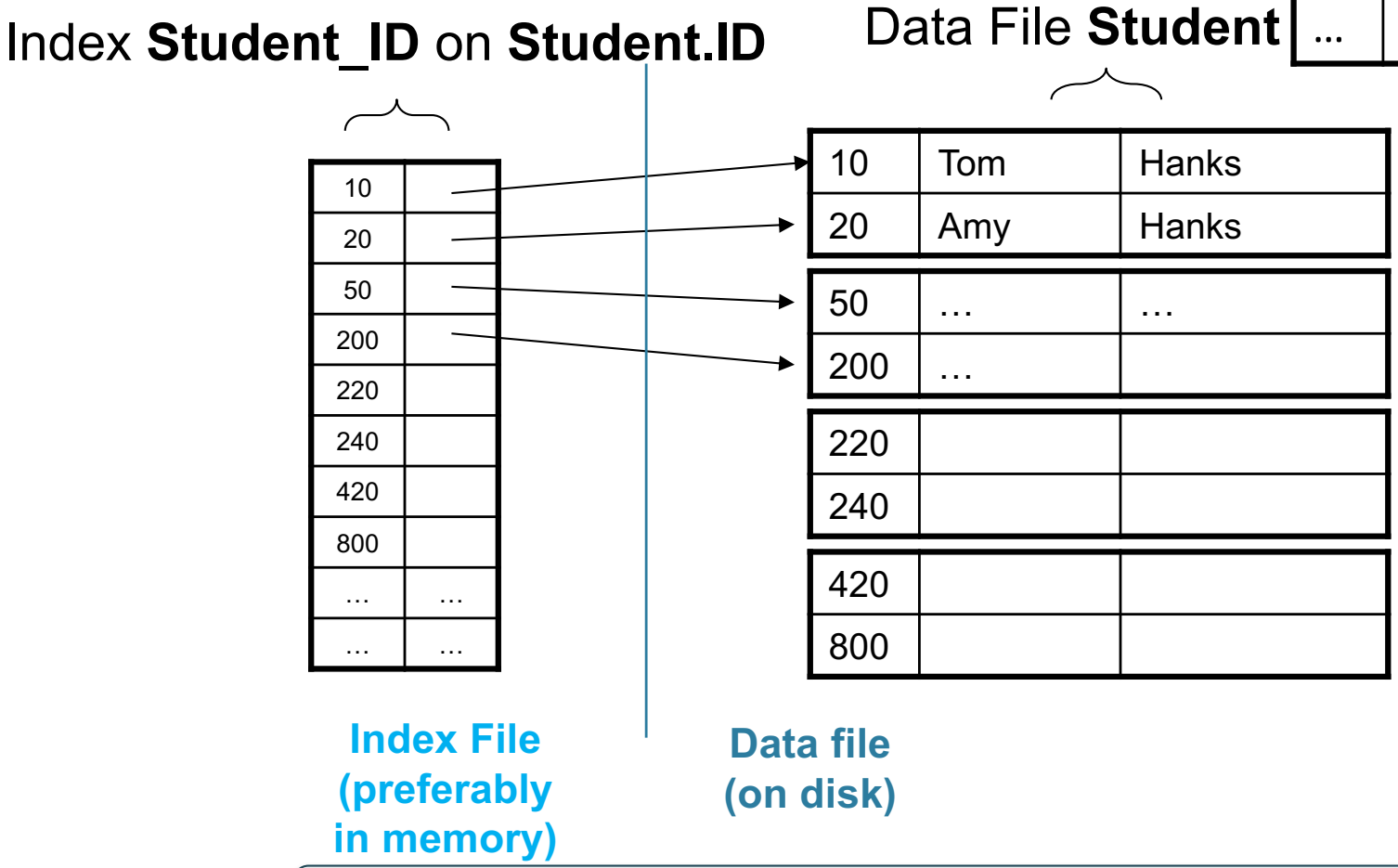
Introduction to Database Systems
CSE 414

Lecture 26: More Indexes and Operator Costs

Hash table example

Student

ID	fName	lName
10	Tom	Hanks
20	Amy	Hanks
…		

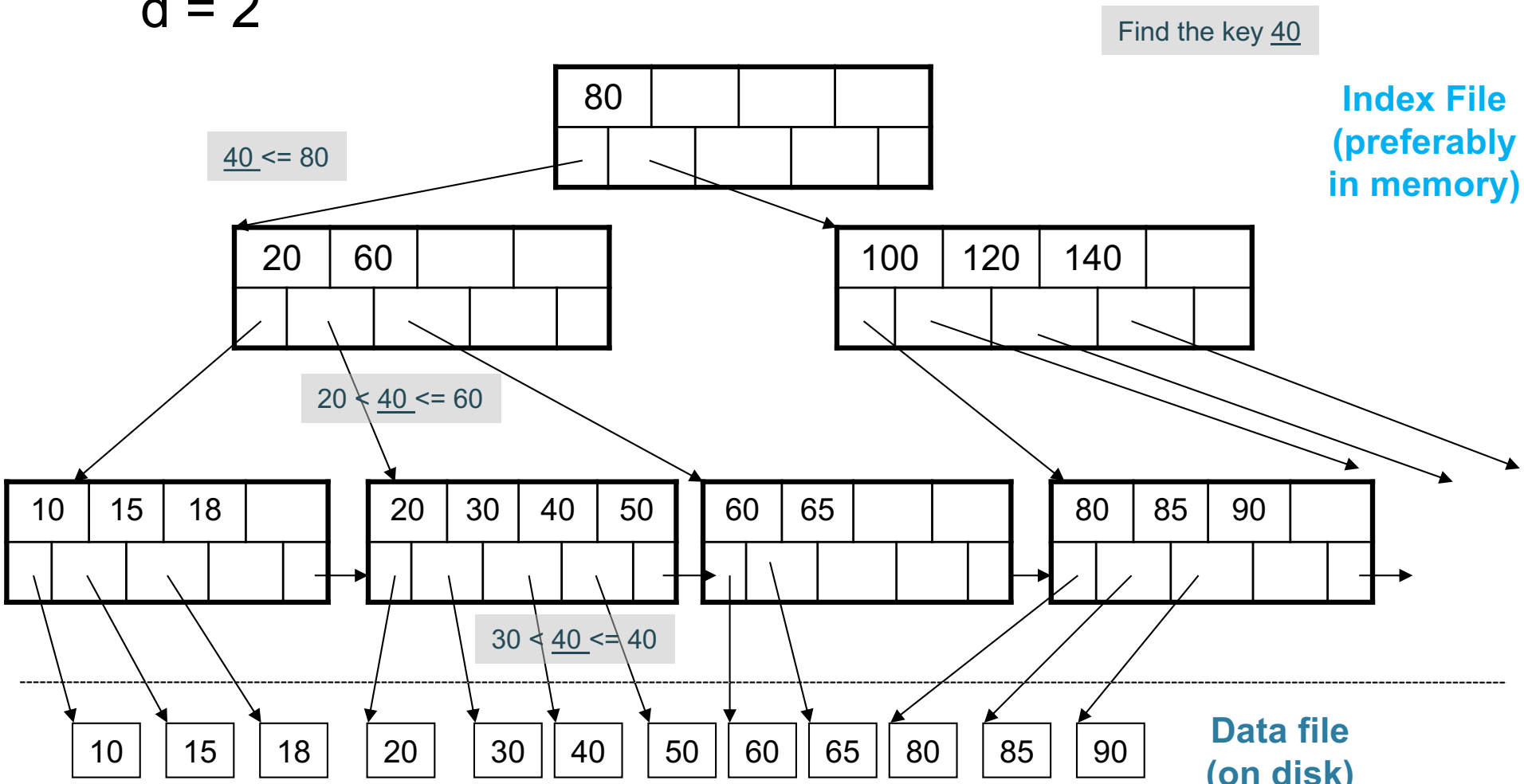


Hash table indexes are good for **point queries**

B+ Tree Index by Example

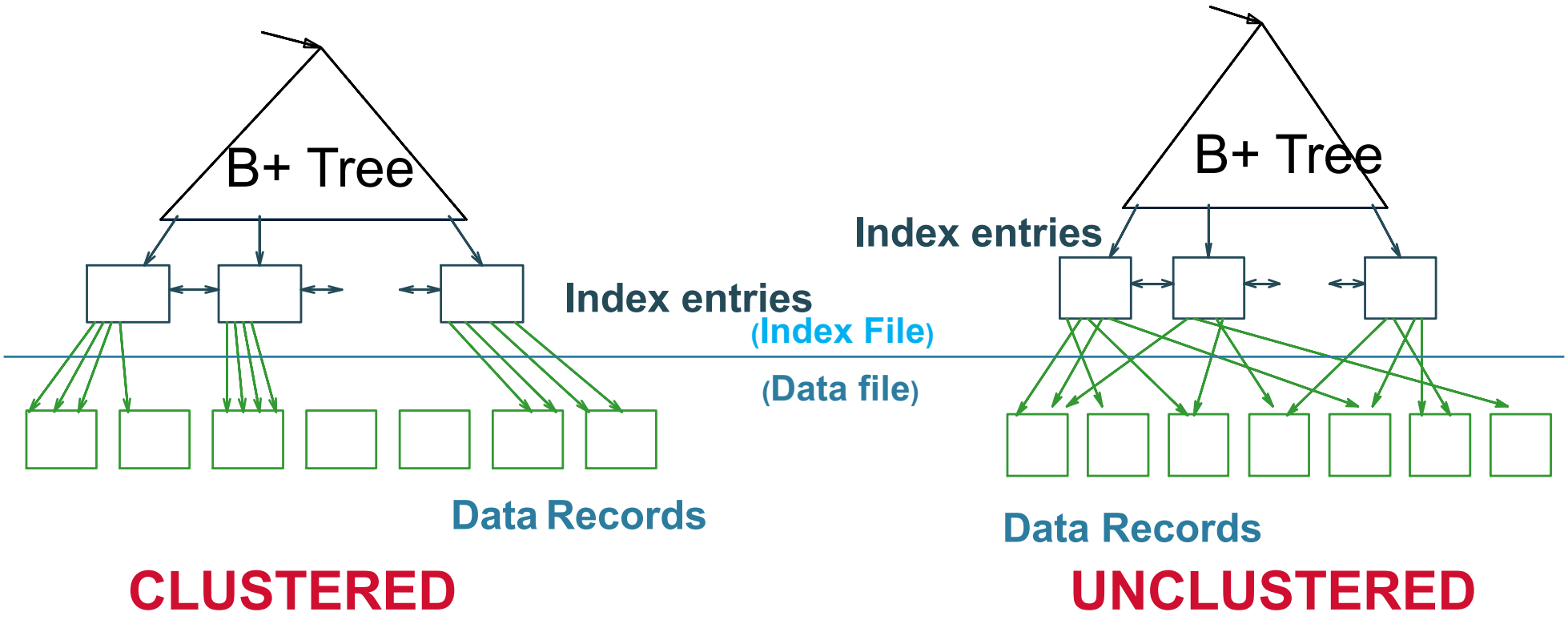
Recall binary trees from CSE 143!

d = 2



B+ indexes are good for **range queries**

Clustered vs Unclustered



Every table can have **only one** clustered and **many** unclustered indexes
Why?

Student(ID, fname, lname)
Takes(studentID, courseID)

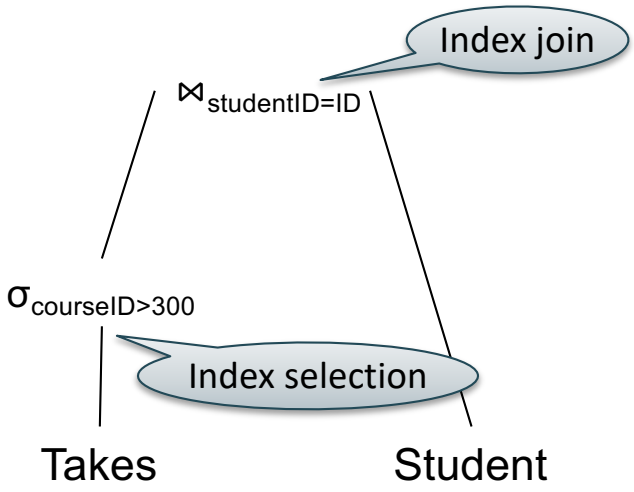
```
SELECT *
FROM  Student x, Takes y
WHERE x.ID=y.studentID AND y.courseID > 300
```

Example

```
for y in Takes
   if courseID > 300 then
     for x in Student
          if x.ID=y.studentID
            output *
```

Assume the database has indexes on these attributes:

- **Takes_courseID** = index on Takes.courseID
- **Student_ID** = index on Student.ID



Index selection

```
for y' in Takes_courseID where y'.courseID > 300
  y = fetch the Takes record pointed to by y'
  for x' in Student_ID where x'.ID = y.studentID
        x = fetch the Student record pointed to by x'
        output *
```

Which Indexes?

Student

ID	fName	lName
10	Tom	Hanks
20	Amy	Hanks
…		

- How many indexes could we create?
- Which indexes should we create?

In general this is a very hard problem

Which Indexes?

Student

ID	fName	lName
10	Tom	Hanks
20	Amy	Hanks
…		

- The *index selection problem*
  - Given a table, and a "workload" (big Java application with lots of SQL queries), decide which indexes to create (and which ones NOT to create!)
- Who does index selection:
  - The database administrator DBA
  - Semi-automatically, using a database administration tool

Index Selection: Which Search Key

- Make some attribute K a search key if the WHERE clause contains:
  - An exact match on K
  - A range predicate on K

The Index Selection Problem 1

```
V(M, N, P);
```

Your workload is this

100000 queries:

```
SELECT *
FROM V
WHERE N=?
```

100 queries:

```
SELECT *
FROM V
WHERE P=?
```

The Index Selection Problem 1

```
V(M, N, P);
```

Your workload is this

100000 queries:

```
SELECT *
FROM V
WHERE N=?
```

100 queries:

```
SELECT *
FROM V
WHERE P=?
```

What indexes ?

The Index Selection Problem 1

```
V(M, N, P);
```

Your workload is this

100000 queries:

```
SELECT *
FROM V
WHERE N=?
```

100 queries:

```
SELECT *
FROM V
WHERE P=?
```

A: V(N) and V(P) (hash tables or B-trees)

The Index Selection Problem 2

V(M, N, P);

Your workload is this

100000 queries:

```
SELECT *
FROM V
WHERE N>? and N<?
```

100 queries:

```
SELECT *
FROM V
WHERE P=?
```

100000 queries:

```
INSERT INTO V
VALUES (?, ?, ?)
```

What indexes ?

The Index Selection Problem 2

```
V(M, N, P);
```

Your workload is this

100000 queries:

```
SELECT *
FROM V
WHERE N>? and N<?
```

100 queries:

```
SELECT *
FROM V
WHERE P=?
```

100000 queries:

```
INSERT INTO V
VALUES (?, ?, ?)
```

A: definitely V(N) (must B-tree); unsure about V(P)

The Index Selection Problem 3

```
V(M, N, P);
```

Your workload is this

100000 queries:

```
SELECT *
FROM V
WHERE N=?
```

1000000 queries:

```
SELECT *
FROM V
WHERE N=? and P>?
```

100000 queries:

```
INSERT INTO V
VALUES (?, ?, ?)
```

What indexes ?

The Index Selection Problem 3

```
V(M, N, P);
```

Your workload is this

100000 queries:

```
SELECT *
FROM V
WHERE N=?
```

1000000 queries:

```
SELECT *
FROM V
WHERE N=? and P>?
```

100000 queries:

```
INSERT INTO V
VALUES (?, ?, ?)
```

A: V(N, P)

How does this index differ from:

1. Two indexes V(N) and V(P)?
2. An index V(P, N)?

The Index Selection Problem 4

V(M, N, P);

Your workload is this

1000 queries:

```
SELECT *
FROM V
WHERE N>? and N<?
```

100000 queries:

```
SELECT *
FROM V
WHERE P>? and P<?
```

What indexes ?

The Index Selection Problem 4

V(M, N, P);

Your workload is this

1000 queries:

```
SELECT *
FROM V
WHERE N>? and N<?
```

100000 queries:

```
SELECT *
FROM V
WHERE P>? and P<?
```

A: V(N) secondary, V(P) primary index

Two typical kinds of queries

```
SELECT *
FROM Movie
WHERE year = ?
```

- Point queries
- What data structure should be used for index?

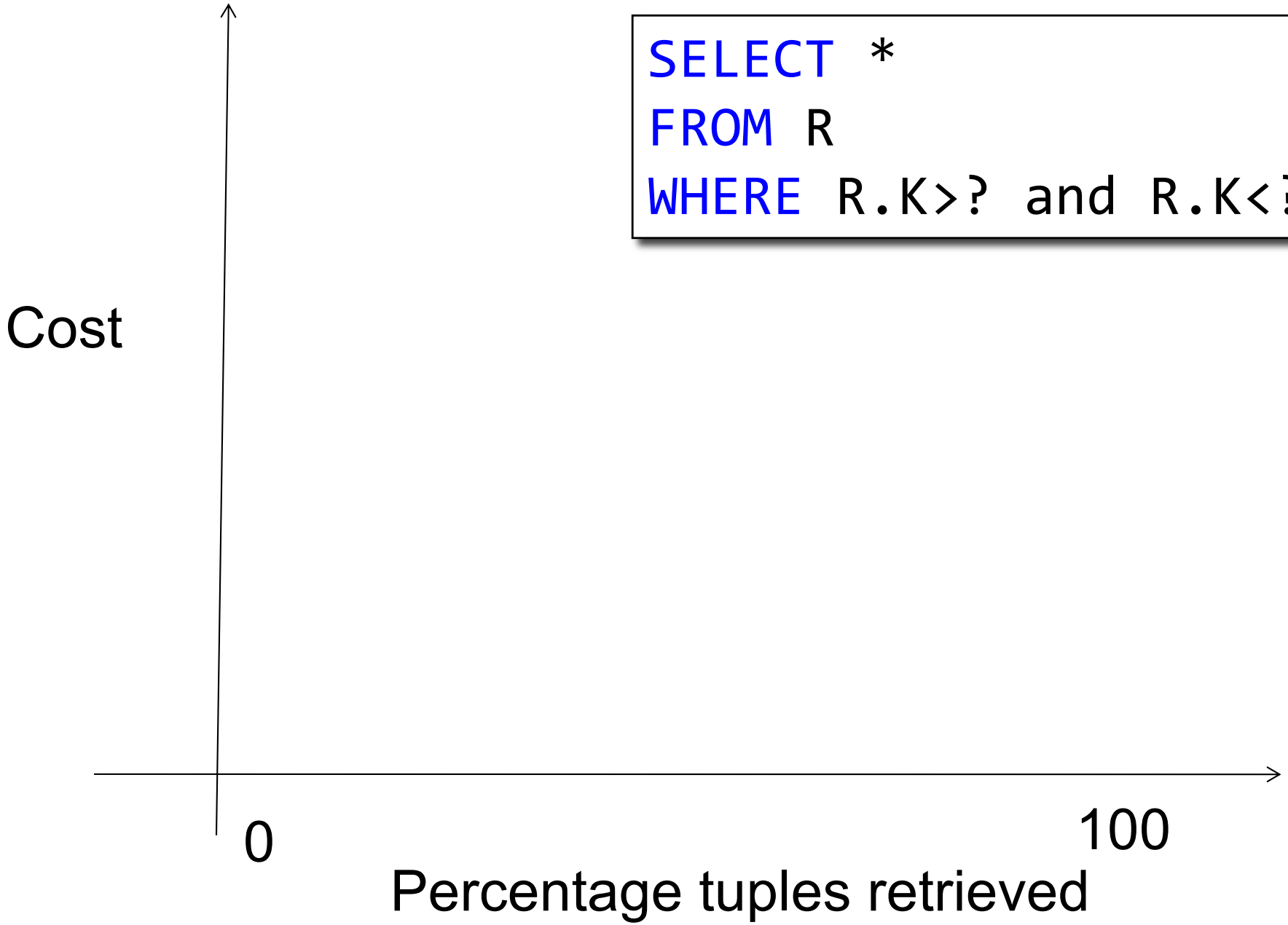
```
SELECT *
FROM Movie
WHERE year >= ? AND
      year <= ?
```

- Range queries
- What data structure should be used for index?

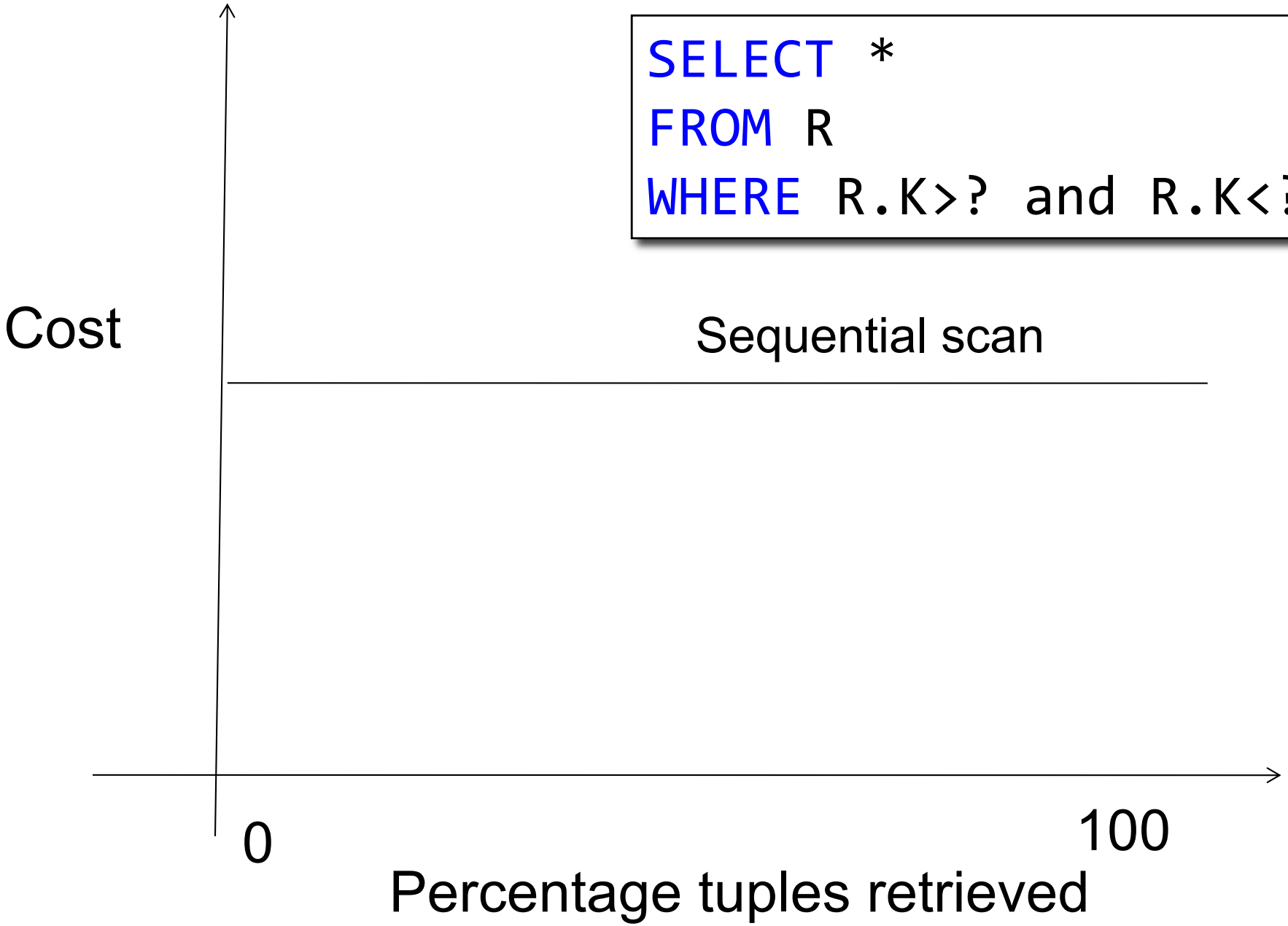
Basic Index Selection Guidelines

- Consider queries in workload in order of importance
- Consider relations accessed by query
  - No point indexing other relations
- Look at WHERE clause for possible search key
- Try to choose indexes that speed-up multiple queries
- Range queries benefit mostly from clustering

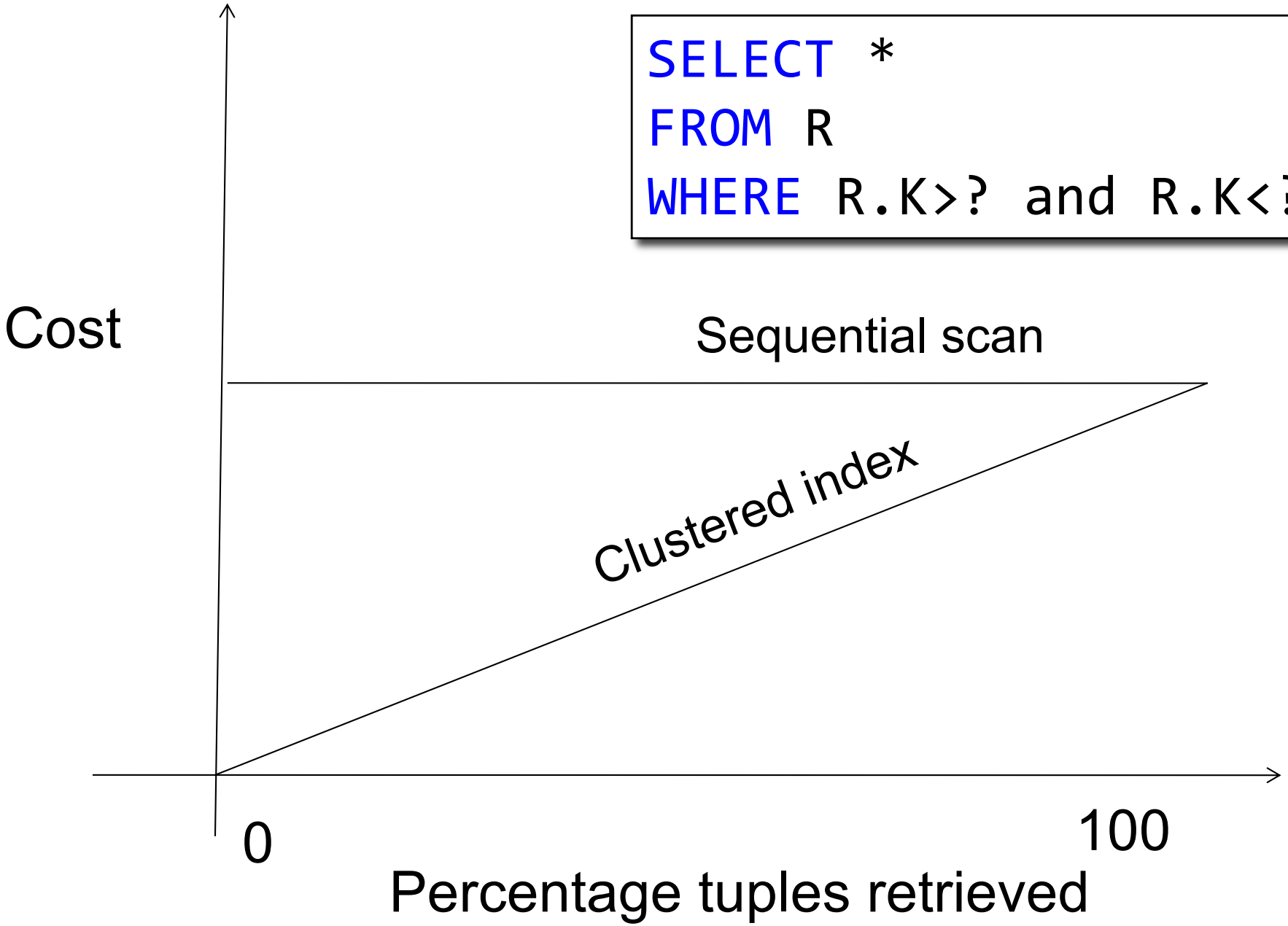
```
SELECT *
FROM R
WHERE R.K>? and R.K<?
```

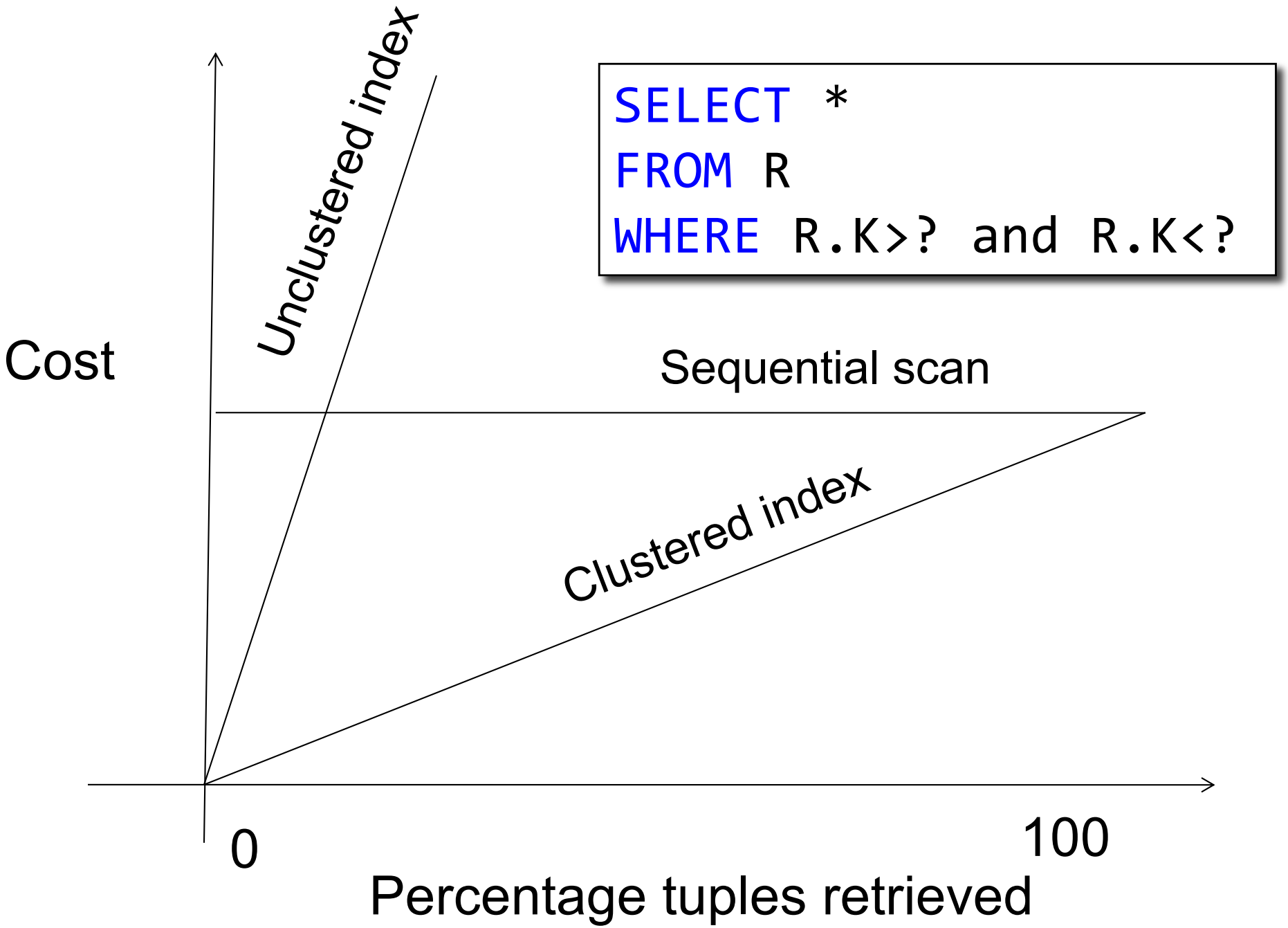


```
SELECT *
FROM R
WHERE R.K>? and R.K<?
```



```
SELECT *
FROM R
WHERE R.K>? and R.K<?
```




SELECT *
FROM R
WHERE R.K>? and R.K<?
Cost
Unclustered index
Sequential scan
Clustered index
0
100
Percentage tuples retrieved

Choosing Index is Not Enough

- To estimate the cost of a query plan, we still need to consider other factors:
  - How each operator is implemented
  - The cost of each operator
  - Let's start with the basics

Cost of Reading Data From Disk

Cost Parameters

- Cost = I/O + CPU + Network BW
  - We will focus on I/O in this class
- Parameters (a.k.a. statistics):
  - **B(R)** = # of blocks (i.e., pages) for relation R
  - **T(R)** = # of tuples in relation R
  - **V(R, a)** = # of distinct values of attribute a

When **a** is a key, **V(R,a) = T(R)**
When **a** is not a key, **V(R,a)** can be anything <= **T(R)**

- DBMS collects statistics about base tables must infer them for intermediate results

Selectivity Factors for Conditions

- A = c /* $\sigma_{A=c}(R)$ */
  - Selectivity = 1/V(R,A)
- A < c /* $\sigma_{A<c}(R)$*/
  - Selectivity = (c - min(R, A))/(max(R,A) - min(R,A))
- c1 < A < c2 /* $\sigma_{c1<A<c2}(R)$*/
  - Selectivity = (c2 – c1)/(max(R,A) - min(R,A))

Cost of Reading Data From Disk

- Sequential scan for relation R costs **B(R)**

Index Based Selection

- Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R) = ?$

- Table scan:
- Index based selection:

Index Based Selection

- Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R) = ?$

- Table scan: B(R) = 2,000 I/Os
- Index based selection:

Index Based Selection

- Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R) = ?$

- Table scan: B(R) = 2,000 I/Os
- Index based selection:
  - If index is clustered:
  - If index is unclustered:

Index Based Selection

- Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R) = ?$

- Table scan: B(R) = 2,000 I/Os
- Index based selection:
  - If index is clustered: B(R) * 1/V(R,a) = 100 I/Os
  - If index is unclustered:

Index Based Selection

- Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R) = ?$

- Table scan: B(R) = 2,000 I/Os
- Index based selection:
  - If index is clustered: B(R) * 1/V(R,a) = 100 I/Os
  - If index is unclustered: T(R) * 1/V(R,a) = 5,000 I/Os

Note: we ignore I/O cost for index pages

Index Based Selection

- Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R) = ?$

- Table scan: B(R) = 2,000 I/Os
- Index based selection:
  - If index is clustered: B(R) * 1/V(R,a) = 100 I/Os
  - If index is unclustered: T(R) * 1/V(R,a) = 5,000 I/Os

Lesson: Don't build unclustered indexes when V(R,a) is small !

Cost of Executing Operators (Focus on Single Node Joins)

Outline

- **Join operator algorithms**
  - One-pass algorithms (Sec. 15.2 and 15.3)
  - Index-based algorithms (Sec 15.6)

- Note about readings:
  - In class, we discuss only algorithms for joins
  - Other operators are easier: read the book

Join Algorithms

- Nested loop join
- Hash join
- Sort-merge join

Nested Loop Joins

- Tuple-based nested loop R ⋈ S
- R is the outer relation, S is the inner relation

```
for each tuple t1 in R do
   for each tuple t2 in S do
      if t1 and t2 join then output (t1,t2)
```

What is the Cost?

Nested Loop Joins

- Tuple-based nested loop R ⋈ S
- R is the outer relation, S is the inner relation

```
for each tuple t1 in R do
   for each tuple t2 in S do
      if t1 and t2 join then output (t1,t2)
```

What is the Cost?

- Cost: B(R) + T(R) B(S)
- Multiple-pass since S is read many times

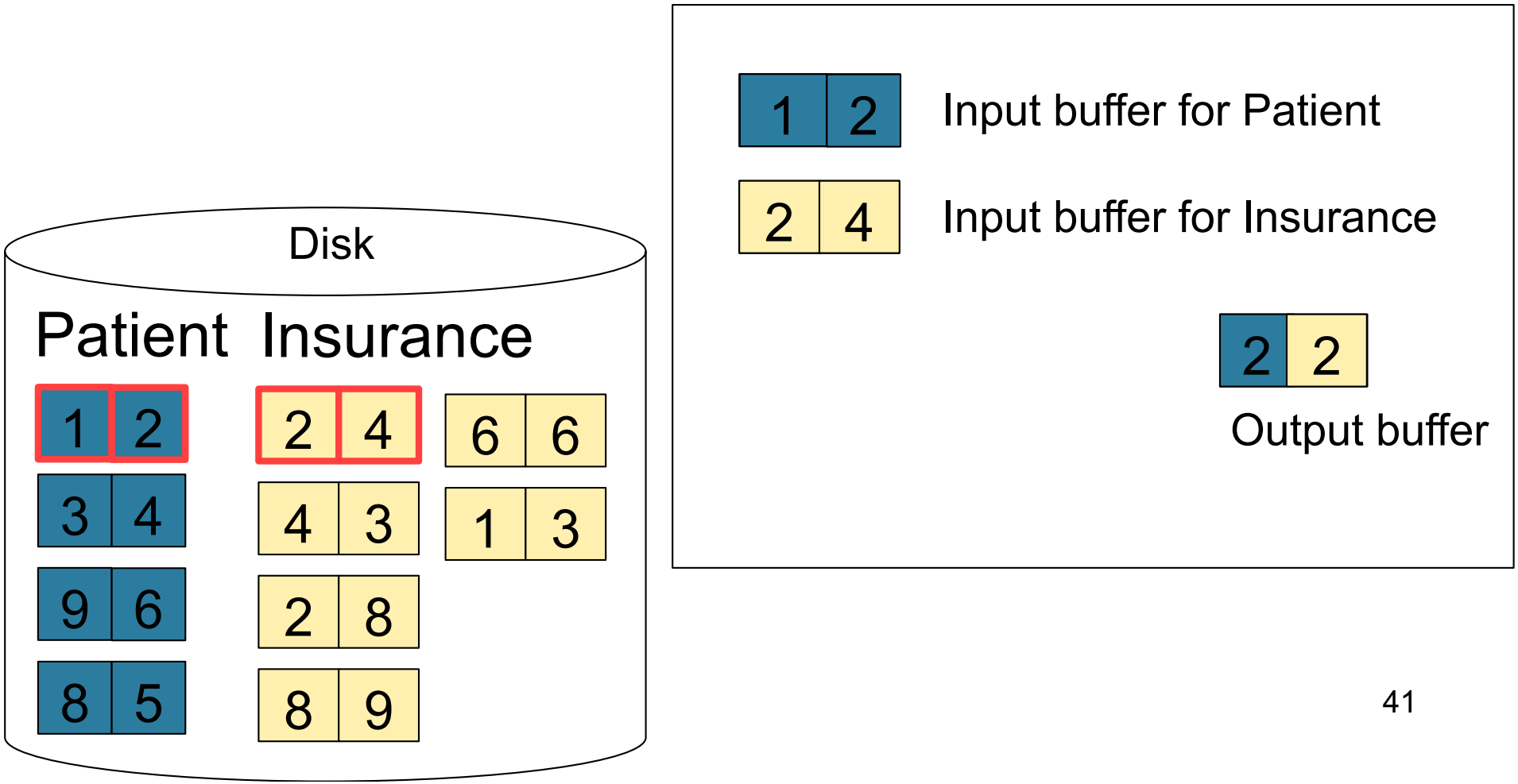
Page-at-a-time Refinement

```
for each page of tuples r in R do
   for each page of tuples s in S do
         for all pairs of tuples t1 in r, t2 in s
               if t1 and t2 join then output (t1,t2)
```

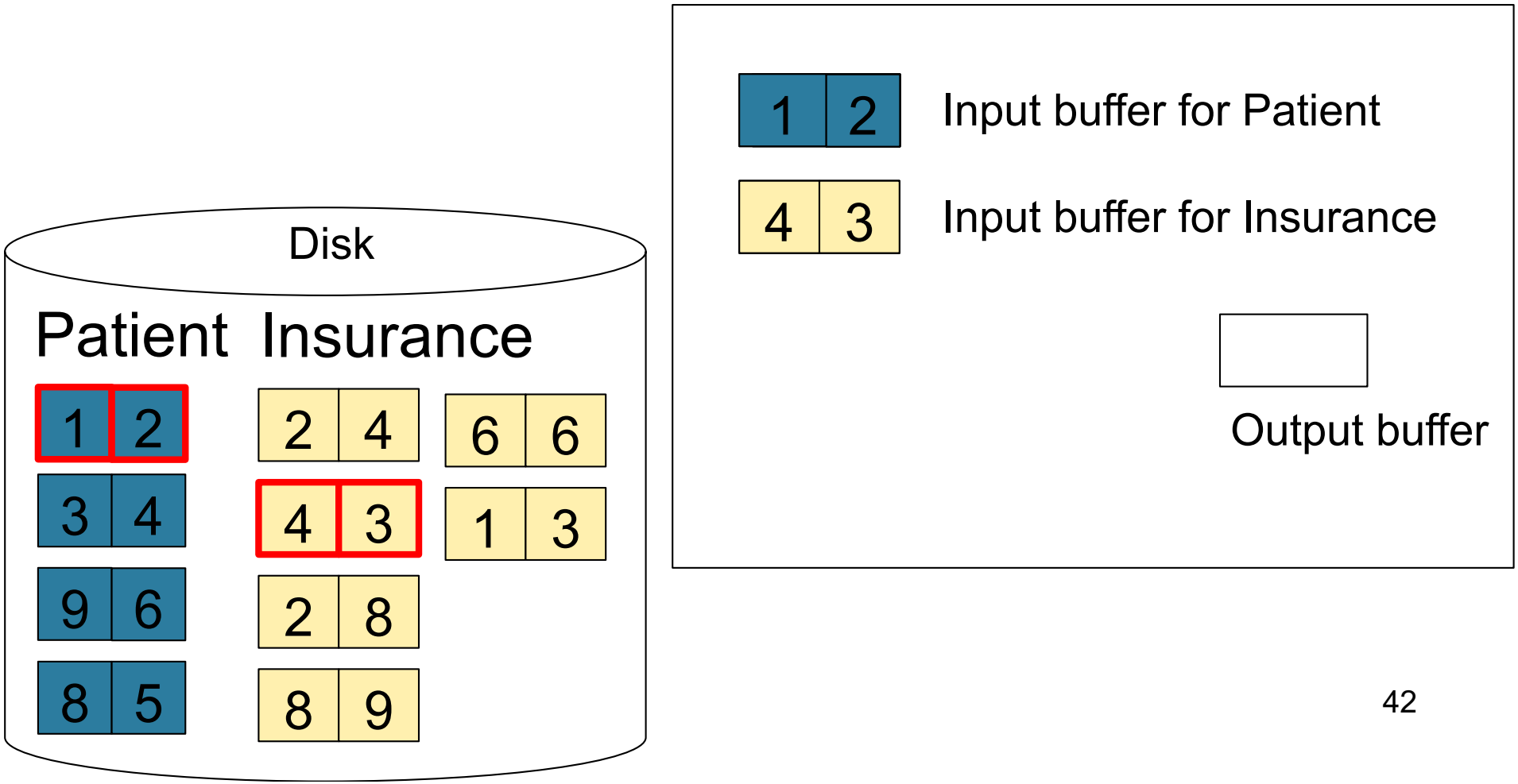
- Cost: B(R) + B(R)B(S)

What is the Cost?

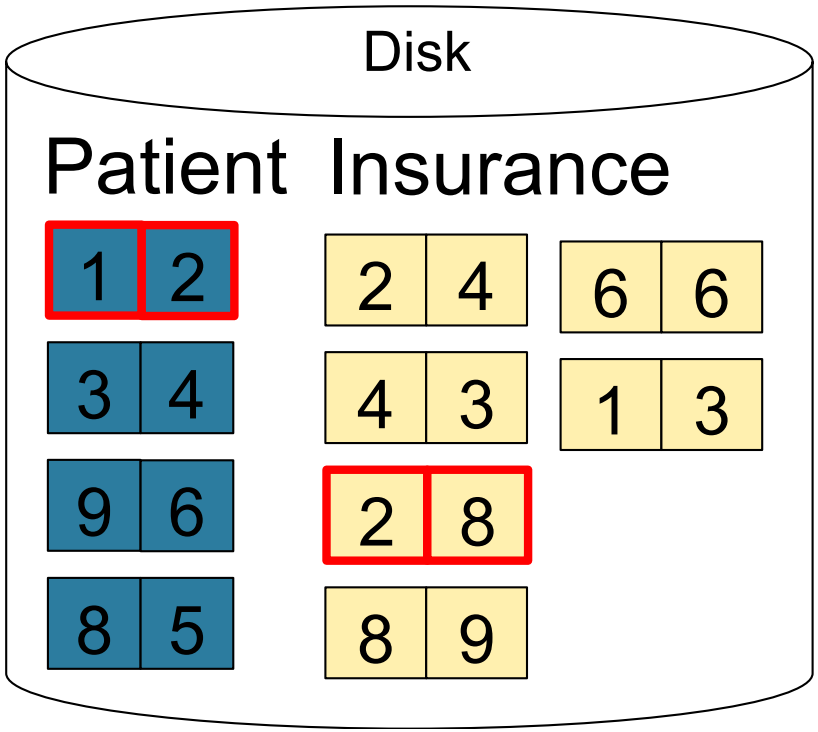
Page-at-a-time Refinement

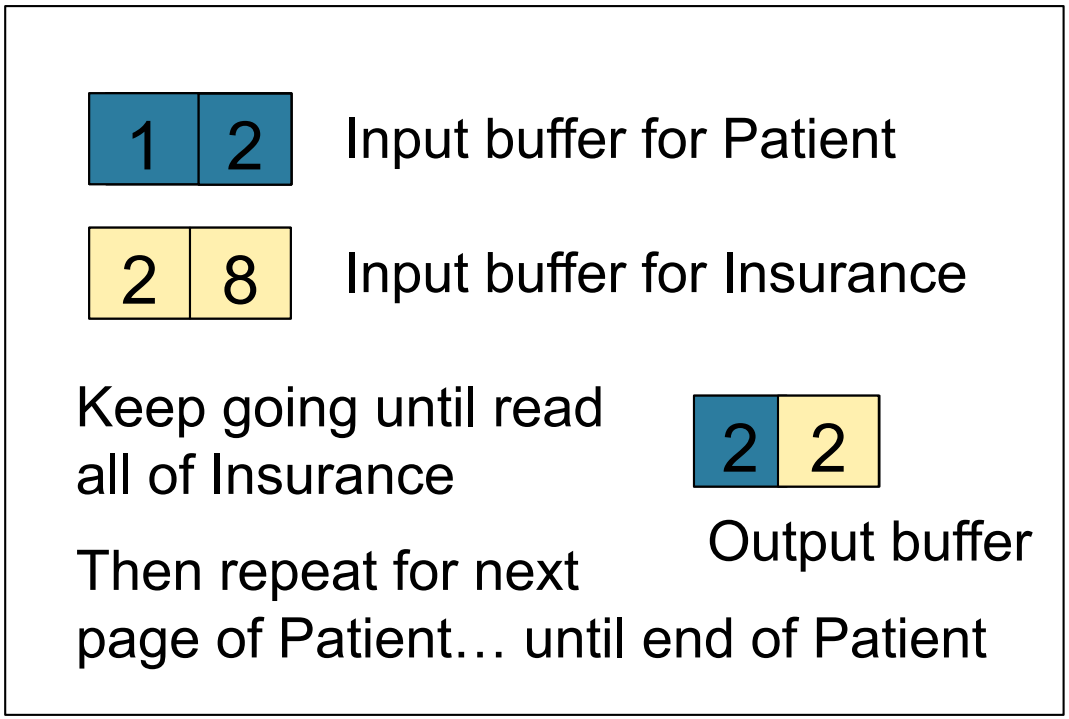


Page-at-a-time Refinement



Page-at-a-time Refinement





Cost: B(R) + B(R)B(S)