Universität Rostock

Traditio et Innovatio

Verteidigung Bachelorarbeit: Berechnung von Quasi-Identifikatoren nach Verbundoperationen

Florian Rose

Gliederung

- Aufgabenstellung
- Definition Quasi-Identifikator
- Motivation
- Berechnung von Quasi-Identifikatoren
- Herausforderung Verbundoperation
- Konzept
- Implementierung
- Evaluation
- Ausblick

Aufgabenstellung

- Quasi-Identifikatoren auf Verbundrelationen berechnen
- Informationen aus Ausgangsrelationen nutzen
- Praktische Anwendung an Beispieldatenbestand zeigen

Definition Quasi-Identifikator

Ein Quasi-Identifikator (QI) ist eine Kombination von Attributen einer Relation, die die Tupel der Relation fast identifiziert.

„fast“ bedeutet dabei:

- für die Attribute des QIs existieren maximal k Tupel mit den gleichen Werten

(vgl. Dalenius, Tore: Finding a Needle In a Haystack. 1986.)

bzw.

- die Attributkombination identifiziert p% der Tupel

Motivation

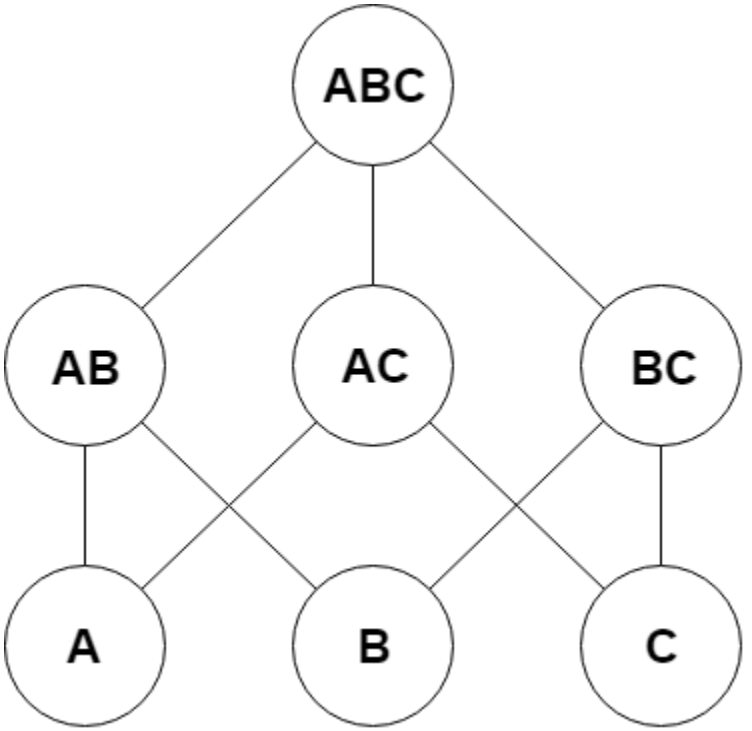
MatrNr.	Vorname	Nachname	Studiengang	Semester
1	Volker	Hagelstein	INF	3
2	Edmund	Hertel	WIN	3
3	Arthur	Ross	Lehramt für INF	5
4	Jakob	Schnell	ITTI	3
5	Natali	Geier	ET	5
6	Monika	Essig	BWL	3
7	Sven	Ferber	MA	5
8	Martin	Schüler	MA	7
9	Karin	Lang	CH	7
10	Erik	Kurz	MA	7

Berechnung von Quasi-Identifikatoren

- Gitternetz aller Attributkombinationen
- Ebenenweise Überprüfung der Knoten
- Für Attributkombinationen gilt:
  - $X \in QI \rightarrow \forall Y, Y \supset X : Y \in QI$
  - $X \notin QI \rightarrow \forall Y, Y \subset X : Y \notin QI$

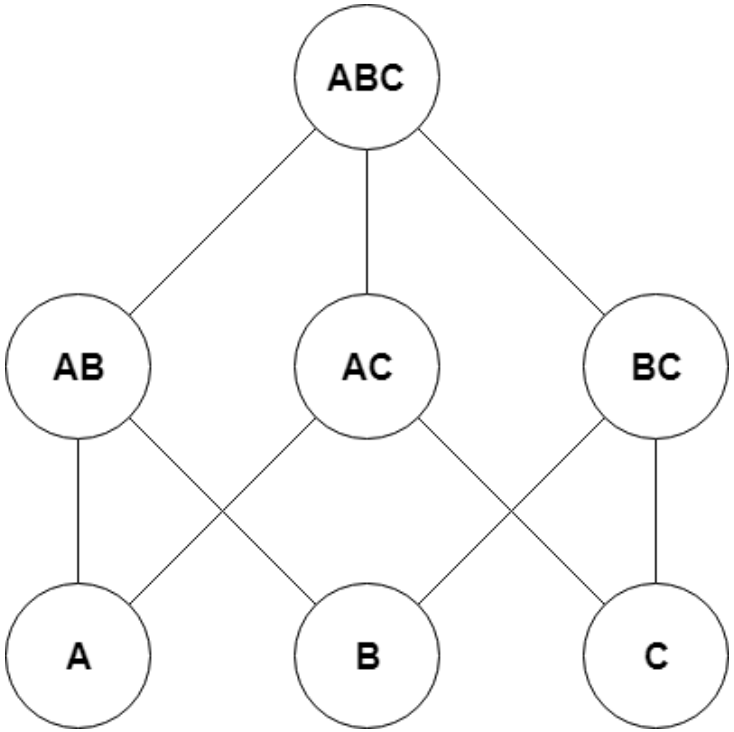
(R. Motwani and Y. Xu. Efficient algorithms for masking and finding quasi-identifiers. 2007.

Grunert, Hannes und Heuer, Andreas:
Big Data und der Fluch der Dimensionalität. 2014)



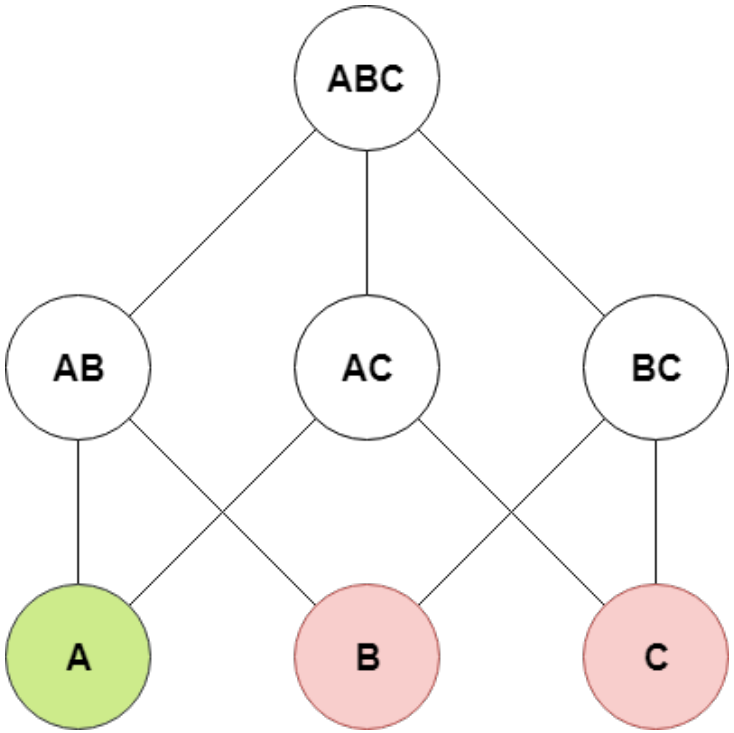
Berechnung von Quasi-Identifikatoren

Bottom-Up



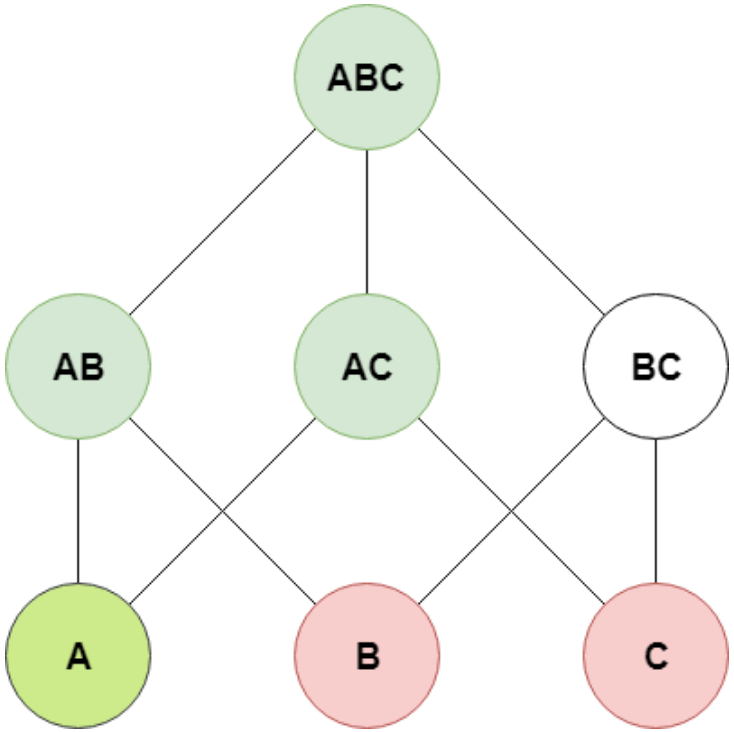
Berechnung von Quasi-Identifikatoren

Bottom-Up



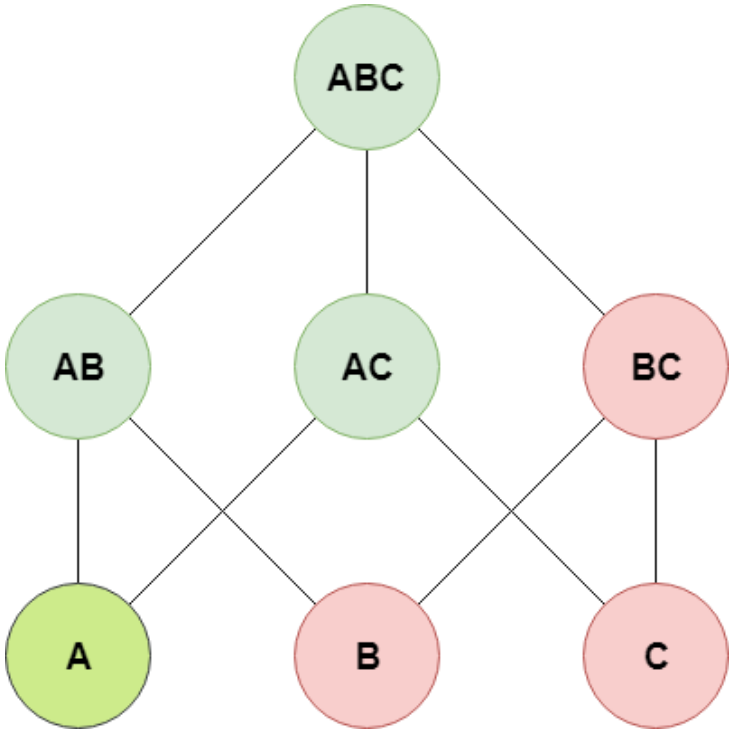
Berechnung von Quasi-Identifikatoren

Bottom-Up



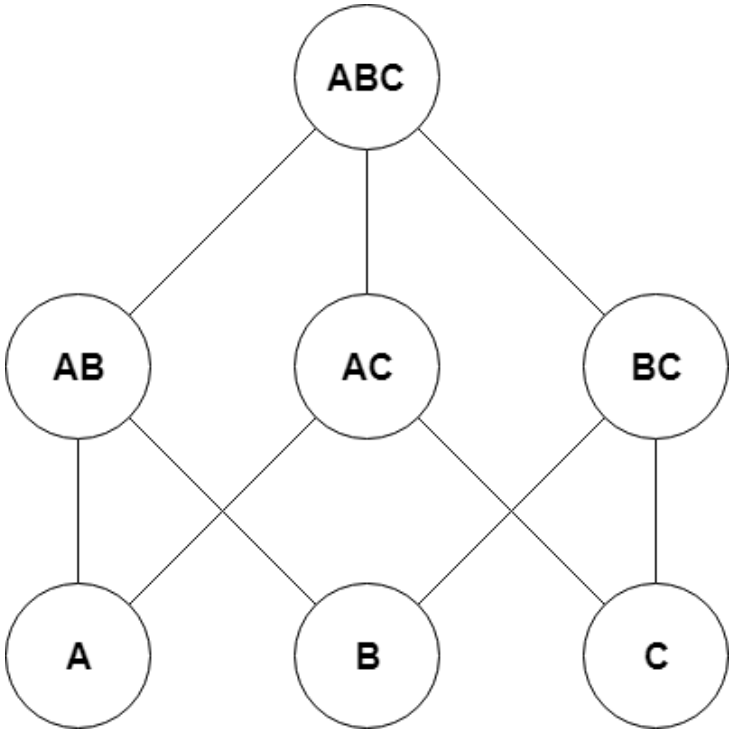
Berechnung von Quasi-Identifikatoren

Bottom-Up



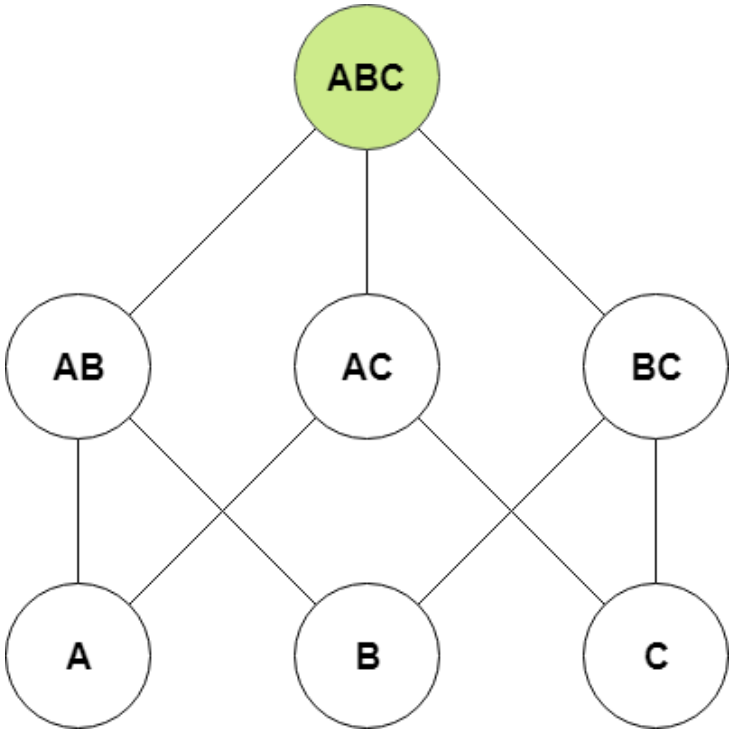
Berechnung von Quasi-Identifikatoren

Top-Down



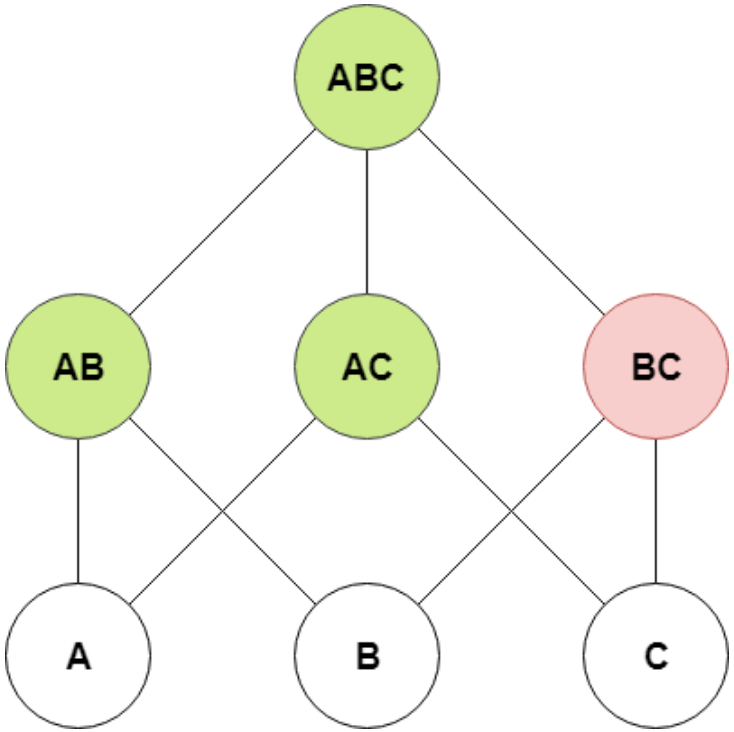
Berechnung von Quasi-Identifikatoren

Top-Down



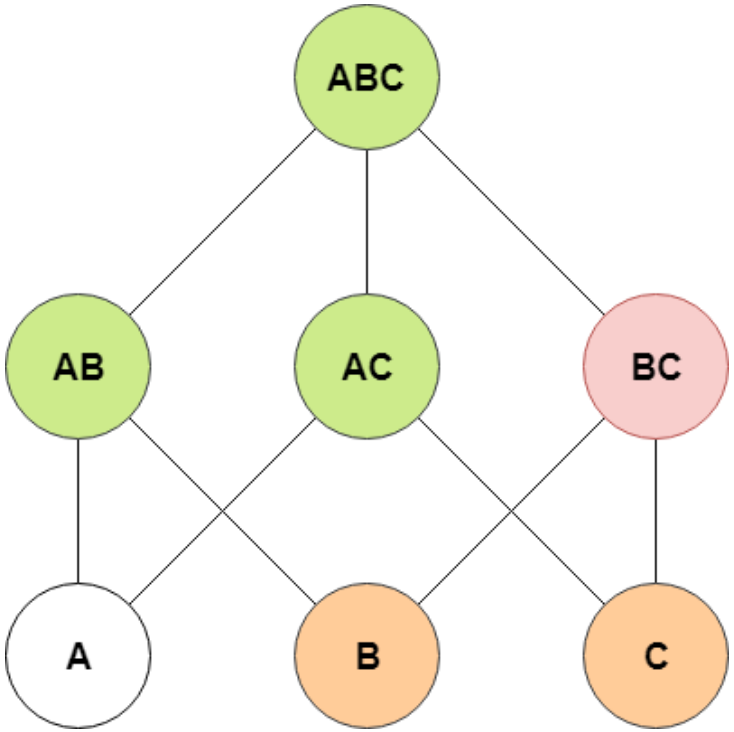
Berechnung von Quasi-Identifikatoren

Top-Down



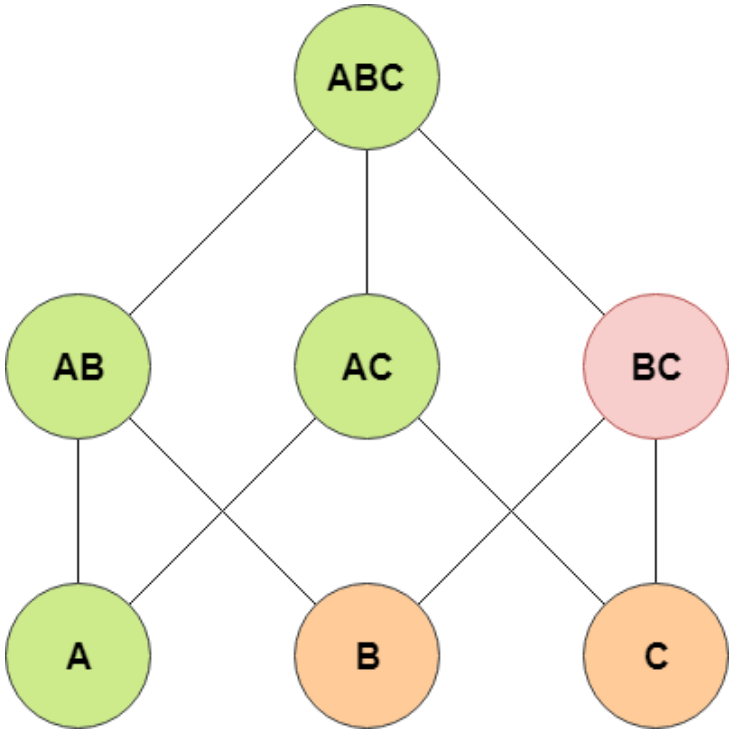
Berechnung von Quasi-Identifikatoren

Top-Down



Berechnung von Quasi-Identifikatoren

Top-Down



Herausforderung Verbundoperation

- QIs sind beliebige Kombinationen von Attributen
  → Suchraum ist Potenzmenge der Attributmenge abzüglich der leeren Menge
  → $2^{\#Attr} - 1$ zu untersuchende Attributkombinationen
- Im Nachfolgenden nur Equi-Joins betrachtet
- Bei einem Verbund werden die Attributmengen vereinigt
  → $2^{\#Attr_A + \#Attr_B} - 1$ zu untersuchende Attributkombinationen

Mit jedem neuen Attribut verdoppelt sich die Menge der zu untersuchenden Attributkombinationen

Konzept

- Existierenden Algorithmus um Herausfiltern bekannter (Nicht-)QIs erweitern
- Bekannte (Nicht-)QIs an Funktion übergeben (Ignore-Set)
- Berechnung dieser Attributkombinationen überspringen

Welche (Nicht-)QIs werden durch die Ausgangsrelationen impliziert?

Konzept

Fremdschlüsselbeziehung

- Fremdschlüsselbeziehung impliziert Existenz von Tupeln
- Jedes Tupel der referenzierenden Seite findet genau* einen Partner in der referenzierten Tabelle
- Verbund entspricht „Erweiterung“ der referenzierenden Tabelle
- QIs und Nicht-QIs der referenzierenden Tabelle gelten auch auf der Verbundrelation

MatrNr	Name	Vorname
1	Max	Mustermann
2	Erika	Musterfrau
3	John	Doe
4	Jane	Doe

MatrNr	Modul	Note
1	DB I	1.0
1	DB II	1.7
2	DB I	2.0
2	DB II	1.3

MatrNr	Name	Vorname	Modul	Note
1	Max	Mustermann	DB I	1.0
1	Max	Mustermann	DB II	1.7
2	Erika	Musterfrau	DB I	2.0
2	Erika	Musterfrau	DB II	1.3

* - in der Arbeit steht fälschlicher Weise “mindestens”

Konzept

Verteilung der Attributwerte

- Tupel, die bei einem Verbund keinen Partner finden, fallen heraus
- Herausfallende Tupel sind problematisch
  - Beeinflussen die Gesamtzahl der Ergebnistupel
  - Beeinflussen die Anzahl der Äquivalenzklassen für eine Attributkombination
- Attributkombination ist QI, falls ihr Identifikationswert größer gleich Grenzwert p ist
- $Identifikationswert = \frac{\#Äquivalenzklassen}{\#Tupel}$

Konzept

Verteilung der Attributwerte

Student

MatrNr.	Vorname	Nachname	Studiengang	Semester
1	Volker	Hagelstein	INF	3
2	Edmund	Hertel	WIN	3
3	Arthur	Ross	Lehramt für INF	5
4	Jakob	Schnell	ITTI	3
5	Natali	Geier	ET	5
6	Monika	Essig	BWL	3
7	Sven	Ferber	MA	5
8	Martin	Schüler	MA	7
9	Karin	Lang	CH	7
10	Erik	Kurz	MA	7

Modul

Fach	Studiengang
1	INF
2	WIN
3	Lehramt für INF
4	ITTI
5	ET
6	MA
7	CH

Konzept

Verteilung der Attributwerte

9 Tupel

MatrNr.	Vorname	Nachname	Studiengang	Semester	Fach
1	Volker	Hagelstein	INF	3	1
2	Edmund	Hertel	WIN	3	2
3	Arthur	Ross	Lehramt für INF	5	3
4	Jakob	Schnell	ITTI	3	4
5	Natali	Geier	ET	5	5
7	Sven	Ferber	MA	5	6
8	Martin	Schüler	MA	7	6
9	Karin	Lang	CH	7	7
10	Erik	Kurz	MA	7	6

Konzept

Verteilung der Attributwerte

- Ziel: Herausfinden ob Äquivalenzklassen herausfallen
- Lösungsansätze:
  - Vergleich Anzahl der Äquivalenzklassen
    ```
    SELECT COUNT(DISTINCT ROW(attr1, attr2, ...) FROM t
    ```
  - Überprüfe ob Tupel herausfallen
    ```
    SELECT joinAttr1
    FROM t1
    WHERE joinAttr1 NOT IN (
          SELECT joinAttr2
          FROM t2)
    ```
  - Fremdschlüsselbeziehung impliziert, dass für die referenzierende Relation keine Äquivalenzklassen herausfallen

Konzept

Verbundkardinalität

- Voraussetzung: keine herausfallenden Äquivalenzklassen
- 1-zu-1-Verbund: QIs beider Ausgangsrelationen gelten auf Verbundrelation
- 1-zu-n-Verbund: QIs der 1-Seite gelten auf der Verbundrelation
- n-zu-m-Verbund: keine QIs können als gültig angenommen werden

- falls keine Tupel herausfallen, gilt dies auch für Nicht-QIs

Implementierung

- Implementierung in PostgreSQL 11 (abwärtskompatibel bis 9.4)
- Implementierung eines angepassten Bottom-Up-Verfahrens
- PostgreSQL unterstützt keinen Datentyp für Mengen
- Benötigen Mengen von Mengen
  - → Simulierung von Mengen durch Arrays
  - → Alternativ möglich: Simulierung von Mengen durch temporäre Tabellen

Evaluation

Testumgebung

- Test der Implementation auf lokaler PostgreSQL-Installation
- TPCH-Datenbank mit zufällig generierten Einträgen als Testdatenbestand

Parameter (PC)	Wert
Hauptspeicher	8GB
CPU	4x 2.2 GHz (ohne Hyperthreading)
Festspeicher	250GB SSD

Parameter (Postgres)	Wert
shared_buffers	2GB
temp_buffers	80MB
work_mem	1024MB
maintenance_work_mem	24MB
max_stack_depth	2MB
max_files_per_process	1000

Evaluation

Laufzeiten

Relation	Anzahl Tupel	Anzahl Attribute	∅Ausführungszeit(hh:mm:ss.SSS)
LINEITEM	6.000.000	16	*
ORDERS	1.500.000	9	00:01:01.198
PARTSUPP	800.000	5	00:00:11.632
PART	200.000	9	00:00:10.676
CUSTOMER	150.000	8	00:00:01.665
SUPPLIER	10.000	7	00:00:00:101
NATION	25	4	00:00:00.016
REGION	5	3	00:00:00.015

Evaluation

Laufzeiten

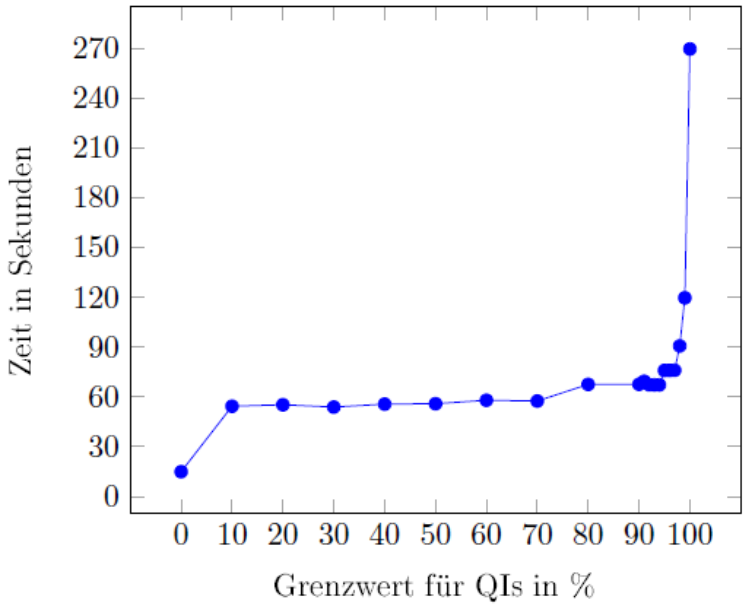
- Array-Implementation ist ineffizient
- Frage: Wie stark beeinflusst diese Umsetzung die Gesamtlaufzeit?

Relation	Anteil $t_{\text{Berechnung QI}}$	Anteil $t_{\text{Erweiterung lowerSet}}$
ORDERS	99,51%	00,02%
PARTSUPP	91,37%	00,12%
PART	95,53%	00,12%
CUSTOMER	98,19%	00,19%

Evaluation

Laufzeiten

- QI-Berechnung stark abhängig von den Daten
- Grenzwert für QIs hat ebenfalls großen Einfluss

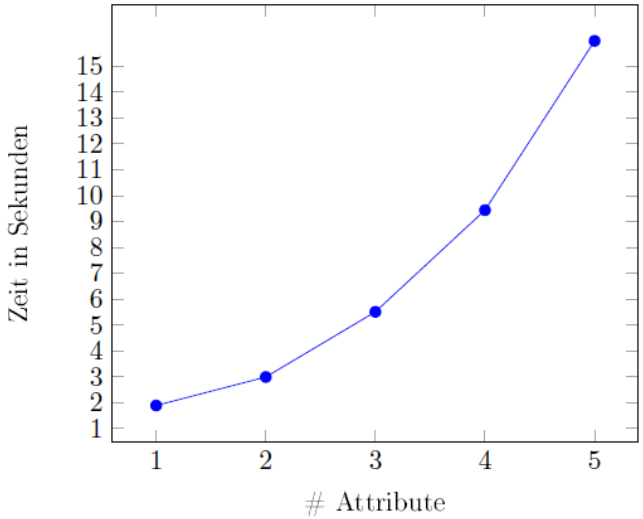


Laufzeit: qi_detect('orders',threshold)

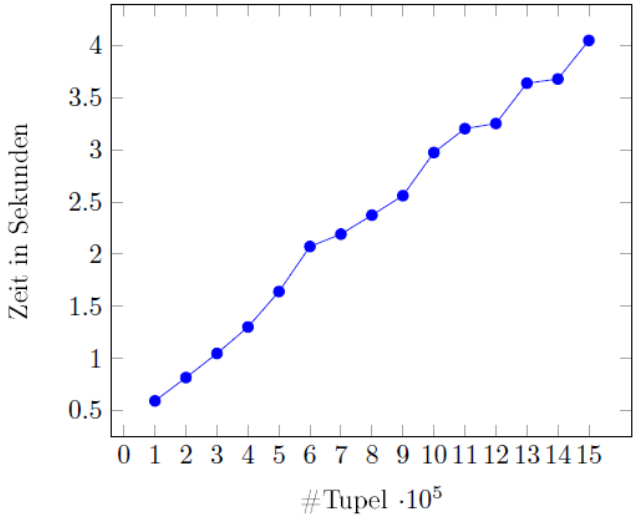
Evaluation

Laufzeiten

- Tupel- und Spaltenanzahl wichtige Größe für Laufzeit
- Theoretisch exponenziell mit Spaltenanzahl, Linear mit Tupelzahl



Laufzeit: qi_detect('partsupp',0.70)



Laufzeit: Berechnung eCount

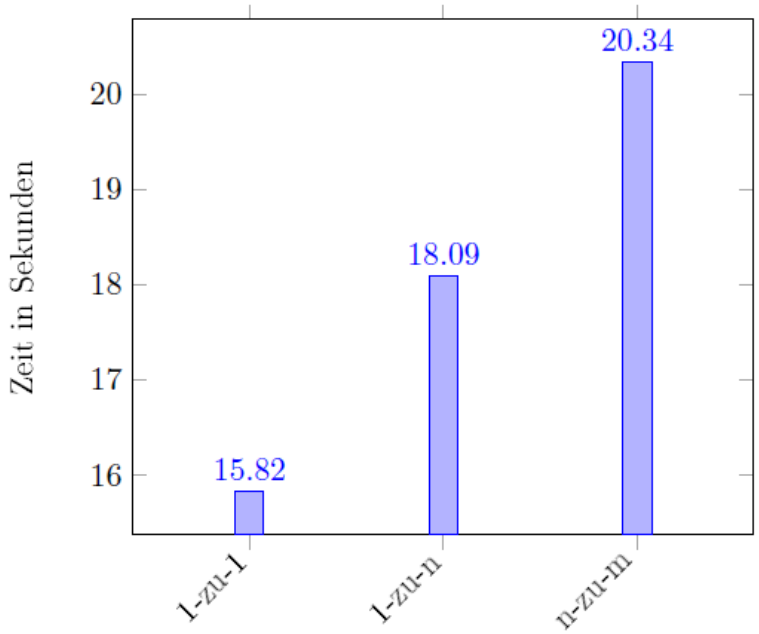
Evaluation

Laufzeit

- Lange Laufzeit für die Bestimmung der Verbundkardinalität
- Deshalb diesen Teilschritt für Berechnung auf Verbundrelation entfernt
- Folglich nur noch zwischen 1-zu-n- und n-zu-m-Verbund unterscheidbar
- Daher Berechnung nach Verbundart in einzelnen Funktionen betrachtet

Evaluation

Laufzeit



Laufzeit: qi_detect_join nach Verbundkardinalität

Evaluation

Vor- und Nachteile der Implementierung

+ Datennahe Auswertung
+ Keine Kommunikation über das Internet, kein Kommunikationsoverhead
+ Möglichkeit QI-Berechnung automatisch über Trigger anzustoßen
+ QIs können auf Datenbank gehalten werden
  + Zugriff kann mit SQL beschränkt werden
  + z. B. dürfen bestimmte Nutzer sensible Attribute und QIs nicht zusammen auslesen

- Datentyp für Menge fehlt in einigen Datenbanksystemen
  - Umständlichere Umsetzung durch Simulierung von Mengen
  - Code wird dadurch unübersichtlich
- Starre Implementierung

Ausblick

- Flexiblere Implementation
  - Implementation für beliebige Verbunde
  - Implementation für ganze SQL-Anfragen
- Optimierung der Implementation
  - Datenbanksystem, welches Set-Typen unterstützt (z. B. Oracle)
  - Bitmap-Tabellen vs. Arrays
  - Bottom-Up + Top-Down
- Berechnung während des Verbundes nur in Ansätzen untersucht (nur für Masterarbeit gefordert)

Vielen Dank für Ihre Aufmerksamkeit!

Implementierung

QI-Berechnung für eine Tabelle

```
SELECT ARRAY(
        SELECT column_name
        FROM information_schema.columns
        WHERE table_name = tableName)
INTO attributeList
FOREACH attr in attributeList
        lowerSet := lowerSet || attr
SELECT COUNT(*) FROM tableName INTO tCount
WHILE lowerSet NOT empty
        copySet := lowerSet
        FOREACH subset IN copySet
                SELECT COUNT(DISTINCT subset) FROM tableName INTO eCount
                IF (eCount/tCount)>= threshold THEN
                        qiList := qiList || subset
                lowerSet := extend(copySet)
```

Hier können auch Statistiken der Datenbank verwendet werden. Diese sind aber nicht genau und lassen daher Ungenauigkeiten zu.

Hier können der ROW-Operator oder Konkatenation benutzt werden.

Jeden Nicht-QI im *copySet* um ein Attribut erweitern, Duplikate und Attributkombinationen, die QIs überdecken entfernen

Implementierung

QI-Berechnung für einen Verbund

```
SELECT ARRAY(
        SELECT column_name
        FROM information_schema.columns
        WHERE table_name = tableName1 OR table_name = tableName2)
INTO attributeList
CREATE TEMP TABLE temp AS
        SELECT *
        FROM tableName1 t1, tableName2 t2
        WHERE t1.joinAttr1 = t2.joinAttr2
qiList := knownQIs
ignoreSet := knownNonQIs || knownQIs
SELECT COUNT(*) FROM temp INTO tCount
...
```

Implementierung

QI-Berechnung für einen Verbund

```
WHILE lowerSet NOT empty
    copySet := lowerSet
    FOREACH subset IN copySet
        SELECT COUNT(DISTINCT subset) FROM tableName INTO eCount
        IF subset NOT IN ignoreSet THEN
            IF (eCount/tCount)>= threshold THEN
                qiList := qiList || subset
        lowerSet := extend(copySet)
```

Implementierung

QI-Berechnung für einen Verbund

- Wichtig: Auswahl der zu verwendenden QIs
- Finden von Fremdschlüsselbeziehungen
- Berechnung der Verbundkardinalität
- Berechnung von herausfallenden Tupeln

Implementation

QI-Berechnung für einen Verbund

```
IF references(tableName1, joinAttr1, tableName2, joinAttr2) THEN
    IF joinCardinality(tableName1, joinAttr1, tableName2, joinAttr2) is 1-to-1 THEN
        IF NOT(tupleDropOut(tableName1, joinAttr1, tableName2, joinAttr2) THEN
            knownQIs := detectQIs(tableName1) || detectQIs(tableName2)
            knownNonQIs := powerSet(tableName1) || powerSet(tableName2) – knownQIs
        ELSE
            knownQIs := detectQIs(tableName1)
            knownNonQIs := powerSet(tableName1) – knownQIs
    ELSE
        knownQIs := detectQIs(tableName1)
        knownNonQIs := powerSet(tableName1)
...
```

Kommen im besten Fall aus einer designierten Tabelle

Implementation

QI-Berechnung für einen Verbund

```
ELSE IF references(tableName2, joinAttr2, tableName1, joinAttr1) THEN
    IF joinCardinality(tableName2, joinAttr2, tableName1, joinAttr1) is 1-to-1 THEN
        IF NOT(tupleDropOut(tableName2, joinAttr2, tableName1, joinAttr1) THEN
            knownQIs := detectQIs(tableName1) || detectQIs(tableName2)
            knownNonQIs := powerSet(tableName1) || powerSet(tableName2) – knownQIs
        ELSE
            knownQIs := detectQIs(tableName2)
            knownNonQIs := powerSet(tableName2) – knownQIs
    ELSE
        knownQIs := detectQIs(tableName2)
        knownNonQIs := powerSet(tableName2)
ELSE
    knownQIs := emptySet
    knownNonQIs := emptySet
detectJoinQIs(tableName1, tableName2, joinAttr1, joinAttr2, knownQIs, knownNonQIs)
```