'	$0.8 \times 0.6 = 0.48$
'n'	1	1	'p'	$0.5 \times 0.6 = 0.30$

Nun noch darauf die Projektion auf Spalte D, ergibt:

D	P
'p'	$(1 - (1 - 0.48)(1 - 0.3)) = 0.636$

Beispiel: Extensionale Evaluierung - Problem

Betrachten wir die die Anfrageausführung zu $\pi_D(S \bowtie_{B=C} T)$ über folgende Tabellen (wie weiter vorne):

$S =$

A	B	P
'm'	1	0.8
'n'	1	0.5

$T =$

C	D	P
1	'p'	0.6

Zuerst der innere Teil, also der Join $S \bowtie_{B=C} T$, ergibt:

A	B	C	D	P
'm'	1	1	'p'	$0.8 \times 0.6 = 0.48$
'n'	1	1	'p'	$0.5 \times 0.6 = 0.30$

Nun noch darauf die Projektion auf Spalte D, ergibt:

D	P
'p'	$(1 - (1 - 0.48)(1 - 0.3)) = 0.636$

Hier stimmt etwas nicht! Die Wahrscheinlichkeit für Antwort 'p' sollte 0.54 sein!

Beispiel: Extensionale Evaluierung - Jetzt OK

Gleiche Tabellen wie zuvor und äquivalente Anfrage, $\pi_D(\pi_B(S) \bowtie_{B=C} T)$

$S =$

A	B	P
'm'	1	0.8
'n'	1	0.5

$T =$

C	D	P
1	'p'	0.6

Zuerst die Projektion von S auf Attribut B, d.h. $\pi_B(S)$, ergibt:

B	P
1	$(1-(1-0.8)(1-0.5)) = 0.9$

Nun der Join, also $\pi_D(\pi_B(S) \bowtie_{B=C} T)$, ergibt:

B	C	D	P
1	1	'p'	$0.9 \times 0.6 = 0.54$

Nun noch darauf die Projektion auf Spalte D, ergibt:

D	P
'p'	0.54

Weiteres Beispiel

$R(x)=$

x	P
a_1	p_1
a_2	p_2
a_3	p_3

$S(x,y)=$

x	y	P
a_1	b_1	q_1
a_1	b_2	q_2
a_2	b_3	q_3
a_2	b_4	q_4
a_2	b_5	q_5

Anfrage: $q() \leftarrow R(x), S(x,y)$, in SQL ausgedrückt:

SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x

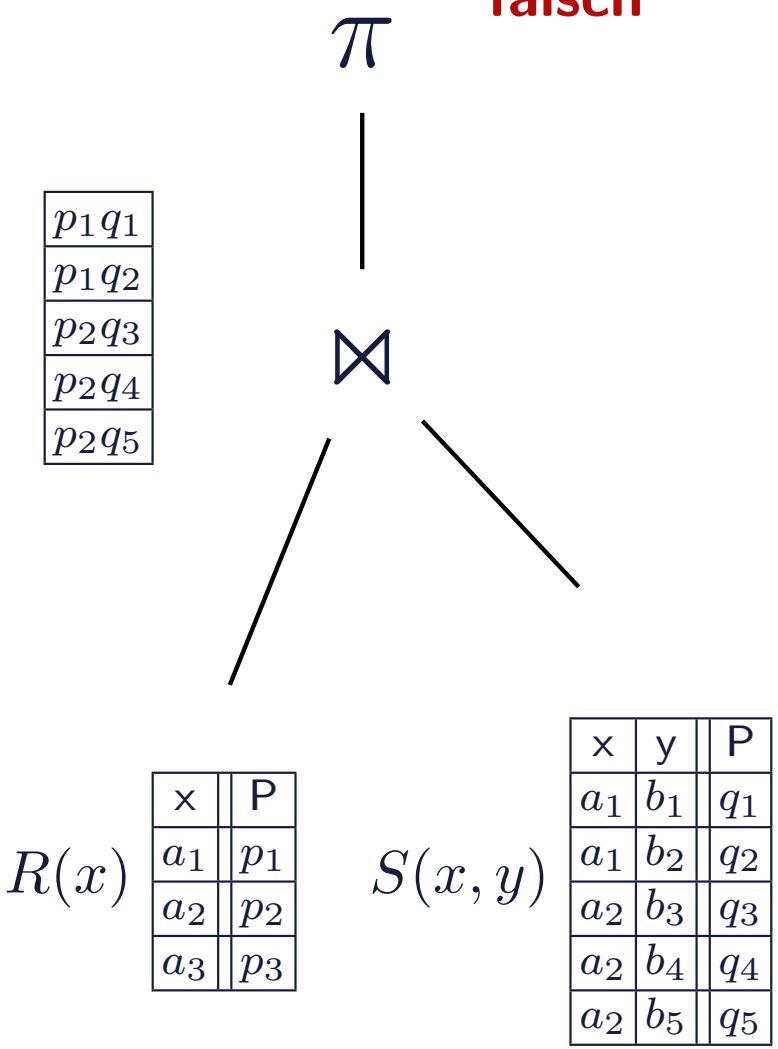
Wie hoch ist also die Wahrscheinlichkeit, dass es eine Antwort gibt?

$$P(q) = 1\text{–}[1 - p_1 \times (1 - (1 - q_1) \times (1 - q_2))] \times [1 - p_2 \times (1 - (1 - q_3) \times (1 - q_4) \times (1 - q_5))]$$

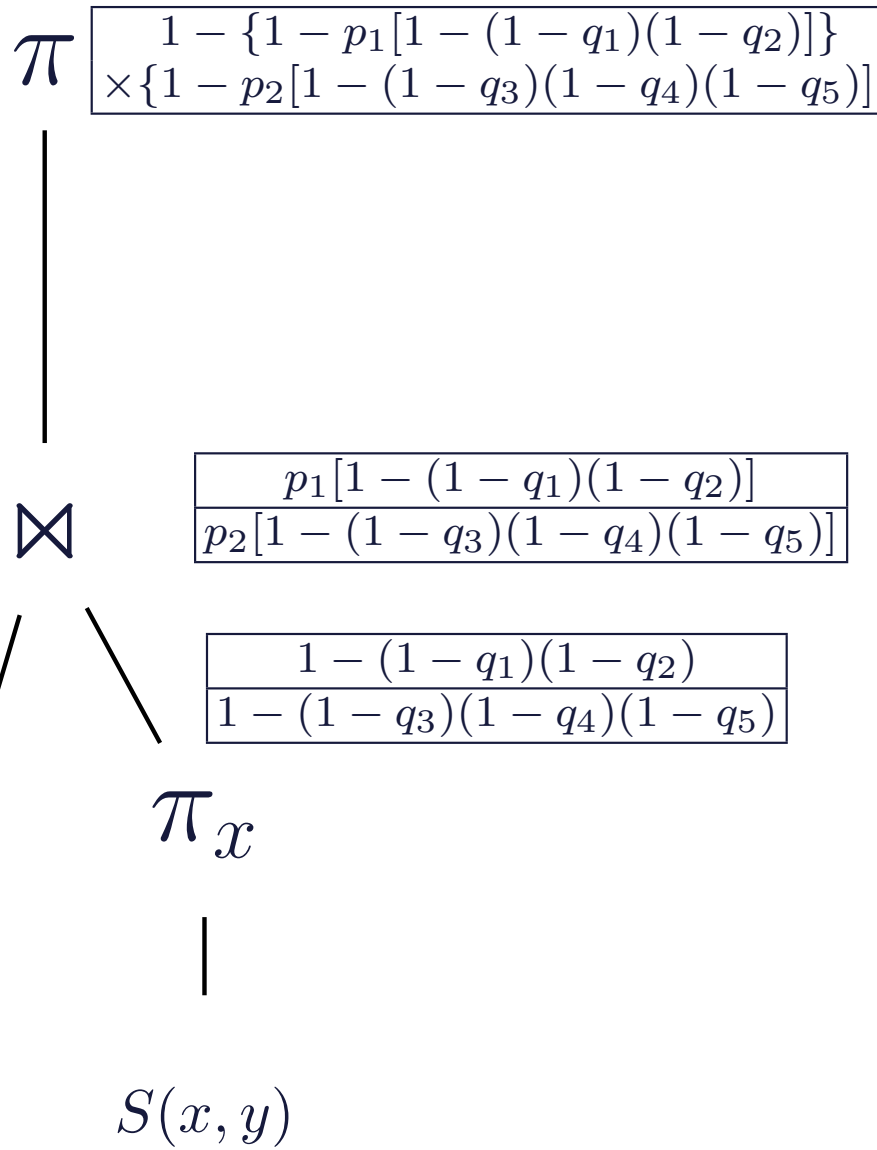
Weiteres Beispiel (Cont'd)

$$1-(1-p_1q_1)(1-p_1q_2)(1-p_2q_3)(1-p_2q_4)(1-p_2q_5)$$

falsch



richtig



Sichere und Unsichere Anfragen

Wir haben gesehen, dass es für eine Anfrage Pläne geben kann, für die die extensionale Auswertung die korrekten Wahrscheinlichkeiten liefert.

Einen Plan, der die korrekte Wahrscheinlichkeit berechnet nennt man auch **sicher (safe)**.

Anfragen, die einen sicheren Plan besitzen, nennt man **sichere Anfragen** (safe queries).

Umgekehrt nennt man Anfragen, die keinen sicheren Plan haben **unsichere Anfragen** (unsafe queries). Unsichere Anfragen kann man nicht effizient berechnen.

Anmerkung: Nicht verwechseln mit sicheren/unsicheren Anfragen im Tupel/Domänenkalkül aus InSy.

Extensionale Pläne ist Postgresql

Mit extensionalen Plänen kann man direkt via SQL arbeiten.
Für die Wahrscheinlichkeiten wird eine extra Spalte benutzt, hier P.

$R(A)=$

A	P
a_1	p_1
a_2	p_2
a_3	p_3

$S(A,B)=$

A	B	P
a_1	b_1	q_1
a_1	b_2	q_2
a_2	b_3	q_3
a_2	b_4	q_4
a_2	b_5	q_5

Dann z.B. Join wie gewohnt, dabei werden aber die Werte für P mitgeführt, entsprechend der Regeln von zuvor.

SELECT R.A, S.B, **R.P*S.P**
FROM R, S
WHERE R.A = S.A

Für Projektion, sieht dies in SQL wie folgt aus

SELECT S.A, **1.0-prod(1.0 - S.p)**
FROM S
GROUP BY S.A

Es gibt keine "Produkt" Aggregationsfunktion. Folgender Code definiert diese (aus Folien von Dan Suciu):

```
create or replace function combine_prod(float, float)
       returns float as 'select $1 * $2' language SQL;
create or replace function final_prod(float)
      returns float as 'select $1' language SQL;
drop aggregate if exists prod (float);
create aggregate prod (float)
(
sfunc = combine_prod,
stype = float,
finalfunc = final_prod,
initcond = '1.0'
);
```