# Motivation

- Problem:
  - select x from Y
    where z = 'k';

- Plan:

  select z = 'k'
  ↑
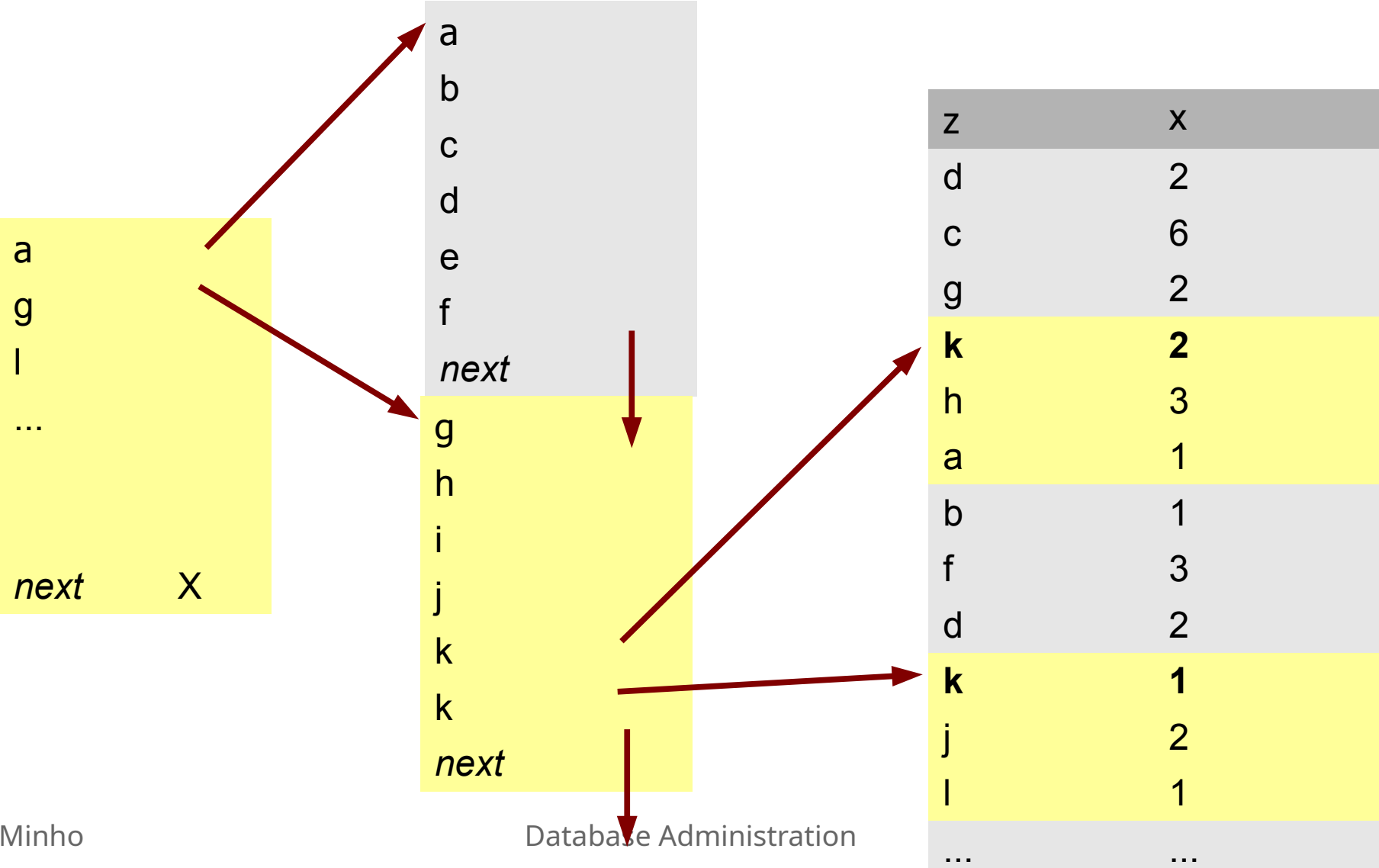  scan Y

- Cost?

| z | x |
|---|---|
| d | 2 |
| c | 6 |
| g | 2 |
| **k** | **2** |
| h | 3 |
| a | 1 |
| b | 1 |
| f | 3 |
| d | 2 |
| **k** | **1** |
| j | 2 |
| l | 1 |
| ... | ... |

# Index

- Makes it easy to find pages containing interesting data

- Smaller that data
  – Fits in memory?

- Efficient look-up:
  – Identity (=)
  – Ranges
  – LIKE
  – …

| z | x |
|---|---|
| d | 2 |
| c | 6 |
| g | 2 |
| **k** | **2** |
| h | 3 |
| a | 1 |
| b | 1 |
| f | 3 |
| d | 2 |
| **k** | **1** |
| j | 2 |
| l | 1 |
| … | … |

# B-Tree

# B-Tree

- Insert:
  - If free entry not available, split leaf
    - Recursively insert new leaf in upper layer
    - Tree grows towards the root!
  - Add entry to leaf

- Delete:
  - Remove entry from leaf
  - If enough space available, collapse leafs
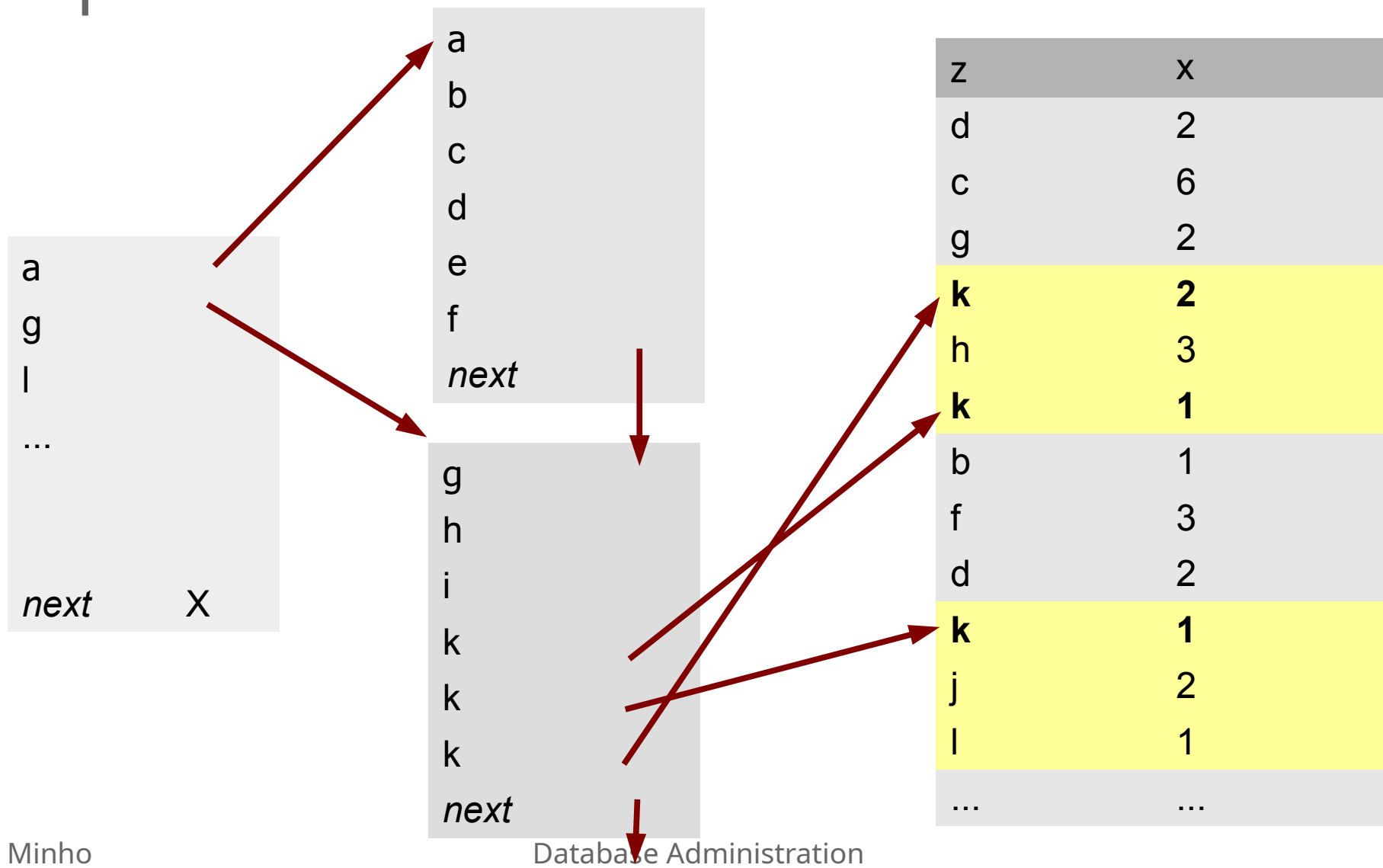    - Recursively delete leaf in upper layer

# B-Tree

- Desirable characteristics:
  - Balanced
  - Log(n) depth
  - Fit for block I/O

- Supports:
  - Identity look up
  - Range queries / Ordered scan
  - Updates

# Composite indexes

- Index on (X,Y):
  - Answers equality and interval on X
  - Answers equality on X and interval on Y
- Index on expression, e.g. X+Y
  - Answers equality and interval on X+Y

# Dispersion

| z | x |
|---|---|
| d | 2 |
| c | 6 |
| g | 2 |
| **k** | **2** |
| h | 3 |
| **k** | **1** |
| b | 1 |
| f | 3 |
| d | 2 |
| **k** | **1** |
| j | 2 |
| l | 1 |
| ... | ... |

a
b
c
d
e
f
*next*

a
g
l
...
*next*     X

g
h
i
k
k
k
*next*

# Clustered indexes

- Problem:
  - #blocks >> (#records / records per block)
  - Read each block multiple times

- A clustered index:
  - Records are (roughly) sorted according to the index
    - No sorting within a block is needed
  - Free space may be kept for insertions

# Multi-criteria filtering

- ## Problem:
  - select x from Y
    where z = 'k'
    and x = 4;

Index on Z

Index on X

| Z | X |
|---|---|
| d | 2 |
| c | 6 |
| g | **4** |
| h | 2 |
| h | 3 |
| **k** | **4** |
| b | 1 |
| f | 3 |
| d | 2 |
| **k** | 1 |
| j | 2 |
| l | 1 |
| ... | ... |

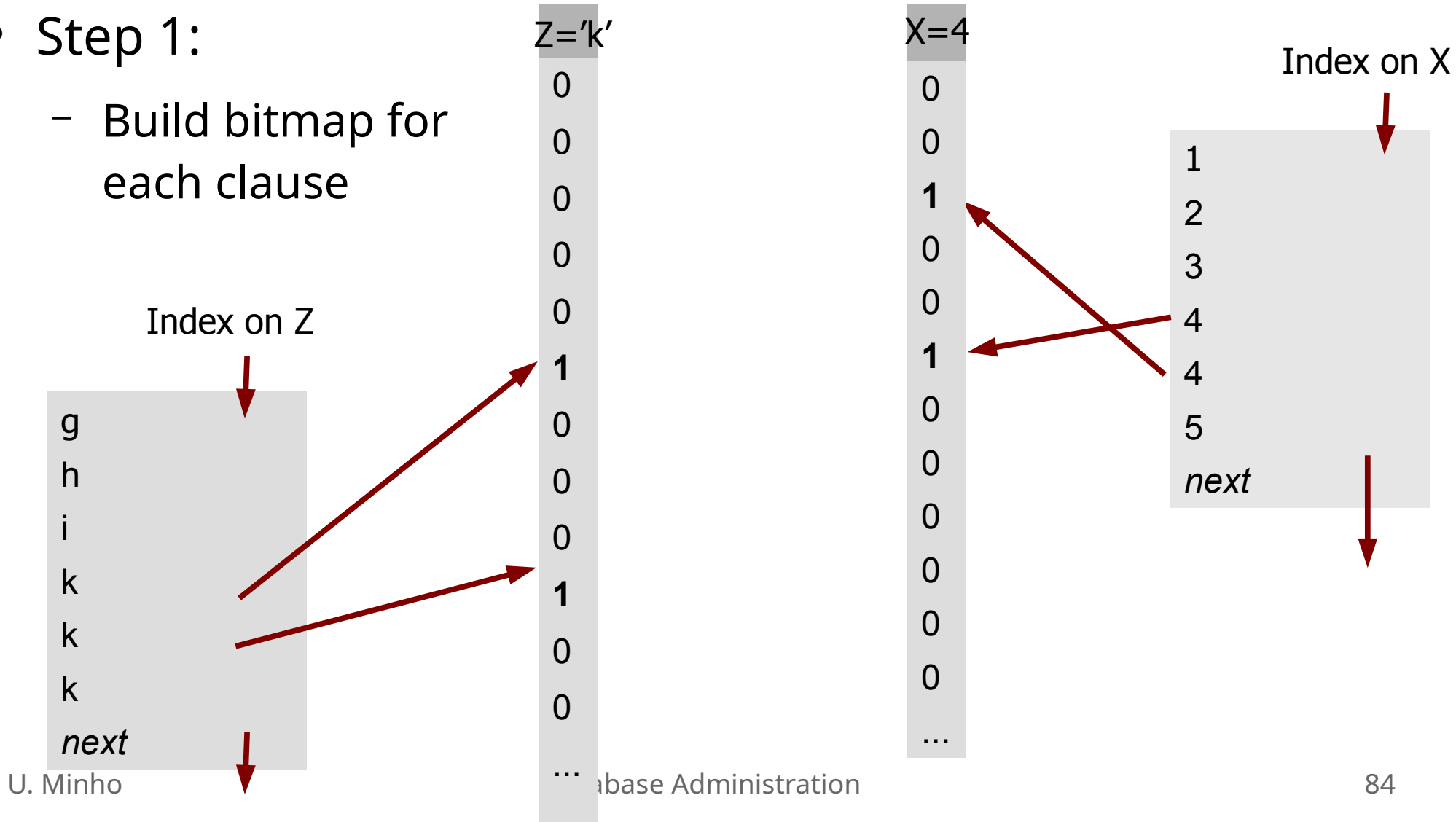| |
|---|
| g |
| h |
| i |
| k |
| k |
| k |
| *next* |

| |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| 4 |
| 5 |
| *next* |

# Multi-criteria filtering

- Sequential scan

- Either X or Z plus scan

- Composite index on (Z,X):

  – Often, many columns and combinations

- Typical examples:

  – Search with a combination of features

  – Hotel booking, real estate, on-line shopping, ...

- What if "or" instead of "and"?

  – (T2 or T3) and in Braga

# Bitmap indexes

- ## Step 1:
  - Build bitmap for each clause

Index on Z

| g |
|---|
| h |
| i |
| k |
| k |
| k |
| *next* |

Z='k'

| 0 |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 0 |
| **1** |
| 0 |
| 0 |
| 0 |
| **1** |
| 0 |
| 0 |
| ... |

X=4

| 0 |
|---|
| 0 |
| **1** |
| 0 |
| 0 |
| **1** |
| 0 |
| 0 |
| 0 |
| 0 |
| 0 |
| 0 |
| ... |

Index on X

| 1 |
|---|
| 2 |
| 3 |
| 4 |
| 4 |
| 5 |
| *next* |

# Bitmap indexes

- Step 2:
  - Combine with logical operators

| Z='k' | ... and ... | X=4 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | **0** ← | **1** |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| **1** → | **1** ← | **1** |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| **1** → | **0** | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| ... | ... | ... |
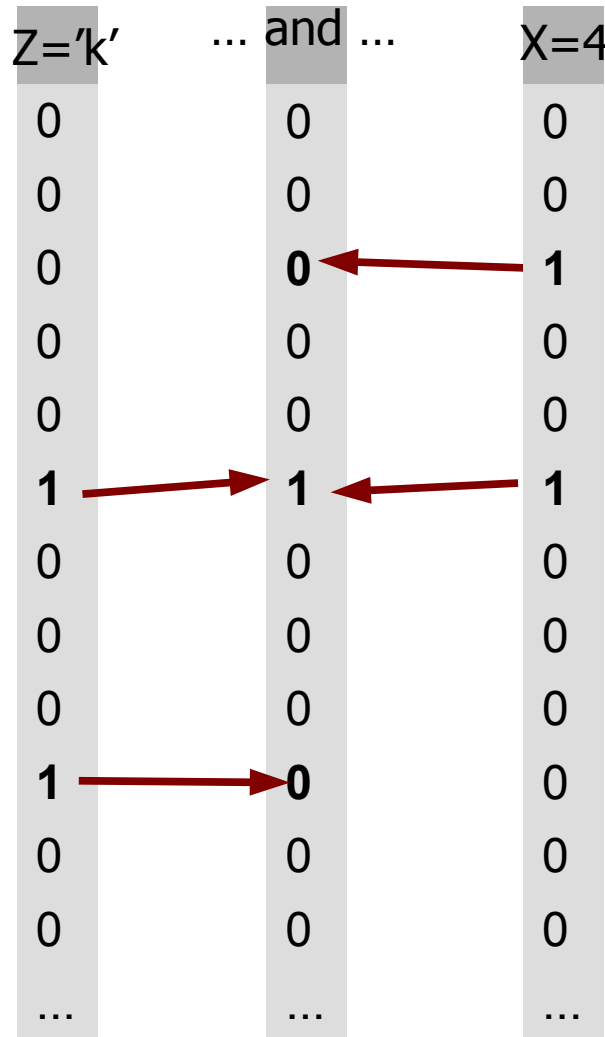
# Bitmap indexes

- ## Step 3:
  - Traverse bitmap and read table

| ... and ... | Z | X |
|---|---|---|
| 0 | d | 2 |
| 0 | c | 6 |
| **0** | g | **4** |
| 0 | h | 2 |
| 0 | h | 3 |
| **1** → | **k** | **4** |
| 0 | b | 1 |
| 0 | f | 3 |
| 0 | d | 2 |
| **0** | **k** | 1 |
| 0 | j | 2 |
| 0 | l | 1 |
| ... | ... | ... |
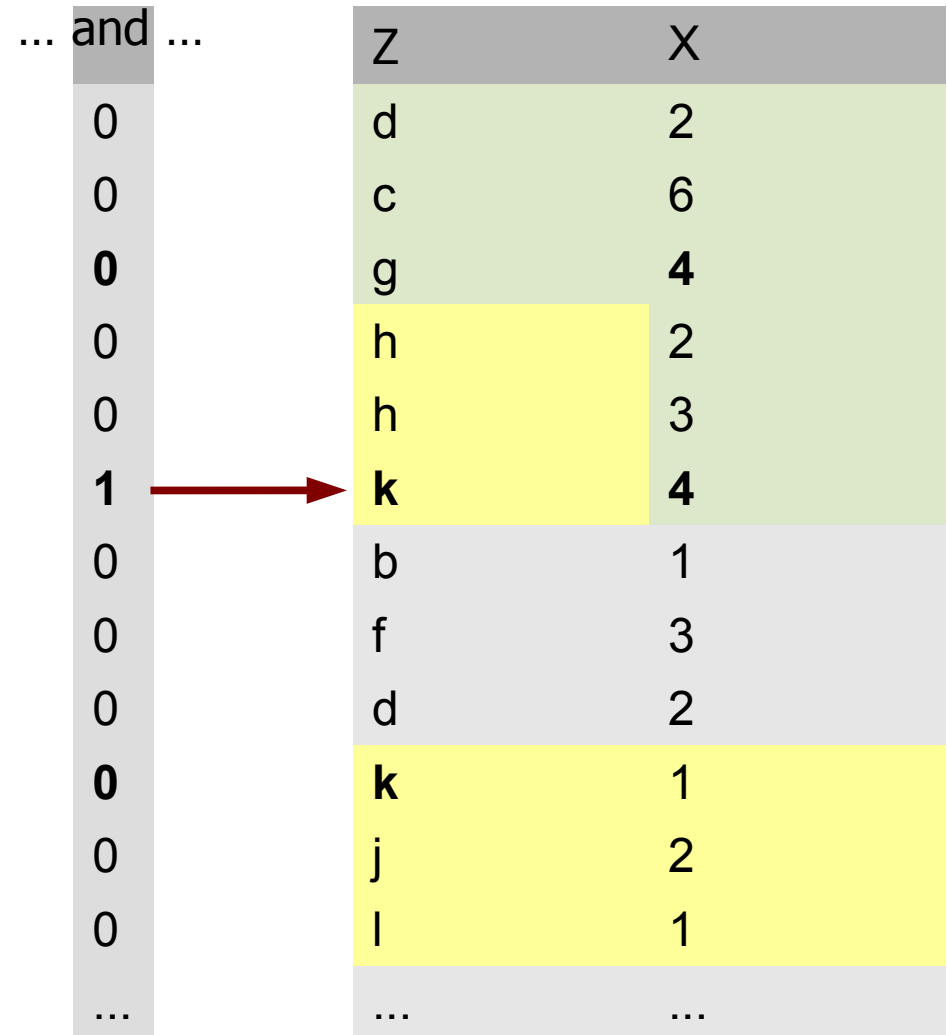
# Motivation

- Assumptions:
  - Several TB of data
  - ~50%, y=1
  - ~50%, y=2
  - a few, y=3

| z | y |
|---|---|
| **d** | **1** |
| c | 2 |
| **g** | **1** |
| k | 2 |
| h | 3 |
| **a** | **1** |
| **b** | **1** |
| f | 2 |
| d | 2 |
| **k** | **1** |
| j | 2 |
| **l** | **1** |
| ... | ... |

# Motivation

- Problem:

  - select count(*) from X
    where y = 1;

- Possible plans:

count(*)

↑

select y = 1

↑

scan X

count(*)

↑

index scan X
(y = 1)

- Cost?

| z | y |
|---|---|
| **d** | **1** |
| c | 2 |
| **g** | **1** |
| k | 2 |
| h | 3 |
| **a** | **1** |
| **b** | **1** |
| f | 2 |
| d | 2 |
| **k** | **1** |
| j | 2 |
| **l** | **1** |
| ... | ... |

# Motivation

X

| z | y |
|---|---|
| **d** | **1** |
| c | 2 |
| **g** | **1** |
| k | 2 |
| h | 3 |
| **a** | **1** |
| **b** | **1** |
| f | 2 |
| d | 2 |
| **k** | **1** |
| j | 2 |
| **l** | **1** |
| ... | ... |

- Keep results cached

- Update when needed

select y, count(*) from X group by y

| y | count |
|---|-------|
| **1** | **773647263** |
| 2 | 765732332 |
| 3 | 1 |

# Views

View usage
{
select count(*) from V

count(*)
↑
V
}

View definition
{
select y = 1
↑
scan X

create view V as select * from X where y=1
}

⇒

count(*)
↑
select y = 1
↑
scan X
}
Plan executed

# Materialized views

View usage
{
select count(*) from V

count(*)
↑
V
} Plan executed

View definition
{
materialize V
↑
select y = 1
↑
scan X

create materialized view V as select * from X where y=1
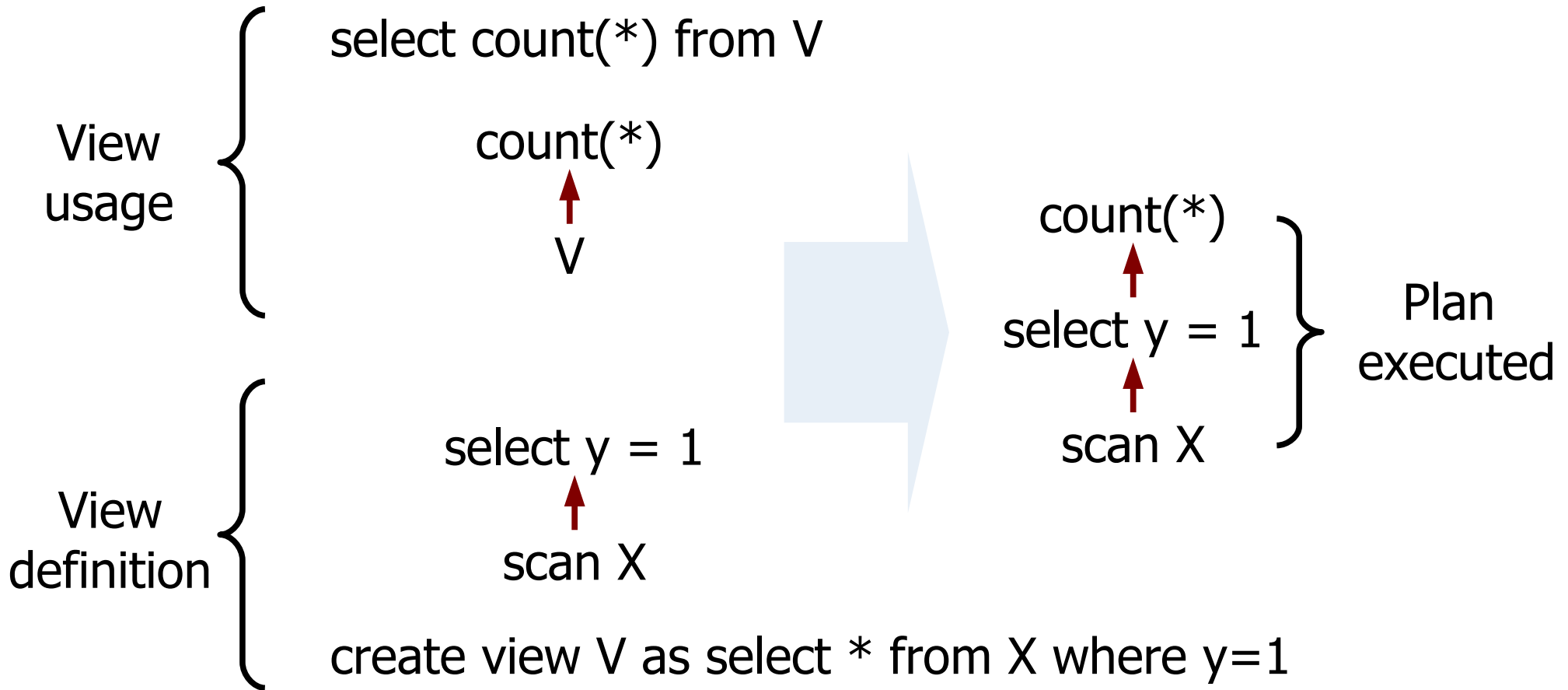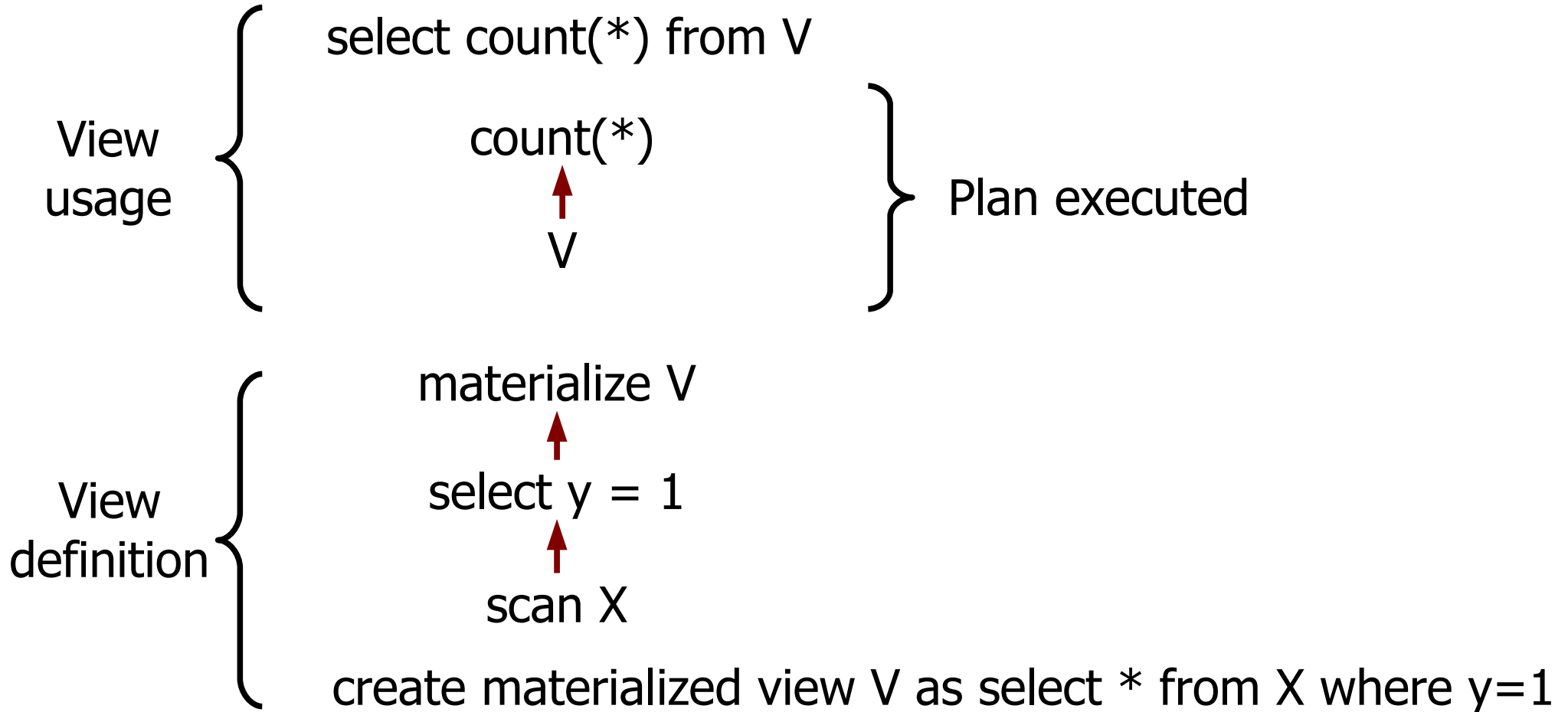
# Maintaining materialized views

- Periodically run the query and update the view

- Update the view when data changes

# DIY Materialized Views

- Updating with AFTER triggers:

```
-- Create view
select sum(value) into mv_sum_items from items;

-- Update view
create function upd_sum_items() returns trigger as '
    BEGIN
        update mv_sum_items set sum = sum + new.value - old.value;
        return new;
    END
' language 'plpgsql';

create trigger upd_sum
    after update on items
    for each row execute procedure upd_sum_items();
```

# Using materialized views

- Automatically used by the planner:
  - Indexed views in MS SQL Server
- Used explicitly in queries:
  - Materialized views in Oracle
  - DIY materialized views everywhere
  - Developer tip:
    - Using views allows the DBA to select which ones to materialize

# Conclusions

- Indexes and mat. views = Redundancy!

- Trade-off between:

  - Complexity of operations

  and:

  - Disk space used

  - Usage of main memory

  - Effort when updating

- Usefulness depends on workload mix