Universität Rostock

Traditio et Innovatio

Neueste Entwicklungen der Informatik
Abschlusspräsentation Cluster Provenance

Tanja Auge, Pia Wilsdorf, Sabrina Brossmann

Gliederung

- Einführung & Grundlagen
- Berechnung von Provenance-Polynomen
- Implementierung der Provenance-Berechnung
- Ausblick

Was ist Data Provenance?

- Data Provenance liefert Informationen über den Ursprung und Erstellungsprozess von Daten

- Anwendungsfälle:
  - Qualitätssicherung
  - Datenverifikation
  - Zugriffskontrolle
  - Debugging

Provenance Arten

- Why: Warum befindet sich ein Tupel in der Ergebnismenge?
- How: Wie ist aus den Ursprungstupeln das Ergebnistupel geworden?
- Where: Von welchen Eingabeorten stammt das Ergebnistupel?
- Why Not: Warum befindet sich ein erwartetes Tupel nicht in der Ergebnismenge?

Mögliche Auswertungsoperationen

- SPJ: Selektion, Projektion, Verbund
- SPJU: zusätzlich Vereinigung
- SPJA: zusätzlich Aggregation (Gruppierung, Summe, Durchschnitt,…)

Provenance Antworten

- Extensionale Antwort
  - Tupel aus den Originaldaten als Antwort
- Intensionale Antwort
  - Beschreibung der Daten als Antwort
- Modifikationsbasierte Antwort:
  - Speziell bei why not sinnvoll
  - Vorschlag zur minimalen Änderung der Auswertung als Antwort, durch die das im Ergebnis vermisste Element sich für das Ergebnis qualifiziert

Eager vs Lazy Provenance

- Eager Provenance
  - Provenance Informationen werden bei Transformation der Originaldaten direkt mitberechnet und im Ergebnis mitgespeichert
  - Pro: Schnellerer Zugriff auf Provenance Informationen
  - Contra: Längere Anfragebearbeitung, erhöhter Speicherbedarf
- Lazy Provenance
  - Provenance Informationen werden erst bei Bedarf aus Originaldaten, Ergebnisdaten und erfolgten Transformationen berechnet
  - Pro: Keine längere Anfragebearbeitung, kein erhöhter Speicherbedarf
  - Contra: Komplexere Berechnung der Provenance Informationen

Provenance-Halbringe

- Ein **Provenance-Halbring** ist ein kommutativer Halbring $(K, +, \cdot, 0, 1)$ mit
  - K = Menge der Tupelidentifikatoren
    (jedes Tupel der Ausgangsdatentabellen hat einen eindeutigen Tupelidentifikator)
  - $(K, +)$ kommutative Halbgruppe mit neutralem Element 0
    (entspricht Vereinigung und Projektion (ohne Duplikate))
  - $(K, \cdot)$ kommutative Halbgruppe mit neutralem Element 1
    (entspricht natürlichem Verbund)
  - geltende Distributivgesetzte: $x, y, z \in K$
    - $(x + y) \cdot z = x \cdot z + y \cdot z$
    - $z \cdot (x + y) = z \cdot x + z \cdot y$

Provenance Polynome

- Jedem Ergebnistupel kann ein eindeutiges Provenance-Polynom bestehend aus den Tupelidentifikatoren der Quelltupel zugeordnet werden.

- Anfragen mit Aggregation (MIN, MAX, SUM, COUNT, AVG):
  Verwendung von K-Semimodulen $(W = K \times M, \oplus, \odot, \underline{0})$ mit
  - $\oplus$ für die Summe von Polynomen;
    kann auch durch Minimumoperation min und Maximumoperation max ersetzt werden
  - $\odot$ für das Produkt von Polynom und gesuchtem Tabellenwert

Regeln beim Aufstellen von Provenance-Polynomen

- Projektion: $(\pi_V\ R)(t) = \sum_{\substack{t=t' \text{ auf } V \\ \text{und} R(t')\neq 0}} R(t')$
- Selektion: $(\sigma_P\ R)(t) = R(t) \cdot P(t)$, wobei P jedes Tupel auf 0 oder 1 abbildet
- Vereinigung: $(R_1 \cup R_2)(t) = R_1(t) + R_2(t)$
- Natürlicher Verbund: $(R_1 \bowtie R_2)(t) = R_1(t_1) \cdot R_2(t_2)$
- Umbenennung: $(\rho_\beta\ R)(t) = R(t \circ \beta)$ mit Bijektion β
- Leere Relation: $\emptyset(t) = 0$

Beispieldaten

	MatrikelNr.	Name	Vorname	Studiengang
S1	1	Damm	Fabian	Lehramt Informatik
S2	2	Müller	Sina	Informatik
S3	3	Müller	Max	Elektrotechnik
S4	4	Wolf	Jana	Informatik
S5	5	Zietz	Johannes	Informatik
S6	6	Gerber	Mia	Informatik
S7	7	Hansen	Max	Elektrotechnik

Studenten

	ModulNr.	MatrikelNr.	Note
N1	001	1	2.0
N2	001	2	1.7
N3	001	4	1.7
N4	001	5	3.0
N5	002	2	2.7
N6	002	3	2.3
N7	003	1	1.0
N8	004	3	1.3
N9	004	2	3.0
N10	005	4	2.7
N11	005	7	1.7
N12	006	4	2.7
N13	006	5	4.0
N14	007	1	2.3
N15	007	3	1.7
N16	008	1	3.3
N17	008	5	2.7

Noten

Beispieldaten

	ModulNr.	Titel	Dozent
M1	001	Datenbanken 1	Professor A
M2	002	Datenbanken 2	Professor A
M3	003	Funktionale Programmierung	Professor B
M4	004	Imperative Programmierung	Professor C
M5	005	Künstliche Intelligenz	Professor B
M6	006	Mathe 1	Professor D
M7	007	Neuste Entwicklungen in der Informatik	Professor A
M8	008	Smart Computing	Professor B
M9	009	Theoretische Informatik 1	Professor D

Module

	ModulNr.	MatrikelNr.
T1	001	1
T2	001	2
T3	001	4
T4	001	5
T5	002	2
T6	002	3
T7	003	1
T8	004	3
T9	004	2
T10	004	7
T11	005	4
T12	005	7
T13	006	4
T14	006	5
T15	007	1
T16	007	3
T17	008	1
T18	008	3
T19	008	4
T20	008	5

Teilnehmer

Provenance-Polynome anhand beispielhafter Anfragen

- Die Prüfungsnoten für den Studenten mit einer bestimmten Matrikelnummer:

```
SELECT Matrikelnummer, Note
FROM Noten
WHERE Matrikelnummer = '4'
```

Provenance	Modulnummer	Note
N3	001	1.7
N10	005	2.7
N12	006	2.7

- Die Durchschnittsnote für einen Studenten mit einer bestimmten Matrikelnummer:

$$P = \frac{N3 \odot 1.7 \oplus N10 \odot 2.7 \oplus N12 \odot 2.7}{N3 \oplus N10 \oplus 12}$$

How	Why	Where
P	N3, N10, N12	Noten

Provenance-Polynome anhand beispielhafter Anfragen

- Die Prüfungsnoten für einen Studenten mit einem bestimmten Vornamen:

```
SELECT Modulnummer, Note
FROM Studeten JOIN Noten ON
(Studenten.Matrikelnummer = Noten.Matrikelnummer)
WHERE Studenten.Vorname = 'Sina'
```

Provenance	Modulnummer	Note
S2 * N2	001	1.7
S2 * N5	002	2.7
S2 * N9	004	3.0

- Die Durchschnittsnote für einen Studenten mit einem bestimmten Vornamen:

$$P = \frac{[(S2 \cdot N2) \odot 1.7 \oplus (S2 \cdot N5) \odot 2.7 \oplus (S2 \cdot N9) \odot 3.0]}{S2 \cdot N2 \oplus S2 \cdot N5 \oplus S2 \cdot N9}$$

How	Why	Where
P	(S2, N2), (S2, N5), (S2, N9)	Studenten, Noten

Provenance-Polynome anhand beispielhafter Anfragen

- Die Prüfungsnoten für einen Studenten mit einem bestimmten Vornamen:

```
SELECT Modulnummer, Note
FROM Studenten JOIN Noten ON
(Studenten.Matrikelnummer = Noten.Matrikelnummer)
WHERE Studenten.Vorname = 'Max'
```

Provenance	Matrikelnummer	Modulnummer	Note
S3 * N6	3	002	2.3
S3 * N8	3	004	1.3
S3 * N15	3	007	1.7
S7 * N11	7	005	1.7

- Die Durchschnittsnote für einen Studenten mit einem bestimmten Vornamen:

$$P = \frac{[(S3 \cdot N6) \odot 2.3 \oplus (S3 \cdot N8) \odot 1.3 \oplus (S3 \cdot N15) \odot 1.7 \oplus (S7 \cdot N11) \odot 1.7]}{S3 \cdot N6 \oplus S3 \cdot N8 \oplus S3 \cdot N15 \oplus S7 \cdot N11}$$

How	Why	Where
P	(S3, N6), (S3, N8), (S3, N15), (S7, N11)	Studenten, Noten

Implementierung - Funktionsumfang

- SPJ – Anfragen
  - Einfache Anfragen
  - Natural Join
- Aggregation
  - MIN, MAX, AVG, SUM, COUNT
- Ausgabe per CSV oder Console
- Eager Evaluation

Implementierung - Schnittstelle

- dbms
- url
- port
- user
- password
- dbname
- schemaname
- query
- multipleTables
- outFormat
- outputPath

Implementierung - Komponenten

- Parser
  - Aufbauen eines Query-Trees für die aktuell unterstützten SQL-Konstrukte
- Provenance-Komponente
  - Modifizieren des Query-Trees
  - Bauen der Provenance-Query
  - Empfangen von Ergebnisrelation, TupleIDs, Schemainformationen
  - Ergebnispräsentation

Implementierung – Extrahieren der Provenance-Informationen

- Query modifizieren, um alle relevanten Tupel + Tupel Identifier zu erhalten
- Bsp:

```
SELECT modulnummer, note
FROM studenten NATURAL JOIN noten
WHERE vorname = 'Sina';
```

→

```
SELECT modulnummer, note, sid, nid
FROM studenten NATURAL JOIN noten
WHERE vorname = 'Sina';
```

```
SELECT AVG(note) from noten;
```

→

```
SELECT note, nid FROM noten ;
```

- REQUIREMENT: global eindeutige Tupel Identifier

Implementierung – Provenance-Arten

- How-Provenance
  - Ohne Aggregat: Tuple Identifier zeilenweise zu Polynom verknüpfen
  - Mit Aggregat: Alle erhaltenen Tuple Identifier formen das Polynom
- Why-Provenance
  - Daraus Witness-Tupel extrahieren
- Where-Provenance
  - Korrespondierende Tabellennamen (Nutzung des DB-Schemas)

Implementierung – Demo

- SELECT modulnummer, note FROM noten WHERE matrikelnummer = 4;

modulnummer	note	How-Provenance	Why-Provenance	Where-Provenance
1	1.7	N3	(N3)	noten
5	2.7	N10	(N10)	noten
6	2.7	N12	(N12)	noten

Implementierung – Demo

- SELECT modulnummer, note FROM studenten NATURAL JOIN noten WHERE vorname = 'Sina';

modulnummer	note	How-Provenance	Why-Provenance	Where-Provenance
1	1.7	S2*N2	(S2,N2)	studenten,noten
2	2.7	S2*N5	(S2,N5)	studenten,noten
4	3.0	S2*N9	(S2,N9)	studenten,noten

Implementierung – Demo

- SELECT modulnummer, note FROM studenten NATURAL JOIN noten WHERE vorname = ‘Max';

modulnummer	note	How-Provenance	Why-Provenance	Where-Provenance
2	2.3	S3*N6	(S3,N6)	studenten,noten
4	1.3	S3*N8	(S3,N8)	studenten,noten
5	1.7	S7*N11	(S7,N11)	studenten,noten
7	1.7	S3*N15	(S3,N15)	studenten,noten

Implementierung – Demo

- SELECT AVG(note) FROM studenten NATURAL JOIN noten WHERE vorname = ‘Sina';

avg(note)	How-Provenance
2.466666666666667	(S2*N2x1.7 +w S2*N5x2.7 +w S2*N9x3.0) / (S2*N2 +w S2*N5 +w S2*N9)

avg(note)	Why-Provenance
2.466666666666667	((S2,N2),(S2,N5),(S2,N9))

avg(note)	Where-Provenance
2.466666666666667	studenten,noten

Implementierung – Demo

- SELECT AVG(note) FROM studenten NATURAL JOIN noten WHERE vorname = ‘Max';

avg(note)	How-Provenance
1.75	(S3*N6x2.3 +w S3*N8x1.3 +w S7*N11x1.7 +w S3*N15x1.7) / (S3*N6 +w S3*N8 +w S7*N11 +w S3*N15)

avg(note)	Why-Provenance
1.75	((S3,N6),(S3,N8),(S7,N11),(S3,N15))

avg(note)	Where-Provenance
1.75	studenten,noten

Implementierung – Demo

- SELECT COUNT(matrikelnummer) FROM module NATURAL JOIN teilnehmer WHERE dozent = 'Professor B‘;

count(modulnummer) 7.0	How-Provenance M3*T7 +w M5*T11 +w M5*T12 +w M8*T17 +w M8*T18 +w M8*T19 +w M8*T20
count(modulnummer) 7.0	Why-Provenance ((M3,T7),(M5,T11),(M5,T12),(M8,T17),(M8,T18),(M8,T19),(M8,T20))
count(modulnummer) 7.0	Where-Provenance module,teilnehmer

Implementierung – Nächste Schritte

- UNION
- Geschachtelte Selects
- Weiter Ausgabeformate
- Intensionale Antworten