Datenbanksysteme 2013

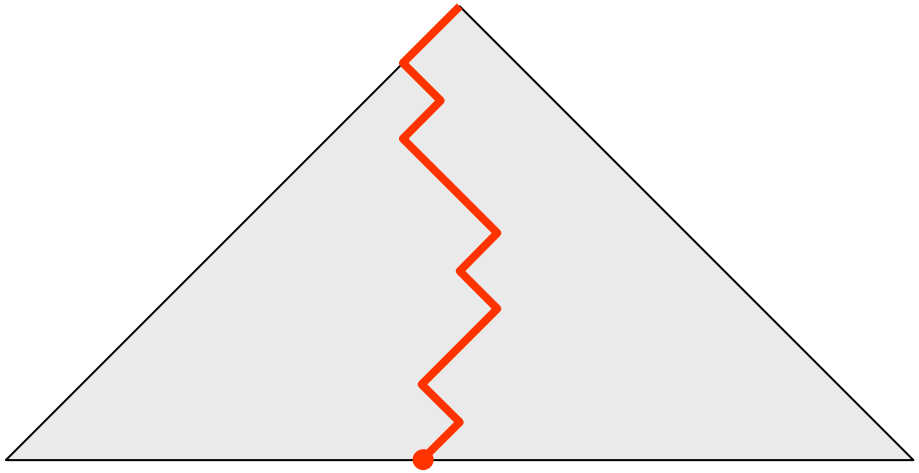
Kapitel 5:
Mehrdimensionale Suchstrukturen
Vorlesung vom 23.04.2013

Oliver Vornberger

Institut für Informatik
Universität Osnabrück

Suche nach Primärschlüssel

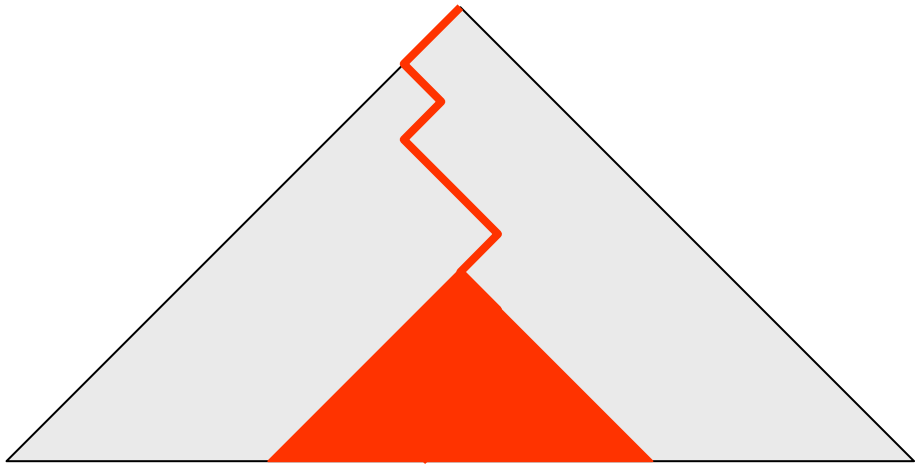
```
select name from studenten
where matrnr = 4711
```



Aufwand : $\log_k (c \cdot n)$

Suche nach Sekundärschlüssel

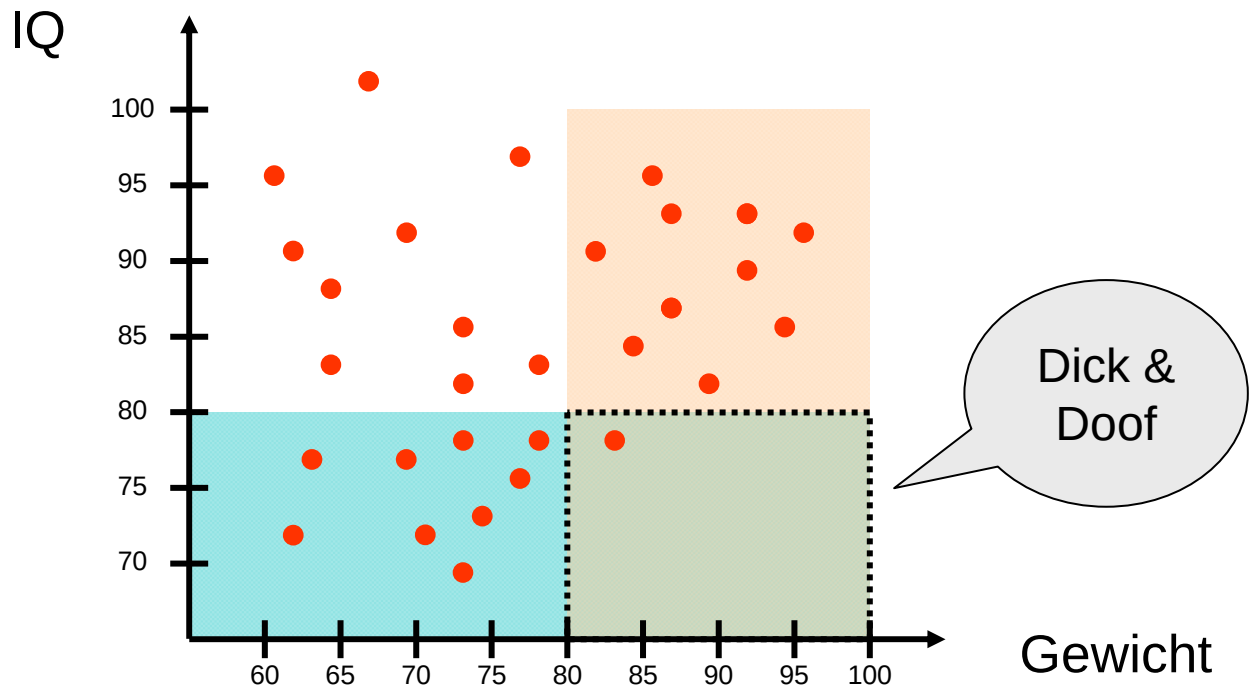
```
select name from studenten
where groesse > 170
and   groesse < 180
```



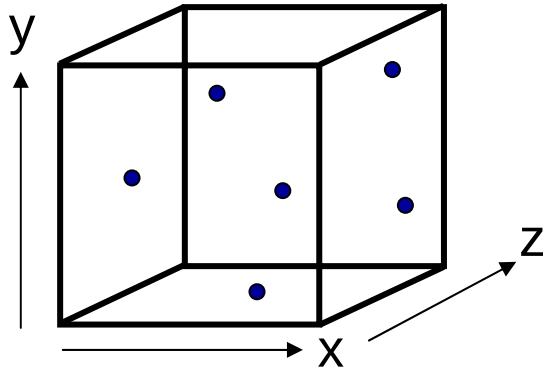
Aufwand bei m Treffern von n Records: $\log_k (c \cdot n) + c \cdot m$

Suche nach 2 Sekundärschlüsseln

```
select name from studenten
where gewicht > 80
and   iq      < 80
```



k-d-Baum



zur Verwaltung von mehrdimensionalen Datenpunkten

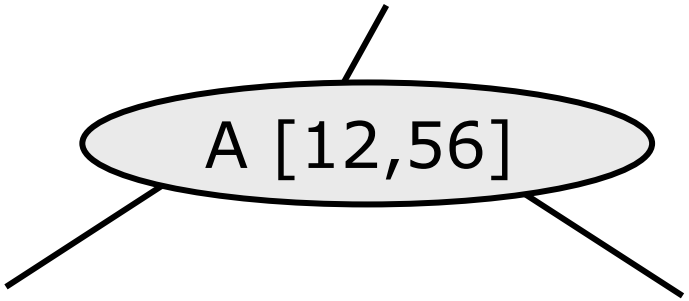
Verallgemeinerung des binären Suchbaums
mit k-dimensionalem Sortierschlüssel

Homogene Variante: Baumknoten = Datenrecord + Zeiger

Inhomogene Variante: Baumknoten = Schlüssel + Zeiger
Blätter zeigen auf Datenrecord

Ebene i modulo k diskriminiert bzgl. Dimension i

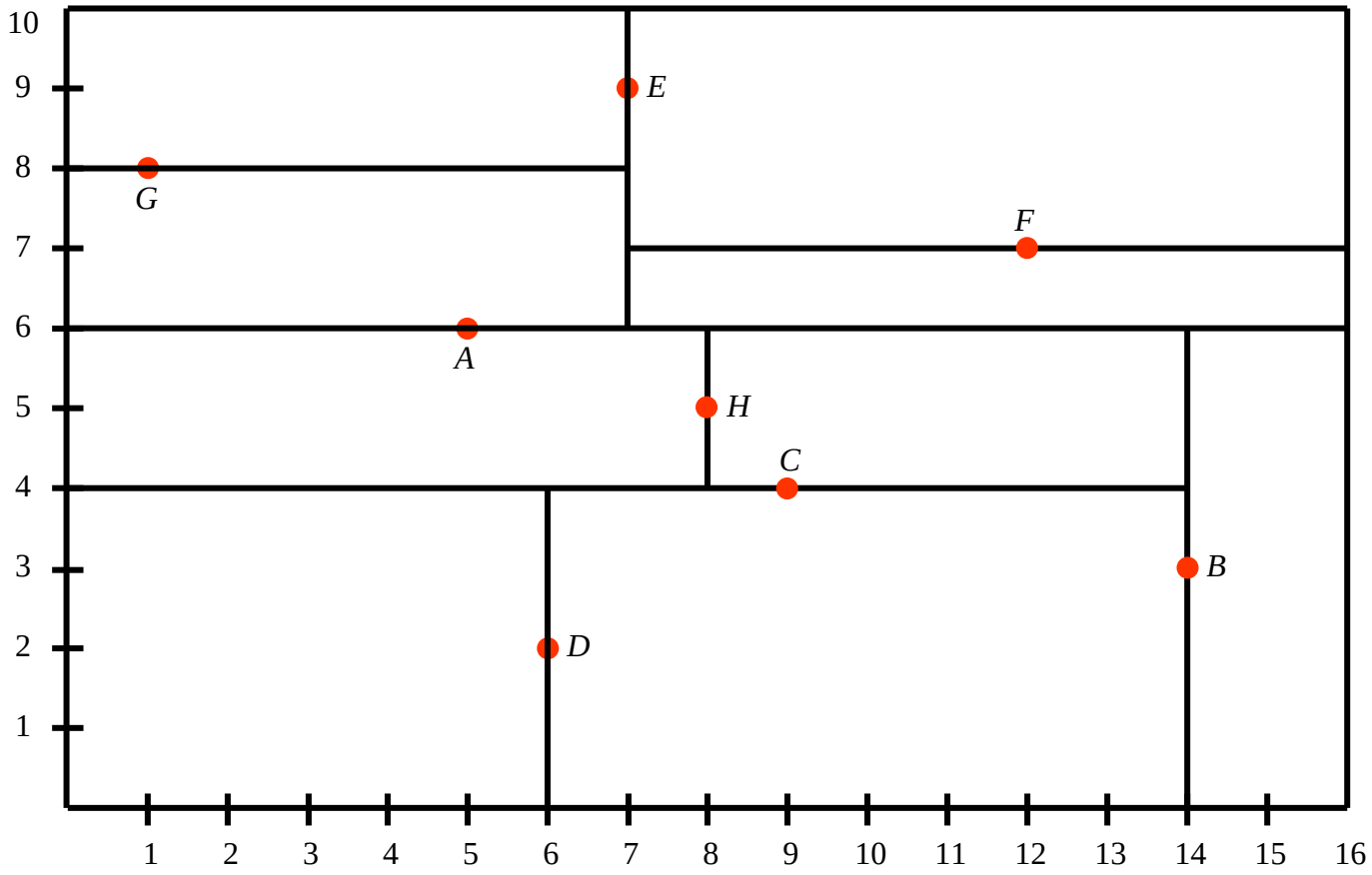
2-d-Baum



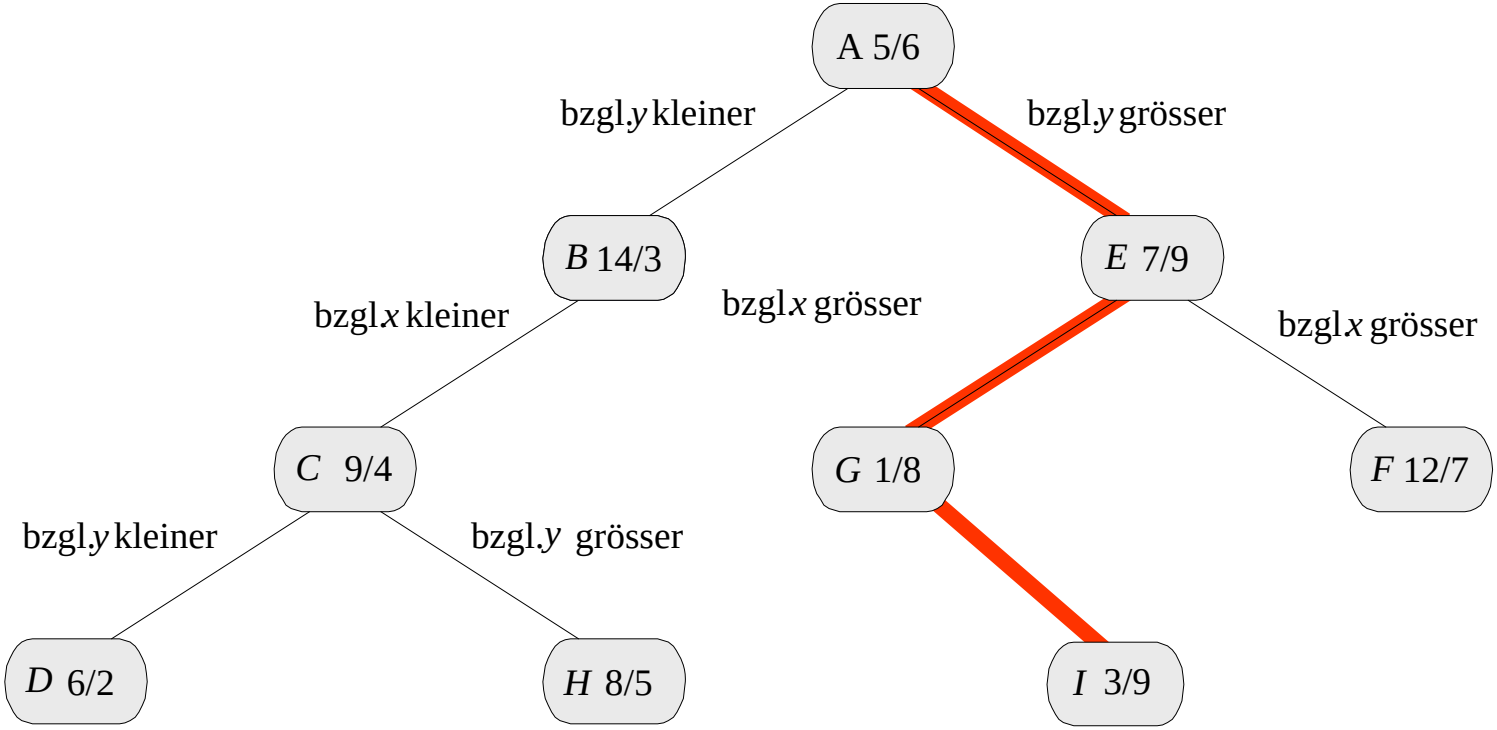
Im 2-dimensionalen Fall gilt für Knoten mit Schlüssel [x/y]:

	im linken Sohn	im rechten Sohn
ungerade Ebene	alle Schlüssel $\leq$ x	alle Schlüssel > x
gerade Ebene	alle Schlüssel $\leq$ y	alle Schlüssel > y

2-d-Baum: Beispiel

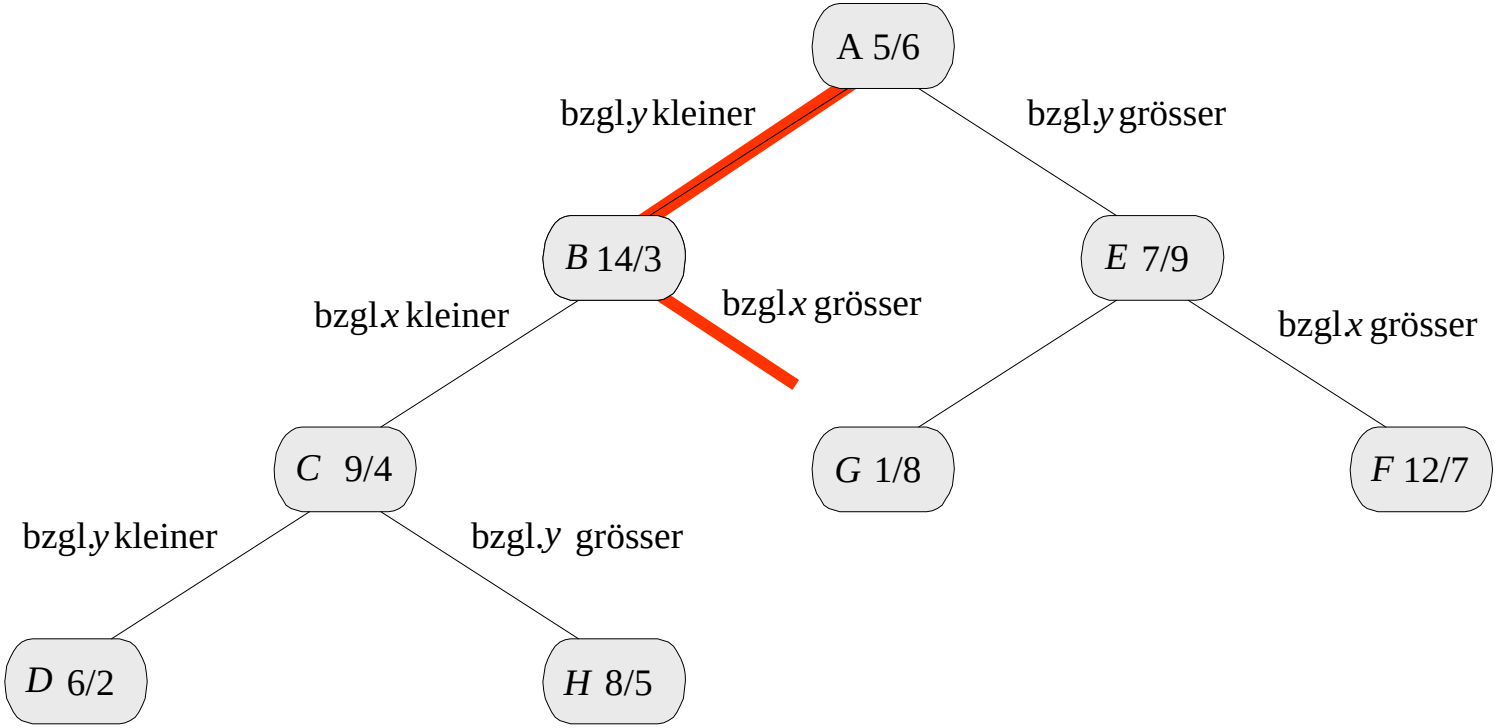


2-d-Baum: Beispiel



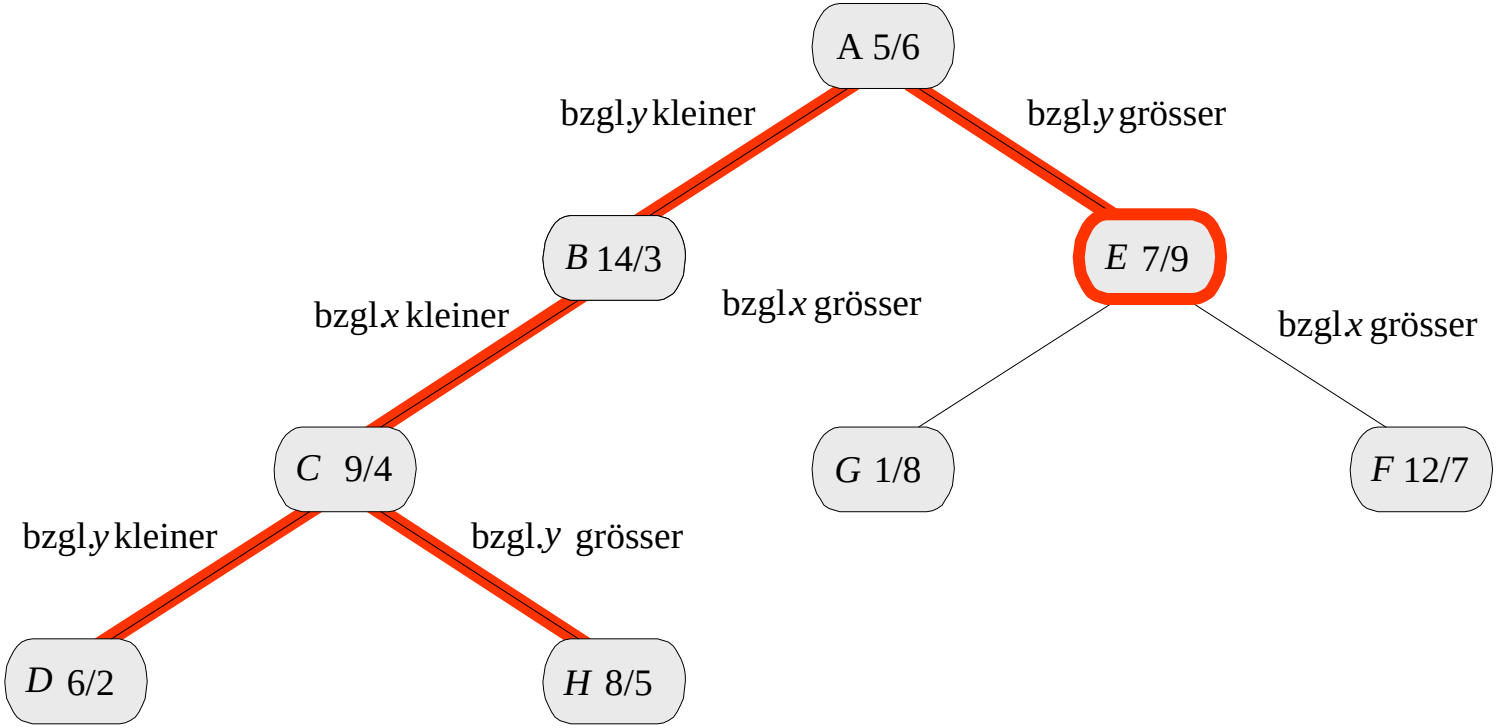
Insert z.B. füge Record [3/9] ein

2-d-Baum: Beispiel



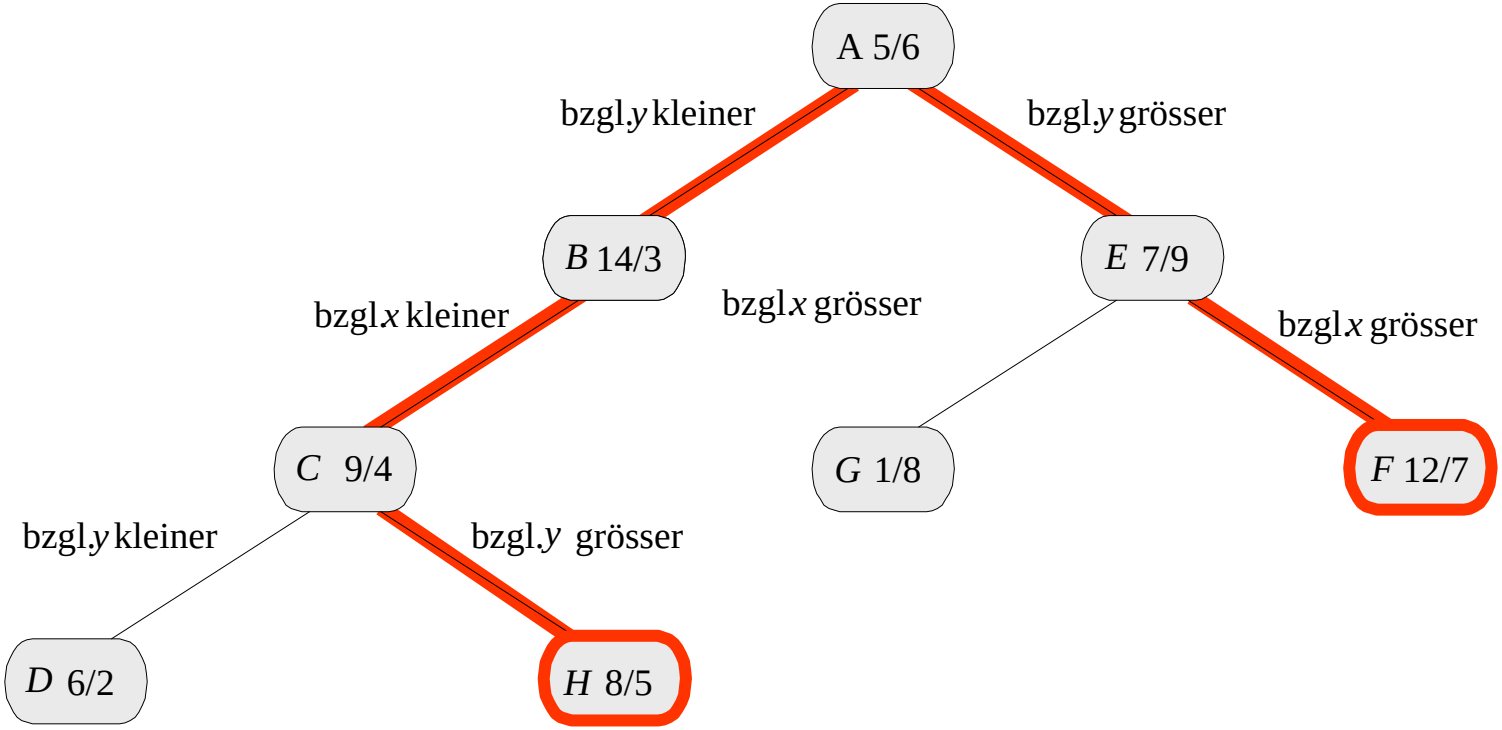
Exact Match z.B. finde Record [15/5]

2-d-Baum: Beispiel



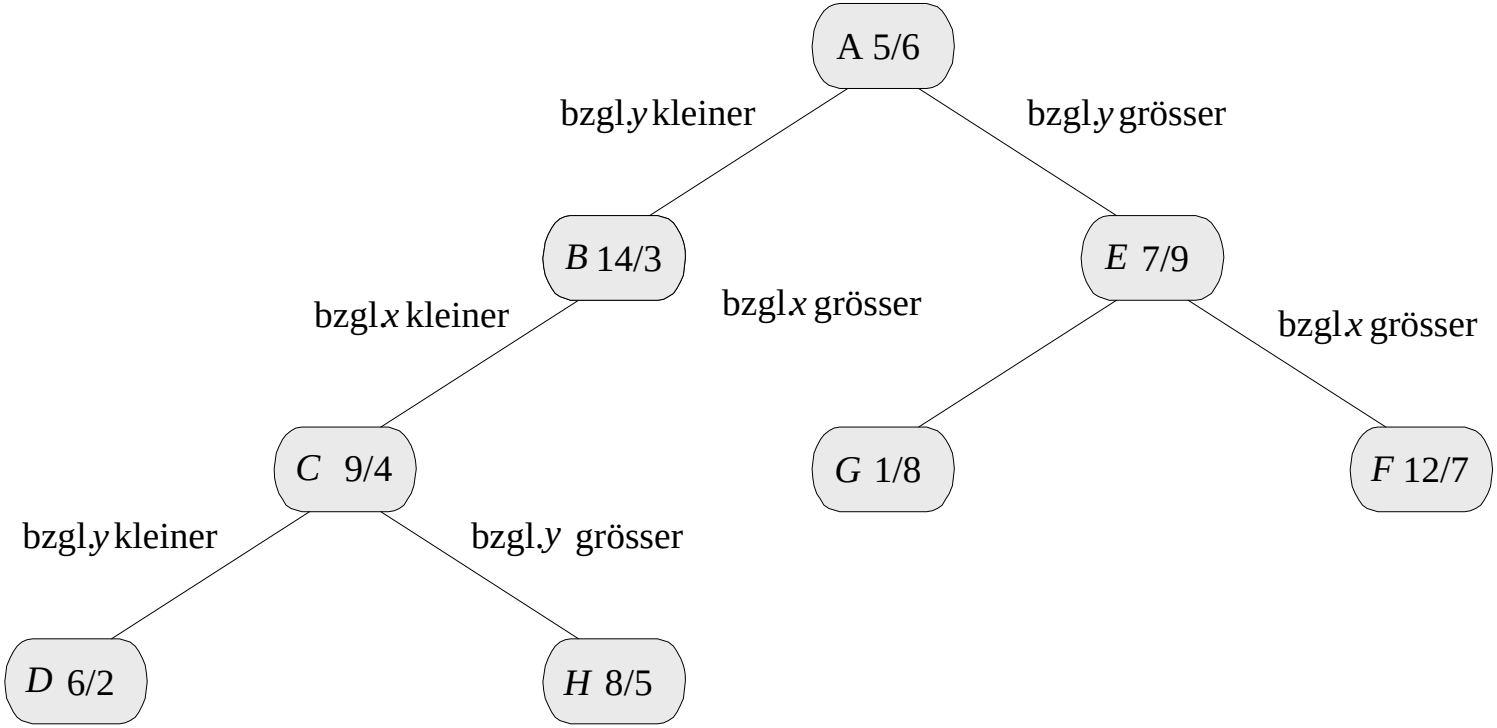
Partial Match z.B. finde alle Records [x/y] mit x=7

2-d-Baum: Beispiel



Range-Query z.B. finde Records [x/y] mit $8 \leq x \leq 13$, $5 \leq y \leq 8$

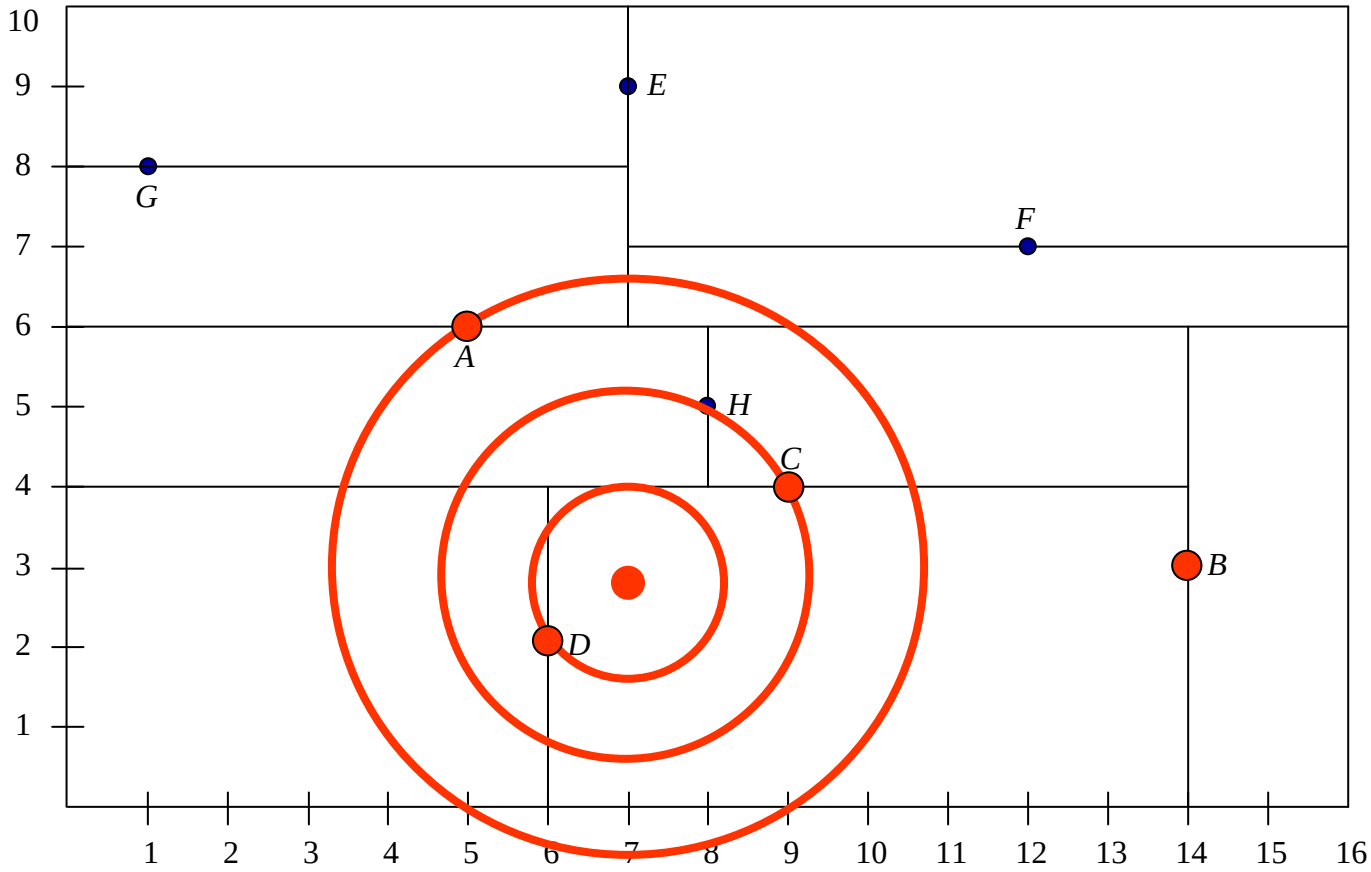
2-d-Baum: Beispiel



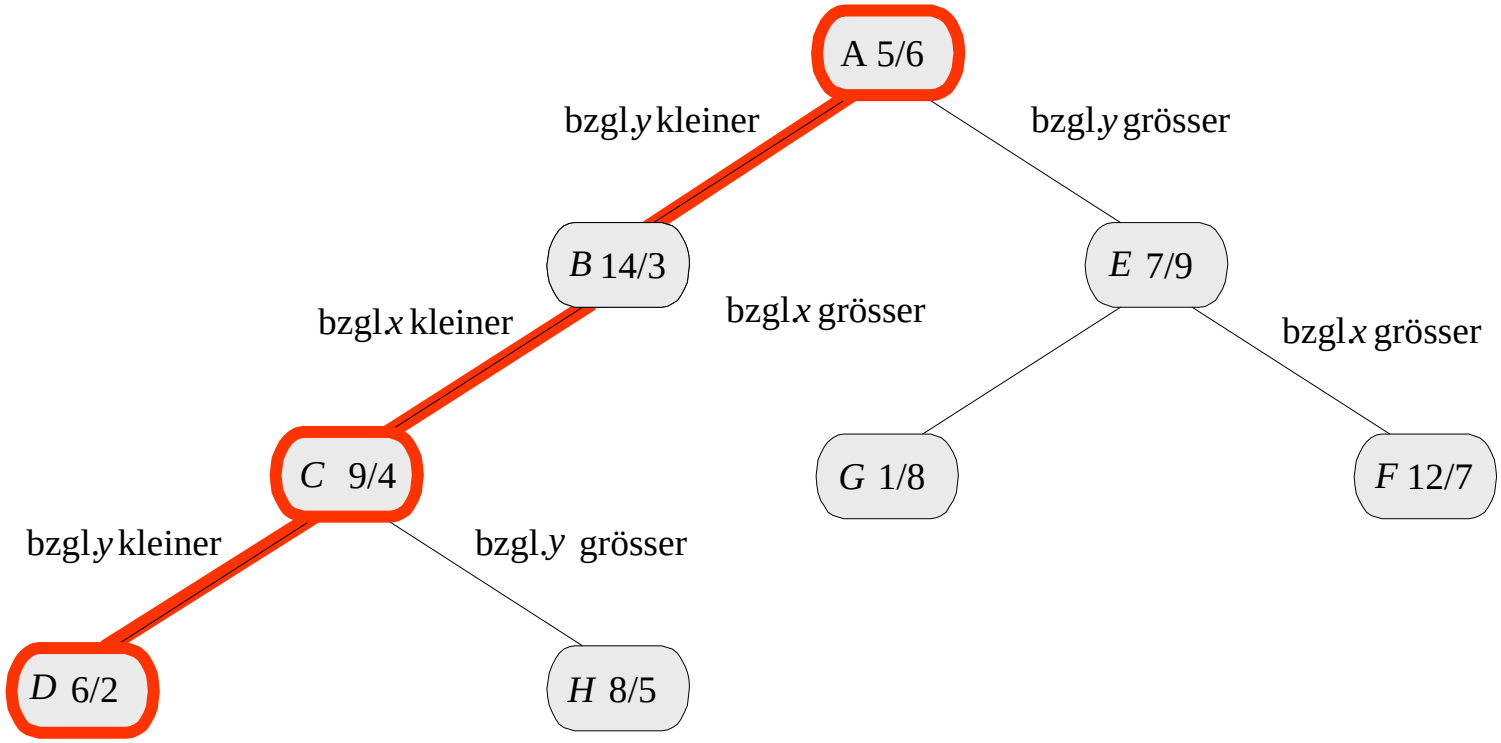
Best Match z.B. finde nächsten Nachbarn zu [7/3]

=> Traversierung mit schrumpfender Rangequery

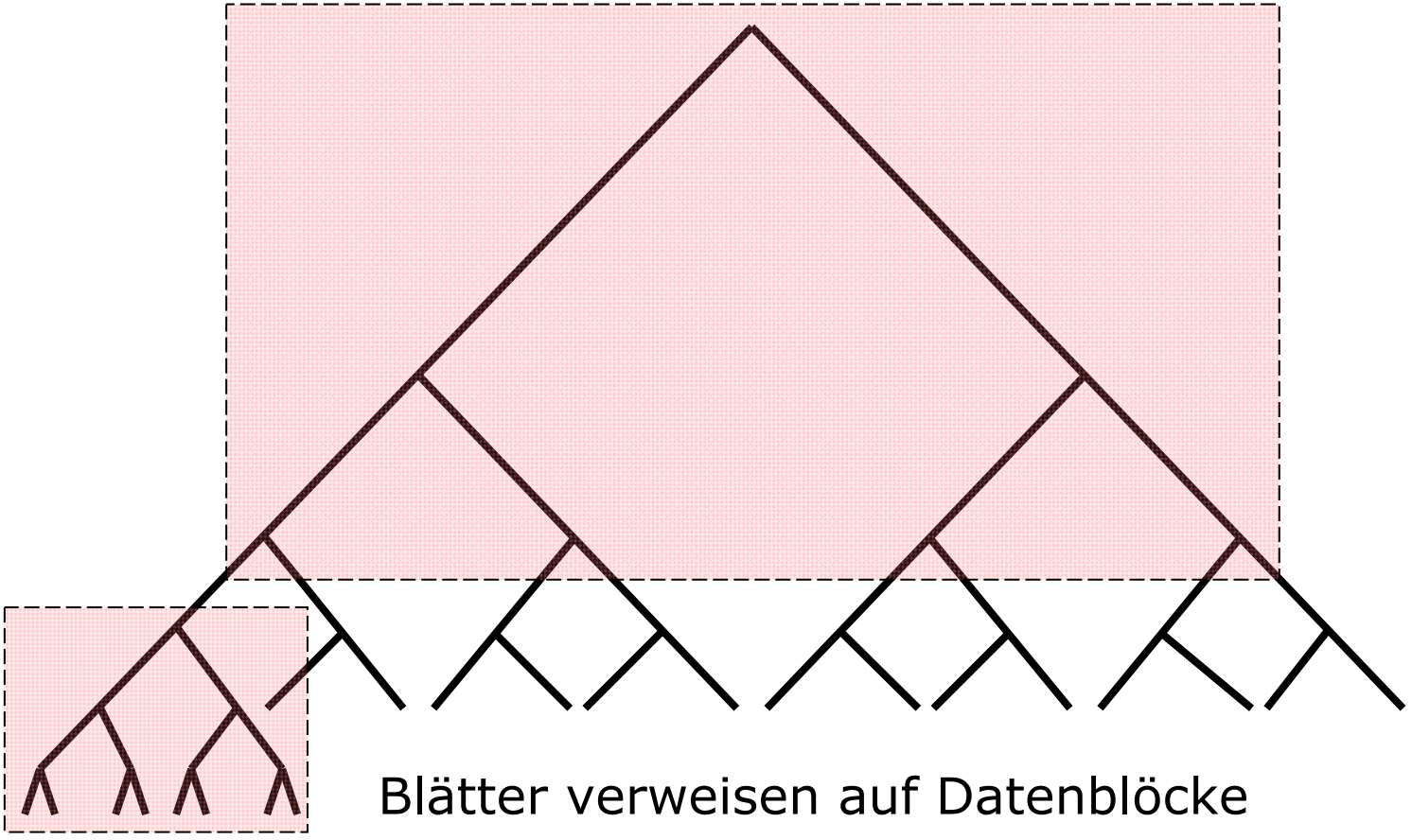
2-d-Baum: Best match



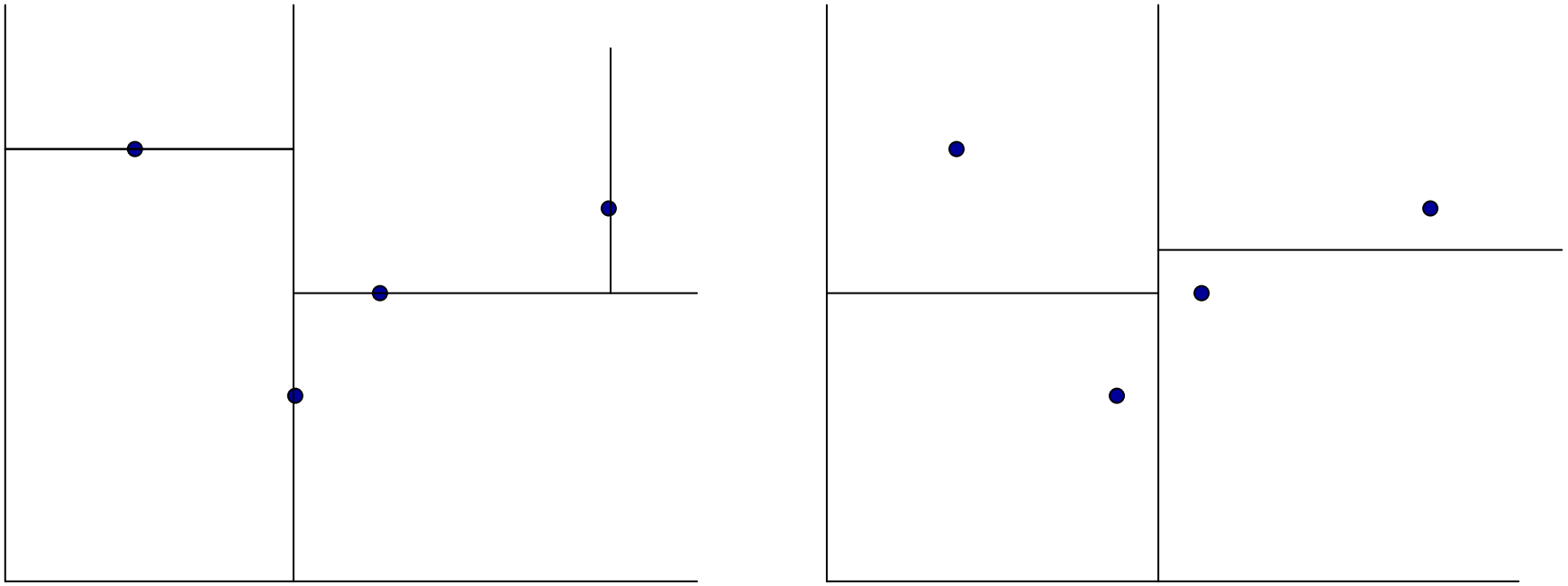
Traversierung für Best Match zu [7/3]



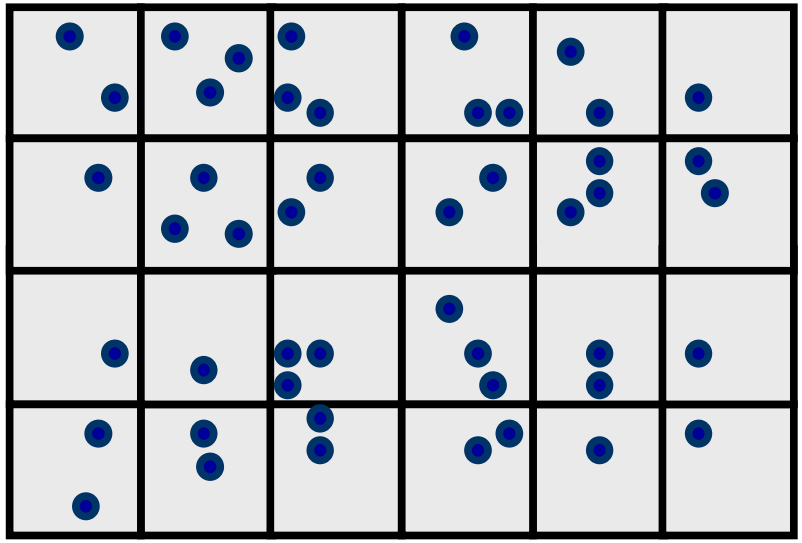
Inhomogene Variante



Partitionierungen des Raumes



Gitterverfahren
mit konstanter Gittergröße



Grid File

Alternative zu fester Gittergröße

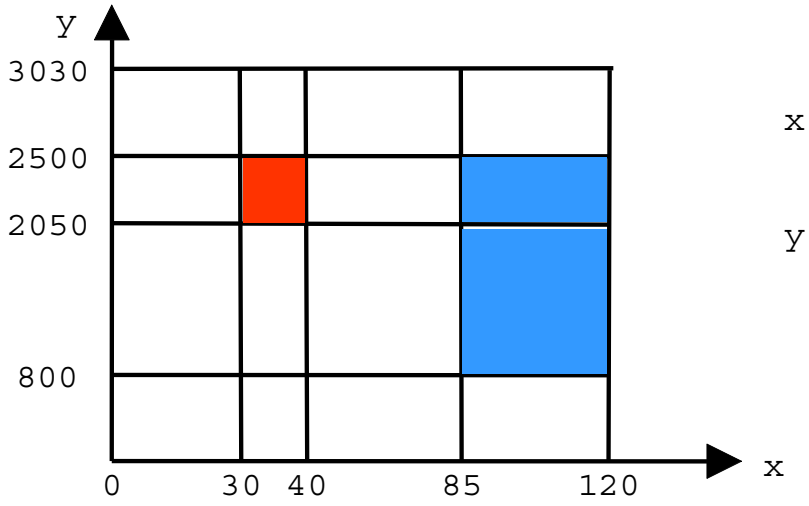
1981 vorgestellt von Hinrichs & Nievergelt

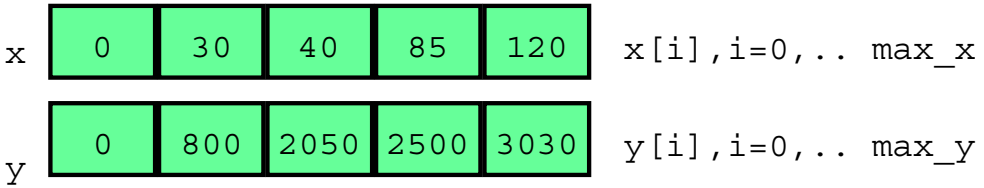
2-Platten-Zugriffsgarantie

Bei k-dimensionalen Tupeln:

- k Skalen zum Einstieg ins Grid-Directory (Haupt)
- Grid Directory zum Finden der Bucket-Nr. (Platte)
- Bucket für Datensätze (Platte)

Grid File im 2-dimensionalen Fall





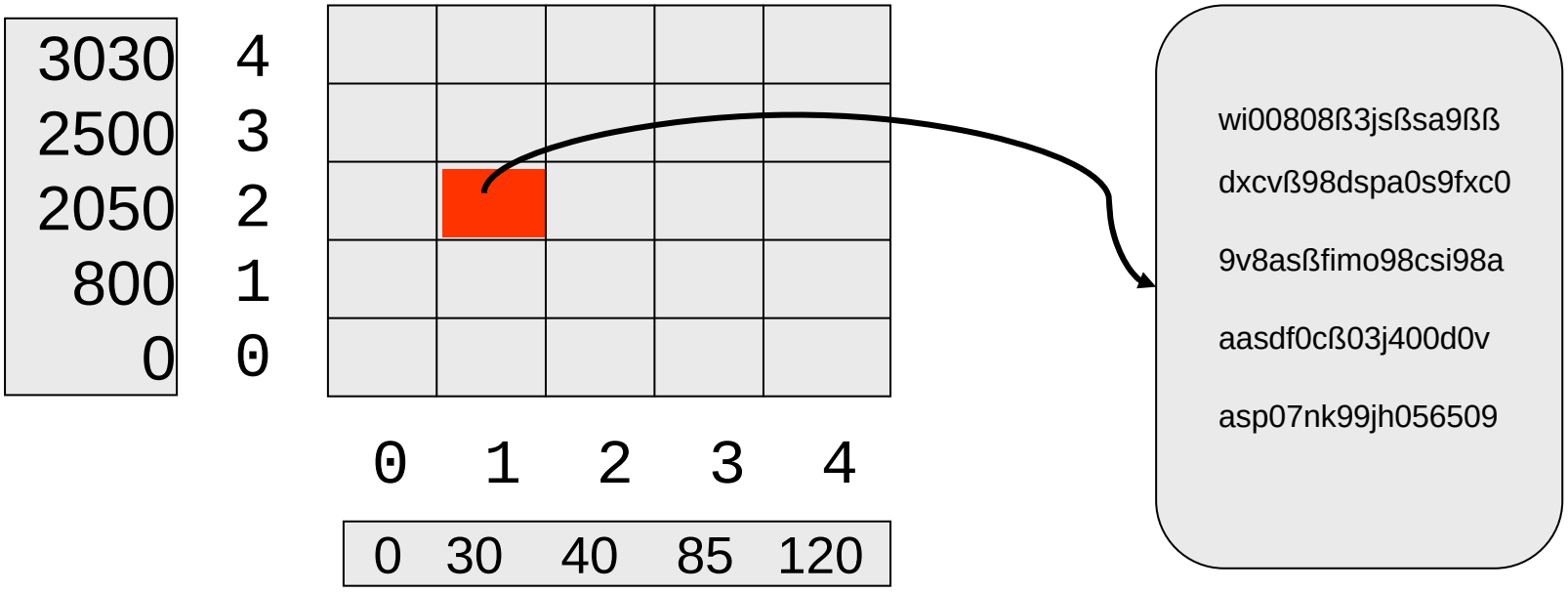
Gitterzelle: rechteckiger Teilbereich

Region: benachbarte Gitterzellen

Bucket: Records einer Region

Alle Records der roten Gitterzelle
sind im Bucket mit Adresse G[1,2]

Bucket Directory



Suche Record [35 / 2400]

Befrage Skalen

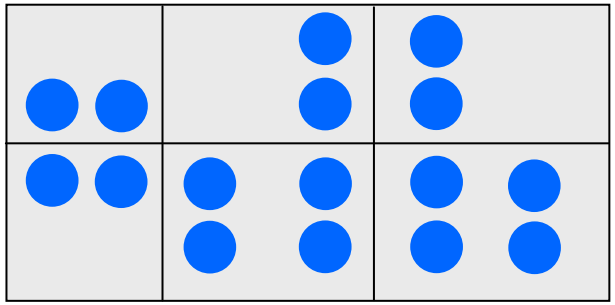
Lade Block mit G[1,2] vom Bucketdirectory

Lade Datenblock

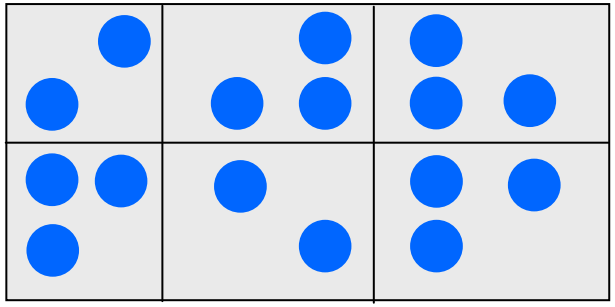
Speichernutzung

Datenblöcke und Directory-Blöcke
nicht immer ausgenutzt:

Beispiel: 4 Datenrecords pro Block:

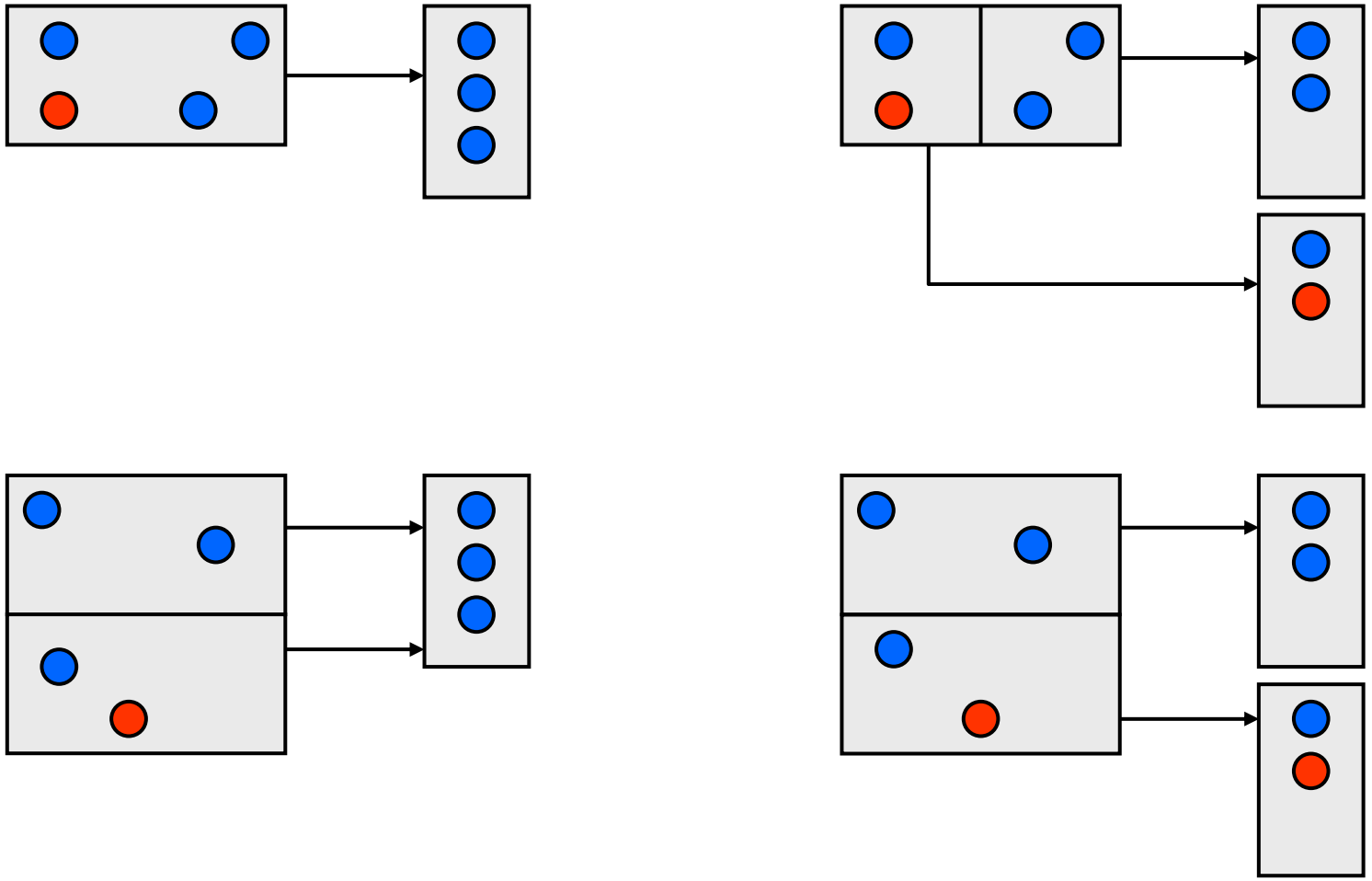


4 volle Datenblöcke
6 Directory-Einträge



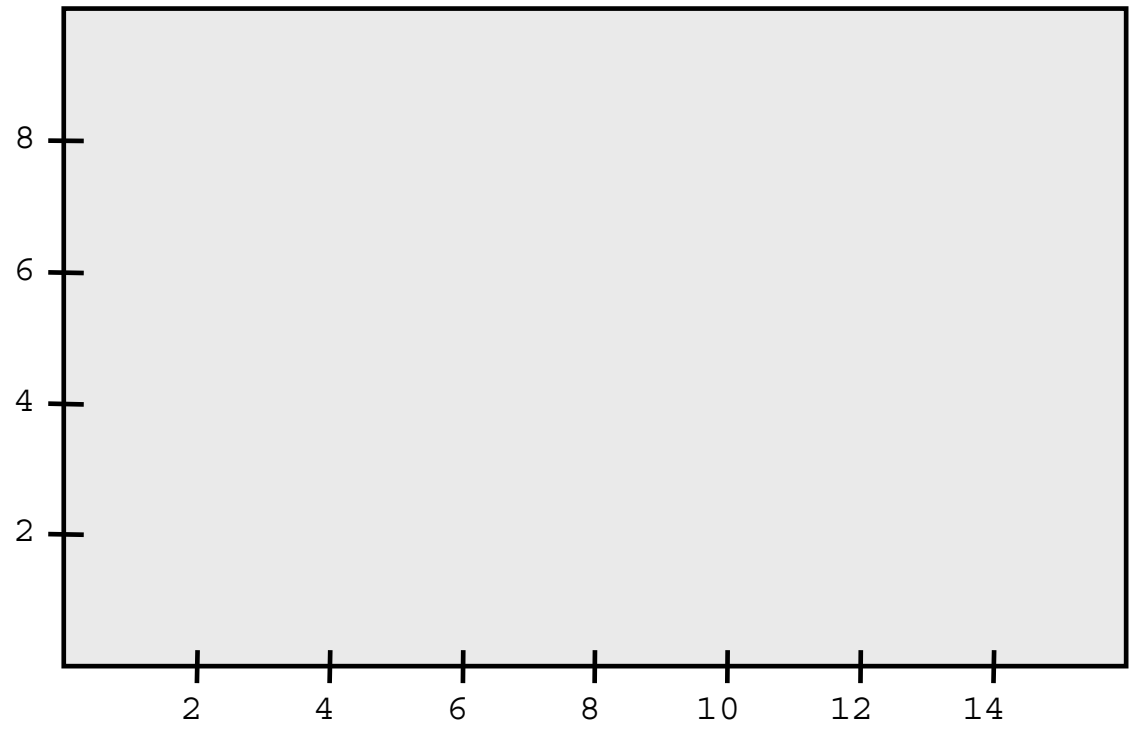
6 Datenblöcke
6 Directory-Einträge

Bucket-Überlauf



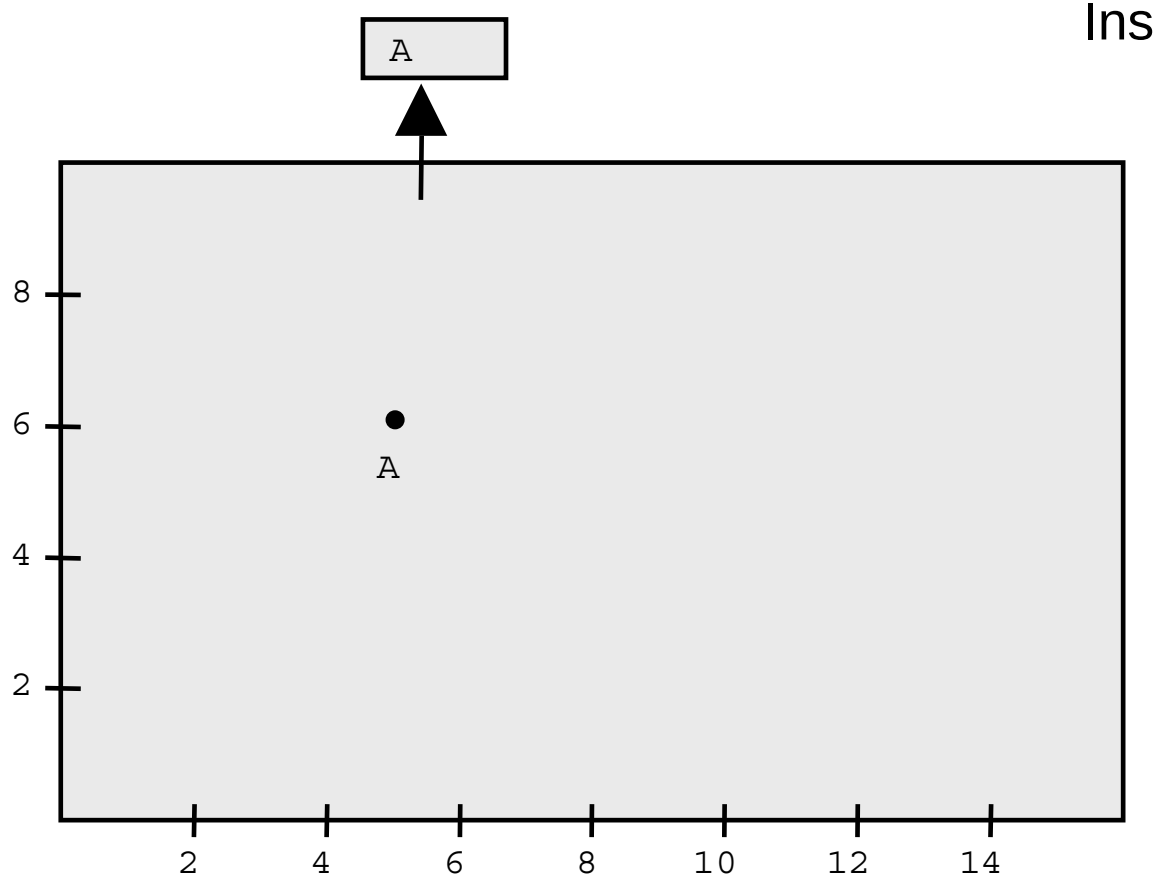
Aufspalten der Regionen

Insert A = [5 / 6]



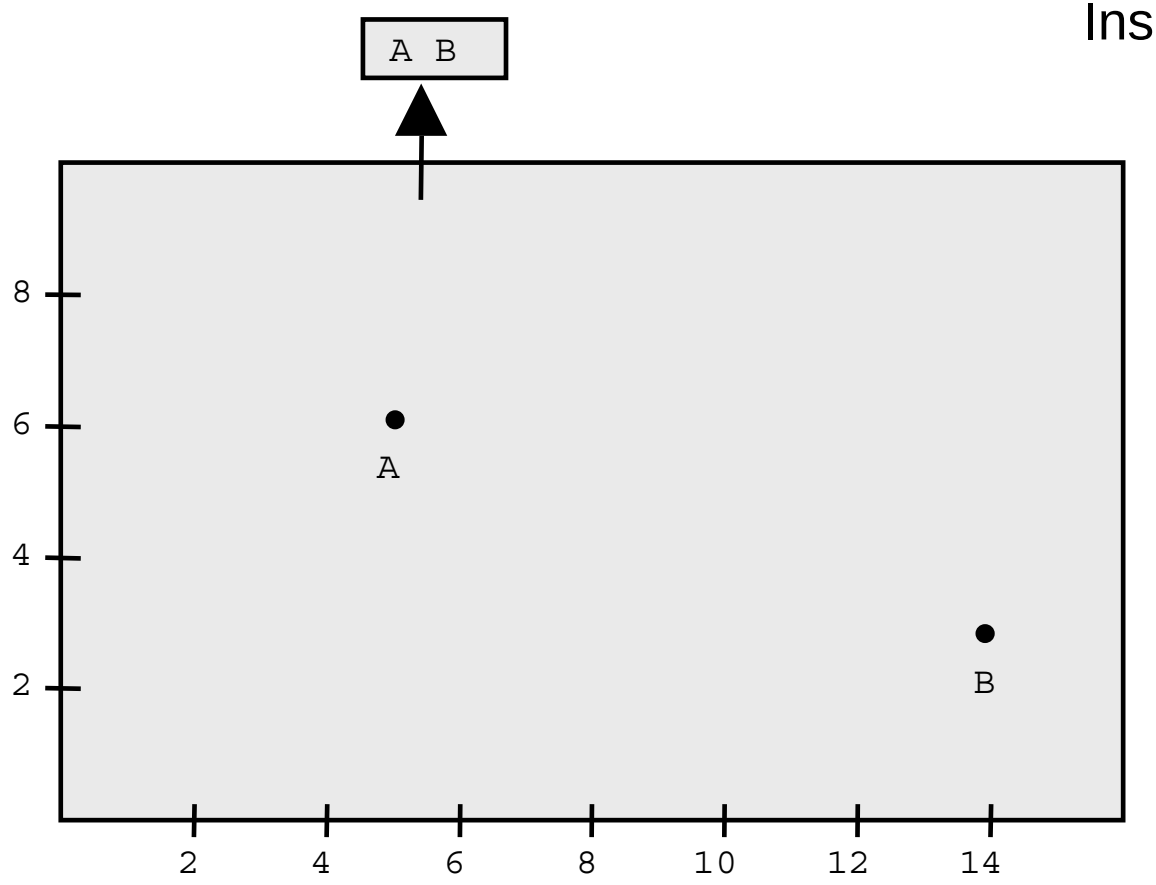
Aufspalten der Regionen

Insert B = [14 / 3]

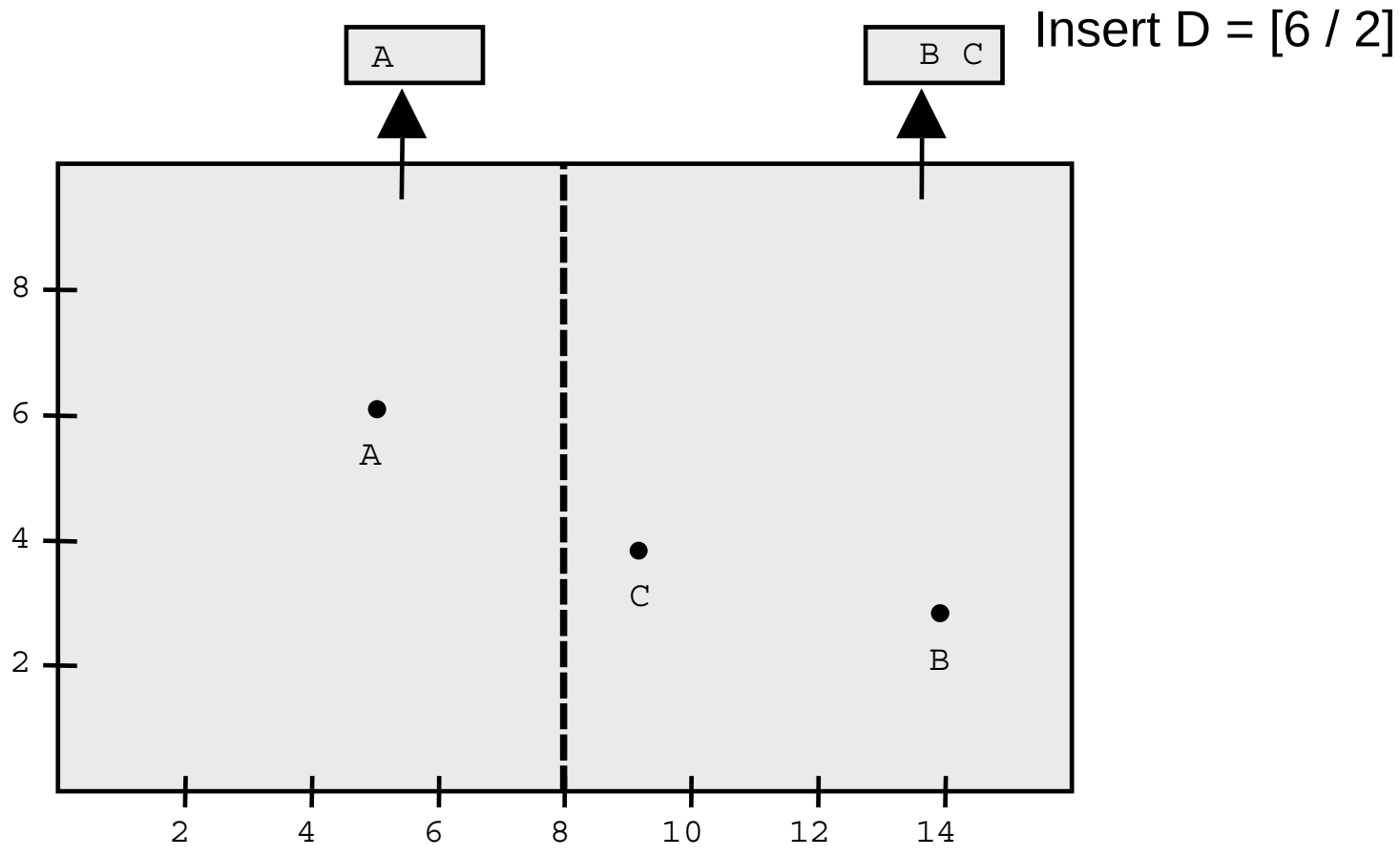


Aufspalten der Regionen

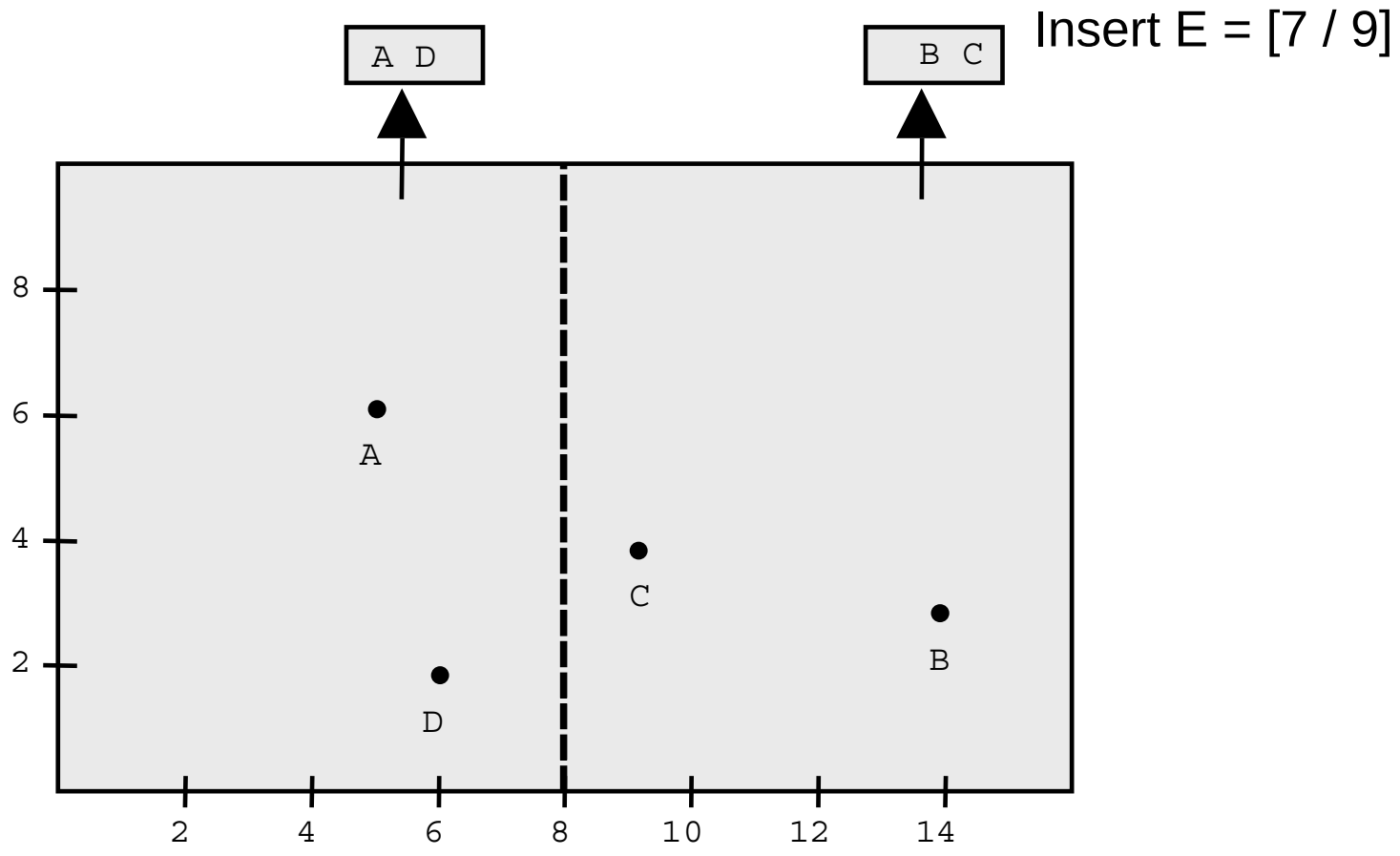
Insert C = [9 / 4]



Aufspalten der Regionen

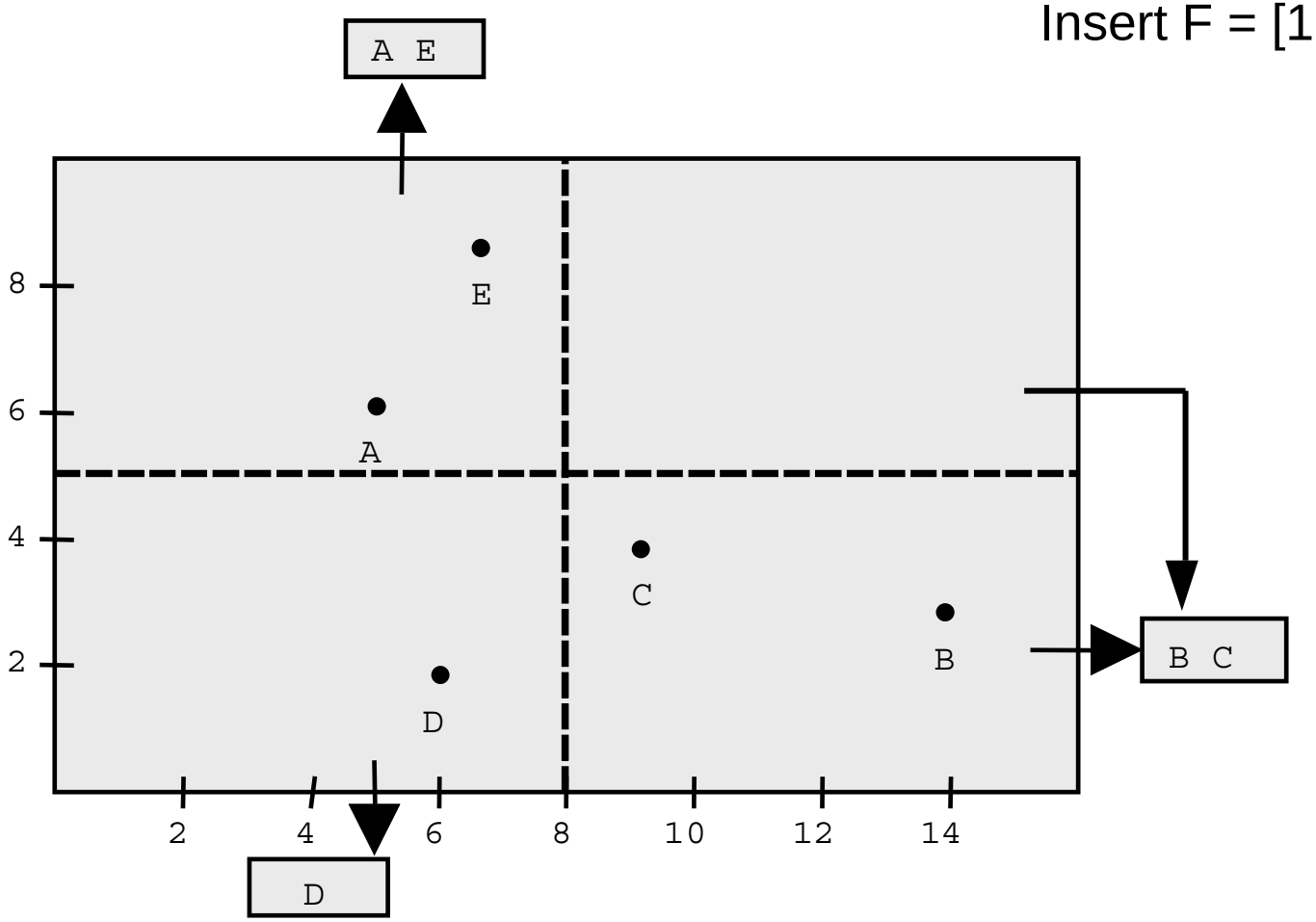


Aufspalten der Regionen



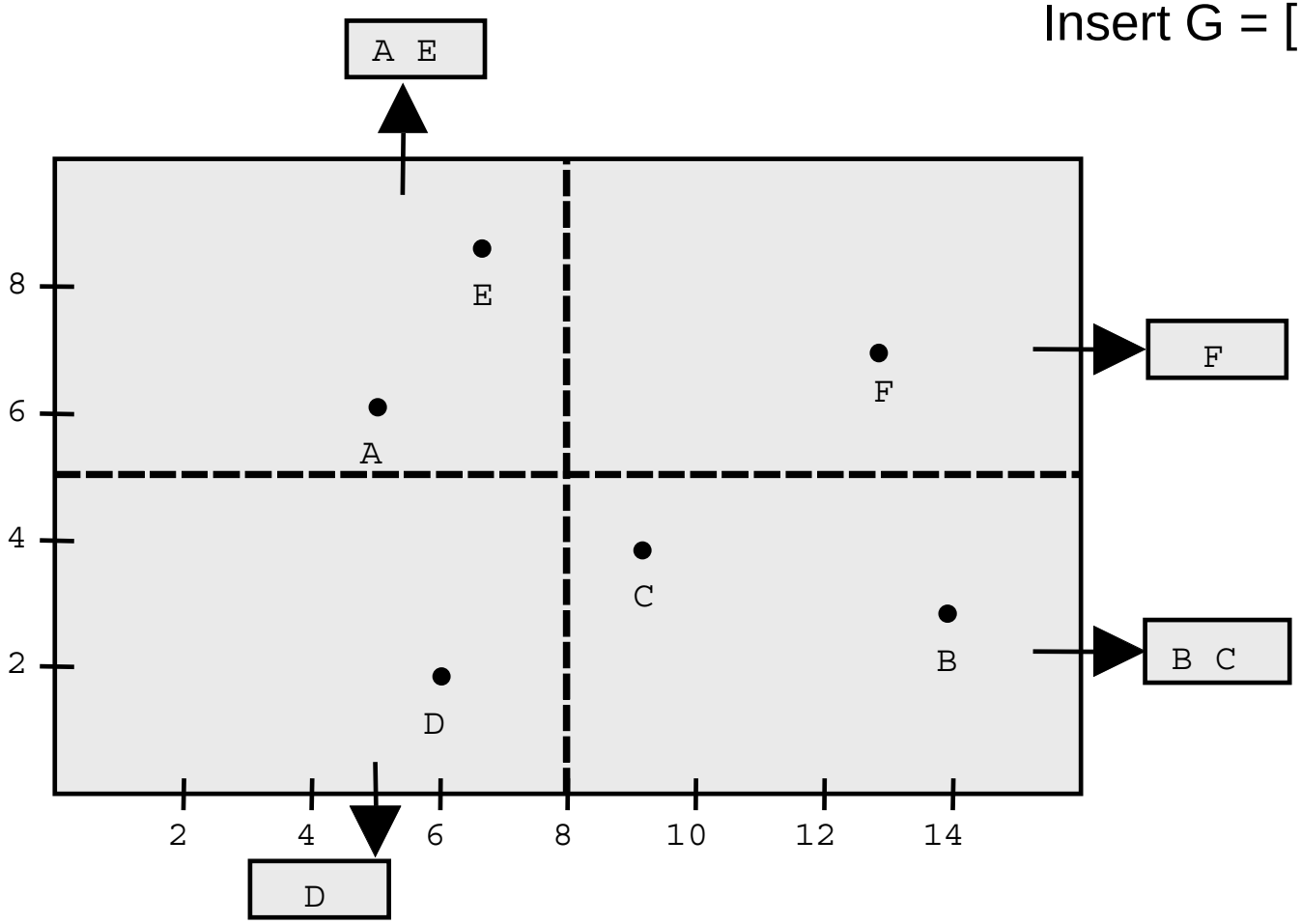
Aufspalten der Regionen

Insert F = [12 / 7]

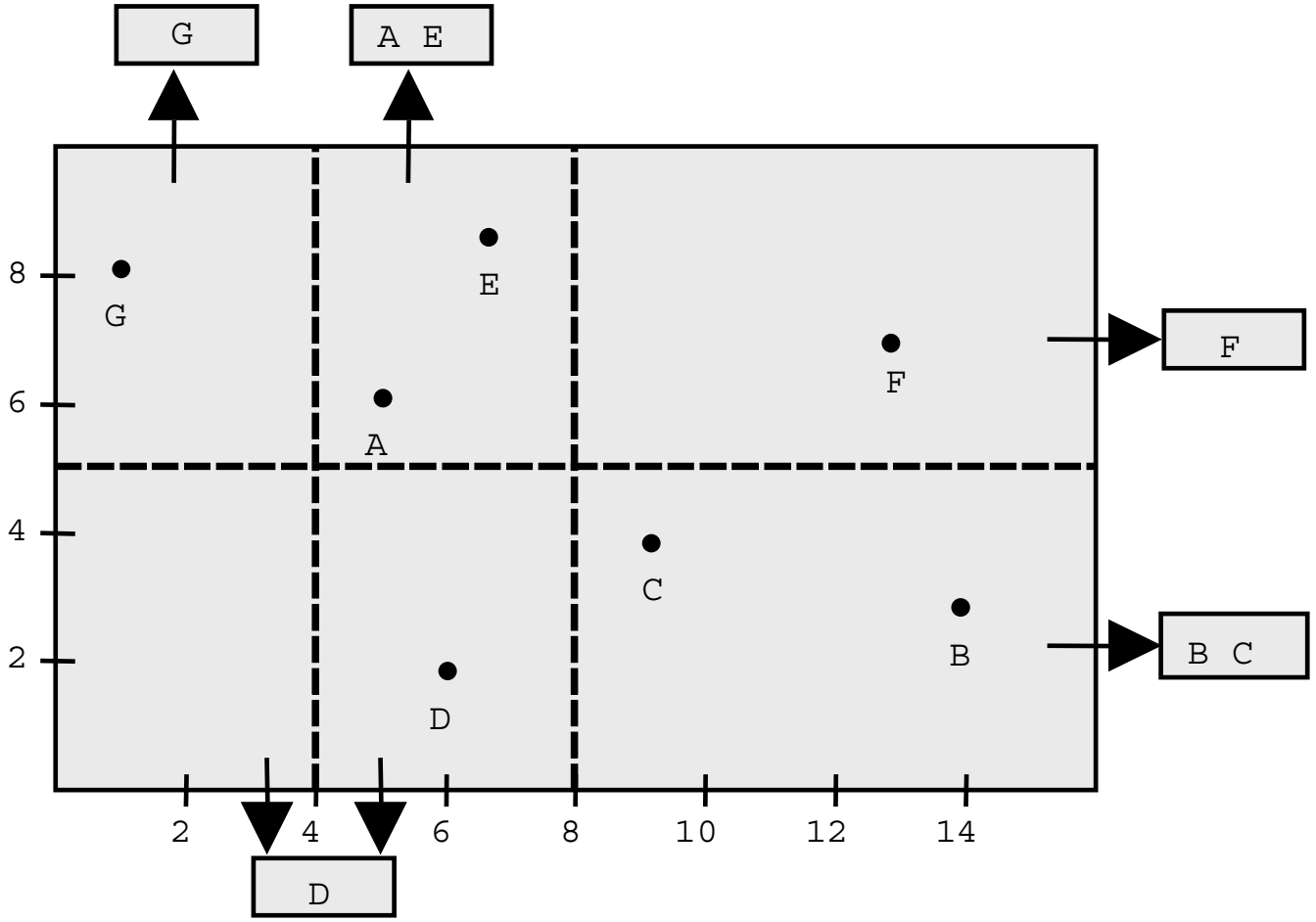


Aufspalten der Regionen

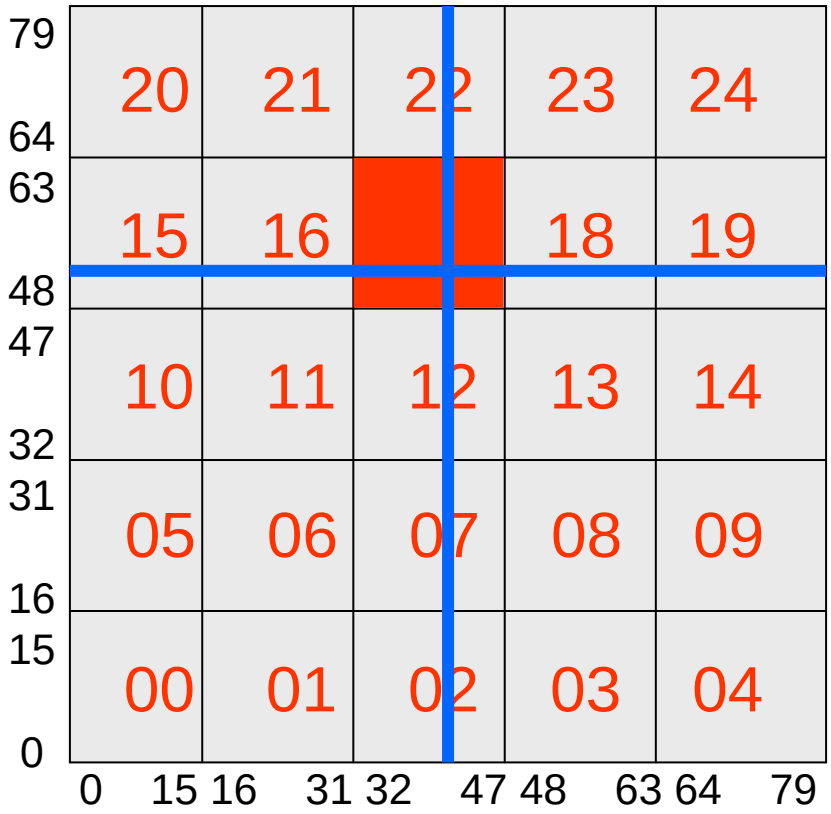
Insert G = [1 / 8]



Aufspalten der Regionen



Directory-Zugriff

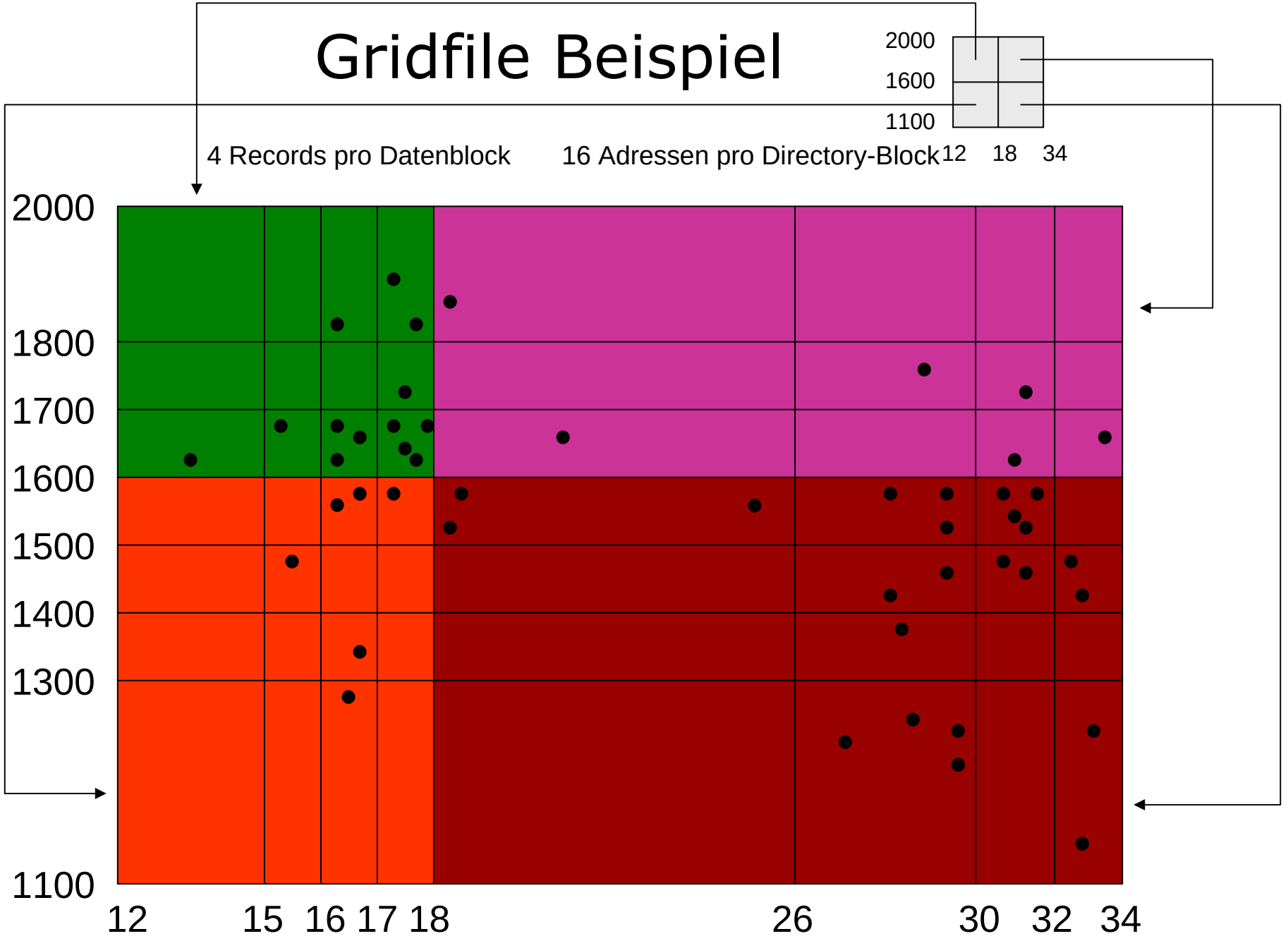


Indizes: $0 \le i, j \le 79$ 42 51

Directoryblock $b = 5 * \lfloor j/16 \rfloor + \lfloor i/16 \rfloor$ 5*3+2=17

Adresse $a = 16 * (j \bmod 16) + (i \bmod 16)$ 16*3+10=58

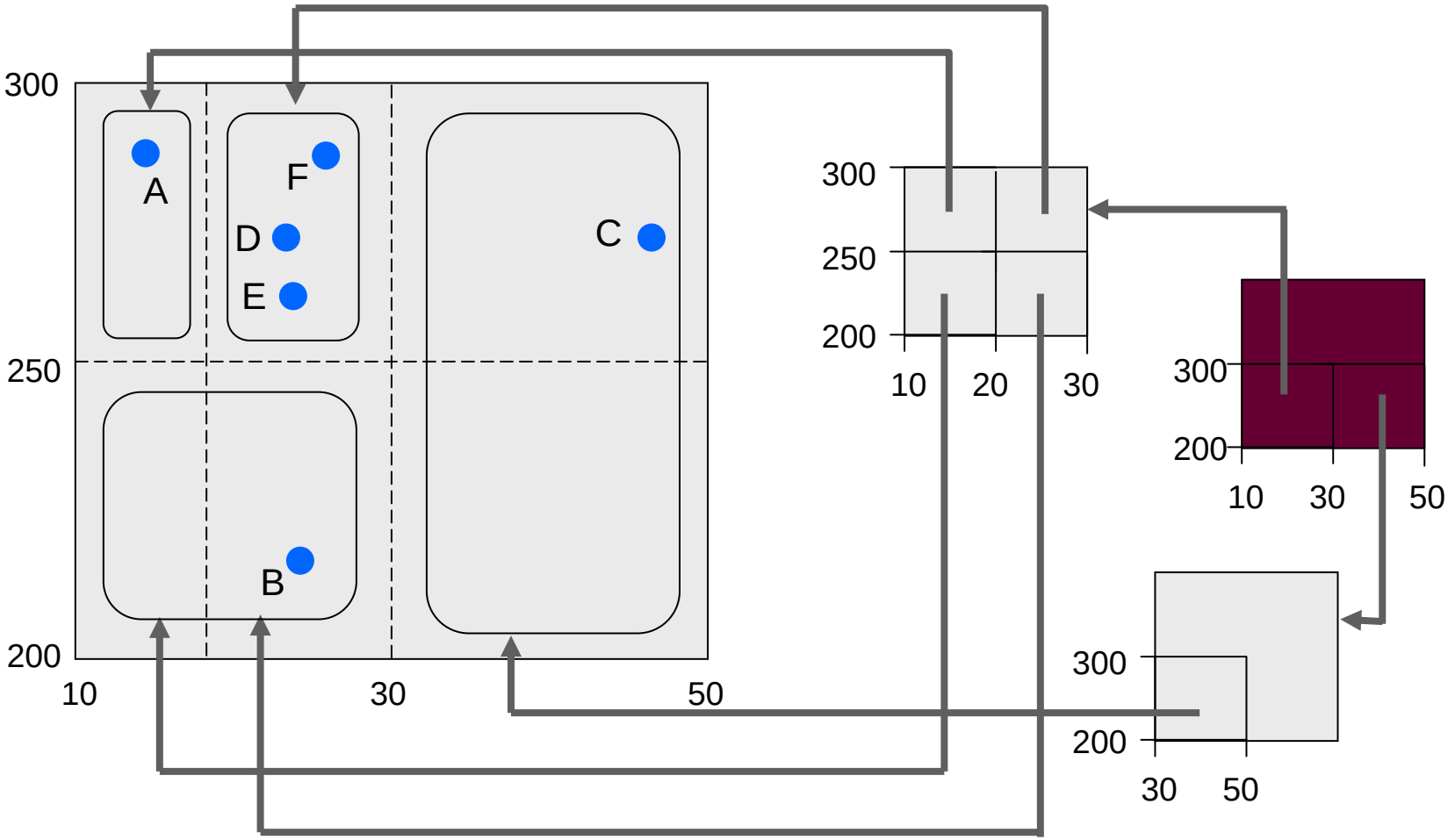
Gridfile Beispiel



Gridfile: Zahlenbeispiel

Größe eines Datenrecords:	100 Byte
Zahl der Datenrecords:	300.000
maximale Zahl der Datenrecords pro Block	
mittlere Zahl der Datenrecords pro Block:	
Zahl der Datenblöcke:	
maximale Zahl von Adressen pro Directoryblock:	
mittlere Zahl von Adressen pro Directoryblock	
Zahl der Directoryblöcke:	
Größe des Root-Directory:	

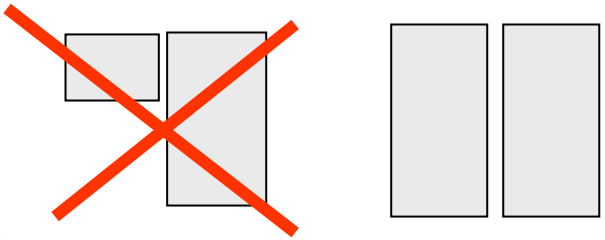
Dynamik des Grid Directory



Vereinigung von Regionen

bei zu geringer Bucketauslastung:

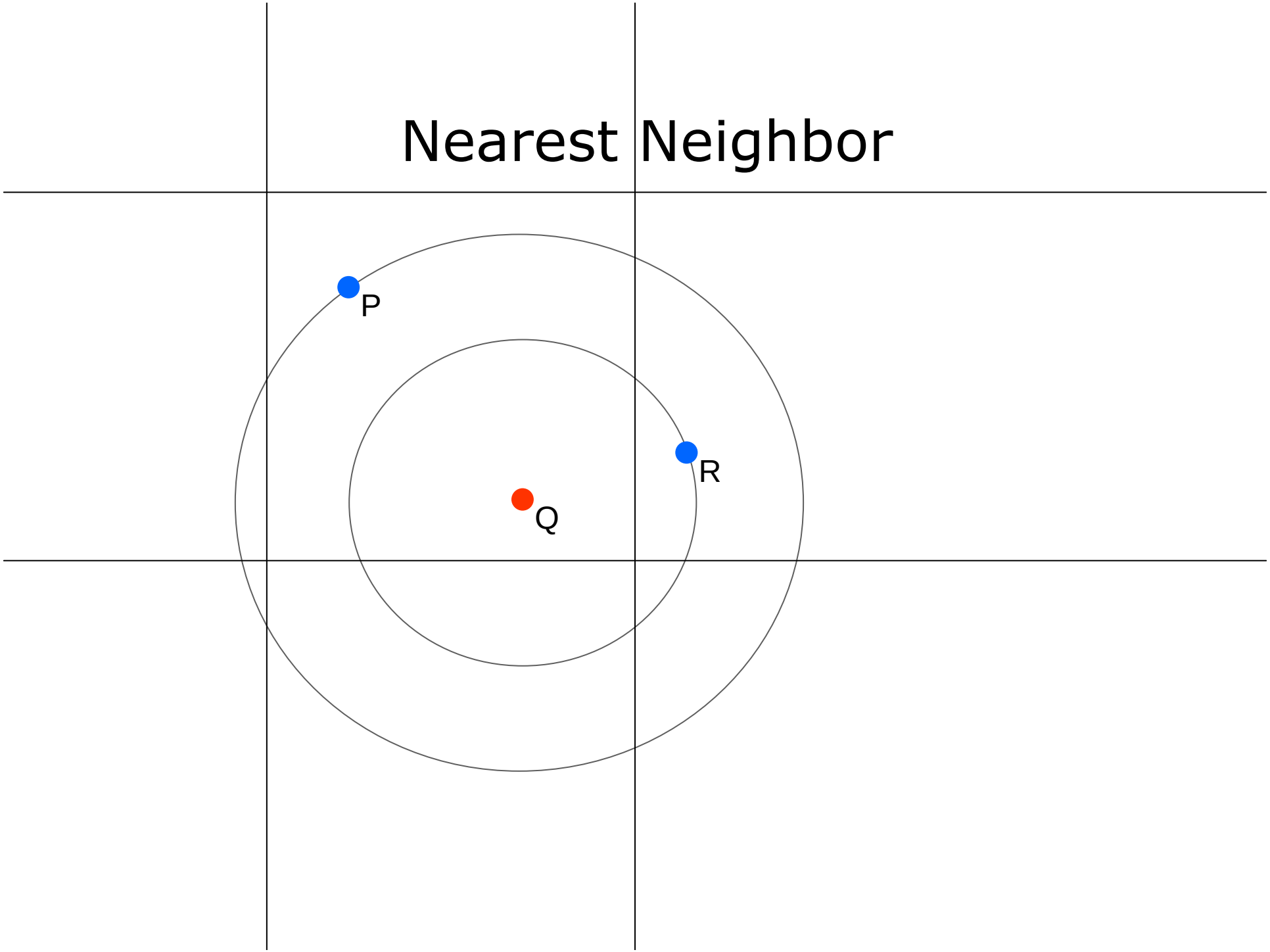
- Benachbarte Regionen vereinigen
- Vereinigung muß Rechteck ergeben



Vereinigung wird ausgelöst

- wenn Bucket < 30 % Auslastung
- wenn vereinigtes Bucket < 70 % Auslastung

Nearest Neighbor

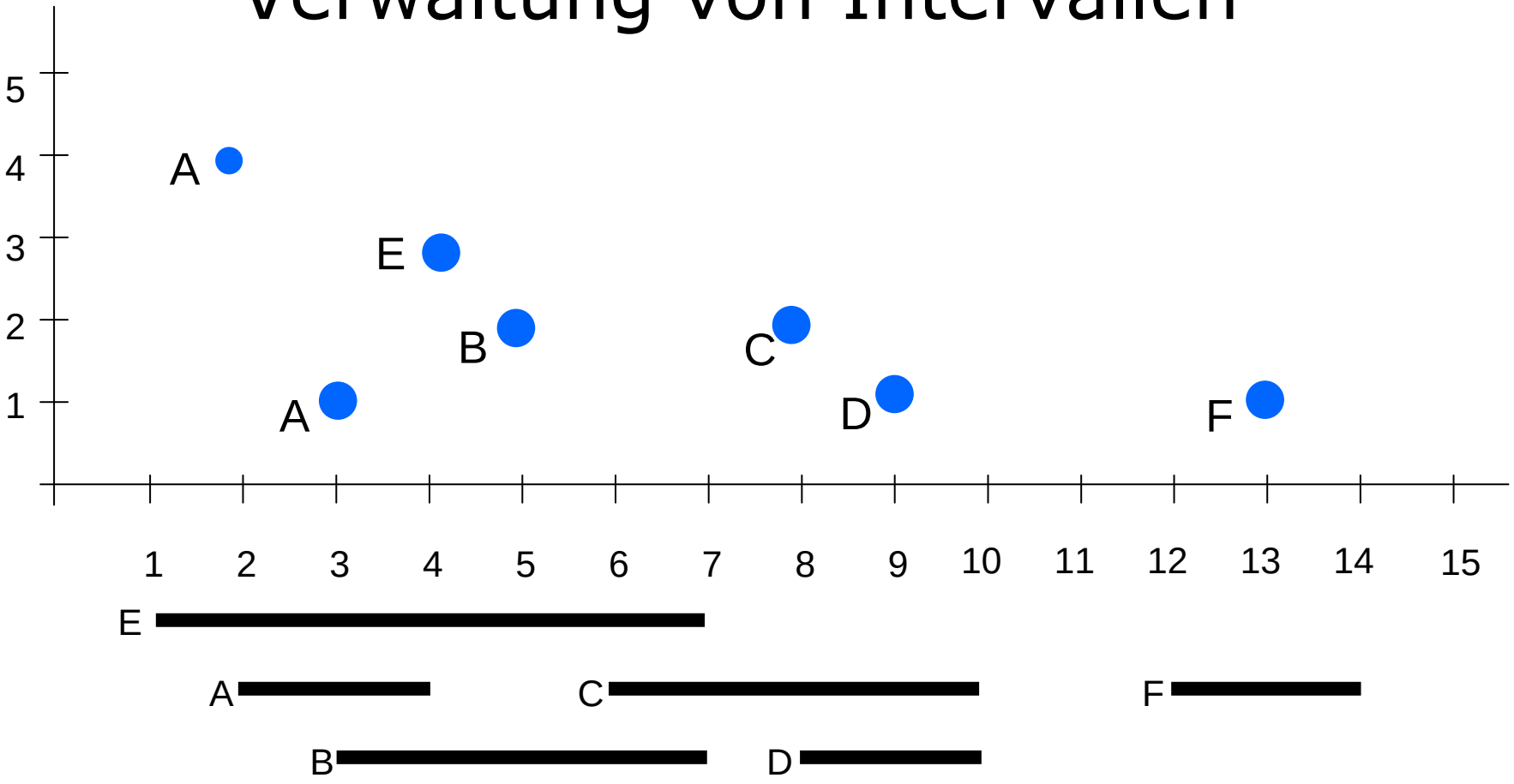


Verwaltung von geometrischen Objekten

Grid File unterstützt Range-Query

bisher:	k Attribute
Jetzt:	k-dimensionale Punkte

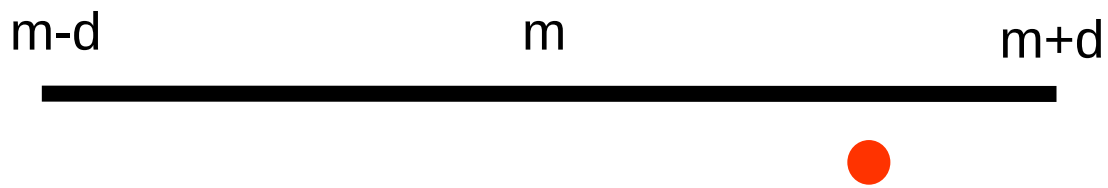
Verwaltung von Intervallen



[Anfang / Ende]

Besser: [Mittelpunkt / halbe Länge]

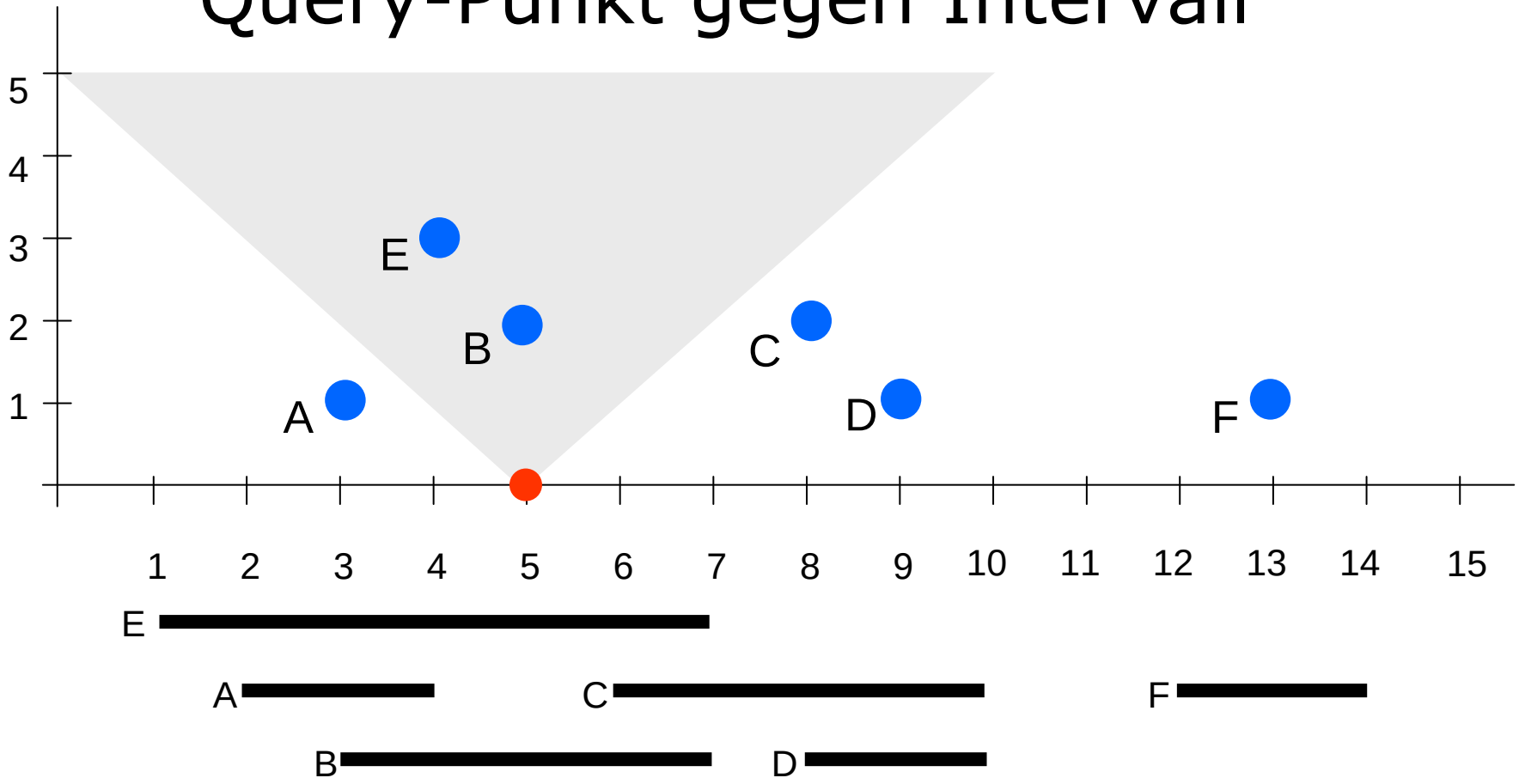
Query-Punkt gegen Intervall



Punkt p liegt im Intervall mit Mitte m und halber Länge d ⇔

$m-d \leq p \leq m+d$

Query-Punkt gegen Intervall



Punkt p liegt im Intervall mit Mitte m und halber Länge d $\Leftrightarrow$

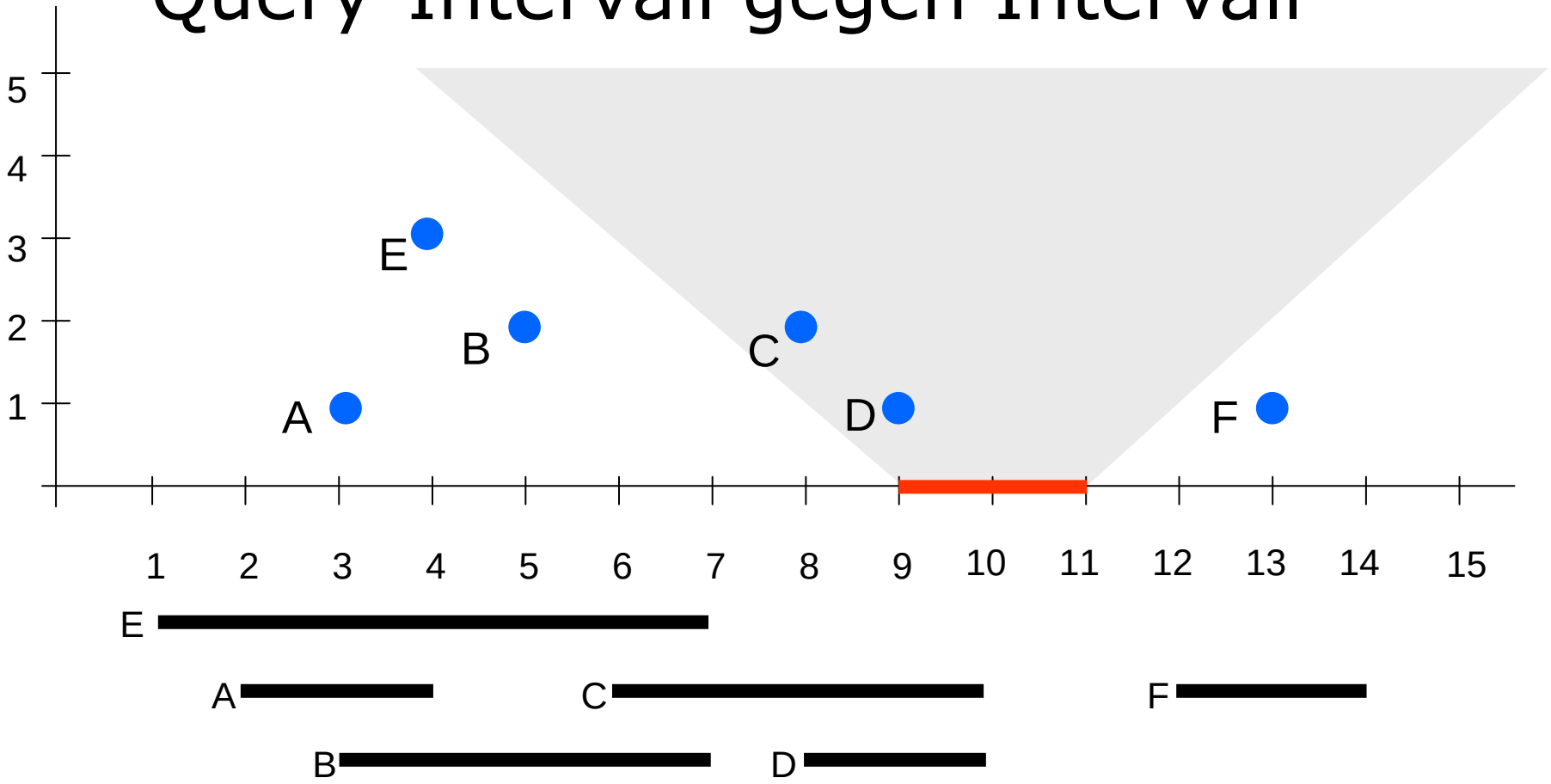
$m-d \leq p \leq m+d$ $\quad p=5 \quad \Rightarrow m-d \leq 5 \leq m+d$

Query-Intervall gegen Intervall

Intervall mit Mitte s und halber Länge t schneidet
Intervall mit Mitte m und halber Länge d ⇔

$m-d \leq s+t$ und $s-t \leq m+d$

Query-Intervall gegen Intervall



Intervall mit Mitte s und halber Länge t schneidet
Intervall mit Mitte m und halber Länge d ⇔

$m-d \leq s+t$ und $s-t \leq m+d$ $\quad$ $s=10, t=1 \Rightarrow m-d \leq 11$ und $9 \leq m+d$

Query-Rechteck gegen Rechteck

Stelle Rechteck durch vierdimensionalen Punkt dar