

Datenbanksysteme 2015

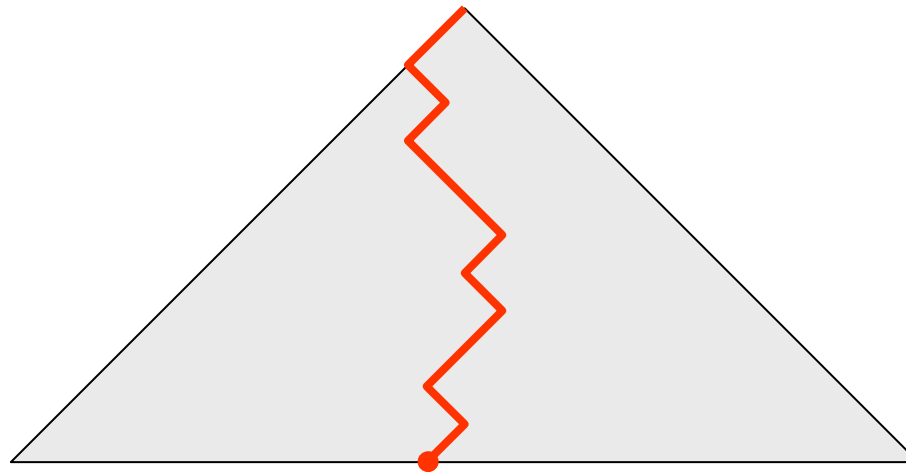
Kapitel 5:
Mehrdimensionale Suchstrukturen
Vorlesung vom 28.04.2015

Oliver Vornberger

Institut für Informatik
Universität Osnabrück

Suche nach Primärschlüssel

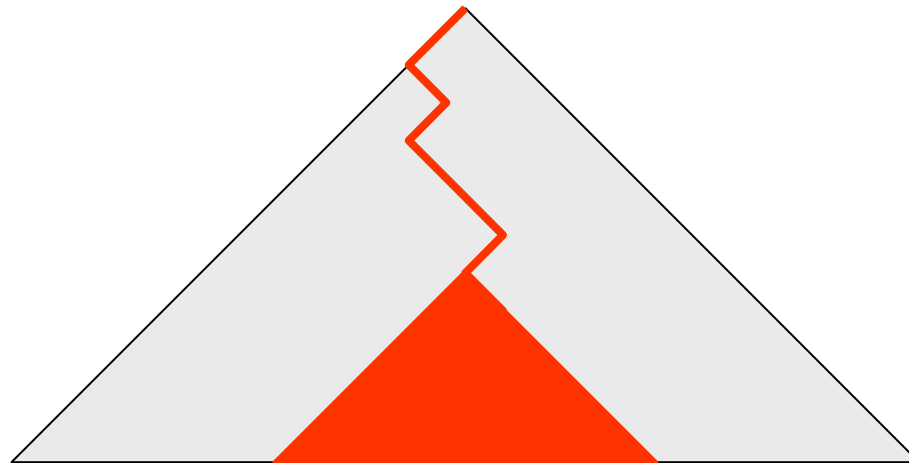
```
select name from studenten  
where matrnr = 4711
```



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

Suche nach Sekundärschlüssel

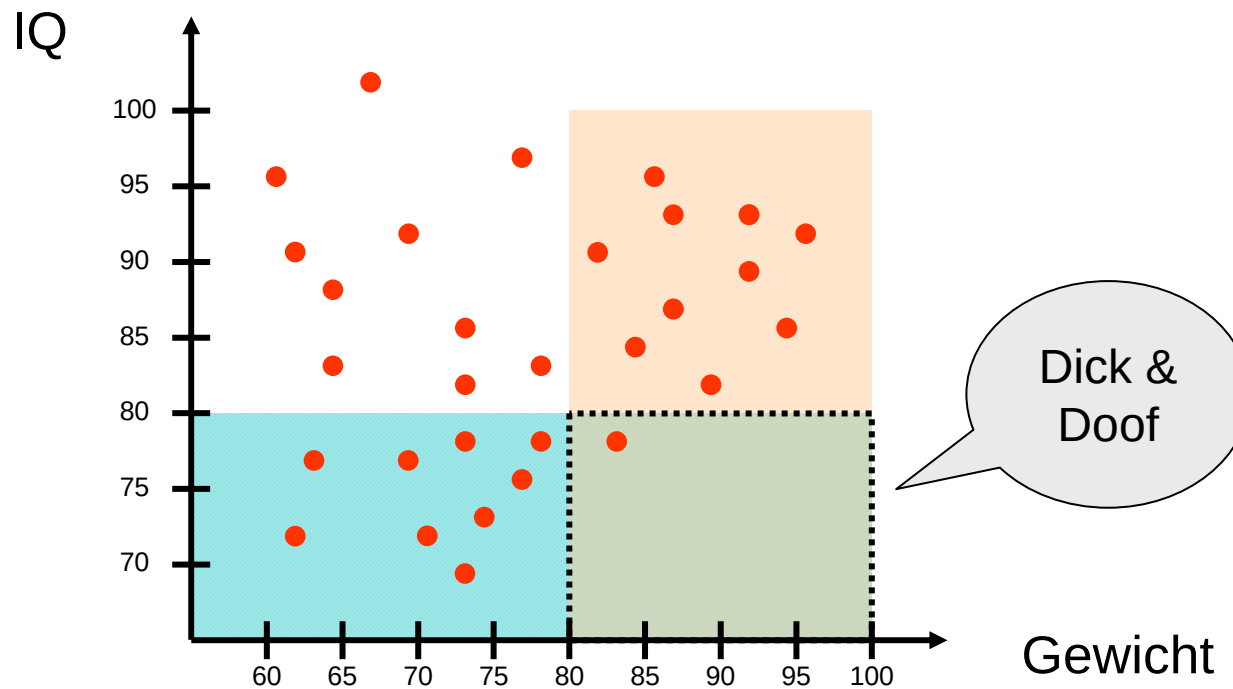
```
select name from studenten  
where groesse > 170  
and    groesse < 180
```



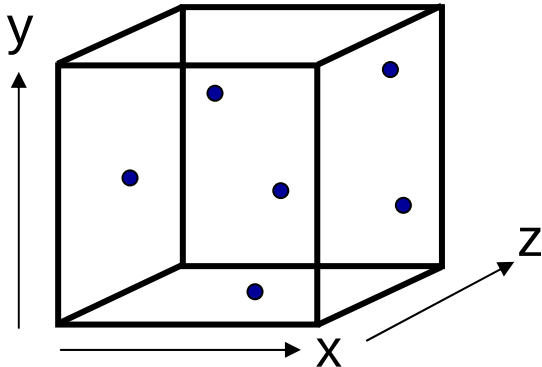
Aufwand bei m Treffern von n Records: $\log_k (c \cdot n) + c \cdot m$

Suche nach 2 Sekundärschlüsseln

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



k-d-Baum



zur Verwaltung von mehrdimensionalen Datenpunkten

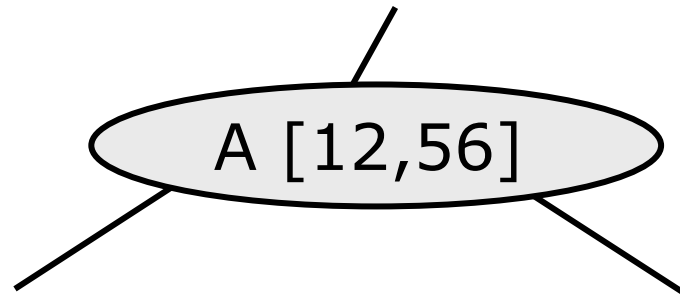
Verallgemeinerung des binären Suchbaums
mit k-dimensionalem Sortierschlüssel

Homogene Variante: Baumknoten = Datenrecord + Zeiger

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

Ebene i modulo k diskriminiert bzgl. Dimension i

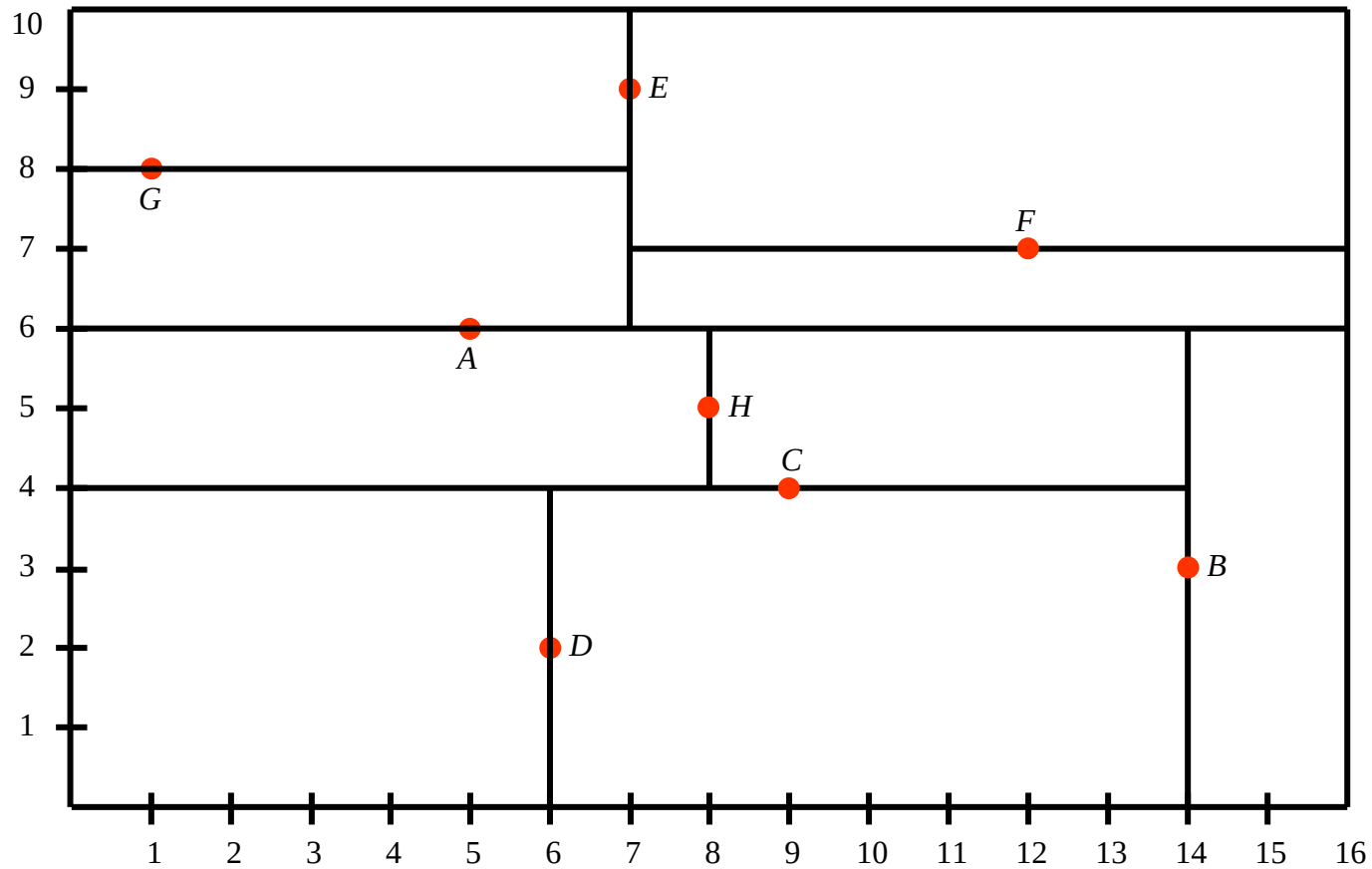
2-d-Baum



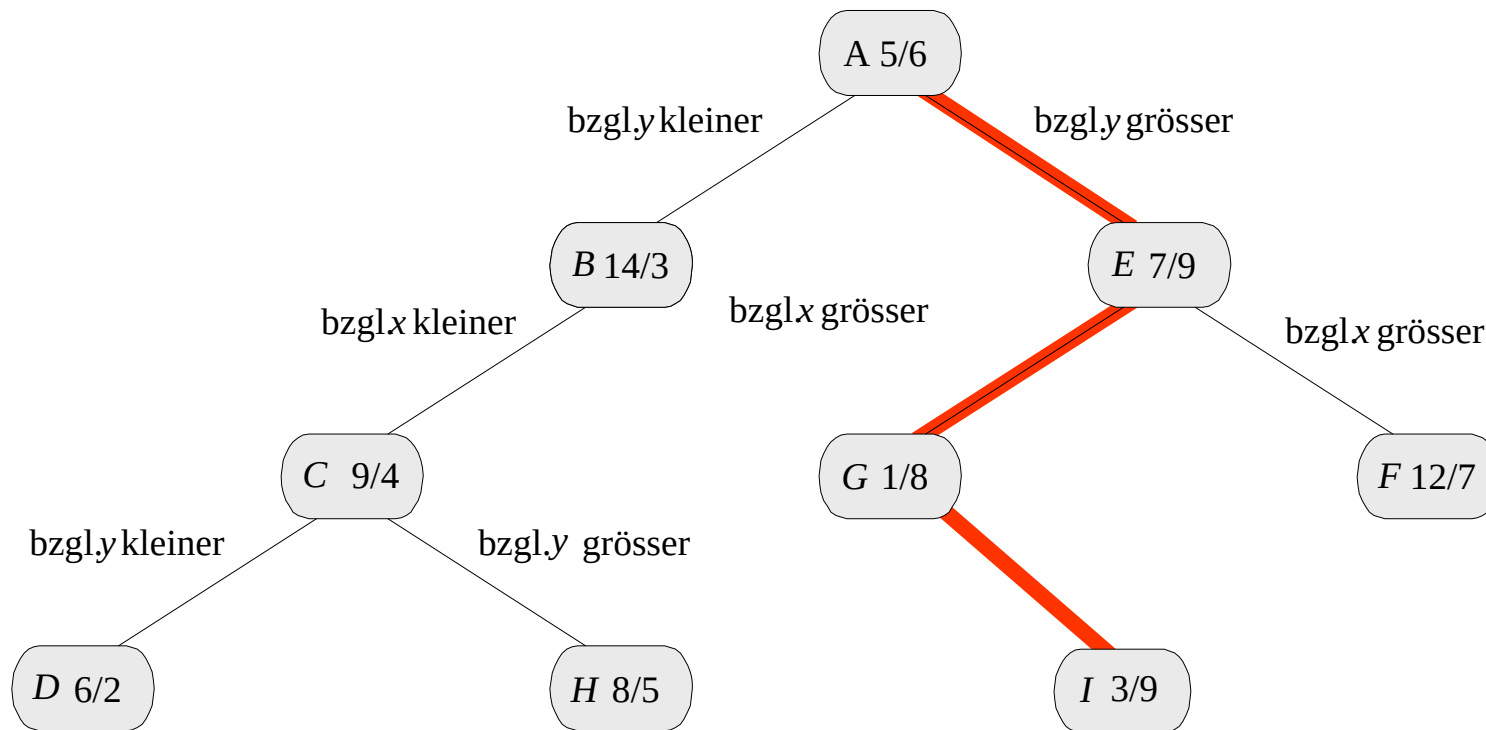
Im 2-dimensionalen Fall gilt für Knoten mit Schlüssel $[x/y]$:

	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



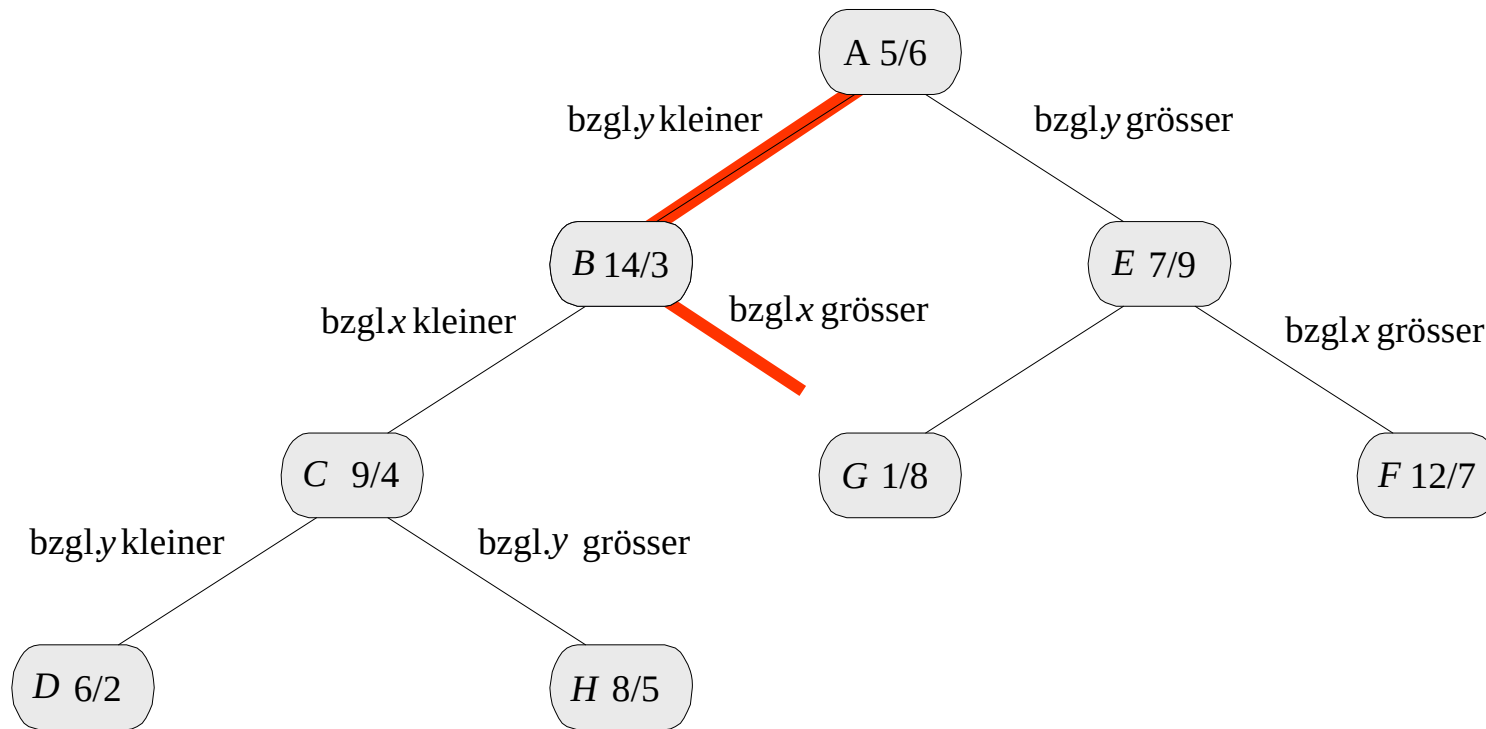
2-d-Baum: Beispiel



Insert

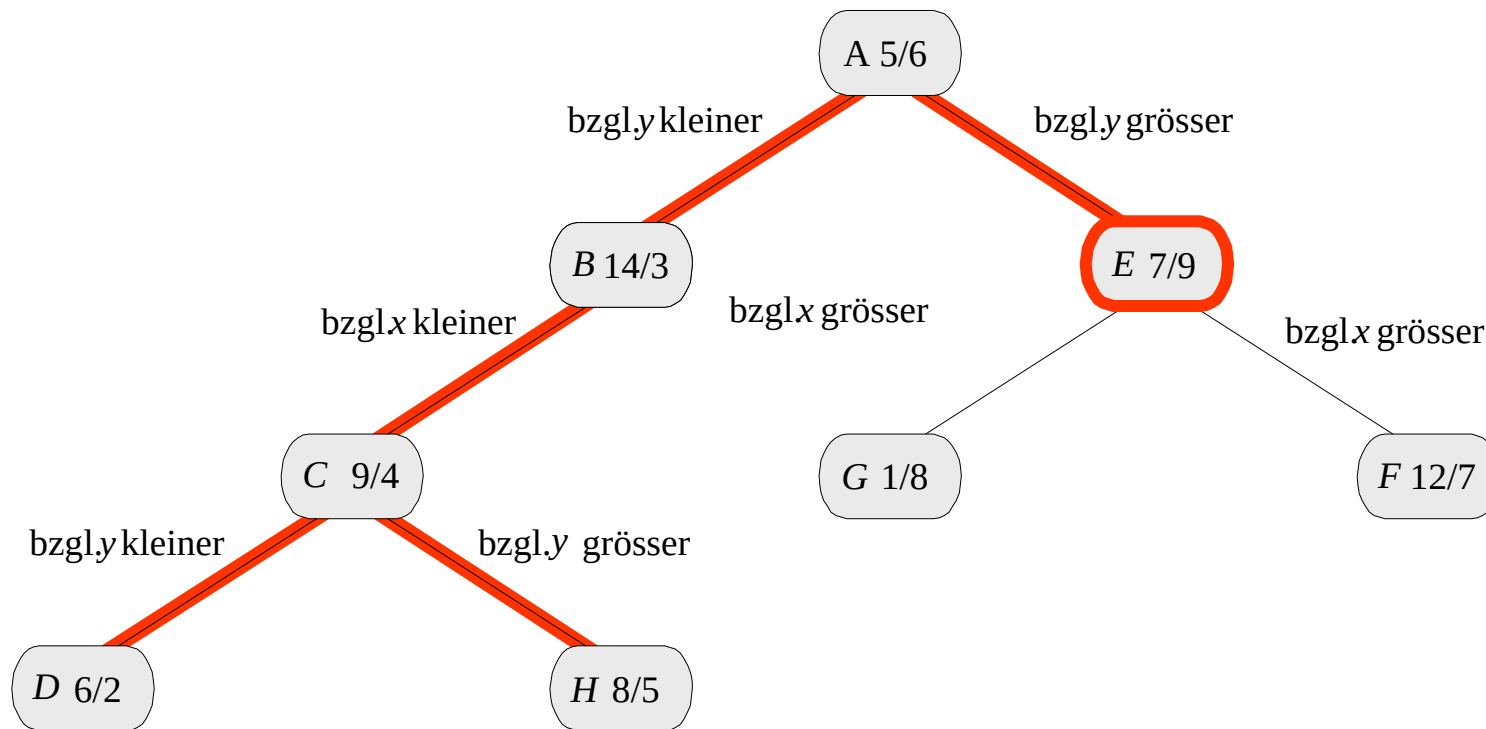
z.B. füge Record [3/9] ein

2-d-Baum: Beispiel



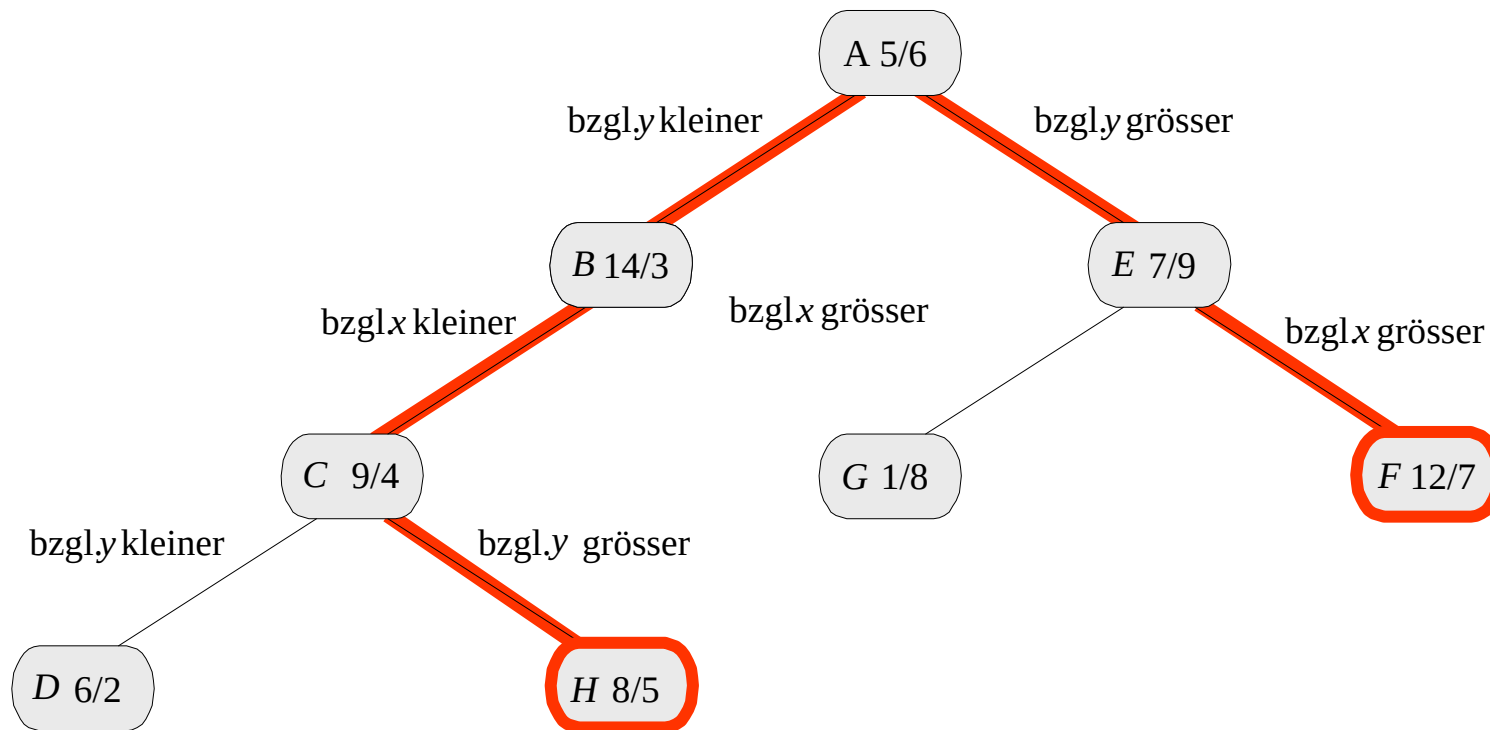
Exact Match z.B. finde Record $[15/5]$

2-d-Baum: Beispiel



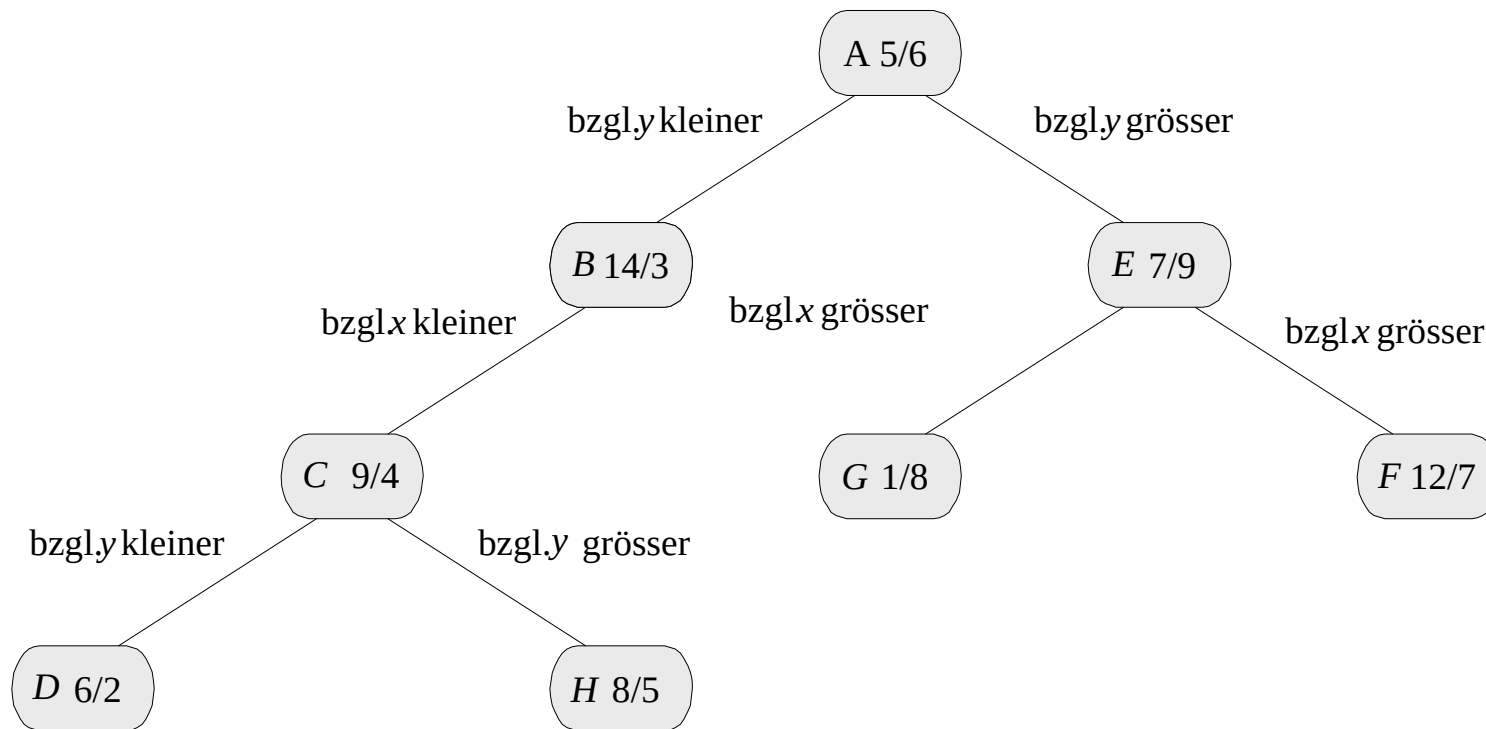
Partial Match z.B. finde alle Records $[x/y]$ mit $x=7$

2-d-Baum: Beispiel



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

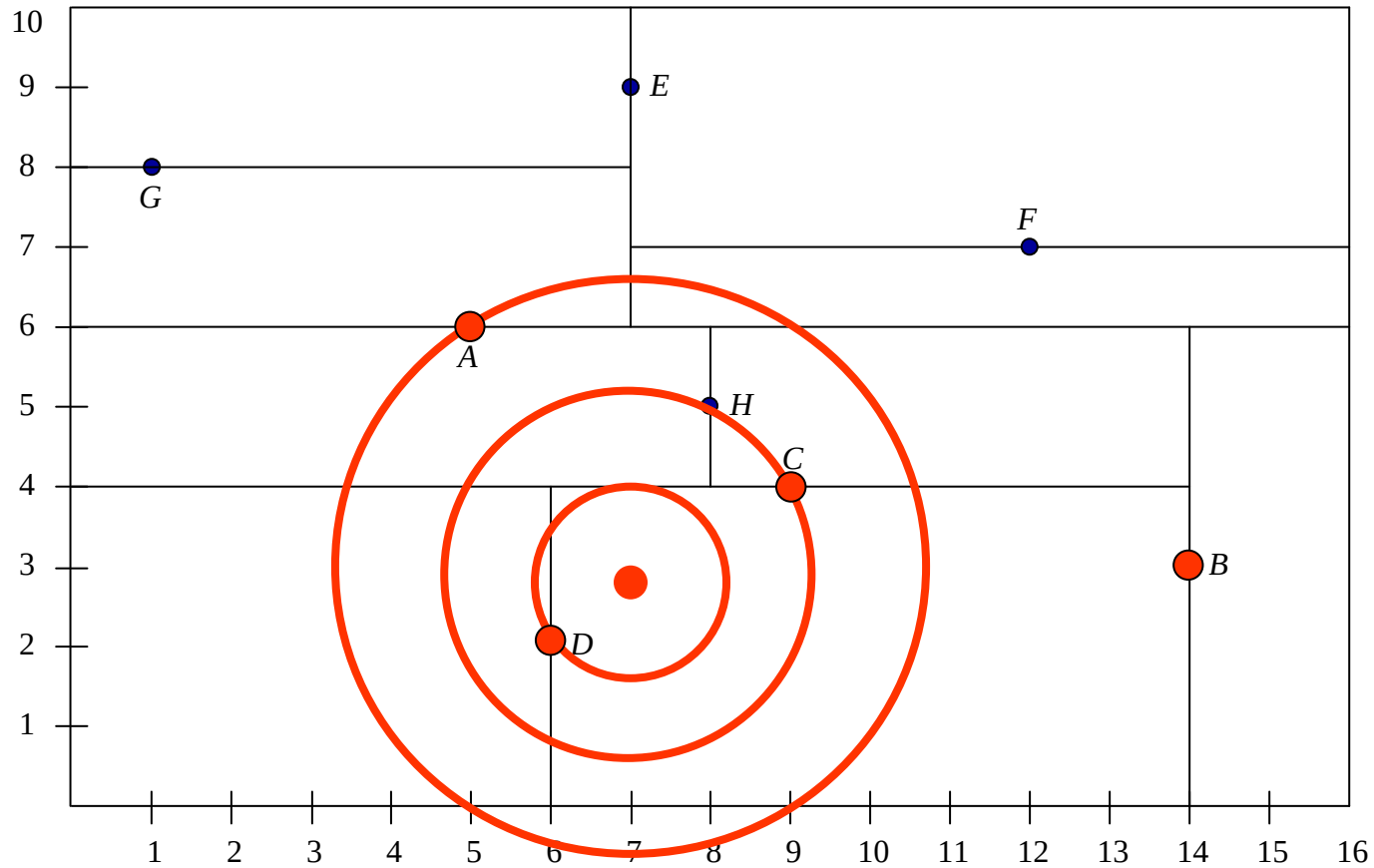
2-d-Baum: Beispiel



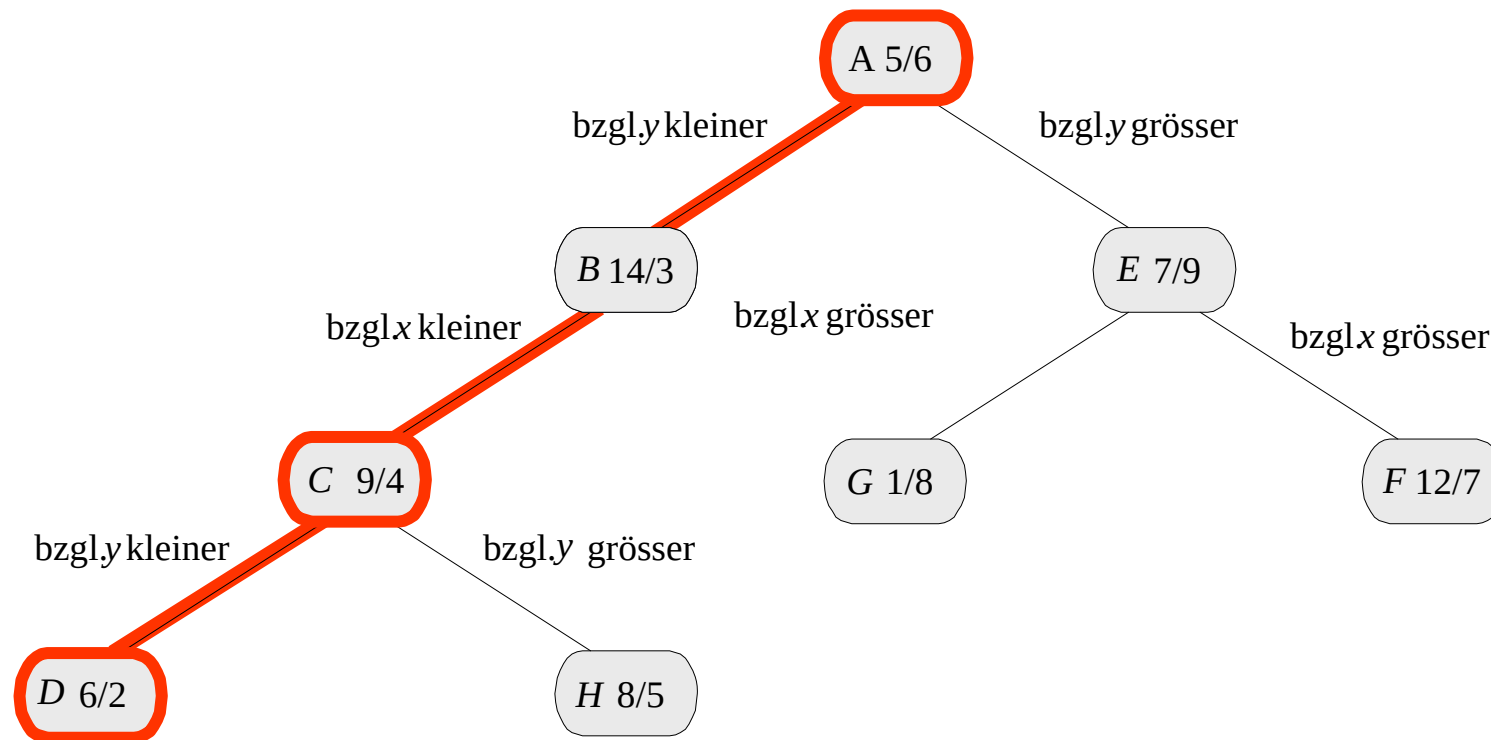
Best Match z.B. finde nächsten Nachbarn zu [7/3]

=> Traversierung mit schrumpfender Rangequery

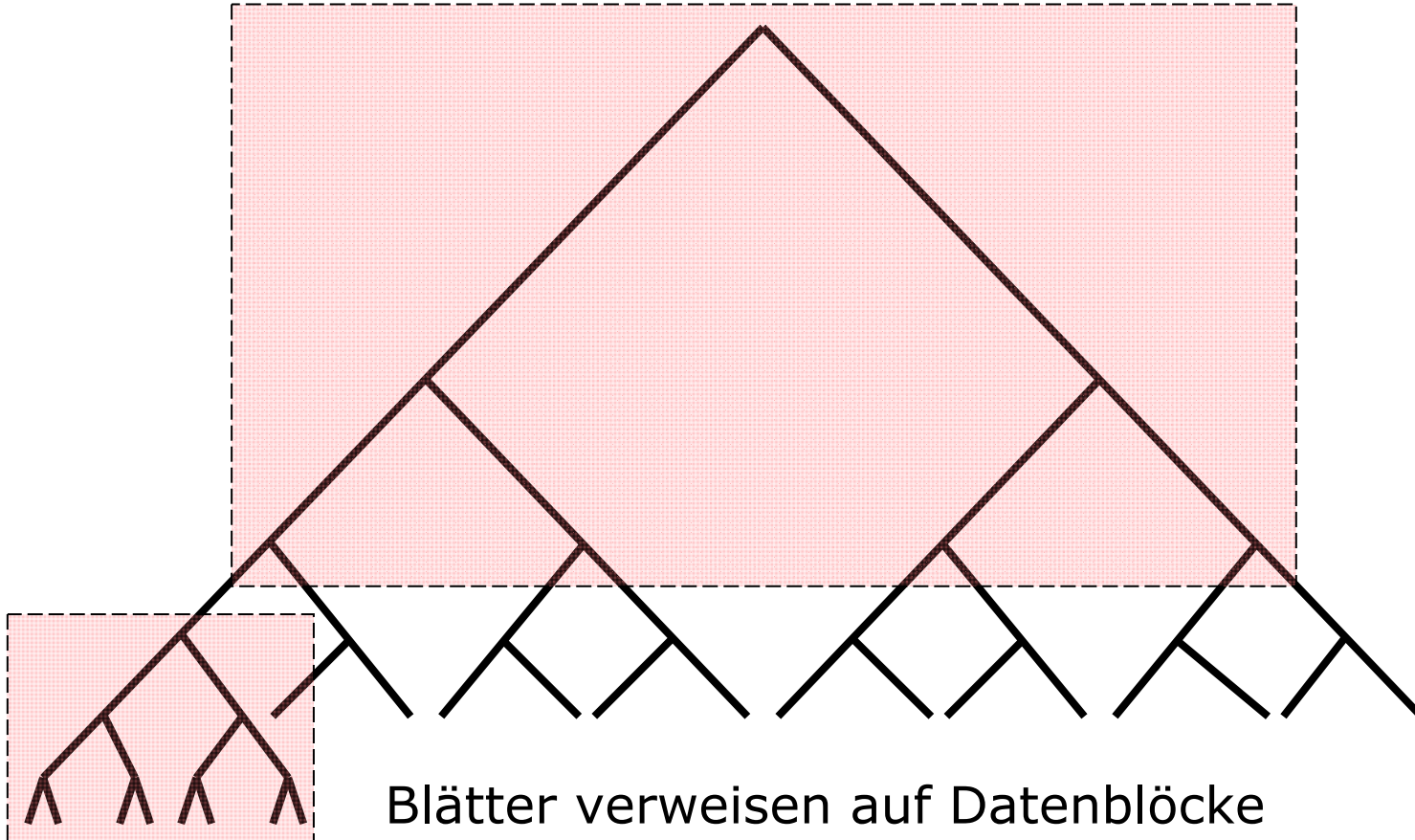
2-d-Baum: Best match



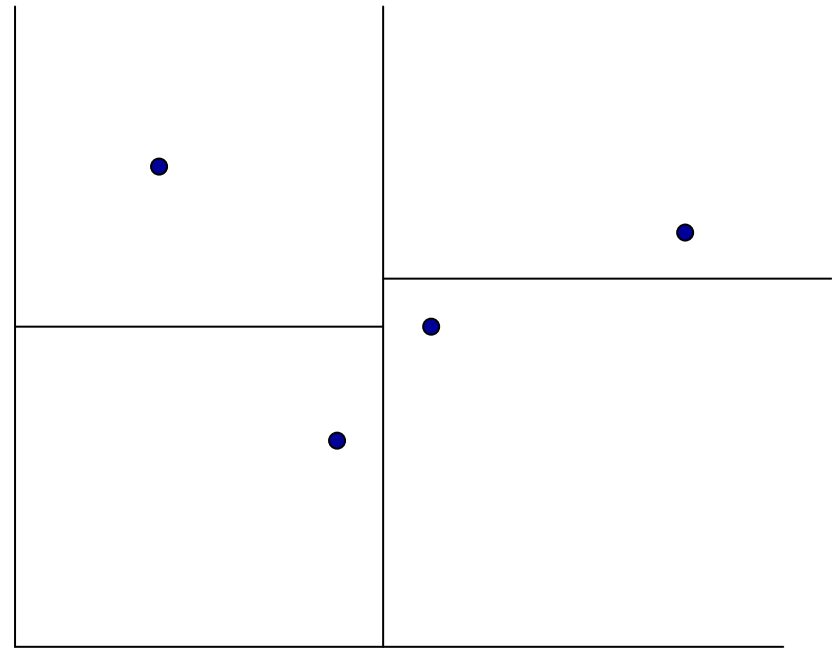
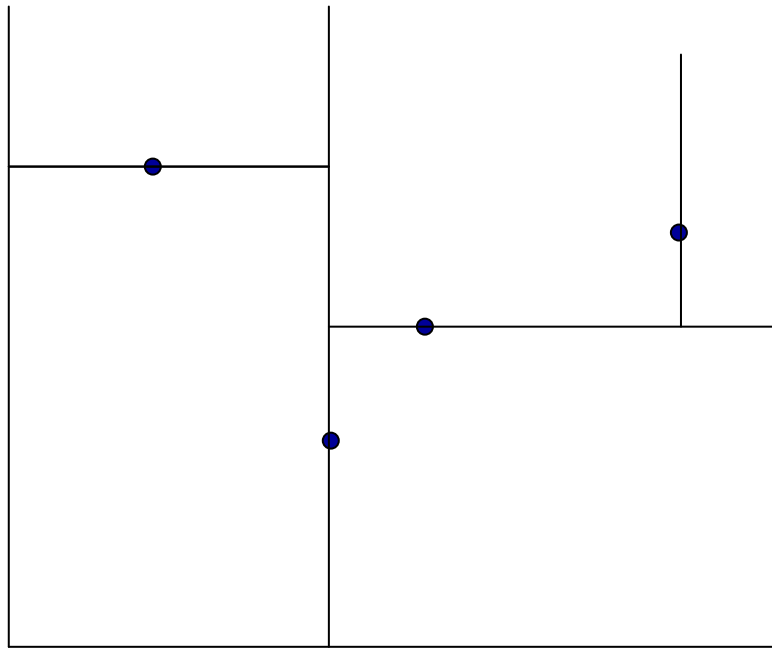
Traversierung für Best Match zu $[7/3]$



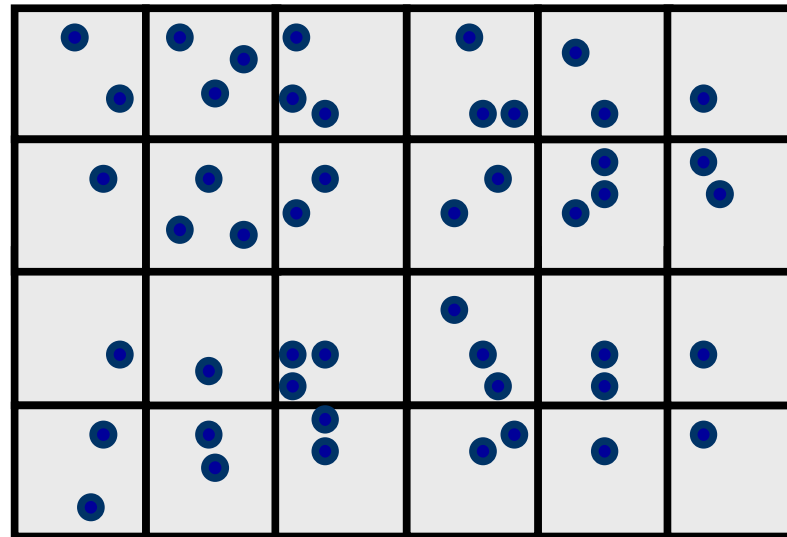
Inhomogene Variante



Partitionierungen des Raumes



Gitterverfahren mit konstanter Gittergröße



Grid File

Alternative zu fester Gittergröße

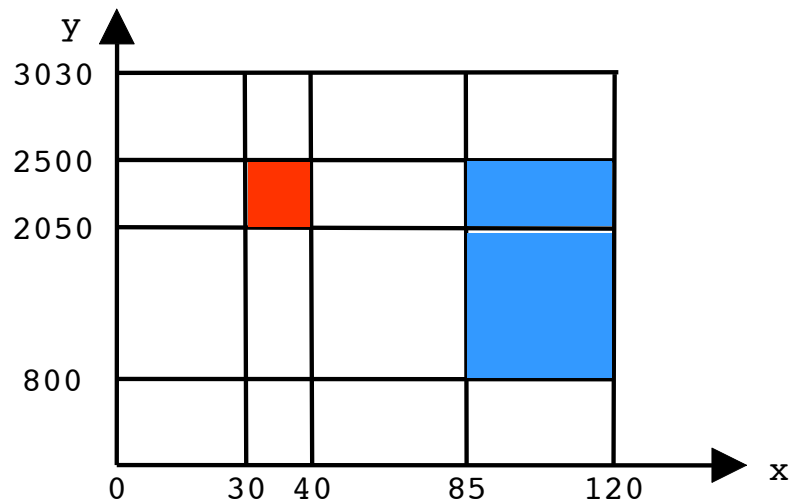
1981 vorgestellt von Hinrichs & Nievergelt

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



x	0	30	40	85	120	$x[i], i=0, \dots, \max_x$
y	0	800	2050	2500	3030	$y[i], i=0, \dots, \max_y$

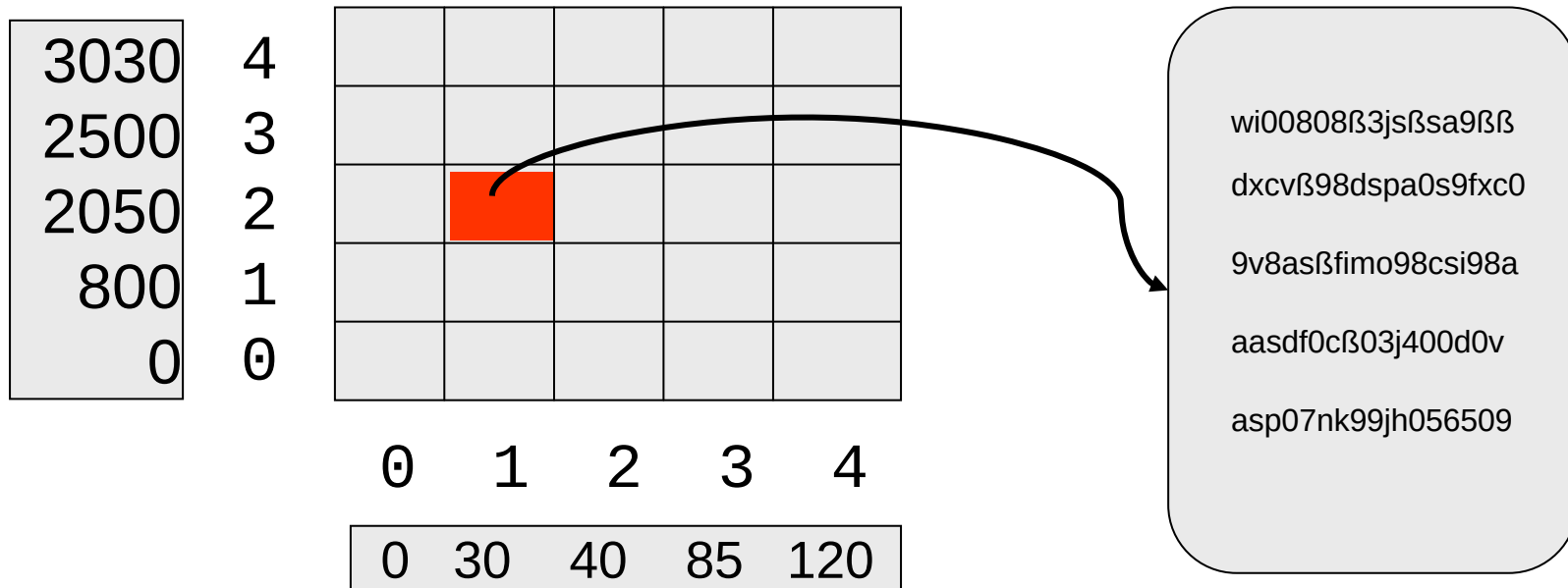
Gitterzelle: rechteckiger Teilbereich

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

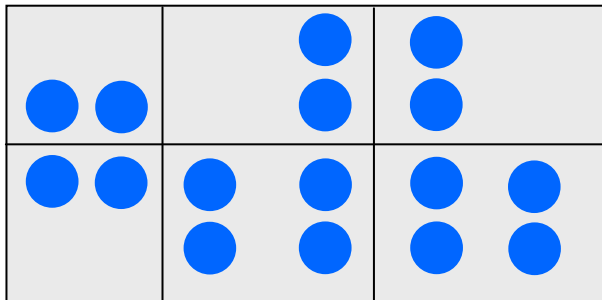
Lade Block mit G[1,2] vom Bucketdirectory

Lade Datenblock

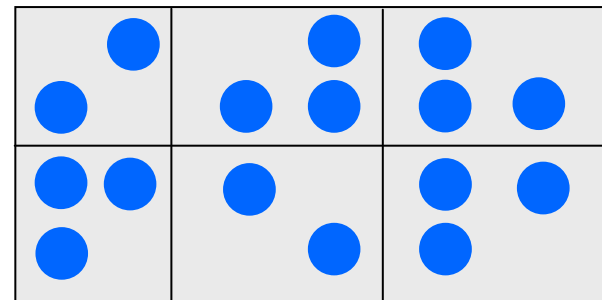
Speichernutzung

Datenblöcke und Directory-Blöcke
nicht immer ausgenutzt:

Beispiel: 4 Datenrecords pro Block:

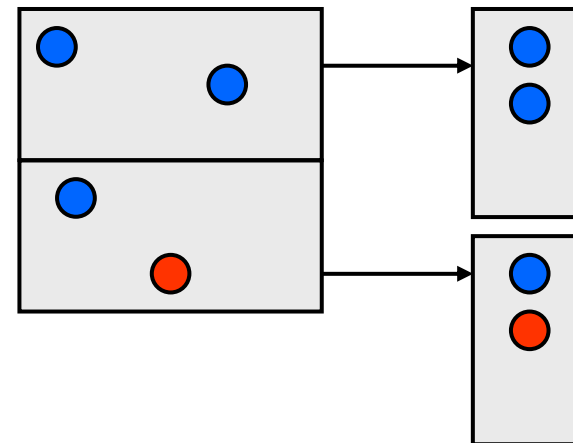
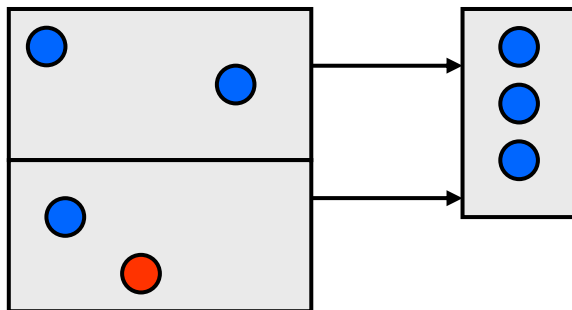
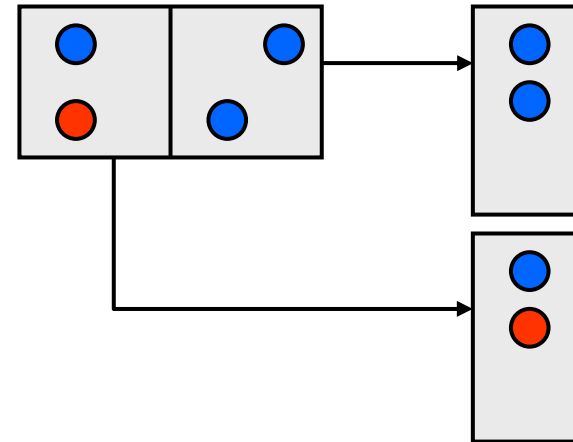
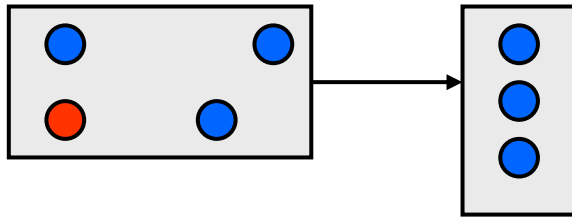


4 volle Datenblöcke
6 Directory-Einträge



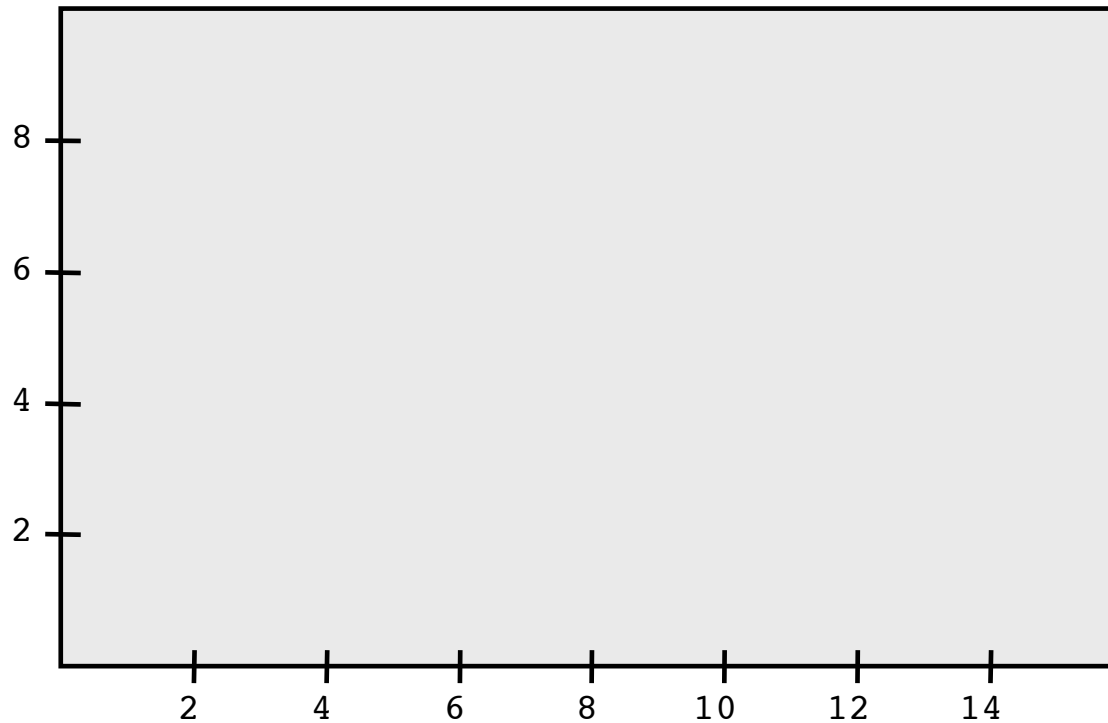
6 Datenblöcke
6 Directory-Einträge

Bucket-Überlauf



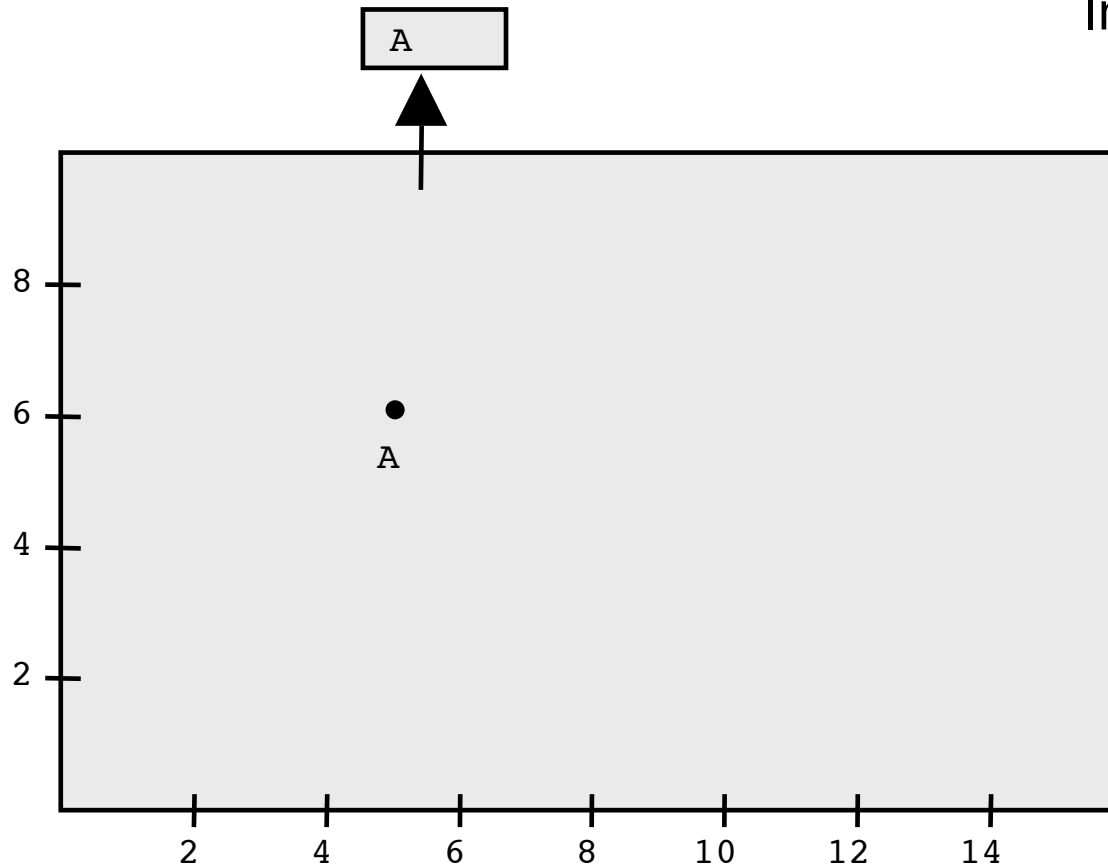
Aufspalten der Regionen

Insert A = [5 / 6]



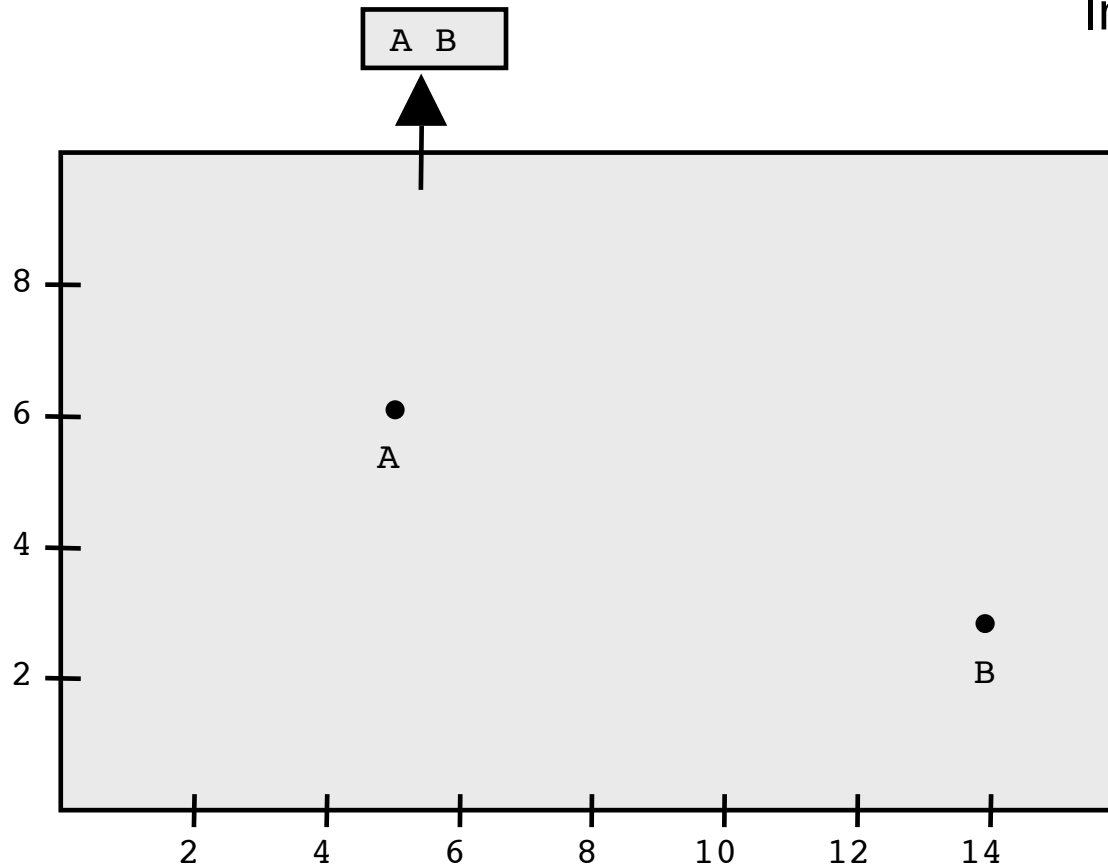
Aufspalten der Regionen

Insert B = [14 / 3]

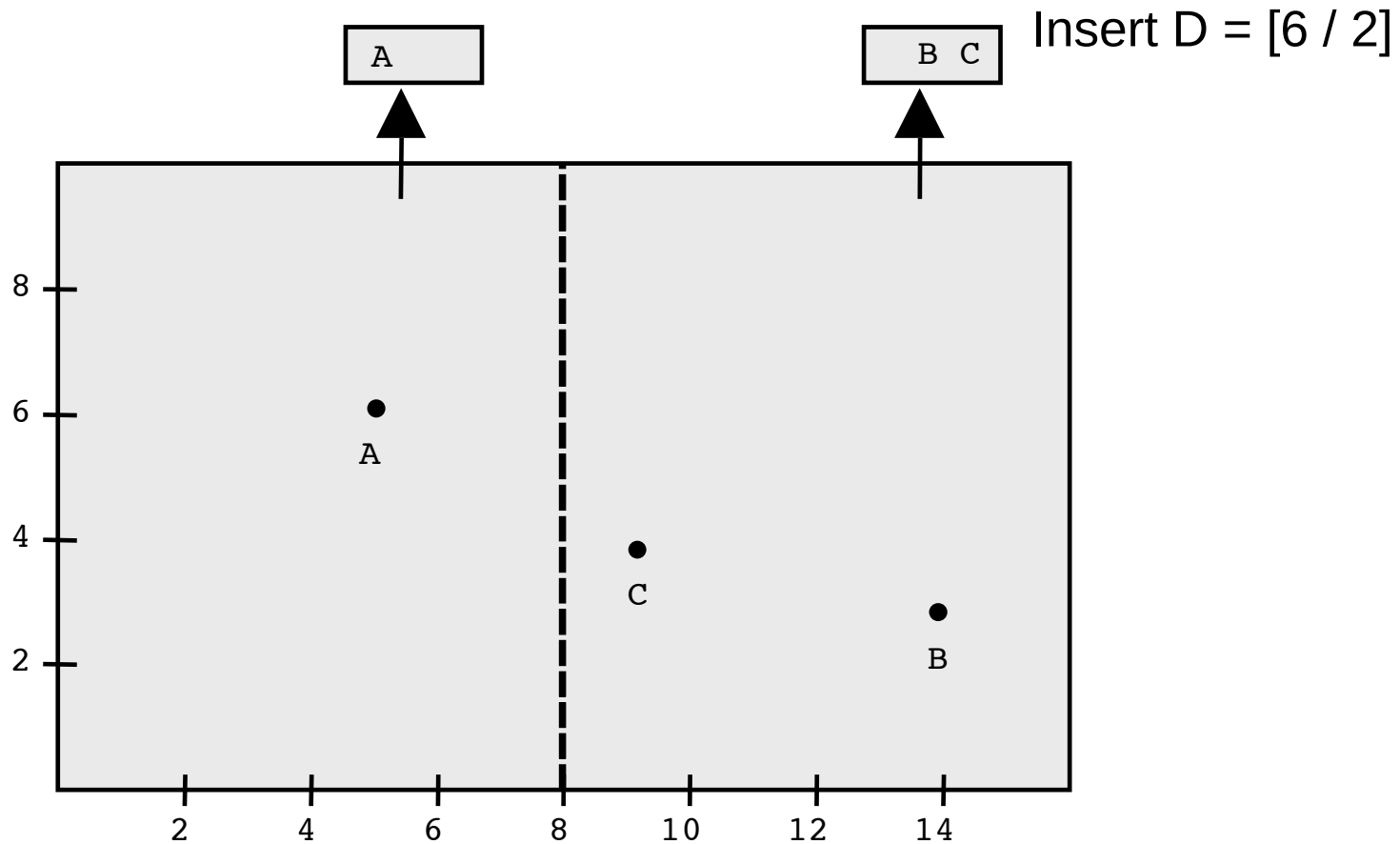


Aufspalten der Regionen

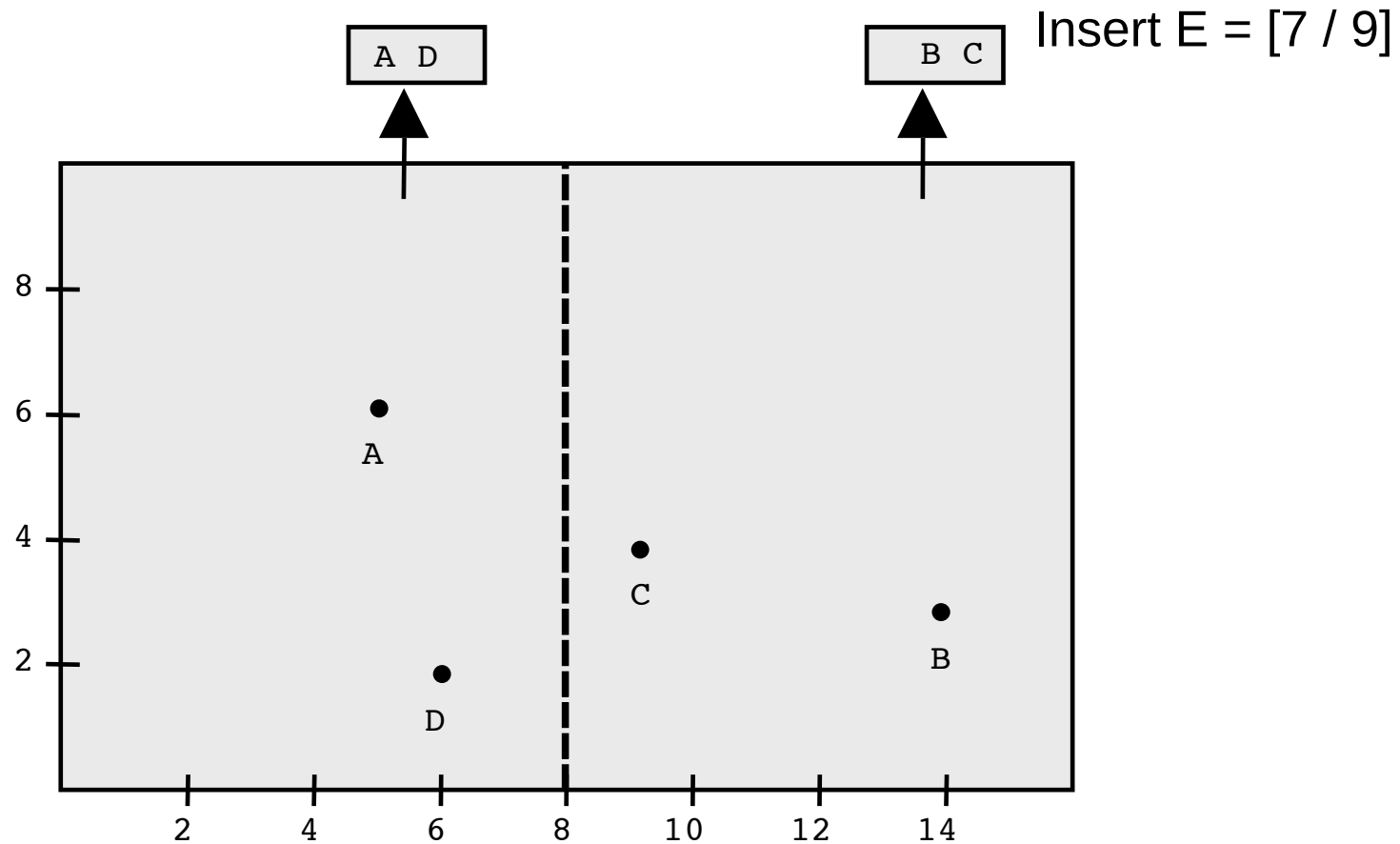
Insert C = [9 / 4]



Aufspalten der Regionen

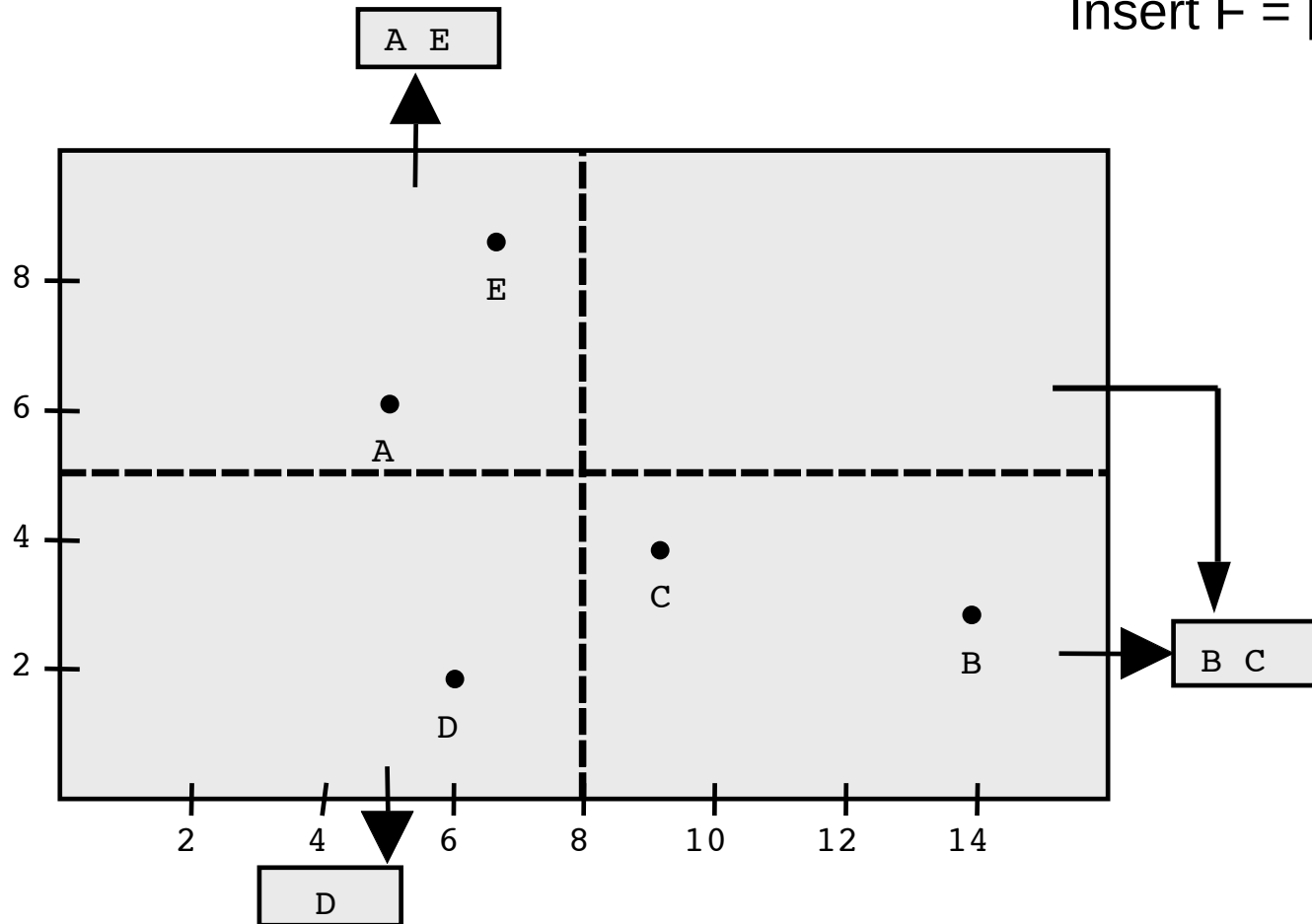


Aufspalten der Regionen



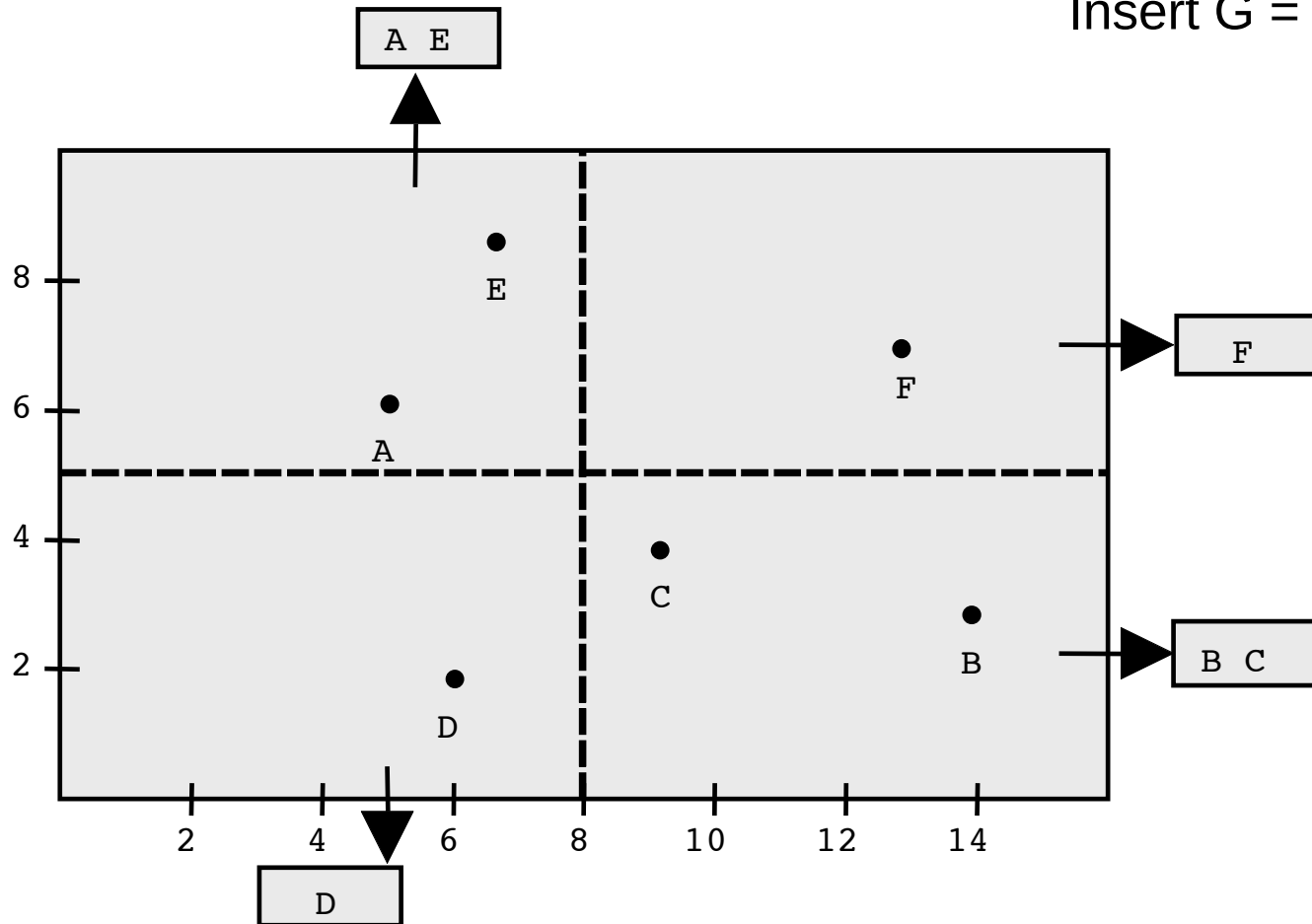
Aufspalten der Regionen

Insert $F = [12 \ / \ 7]$

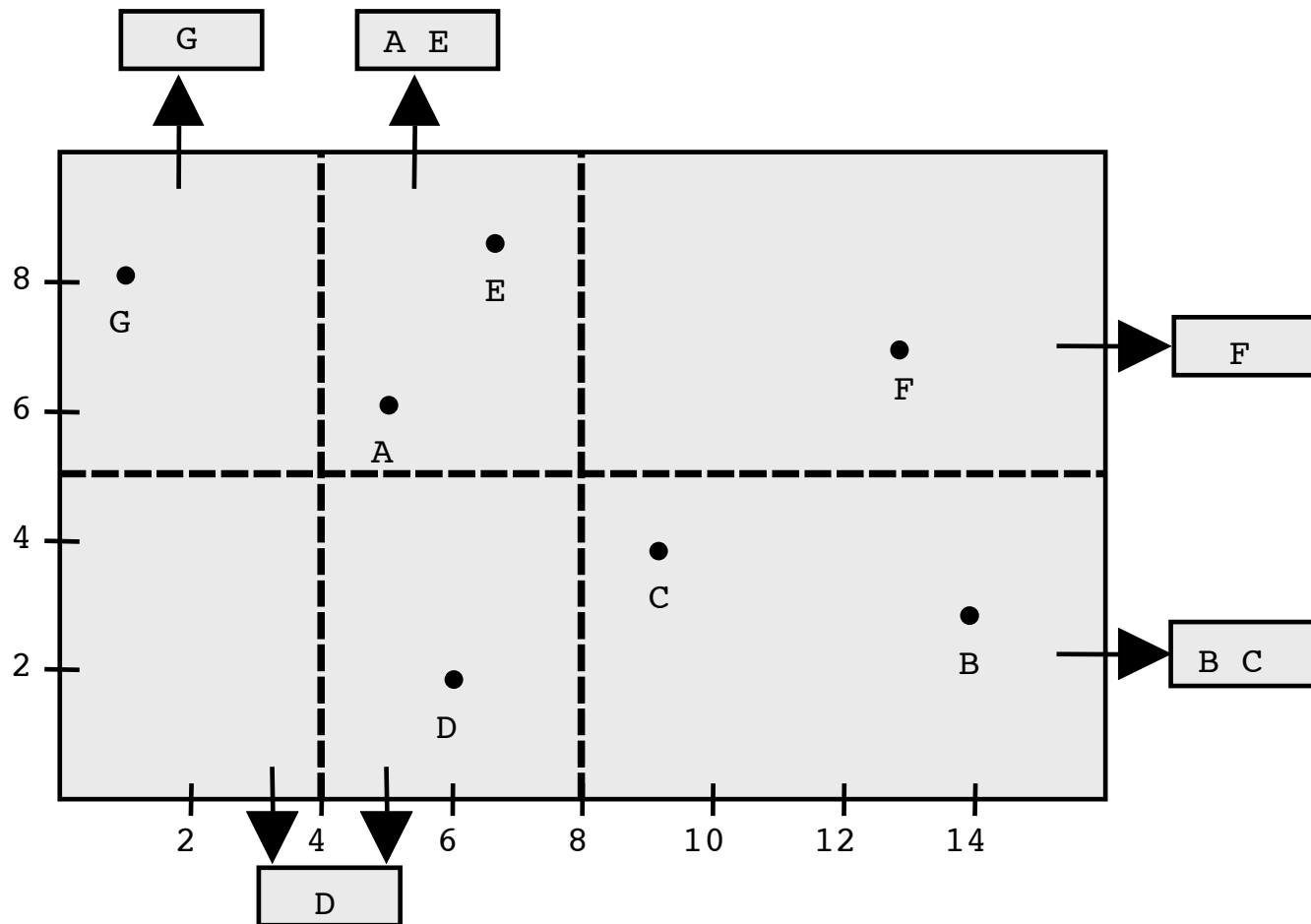


Aufspalten der Regionen

Insert $G = [1 / 8]$



Aufspalten der Regionen



Directory-Zugriff

79	20	21	22	23	24					
64										
63	15	16		18	19					
48										
47	10	11	12	13	14					
32										
31	05	06	07	08	09					
16										
15	00	01	02	03	04					
0										
	0	15	16	31	32	47	48	63	64	79

Indizes:

$$0 \leq i, j \leq 79$$

42 51

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

$$5*3+2=17$$

Adresse

$$a = 16 * (j \bmod 16) + (i \bmod 16)$$

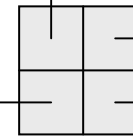
$$16*3+10=58$$

Gridfile Beispiel

2000

1600

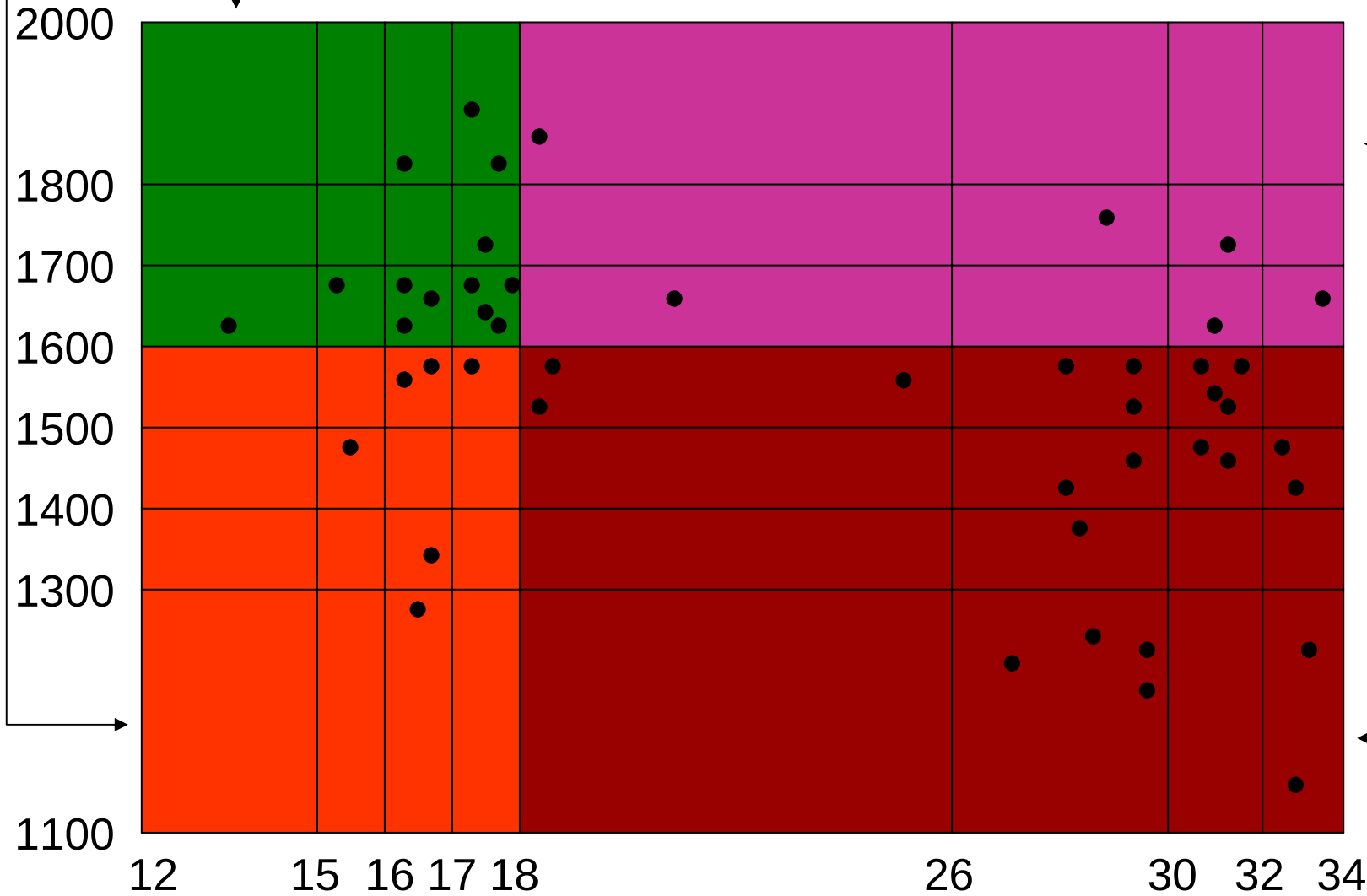
1100



4 Records pro Datenblock

16 Adressen pro Directory-Block

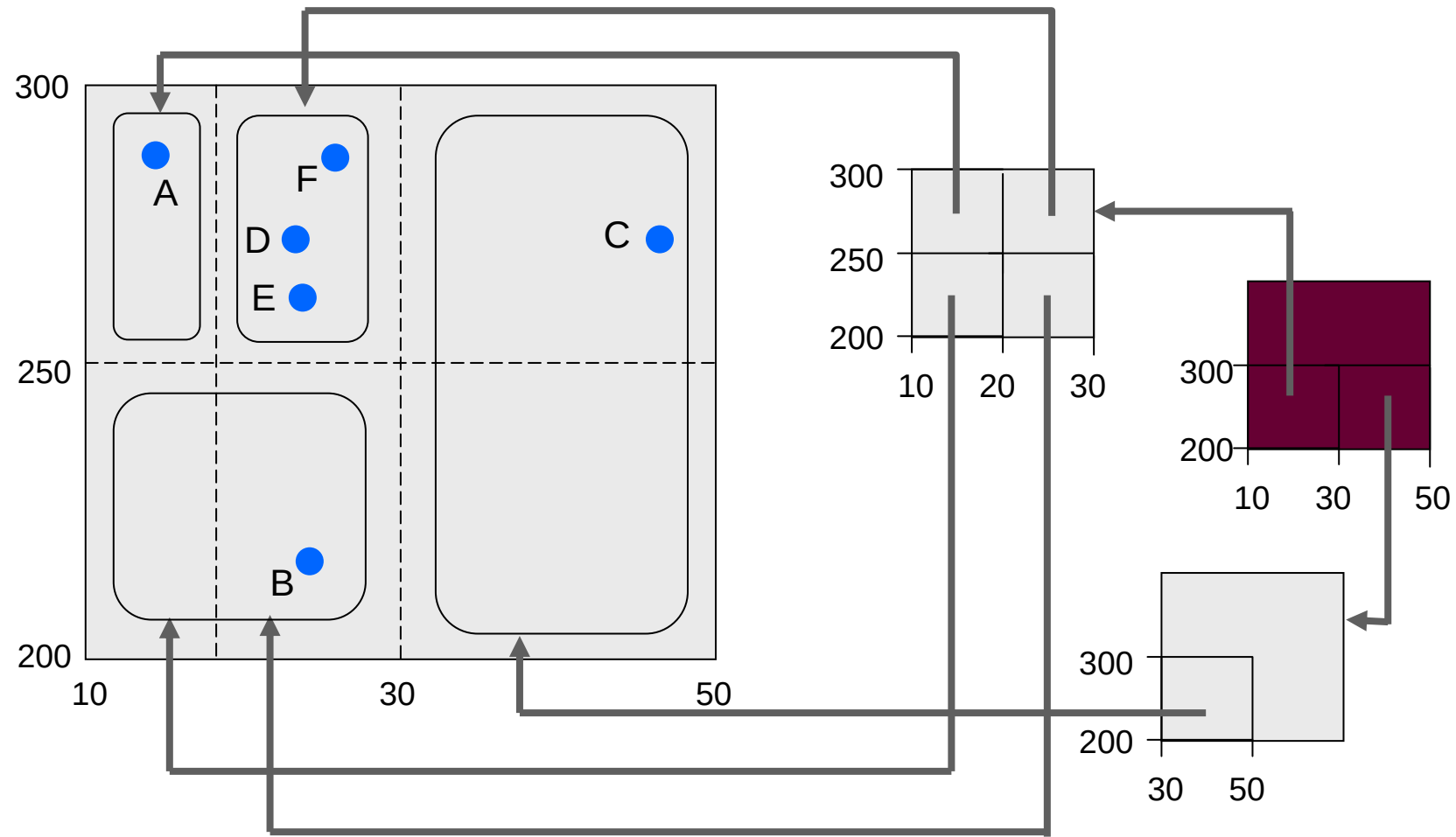
12 18 34



Gridfile: Zahlenbeispiel

Größe eines Datenrecords:	100 Byte
Zahl der Datenrecords:	300.000
maximale Zahl der Datenrecords pro Block	10
mittlere Zahl der Datenrecords pro Block:	7.5
Zahl der Datenblöcke:	40.000
maximale Zahl von Adressen pro Directoryblock:	256
mittlere Zahl von Adressen pro Directoryblock:	175
Zahl der Directoryblöcke:	225
Größe des Root-Directory:	15 x 15

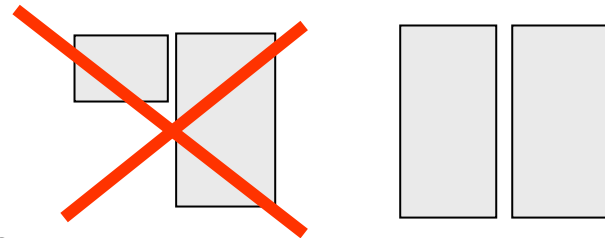
Dynamik des Grid Directory



Vereinigung von Regionen

bei zu geringer Bucketauslastung:

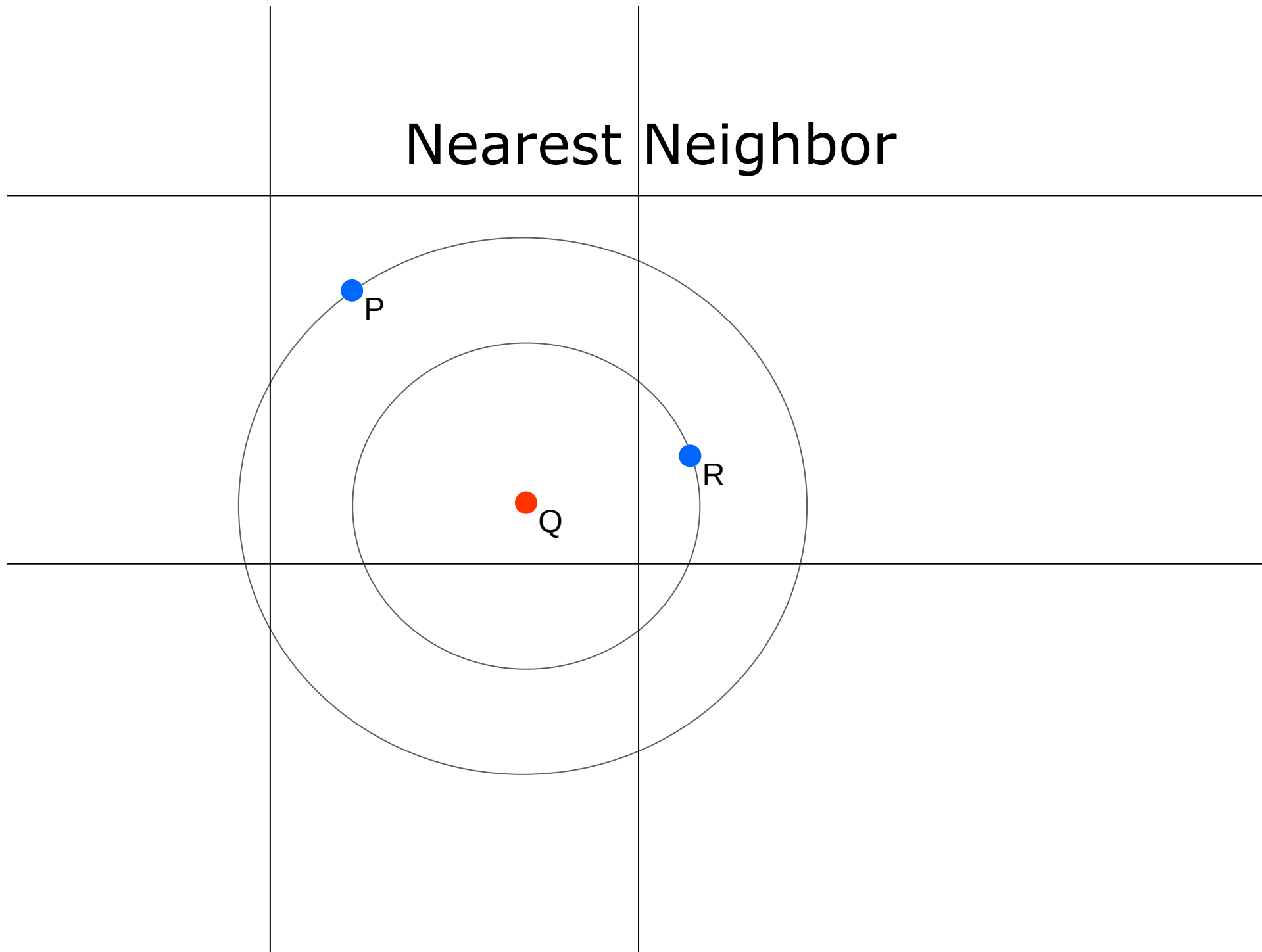
- Benachbarte Regionen vereinigen
- Vereinigung muß Rechteck ergeben



Vereinigung wird ausgelöst

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

Nearest Neighbor



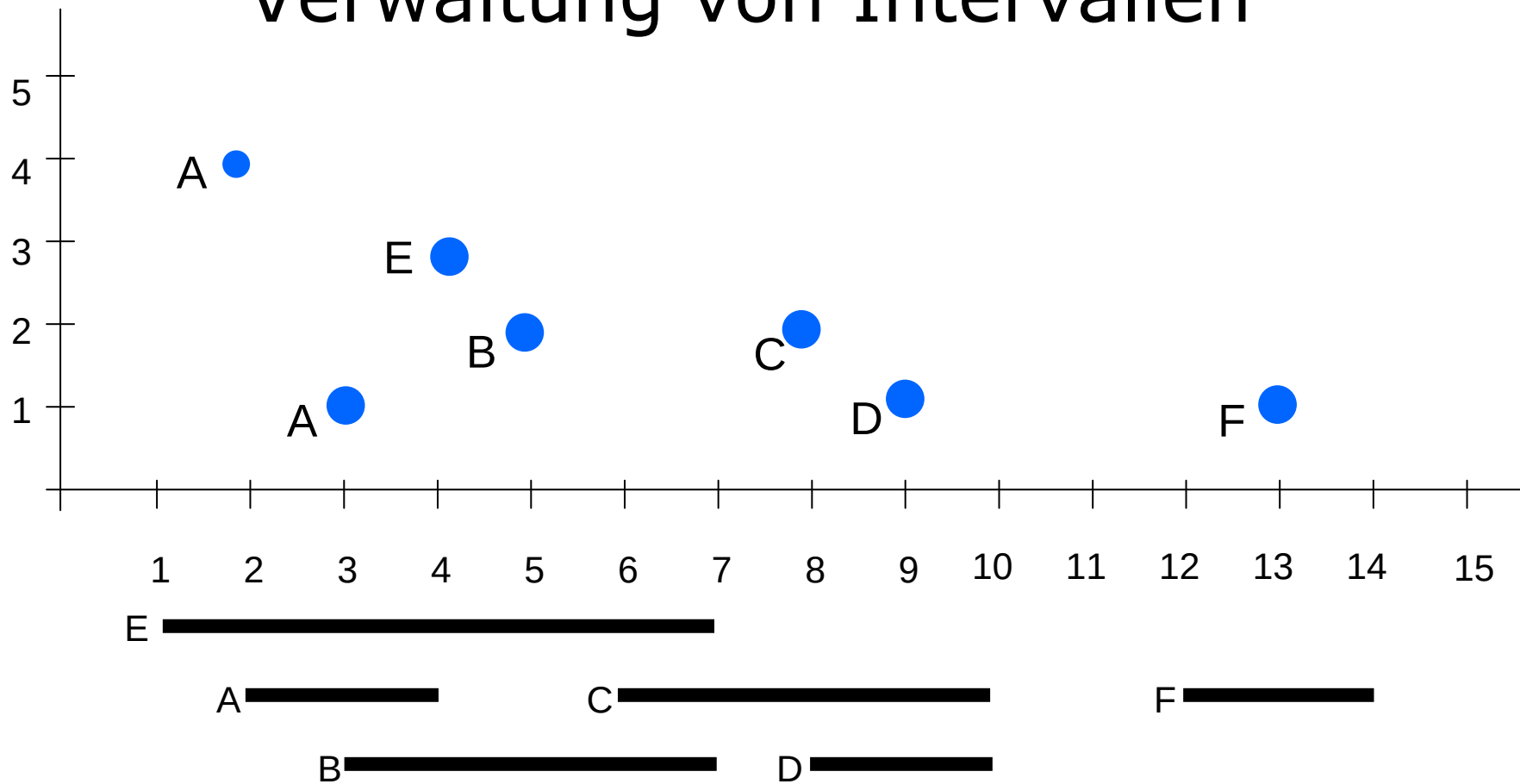
Verwaltung von geometrischen Objekten

Grid File unterstützt Range-Query

bisher: k Attribute

Jetzt: k-dimensionale Punkte

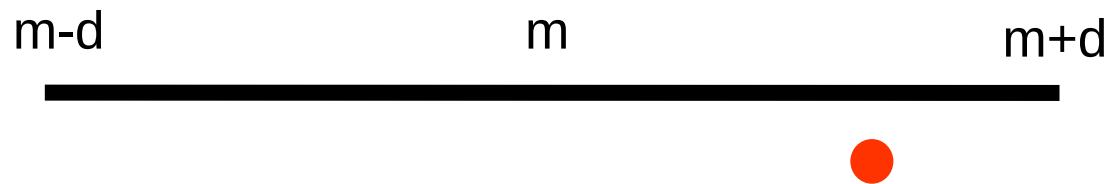
Verwaltung von Intervallen



[Anfang / Ende]

Besser: [Mittelpunkt / halbe Länge]

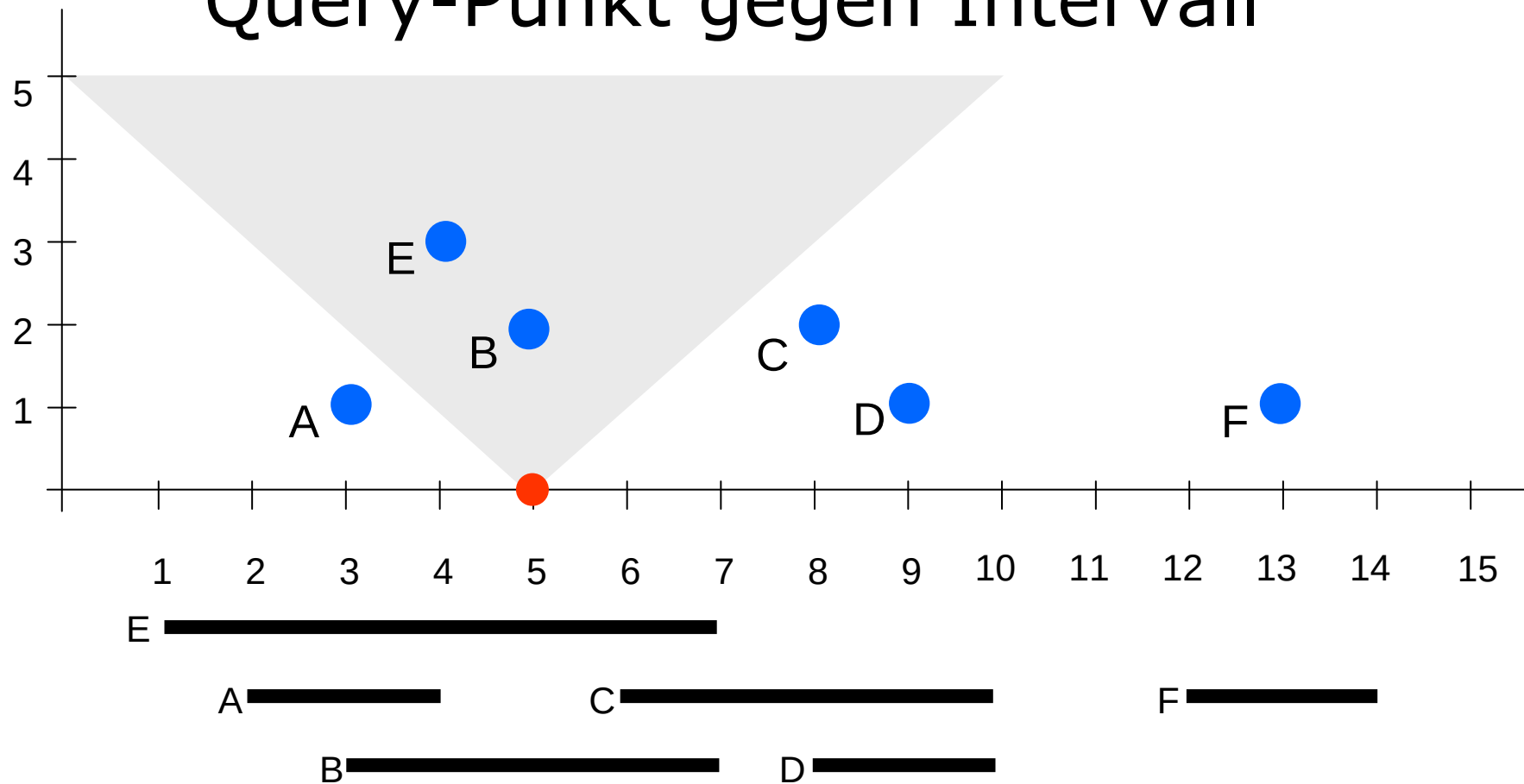
Query-Punkt gegen Intervall



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

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

Query-Punkt gegen Intervall



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

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

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

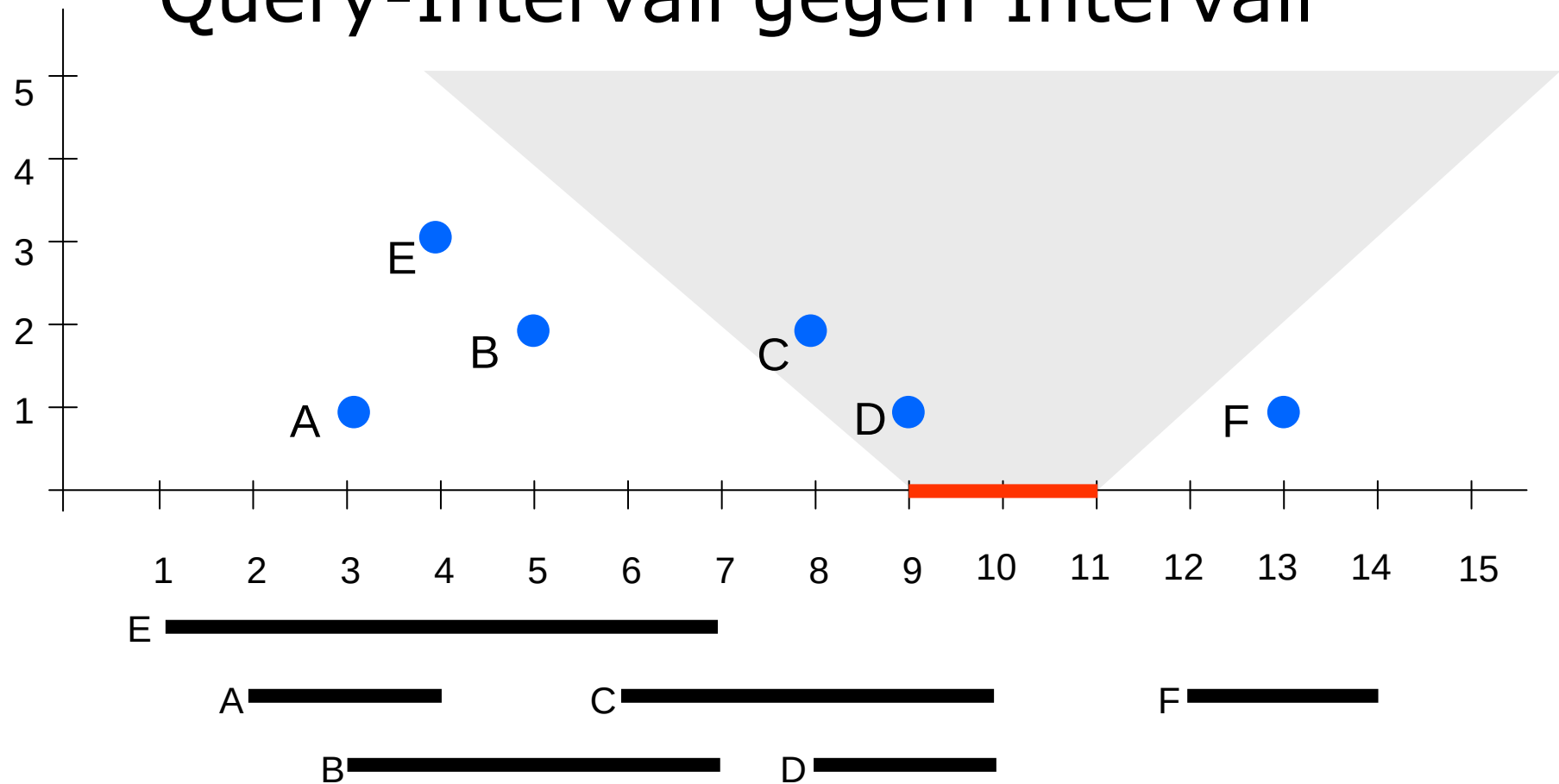
Query-Intervall gegen Intervall



Intervall mit Mitte s und halber Länge t schneidet
Intervall mit Mitte m und halber Länge d $\Leftrightarrow$

$$m-d \leq s+t \text{ und } s-t \leq m+d$$

Query-Intervall gegen Intervall



Intervall mit Mitte s und halber Länge t schneidet
 Intervall mit Mitte m und halber Länge $d \Leftrightarrow$

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

Query-Rechteck gegen Rechteck

Stelle Rechteck durch vierdimensionalen Punkt dar

