Christian Albrechts Universität zu Kiel
Institut für Informatik
AG Archäoinformatik (Data Science)

Skript zur Vorlesung

Informationssysteme

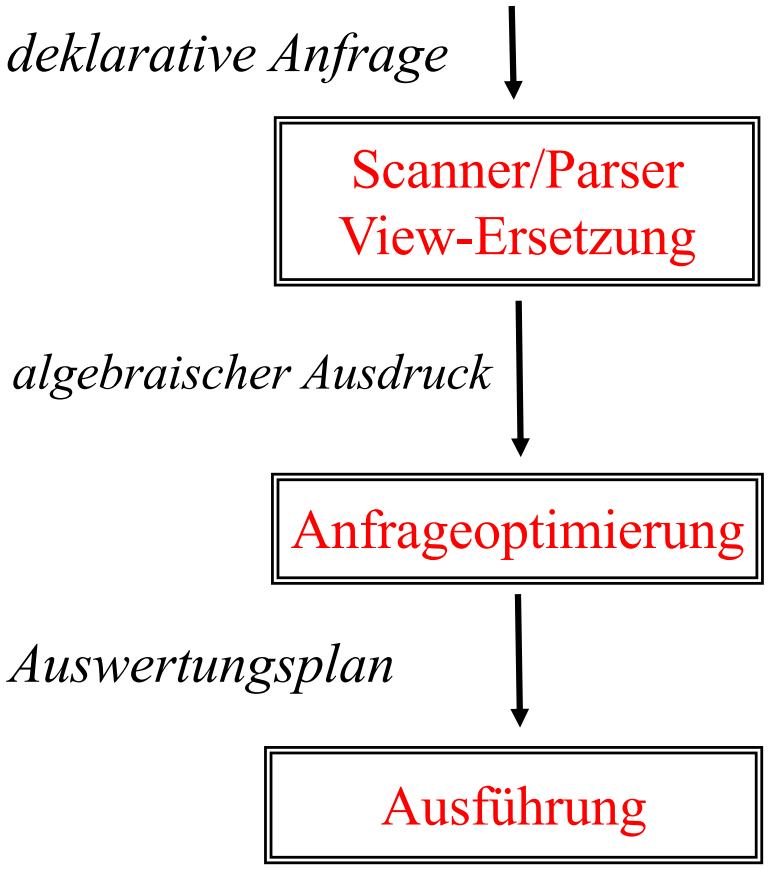
Wintersemester 2019/20

Kapitel 11: Relationale Anfragebearbeitung

(basiert auf dem Skript der LMU München)

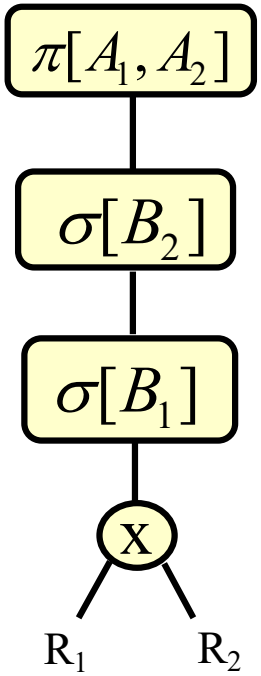
Relationale Anfragebearbeitung

Zentrale Aufgabe der Anfragebearbeitung ist die Übersetzung der *deklarativen* Anfrage in einen *effizienten, prozeduralen* Auswertungsplan



Relationale Anfragebearbeitung

Kanonischer Auswertungsplan zu einer SQL-Anfrage
(Ergebnis der ersten Übersetzungsphase)



select A_1, A_2, …

from R_1, R_2, …

where B_1 and B_2,

…

1. Bilde das kartesische Produkt der Relationen R_1, R_2, …
2. Führe Selektionen mit den einzelnen Bedingungen B_1, B_2, … durch.
3. Projiziere die Ergebnistupel auf die erforderlichen Attribute A_1, A_2, …

$$\pi_{A_1,A_2}(\sigma_{B_2}(\sigma_{B_1}(R_1 \times R_2)))$$

Relationale Anfragebearbeitung

Beispiel Autodatenbank

Kunden(KNr, Name, Adresse, Region, Saldo)

KNr	Name	Adresse	Region	Saldo
201	Klein	Lilienthal	Bremen	200.000
337	Horn	Dieburg	Rhein-Main	100.000
444	Berger	München	München	300.000
108	Weiss	Würzburg	Unterfranken	500.000

View GuteKunden(KNr, Name, Adresse, Region, Saldo) =

select * from Kunden where Saldo ≥ 300.000

Bestellt(BNr, Datum, KNr, Region, Saldo)

BNr	Datum	KNr	PNr
221	10.05.04	201	12
312	11.05.04	201	4
401	20.05.04	337	330
456	13.05.04	444	330
458	14.05.04	444	98

Produkt(PNr, Bezeichnung, Anzahl, Preis)

PNr	Bezeichnung	Anzahl	Preis
12	BMW 318i	10	40.000
4	Golf 5	40	25.000
330	Fiat Uno	5	18.000
98	Ferrari 380	1	180.000
14	Opel Corsa	14	17.000

Relationale Anfragebearbeitung

Einfache SQL-Anfrage:
Welche guten Kunden (Name) haben einen Fiat Uno bestellt (und Saldo ≥ 300.000) ?

```
select   Name
from     GuteKunden k, Bestellt b, Produkt p
where    b.KNr = k.KNr
and      b.PNr = p.PNr
and      Bezeichnung = ‚Fiat Uno‘
```

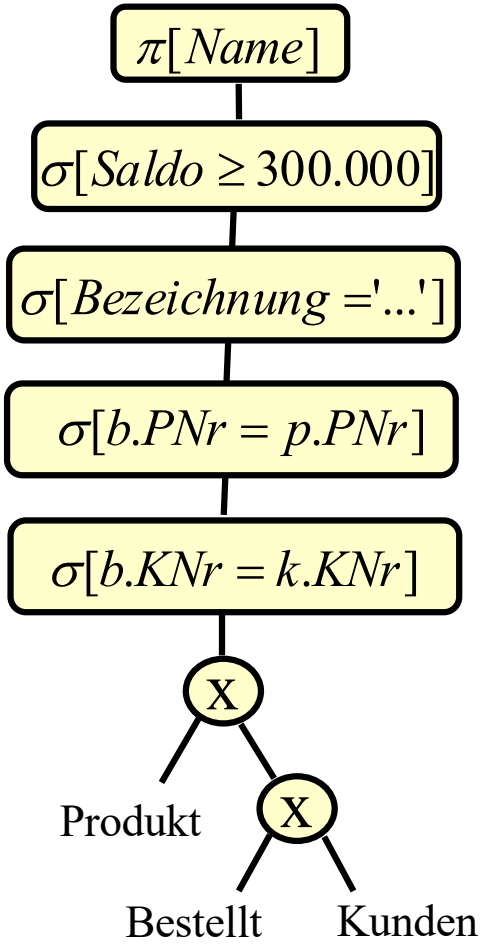
Expansion der View:

```
select   Name
from     Kunden k, Bestellt b, Produkt p
where    b.KNr = k.KNr
and      b.PNr = p.PNr
and      Bezeichnung = ‚Fiat Uno‘
and      Saldo ≥ 300.000
```

Relationale Anfragebearbeitung

Übersetzung in relationale Algebra (kanonisch):

$$\pi_{Name}(\sigma_{Saldo \geq 30000}(\sigma_{Bezeichung='FiatUno'}(\sigma_{b.PNr=p.PNr}(\sigma_{b.KNr=k.KNr}(\mathrm{Pr}\,odukt \times (Bestellt \times Kunden))))))$$



Relationale Anfragebearbeitung

Kunden:

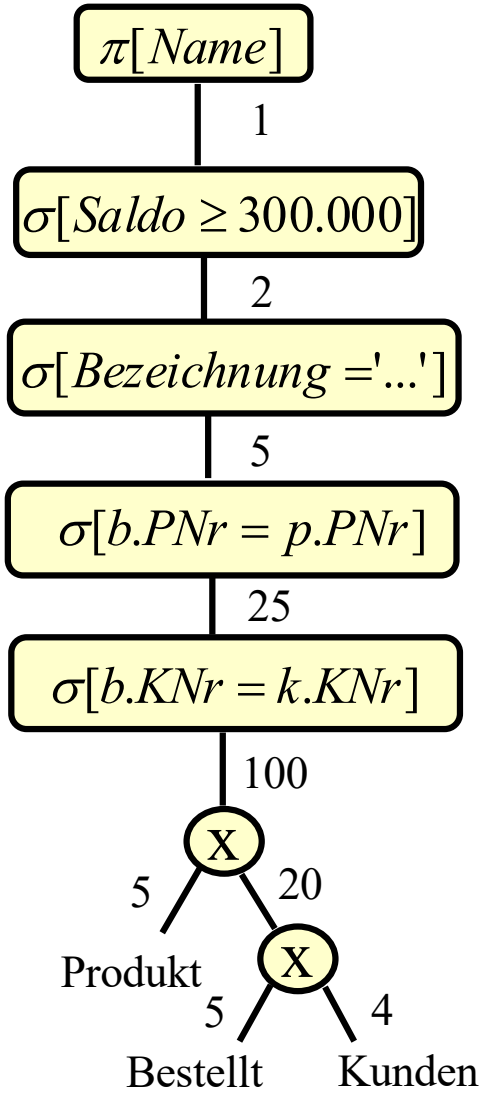
KNr	Name	Adresse	Region	Saldo
201	Klein	Lilienthal	Bremen	200.000
337	Horn	Dieburg	Rhein-Main	100.000
444	Berger	München	München	300.000
108	Weiss	Würzburg	Unterfranken	500.000

Bestellt:

BNr	Datum	KNr	PNr
221	10.05.04	201	12
312	11.05.04	201	4
401	20.05.04	337	330
456	13.05.04	444	330
458	14.05.04	444	98

Produkt:

PNr	Bezeichnung	Anzahl	Preis
12	BMW 318i	10	40.000
4	Golf 5	40	25.000
330	Fiat Uno	5	18.000
98	Ferrari 380	1	180.000
14	Opel Corsa	14	17.000

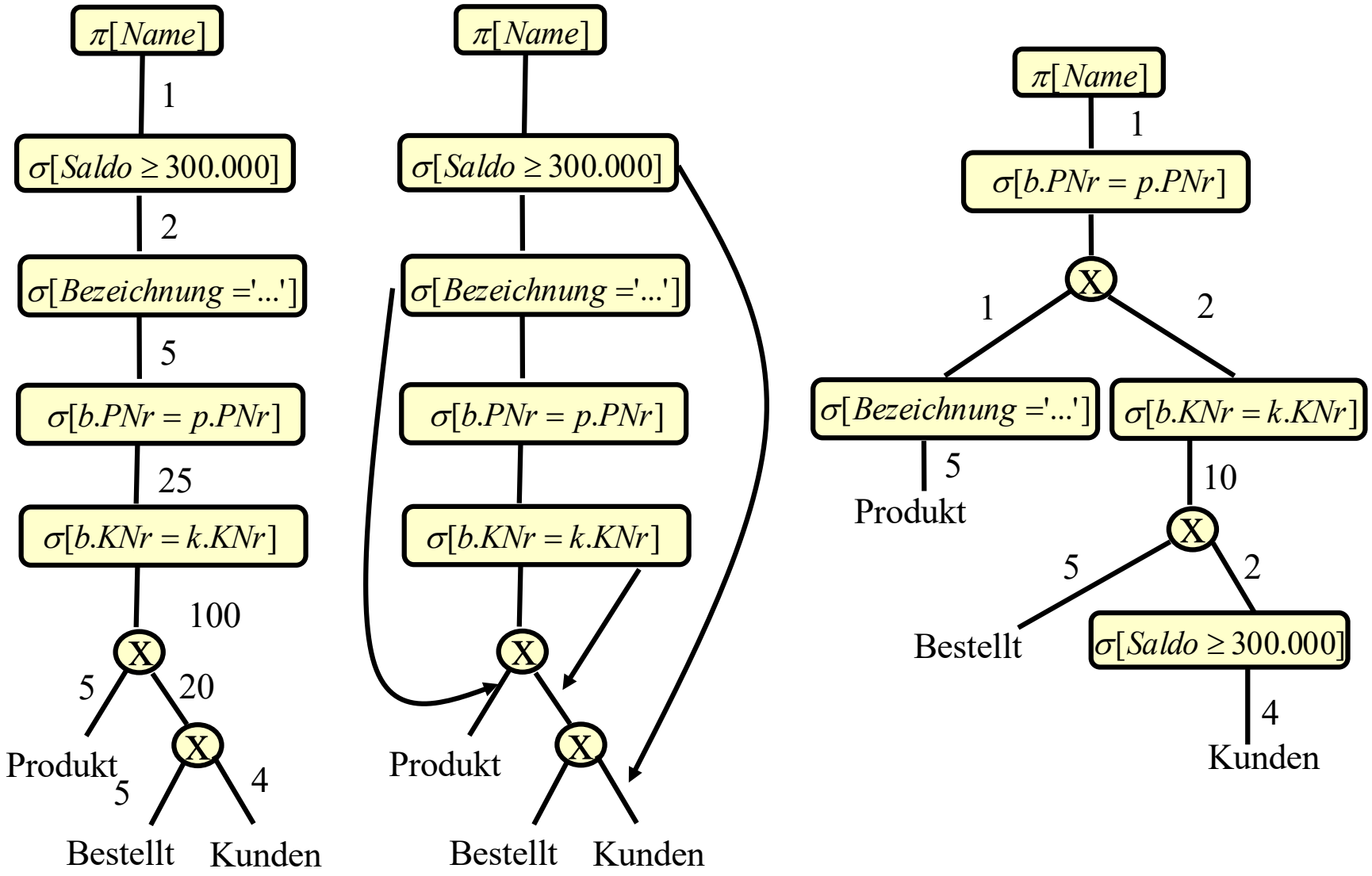


Relationale Anfragebearbeitung

Beobachtungen:

- Der kanonische Auswertungsplan erzeugt das kartesische Produkt der 3 Relationen
- Die Kardinalität des kartesischen Produkts ist |Kunden| * |Bestellt| * |Produkt| = 100 Tupel
- Für jedes der 100 Tupel muss z.B. die Bedingung b.KNr=k.KNr ausgewertet werden
- Günstiger wäre es z.B., wenn man sich gleich von Anfang an auf das Produkt ‚Fiat Uno' und die Kunden mit hohem Saldo beschränken würde:

Relationale Anfragebearbeitung



Relationale Anfragebearbeitung

- i.A. gibt es viele verschiedene, *gleichwertige* Auswertungspläne für dieselbe Anfrage
- Die Performanz gleichwertiger Auswertungspläne variiert häufig zwischen wenigen Sekunden (schnellster Plan) und vielen Stunden (Standardplan)
- Die Aufgabe der Anfrageoptimierung ist es, den günstigsten Auswertungsplan zu ermitteln (bzw. zumindest einen sehr günstigen Plan zu ermitteln)
- Wegen des großen Unterschiedes zwischen günstigstem und ungünstigstem Plan ist die Optimierung bei der relationalen Anfragebearbeitung wesentlich wichtiger als z.B. bei der Übersetzung von (imperativen) Programmiersprachen

Relationale Anfragebearbeitung

Logische und physische Anfrageoptimierung:

- Optimierungstechniken, die den Auswertungsplan betrachten und „umbauen“ werden als logische Anfrageoptimierung bezeichnet
- Physische Anfrageoptimierung: Auswahl einer geeigneten Auswertungsstrategie für Join-Operationen oder Entscheidung, ob für eine Selektionsoperation ein Index verwendet wird.

Beispiel: Auswertungsstrategien für Joins

- Erzeuge alle Tupel des kartesischen Produkts und prüfe Join-Bedingung (Nested Loop)
- Sortiere beide Relationen nach dem Joinattribut und filtere passende Paare (Sort Merge)
- Betrachte alle Tupel der einen Relation und greife auf die Joinpartner über einen passenden Index der anderen Relation zu (Indexed Loop)

Relationale Anfragebearbeitung

Regel- und kostenbasierte Optimierung

- Es gibt zahlreiche Regeln (Heuristiken), um die Reihenfolge der Operatoren im Auswertungsplan zu modifizieren und so eine Performanz-Verbesserung zu erreichen, z.B. Push Selection: Führe Selektionen möglichst frühzeitig (vor Joins) aus

  Optimierer, die sich ausschließlich nach solchen starren Regeln richten, nennt man **regelbasierte Optimierer** oder auch algebraische Optimierer.

Relationale Anfragebearbeitung

- Kostenbasierte Anfrageoptimierung
  - Optimierer, die die voraussichtliche Performanz von Auswertungsplänen ermitteln, werden als **kostenbasierte Optimierer** bezeichnet.
    Die Vorgehensweise ist meist folgende:
    1. Generiere einen initialen Plan (z.B. Standardauswertungsplan)
    2. Schätze bei der Auswertung entstehende Kosten
    3. Modifiziere den aktuellen Plan gemäß vorgegebener Heuristiken
    4. Wiederhole die Schritte 2 und 3 bis ein Stop-Kriterium erreicht ist
    5. Gib den besten erhaltenen Plan aus
  - Als Kostenmaß eignen sich der
    - Erwartungswert der **Antwortzeit** (Einbenutzerbetrieb) oder
    - die Belegung von **Ressourcen,** wie z.B. Anzahl zugegriffener Blöcke oder CPU-Nutzung (Durchsatz-Optimierung v.a. im Mehrbenutzerbetrieb)

Relationale Anfragebearbeitung

- Kostenbasierte Anfrageoptimierung
  - Schätzung der bei der Ausführung des Anfrageplans entstehende Kosten mittels Abschätzung der Kosten der Einzeloperationen (Kartesisches Produkt, Join, Selektion, …)
  - Bei Operationen mit Zugriff auf gesamte Relationen (z.B. bei Kartesischem Produkt), werden die Kosten über die Tabellengrößen ermittelt.
    Kosten für $R \times S = |R|*|S|$
  - Bei Operationen mit Zugriff auf teile von Relationen (z.B. bei Join, Selektion, …), werden die Kosten über Selektivitätsabschätzungen ermittelt.

Relationale Anfragebearbeitung

- Selektivität
  - Mit Selektivität *sel* bezeichnet man die relative Anzahl der qualifizierenden Tupel in den betroffen Relationen.
  - Für Selektion und Join ist die Selektivität *sel* folgendermaßen definiert:

  - Selektion mit Bedingung B:

$$sel_{\sigma_B}(R) = \frac{|\sigma_B(R)|}{|R|}$$

(relativer Anteil der Tupel, die B erfüllen)

  - Join von R und S:

$$sel_{R \bowtie S} = \frac{|R \bowtie S|}{|R \times S|} = \frac{|R \bowtie S|}{|R| \cdot |S|}$$

Relationale Anfragebearbeitung

- Selektivität
  - Die Selektivität muss geschätzt werden, für Spezialfälle gibt es einfache Methoden:
  - S1: Die Selektivität von $\sigma_{R.A=c}$, also Vergleich mit einer Konstante c, beträgt
    - 1 / |R|, falls A ein **Schlüssel** ist
    - 1/I, falls A **kein Schlüssel** ist, aber die Werte gleichverteilt sind.

    (I ist dabei die Anzahl versch. A-Werte in R)
  - S2: Besitzt bei einem Equi-Join von R und S (Join Bed.: R.A=S.B) das Attribut A Schlüsseleigenschaft, kann die Größe des Join-Ergebnisses mit |S| abgeschätzt werden, da jedes Tupel aus S maximal einen Joinpartner findet.

Relationale Anfragebearbeitung

- Selektivität

  Selektivität bei Selektion mit verknüpften Bedingungen:

  – logisches UND: $sel_{\sigma_{B1 \wedge B2}}(R) = sel_{\sigma_{B1}} \cdot sel_{\sigma_{B2}}$

  – logisches ODER:
  $sel_{\sigma_{B1 \vee B2}}(R) = sel_{\sigma_{B1}} + sel_{\sigma_{B2}} - sel_{\sigma_{B1}} \cdot sel_{\sigma_{B2}}$

  – logisches NICHT: $sel_{\sigma_{\neg B1}} = 1 - sel_{\sigma_{B1}}$

Relationale Anfragebearbeitung

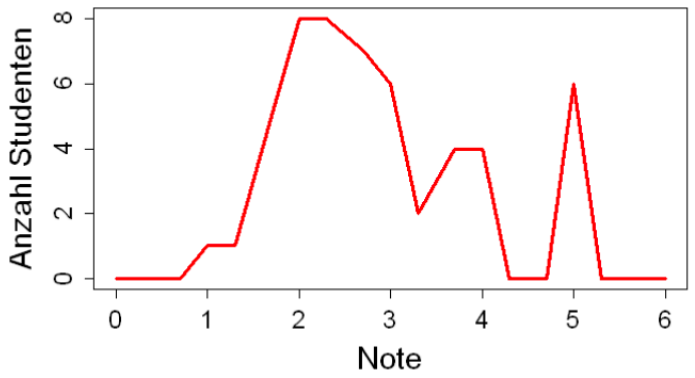
- Selektivität

  Im Allgemeinen benötigt man anspruchsvollere Methoden um zu schätzen, wieviele Tupel sich in einem bestimmten Wertebereich befinden.

  Drei Grundsätzliche Arten von Schätzmethoden:
  - Parametrische Verteilungen
  - Histogramme
  - Stichproben

  Beispiel:
  Schätzung der Verteilung der Noten einer Klausur anhand des Ergebnisse vom Vorjahr:

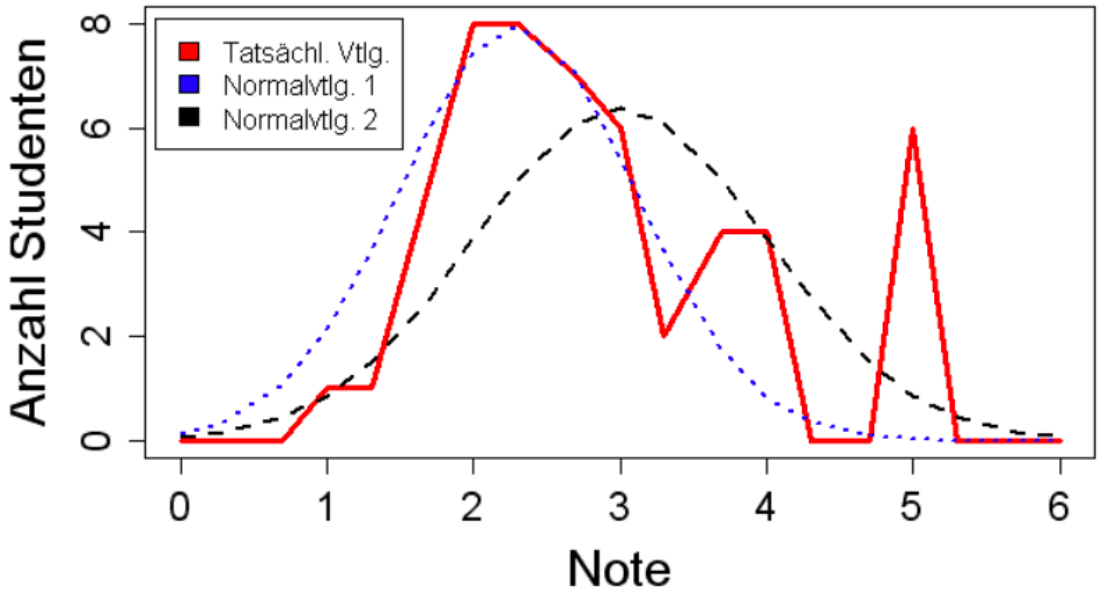
[Quelle: LMU, Datenbanksysteme II im SS 2016]



Relationale Anfragebearbeitung

- Selektivität
  - Parametrische Verteilung

[Quelle: LMU, Datenbanksysteme II im SS 2016]

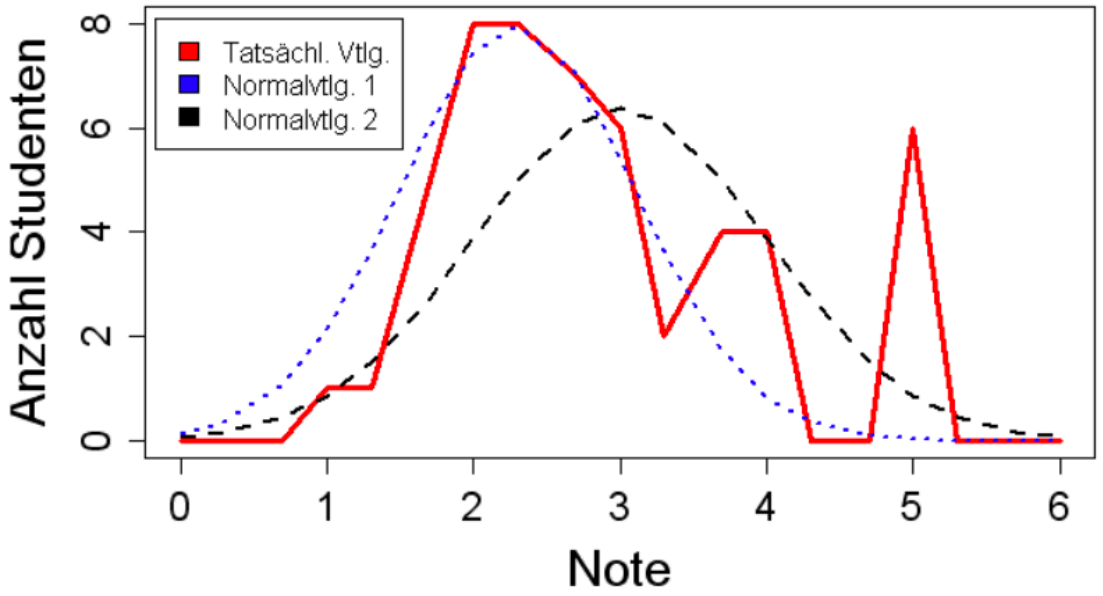


  - Probleme:
    - Art der Verteilungsfunktion (Normal, Exponentiell, …)
    - Wahl der Parameter
    - Abhängigkeiten der Attribute

Relationale Anfragebearbeitung

- Selektivität
  - Parametrische Verteilung

[Quelle: LMU, Datenbanksysteme II im SS 2016]

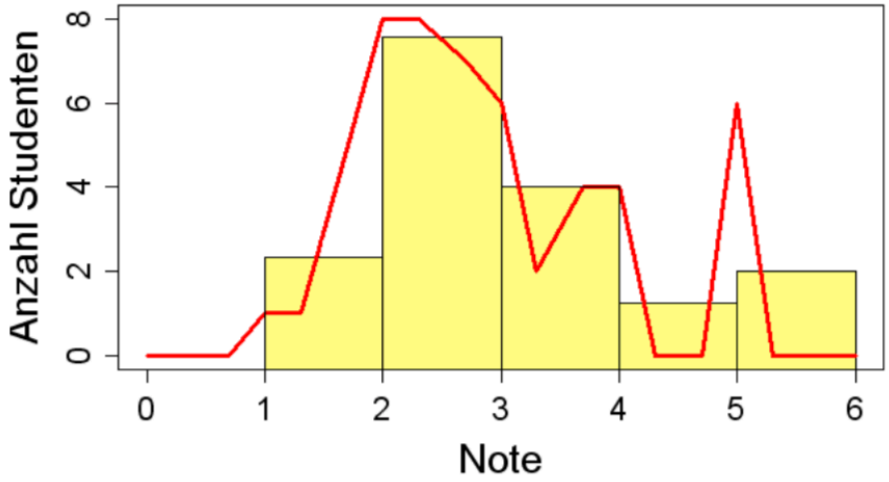


  - Probleme:
    - Art der Verteilungsfunktion (Normal, Exponentiell, …)
    - Wahl der Parameter
    - Abhängigkeiten der Attribute

Relationale Anfragebearbeitung

- Selektivität
  - Histogramme
    - Unterteile den Wertebereich des Attributs in Intervalle und zähle die Tupel, die in ein bestimmtes Intervall fallen.
    - Equi-Width-Histograms: Intervalle gleicher Breite.
    - Equi-Depth-Histograms: Unterteilung so, dass in jedem Intervall gleich viele Tupel sind.

[Quelle: LMU, Datenbanksysteme II im SS 2016]



=> Flexible Annäherung an die Verteilung.

Relationale Anfragebearbeitung

- Selektivität
  - Stichproben
    - Sehr einfaches Verfahren
    - Prinzip: Ziehe eine zufällige Menge von Tupeln aus einer Relation, und betrachte deren Verteilung als repräsentativ für die gesamte Relation.
    - Problem der Größe des Stichprobenumfangs n:
      - n zu klein: Wenig repräsentative Stichprobe
      - n zu gross: Ziehen der Stichprobe erfordert zu viele „teure“ Zugriffe auf den Hintergrundspeicher

Relationale Anfragebearbeitung

- Selektivität
  - Probleme bei Anfragen über mehrere Attribute (mehr-dimensionale Anfragen)
  - Sampling
    Problem: Genauigkeit abhängig von der Samplegröße
  - 1DHistogramme
    Problem: Annahme der Unabhängigkeit zwischen den Attributen
  - Multi-Dim-Histogramme
    Problem: Anzahl der Gridzellen steigt exponentiell mit d
  - Parametrische Methoden
    Problem: nur für max. 2-3 Attribute geeignet