Ereignismuster für die Verwaltung von komplexen Tupelereignissen in Probabilistischen Datenbanken

Sebastian Lehrack
Brandenburgische Technische Universität Cottbus
Institut für Informatik, Postfach 10 13 44
D-03013 Cottbus, Deutschland
slehrack@informatik.tu-cottbus.de

ABSTRACT

Probabilistische Datenbanken haben sich als adäquate Technik zur Verwaltung und Verarbeitung von umfangreichen unsicheren Datenmengen etabliert. Eine der größten Herausforderungen für probabilistische Datenbanken ist eine effiziente Anfrageverarbeitung. Eine zentrale Rolle spielen dabei Ableitungsformeln, welche komplexe Tupelereignisse in Form von aussagenlogischen Formeln verkörpern. Eine *direkte* Abspeicherung und Verarbeitung von komplexen logischen Formeln wird von relationalen Datenbanksystemen jedoch nicht unterstützt. Diese Arbeit stellt Ereignismuster als geeignetes Mittel vor, um Ableitungsformeln mittels eines RBDMS verwalten zu können.

1. MOTIVATION

Probabilistische Datenbanken standen in den letzten Jahren im Mittelpunkt intensiver Untersuchungen (für einen Überblick verweisen wir auf [18]). In einer solchen Datenbank gehört ein Tupel lediglich mit einer gewissen Wahrscheinlichkeit zu einer Datentabelle oder zu einem Anfrageergebnis. Die Wahrscheinlichkeit drückt dabei entweder eine *Unsicherheit* über die Datengrundlage oder die Anfrageantwort aus.

Eine der größten Herausforderungen für probabilistische Datenbanken ist eine effiziente Anfrageverarbeitung. Ein zentrales Konzept sind dabei Ableitungsformeln, welche komplexe Tupelereignisse in Form von aussagenlogischen Formeln darstellen [4]. Eine direkte Abspeicherung und Verarbeitung von komplexen logischen Formeln wird von relationalen Datenbanksystemen jedoch *nicht* unterstützt. Diese Arbeit stellt Ereignismuster als geeignetes Mittel vor, um Ableitungsformeln in einer strukturierten Form mittels eines RBDMS verwalten zu können.

Fortlaufendes Beispiel: Die grundlegenden Ideen werden in den nächsten Kapiteln anhand eines fortlaufenden Beispielszenarios aufgezeigt. Dieses Szenario ist motiviert durch die Neuentwicklung des CISAR-Projektes[1] [5], welches als internet-basiertes Geo-Informationssystem für Archäologie und Gebäudegeschichte entwickelt worden ist. Die hier vorgestellten Techniken werden umfassend in dem Nachfolgesystem OpenInfRA eingesetzt [17].

Arte

tid	aid	type	sond	age	ET_{Arte}
t_1	art1	vase fragment	3	300	$(X_1 = t_1)$
t_2	art2	spear head	10	500	$(X_2 = t_2)$
t_3	art3	vase fragment	4	300	$(X_3 = t_3)$

ArteExp

tid	expert	aid	culture	conf	$ET_{ArteExp}$
t_4	Peter	art1	roman	0.3	$(X_4 = t_4)$
t_5	Peter	art1	greek	0.4	$(X_4 = t_5)$
t_6	Kathleen	art1	roman	0.4	$(X_5 = t_6)$
t_7	John	art2	egyptian	0.6	$(X_6 = t_7)$

ArteMat

tid	method	aid	culture	conf	$ET_{ArteMat}$
t_8	XRF	art1	roman	0.3	$(X_7 = t_8)$
t_9	XRF	art1	greek	0.3	$(X_7 = t_9)$
t_{10}	ICS-MS	art2	punic	0.8	$(X_8 = t_{10})$
t_{11}	XRF	art2	egyptian	0.5	$(X_9 = t_{11})$

Figure 1: Datentabellen *Arte*, *ArteExp* und *ArteMat* des fortlaufenden Beispielszenarios (die unterstrichenen Attribute kennzeichnen den jeweiligen Ereignisschlüssel (siehe Kap. (2)))

In dem stark vereinfachten Beispielszenario werden die deterministische Tabelle *Artefakte* (*Arte*) und die zwei probabilistischen Tabellen *Artefakte klassifiziert bei Experten*[2] (*ArteExp*) und *Artefakte klassifiziert bei Material* (*ArteMat*) verwendet, siehe Abb. (1). In der Datentabelle *Arte* werden Informationen über mehrere Artefakten gespeichert, welche während einer archäologischen Ausgrabung gefunden worden sind. Dabei wird mittels der *Sondage*-Nummer (Attribut *sond*) die geographische Fundstelle eines Artefaktes beschrieben.

Zusätzlich geben mehrere Spezialisten verschiedene Expertisen über die Ursprungskultur eines Artefakts (Attribut *culture*) ab, siehe Tabelle *ArteExp*. Diese Einschätzungen werden mit einem Konfidenzwert annotiert (Attribut *conf*),

[1] http://www.dainst.org/en/project/cisar/

[2] Die Spalten *tid*, *conf* und $ET_{(...)}$ gehören nicht zu den eigentlichen Datentabellen. Mittels der Spalte *tid* werden Tupel adressiert. Die Bedeutung von *conf* und $ET_{(...)}$ wird in den nächsten Abschnitten erläutert.

24^{th} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 29.05.2012 - 01.06.2012, Lübbenau, Germany.

```
select aid, type, culture
from    (  select aid, culture
           from ArteExp
         union
           select aid, culture
           from ArteMat
        ) origin
     inner join
        ( select *
          from Arte
        ) prop
     on ( origin.aid = prop.aid )
```

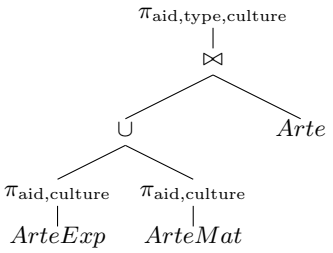

Figure 2: Beispielanfrage Q_e als QSQL2-Anfrage und als abstrahierte Algebraanfrage

welche die Wahrscheinlichkeit ausdrückt, dass das entsprechende Artefakt zu der bestimmten Kultur gehört. Neben den subjektiven Expertenmeinungen werden auch objektive Methoden einbezogen. Diese *archäometrischen Methoden* (z.B. XRF und ICS-MS[3]) basieren auf einer Materialanalyse. In Kombinationen mit den Fundstellen und dem Artefaktalter kann die Materialzusammensetzung wichtige Hinweise auf die Ursprungskultur geben, welche dann ebenfalls mittels Konfidenzwerten quantifiziert werden.

Basierend auf den eingeführten Datentabellen soll exemplarisch die folgende Anfrage Q_e bearbeitet werden: *Bestimme alle Artefakte mit ihren jeweiligen Typ und ihren möglichen Ursprungskulturen.* Um diese Anfrage zu beantworten wird sie zunächst in der Anfragesprache QSQL2 [8] formuliert, siehe Abb. (2). Anschließend wird von dem eigentlichen SQL-Syntax abstrahiert, indem sie in Form einer Algebraanfrage in den nächsten Kapiteln weiter verarbeitet wird.

Die Arbeit ist wie folgt gegliedert. In Kap. (2) werden zunächst die Grundlagen für die weiteren Betrachtungen gelegt. Danach steht der wesentliche Beitrag dieser Arbeit in Form der Ereignismuster in Kap. (3) im Mittelpunkt. In den Kap. (4) und (5) wird die Arbeit mit einer Diskussion über verwandte Arbeiten und einer Zusammenfassung abgeschlossen. Der Beweis des Satzes (1) wird im Anhang (A) geführt.

2. GRUNDLAGEN

In einer probabilistischen Datenbanken werden mehrere möglichen Datenbankinstanzen, welche auch *Welten* genannt werden, gleichzeitig verwaltet und abgefragt. Dabei wird die „reale Welt“ als unbekannt angenommen. Um diese Unsicherheit abzubilden wird ein Wahrscheinlichkeitsmaß über der Menge aller möglichen Welten vereinbart. Konkret wird hier auf die Definition von Suciu et al. [18] zurück gegriffen.

Definition 1. Angenommen werden k Relationennamen: $R_1, \ldots, R_k$. Eine *unvollständige Datenbank* ist dann eine endliche Menge von Dateninstanzen $\mathbf{W} = \{W^1, W^2, \ldots, W^n\}$, wobei jede Dateninstanz (Welt) durch $W^i = (R_1^i, \ldots, R_k^i)$ beschrieben ist. Eine *probabilistische Datenbank* ist ein Wahrscheinlichkeitsraum $\mathbf{D} = (\mathbf{W}, \mathbf{P})$ über einer unvollständigen Datenbanken $\mathbf{W}$. Damit ist $\mathbf{P} : \mathbf{W} \rightarrow [0, 1]$ eine Funktion, sodass $\sum_{W \in \mathbf{W}} \mathbf{P}(W) = 1$ gilt. Es wird vorausgesetzt, dass $\forall W \in \mathbf{W} : \mathbf{P}(W) > 0$ gilt.

Im Allgemeinen ist die Semantik des Wahrscheinlichkeitsmaß $\mathbf{P}$ nicht vordefiniert. Konkret wird hier der Spezialfall der block-independent-disjoint Datenbanken (BID) [2] benutzt, da Tupel- *und* Attributunsicherheit untersützt werden soll. Eine BID ist eine probabilistische Datenbank in der die gegebenen Tupel in Blöcke unterteilt werden. Dabei kann ein Block sich nicht über mehrere Relationen erstrecken. Es wird vereinbart, dass alle Tupel innerhalb eines Blockes mit disjunkten Tupelereignissen verbunden sind. Ein Tupelereignis beschreibt das Vorhandensein oder das Nicht-Vorhandensein eines Tupel in einer beliebigen Welt. Wegen der Disjunktheit der Tupelereignisse kann maximal ein Tupel eines Blockes in einer bestimmten Welt vorhanden sein. Dagegen sind Tupel von verschiedenen Blöcken mit gegenseitig unabhängigen Tupelereignissen assoziiert.

Für die Definition von Blöcken werden Ereignisschlüssel in Form von Attributmengen angewendet[4]. So wird ein Block wird von Tupeln gebildet, welche die gleichen Werte für die Attribute des Schlüssel haben. Anschließend wird für jeden Block eine unabhängige Zufallsvariable X_k eingeführt. Das konkrete Eintreten einer Zufallsvariable ($X_k = t_{id}$) repräsentiert ein Tupelereignis und wird quantifiziert mit dem Konfidenzwert des Tupels t_{id}, siehe die Spalten *conf* und $ET_{(\ldots)}$ in der Abb. (1)[5]. Die kombinierte Wahrscheinlichkeitsverteilung aller Blockvariablen bilden schließlich $\mathbf{P}$. Um eine Algebraanfrage auf einer probabilistischen Datenbank auszuführen, wird folgende Anfragesemantik verwendet [18].

Definition 2. Sei Q eine Algebraanfrage und $\mathbf{D} = (\mathbf{W}, \mathbf{P})$ eine probabilistische Datenbank. Die Menge aller *möglichen Antworttupel* einer Anfrage Q ist definiert als $Q_{\text{poss}(\mathbf{W})} = \{t \mid \exists W \in \mathbf{W} : t \in Q(W)\}$. Zusätzlich werden die Eintrittswahrscheinlichkeiten aller möglichen Antworten in der Funktion $\Pr_Q : Q_{\text{poss}(\mathbf{W})} \rightarrow (0, 1]$ als $\Pr_Q(t) := \sum_{W \in \mathbf{W} : t \in Q(W)} \mathbf{P}(W)$. berechnet.

Dies bedeutet, dass die Anfrage Q konzeptionell in jeder Welt separat ausgeführt wird. Dann wird die Ergebnisrelation $Q_{\text{poss}(\mathbf{W})}$ gebildet, in dem alle möglichen Antworten der verschiedenen Auswertungen gesammelt werden. Zusätzlich wird die Eintrittswahrscheinlichkeit einer möglichen Antwort durch das Aufsummieren der Welten, in der das Antworttupel in der Antwort auftritt, gebildet. Offensichtlich gibt die Def. (2) nur die Semantik der Anfrageauswertung vor. Eine einzelne Auswertung in allen Welten ist praktisch

[3]XRF und ICP-MS stehen für die *x-ray fluorescence* und die *inductively coupled plasma mass spectrometry* Methode.

[4]In dem Beispielszenario werden die Ereignisschlüssel von *Arte*, *ArteExp* und *ArteMat* als $\{aid\}$, $\{exp, aid\}$ und $\{method, aid\}$ vereinbart, siehe Abb. (1).

[5]Da die Tabelle *Arte* deterministisch ist, wird $\mathrm{P}(X_1 = t_1) = \ldots = \mathrm{P}(X_3 = t_3) = 1$ gesetzt.

nicht umsetzbar, da die Anzahl aller Welten exponentiell in Datengröße anwachsen kann.

3. EREIGNISMUSTER

In diesem Kapitel wird die praktische Auswertung einer Algebraanfrage Q auf einer probabilistischen Datenbank diskutiert. Dabei wird das Konzept der Ableitungsformeln vorgestellt und eine neue Technik zur Verwaltung von komplexen Tupelereignissen eingeführt.

3.1 Ableitungsformeln

Entsprechend Def. (2) muss zur Auswertung einer Anfrage die Ergbeniswahrscheinlichkeit $\Pr_Q(t)$ für jedes Ergebnistupel berechnet werden. Fuhr und Röllecke schlugen in [4] vor, diese Wahrscheinlichkeiten mittels von *Ableitungsformeln* ϕ_t zu berechnen. Dabei werden alle Ableitungsformeln neben dem eigentlichen Datenanteil der Tupel verwaltet. Prinzipiell ist eine Ableitungsformel ϕ_t durch eine aussagenlogische Formel gegeben, welche mittels den Zufallsvariablen $(X_k = t_{id})$[6] und den logischen Operatoren $\wedge, \vee$ und $\neg$ gebildet wird. Fuhr und Röllecke haben geschlussfolgert, dass $\Pr_Q(t)$ durch Wahrscheinlichkeit $\mathbf{P}(\phi_t \equiv \text{true})$ ermittelt werden kann, d.h. $\Pr_Q(t) = \mathbf{P}(\phi_t \equiv \text{true})$. Somit kann $\Pr_Q(t)$ durch $\mathbf{P}(\phi_t \equiv \text{true})$ berechnet werden, ohne dass über alle Welten von $\mathbf{W}$ iteriert werden muss (siehe Def. (2)). Im Folgenden wird $\mathbf{P}(\phi_t \equiv \text{true})$ durch $\mathbf{P}(\phi_t)$ abgekürzt. Die Konstruktion von ϕ_t greift auf die Struktur der betrachteten Anfrage Q zurück.

Definition 3. Angenommen Q ist eine Algebraanfrage und $\mathbf{D} = (\mathbf{W}, \mathbf{P})$ ist eine probabilistische Datenbank. Die Ableitungsformel ϕ_t ist dann wie folgt rekursiv definiert:

$$
\begin{aligned}
Q \equiv R \;&:\; \phi_t := (X_k = t_{id}), \text{wenn } t \in \text{TupBl}(k) \\
Q \equiv \sigma_c(Q_1) \;&:\; \phi_t := \phi_{t_1} \\
Q \equiv \pi_{\mathcal{A}}(Q_1) \;&:\; \phi_t := \bigvee_{\hat{t} \in Q_1, \hat{t}[\mathcal{A}] = t} \phi_{\hat{t}} \\
Q \equiv Q_1 \bowtie Q_2 \;&:\; \phi_t := \phi_{t_1} \wedge \phi_{t_2} \\
Q \equiv Q_1 \cup Q_2 \;&:\; \phi_t := \phi_{t_1} \vee \phi_{t_2} \\
Q \equiv Q_1 \setminus Q_2 \;&:\; \phi_t := \phi_{t_1} \wedge \neg(\phi_{t_2})
\end{aligned}
$$

wobei X_k eine Blockvariable ist und $\text{TupBl}(k)$ alle Tupel eines Blockes k zurück gibt (siehe Kap. (2)). Falls t_i nicht in einer Eingabeanfrage existiert (d.h. $t_i \notin Q_i$), wird $\phi_{t_i} :=$ false gesetzt (siehe Vereinigung- und Differenzoperator).

Abb. (3) zeigt die Ableitungsformeln für die Ergebnistupel der Beispielanfrage Q_e. Die Wahrscheinlichkeit $\mathbf{P}(\phi_t)$ kann mit Standardalgorithmen berechnet werden (z.B. [12, 6]). Wie man dort sieht, besitzen Ereignistupel im Allgemeinen unterschiedliche Ableitungsformeln. Mehrere Verfahren (z.B. [4, 3, 13]) müssen eine komplexe Ableitungsformel für jedes einzelne Tupel, welches innerhalb der Anfrageverarbeitung entsteht, verwalten. Praktische Anwendungsszenarien haben jedoch bereits gezeigt, dass Ableitungsformel mit einer Größe von 10 MB für ein Ergebnistupel auftreten können [14]. Dementsprechend folgt der hier entwickelte Ansatz der Argumentation von Antova et al. [1] und Das Sarma et al. [16], dass die Verwaltung und die Verarbeitung von komplexen Ableitungsformeln durch relationale Datenbanksysteme nicht zielführend sind, da solche Systeme nicht für die Verarbeitung von komplexen logischen Formeln ausgelegt sind.

[6]Ein solches Ereignis wird auch als *Basisereignis* bezeichnet.

Algorithm 1: $\text{gen}(\Phi^{pa}, t)$

```
Data: Klauselmustermenge Φ^pa, Tupel t ∈ R^ev
Result: Formel in DNF kodiert als Φ_t
Φ_t := ∅;
foreach CP ∈ Φ^pa do
    C := (true);
    foreach ET_{R_i} in CP do
        if t[ET_{R_i}] ≠ null then
            C := C • t[ET_{R_i}];
        else
            C := C • (false);
        end
    end
    if simpl(C) ≠ false then
        Φ_t := Φ_t ∪ {simpl(C)};
    end
end
return Φ_t;
```

3.2 Verwaltung von Ableitungsformeln mittels Ereignismustern und Basisereignismengen

Um eine logische Formel in einer strukturierten Form zu verwalten wird sie in die bekannte disjunktive Normalform (DNF) transformiert.

Definition 4. Angenommen ϕ_t ist eine Ableitungsformel. Ein *Literal* ist gegeben durch ein negiertes oder unnegiertes Basisereignis von ϕ_t, d.h. $L \equiv \neg(X_k = t_{id})$ oder $L \equiv (X_k = t_{id})$. Eine Konjunktion von Literalen wird als *Klausel* C bezeichnet, d.h. $C = L_1 \wedge \ldots \wedge L_m$. Eine Formel ϕ_t^{dnf} in DNF ist eine Disjunktion von Klauseln, d.h. $\phi_t^{\text{dnf}} = C_1 \vee \ldots \vee C_r$, sodass $\phi_t \equiv \phi_t^{\text{dnf}}$.

Da relationale Standardoperatoren zur Manipulation von DNF-Formeln verwendet werden sollen (siehe Def. (6) unten), empfiehlt es sich DNF-Formeln in Form von Tupeln und Mengen zu kodieren.

Definition 5. Um eine Formel ϕ_t^{dnf} in DNF zu kodieren, wird eine *Menge von Klauseltupeln* (bezeichnet als Φ_t) benutzt, d.h. $\Phi_t := \{C_1, \ldots, C_r\}$, wobei C_i ein Klauseltupel ist. Ein Klauseltupel $C_i = (L_1, \ldots, L_{m_i})$ korrespondiert dann mit einer Klausel $L_1 \wedge \ldots \wedge L_{m_i}$ von ϕ_t^{dnf}.

Die grundlegende Idee des hier vorgestellten Ansatzes lässt sich dann in folgenden vier Schritten beschreiben:

(i) das Bilden einer Menge von Klauselmustern (gekennzeichnet als Φ^{pa}), welche direkt von der Algebraanfrage Q abgeleitet wird (d.h. unabhängig von den aktuellen Daten in der Datenbank),
(ii) das Generieren einer Menge relevanter Basisereignissen für jedes Ergebnistupel während der relationalen Anfrageverarbeitung (verwaltet in einer erweiterten Ergebnisdatenrelation R^{ev}),
(iii) die Konstruktion einer DNF-Formel kodiert als Φ_t für jedes Ergebnistupel mittels des Generierungsalgorithmus $\text{gen}(\Phi^{pa}, t)$, $t \in R^{ev}$ und
(iv) die Berechnung von $\mathbf{P}(\phi_t^{\text{dnf}})$ mit Hilfe eines Standardalgorithmus (z.B. [12, 6]).

Bevor der Generierungsalgorithmus $\text{gen}(\Phi^{pa}, t)$ diskutiert wird, werden zunächst die Bedeutung von Φ^{pa} und R^{ev} näher beleuchtet.

tid		aid	type	culture	ϕ_t	$\mathbf{P}(\phi_t)$
r_1	(t_4, t_6, t_8, t_1)	art1	vase fragment	roman	$([(X_4 = t_4) \vee (X_5 = t_6)] \vee (X_7 = t_8)) \wedge (X_1 = t_1)$	0.706
r_2	(t_5, t_9, t_1)	art1	vase fragment	greek	$[(X_4 = t_5) \vee (X_7 = t_9)] \wedge (X_1 = t_1)$	0.58
r_3	(t_7, t_{11}, t_2)	art2	spear head	egyptian	$[(X_6 = t_7) \vee (X_9 = t_{11})] \wedge (X_2 = t_2)$	0.8
r_4	(t_{10}, t_2)	art2	spear head	punic	$[(X_8 = t_{10}) \vee (False)] \wedge (X_2 = t_2)$	0.8

Figure 3: Die Ergebnistupel von Q_e mit ihren Ableitungsformeln (ϕ_t) und ihren Ergebniswahrscheinlichkeiten ($\mathbf{P}(\phi_t)$)

Mustermenge Φ^{pa}: Die Mustermenge Φ^{pa} besteht aus einer Menge von Klauselmustern $\{CP_1, \ldots, CP_l\}$. Ein Klauselmuster $CP = (ET_{R_{i_1}}, \ldots, ET_{R_{i_m}})$ ist wiederum gegeben durch ein Tupel von *Basisereignismustern*. Dabei symbolisiert ein Muster ET_{R_i} genau ein Basisereignis $(X_k = t_{id})$ der Basisrelation R_i. Falls z.B. das Klauselmuster $CP = (ET_{ArteExp}, ET_{ArteMat})$ betrachtet wird, verkörpert CP alle Klauseln in denen das erste Basisereignis aus der Relation $ArteExp$ stammt und das zweite Basisereignis aus $ArteMat$ genommen wird.

Basisereignisse in R^{ev}: Die Ereignisdatenrelation R^{ev} besteht aus allen Datentupeln, sowie aus jeweils einer Menge von Basisereignissen für jedes Datentupel. Dabei werden genau die Basisereignisse gespeichert, welche notwendig sind um die Ableitungsformel für ein bestimmtes Datentupel zu bilden. Zu diesem Zwecke werden die Datenrelationen durch zusätzliche Spalten erweitert, welche Basisereignisse speichern. Alle Basisereignisse einer spezifischen Zeile gehören dabei zu dem jeweiligen Datentupel. Jede neue Spalte wird mit einer Basisrelation R_i assoziiert und entsprechend mit einem Musternamen ET_{R_i} bezeichnet, siehe Abb. (1) und (4).

Algorithmus gen(Φ^{pa}, t): Nach der Konstruktion von Φ^{pa} und R^{ev} generiert der Algoritmus gen(Φ^{pa}, t) eine Ableitungsformel (kodiert als Φ^{pa}) für ein Tupel der Relation R^{ev}. Im Wesentlichen ersetzt der Algorithmus der Basisereignismuster mit den jeweiligen zuordenbaren Basisereignissen (siehe Zeilen 2 bis 11 in Algorithmus (1)). Die Vergleichskriterien sind durch die Musternamen ET_{R_i} und die korrespondieren Bezeichnungen der Ereignisspalten von R^{ev} gegeben. Der Operator $\bullet$ konkateniert zwei Tupel. Bevor eine erzeugte Klausel C zu der Ergebnisformel Φ_t hinzugenommen wird, werden alle Klauseln C logisch vereinfacht (siehe Zeilen 12 und 13). Hierfür werden logische Gesetze wie Idempotenz und Kontradiktion eingesetzt.

Die Klauselmustermenge Φ^{pa} und die Ereignisdatenrelation R^{ev} werden rekursiv über die Struktur der betrachteten Algebraanfrage Q definiert. Prinzipiell wird Φ^{pa} in einer Art und Weise konstruiert, dass die erzeugten Muster der grundlegenden Semantik der Def. (3) genügt und alle möglichen Ereignisse erzeugt werden können, die nötig sind um ϕ_t^{dnf} zu erzeugen. Exemplarisch wird die Ereignisdatenrelation der Beispielanfrage Q_e in Abb. (4) gezeigt.

Definition 6. Sei Q eine positive Algebraanfrage[7] und $\mathbf{D} = (\mathbf{W}, \mathbf{P})$ eine probabilistischen Datenbank. Die Klauselmustermenge Φ^{pa} und die Ereignisdatenrelation R^{ev} werden dann rekursiv mittels der folgenden Regel gebildet[8]:

$$
\begin{aligned}
Q \equiv R_i \;:\; & \Phi^{pa} := \{(ET_{R_i})\}, \\
& R^{ev} := \{t \bullet (X_k = t_{id}) \mid t \in R_i\} \\
Q \equiv \sigma_c(Q_1) \;:\; & \Phi^{pa} := \Phi_1^{pa}, R^{ev} := \sigma_c(R_1^{ev}) \\
Q \equiv \pi_{\mathcal{A}}(Q_1) \;:\; & \Phi^{pa} := \Phi_1^{pa}, \\
& R^{ev} := \pi_{\mathcal{A} \cup \{\text{all } ET_{R_i} \text{ columns}\}}(R_1^{ev}) \\
Q \equiv Q_1 \bowtie Q_2 \;:\; & \Phi^{pa} := \Phi_1^{pa} \times \Phi_2^{pa}, R^{ev} := R_1^{ev} \bowtie R_2^{ev} \\
Q \equiv Q_1 \cup Q_2 \;:\; & \Phi^{pa} := \Phi_1^{pa} \cup \Phi_2^{pa}, \\
& R^{ev} := R_1^{ev} \bowtie^{\text{full } \underline{\text{o}}\text{uter}} R_2^{ev}
\end{aligned}
$$

Um die endgültige Ableitungsformel ϕ_t^{dnf} für ein Ergebnistupel t zu bilden, werden die generierten Φ_t-Formeln aller Tupel in R^{ev} mit den selben Datenwerten wie das Ergebnistupel t disjunktiv verknüpft:

$$\phi_t^{\text{dnf}} := \bigvee_{\hat{t} \in R^{ev}, \hat{t}[\text{datAttr}] = t} \text{gen}(\Phi^{pa}, \hat{t}),$$

wobei datAttr die Menge aller Datenattribute der Ergebnisdatenrelation R^{ev} repräsentiert, d.h. alle Attribut ohne die jeweiligen ET_{R_i} Spalten[9].

Satz 1. Sei R^{ev} und ϕ_t^{dnf} konstruiert wie in Def. (6) angegeben, dann gilt (i) $Q_{\text{poss}(\mathbf{W})} = \pi_{\text{datAttr}}(R^{ev})$ und (ii) $\forall t \in Q_{\text{poss}(\mathbf{W})} : \Pr_Q(t) = \mathbf{P}(\phi_t^{\text{dnf}})$.

4. VERWANDTE ARBEITEN

Eine umfassende Monographie über probabilistische Datenbank wurde kürzlich von Suciu et al. [18] veröffentlicht. Daneben wurden in den letzten Jahren mehrere probabilistische Datenbanksysteme erfolgreich umgesetzt (z.B. [15, 7, 19]). In vorangegangenen Arbeiten [10, 11] des Autors wurde ein probabilistisches Daten- und Anfargemodell entworfen, welches Konzepte aus dem Gebiet des Information Retrievals mit Technologien der Datenbankwelt kombiniert. Die dabei

[7]Genau wie die Systeme [15], [19] und [7] fokussiert sich der hier vorgestellte Ansatz momentan auf Anfragen ohne Differenzoperationen.

[8]Um die DNF-Repräsentation von Def. (5) zu bewahren werden Tupel verflacht. Wenn z.B. $\Phi_1^{pa} = \{(L_1, L_2)\}$ und $\Phi_2^{pa} = \{(L_3, L_4)\}$ gegeben sind, dann ergibt sich $\Phi_1^{pa} \times \Phi_2^{pa} = \{(L_1, L_2, L_3, L_4)\}$ anstelle von $\Phi_1^{pa} \times \Phi_2^{pa} = \{((L_1, L_2), (L_3, L_4))\}$.

[9]Diese Notation von datAttr wird in der restlichen Arbeit weiter verwendet.

tid	aid	type	culture	$\mathtt{ET}_{ArteExp}$	$\mathtt{ET}_{ArteMat}$	$\mathtt{ET}_{Arte}$
(t_4, t_8, t_1)	art1	vase fragment	roman	$X_4 = t_4$	$X_7 = t_8$	$X_1 = t_1$
(t_5, t_9, t_1)	art1	vase fragment	greek	$X_4 = t_5$	$X_7 = t_9$	$X_1 = t_1$
(t_6, t_8, t_1)	art1	vase fragment	roman	$X_5 = t_6$	$X_7 = t_8$	$X_1 = t_1$
(t_7, t_{11}, t_2)	art2	spear head	egyptian	$X_6 = t_7$	$X_9 = t_{11}$	$X_2 = t_2$
(t_{10}, t_2)	art2	spear head	punic	null	$X_8 = t_{10}$	$X_2 = t_2$
$\Phi_e^{pa} = \{(ET_{ArteExp}, ET_{Arte}), (ET_{ArteMat}, ET_{Arte})\}$						

Figure 4: Die Ergebnisdatenrelation R_e^{ev} für die Beispielanfrage Q_e.

entwickelten Techniken werden in dem erweiterten probabilistischen Datenbanksystem *ProQua*[10] umgesetzt. Im Gegensatz zu anderen Systemen (z.B. [15, 7, 19]) unterstützt ProQua logik-basierte Ähnlichkeitsanfragen, sowie die Gewichtung von Teilanfragen innerhalb seiner Anfragesprache QSQL2 [8, 9].

5. ZUSAMMENFASSUNG

In dieser Arbeit wurde ein Konzept vorgestellt, welches mit der Hilfe von Ereignismustern und Basisereignismengen komplexe Tupelereignisse verwalten kann. Diese Tupelereignisse entstehen bei der Auswertung einer komplexen positiven Algebraanfrage auf einer probabilistischen BID-Datenbank. Der wesentliche Vorteil dieser Methode liegt in dem natürlichen Ablegen von Ableitungsformeln in einer strukturierten Form innerhalb einer erweiterten Datenrelation. Dadurch können die Tupelereignisse direkt mittels eines RDBMS verarbeitet werden.

Danksagung: Sebastian Lehrack wurde innerhalb der Projekte SCHM 1208/11-1 und SCHM 1208/11-2 von der Deutschen Forschungsgesellschaft unterstützt.

APPENDIX

A. BEWEIS FüR SATZ (1)

Es wird angenommen, dass $\Pr_Q(t) = \mathbf{P}(\phi_t)$, falls ϕ_t gebildet wurde wie in Def. (3) spezifiziert (siehe [4]). Somit muss gezeigt werden, dass (i) $Q_{\text{poss}(\mathbf{W})} = \pi_{\text{datAttr}}(R^{ev})$ und (ii) $\forall t \in Q_{\text{poss}(\mathbf{W})} : \phi_t \equiv \phi_t^{\text{dnf}}$.

Induktion über die Anzahl der Operatoren n von Q

- **Induktionsanfang:** $n = 1 : Q = R_i$, d.h. zu zeigen
  (i) $t \in \pi_{\text{datAttr}}(\{t \bullet (X_k = t_{id}) \mid t \in R_i\}) \overset{?}{\Leftrightarrow} t \in R_i$,
  (ii) $\forall t \in R_i : \bigvee_{\substack{\hat{t} \in \{t \bullet (X_k = t_{id}) \mid t \in R_i\}, \\ \hat{t}[\text{datAttr}] = t}} \text{gen}(ET_{R_i}, \hat{t}) \overset{?}{\equiv} (X_k = t_{id})$,

  Beweis:

  (i) $\pi_{\text{datAttr}}(\{t \bullet (X_k = t_{id}) \mid t \in R_i\}) = R_i$ ✓
  (ii) $\forall \hat{t} \in \{t \bullet (X_k = t_{id}) \mid t \in R_i\}$: nach Konstruktion gibt es eineindeutiges $t \in R_i$, sodass $\hat{t} = t \bullet (X_k = t_{id})$

  $\Rightarrow \forall t \in R_i : \bigvee_{\substack{\hat{t} \in \{t \bullet (X_k = t_{id}) \mid t \in R_i\}, \\ \hat{t}[\text{datAttr}] = t}} \text{gen}(ET_{R_i}, \hat{t}) \equiv \text{gen}(ET_{R_i}, t \bullet (X_k = t_{id})) \equiv (X_k = t_{id})$ ✓

[10] http://dbis.informatik.tu-cottbus.de/ProQua/

- **Induktionsschritt**: $n \to n + 1$ mit der Induktionsannahme (IA), dass für alle Anfragen mit bis zu n Operatoren gilt:

  (i) $t \in R_1^{ev} \Leftrightarrow t \in Q_1$ und
  (ii) $\forall t \in Q_1 : \bigvee_{\substack{\hat{t} \in R_1^{ev}, \\ \hat{t}[\text{datAttr}] = t}} \text{gen}(\Phi_1^{pa}, \hat{t}) \equiv \phi_{t_1}$.

- **Operator: $\mathbf{Q} \equiv \sigma_{\mathbf{sc}}(\mathbf{Q_1})$**, d.h. zu zeigen

  (i) $t \in \pi_{\text{datAttr}}(\sigma_{sc}(R_1^{ev})) \overset{?}{\Leftrightarrow} t \in \sigma_{sc}(Q_1)$,
  (ii) $\forall t \in \sigma_{sc}(Q_1) : \bigvee_{\substack{\hat{t} \in \sigma_{sc}(R_1^{ev}), \\ \hat{t}[\text{datAttr}] = t}} \text{gen}(\Phi_1^{pa}, \hat{t}) \overset{?}{\equiv} \phi_{t_1}$

  Beweis:

  (i) $t \in \pi_{\text{datAttr}}(\sigma_{sc}(R_1^{ev})) \Leftrightarrow \exists \hat{t} \in R_1^{ev} \wedge \hat{t}[\text{datAttr}] = t \wedge sc(t) = \text{true} \overset{IA}{\Leftrightarrow} \hat{t}[\text{datAttr}] \in Q_1 \wedge \hat{t}[\text{datAttr}] = t \wedge sc(t) = \text{true} \Leftrightarrow t \in \sigma_{sc}(Q_1)$ ✓

  (Bed. $sc(t)$ ist nur über Attribute von datAttr definiert)

  (ii) $\forall t \in \sigma_{sc}(Q_1) \Rightarrow sc(t) = \text{true} \wedge (\hat{t}[\text{datAttr}] = t \Rightarrow sc(\hat{t}) = \text{true}) \Rightarrow$
  $\forall t \in \sigma_{sc}(Q_1) : \bigvee_{\substack{\hat{t} \in \sigma_{sc}(R_1^{ev}), \\ \hat{t}[\text{datAttr}] = t}} \text{gen}(\Phi_1^{pa}, \hat{t}) = \bigvee_{\substack{\hat{t} \in R_1^{ev}, \\ \hat{t}[\text{datAttr}] = t}} \text{gen}(\Phi_1^{pa}, \hat{t}) \overset{IA}{\equiv} \phi_{t_1}$ ✓

  (Bed. $\hat{t}[\text{datAttr}] = t$ ist strenger als $sc(t)$)

- **Operator: $\mathbf{Q} \equiv \pi_{\mathcal{A}}(\mathbf{Q_1})$**, d.h. zu zeigen

  (i) $t \in \pi_{\text{datAttr}}(\pi_{\mathcal{A} \cup \{\text{all ETs}\}}(R_1^{ev})) \overset{?}{\Leftrightarrow} t \in \pi_{\mathcal{A}}(Q_1)$,
  (ii) $\forall t \in \pi_{\mathcal{A}}(Q_1) : \bigvee_{\substack{\hat{t} \in \pi_{\mathcal{A} \cup \{\text{all ETs}\}}(R_1^{ev}), \\ \hat{t}[\text{datAttr}] = t}} \text{gen}(\Phi_1^{pa}, \hat{t}) \overset{?}{\equiv} \bigvee_{\substack{\hat{t} \in Q_1, \\ \hat{t}[\mathcal{A}] = t}} \phi_{\hat{t}}$,

  Beweis:

  (i) $t \in \pi_{\text{datAttr}}(\pi_{\mathcal{A} \cup \{\text{all ETs}\}}(R_1^{ev})) \Leftrightarrow t \in \pi_{\mathcal{A}}(R_1^{ev}) \Leftrightarrow \exists \hat{t} : \hat{t}[\mathcal{A}] = t \wedge \hat{t} \in R_1^{ev} \overset{IA}{\Leftrightarrow} \exists \hat{t} : \hat{t}[\mathcal{A}] = t \wedge \hat{t} \in Q_1 \Leftrightarrow t \in \pi_{\mathcal{A}}(Q_1)$ ✓

  ($\mathcal{A}$ besteht nur aus Datenattributen, d.h. $\mathcal{A} = \text{datAttr}$)

  (ii) $\forall t \in \pi_{\mathcal{A}}(Q_1) : \bigvee_{\substack{\hat{t} \in \pi_{\mathcal{A} \cup \{\text{all ETs}\}}(R_1^{ev}), \\ \hat{t}[\text{datAttr}] = t}} \text{gen}(\Phi_1^{pa}, \hat{t}) =$

  $$\bigvee_{\substack{\hat{t} \in \pi_{\mathcal{A} \cup \{\text{all ETs}\}}(R_1^{ev}), \\ \hat{t}[\mathcal{A}] = t}} \left(\bigvee_{\substack{\dot{t} \in R_1^{ev}, \\ \dot{t}[\text{datAttr}_1] = \hat{t}}} \text{gen}(\Phi_1^{pa}, \dot{t}) \right) \overset{IA}{=}$$

$$\bigvee_{\substack{\hat{t}\in\pi_{\mathcal{A}\cup\{\text{all ETs}\}}(R_1^{ev}),\\ \hat{t}[\mathcal{A}]=t}} \phi_{\hat{t}} = \bigvee_{\substack{\hat{t}\in Q_1,\\ \hat{t}[\mathcal{A}]=t}} \phi_{\hat{t}}\ \checkmark$$

(da $\mathcal{A} \subseteq \text{datAttr}_1$ und Idempotenz gilt)

- **Operator: $\mathbf{Q \equiv Q_1 \bowtie Q_2}$**, d.h. zu zeigen

(i) $t \in \pi_{\text{datAttr}}(R_1^{ev} \bowtie R_2^{ev}) \overset{?}{\Leftrightarrow} t \in Q_1 \bowtie Q_2$,
(ii) $\forall t \in Q_1 \bowtie Q_2 : \bigvee_{\substack{\hat{t}\in R_1^{ev}\bowtie R_2^{ev},\\ \hat{t}[\text{datAttr}]=t}} \text{gen}(\Phi_1^{pa} \times \Phi_2^{pa}, \hat{t}) \overset{?}{\equiv} \phi_{t_1} \wedge \phi_{t_2}$

Beweis:

(i) $t \in \pi_{\text{datAttr}}(R_1^{ev} \bowtie_{jc} R_2^{ev}) \Leftrightarrow \exists t_1, t_2 : t = t_1[\text{datAttr}_1] \bullet t_2[\text{datAttr}_2] \wedge jc(t_1, t_2) = \text{true} \Leftrightarrow t_1 \in \pi_{\text{datAttr}_1}(R_1^{ev}) \wedge t_2 \in \pi_{\text{datAttr}_2}(R_2^{ev}) \overset{IA}{\Leftrightarrow} t_1 \in Q_1 \wedge t_2 \in Q_2 \wedge jc(t_1, t_2) = \text{true} \Leftrightarrow t_1 \bullet t_2 \in Q_1 \bowtie_{jc} Q_2 \Leftrightarrow t \in Q_1 \bowtie Q_2\ \checkmark$

(Verbundbed. $jc(t_1, t_2)$ (nat. Verbund) bezieht sich nur auf Datenattributen, da durch Konstruktion stets gilt $\text{evAttr}_1 \cap \text{evAttr}_2 = \emptyset$)

(ii) $\forall t \in Q_1 \bowtie Q_2 : \bigvee_{\substack{\hat{t}\in R_1^{ev}\bowtie R_2^{ev},\\ \hat{t}[\text{datAttr}]=t}} \text{gen}(\Phi_1^{pa} \times \Phi_2^{pa}, \hat{t}) =$

$$\bigvee_{\substack{t_1\in R_1^{ev}, t_2\in R_2^{ev},\\ (t_1\bullet t_2)[\text{datAttr}]=t}} \text{gen}(\Phi_1^{pa} \times \Phi_2^{pa}, t_1 \bullet t_2) =$$

$$\bigvee_{\substack{t_1\in R_1^{ev}, t_2\in R_2^{ev},\\ (t_1\bullet t_2)[\text{datAttr}]=t}} \text{gen}(\Phi_1^{pa}, t_1) \wedge \text{gen}(\Phi_2^{pa}, t_2) =$$

$$\bigvee_{\substack{t_1\in R_1^{ev},\\ t_1[\text{datAttr}_1]=t}} \text{gen}(\Phi_1^{pa}, t_1) \wedge \bigvee_{\substack{t_2\in R_2^{ev},\\ t_2[\text{datAttr}_2]=t}} \text{gen}(\Phi_2^{pa}, t_2) \overset{IA}{=} \phi_{t_1} \wedge \phi_{t_2}\ \checkmark$$

(da $(t_1 \bullet t_2)[\text{datAttr}] = t \Rightarrow jc(t_1 \bullet t_2) = \text{true}$; konkatenierte Muster (erzeugt durch $\times$) drücken Konjunktion aus und Φ_i^{pa} enthält nur Basisereignismuster von t_i)

- **Operator: $\mathbf{Q \equiv Q_1 \cup Q_2}$**, d.h. zu zeigen

(i) $t \in \pi_{\text{datAttr}}(R_1^{ev} \bowtie^{fo} R_2^{ev}) \overset{?}{\Leftrightarrow} t \in Q_1 \cup Q_2$,
(ii) $\forall t \in Q_1 \cup Q_2 : \bigvee_{\substack{\hat{t}\in R_1^{ev}\bowtie^{fo} R_2^{ev},\\ \hat{t}[\text{datAttr}]=t}} \text{gen}(\Phi_1^{pa} \cup \Phi_2^{pa}, \hat{t}) \overset{?}{\equiv} \phi_{t_1} \vee \phi_{t_2}$

Beweis:

(i) $t \in \pi_{\text{datAttr}}(R_1^{ev} \bowtie^{fo} R_2^{ev}) \Rightarrow t \in \pi_{\text{datAttr}_1}(R_1^{ev}) \vee t \in \pi_{\text{datAttr}_2}(R_2^{ev}) \overset{IA}{\Leftrightarrow} t \in Q_1 \vee t_2 \in Q_2 \Leftrightarrow t \in Q_1 \cup Q_2$

(wegen der Def. eines vollen äußeren Verbundes und datAttr = $\text{datAttr}_1 = \text{datAttr}_2$)

(ii) $\forall t \in Q_1 \cup Q_2 : \bigvee_{\substack{\hat{t}\in R_1^{ev}\bowtie^{fo} R_2^{ev},\\ \hat{t}[\text{datAttr}]=t}} \text{gen}(\Phi_1^{pa} \cup \Phi_2^{pa}, \hat{t}) =$

$$\bigvee_{\substack{\hat{t}\in R_1^{ev}\vee\hat{t}\in R_2^{ev},\\ \hat{t}[\text{datAttr}]=t}} \big(\text{gen}(\Phi_1^{pa}, \hat{t}) \vee \text{gen}(\Phi_2^{pa}, \hat{t})\big) =$$

$$\bigvee_{\substack{\hat{t}\in R_1^{ev},\\ \hat{t}[\text{datAttr}]=t}} \text{gen}(\Phi_1^{pa}, \hat{t}) \vee \bigvee_{\substack{\hat{t}\in R_2^{ev},\\ \hat{t}[\text{datAttr}]=t}} \text{gen}(\Phi_2^{pa}, \hat{t}) \overset{IA}{=} \phi_{t_1} \vee \phi_{t_2}\ \checkmark$$

(wegen der Def. eines vollen äußeren Verbundes und da Φ^{pa} eine disjunktiv verknüpfte Klauselkombination beschreibt)

B. REFERENCES

[1] L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In *ICDE*, pages 983–992, 2008.

[2] D. Barbará, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Trans. on Knowl. and Data Eng.*, 4:487–502, October 1992.

[3] R. Fink, D. Olteanu, and S. Rath. Providing support for full relational algebra in probabilistic databases. In *ICDE*, pages 315–326, 2011.

[4] N. Fuhr and T. Roelleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. IS*, 15(1):32–66, 1997.

[5] F. Henze, H. Lehmann, and W. Langer. CISAR - A Modular Database System as a Basis for Analysis and Documentation of Spatial Information. In *CAA*, pages 228–233, 2007.

[6] R. M. Karp, M. Luby, and N. Madras. Monte-carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, 10(3):429–448, 1989.

[7] C. Koch. MayBMS: A System for Managing Large Uncertain and Probabilistic Databases. In *Managing and Mining Uncertain Data, ch. 6.* Springer-Verlag, 2008.

[8] S. Lehrack, S. Saretz, and I. Schmitt. QSQL2: Query Language Support for Logic-Based Similarity Conditions on Probabilistic Databases. In *RCIS*, 2012 (to appear).

[9] S. Lehrack and I. Schmitt. QSQL: Incorporating Logic-Based Retrieval Conditions into SQL. In *DASFAA*, pages 429–443, 2010.

[10] S. Lehrack and I. Schmitt. A Probabilistic Interpretation for a Geometric Similarity Measure. In *ECSQARU*, pages 749–760, 2011.

[11] S. Lehrack and I. Schmitt. A Unifying Probability Measure for Logic-Based Similarity Conditions on Uncertain Relational Data. In *NTSS*, pages 14–19, 2011.

[12] D. Olteanu, J. Huang, and C. Koch. Approximate confidence computation in probabilistic databases. In *ICDE*, pages 145–156, 2010.

[13] D. Olteanu and H. Wen. Ranking Query Answers in Probabilistic Databases: Complexity and Efficient Algorithms. In *ICDE*, 2012 (to appear).

[14] C. Ré and D. Suciu. Approximate lineage for probabilistic databases. *PVLDB*, 1(1):797–808, 2008.

[15] C. Ré and D. Suciu. Managing Probabilistic Data with MystiQ: The Can-Do, the Could-Do, and the Can't-Do. In *SUM*, pages 5–18, 2008.

[16] A. D. Sarma, O. Benjelloun, A. Y. Halevy, and J. Widom. Working models for uncertain data. In *ICDE*, page 7, 2006.

[17] F. Schaefer and A. Schulze. OpenInfRA – Storing and retrieving information in a heterogenous documentation system. In *CAA*, 2012 (to appear).

[18] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[19] J. Widom. Trio: A system for data, uncertainty, and lineage. In *Managing and Mining Uncertain Data*, pages 113–148. Springer, 2008.