Informationssyteme[1]

Kapitel 3: Relationen-Kalkül

Wintersemester 2020/21

Prof. Dr. Peer Kröger
Institut für Informatik
Arbeitsgruppe Informationssysteme und Data Mining

[1] Die folgenden Folien basieren in großen Teilen auf Material zur Vorlesung "Datenbanksysteme", die ich in meiner Zeit an der LMU München mehrmals gehalten habe. Das Material ist von den damaligen Kollegen maßgeblich (mit-)gestaltet worden, insbes. von Prof. Dr. C. Böhm.

Übersicht

1. Einleitung
2. Tupelkalkül
3. Bereichskalkül
4. Query by Example
5. Relationenkalkül und SQL

Agenda

1. Einleitung
2. Tupelkalkül
3. Bereichskalkül
4. Query by Example
5. Relationenkalkül und SQL

Intuition

- Mathematik: Prädikatenkalkül
  - Ausdrücke wie z.B. $\{x \mid x \in \mathbb{N} \wedge x^2 > 0 \wedge x^2 < 100\}$
- Wie können wir solche Ausdrücke für DB-Anfragen nutzen?
  - Bezugnahme auf DB-Relationen im Bedingungsteil (Formelteil), z.B.
    $(x_1, y_1, z_1) \in$ Mitarbeiter, $(s_1) \in$ Abteilung, ...
  - Terme werden gebildet aus Variablen, Konstanten usw.
  - Atomare Formeln werden aus der Verknüpfung von Termen mit log. Prädikaten (z.B. $=, <, >, ...$) gebildet
  - Atomare Formeln können mit Boolschen Operatoren ($\wedge, \vee, \neg$) zu komplexen Formeln zusammen gefasst werden
- Beispiel: Finde alle Großstädte (Einwohner $\geq$ 500T) in Sachsen:
  $\{t \mid \mathit{Schema}(t) = \text{Staedte} \wedge t[\text{Land}] = \mathit{Sachsen} \wedge t[\text{SEinw}] \geq 500.000\}$

Unterschied zur Relationalen Algebra

Relationale Algebra ist prozedurale Sprache

- Ausdruck gibt an, unter Benutzung welcher Operationen das Ergebnis berechnet werden soll
- "WIE soll das Ergebnis der Anfrage berechnet werden"

Relationen-Kalkül ist deklarative Sprache

- Ausdruck beschreibt, welche Eigenschaften die Tupel der Ergebnisrelation haben müssen ohne eine Berechnungsprozedur dafür anzugeben
- "WAS soll als Ergebnis der Anfrage berechnet werden"

Grundidee

- Man arbeitet mit
  - Variablen: t
  - Formeln: $\psi(t)$
  - Ausdrücken: $\{t \mid \psi(t)\}$
- Idee: Ein Ausdruck beschreibt die Menge aller Tupel, die die Formel ψ "erfüllen" ("wahr machen")
- Ein Kalkül besteht immer aus
  - Syntax: Wie sind Formeln aufgebaut?
  - Semantik: Was bedeuten die Formeln? (ist eine Formel erfüllt?)
- Es gibt zwei verschiedene Ansätze
  - Tupelkalkül: Variablen sind vom Typ Tupel
  - Bereichskalkül: Variablen haben einfachen Typ

Agenda

1. Einleitung
2. Tupelkalkül
3. Bereichskalkül
4. Query by Example
5. Relationenkalkül und SQL

Syntax: Intuition

- Wir definieren zunächst die Syntax von Formeln und Ausdrücken des Tupelkalkül
- Dies können wir tun, ohne eine konkrete Semantik dafür anzugeben (natürlich machen wir das später noch)
- Betrachten wir zunächst noch einmal den Ausdruck von oben:
  $\{t \mid \mathit{Schema}(t) = \text{Staedte} \wedge t[\text{Land}] = \mathit{Sachsen} \wedge t[\text{SEinw}] \geq 500.000\}$
  dieser hat die Form $\{t \mid \psi(t)\}$, wobei die Formel $\psi(t)$ zusammengesetzt ist:
  $\psi(t) := \mathit{Schema}(t) = \text{Staedte} \wedge t[\text{Land}] = \mathit{Sachsen} \wedge t[\text{SEinw}] \geq 500.000\}$

Syntax: Intuition

- Diese Formel besteht offenbar aus:
  - Variablen (hier t, grundsätzlich können mehrere Variablen vorkommen)
  - Variablen haben eine Struktur / ein Schema (hier $\mathit{Schema}(t) = \text{Staedte}$)
  - Variablen haben Zugriff auf Attribute des Schemas (hier z.B. $t[\text{Land}]$)
  - (atomare) Aussagen über die Variablen, sog. "Atome" (hier z.B. $t[\text{Land}] = \mathit{Sachsen}$)
  - Diese Atome sind mit logischen Operatoren (hier $\wedge$) verknüpft
- Grundsätzlich sind Formeln aus Atomen (über Varibalen), die mit logischen Operatoren zu komplexeren "Gebilden" zusammen gesetzt werden, aufgebaut.
- Diese zusammengesetzte Struktur ist wie gesagt die Syntax und die schauen wir uns jetzt genauer an ...

Syntax: Tupelvariablen

- Variablen sind "Platzhalter" für irgend etwas ...
- Beim Tupelkalkül sind die Variablen vom Typ Tupel, d.h. sie sind "Platzhalter" für Tupel
- Sie heißen daher auch Tupelvariablen
- Tupel haben eine Struktur, definiert durch ein Schema
- Daher haben Tupelvariablen ebenfalls ein definiertes Schema:
  - $Schema(t) = (A_1 : D_1, A_2 : D_2, ...)$
  - $Schema(t) = R$ (t hat dasselbe Schema wie Relation R) oder auch $Schema(t) = Schema(R)$
- Für Zugriff auf die Komponenten
  - $t[A]$ für einen Attributnamen $A \in Schema(t)$
  - Alternative Schreibweise: $t.A$ für $t[A]$; in manchen Büchern auch $t[1]$ (Zugriff auf das erste Attribut — beim geordneten Relationen-Schema)
- Tupelvariable kann in einer Formel ψ frei oder gebunden auftreten (siehe später)

Syntax: Atome

- Atome (atomare Aussagen) definieren Eigenschaften von Variablen
- Es gibt drei Arten von Atomen:
  - $R(t)$
    dabei ist R ein Relationenname und t eine Tupelvariable
  - $t[A]\ \theta\ s[B]$
    dabei sind
    - t und s Tupelvariablen
    - $A \in Schema(t)$ und $B \in Schema(s)$ (und $dom(A) = dom(B)$)
    - θ ein Vergleichsoperator mit $\theta \in \{=, <, \leq, >, \geq, \neq\}$
  - $t[A]\ \theta\ c$
    dabei sind
    - t Tupelvariable
    - $A \in Schema(t)$
    - c Kontante mit $c \in dom(A)$
    - θ ein Vergleichsoperator mit $\theta \in \{=, <, \leq, >, \geq, \neq\}$

Syntax: Formeln

Der Aufbau (Syntax) von Formeln ψ ist induktiv definiert:

- Atome:
  Jedes Atom ist eine Formel, alle vorkommenden Variablen sind frei
- Logische Operatoren:
  Sind ψ_1 und ψ_2 Formeln, dann sind ebenfalls Formeln:
  - $\neg\psi_1$
  - $(\psi_1 \wedge \psi_2)$
  - $(\psi_1 \vee \psi_2)$

  Alle Variablen in ψ_1 bzw. ψ_2 behalten ihren Status
- Quantoren:
  Ist ψ eine Variable, in der t als freie Variable auftritt, dann sind ebenfalls Formeln:
  - $(\exists t)(\psi)$
  - $(\forall t)(\psi)$

  Die Variable t ist dann (in ψ) gebunden

Syntax: Abkürzungen

- Um die freien Variablen einer Formel anzuzeigen, verwenden wir folgende Schreibweise:
  Sind $t_1, ...t_k$ die freien Variablen in einer Formel ψ so schreiben wir auch $\psi(t_1, ..., t_k)$
- Gebräuchliche Abkürzungen:
  - $\psi_1 \Rightarrow \psi_2$ für $(\neg\psi_1) \vee \psi_2$
  - $\exists t_1, ...t_k : \psi(t_1, ..., t_k)$ für $(\exists t_1)(...((\exists t_k)(\psi(t_1, ..., t_k)))...)$
  - analog $\forall t_1, ...t_k : \psi(t_1, ..., t_k)$
  - $(\exists t \in R)(\psi(t))$ für $(\exists t)(R(t) \wedge \psi(t))$
  - $(\forall t \in R)(\psi(t))$ für $(\forall t)(R(t) \Rightarrow \psi(t))$
- Bei Eindeutigkeit können Klammern weggelassen werden (es gelten die üblichen Präzedenzen)

Syntax: gebundene und freie Variablen

- Wie besprochen sind alle Variablen einer Formel, die nicht durch Quantoren gebunden sind frei
- Beispiel:

  $(\forall s)(s.A \leq u.B \vee (\exists u)(R(u) \wedge u.C > t.D))$

  - t ist frei
  - s ist gebunden:

    $$(\forall s)(\underbrace{s.A \leq u.B \vee (\exists u)(R(u) \wedge u.C > t.D)}_{\text{hier ist } s \text{ gebunden}})$$

  - u ist beim ersten Auftreten frei und dann gebunden:

    $$(\forall s)(\underbrace{s.A \leq u.B}_{\text{hier ist } u \text{ frei}} \vee (\exists u)\underbrace{(R(u) \wedge u.C > t.D)}_{\text{hier ist } u \text{ gebunden}})$$

Syntax: Ausdruck

- Ein Ausdruck des Tupelkalküls hat die Form

$$\{t \mid \psi(t)\}$$

- Dabei ist (wie durch die Schreibweise $\psi(t)$ ja angedeutet) t die einzige freie Variable in ψ
- Eine Anfrage an die Datenbank wird im Relationenkalkül durch einen Ausdruck spezifiziert (d.h. Ausdruck = Anfrage)

Semantik

- Die Semantik gibt einem (syntaktisch korrekt gebildeten) Ausdruck eine Bedeutung
- Man sagt auch: der Ausdruck wird interpretiert (daher auch "Interpretation")
- Intuitiv: was benötigen wir zur Interpretation eines Ausdrucks?
  - Tupelvariablen sind Platzhalter für Tupel, d.h. wir benötigen eine Belegung mit konkreten Tupeln
  - Atome: Aussagen (Prädikate) über Tupel, insbesondere basierend auf Vergleichsoperatoren, d.h. wir benötigen die Interpretation dieser Prädikate ("true" oder "false")
  - Formeln: Verknüpfung von Atomen mittels Bool'scher Operatoren und Quantoren, d.h. wir benötigen Interpretation dieser Operatoren/Quantoren ("true" oder "false")
  - Ausdrücke: "die Menge aller Tupel, die ψ erfüllen", d.h. wahr machen (für die ψ als "true" interpretiert wird)

Semantik

- Die Interpretation von Ausdrücken ist tatsächlich eine Abbildung der Menge der Ausdrücke auf eine Menge von Tupeln (Relationen), die auf einer Abbildung für Formeln (auf true/false) beruht, die wiederum auf einer Abbildung von Atomen (auf true/false) beruht, die wiederum auf einer Belegung der Tupelvariablen beruht:

Syntax	$\overset{\text{Interpretation}}{\longrightarrow}$	**Semantik**
Tupelvariable	$\longrightarrow$	konkrete Tupel
Atom	$\longrightarrow$	$\{$true, false$\}$
Formel	$\longrightarrow$	$\{$true, false$\}$
Ausdruck	$\longrightarrow$	Relation, d.h. Menge von Tupel für die ψ true ist

Semantik: Variablenbelegung

- Gegeben:
  - eine Tupelvariable t mit Schema $\mathit{Schema}(t)$
  - eine Formel $\psi(t)$, in der t frei vorkommt
  - ein beliebiges konkretes Tupel r (d.h. mit konkreten Werten), dabei muss r nicht zu einer Relation der DB gehören
- Bei der Belegung von t durch r in ψ wird jedes freie Vorkommen von t (Variable) durch r (konkretes Tupel) ersetzt: d.h. insbs. wird $t.A$ durch den Attributwert von $r.A$ ersetzt
- Wir schreiben $\psi(r \mid t)$ für die Belegung von t durch r in ψ

Semantik: Variablenbelegung

Beispiele

Gegeben das Relationenschema aus Kapitel 2 (Rel. Algebra):

Städte (*SName*: String, *SEinw*: Int, *Land*: String)

Länder (*LName*: String, *LEinw*: Int, *Partei*: String)

(Bei Koalitionsregierungen: jeweils eigenes Tupel fro RP in "Länder")

$$\psi(t) := (t.Land = \text{Bayern} \wedge t.SEinw \geq 500.000)$$

mit $Schema(t) = Schema(\text{Städte})$

- $r_1 = (\text{Passau}, 49.800, \text{Bayern})$:

  $\psi(r_1 \mid t) = (\text{Bayern} = \text{Bayern} \wedge 49.800 \geq 500.000)$
- $r_2 = (\text{Bremen}, 535.058, \text{Bremen})$:

  $\psi(r_1 \mid t) = (\text{Bremen} = \text{Bayern} \wedge 535.058 \geq 500.000)$

Semantik: Interpretation von Atomen

Die Interpretation $\mathcal{I}(\alpha)$ von Atomen α ist wie folgt definiert:

- Fall $\alpha = R(r)$:

  $\mathcal{I}(R(r))$ ergibt **true** gdw. r in R enthalten ist

- Fall $\alpha = c_1 \theta c_2$ mit Vergleichsoperator θ:

  $\mathcal{I}(c_1 \theta c_2)$ ergibt **true** gdw. der Vergleich durch θ erfüllt ist

  (c_1 und c_2 sind dabei Konstanten oder Tupelvariablen ggfls. mit Attributs-Einschränkungen)

Semantik: Interpretation von Formeln

Die Interpretation $\mathcal{I}(\psi)$ von Formeln ψ ist rekursiv über den Aufbau von Formeln definiert:

- Atome, d.h. $\psi = \alpha$: $\mathcal{I}(\psi) = \mathcal{I}(\alpha)$
- Logische Operatoren:
  - Negation $\neg$:

    $\mathcal{I}(\neg\psi) = \textbf{true}$ gdw. $\mathcal{I}(\psi) = \textbf{false}$
  - Konjunktion $\wedge$:

    $\mathcal{I}(\psi_1 \wedge \psi_2) = \textbf{true}$ gdw. $\mathcal{I}(\psi_1) = \textbf{true}$ und $\mathcal{I}(\psi_2) = \textbf{true}$
  - Disjunktion $\vee$:

    $\mathcal{I}(\psi_1 \vee \psi_2) = \textbf{true}$ gdw. $\mathcal{I}(\psi_1) = \textbf{true}$ oder $\mathcal{I}(\psi_2) = \textbf{true}$ (oder beide **true**)
- Quantoren: coming up ...

Semantik: Interpretation

Beispiele: Interpretation von Formeln (ohne Quantoren)

- Atome:
  - $\mathcal{I}(\text{Städte}(\text{Passau}, 49.800, \text{Bayern})) = \textbf{true}$

    da $(\text{Passau}, 49.800, \text{Bayern}) \in \text{Städte}$
  - $\mathcal{I}(49.800 \geq 500.000) = \textbf{false}$
- Operatoren:
  - $\mathcal{I}(\neg(49.800 \geq 500.000)) = \textbf{true}$

    da $\mathcal{I}(49.800 \geq 500.000) = \textbf{false}$
  - $\mathcal{I}(\text{Städte}(\text{Passau}, 49.800, \text{Bayern}) \vee 49.800 \geq 5000.00) = \textbf{true}$

    da $\mathcal{I}(\text{Städte}(\text{Passau}, 49.800, \text{Bayern})) = \textbf{true}$
  - $\mathcal{I}(\text{Städte}(\text{Passau}, 49.800, \text{Bayern}) \wedge 49.800 \geq 5000.00) = \textbf{false}$

    da $\mathcal{I}(49.800 \geq 500.000) = \textbf{false}$

Semantik: Interpretation von Formeln

Fortsetzung: Die Interpretation $\mathcal{I}(\psi)$ von Formeln ψ rekursiv über den Aufbau von Formeln:

- Quantoren:
  - Voraussetzung: in $(\exists t)(\psi)$ bzw. $(\forall t)(\psi)$ ist t die einzige freie Variable in ψ (also Schreibweise $\psi(t)$)
  - Existenz-Quantor $\exists$:

    $\mathcal{I}(\,(\exists t)(\psi(t))\,) = \textbf{true}$ gdw. es ein Tupel r mit $\mathit{Schema}(r) = \mathit{Schema}(t)$ gibt, für das gilt: $\mathcal{I}(\psi(r \mid t)) = \textbf{true}$
  - All-Quantor $\forall$:

    $\mathcal{I}(\,(\forall t)(\psi(t))\,) = \textbf{true}$ gdw. für alle Tupel r mit $\mathit{Schema}(r) = \mathit{Schema}(t)$ gilt: $\mathcal{I}(\psi(r \mid t)) = \textbf{true}$

Semantik: Interpretation

Beispiele Interpretation von Formeln (mit Quantoren)

- Quantoren:
  - $\mathcal{I}((\exists t)(\text{Städte}(t) \wedge t.\mathit{Land} = \text{Bayern})) = \textbf{true}$

    z.B. $r = (\text{Passau}, 49.800, \text{Bayern})$ erfüllt ψ
  - $\mathcal{I}((\forall t)(t.\mathit{Name} = \text{Passau})) = \textbf{false}$

    z.B. $r = (\text{Bremen}, 535.058, \text{Bremen})$ erfüllt ψ nicht

Semantik: Interpretation von Ausdrücken (Anfragen)

- Interpretation $\mathcal{I}(\{t \mid \psi(t)\})$ basiert wie besprochen auf
  - der Belegung der Variablen t durch konkrete Tupel
  - der dadurch möglichen Interpretation der Formel ψ

- Formal:

  Die Interpretation des Ausdrucks $\{t \mid \psi(t)\}$ die Menge aller (denkbaren)[2] Tupel r mit $\mathit{Schema}(r) = \mathit{Schema}(t)$ für die gilt:

$$\mathcal{I}(\psi(r \mid t)) = \textbf{true}$$

[2] Grundmenge sind hier nicht nur die gespeicherten Tupel in der DB

Beispiel-Anfragen

Running Example

Gegeben das Relationen-Schema aus Kapitel 2 (Rel. Algebra):

Städte (*SName*: String, *SEinw*: Int, *Land*: String)

Länder (*LName*: String, *LEinw*: Int, *Partei*: String)

(Bei Koalitionsregierungen: jeweils eigenes Tupel fro RP in "Länder")

- Finde alle Großstädte (*SName*, *SEinw*, *Land*) in Bayern:

  $Schema(t) = Schema(\text{Städte})$

  $\{t \mid \text{Städte}(t) \wedge t.Land = \text{Bayern} \wedge t.SEinw \geq 500.000\}$

- In welchem Land liegt Passau:

  $Schema(t) = (Land : \text{String})$

  $\{t \mid (\exists u \in \text{Städte})(u.SName = \text{Passau} \wedge u.Land = t.Land)\}$

Beispiel-Anfragen

Running Example

Gegeben das Relationen-Schema aus Kapitel 2 (Rel. Algebra):

Städte (*SName*: String, *SEinw*: Int, *Land*: String)

Länder (*LName*: String, *LEinw*: Int, *Partei*: String)

- Finde alle Städte in von Partei xyz (mit-)regiert Ländern:

  $Schema(t) = Schema(\text{Städte})$

  $\{t \mid \text{Städte}(t) \wedge (\exists u \in \text{Länder})(u.LName = t.Land \wedge u.Partei = xyz)\}$

- Welche Länder werden von Partei xyz alleine regiert:

  $Schema(t) = Schema(\text{Länder})$

  $\{t \mid \text{Länder}(t) \wedge (\forall u \in \text{Länder})(\neg(u.LName = t.LName \wedge u.Partei \neq xyz))\}$

Sichere Ausdrücke

- Mit den bisherigen Definitionen ist es möglich, unendliche Relationen zu beschreiben
- Beispiel:

  $Schema(t) = (String, String)$

  $\{t \mid t.1 = 1.2\}$

  Ergebnis:
  $\{(A, A), (B, B), ...(AA, AA), (BB, BB), ...(AB, AB), ...\}$
- Probleme
  - Ergebnis kann nicht gespeichert werden
  - Ergebnis kann nicht in endlicher Zeit berechnet werden
- Ein Ausdruck heißt sicher, wenn jede Tupelvariable nur Werte einer gespeicherten Relation annehmen kann.

Agenda

1. Einleitung
2. Tupelkalkül
3. Bereichskalkül
4. Query by Example
5. Relationenkalkül und SQL

Definitionen

- Unterschied:
  - Tupelkalkül: Tupelvariablen t (ganze Tupel)
  - Bereichskalkül: Bereichsvariablen $x_1 : D_1, x_2 : D_2, ...$ für einzelne Attribute ("Bereich" = Wertebereich / Domäne)
- Ein Ausdruck hat die Form:

$$\{x_1, x_2, ... \mid \psi(x_1, x_2, ...)\}$$

"Die Menge aller Wertekombinationen $x_1, x_2, ...$, die ψ erfüllen"

- Atome haben entspr. die Form:
  - $R(x_1, x_2, ...)$: (Tupel $(x_1, x_2, ...)$ tritt in Relation R auf)
  - $x\theta y$: mit x, y Bereichsvariablen oder Konstanten und θ wie oben (Vergleichsoperator)
- Formeln (Syntax und Semantik) analog zum Tupelkalkül, Variablen werden nun mit Attributwerten belegt

Beispiel-Anfragen

Running Example

Gegeben das Relationenschema aus Kapitel 2 (Rel. Algebra):

Städte (*SName*: String, *SEinw*: Int, *Land*: String)

Länder (*LName*: String, *LEinw*: Int, *Partei*: String)

- In welchem Land liegt Passau:

  $\{x_3 \mid (\exists x_1, x_2)(\text{Städte}(x_1, x_2, x_3) \wedge x_1 = \text{Passau})\}$

- Finde alle Städte in von Partei xyz (mit-)regiert Ländern:

  $\{x_1 \mid (\exists x_2, x_3, y_1)(\text{Städte}(x_1, x_2, x_3) \wedge \text{Länder}(x_3, y_1, xyz))\}$

- Welche Länder werden von Partei xyz alleine regiert:

  $\{x_1 \mid (\exists x_2)(\text{Länder}(x_1, x_2, xyz) \wedge \neg(\exists y_3)(\text{Länder}(x_1, x_2, y_3) \wedge y_3 \neq xyz))\}$

Agenda

1. Einleitung
2. Tupelkalkül
3. Bereichskalkül
4. Query by Example
5. Relationenkalkül und SQL

Einleitung

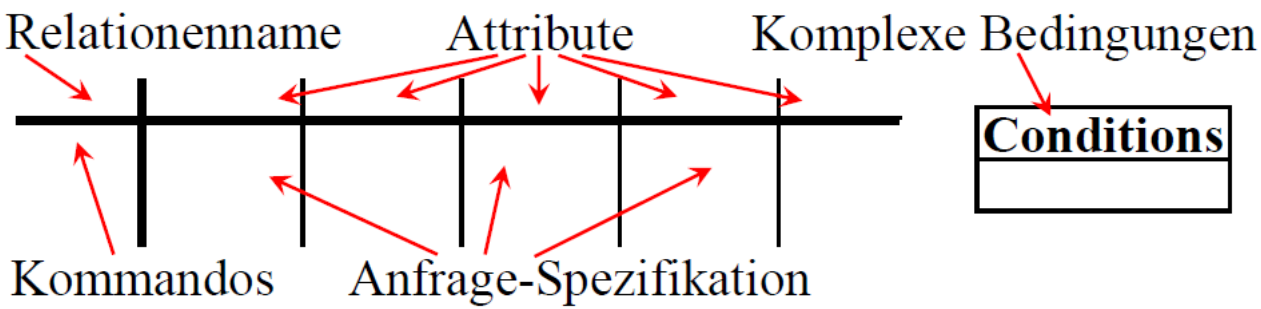
- Query by Example (QBE) ist eine Anfragesprache basierend auf dem Bereichskalkül
- Ausdrücke nicht wie in SQL als Text (der vom SQL-Intepreter geparsed wird)
- Dem Benutzer wird am Bildschirm ein Tabellen-Gerüst angeboten, das mit Spezial-Editor bearbeitet werden kann
- Nach Eintrag von Werten in das Tabellengerüst (Anfrage) füllt das System die Tabelle
- Zielgruppe: Gelegentliche Benutzer ohne Programmier-Kenntnisse
- QBE ist bei Weitem nicht so verbreitet wie SQL (statt QBE-Editoren werden heute i.d.R. Anwendungsprogramme mit entsprechenden GUIs implementiert), daher im folgenden nur ein Überblick ...

Sprachelemente

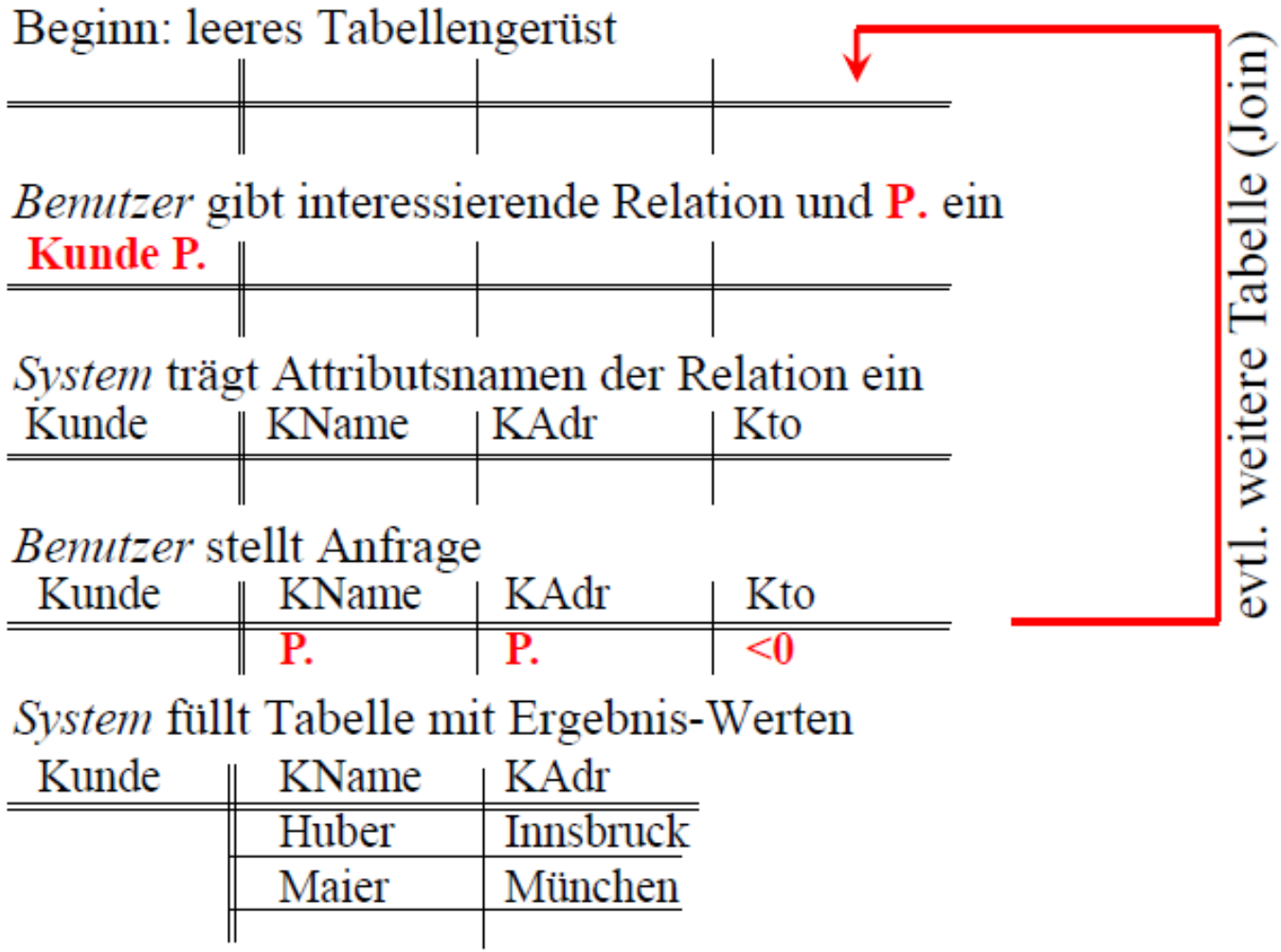
Folgende Sprachelemente können innerhalb des QBE-Editors typw. verwendet werden:

- Kommandos, z.B. **P.** (print), **I.** (insert), **D.** (delete) ...
- Bereichsvariablen (beginnen mit "_"), z.B. _x, _y
- Konstanten: z.B. Huber, Milch, 42
- Vergleichsoperatoren und arithmetische Operatoren
- Condition-Box: Zusätzlicher Kasten zum Eintragen einer Liste von Bedingungen (**AND**, **OR**, kein **NOT**)

Schematisch:



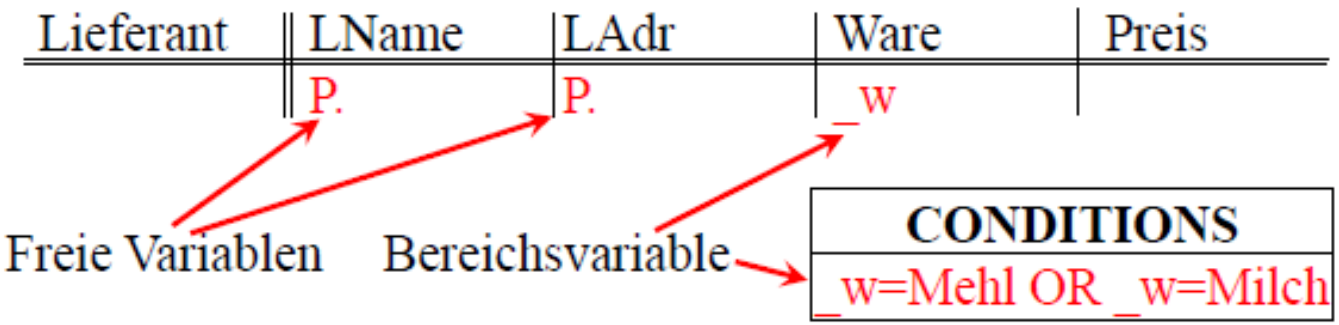
Beispiel-Dialog



Quelle: Skript zur Vorlesung "Datenbanksysteme" an der LMU München von Prof. Christian Böhm

Anfragen mit Bedingungen

- Welche Lieferanten liefern Mehl oder Milch?



- Bedeutung:

  $\{x_1, x_2 \mid (\exists w, x_4)(\text{Lieferant}(x_1, x_2, w, x_4) \wedge (w = \mathit{Mehl} \vee w = \mathit{Milch}))\}$

- Komando **P.** für "print" (entspricht einer Projektion)

Anfragen mit Bedingungen

- Welche Lieferanten liefern Brie und Perrier, wobei der Gesamtpreis 7 EUR nicht übersteigt?

Lieferant	LName	LAdr	Ware	Preis
	P._L		Brie	_y
	_L		Perrier	_z

CONDITIONS
_y + _z <= 7.00

- Bedeutung:

$$\{l \mid (\exists x_1, x_2, y, z)(\text{Lieferant}(l, x_1, \mathit{Brie}, y) \wedge \text{Lieferant}(l, x_2, \mathit{Perrier}, z) \wedge y + z \leq 7.00)\}$$

Join-Anfragen

- Welche Lieferanten liefern etwas, was Huber bestellt hat?

Lieferant	LName	LAdr	Ware	Preis
	P.		_w	

Auftrag	KName	Ware	Menge
	Huber	_w	

- Bedeutung:

$$\{l \mid (\exists x_2, w, x_4, y_3)(\text{Lieferant}(l, x_2, w, x_4) \wedge \text{Auftrag}(\textit{Huber}, w, y_3))\}$$

- Achtung: automatische Duplikat-Elemination in QBE

Anfragen mit Ungleichungen

- Wer liefert Milch zu einem Preis zwischen 0.50 EUR und 0.60 EUR?
  - Variante mit Condition-Box

Lieferant	LName	LAdr	Ware	Preis
P.			Milch	_p

CONDITIONS
_p >= 0.5 AND _p <= 0.6

  - Variante mit zwei Zeilen

Lieferant	LName	LAdr	Ware	Preis
P.	_L		Milch	>= 0.5
	_L		Milch	<= 0.6

Anfragen mit Negation

- Finde für jede Ware den billigsten Lieferanten

Lieferant	LName	LAdr	Ware	Preis
P.			_w	_p
¬			_w	< _p

- Das Symbol ¬ in erster Spalte bedeutet: es gibt kein solches Tupel
- Bedeutung:

$$\{x_1, x_2, w, p \mid (\neg\exists y_1, y_2, y_3)(\text{Lieferant}(x_1, x_2, w, p) \wedge \text{Lieferant}(y_1, y_2, w, y_3) \wedge y_3 < p)\}$$

Einfügen

- Einfügen einzelner Tupel: Kommando **I.**

Kunde	KName	KAdr	Kto
I.	Schulz	Wien	0

- Einfügen von Tupeln aus einem Anfrage-Ergebnis (hier: alle Lieferanten in Kundentabelle einfügen)

Kunde	KName	KAdr	Kto
I.	_n	_a	0

Lieferant	LName	LAdr	Ware	Preis
	_n	_a		

Löschen und Ändern

- Löschen aller Kunden mit negativem Kontostand

Kunde	KName	KAdr	Kto
D.			< 0

- Ändern eines Tupels

Kunde	KName	KAdr	Kto
	Schulz	Wien	U. 100

oder alternativ

Kunde	KName	KAdr	Kto
	Meier	_a	_k
U.	Meier	_a	_k – 110

oder mit Condition-Box

Vergleich

QBE	**Bereichskalkül**
Konstanten	Konstanten
Bereichsvariablen	Bereichsvariablen
leere Spalten	paarweise verschiedene Bereichsvariablen, $\exists$-quantifiziert
Spalten mit **P.**	freie Variablen
Spalten ohne **P.**	$\exists$-quantifizierte Variablen

Bemerkung: QBE ist relational vollständig, jedoch ist für manche Anfragen der relationalen Algebra eine Folge von QBE-Anfragen nötig

Agenda

1. Einleitung
2. Tupelkalkül
3. Bereichskalkül
4. Query by Example
5. Relationenkalkül und SQL

Quantoren in SQL

- Quantoren sind Konzept des Relationenkalküls
- In relationaler Algebra nicht vorhanden
- Können zwar simuliert werden
  - Existenzquantor implizit durch Join und Projektion
    $\{x \in R \,|\, \exists y \in S : ...\} \quad \equiv \quad \pi_{R.*}(\sigma...(R \times S))$
  - Allquantor mit Hilfe des Quotienten
    $\{x \in R \,|\, \forall y \in S : ...\} \quad \equiv \quad (\sigma...(R \times S)) \div S$
- Häufig Formulierung mit Quantoren natürlicher
- SQL: Quantifizierter Ausdruck in einer Subquery

Subqueries in SQL

- Beispiel für eine Subquery:

```
SELECT *
FROM Kunde
WHERE EXISTS (SELECT ... FROM ... WHERE ...)
```

Subquery

- In `WHERE`-Klausel der Subquery auch Zugriff auf Relationen/Attribute der Hauptquery
- Eindeutigkeit ggf. durch Aliasnamen für Relationen (wie z.B. bei Self-Join)

Existenz-Quantor in SQL

- Realisiert mit dem Schlüsselwort `EXISTS`
- Der $\exists$-quantifizierte Term wird in einer Subquery notiert
- Der Term wird **true** gdw. das Ergebnis der Subquery nicht leer ist
- Beispiel: KAdr der Kunden, zu denen ein Aufrag existiert:

```
SELECT KAdr FROM Kunde k WHERE EXISTS
    (SELECT * FROM Auftrag a
     WHERE a.KName = k.KName)
```

All-Quantor in SQL

- Keine direkte Unterstützung in SQL
- Aber leicht ausdrückbar durch die Äquivalenz:
  $\forall x : \psi(x) \Leftrightarrow \neg\exists x : \neg\psi(x)$
- Also Notation in SQL:
  ... `WHERE NOT EXISTS (SELECT ... FROM ... WHERE NOT ...)`
- Beispiel: Die Länder, die von der Partei "xyz" allein regiert werden:

```
SELECT * FROM Länder l1 WHERE NOT EXISTS
      (SELECT * FROM Länder l2
       WHERE l1.LName = l2.LName AND NOT l2.Partei = xyz)
```

Direkte Subquery

- An jeder Stelle in der `SELECT`-und `WHERE`-Klausel, an der ein konstanter Wert stehen kann, kann auch eine Subquery (`SELECT ... FROM ... WHERE ...`) stehen
- Einschränkungen:
  - Subquery darf nur ein Attribut ermitteln (Projektion)
  - Subquery darf nur ein Tupel ermitteln (Selektion)
- Beispiel: Dollarkurs aus Kurstabelle:

```
SELECT Preis, Preis *
      (SELECT Kurs FROM Devisen
       WHERE DName = US$ )AS USPreis) FROM Waren WHERE
...
```

- Oft schwierig, Eindeutigkeit zu gewährleisten ...

Subquery mit IN

- Nach dem Ausdruck A_i [NOT] IN ... kann stehen
  - Explizite Aufzählung von Werten:
    A_i IN {2,3,5,7}
  - Eine Subquery
    A_i IN (SELECT Wert FROM Primzahlen WHERE Wert $<=$7)
- Auswertung
  1. Ggfls. Subquery auswerten und in explizite Form (2,3,5,7) umschreiben
  2. In Hauptquery einsetzen
  3. Hauptquery auswerten