

Anfrageoptimierung unter Constraints

- 1 Kodierung von Datenbankconstraints
- 2 Äquivalenz Kojunktiver Anfragen unter Constraints
- 3 Chase-Algorithmus und Eigenschaften

Literatur:

- R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa: *Semantics and Query Answering*. In Proc. ICDT 2003.
- A. Deutsch, L. Popa, V. Tannen: *Query Reformulation with Constraints*. In SIGMOD Record, Vol. 35, No. 1, March 2006.
- A. Deutsch, A. Nash : *Chase (DB Encyclopedia Entry)*. Springer, 2008. <http://db.ucsd.edu/pubsFileFolder/304.pdf>

Motivation: Anfrageoptimierung unter Constraints

Beispiel

Sei

$$\begin{aligned}
 &Sales(\underline{PName}, \underline{SName}, \underline{CName}), \\
 &Part(\underline{PName}, \underline{Type}), \\
 &Cust(\underline{CName}, \underline{CAddr}), \\
 &Supp(\underline{SName}, \underline{SAddr}),
 \end{aligned}$$

wobei die Schlüsselattribute unterstrichen sind.

Sind die folgenden beiden Konjunktiven Anfragen Q, Q' äquivalent unter Einbeziehung der Schlüsselbedingungen?

$$Q : \quad ans(A, A) \leftarrow Sales(p1, s1, C), Cust(C, A)$$

$$Q' : \quad ans(A, A') \leftarrow Sales(p1, s1, C), Cust(C, A), Cust(C, A')$$

Falls ja, dann ist Q der Version Q' offensichtlich vorzuziehen.

Darstellung von Constraints mittels First-order Logic

Beispiel

Betrachten Sie die Relation

$$\text{Cust}(\underline{\text{CName}}, \text{CAddr}).$$

Der Schlüssel-Constraint für Attribut *CName* kann als First-order Logic Satz

$$\forall n, a_1, a_2 (\text{Cust}(n, a_1) \wedge \text{Cust}(n, a_2) \rightarrow a_1 = a_2)$$

dargestellt werden.

Klassen von Constraints:

Wir betrachten zwei Klassen von Constraints:

- Tuple-generating Dependencies
z.B. zur Kodierung von Fremdschlüssel-Beziehungen
- Equality-generating Dependencies
z.B. zur Kodierung von Schlüssel-Beziehungen oder funktionalen Abhängigkeiten



Tuple-generating Dependencies

Definition

Sei $\mathcal{R} = \{R_1, \dots, R_n\}$ ein Datenbankschema und \bar{x}, \bar{y} Vektoren von Variablen.

Eine *Tuple-generating Dependency* ist ein First-order Logic Satz der Form

$$\varphi := \forall \bar{x} (\phi(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y})),$$

so dass die folgenden Bedingungen erfüllt sind:

- $\phi(\bar{x})$ ist eine Konjunktion von atomaren Formeln der Form $R_i(\bar{x})$ (mit $R_i \in \mathcal{R}$) über Variablen aus \bar{x} ,
- $\psi(\bar{x}, \bar{y})$ ist eine nicht leere Konjunktion von atomaren Formeln der Form $R_i(\bar{x}, \bar{y})$ (mit $R_i \in \mathcal{R}$) über Variablen aus \bar{x} und \bar{y} , und
- alle Variablen aus \bar{x} , die in $\psi(\bar{x}, \bar{y})$ auftreten, kommen auch in $\phi(\bar{x})$ vor.

Wir schreiben $body(\varphi)$ für die Menge von Atomen in $\phi(\bar{x})$ und $head(\varphi)$ für die Menge von Atomen in $\psi(\bar{x}, \bar{y})$.



Tuple-generating Dependencies

Beispiel

Der First-order Logic Satz

$$\varphi_1 := \forall p, s, c (Sales(p, s, c) \rightarrow \exists t Part(p, t))$$

drückt die Fremdschlüsselbeziehung des ersten Attributes von *Sales* bezüglich des ersten Attributs von *Part* aus: jedes Element das in der ersten Position von Relation *Sales* auftaucht muss als Schlüssel in der Relation *Part* auftauchen.

φ_1 ist eine Tuple-generating Dependency. Es gilt:

- $\bar{x} := (p, s, c)$ und $\bar{y} := (t)$,
- $\phi(\bar{x}) := Sales(p, s, c)$ ist eine Konjunktion von atomaren Formeln über \bar{x} ,
- $\psi(\bar{x}, \bar{y}) := Part(p, t)$ ist eine nicht leere Konjunktion von atomaren Formeln über Variablen aus \bar{x} und \bar{y} , und
- die Variablen aus \bar{x} die in $\psi(\bar{x}, \bar{y})$ auftreten (p), treten auch in $\phi(\bar{x})$ auf.

Weiterhin gilt $body(\varphi_1) = \{Sales(p, s, c)\}$ und $head(\varphi_1) = \{Part(p, t)\}$.



Equality-generating Dependencies

Definition

Sei $\mathcal{R} = \{R_1, \dots, R_n\}$ ein Datenbankschema und \bar{x} ein Vektor von Variablen. Eine *Equality-generating Dependency* ist ein First-order Logic Satz der Form

$$\varphi := \forall \bar{x} (\phi(\bar{x}) \rightarrow x_i = x_j),$$

so dass die folgenden Bedingungen erfüllt sind:

- $\phi(\bar{x})$ ist eine nicht leere Konjunktion von atomaren Formeln der Form $R_i(\bar{x})$ (mit $R_i \in \mathcal{R}$) über Variablen aus \bar{x} und
- x_i, x_j kommen in $\phi(\bar{x})$ vor.

Wir schreiben $body(\varphi)$ für die Menge von Atomen in $\phi(\bar{x})$ und $head(\varphi)$ für die Menge $\{x_i = x_j\}$.



Equality-generating Dependencies

Beispiel

Der First-order Logic Satz

$$\varphi_2 := \forall n, a_1, a_2 (Cust(n, a_1) \wedge Cust(n, a_2) \rightarrow a_1 = a_2)$$

stellt sicher dass das erste Attribut der Relation *Cust* ein Schlüsselattribut ist. φ_2 ist eine Equality-generating Dependency. Es gilt:

- $\bar{x} := (n, a_1, a_2)$,
- $\phi := Cust(n, a_1) \wedge Cust(n, a_2)$ ist eine nicht leere Konjunktion von atomaren Formeln über Variablen aus \bar{x} , und
- a_1 und a_2 kommen in $\phi(\bar{x})$ vor.

Weiterhin gilt $body(\varphi_2) = \{Cust(n, a_1), Cust(n, a_2)\}$ und $head(\varphi_2) = \{a_1 = a_2\}$.



Constraint-erfüllende Datenbankinstanzen

Definition

Sei Σ eine Menge von Tuple-generating und Equality-generating Dependencies und \mathcal{I} eine Datenbankinstanz. \mathcal{I} erfüllt Σ , $\mathcal{I} \models \Sigma$, genau dann wenn \mathcal{I} jeden einzelnen Constraint in Σ erfüllt, d.h. es gilt dass $\mathcal{I} \models \varphi$ für alle $\varphi \in \Sigma$.

Beispiel

Betrachten Sie die Constraintmenge $\Sigma = \{\varphi_1, \varphi_2\}$ mit

$$\begin{aligned} \varphi_1 &:= \forall p, s, c (Sales(p, s, c) \rightarrow \exists t Part(p, t)), \\ \varphi_2 &:= \forall n, a_1, a_2 (Cust(n, a_1) \wedge Cust(n, a_2) \rightarrow a_1 = a_2), \end{aligned}$$

und die Datenbankinstanz \mathcal{I}_1 :

<i>Sales</i>	<i>PName</i>	<i>SName</i>	<i>CName</i>	<i>Part</i>	<i>PName</i>	<i>Type</i>	<i>Cust</i>	<i>CName</i>	<i>CAddr</i>
	p_1	s_1	c_1		p_1	t_1		c_1	a_1
	p_2	s_2	c_2		p_3	t_2		c_2	a_2

Offensichtlich gilt $\mathcal{I}_1 \models \varphi_2$, $\mathcal{I}_1 \not\models \varphi_1$ und folglich $\mathcal{I}_1 \not\models \Sigma$.



Problemstellungen

Offene Fragestellungen

Gegeben eine Menge von Tuple-generating und Equality-generating Dependencies Σ und zwei Konjunktive Anfragen Q, Q' :

- Ist das Ergebnis von Q enthalten im Ergebnis von Q' auf allen Datenbankinstanzen $\mathcal{I} \models \Sigma$?
- Ist Q äquivalent zu Q' auf allen Datenbankinstanzen $\mathcal{I} \models \Sigma$?
- Finde minimale Anfragen die äquivalent zu Q auf allen Datenbankinstanzen $\mathcal{I} \models \Sigma$ sind.

Definition

Sei Σ eine Menge von Tuple-generating und Equality-generating Dependencies und Q, Q' Konjunktive Anfragen. Q ist in Q' enthalten unter Σ ($Q \sqsubseteq_{\Sigma} Q'$) genau dann wenn $Q(\mathcal{I}) \subseteq Q'(\mathcal{I})$ für alle Datenbankinstanzen $\mathcal{I} \models \Sigma$.

Q ist Σ -äquivalent zu Q' ($Q \equiv_{\Sigma} Q'$) genau dann wenn $Q \sqsubseteq_{\Sigma} Q'$ und $Q' \sqsubseteq_{\Sigma} Q$.



Motivation

Wir können das Enthaltensein-Problem unter Constraints mit Hilfe des sogenannten Chase-Algorithmus auf das Enthaltensein-Problem ohne Constraints zurückführen.



Konvention

Gegeben eine konjunktive Anfrage

$$Q: \text{ans}(\bar{x}) \leftarrow \text{body}(\bar{x}, \bar{y})$$

dann bezeichnen die rechte Seite als $\text{body}(Q)$.

Beispiel-Szenario

Schema: $\text{hasAirport}(c_id)$, $\text{fly}(c_id1, c_id2, dist)$, $\text{rail}(c_id1, c_id2, dist)$

Constraints:

Wenn es zwischen zwei Städten einen Flug gibt, dann haben beide Städte einen Flughafen:

$$\alpha_1 : \forall x_1, x_2, y \ (\text{fly}(x_1, x_2, y) \rightarrow \text{hasAirport}(x_1) \wedge \text{hasAirport}(x_2))$$

Bahn-Verbindungen sind symmetrisch:

$$\alpha_2 : \forall x_1, x_2, y \ (\text{rail}(x_1, x_2, y) \rightarrow \text{rail}(x_2, x_1, y))$$

Jede Stadt die per Flugzeug erreichbar ist, kann auf dem Luftweg auch wieder verlassen werden:

$$\alpha_3 : \forall x_1, x_2, y_1 \ (\text{fly}(x_1, x_2, y_1) \rightarrow \exists x_3, y_2 \ \text{fly}(x_2, x_3, y_2))$$

Flug-Verbindungen sind symmetrisch:

$$\alpha_4 : \forall x_1, x_2, y \ (\text{fly}(x_1, x_2, y) \rightarrow \text{fly}(x_2, x_1, y))$$

Chase am Beispiel I

$Q_2: \text{ans}(x_2) \leftarrow \text{rail}(c_1, x_1, y_1), \text{fly}(x_1, x_2, y_2), \text{fly}(x_2, x_1, y_2), \text{rail}(x_1, c_1, y_1)$

Repariere eine Verletzung von α_1 :

$Q'_2: \text{ans}(x_2) \leftarrow \text{rail}(c_1, x_1, y_1), \text{fly}(x_1, x_2, y_2), \text{fly}(x_2, x_1, y_2), \text{rail}(x_1, c_1, y_1),$
 $\text{hasAirport}(x_1), \text{hasAirport}(x_2)$

Wir sehen, dass $\text{body}(Q'_2)$ alle Constraints erfüllt.

Der Chase-Algorithmus terminiert in diesem Fall.

Wir bezeichnen Q'_2 als Q_2^Σ .

Bezeichne Q^Σ ein Ergebnis des Chase (falls existent).

Satz

Sei Q eine konjunktive Anfrage und existiere Q^Σ .

Dann gilt: $Q \equiv_\Sigma Q^\Sigma$.

Zum Beweis beachte man, dass die Äquivalenz unter Σ in jedem Chase-Schritt erhalten wird.

Chase am Beispiel II

Im Allgemeinen ist die Situation nicht so einfach:

$Q: \text{ans}(x_2) \leftarrow \text{hasAirport}(x_1), \text{hasAirport}(x_2), \text{fly}(x_1, x_2, y_2)$

- Chase von Q mit α_4 :

$Q_2: \text{ans}(x_2) \leftarrow \text{hasAirport}(x_1), \text{hasAirport}(x_2), \text{fly}(x_1, x_2, y_2), \text{fly}(x_2, x_1, y_2)$

- Chase von Q mit mehrmals α_3 :

$Q_3: \text{ans}(x_2) \leftarrow \text{hasAirport}(x_1), \text{hasAirport}(x_2), \text{fly}(x_1, x_2, y_2), \text{fly}(x_2, x_3, y_3), \text{fly}(x_3, x_4, y_4), \text{fly}(x_4, x_5, y_5), \dots$

In diesem Fall terminiert der Algorithmus nicht !

Q^Σ ist nicht eindeutig. Z.B. kann man ein paar mal α_3 anwenden und dann α_4 .

Bezeichne Q^Σ ein nicht-eindeutiges Ergebnis des Chase (falls existent).

Satz

Seien Q und Q' konjunktive Anfragen und existiere Q^Σ .

Dann gilt: $Q \sqsubseteq_\Sigma Q'$ genau dann wenn $Q^\Sigma \sqsubseteq Q'$.

Dies liefert einen Algorithmus welcher das Enthaltensein konjunktiver Anfragen unter Constraints entscheidet.