Datenbanksysteme

Wintersemester 2015/16

Prof. Dr.-Ing. Sebastian Michel
TU Kaiserslautern

smichel@cs.uni-kl.de

Hash Join

ID	name
10	Jim
13	Joe
14	Sue
15	Pete
21	Dave
23	Anne

Angestellte

$\bowtie$

number	ID
100	23
110	10
120	15
130	23
140	23
150	13
160	15
170	21

Telefon

Wende Hashfunktion an auf Join-Attribut(e)
$\rightarrow$ Partitioniert Tupel in Buckets

Hash Join

ID	name
15	Pete
21	Dave

Angestellte$_0$

$\bowtie$

number	ID
120	15
160	15
170	21

Telefon$_0$

$=$

ID	name	number
15	Pete	120
15	Pete	160
21	Dave	170

Ergebnis$_0$

ID	name
10	Jim
13	Joe

Angestellte$_1$

$\bowtie$

number	ID
110	10
150	13

Telefon$_1$

$=$

ID	name	number
10	Jim	110
13	Joe	150

Ergebnis$_1$

ID	name
14	Sue
23	Anne

Angestellte$_2$

$\bowtie$

number	ID
100	23
130	23
140	23

Telefon$_2$

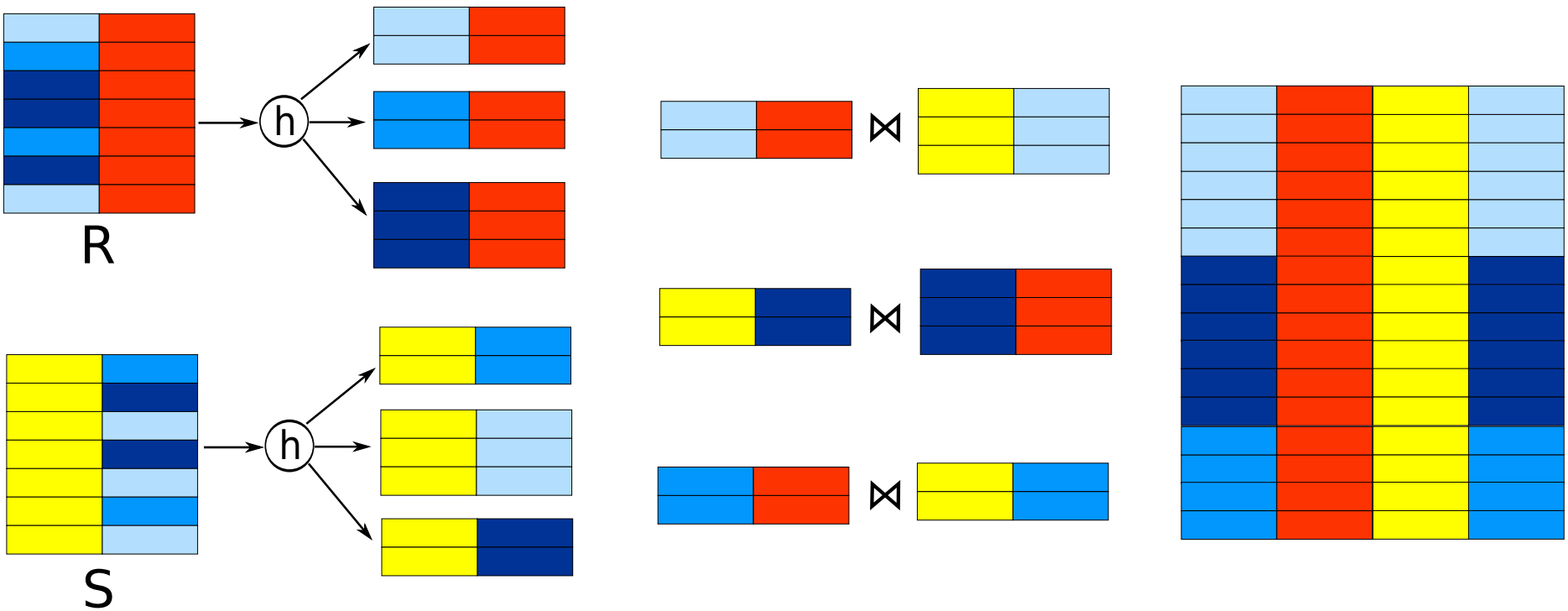
$=$

ID	name	number
23	Anne	100
23	Anne	130
23	Anne	140

Ergebnis$_2$

Hash Join: Idee

- Hashing jeder Relation basierend auf Join Attribut(en)
- Jeder Bucket muss klein genug für Hauptspeicher sein
- Tupel in korrespondierenden Buckets der beiden Relationen werden dann verbunden



Eingabe-Relationen

Ausgabe-Relation

Hash Join: Pseudocode

Aka. **Grace Hash Join** (in Literatur)

Partitioniere R in k Partitionen R_i

for each tuple $r \in R$ **do**
    lese r und füge es Puffer-Seite (Block) $h(r)$ hinzu

Partitioniere S in k Partitionen S_i

for each tuple $s \in S$ **do**
    lese s und füge es Puffer-Seite (Block) $h(s)$ hinzu

Hash Join: Pseudocode (2)

Sondierung (engl. Probing) Phase

for $l = 1, \ldots, k$ **do** $\{$

//Erzeuge im Hauptspeicher Hash-Tabelle für R_l mit Hash-Funktion h_2
for each tuple $r \in$ partition R_l **do**
lese r und füge es in Hash-Tabelle anhand $h_2(r)$ ein

//Lese S_l und teste nach passenden Tupeln aus R_l
for each tuple $s \in$ partition S_l **do** $\{$
lese s und teste Hash-Tabelle unter Verwendung von $h_2(s)$
für passende Tupel $r \in R$: $Res := Res \cup (r \circ s)$ $\}$
leere Hash-Tabelle und betrachte nächste Partition.

$\}$

Kosten und Anwendbarkeit der verschiedenen Join-Strategien

Nested Loop Join

- Kann für alle Arten von Joins benutzt werden
- Kann aber sehr teuer sein

Merge Join

- Eingaben müssen bereits sortiert vorliegen
- Oder für den Join extra sortiert werden
- Kann ggf. Indexe ausnutzen

Hash Join

- Gute Hashfunktionen sind elementar
- Performance am besten, wenn kleinere Relation direkt in Hauptspeicher passt

Was ist mit Joins über mehrere Attribute? Was mit Joins, die nicht auf Gleichheit (=) testen? Welche Implementierungen sind anwendbar?

Implementierung anderer Operatoren

Selektion

- Gibt es Indexe, die ausgenutzt werden können? Clustered oder nicht clustered? Ansonsten "full-table scan".
- Wie sieht das Prädikat aus? Disjunktion, Konjunktion?
- select ... from table where (x<100 **OR** name='Joe') mit Index auf name. Index benutzen?? Brauchen sowieso full scan wegen x<100.
- was ist wenn (x<100 **AND** name='Joe') ?

Duplikate Eliminieren und Aggregation

- Verschiedene Möglichkeiten, basierend auf Sortierung oder Hashing.
- Idee: Bringe Duplikate nah zueinander: Sortiert oder in gleichen Hash-Bucket.

Projektion

- Trivial. Bis auf den Fall wo Duplikate entfernt werden sollen "select distinct", dann siehe vorherigen Punkt.

Zusammenfassung Implementierung von Operatoren

- Verschiedene Implementierungen haben in der Regel verschiedene Ausführungskosten
- Diese hängen wiederum von den Eingaben und ggf. Reihenfolge der Eingaben ab.
- Ob Implementierung überhaupt benutzt werden kann ist auch abhängig von Eingabe, bzw. ob z.B. ein geeigneter Index existiert.
- Diese Kosten müssen geschätzt werden, vor der Ausführung (natürlich)

Kostenschätzung

(c) xkcd.com

Selektivitätsschätzung für kostenbasierte Optimierung

Selektivitätsschätzung: Idee

- Gegeben: Anfrage und Relationen
- Wie viele Tupel sind als Ergebnis zu erwarten?
- Wie viele Tupel fallen als Zwischenergebnisse an?

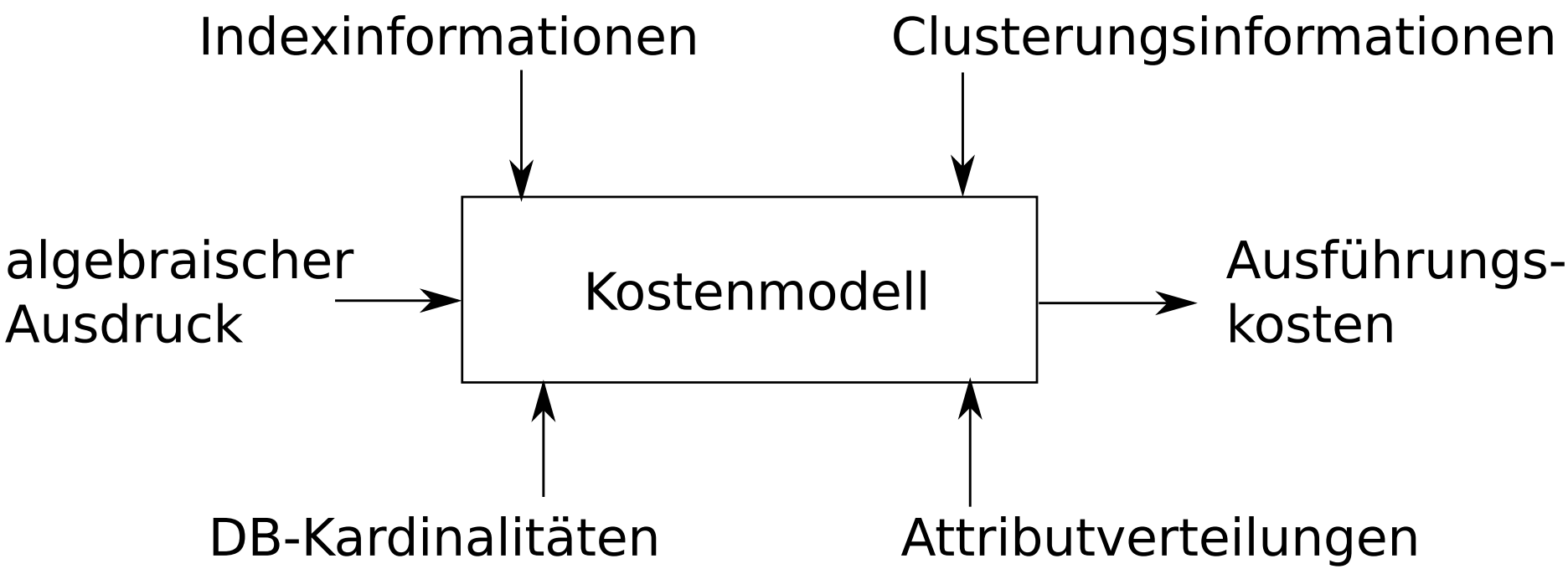
Selektivitätsschätzung: Werkzeuge

- Kenntnisse/Statistiken über zugrunde liegende Daten
- Generische Annahmen über Selektivität benutzter Prädikate
- Schätzung der Selektivität von o.g. Operatoren (z.B., Join)

Kostenbasierte Optimierung

- Generiere **alle** denkbaren Anfrageauswertungspläne:
  = Enumeration/Aufzählung möglicher Pläne
- Schätze deren Kosten ab:
  - Kostenmodell
  - Statistiken
  - Histogramme
  - Kalibrierung gemäß verwendetem Rechner
  - Abhängig vom verfügbaren Speicher
  - Aufwands-Kostenmodell
    - Durchsatz-maximierend
    - versus Antwortzeit-minimierend
- Behalte den Plan mit den geringsten geschätzten Kosten

Kostenmodelle



Grundlagen

- $T(R)$ ist die **Anzahl der Tupel** in Relation R
- $V(R,A)$ ist die **Anzahl der verschiedenen Attributsausprägungen** für Attribut A.
- Dementsprechend für mehrere Attribute: $V(R,[A_1, A_2, ..., A_n])$

Aufgabe der Kostenschätzung

- Gegeben eine Anfrage, wie groß ist das Ergebnis und wie viele Tupel fallen als Zwischenergebnisse an?
- z.B. wie viele Tupel sind in $\sigma_{A=13434}(R)$

Schätzungen für Selektion

Gegeben eine Selektion $S = \sigma_{A=c}(R)$.
Wie viele Tupel sind in S?
Schätzung:

$$T(S) = \frac{T(R)}{V(R,A)}$$

Gilt falls die Werte für A zufällig aus allen möglichen Werten gezogen wurden.

Ungleichheit

Was ist bei $S = \sigma_{A<c}(R)$?
Im Allgemeinen, ohne weitere Annahmen: $T(S) = T(R)/2$

Aber, Intuition: man wählt mit solchen Bedingungen meist weniger Tupel aus.
Besser: $T(S) = T(R)/3$

Schätzung für "not-equals"

Gegeben eine Selektion $S = \sigma_{A \neq c}(R)$.
Wie viele Tupel sind in S?

Einfache Schätzung

- "Mehr oder weniger" alle Tupel erfüllen die Bedingung (naja, bis auf ein paar, aber egal)
- Also: $T(S) = T(R)$

Leicht verbessert

- Die Tupel mit $A = c$ erfüllen die Bedingung nicht.
- Also: $T(S) = T(R)\frac{V(R,A) - 1}{V(R,A)}$

Was passiert mit Kaskaden von Selektionen?
Selektivität ist Produkt der einzelnen Selektivitäten!

Selektion mit ODER-Bedingungen

Gegeben:

$$S = \sigma_{C_1 \vee C_2}(R)$$

Annahme: die Bedingungen werden nie gemeinsam erfüllt

- Also: **entweder** gilt C_1 **oder** C_2
- Schätzung: Summe der beiden einzelnen Selektivitäten.
- Beobachtung: Überschätzt oft. Was kann dann passieren?

Besser: Annahme die Bedingungen sind unabhängig

- Annahme: m_1 Tupel erfüllen C_1 und m_2 Tupel erfüllen C_2
- Dann $T(S) = T(R) * (1 - (1 - \frac{m_1}{T(R)})(1 - \frac{m_2}{T(R)}))$

Selektion mit ODER-Bedingungen: Erläuterung

Wieso $T(S) = T(R) * (1 - (1 - \frac{m_1}{T(R)})(1 - \frac{m_2}{T(R)}))$?

$1 - \frac{m_1}{T(R)}$ ist der Anteil der Tupel die C_1 **nicht** erfüllen.

analog für C_2. Dann ist

$(1 - \frac{m_1}{T(R)})(1 - \frac{m_2}{T(R)})$ ist der Anteil der Tupel die C_1 **und** C_2 **nicht erfüllen**.

$(1 - ***)$ der Anteil der Tupel die C_1 **oder** C_2 erfüllen.

Klar: Multipliziert mit $T(R)$ ergibt $T(S)$

Andere Schätzer

Projektion

- Ändert die Kardinalität nicht
- Sehr wohl aber die Größe der Tupel!

Kartesisches Produkt

- Einfach: Produkt der beteiligten Kardinalitäten

Jetzt wird es **unklar was zu tun ist**:

Union

- Obere Schranke: Summe der beiden Kardinalitäten
- Untere Schranke: Größere der beiden Kardinalitäten
- Empfehlung aus Literatur: Irgendwas dazwischen, z.B. größere plus die halbe kleinere Kardinalität

Andere Schätzer

Schnitt

- Obere Schranke: Kleinste der beiden Kardinalitäten
- Untere Schranke: 0
- Empfehlung aus Literatur: Durchschnitt dieser beiden Werte.

Differenz $R - S$

- Obere Schranke: $T(R)$
- Untere Schranke: $T(R) - T(S)$
- Empfehlung aus Literatur: $T(R) - \frac{T(S)}{2}$

Schätzung für Joins: Natürlicher Join

Zwei Relationen: $R = (X,Y)$ und $S = (Y,Z)$
Annahme: Y ist ein einfaches Attribut, keine Menge von Attributen. X und Z dürfen Mengen sein.

Was können wir dann sagen? Leider nur sehr wenig, da z.B. folgende Fälle auftreten können:

- **Die beiden Relationen haben disjunkte Mengen für Y-Werte.** Also ist der Join leer, d.h. $T(R \bowtie S) = 0$.
- **Y ist Schlüssel von S und Fremdschlüssel in R.** Dann findet jedes Tupel in R einen Joinpartner in S, also $T(R \bowtie S) = T(R)$
- **Fast alle Tupel aus R und S haben den gleichen Wert für Y,** also $T(R \bowtie S) = T(R) * T(S)$

Schätzung für Joins: Natürlicher Join: Annahmen

Häufig auftretende Fälle. Weitere Annahmen:

- Es gibt Y-Werte $y_1, y_2, y_3, ...$
- Relationen benutzen diese Werte in dieser Reihenfolge.
- Dann: Falls $V(R,Y) \leq V(S,Y)$ so gilt auch, dass jeder Y-Wert in R auch ein Y-Wert in S ist.

Erhaltung der Wertemengen

- Falls A kein Join-Attribut ist, gilt

$$V(R \bowtie S, A) = V(R,A)$$

- D.h. Attribut A verliert durch den Join keine möglichen Werte.

Schätzung für Joins: Natürlicher Join

Gesucht: Größe des Joins $R(X,Y) \bowtie S(Y,Z)$

- Gegeben, zwei Tupel: $r \in R$ und $s \in S$
- Was ist die Wahrscheinlichkeit, dass $r.Y = s.Y$?
- Annahme: $V(R,Y) \geq V(S,Y)$, also gibt es den Y-Wert von s in R.
- Also: Wahrscheinlichkeit, dass $r.Y = s.Y$ ist $1/V(R,Y)$
- Umgekehrt: falls $V(R,Y) < V(S,Y)$ analog.
- **Insgesamt:** Übereinstimmung in Y mit Wahrscheinlichkeit $1/max(V(R,Y), V(S,Y))$

$$T(R \bowtie S) = \frac{T(R) * T(S)}{max(V(R,Y), V(S,Y))}$$

"Essentially, all models are wrong, but some are useful."
(George E. P. Box)

Wie können Größen abgeschätzt werden?

Wichtig: Statistiken über Werte von Attributen, Größen von Relationen
Aber wie kann man diese berechnen und repräsentieren?

Durch Scannen der gesamten Mengen

- Kann hin und wieder berechnet werden, natürlich besser nicht zur Anfragezeit.
- Moderne DMBS haben bestimmte Befehle dafür.

Ausprägungen für Attribut A

- Falls $V(R,A)$ nicht zu groß ist: speichere einen Zähler für jeden Wert von A.
- Sonst: Gruppierung. Evtl: Exaktes Speichern für eine Teilmenge der Werte (z.B. der häufigsten).

$\Rightarrow$ Histogramme oder parametrisierte Verteilungen!

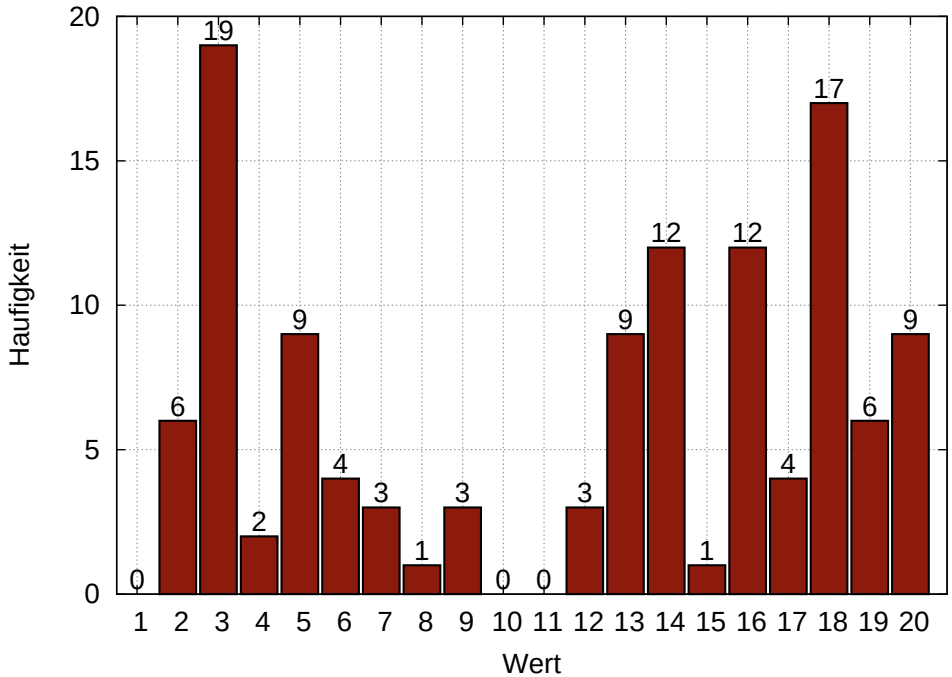
Beispieldaten

{2, 5, 3, 20, 18, 7, 16, 18, 18, 2, 2, 17, 14, 3, 20, 3, 16, 6, 7, 3, 16, 16, 15, 5, 20, 13, 16, 20, 12, 14, 13, 3, 14, 18, 14, 14, 16, 18, 19, 3, 5, 2, 5, 14, 20, 17, 3, 17, 16, 3, 2, 19, 3, 9, 13, 4, 3, 16, 14, 13, 13, 16, 20, 14, 4, 2, 3, 18, 7, 3, 5, 3, 6, 9, 18, 3, 16, 18, 20, 18, 5, 18, 5, 18, 13, 14, 19, 13, 14, 3, 14, 18, 14, 18, 18, 16, 19, 5, 3, 17, 18, 3, 19, 3, 20, 9, 16, 12, 20, 8, 12, 13, 13, 19, 18, 6, 3, 5, 18, 6}

Verteilung der Daten

Wert $\rightarrow$ Häufigkeit.

Dargestellt im "Histogramm-Stil":



Wir sehen: Es liegen 20 Werte im Wertebereich. Es gibt 120 Datenpunkte.

Alternativ: nicht die absolute Häufigkeit sondern normalisiert in [0,1], also "Wahrscheinlichkeit" bei zufälligem Zugriff auf Daten einen bestimmten Datenpunkt zu treffen.

Zusammenfassen/Beschreiben großer Datenmengen

Beschreibung durch parametrisierte Verteilung (Funktion)

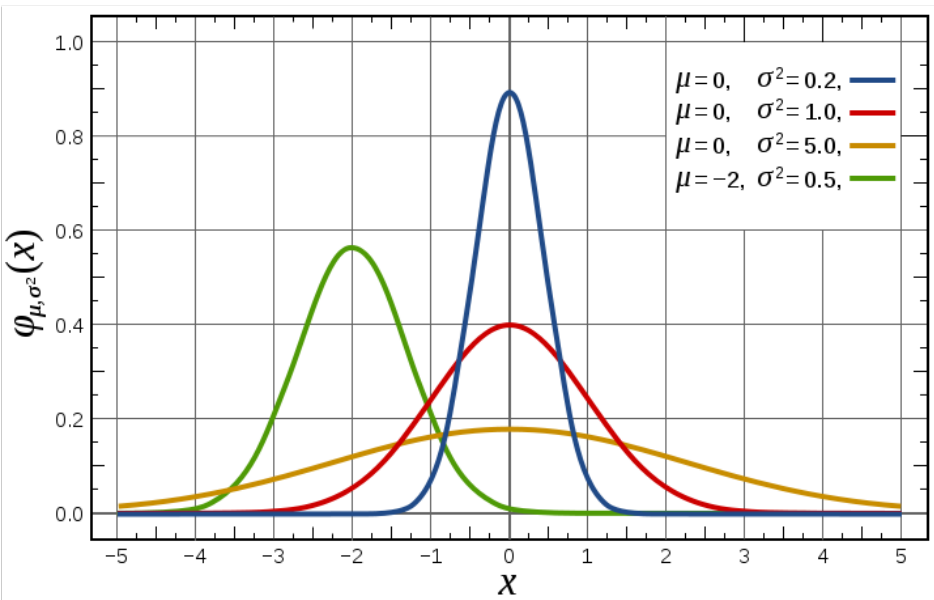
- Z.B. die Daten folgen einer Normalverteilung mit Erwartungswert μ und Varianz σ^2.

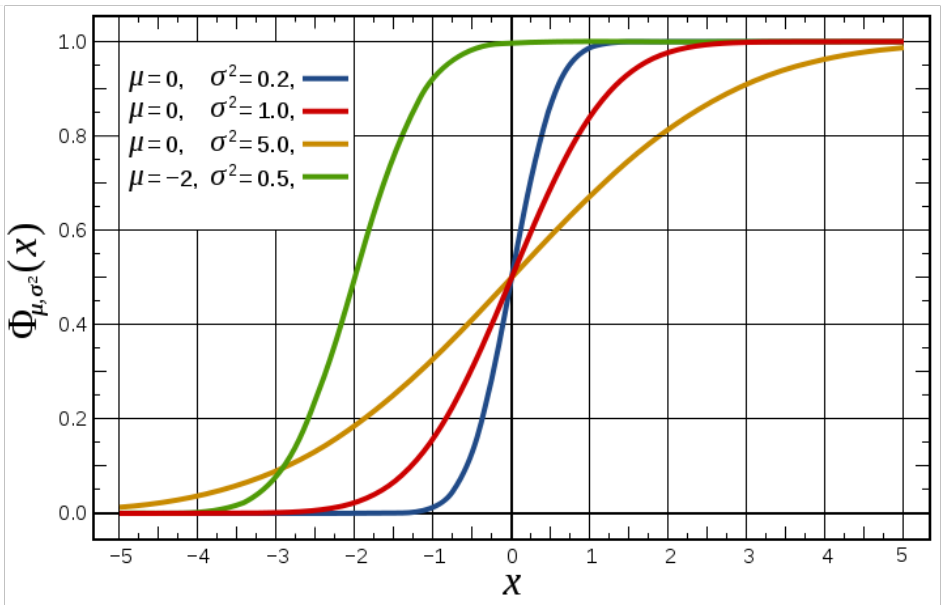
Zusammenfassen von Werten in Zellen (aka. Eimer oder Buckets)

- Wie viele Werte fallen in $[0,5[$, wie viele Werte fallen in $[5,10[$, etc.

Parametrisierte Verteilungen

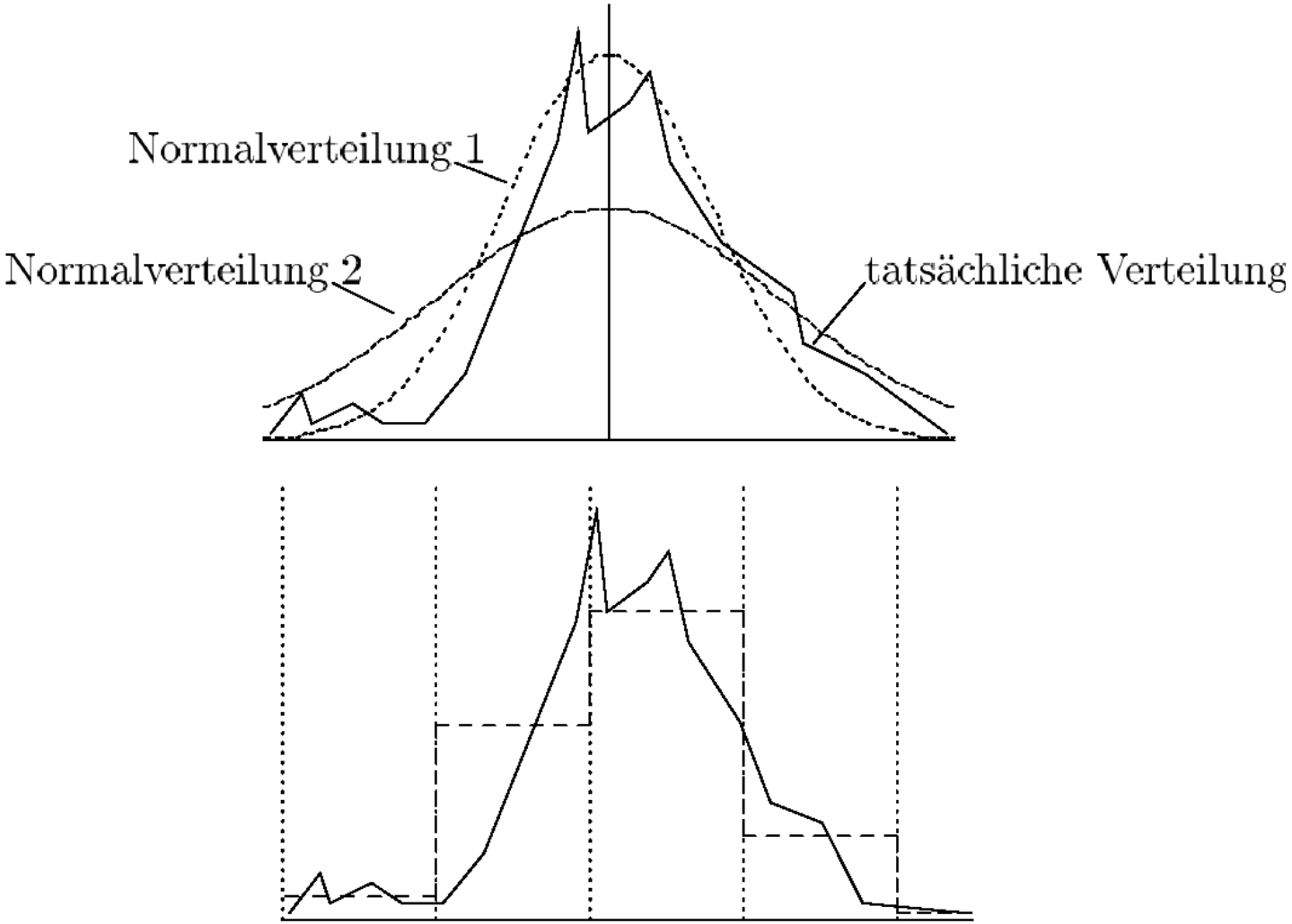
Dichtefunktion und Verteilungsfunktion der Normalverteilung:





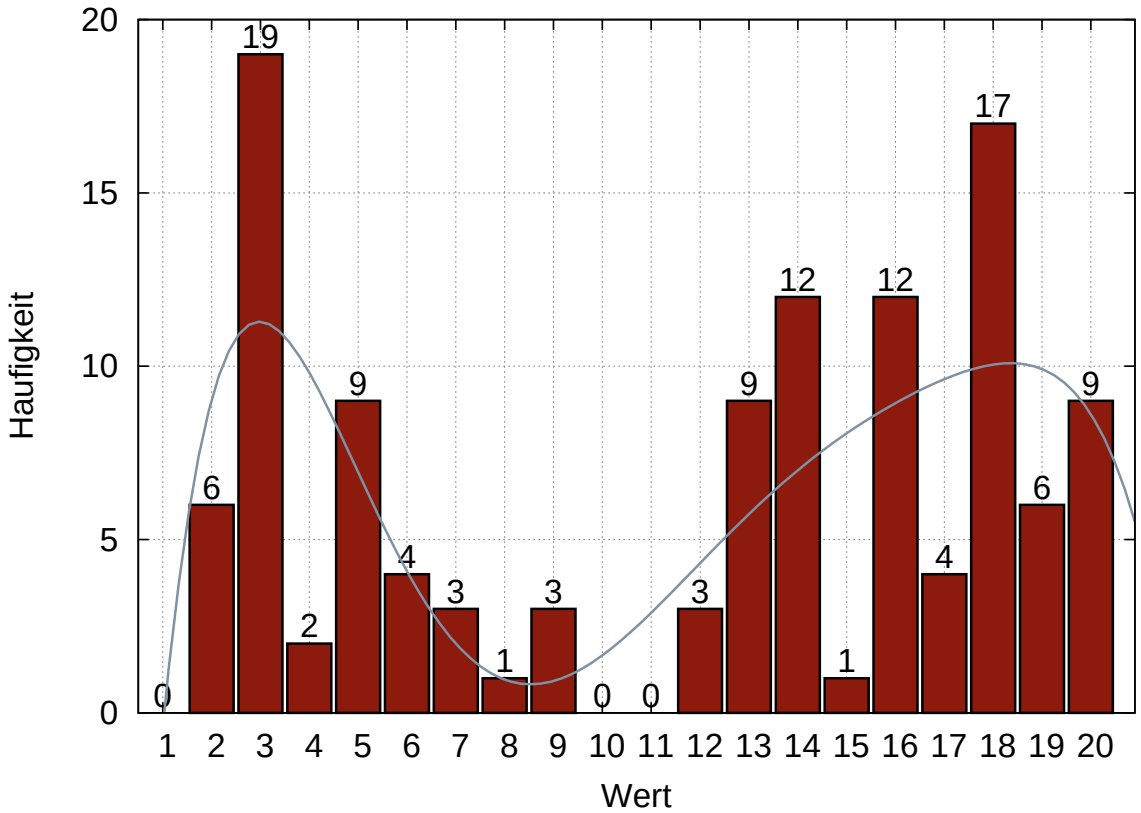
Abbildungen aus Wikipedia

Parametrisierte Verteilungen und Histogramme



Oft ist es schwierig eine tatsächliche Verteilung durch eine parametrisierte Verteilung auszudrücken. Histogramme sind flexibler.

Parametrisierte Verteilungen



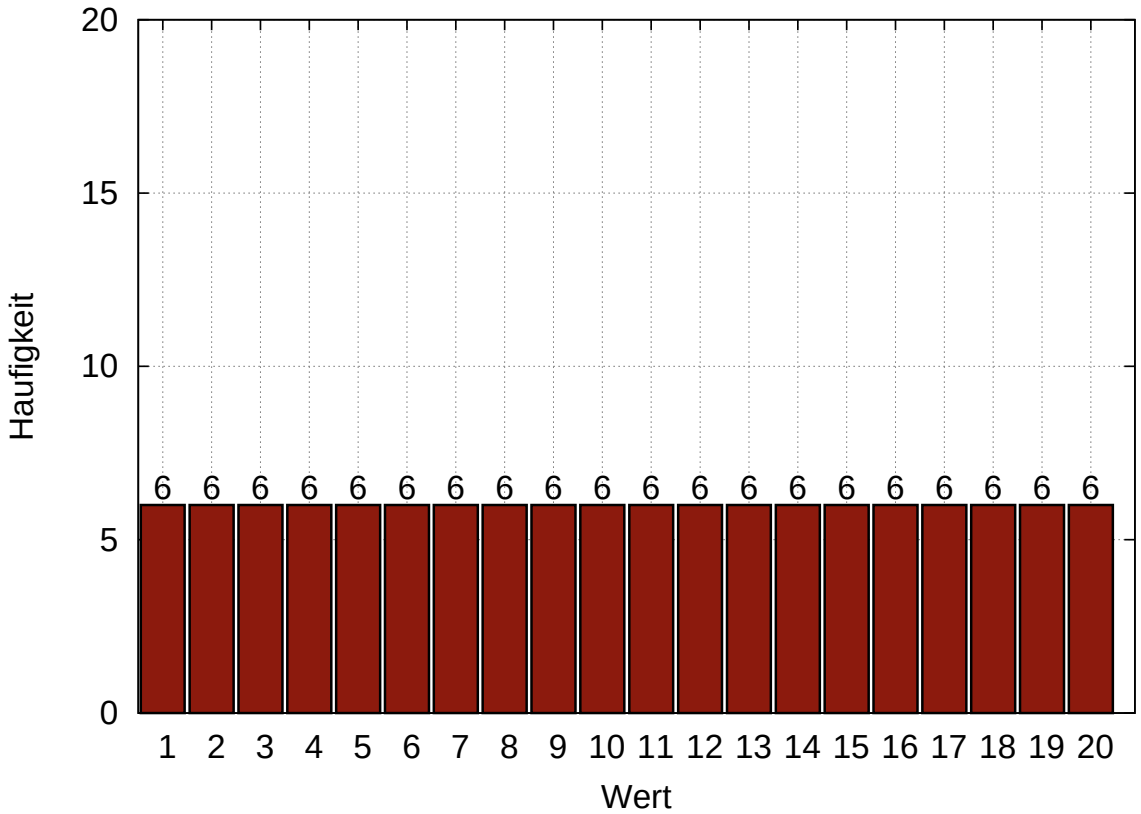
Hier, fit durch Polynom 6. Grades (mit Hilfe des Tools xmgrace):

$$f(x) := -21.884 + 29.533 * x - 9.2268 * x^2 + 1.2529 * x^3 - 0.085019 * x^4$$

$$+0.0028667 * x^5 - 3.8485 * 10^{-5} * x^6$$

Annahme Gleichverteilung

Es liegen 20 Werte im Wertebereich. Es gibt 120 Datenpunkte. Jeder Wert kommt, unter Annahme einer Gleichverteilung, also 6 Mal vor.



Oft nicht sehr realistische Darstellung (aber äußerst kompakt).

Histogramme

Histogram teilt den Wertebereich in Zellen oder Intervalle (Englisch: buckets oder cells). Für jede dieser Zellen wird die Anzahl der Elemente gespeichert, die in die Zelle fallen.

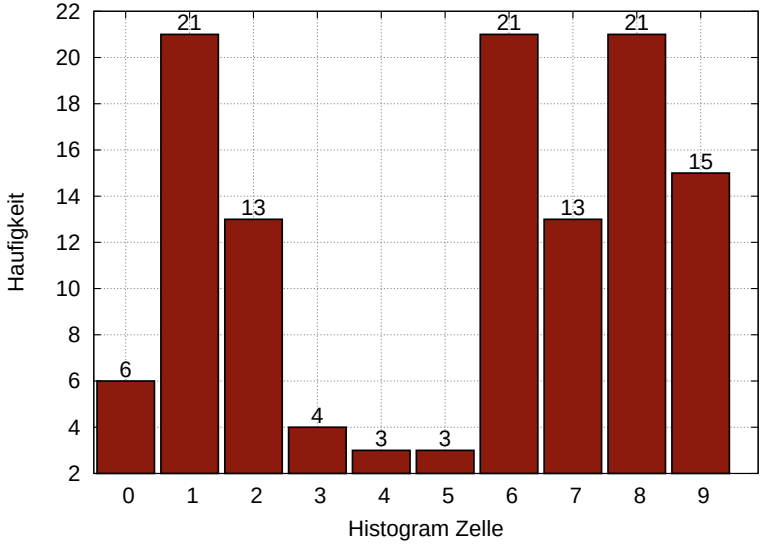
Equi-Width-Histogramme:

- Zellen habe immer die gleiche Breite im Wertebereich

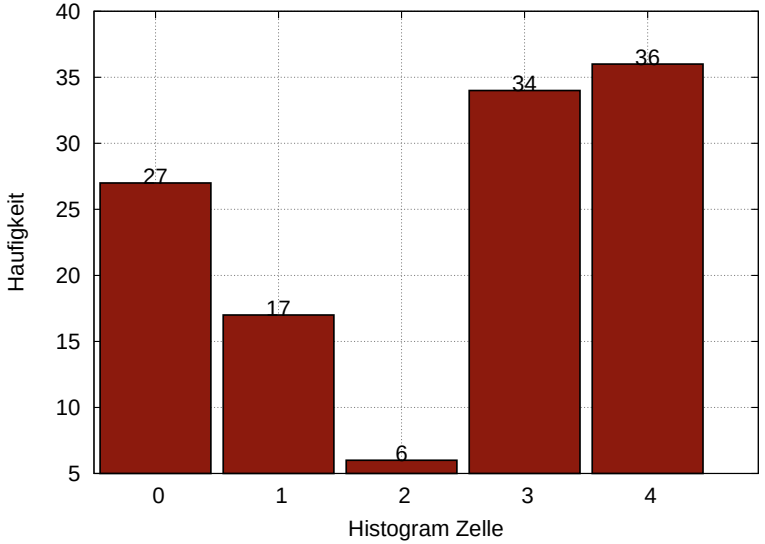
Equi-Depth (oder equi-height genannt) Histogramme:

- Zellen haben die gleiche "Höhe"
- Um dies zu erreichen: Breite der Zellen wird angepasst

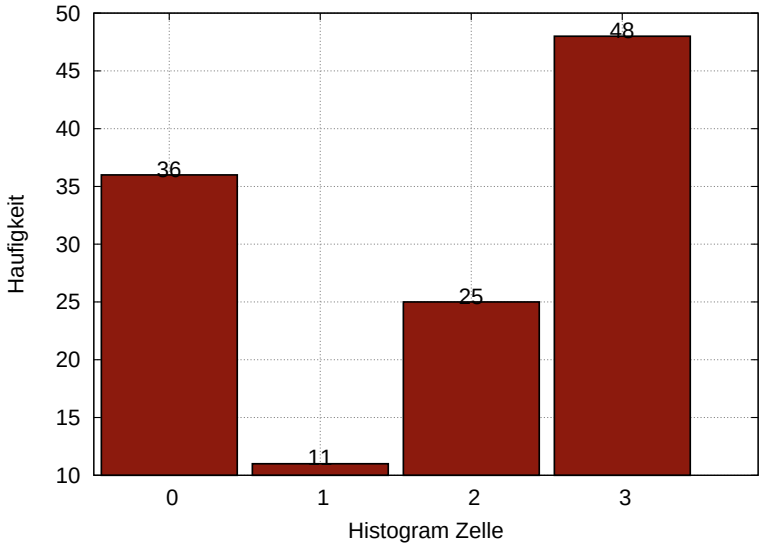
Equi-Width-Histogramme (Beispiele an o.g. Daten)



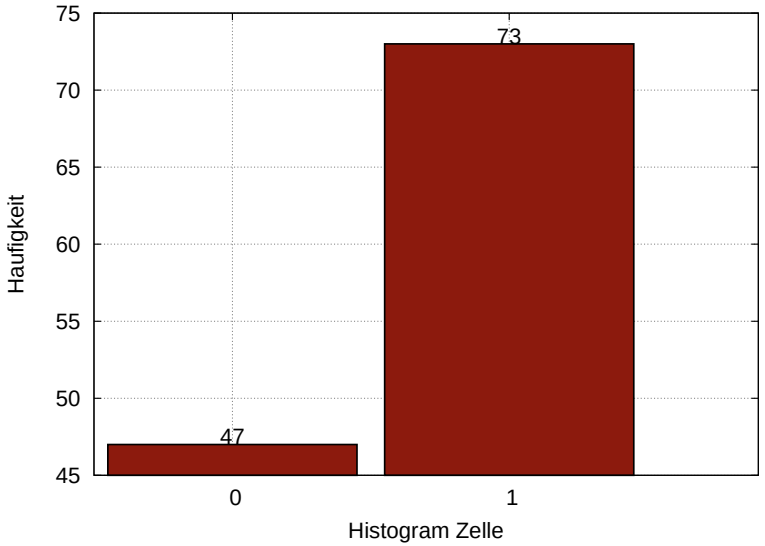
(a) Width = 2



(b) Width = 4

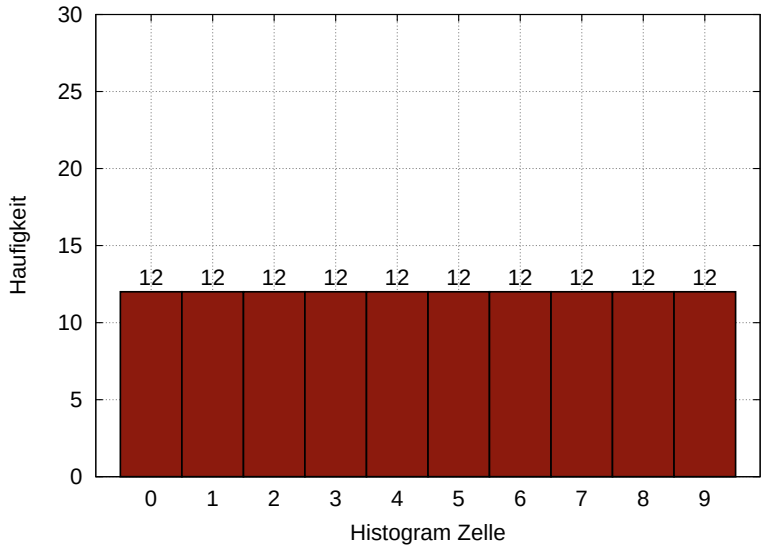


(a) Width = 5

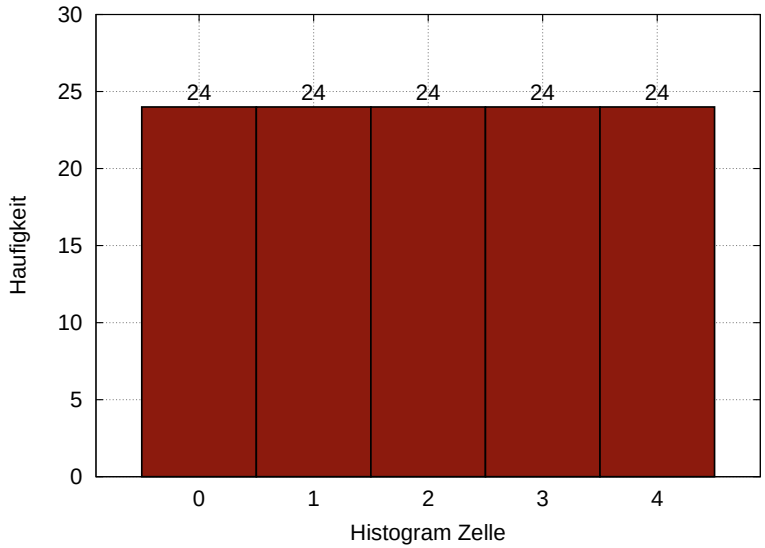


(b) Width = 10

Equi-Depth-Histogramme (Beispiele an o.g. Daten)



(a) Depth = 10%



(b) Depth = 20%

Grenzen der Zellen (je rechter Punkt, d.h. Ende)

- (a) 3,3,6,12,14,16,17,18,19
- (b) 3,12,15,18,20

Punktanfragen und Bereichsanfragen auf Histogrammen

Punktanfragen

- Wie viele Tupel haben den Wert $A = 10$?
- Nachschauen: Zelle in die der Wert 10 fällt.
- Annahme hier: Gleichverteilung innerhalb der Zelle.
- Resultat: Anzahl der Tupel geteilt durch Breite der Zelle.
- Bzw. wenn bereits "normalisiert" dann nur Wert der Zelle.

Bereichsanfragen

- Wie viele Tupel haben einen Wert $A > 10$?
- Nachschauen: In welche Zelle fällt der Wert 10? (Achtung insbesondere bei Equi-Depth Histogrammen)
- Resultat: Summe der Größen der darüber liegenden Zellen und anteilsmäßig diese Zelle (wie oben).
- Oder: Histogramm für kumulative Verteilung betrachten