CS 525:
Advanced Database Organization

01: Introduction

Boris Glavic

Slides: adapted from a course taught by
Hector Garcia-Molina, Stanford InfoLab

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Advanced Database Organization?

- =Database Implementation
- =How to implement a database system
- … and have fun doing it ;-)

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Isn't Implementing a Database System Simple?

Relations $\Longrightarrow$ Statements $\Longrightarrow$ Results

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Introducing the

MEGATRON 3000

Database Management System

- The latest from Megatron Labs
- Incorporates latest relational technology
- UNIX compatible

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Megatron 3000 Implementation Details

! First sign non-disclosure agreement !

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Megatron 3000 Implementation Details

- Relations stored in files (ASCII)

  e.g., relation R is in /usr/db/R

```
Smith # 123 # CS
Jones # 522 # EE
   .
   .
   .
```

COMPUTER
SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Megatron 3000 Implementation Details

- Directory file (ASCII) in /usr/db/directory

```
R1 # A # INT # B # STR …
R2 # C # STR # A # INT …
    ⋮
```

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Megatron 3000
Sample Sessions

```
% MEGATRON3000
   Welcome to MEGATRON 3000!
&
   :
   :
& quit
%
```

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Megatron 3000
Sample Sessions

```
& select *
  from R #

   Relation R
  A         B      C
   SMITH   123    CS


&
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Megatron 3000
Sample Sessions

```
& select A,B
  from R,S
  where R.A = S.A and S.C > 100 #

   A     B
  123   CAR
  522   CAT


&
```

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Megatron 3000
Sample Sessions

```
& select *
  from R | LPR #
&
```

Result sent to LPR (printer).

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Megatron 3000 Sample Sessions

```
& select *
  from R
  where R.A < 100 | T #
&
```

New relation T created.

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Megatron 3000

- To execute "`select * from R where condition`":

 (1) Read dictionary to get R attributes

 (2) Read R file, for each line:

 (a) Check condition

 (b) If OK, display

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Megatron 3000

- To execute "`select * from R`
   `where condition | T`":
  - (1) Process select as before
  - (2) Write results to new file T
  - (3) Append new line to dictionary

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Megatron 3000

- To execute "**select A,B from R,S where *condition***":

     (1) Read dictionary to get R,S attributes

     (2) Read R file, for each line:

          (a) Read S file, for each line:

               (i) Create join tuple

               (ii) Check condition

               (iii) Display if OK

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

What's wrong with the Megatron 3000 DBMS?

CS 525

COMPUTER
SCIENCE

Notes 1 - Introduction   16

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's wrong with the Megatron 3000 DBMS?

- Tuple layout on disk

e.g.,   - Change string from 'Cat' to 'Cats' and we have to rewrite file

       - ASCII storage is expensive

       - Deletions are expensive

What's wrong with the Megatron 3000 DBMS?

- Search expensive; no indexes

e.g.,   - Cannot find tuple with given key quickly

       - Always have to read full relation

What's wrong with the Megatron 3000 DBMS?

- Brute force query processing

e.g.,   `select *`

       `from R,S`

       `where R.A = S.A and S.B > 1000`

       - Do select first?

       - More efficient join?

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

What's wrong with the Megatron 3000 DBMS?

- No buffer manager

e.g., Need caching

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's wrong with the Megatron 3000 DBMS?

- No concurrency control

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's wrong with the Megatron 3000 DBMS?

- No reliability

e.g.,   - Can lose data

       - Can leave operations half done

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

What's wrong with the Megatron 3000 DBMS?

- No security

e.g.,   - File system insecure

        - File system security is coarse

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's wrong with the Megatron 3000 DBMS?

- No application program interface (API)

e.g.,   How can a payroll program get at the data?

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

What's wrong with the Megatron 3000 DBMS?

- Cannot interact with other DBMSs.

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

What's wrong with the Megatron 3000 DBMS?

- Poor dictionary facilities

What's wrong with the Megatron 3000 DBMS?

- No GUI

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

What's wrong with the Megatron 3000 DBMS?

- Lousy salesman!!

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Course Overview

- ## File & System Structure

    Records in blocks, dictionary, buffer management,…

- ## Indexing & Hashing

    B-Trees, hashing,…

- ## Query Processing

    Query costs, join strategies,…

- ## Crash Recovery

    Failures, stable storage,…

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Course Overview

- ## Concurrency Control

  Correctness, locks,…

- ## Transaction Processing

  Logs, deadlocks,…

- ## Security & Integrity

  Authorization, encryption,…

- ## Advanced Topics

  Distribution, More Fancy Optimizations, …

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

System Structure

Strategy Selector ← Query Parser ← User

Strategy Selector → User Transaction

User Transaction ↔ Transaction Manager

Transaction Manager → Recovery Manager

User Transaction ↔ Concurrency Control ↔ Buffer Manager

Buffer Manager — Recovery Manager

Concurrency Control → Lock Table

Buffer Manager → File Manager

Buffer Manager ⇢ M.M. Buffer

Recovery Manager → Log

File Manager → Database

Statistical Data

Indexes

User Data   System Data

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Some Terms

- Database system
- Transaction processing system
- File access system
- Information retrieval system

Course Information

- **Webpage**: http://www.cs.iit.edu/~cs525/

- **Instructor**: Boris Glavic
  - http://www.cs.iit.edu/~glavic/
  - **DBGroup:** http://www.cs.iit.edu/~dbgroup/
  - **Office Hours: Mondays, 12pm-1pm**
  - **Office:** Stuart Building, Room 226 C

- **TA: TBA**

- **Time:** Mon + Wed 1:50pm – 3:05pm

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Google Group

- https://groups.google.com/forum/#!forum/cs525-2017-spring-group
- Mailing-list for announcements
- Discussion forum
  - Student - Instructor/TA
  - Student – Student
- ->please join the group to keep up to date

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Workload and Grading

- Schedule and Important Dates
    - On webpage & updated there

- Programming Assignments (50%)
    - 4 Assignments
    - Groups of 3 students
    - Plagiarism -> 0 points and administrative action

- Quizzes (10%)

- Mid Term (20%) and Final Exam (20%)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Textbooks

- Elmasri and Navathe , **Fundamentals of Database Systems**, 6th Edition , Addison-Wesley , 2003
- Garcia-Molina, Ullman, and Widom, **Database Systems: The Complete Book**, 2nd Edition, Prentice Hall, 2008
- Ramakrishnan and Gehrke , **Database Management Systems**, 3nd Edition , McGraw-Hill , 2002
- Silberschatz, Korth, and Sudarshan , **Database System Concepts**, 6th Edition , McGraw Hill , 2010

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Programming Assignments

- 4 assignments one on-top of the other
- Optional 5th assignment for extra credit
- Code has to compile & run on server account
  - **Email-ID**@fourier.cs.iit.edu
  - Linux machine
  - SSH with X-forwarding
- Source code managed in **git** repository on Bitbucket.org
  - Handing in assignments = submit (push) to repository
  - One repository per student
  - You should have gotten an invitation (if not, contact me/TA)
  - Git tutorials linked on course webpage!

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Next:

- Hardware

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

CS 525: Advanced Database Organization
02: **Hardware**

Boris Glavic

Slides: adapted from a course taught by
Hector Garcia-Molina, Stanford InfoLab

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

- Hardware: Disks
- Access Times
- Example - Megatron 747
- Optimizations
- Other Topics:
  - Storage costs
  - Using secondary storage
  - Disk failures

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Hardware

DBMS

Data Storage

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
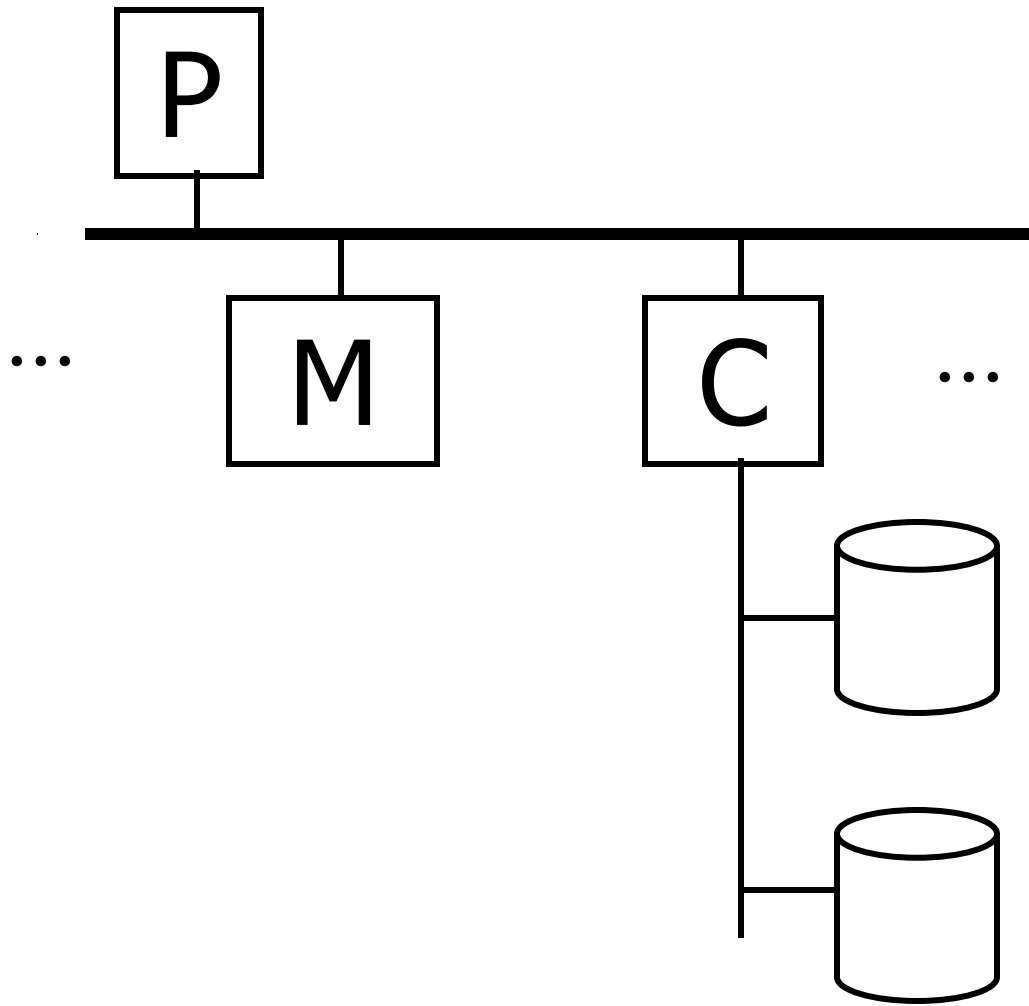
Typical Computer

Secondary Storage

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Processor

Fast, slow, reduced instruction set, with cache, pipelined…

Speed: $100 \rightarrow 500 \rightarrow 1000$ MIPS

Memory

Fast, slow, non-volatile, read-only,…

Access time: $10^{-6} \rightarrow 10^{-9}$ sec.

$1 \, \mu s \rightarrow 1 \, ns$

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Secondary storage

Many flavors:

- Disk: Floppy (hard, soft)
          Removable Packs
          Winchester
          Ram disks
          Optical, CD-ROM...
          Arrays

- Tape   Reel, cartridge
          Robots

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Focus on: "Typical Disk"
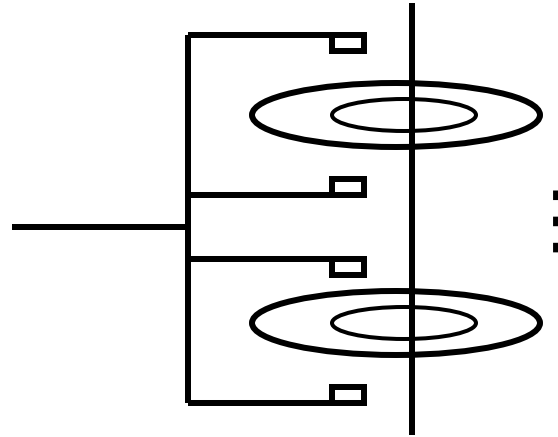
Terms: Platter, Head, Actuator
Cylinder, Track
Sector (physical),
Block (logical), Gap

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Top View

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
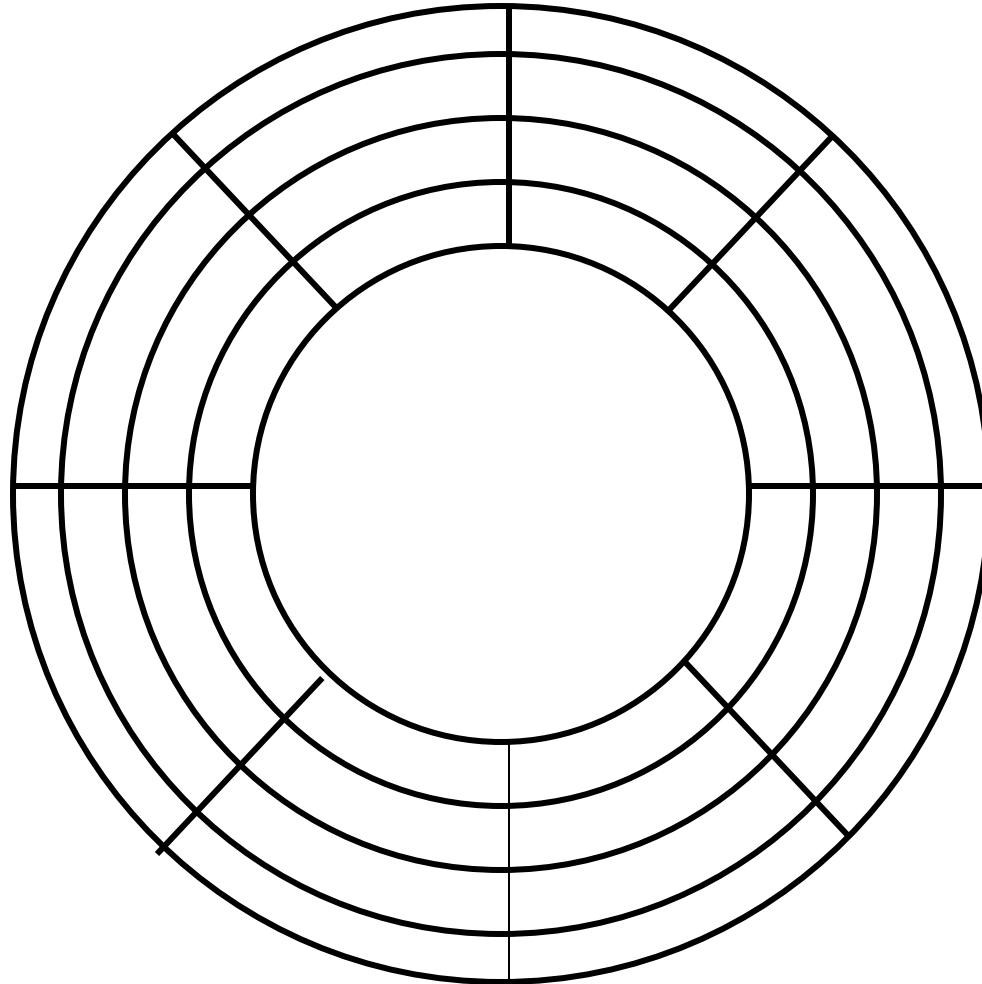
"Typical" Numbers

Diameter: 1 inch → 15 inches

Cylinders: 100 → 2000

Surfaces: 1 (CDs) →

(Tracks/cyl)   2 (floppies) → 30

Sector Size: 512B → 50K

Capacity: 360 KB (old floppy)
          → 1 TB (I use)

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Disk Access Time

I want
block X

$\longrightarrow$

?

$\longrightarrow$

block x
in memory

COMPUTER
SCIENCE

IIT College of
Science and Letters
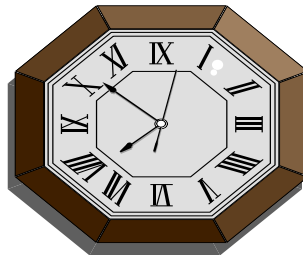
ILLINOIS INSTITUTE OF TECHNOLOGY

Time = Seek Time +
Rotational Delay +
Transfer Time +
Other
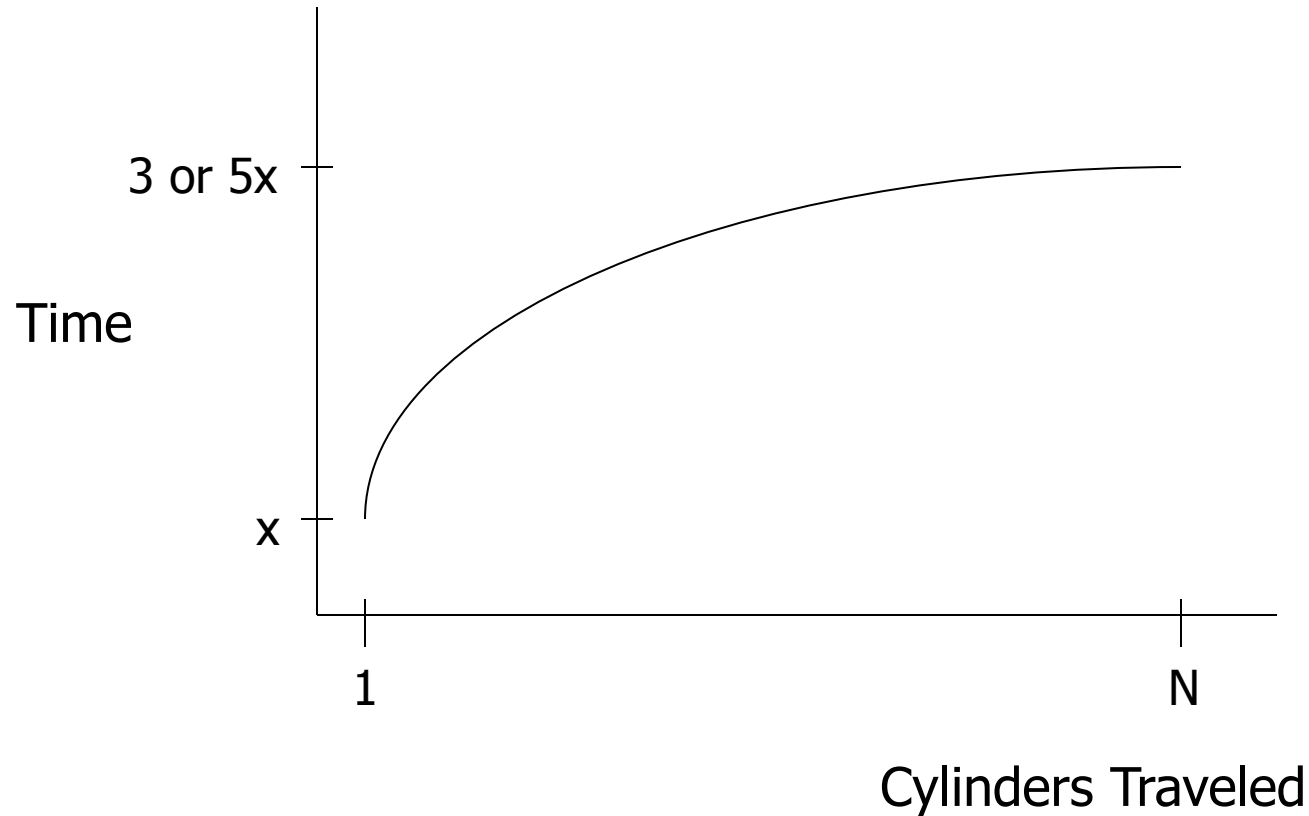
Notes 2 - Hardware

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Seek Time

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Average Random Seek Time

$$S = \frac{\displaystyle\sum_{\substack{i=1}}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \text{SEEKTIME } (i \rightarrow j)}{N(N-1)}$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Average Random Seek Time

$$S = \frac{\displaystyle\sum_{\substack{i=1}}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \text{SEEKTIME} (i \to j)}{N(N-1)}$$

"Typical" S: 10 ms → 40 ms

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rotational Delay

Head Here

Block I Want

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
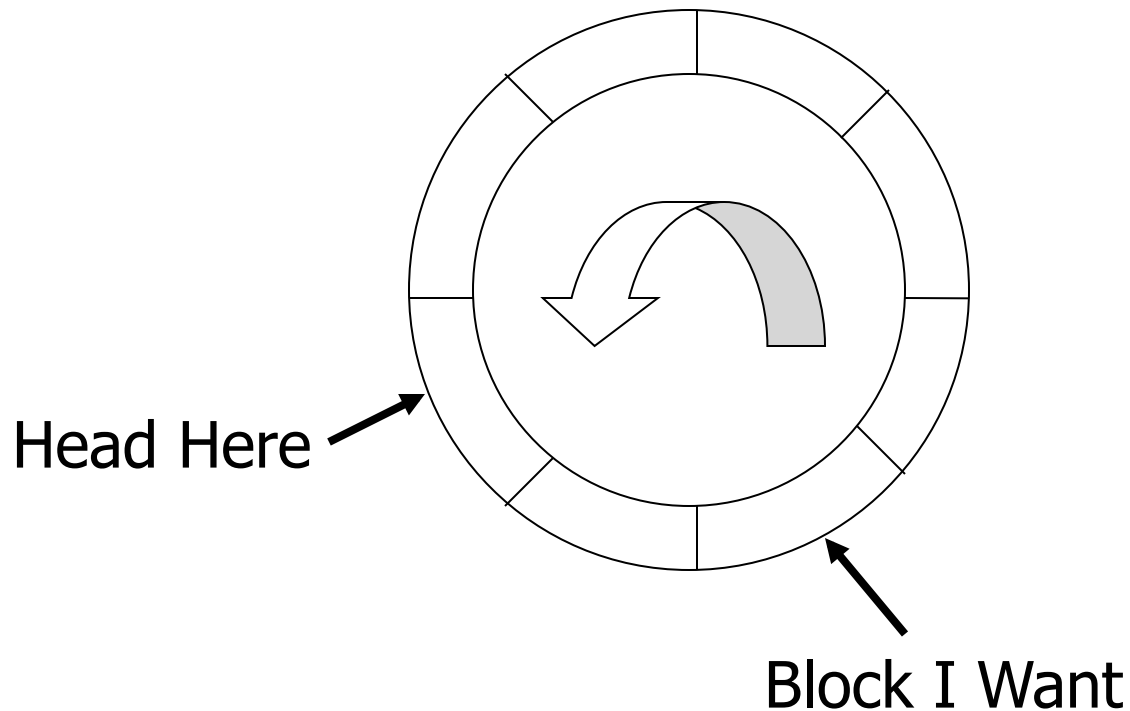
Average Rotational Delay

R = 1/2 revolution

"typical" R = 8.33 ms (3600 RPM)

Transfer Rate: t

- "typical" t: 10's → 100's MB/second
- transfer time: $\dfrac{\text{block size}}{t}$

Notes 2 - Hardware

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Other Delays

- CPU time to issue I/O
- Contention for controller
- Contention for bus, memory

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Other Delays

- CPU time to issue I/O
- Contention for controller
- Contention for bus, memory

"Typical" Value: 0

COMPUTER
SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Other Delays (now and near future)

- Increasing amount of parallelism
- Contention can become a problem
- -> need rethink approach to scale

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

- So far: Random Block Access
- What about: Reading "Next" block?

COMPUTER SCIENCE

Notes 2 - Hardware

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

If we do things right (e.g., Double Buffer, Stagger Blocks...)

$$\text{Time to get block} = \frac{\text{Block Size}}{t} + \text{Negligible}$$

- skip gap
- switch track
- once in a while, next cylinder

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Rule of Thumb	Random I/O: Expensive
	Sequential I/O: Much less

- Ex: 1 KB Block
  - » Random I/O: ~ 20 ms.
  - » Sequential I/O: ~ 1 ms.

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Cost for <u>Writing</u> similar to <u>Reading</u>

…. unless we want to verify!
need to add (full) rotation + <u>Block size</u>
t

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- To <u>Modify</u> a Block?

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

- To <u>Modify</u> a Block?

<u>To Modify Block:</u>
(a) Read Block
(b) Modify in Memory
(c) Write Block
[(d) Verify?]

Notes 2 - Hardware

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Block Address:

- Physical Device
- Cylinder #
- Surface #
- Sector

Complication: Bad Blocks

- Messy to handle
- May map via software to integer sequence

1
2
.
.
m
→ (Map) → Actual Block Addresses

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
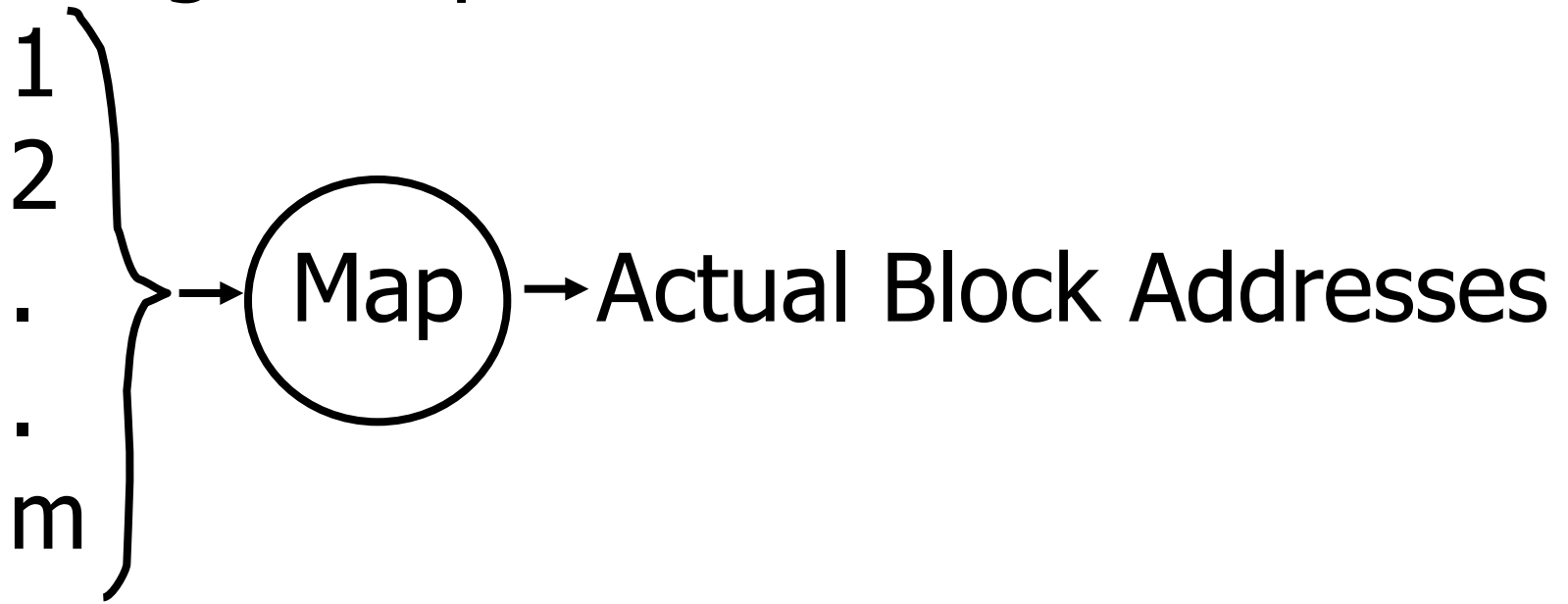
An Example — Megatron 747 Disk (old)

- 3.5 in diameter
- 3600 RPM
- 1 surface
- 16 MB usable capacity (16 X 2^{20})
- 128 cylinders
- seek time: average = 25 ms.

                              adjacent cyl = 5 ms.

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

- 1 KB blocks = sectors
- 10% overhead between blocks
- capacity = 16 MB = $(2^{20})16 = 2^{24}$
- # cylinders = 128 = 2^7
- bytes/cyl = $2^{24}/2^7 = 2^{17}$ = 128 KB
- blocks/cyl = 128 KB / 1 KB = 128

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

3600 RPM → 60 revolutions / sec
→ 1 rev. = 16.66 msec.

One track:

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
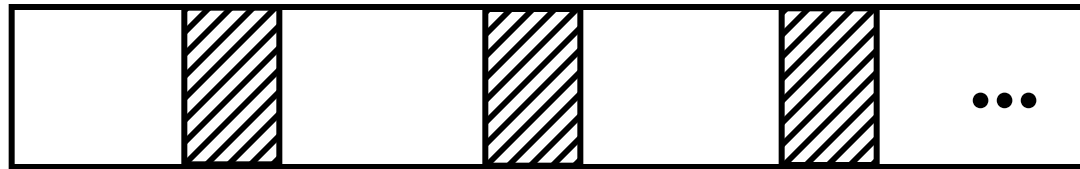
3600 RPM $\rightarrow$ 60 revolutions / sec
$\longrightarrow$ 1 rev. = 16.66 msec.

One track:
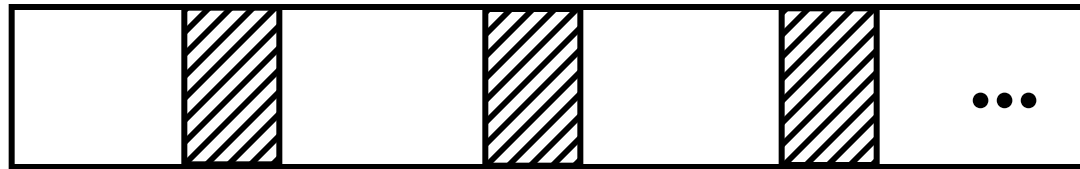
Time over useful data:(16.66)(0.9)=14.99 ms.
Time over gaps: (16.66)(0.1) = 1.66 ms.
Transfer time 1 block = 14.99/128=0.117 ms.
Trans. time 1 block+gap=16.66/128=0.13ms.

Burst Bandwith

$$1 \text{ KB in } 0.117 \text{ ms.}$$

$$BB = 1/0.117 = 8.54 \text{ KB/ms.}$$

or

$$BB = 8.54 \text{KB/ms} \times 1000 \text{ ms/1sec} \times 1 \text{MB/1024KB}$$
$$= 8540/1024 = 8.33 \text{ MB/sec}$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Sustained bandwith (over track)
128 KB in 16.66 ms.

SB = 128/16.66 = 7.68 KB/ms

or

SB = 7.68 x 1000/1024 = 7.50 MB/sec.

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

T_1 = Time to read one random block

T_1 = seek + rotational delay + TT

$$= 25 + (16.66/2) + .117 = 33.45 \text{ ms.}$$

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Suppose OS deals with 4 KB blocks
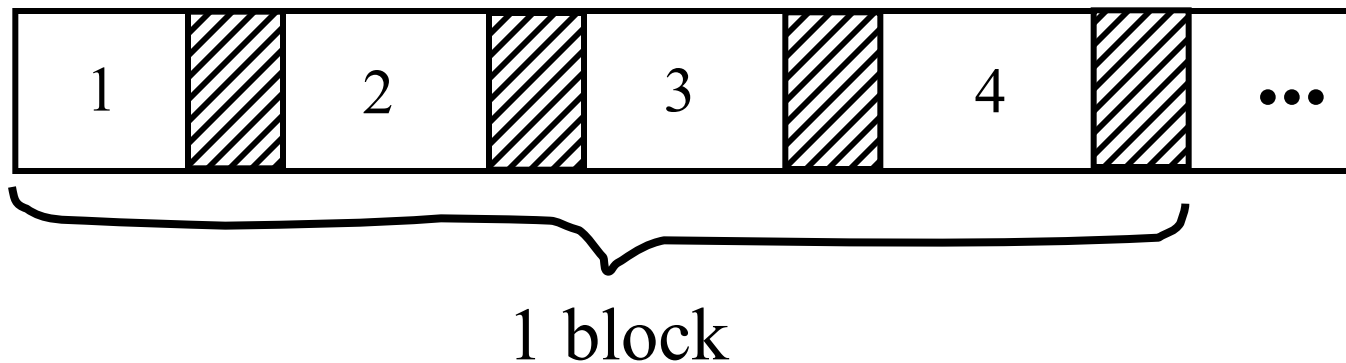
1 block

$$T_4 = 25 + (16.66/2) + (.117) \times 1$$
$$+ (.130) \times 3 = 33.83 \text{ ms}$$

[Compare to $T_1 = 33.45$ ms]

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

T_T = Time to read a full track
      (start at any block)

$T_T = 25 + (0.130/2) + 16.66^* = 41.73$ ms

to get to first block

* Actually, a bit less; do not have to read last gap.

The <u>NEW</u> Megatron 747

- 8 Surfaces, 3.5 Inch diameter
  - outer 1 inch used
- 2^{13} = 8192 Tracks/surface
- 256   Sectors/track
- 2^9 = 512 Bytes/sector

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- 8 GB Disk
- If all tracks have 256 sectors
  - Outermost density: 100,000 bits/inch
  - Inner density: 250,000 bits/inch

1

CS 525

COMPUTER
SCIENCE

Notes 2 - Hardware

39

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
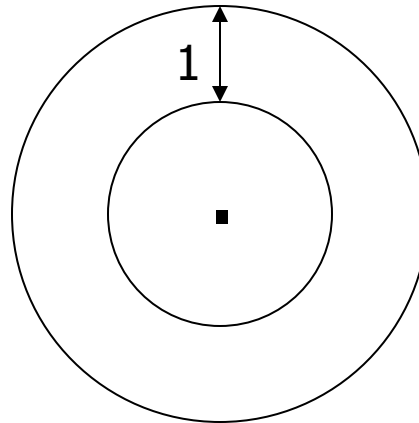
- Outer third of tracks: 320 sectors
- Middle third of tracks: 256
- Inner third of tracks: 192

- Density: 114,000 → 182,000 bits/inch

Timing for <u>new</u> Megatron 747 (Ex 2.3)

- Time to read 4096-byte block:
  - MIN: 0.5 ms
  - MAX: 33.5 ms
  - AVE: 14.8 ms

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

- Hardware: Disks
- Access Times
- Example: Megatron 747 ⟵ here
- Optimizations
- Other Topics
  - Storage Costs
  - Using Secondary Storage
  - Disk Failures

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
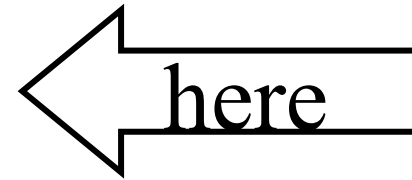
Optimizations (in controller or O.S.)

- Disk Scheduling Algorithms
  - e.g., elevator algorithm
- Track (or larger) Buffer
- Pre-fetch
- Arrays
- Mirrored Disks
- On Disk Cache

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Double Buffering

Problem: Have a File
>> Sequence of Blocks B1, B2

Have a Program
>> Process B1
>> Process B2
>> Process B3
⋮

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Single Buffer Solution

(1) Read B1 → Buffer

(2) Process Data in Buffer

(3) Read B2 → Buffer

(4) Process Data in Buffer …

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Say   P = time to process/block

R = time to read in 1 block

n = # blocks

Single buffer time = n(P+R)

COMPUTER SCIENCE

Notes 2 - Hardware

46

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Double Buffering

process

Memory:

Disk:

A	B	C	D	E	F	G	

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Double Buffering

process

Memory:

A   B

Disk: B   C   D   E   F   G

done

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Double Buffering

process

Memory:

C   B

Disk:

done

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Double Buffering

process   process

Memory:

C   B

Disk:

C D E F G

done done

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Say P ≥ R

P = Processing time/block
R = IO time/block
n = # blocks

What is processing time?

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Say P ≥ R

P = Processing time/block
R = IO time/block
n = # blocks

What is processing time?

- Double buffering time   = R + nP

- Single buffering time   = n(R+P)

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Disk Arrays

- RAIDs (various flavors)
- Block Striping
- Mirrored

logically one disk

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

On Disk Cache

P

... M C ...

cache

cache

Notes 2 - Hardware

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
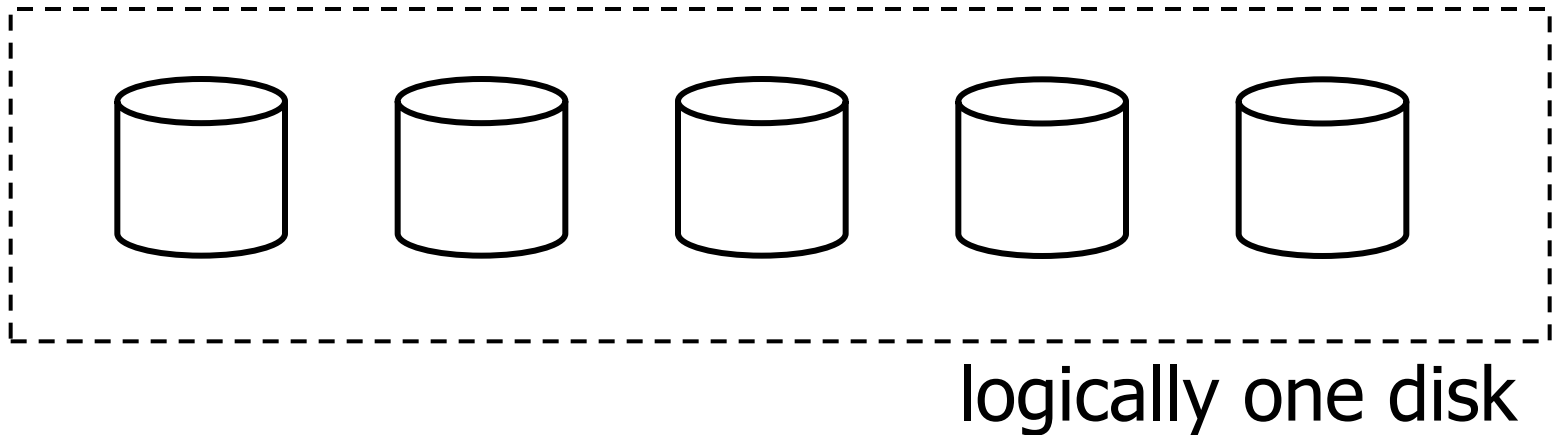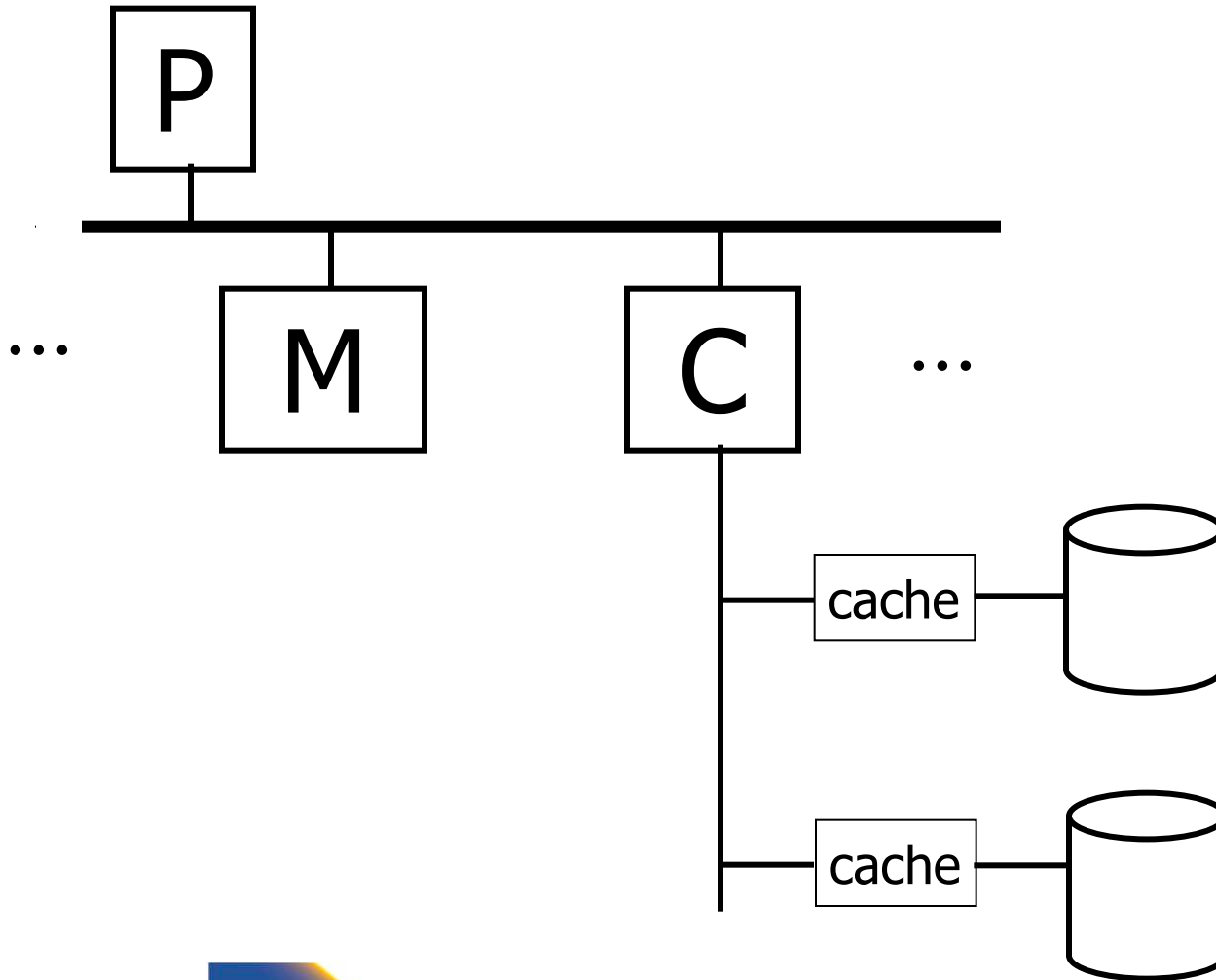
Block Size Selection?

- Big Block → Amortize I/O Cost, Less Management Overhead

Unfortunately...

- Big Block ⇒ Read in more useless stuff!
  and takes longer to read

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Trend

- As memory prices drop, blocks get bigger …

Storage Cost

typical capacity (bytes) vs access time (sec)

- cache — 10^5 bytes, 10^{-9} sec
- electronic main — 10^9 bytes
- electronic secondary — 10^{11} bytes, 10^{-6} sec
- magnetic optical disks — 10^{13} bytes
- online tape — 10^{-3} sec
- nearline tape & optical disks — 10^{15} bytes, 10^{-0} sec
- offline tape — 10^3 sec

y-axis: 10^3, 10^5, 10^7, 10^9, 10^{11}, 10^{13}, 10^{15}

x-axis: 10^{-9}, 10^{-6}, 10^{-3}, 10^{-0}, 10^3

from Gray & Reuter

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Storage Cost

from Gray & Reuter

cache

electronic main

electronic secondary

online tape

magnetic optical disks

nearline tape & optical disks

offline tape

dollars/MB

10^4

10^2

10^0

10^{-2}

10^{-4}

10^{-9} 10^{-6} 10^{-3} 10^{-0} 10^3

access time (sec)

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
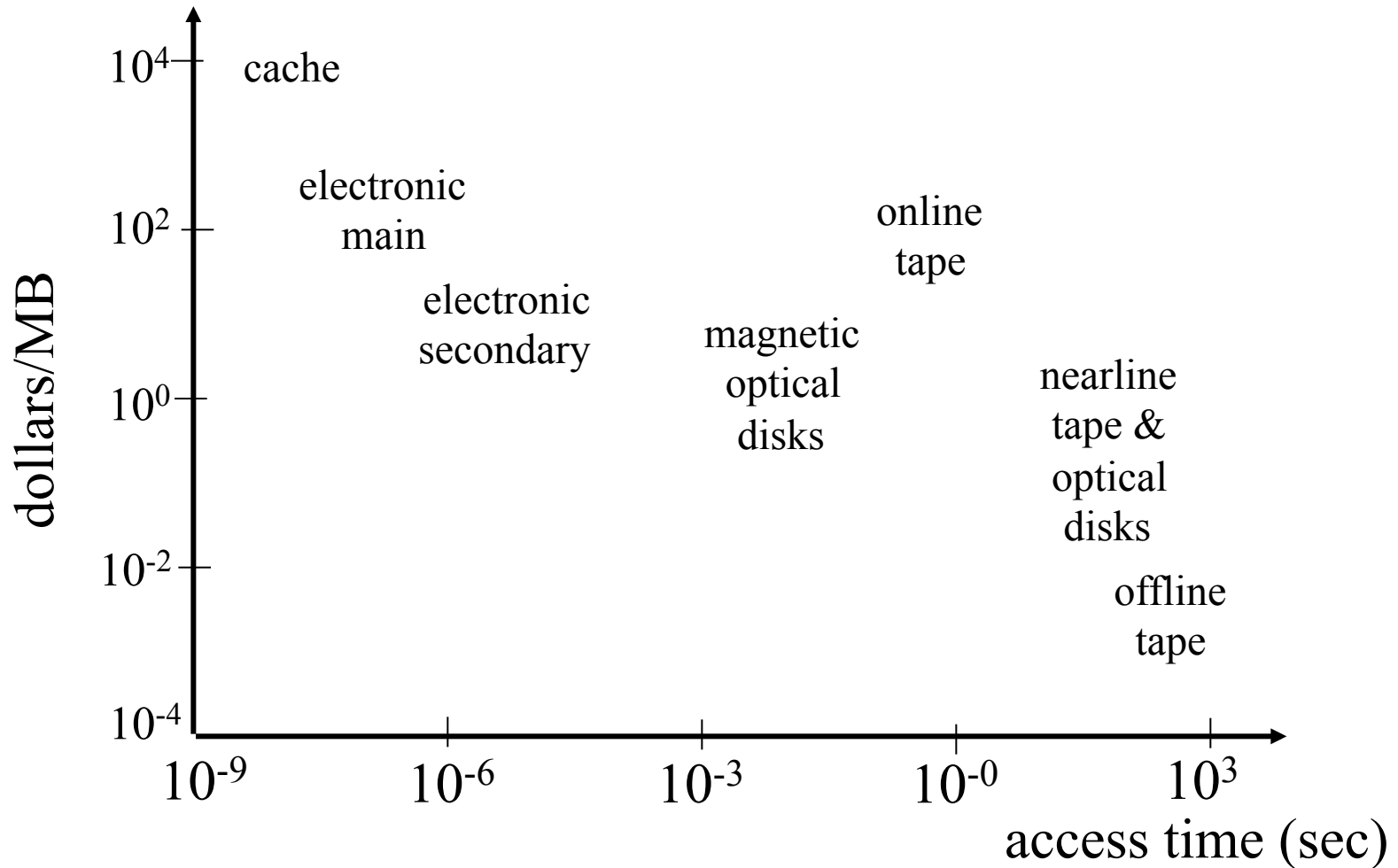
Using secondary storage effectively

- Example: Sorting data on disk
- Conclusion:
  - I/O costs dominate
  - Design algorithms to reduce I/O

- Also: How big should blocks be?

Five Minute Rule

- THE 5 MINUTE RULE FOR TRADING MEMORY FOR DISC ACCESSES
Jim Gray & Franco Putzolu
May 1985

- The Five Minute Rule, Ten Years Later
Goetz Graefe & Jim Gray
December 1997

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Five Minute Rule

- Say a page is accessed every X seconds
- CD = cost if we keep that page on disk
  - $D = cost of disk unit
  - I = numbers IOs that unit can perform per second
  - In X seconds, unit can do XI IOs
  - So   CD = $D / XI

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Five Minute Rule

- Say a page is accessed every X seconds
- CM = cost if we keep that page on RAM
  - $M = cost of 1 MB of RAM
  - P = numbers of pages in 1 MB RAM
  - So   CM = $M / P

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Five Minute Rule

- Say a page is accessed every X seconds
- If CD is smaller than CM,
  - keep page on disk
  - else keep in memory
- Break even point when CD = CM, or

$$X = \frac{\$D \quad P}{I \quad \$M}$$

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Using '97 Numbers

- P = 128 pages/MB (8KB pages)

- I = 64 accesses/sec/disk

- $D = 2000 dollars/disk (9GB + controller)

- $M = 15 dollars/MB of DRAM

- X = 266 seconds (about 5 minutes)
  (did not change much from 85 to 97)

Disk Failures

- Partial → Total
- Intermittent → Permanent

Coping with Disk Failures

- Detection
  - e.g. Checksum

- Correction
  $\Rightarrow$ Redundancy

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

At what level do we cope?

- Single Disk
  - e.g., Error Correcting Codes
- Disk Array

Logical   Physical

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
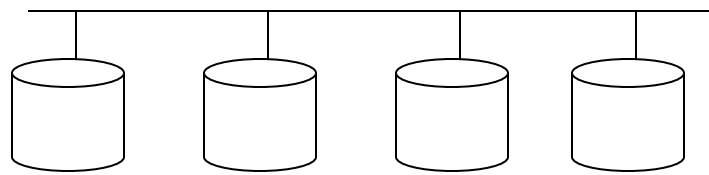
Operating System

e.g., Stable Storage
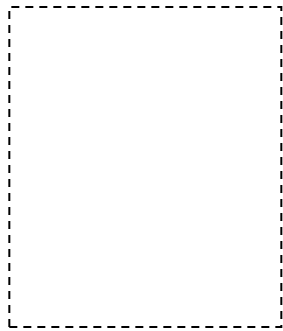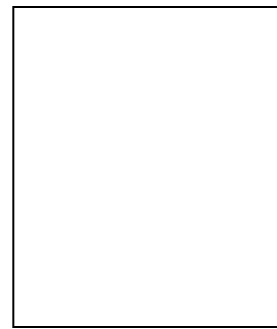
Logical Block   Copy A   Copy B
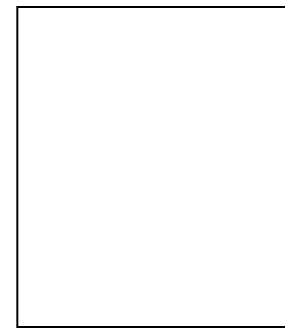
→ Database System

- e.g.,
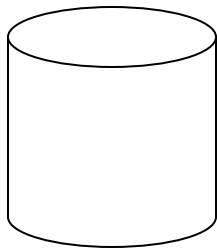
Current DB   Log   Last week's DB

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
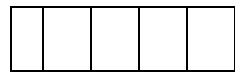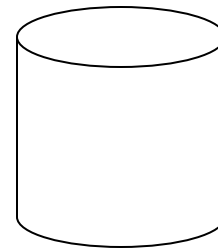
Summary

- Secondary storage, mainly disks
- I/O times + formulas
  - Sequential vs. random
- I/Os should be avoided,
  especially random ones…..
- OS optimizations
- Disk errors

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

- Hardware: Disks
- Access Times
- Example: Megatron 747
- Optimizations
- Other Topics
  - Storage Costs
  - Using Secondary Storage
  - Disk Failures

here ⟵

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
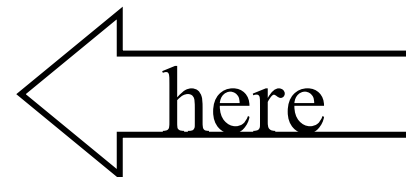
Outlook - Hardware

- Disk Access is the main limiting factor

- However, to implement fast DBMS
  - need to understand other parts of the hardware
    - Memory hierarchy
    - CPU architecture: pipelining, vector instructions, OOE, ...
    - SSD storage
  - need to understand how OS manages hardware
    - File access, VM, Buffering, ...

Memory Hierarchy

CPU Register
(< 1KB, 1 cycle)

L1 Cache (10 KB's, few cycles)

L2 Cache (e.g., 512 KB, 2-10 x L1)

L3 Cache (MB)

Main Memory (GB, 100's cycles)
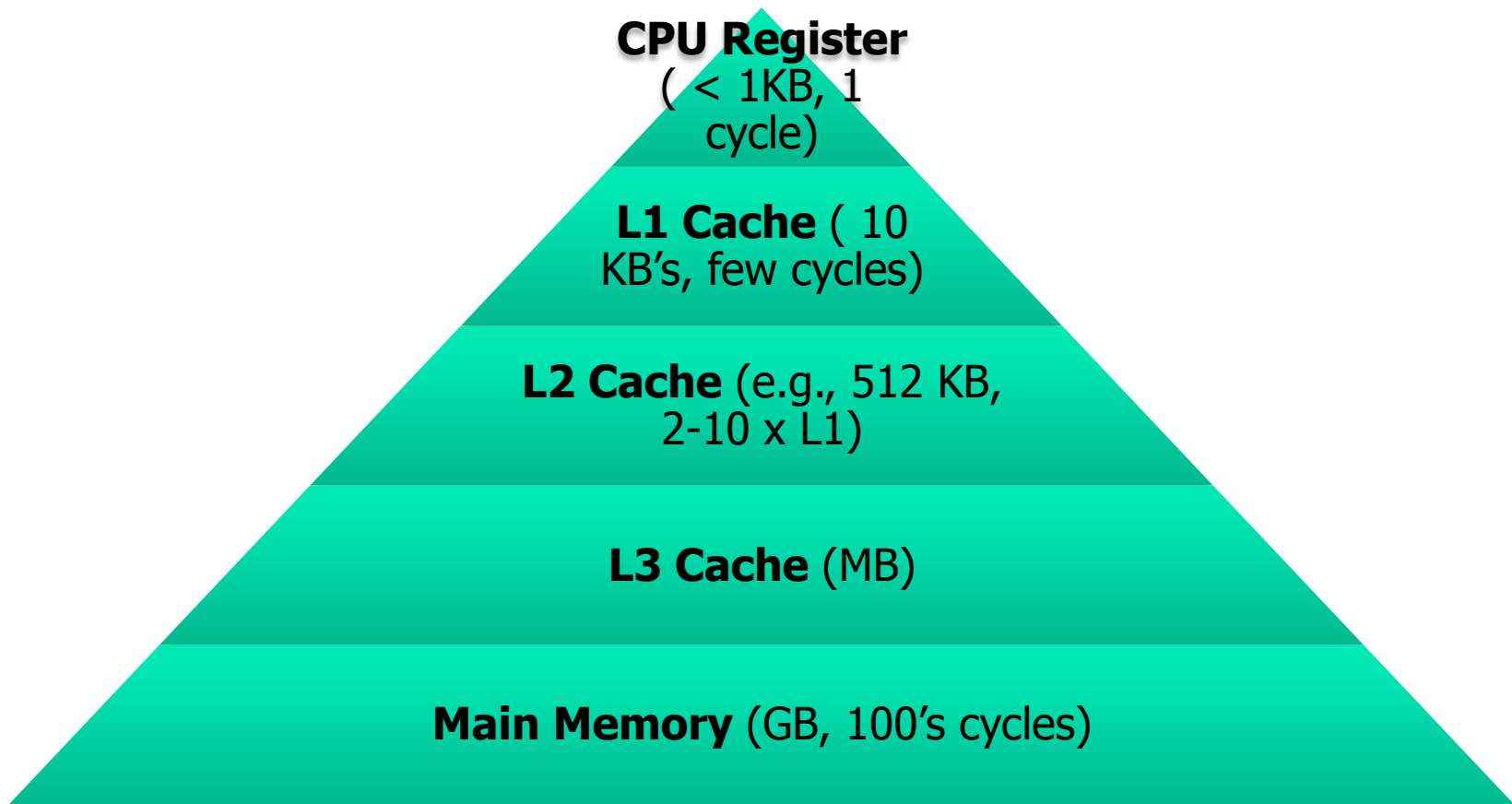
Memory Hierarchy

- **Compare**: Disk vs. Main Memory
- Reduce accesses to main memory
- Cache conscious algorithms

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Increasing Amount of Parallelism

- Contention on, e.g., Memory
- NUMA
- Algorithmic Challenges
  - How to parallelize algorithms?
  - Sometime: Completely different approach required
  - -> Rewrite large parts of DBMS

New Trend: Software/Hardware Co-design

- Actually, revived trend: database machines (80's)

- New goals: power consumption

- Design specific hardware and write special software for it

- E.g., Oracle Exadata, Oracle Labs

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

CS 525: Advanced Database Organization
03: **Disk Organization**

Boris Glavic

Slides: adapted from a course taught by Hector Garcia-Molina, Stanford InfoLab

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Topics for today

- How to lay out data on disk
- How to move it to/from memory

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What are the data items we want to store?

- a salary

- a name

- a date

- a picture

What are the data items we want to store?

- a salary

- a name

- a date

- a picture

⇒ What we have available: <u>Bytes</u>

← 8 →
bits

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
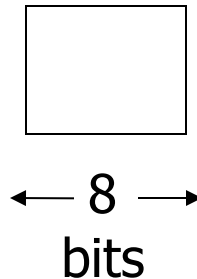
To represent:

- ## Integer (short): 2 bytes

  e.g., 35 is

00000000	00100011

  Endian! Could as well be

00100011	00000000

- ## Real, floating point
  n bits for mantissa, m for exponent....

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

To represent:

- Characters

  $\longrightarrow$ various coding schemes suggested,

  most popular is ASCII (1 byte encoding)

  Example:
  A: 1000001
  a: 1100001
  5:     0110101
  LF: 0001010

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

To represent:

- Boolean

  e.g., TRUE   1111 1111

        FALSE   0000 0000

- Application specific

  e.g., enumeration

       RED → 1   GREEN → 3

       BLUE → 2   YELLOW → 4 ...

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

To represent:

- Boolean

  e.g., TRUE   1111 1111

       FALSE   0000 0000

- Application specific

  e.g., RED → 1   GREEN → 3

       BLUE → 2   YELLOW → 4 ...

⇒ Can we use less than 1 byte/code?

    Yes, but only if desperate...

To represent:

- Dates

  e.g.: - Integer, # days since Jan 1, 1900

  - 8 characters, YYYYMMDD

  - 7 characters, YYYYDDD

  (not YYMMDD! Why?)

- Time

  e.g. - Integer, seconds since midnight

  - characters, HHMMSSFF

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

To represent:

- String of characters
  - Null terminated

    e.g.,

c	a	t	✕		

  - Length given

    e.g.,

3	c	a	t	✕		

  - Fixed length

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
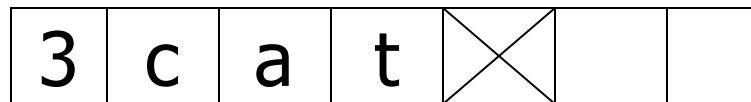
To represent:

- Bag of bits

Length	Bits

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Key Point

- Fixed length items

- Variable length items
  - usually length given at beginning

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Also

- Type of an item: Tells us how to
  interpret
  (plus size if fixed)

Overview

Data Items

↓

Records

↓

Blocks

↓

Files

⋮

Memory

Record - Collection of related data items (called FIELDS)

E.g.: Employee record:

      name field,

      salary field,

      date-of-hire field, ...

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Types of records:

- Main choices:
  - FIXED vs VARIABLE FORMAT
  - FIXED vs VARIABLE LENGTH

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Fixed format

A <u>SCHEMA</u> (not record) contains
  following information
        - # fields

        - type of each field

        - order in record

        - meaning of each field

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: fixed format and length

Employee record
    (1) E#, 2 byte integer
    (2) E.name, 10 char.
    (3) Dept, 2 byte code

Schema

| 55 | s m i t h | 02 |

| 83 | j o n e s | 01 |

Records

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Variable format

- Record itself contains format "Self Describing"

Notes 3

19

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Example: variable format and length

| 2 | 5 | I | 46 | 4 | S | 4 | F | O | R | D |

- # Fields
- Code identifying field as E#
- Integer type
- Code for Ename
- String type
- Length of str.
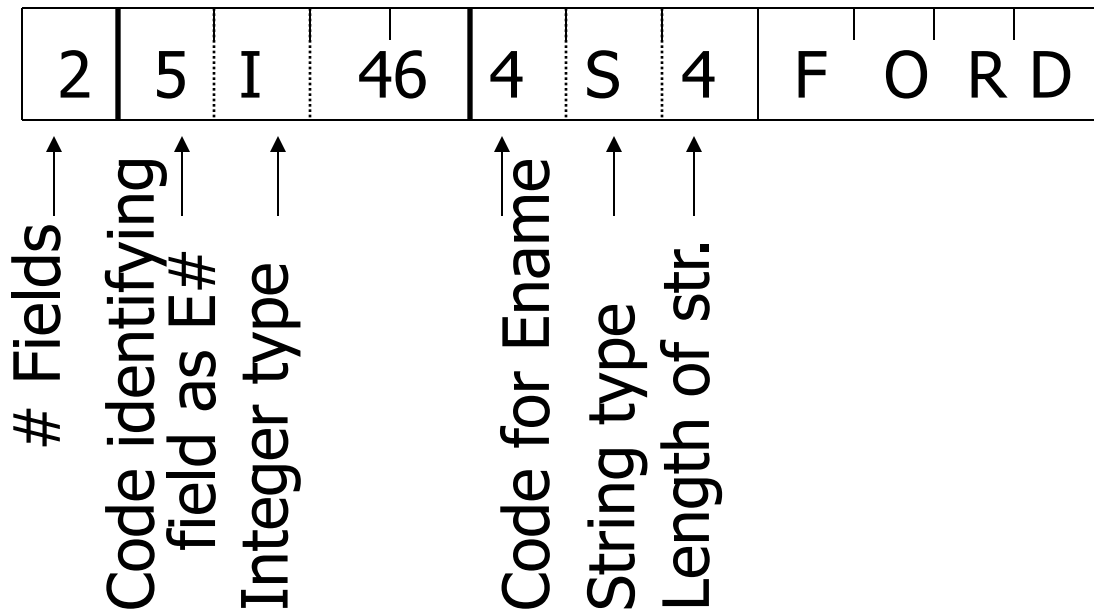
Field name codes could also be strings, i.e. TAGS

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Variable format useful for:

- "sparse" records
- repeating fields
- evolving formats

············▸

But may waste space...

Additional indirection...

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

- <u>EXAMPLE:</u> var format record with repeating fields

Employee $\rightarrow$ one or more $\rightarrow$ children

| 3 | E_name: Fred | Child: Sally | Child: Tom |

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Note: Repeating fields does not imply
- variable format, nor
- variable size

John	Sailing	Chess	--

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Note: Repeating fields does not imply
- variable format, nor
- variable size

John	Sailing	Chess	--

- Key is to allocate maximum number of repeating fields (if not used $\rightarrow$ null)

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

☆ Many variants between fixed - variable format:

Example: Include <u>record type</u> in record

5	27	. . . .

record type   record length

tells me what

to expect

(i.e. points to schema)

Record header - data at beginning that describes record

May contain:

- record type

- record length

- time stamp

- null-value bitmap

- other stuff ...

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Other interesting issues:

- Compression
  - within record - e.g. code selection
  - collection of records - e.g. find common patterns

- Encryption

- Splitting of large records
  - E.g., image field, store pointer

COMPUTER
SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Record Header – null-map

- SQL: NULL is special value for every data type
  - Reserve one value for each data type as NULL?

- Easier solution
  - Record header has a bitmap to store whether field is NULL
  - Only store non-NULL fields in record

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Separate Storage of Large Values

- Store fields with large values separately
  - E.g., image or binary document
  - Records have pointers to large field content

- Rationale
  - Large fields mostly not used in search conditions
  - Benefit from smaller records
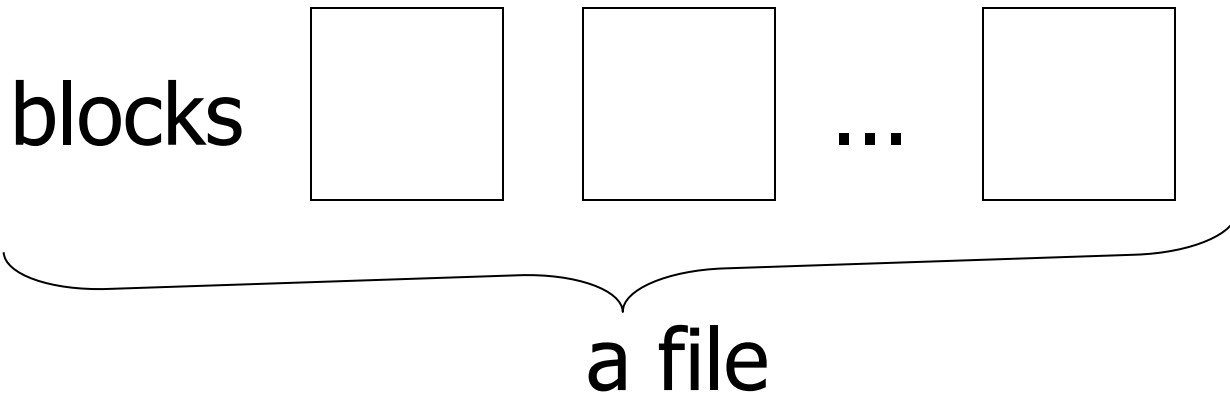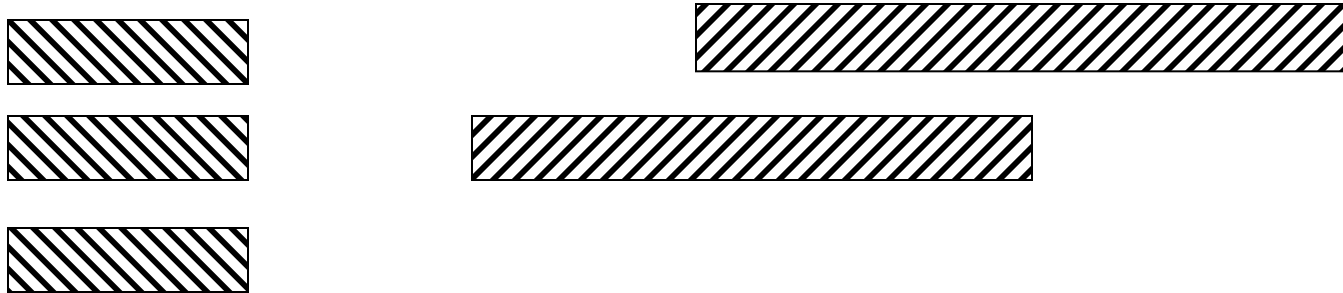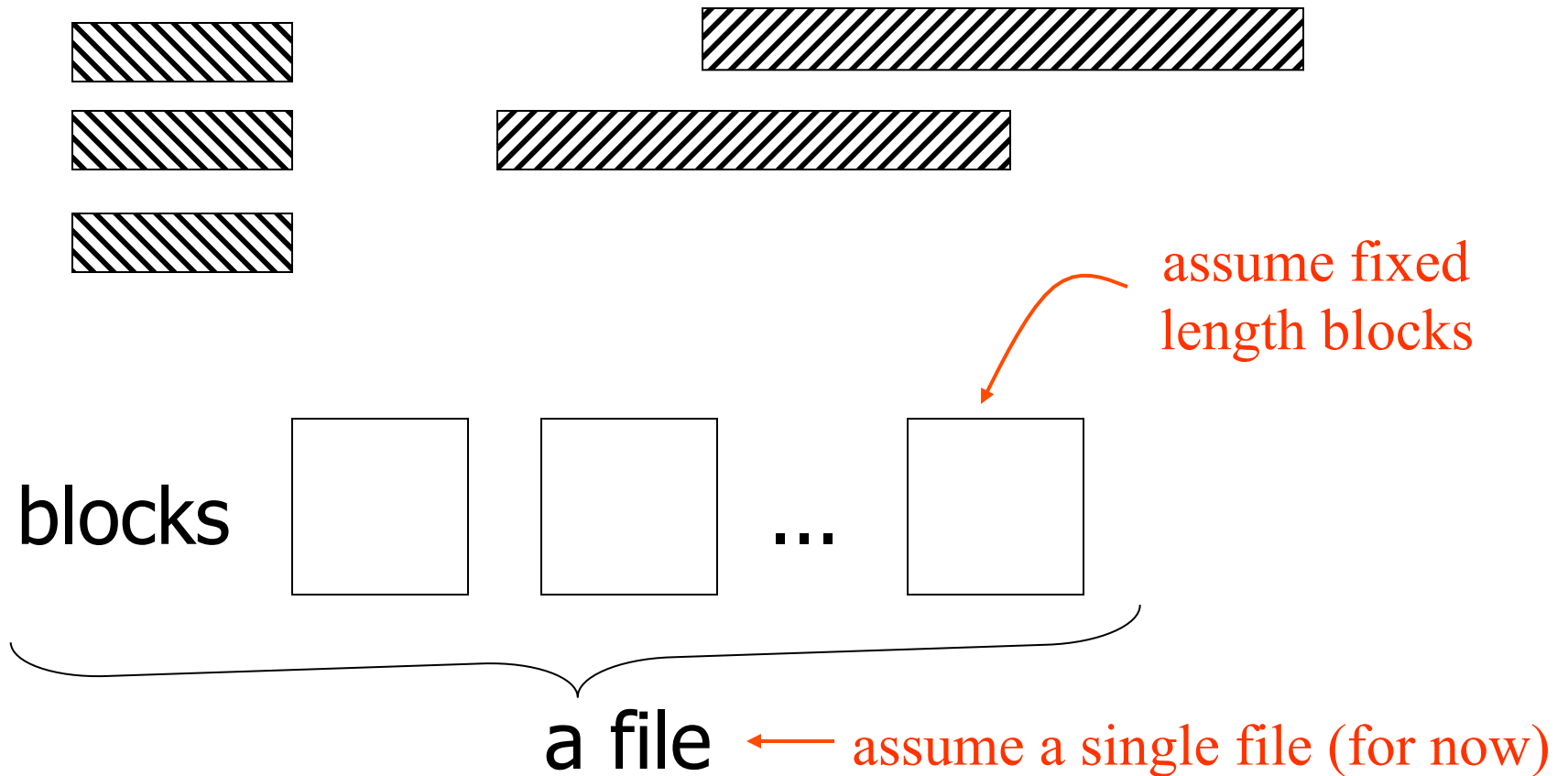
COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Next: placing records into blocks

blocks

...

a file

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Next: placing records into blocks

assume fixed
length blocks

blocks [ ] [ ] ... [ ]

a file ← assume a single file (for now)

Options for storing records in blocks:

(1) separating records

(2) spanned vs. unspanned

(3) sequencing

(4) indirection

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(1) Separating records

Block   R1   R2   R3
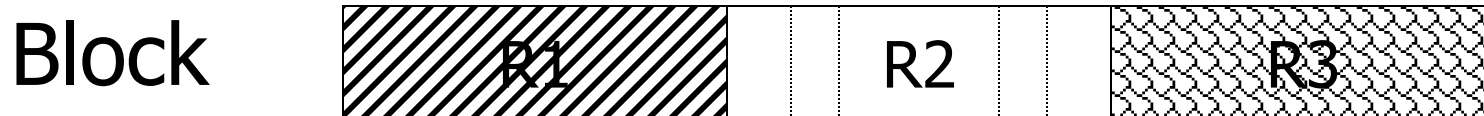
(a) no need to separate - fixed size recs.

(b) special marker

(c) give record lengths (or offsets)

- within each record

- in block header

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

(2) Spanned vs. Unspanned

- Unspanned: records must be within one block
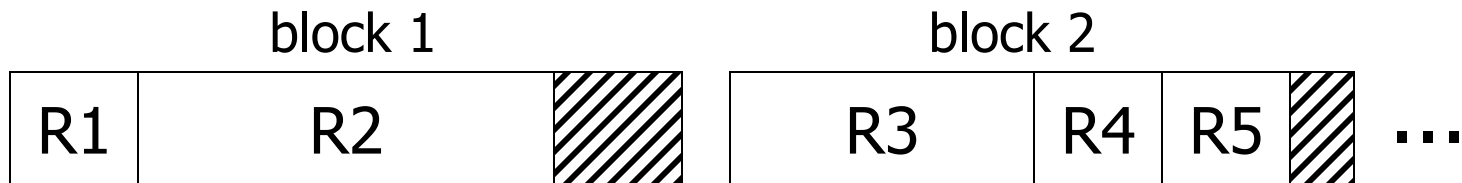
- Spanned
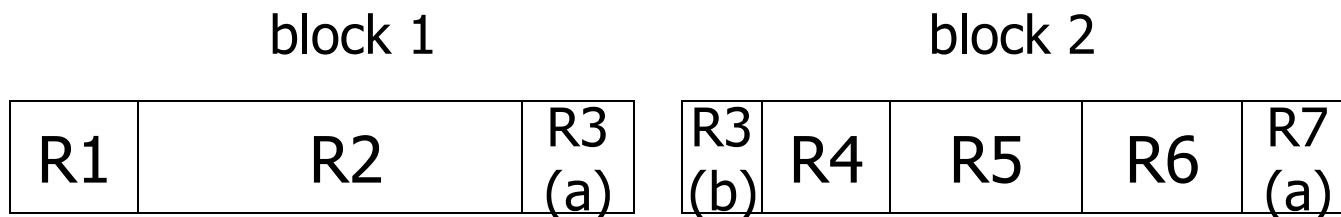
With spanned records:

R1	R2	R3 (a)

R3 (b)	R4	R5	R6	R7 (a)
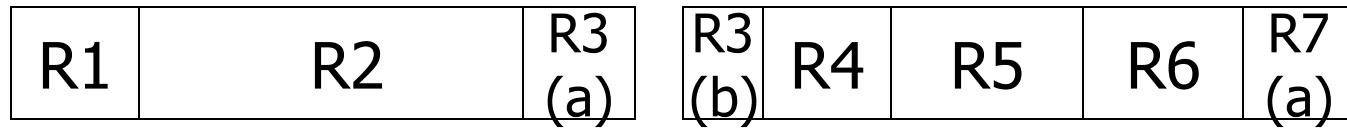
need indication
of partial record
"pointer" to rest

need indication
of continuation
(+ from where?)

Spanned vs. unspanned:

- Unspanned is <u>much</u> simpler, but may waste space…
- Spanned essential if

    record size > block size

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(3) Sequencing

- Ordering records in file (and block) by some key value

  <u>Sequential file</u> ($\Rightarrow$ sequenced)

Why sequencing?

Typically to make it possible to efficiently read records in order

(e.g., to do a merge-join — discussed later)

Sequencing Options

(a) Next record physically contiguous

R1	Next (R1)	$\cdots$

(b) Linked

R1		Next (R1)	

CS 525

COMPUTER SCIENCE

Notes 3

39

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
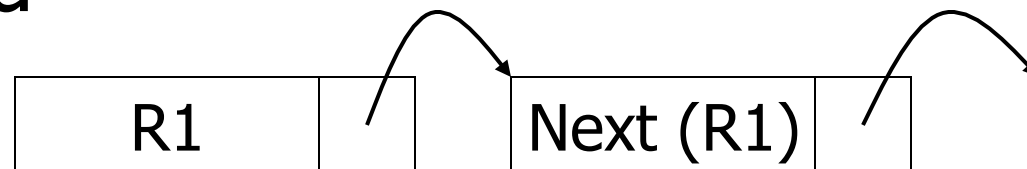
Sequencing Options

(c)   Overflow area

Records

in sequence

R1
R2
R3
R4
R5
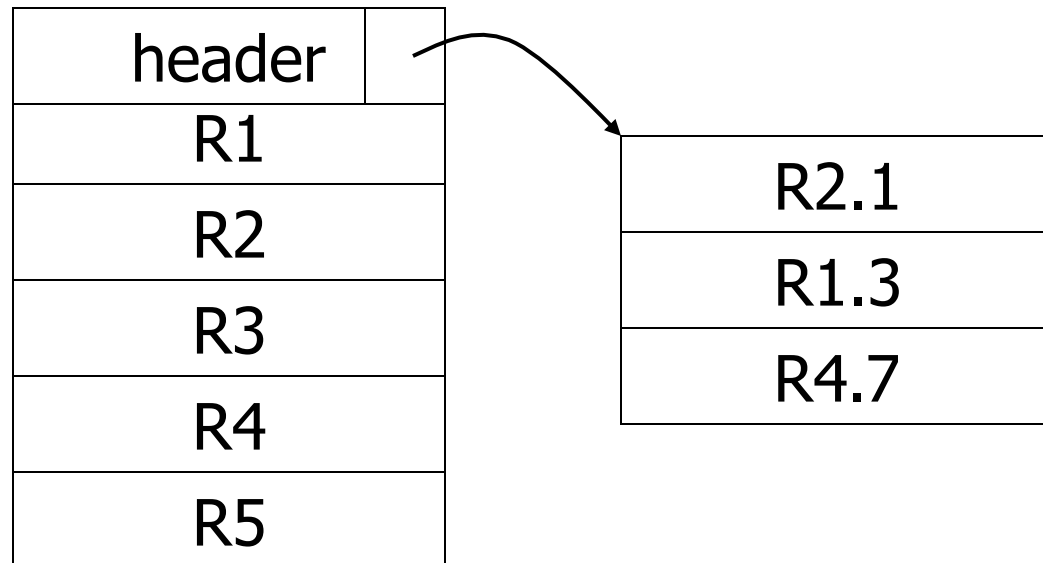
Notes 3

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Sequencing Options

(c)   Overflow area

Records
in sequence

header	
R1	
R2	
R3	
R4	
R5	

R2.1
R1.3
R4.7

(4) Indirection

- How does one refer to records?

$$\longrightarrow \boxed{\qquad Rx \qquad}$$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

(4) Indirection

- How does one refer to records?

$$\longrightarrow \boxed{\text{Rx}}$$
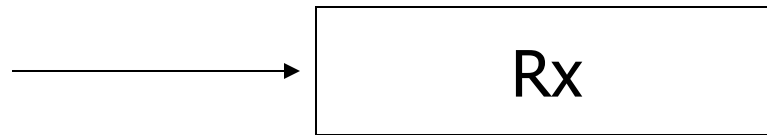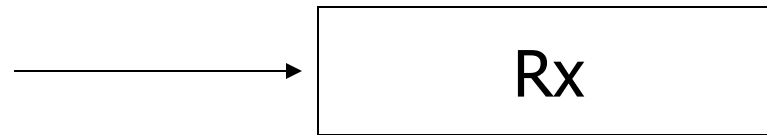
Many options:

Physical ⟷ Indirect

☆ Purely Physical

E.g., Record
Address =
or ID

$\left\{\begin{array}{l}\text{Device ID}\\ \text{Cylinder \#}\\ \text{Track \#}\\ \text{Block \#}\\ \text{Offset in block}\end{array}\right.$

Block ID

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
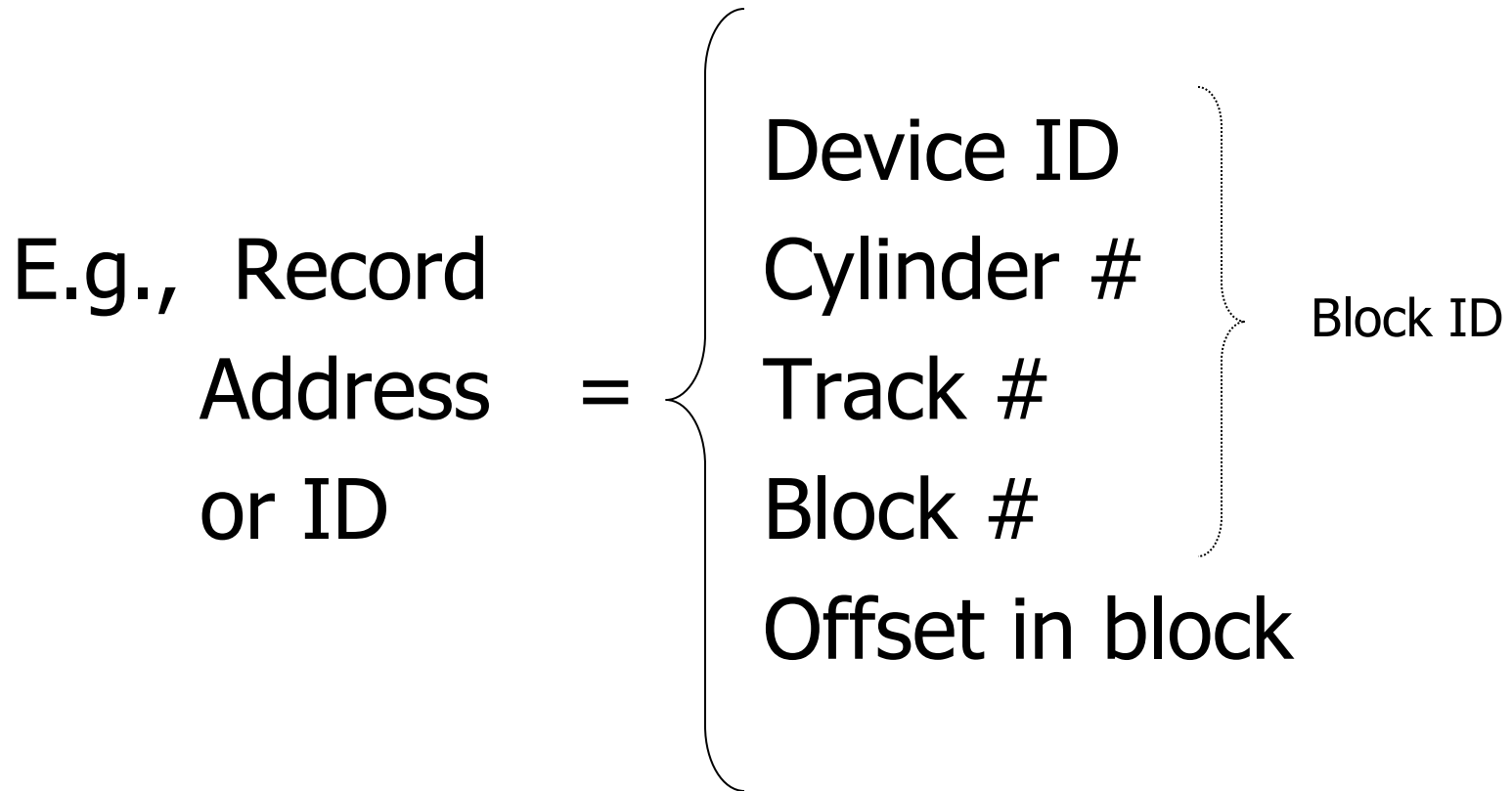
☆ Fully Indirect

E.g., Record ID is arbitrary bit string

map

rec ID

r

Rec ID	Physical addr.

address

a

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
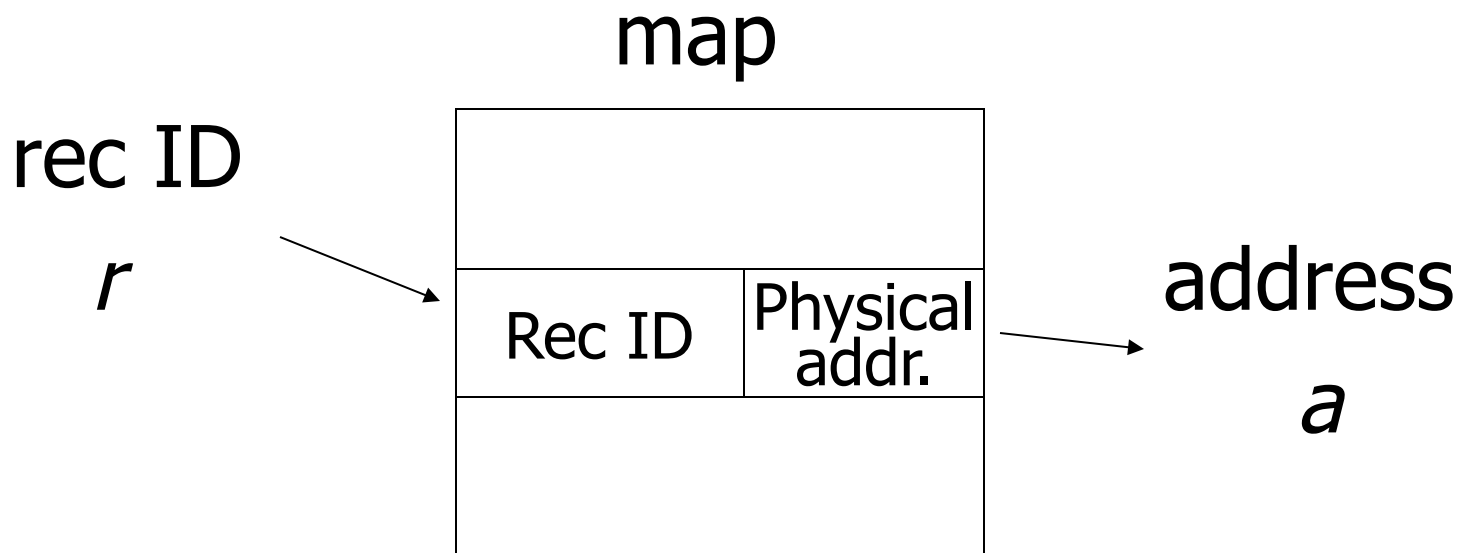
Tradeoff

Flexibility $\longleftrightarrow$ Cost

to move records   of indirection

(for deletions, insertions)

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Physical ⟵⟶ Indirect

↑

Many options
in between ...

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

<u>Block header</u> - data at beginning that describes block

May contain:

- File ID (or RELATION or DB ID)
- This block ID
- Record directory
- Pointer to free space
- Type of block (e.g. contains recs type 4; is overflow, …)
- Pointer to other blocks "like it"
- Timestamp …

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Indirection in block
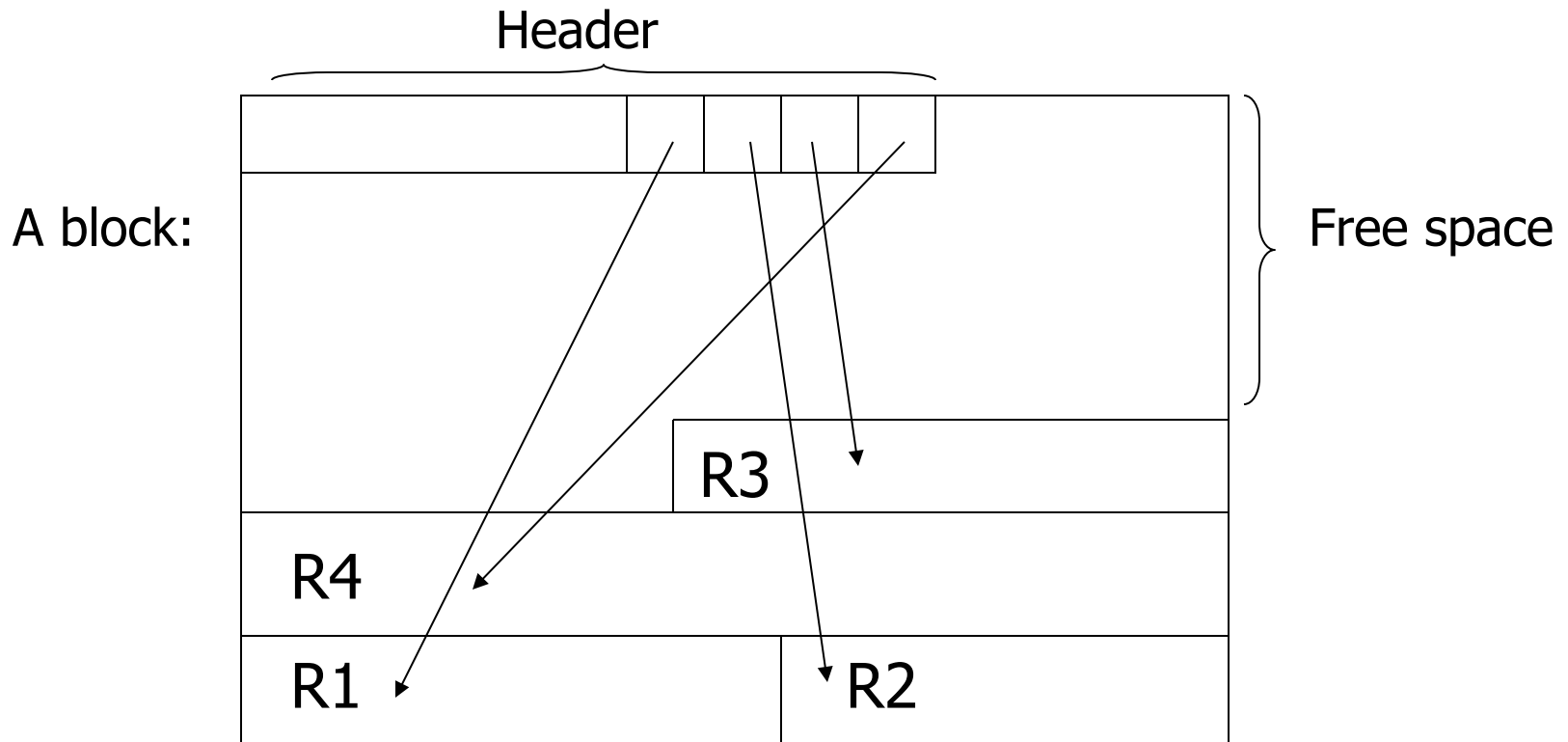
Tuple Identifier (TID)

- TID is
  - Page identifier
  - Slot number
- Slot stores either record or pointer (TID)
- TID of a record is fixed for all time

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

TID Operations

- Insertion
  - Set TID to record location (page, slot)
- Moving record
  - e.g., update variable-size or reorganization
  - Case 1: TID points to record
    - Replace record with pointer (new TID)
  - Case 2: TID points to pointer (TID)
    - Replace pointer with new pointer

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

TID: Block 1, Slot 2

Block 1

Block 2

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
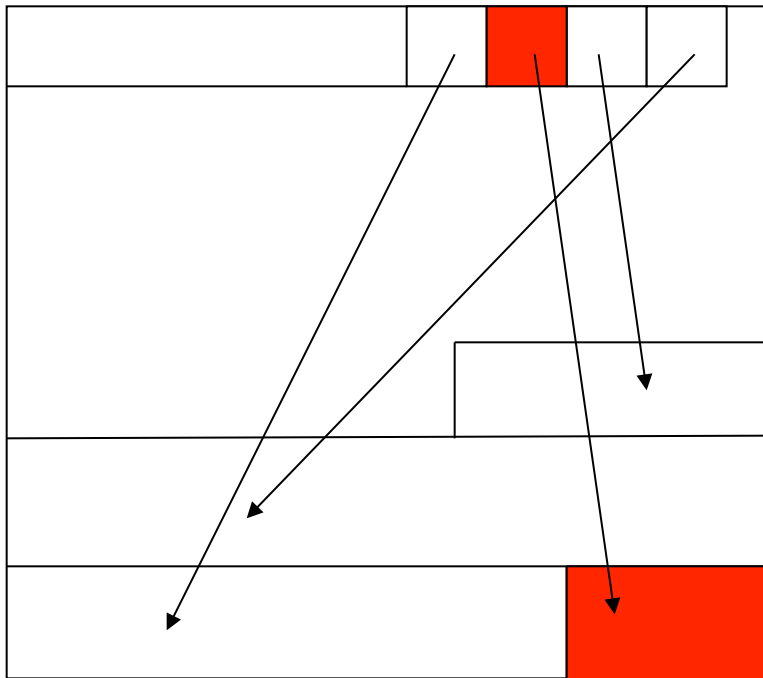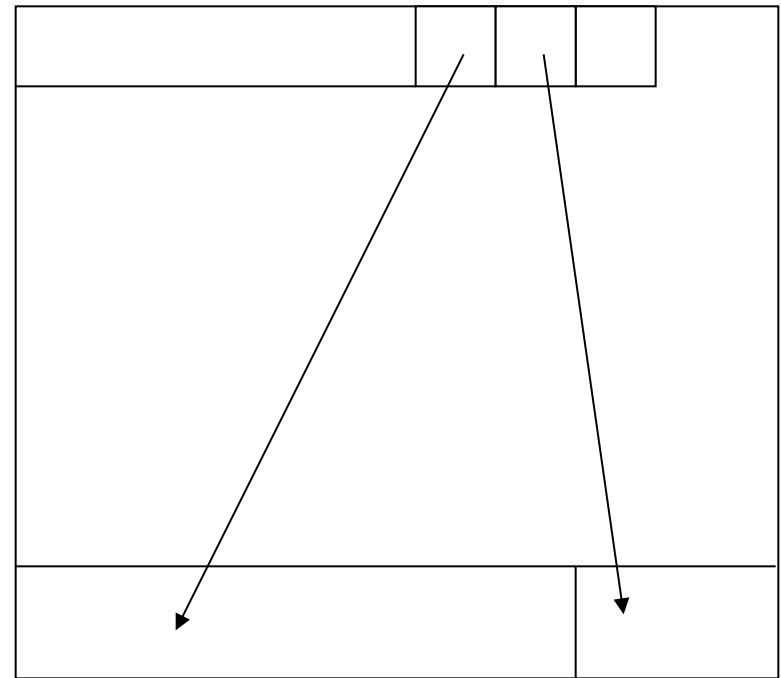
Move record to Block 2 slot 3 -> TID does not change!

TID: Block 1, Slot 2

Block 1

Block 2

Block 2, Slot 3

COMPUTER
SCIENCE

Notes 3

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
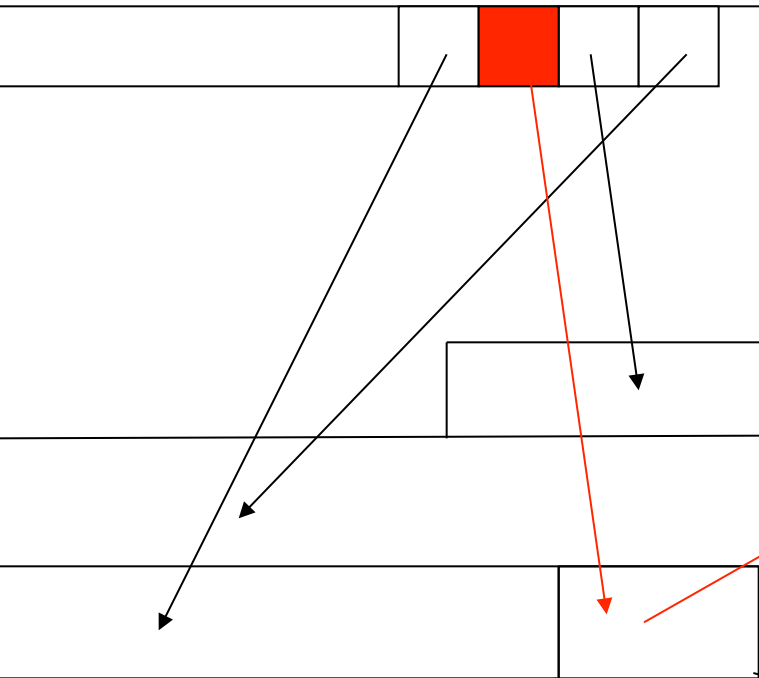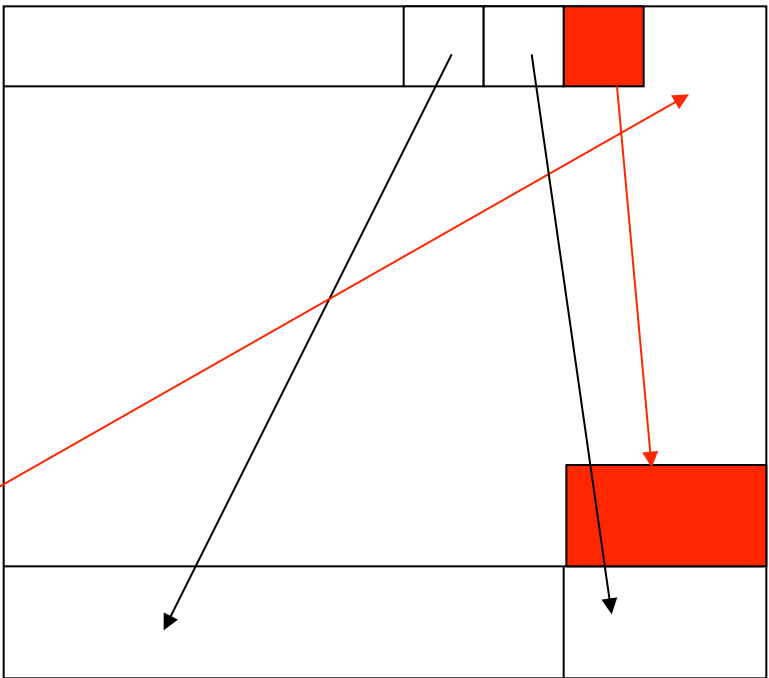
Move record again to Block 2 slot 2 -> still one level of indirection

TID: Block 1, Slot 2

Block 1
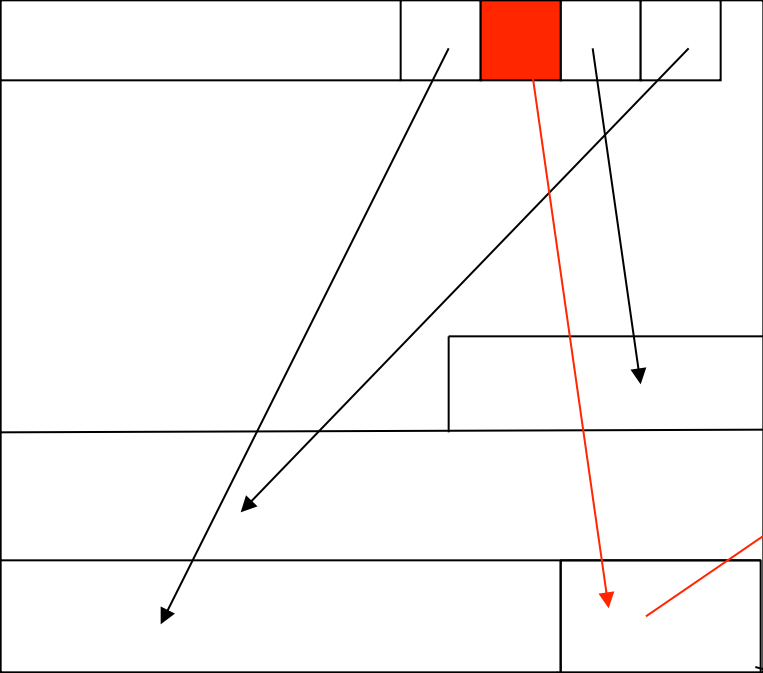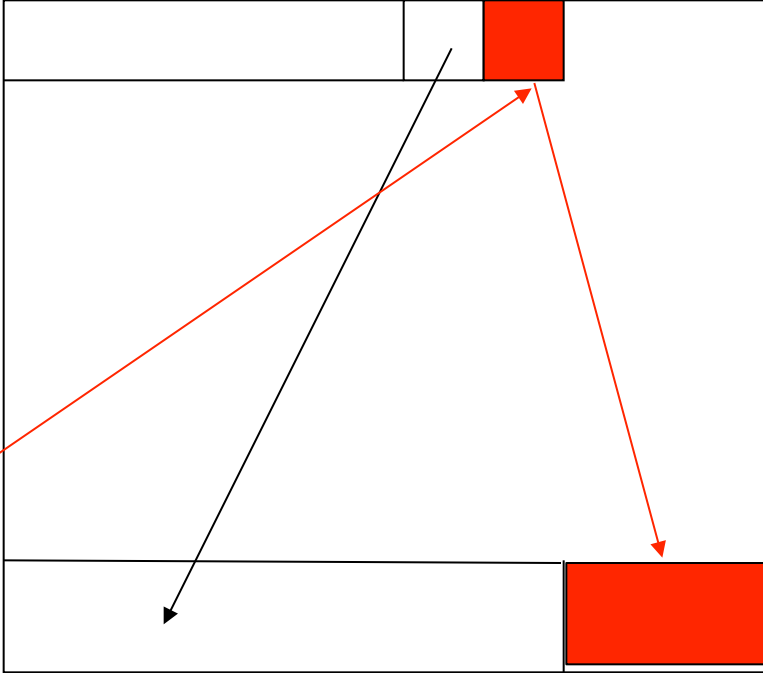
Block 2

Block 2, Slot 2

TID Properties

- TID of record never changes
  - Can be used safely as pointer to record (e.g., in index)
- At most one level of indirection
  - Relatively efficient
  - Changes to physical address - changing max 2 pages

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Options for storing records in blocks:

(1) separating records

(2) spanned vs. unspanned

(3) sequencing

(4) indirection

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Other Topics

(1) Insertion/Deletion

(2) Buffer Management

(3) Comparison of Schemes

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion

Block
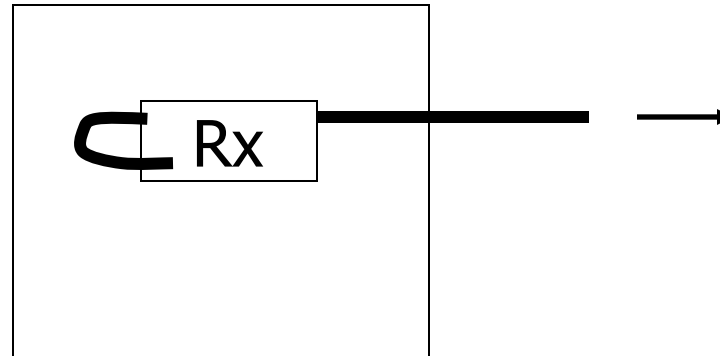
Options:

(a)   Immediately reclaim space
(b)   Mark deleted

Options:

(a)   Immediately reclaim space

(b)   Mark deleted

   – May need chain of deleted records

      (for re-use)

   – Need a way to mark:

      • special characters

      • delete field

      • in map

☆ As usual, many tradeoffs...

- How expensive is it to move valid record to free space for immediate reclaim?

- How much space is wasted?
  - e.g., deleted records, delete fields, free space chains,...

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Concern with deletions

Dangling pointers

Notes 3

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
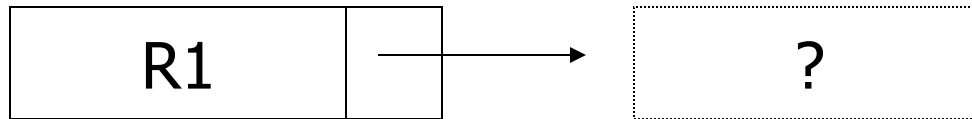
Solution #1: Do not worry

Solution #2: Tombstones

E.g., Leave "MARK" in map or old location

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Solution #2: Tombstones

E.g., Leave "MARK" in map or old location

- Physical IDs

A block

This space never re-used

This space can be re-used

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
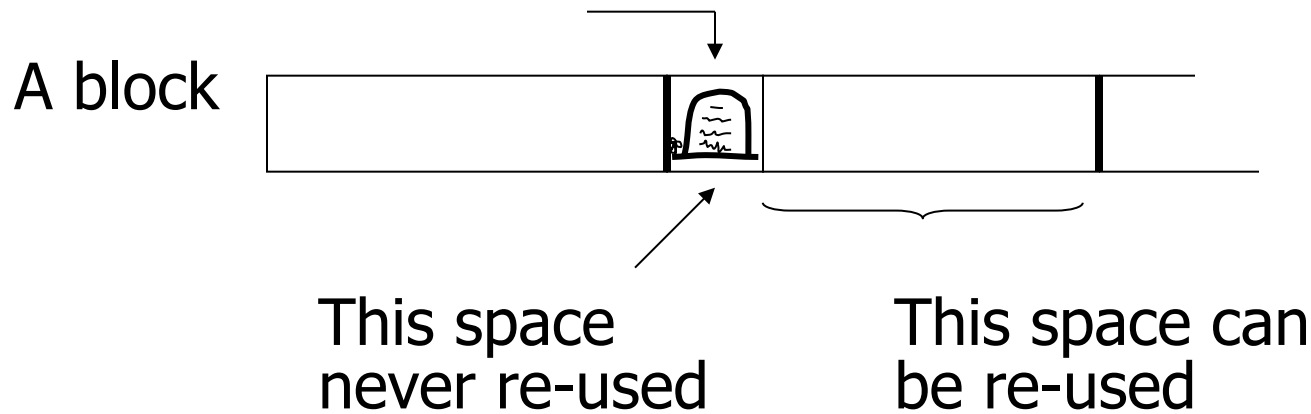
Solution #2: Tombstones

E.g., Leave "MARK" in map or old location

- Logical IDs

map

ID	LOC
7788	🪦

Never reuse ID 7788 nor space in map...

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Insert

Easy case: records not in sequence

→ Insert new record at end of file or in deleted slot

→ If records are variable size, not as easy...

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Insert

Hard case: records in sequence

→ If free space "close by", not too bad...

→ Or use overflow idea...

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Interesting problems:

- How much free space to leave in each block, track, cylinder?

- How often do I reorganize file + overflow?

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Free space

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
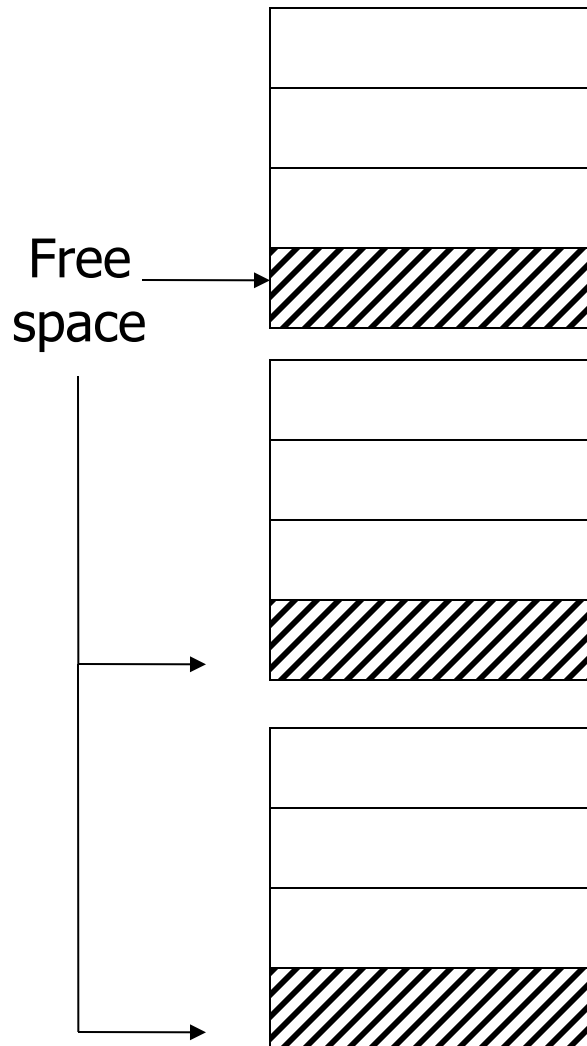
Buffer Management

- For Caching of Disk Blocks
- Buffer Replacement Strategies
  - E.g., LRU, clock
- Pinned blocks
- Forced output   - - - - - - - - - - - - - - → in Notes02
- Double buffering
- Swizzling

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Buffer Manager

- Manages blocks cached from disk in main memory
- Usually -> fixed size buffer (M pages)
- DB requests page from Buffer Manager
  - Case 1: page is in memory -> return address
  - Case 2: page is on disk -> load into memory, return address

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Goals

- Reduce the amount of I/O
- Maximize the *hit rate*
  - Ratio of number of page accesses that are fulfilled without reading from disk
- -> Need strategy to decide when to

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Buffer Manager Organization

- Bookkeeping
  - Need to map (hash table) page-ids to locations in buffer (**page frames**)
  - Per page store *fix count, dirty bit, ...*
  - Manage free space

- Replacement strategy
  - If page is requested but buffer is full
  - Which page to emit remove from buffer

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

FIFO

- **F**irst **I**n, **F**irst **O**ut

- Replace page that has been in the buffer for the longest time

- Implementation: E.g., pointer to oldest page (circular buffer)

  - Pointer->next = Pointer++ % M

- Simple, but not prioritizing frequently accessed pages

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

LRU

- Least Recently Used
- Replace page that has not been accessed for the longest time
- Implementation:
  - List, ordered by LRU
  - Access a page, move it to list tail
- Widely applied and reasonable performance

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Clock

- Frames are organized clock-wise
- Pointer S to current frame
- Each frame has a reference bit
  - Page is loaded or accessed -> bit = 1
- Find page to replace (advance pointer)
  - Return first frame with bit = 0
  - On the way set all bits to 0

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Clock Example

Reference bit

S →

0	**Page 0**
1	Page 1
1	Page 2
0	Page 3
1	Page 4

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Other Replacement Strategies

- LRU-K
- GCLOCK
- Clock-Pro
- ARC
- LFU

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Swizzling

Notes 3

Swizzling

Memory | Disk

block 1

block 2   Rec A

block 1

block 2   Rec A

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
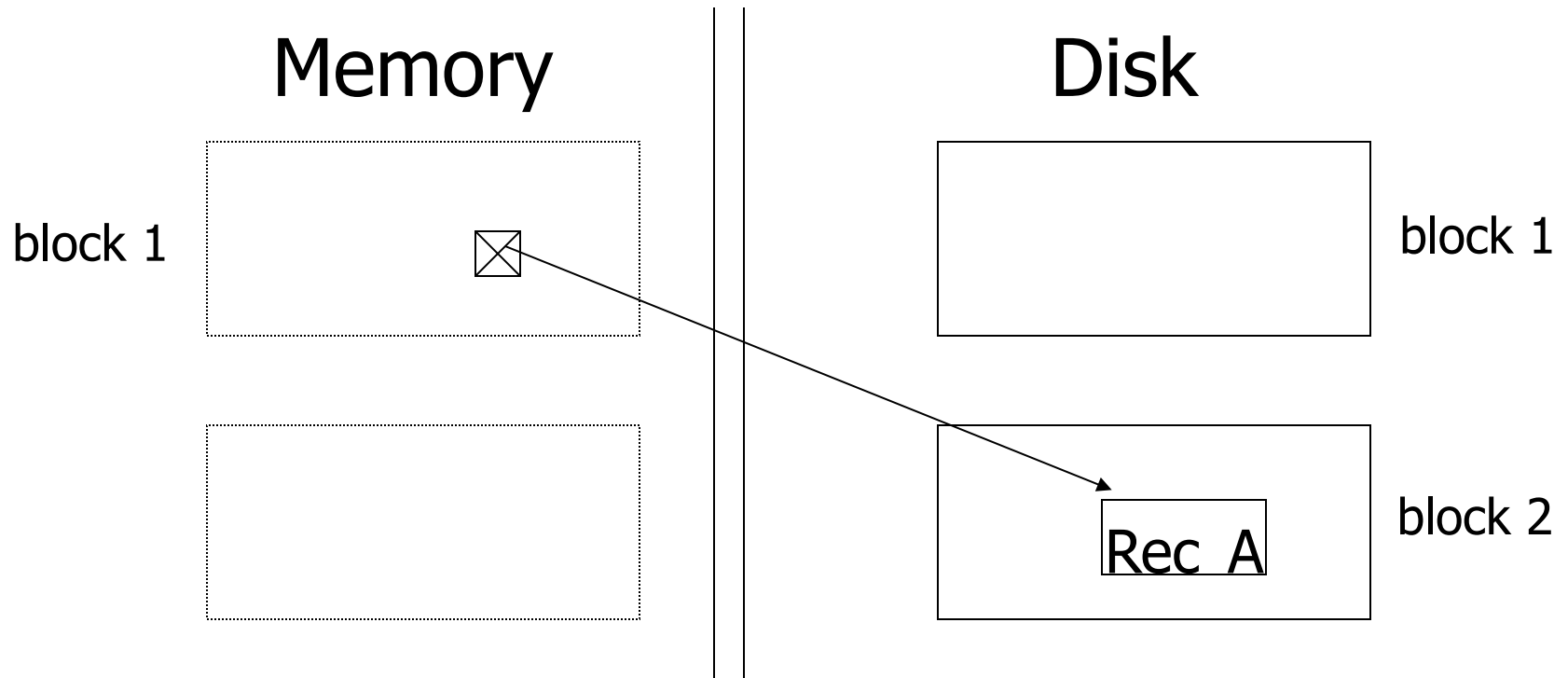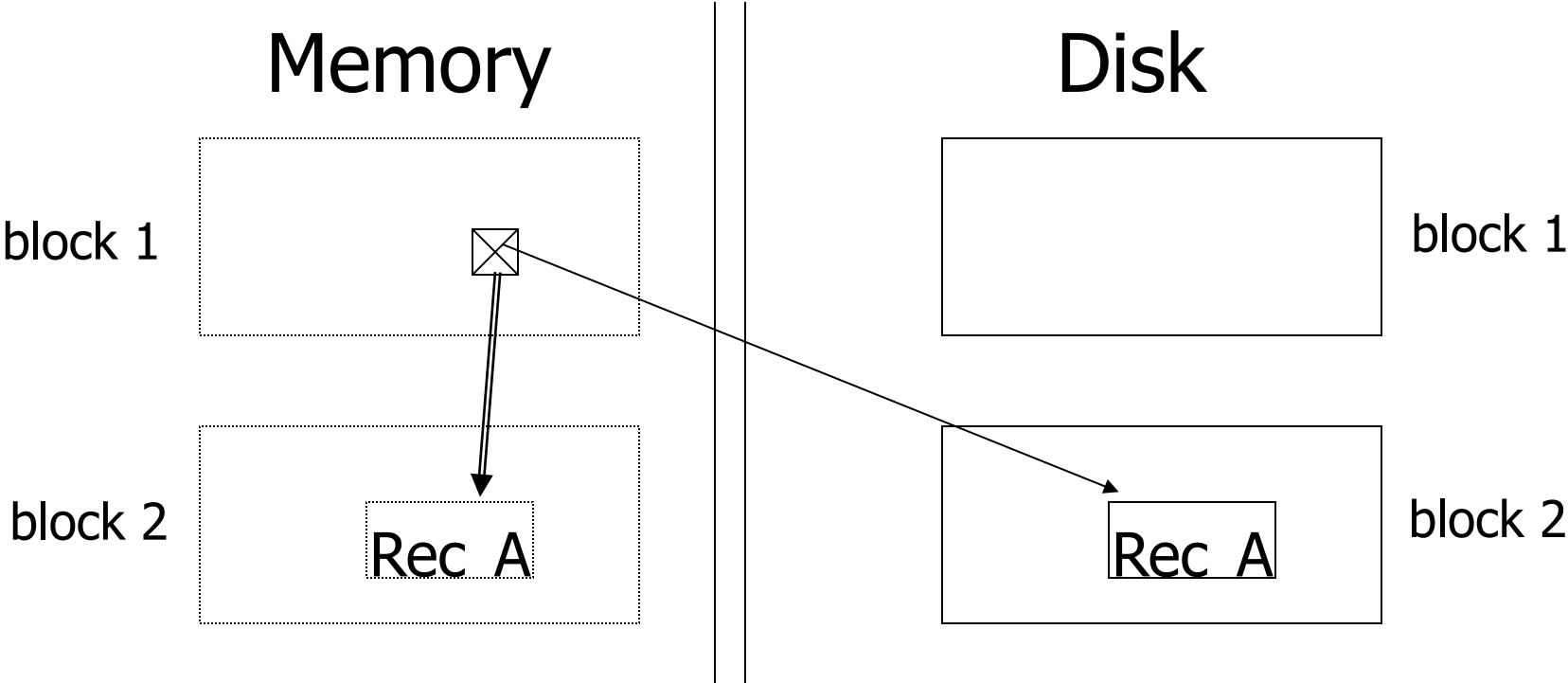
Row vs Column Store

- So far we assumed that fields of a record are stored contiguously (<u>row store</u>)…

- Another option is to store all values of a field together (<u>column store</u>)

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Row Store

- Example: Order consists of
  - id, cust, prod, store, price, date, qty

id1	cust1	prod1	store1	price1	date1	qty1

id2	cust2	prod2	store2	price2	date2	qty2

id3	cust3	prod3	store3	price3	date3	qty3

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Column Store

- Example: Order consists of
  - id, cust, prod, store, price, date, qty

id1	cust1
id2	cust2
id3	cust3
id4	cust4
...	...

id1	prod1
id2	prod2
id3	prod3
id4	prod4
...	...

id1	price1	qty1
id2	price2	qty2
id3	price3	qty3
id4	price4	qty4
...	...	...

ids may or may not be stored explicitly

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Row vs Column Store

- Advantages of Column Store
  - more compact storage (fields need not start at byte boundaries)
  - Efficient compression, e.g., RLE
  - efficient reads on data mining operations
- Advantages of Row Store
  - writes (multiple fields of one record)more efficient
  - efficient reads for record access (OLTP)

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Comparison

- There are 10,000,000 ways to organize my data on disk…

  Which is right for me?

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Issues:

Flexibility ——————— Space Utilization

Complexity ——————— Performance

COMPUTER SCIENCE

Notes 3   87

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
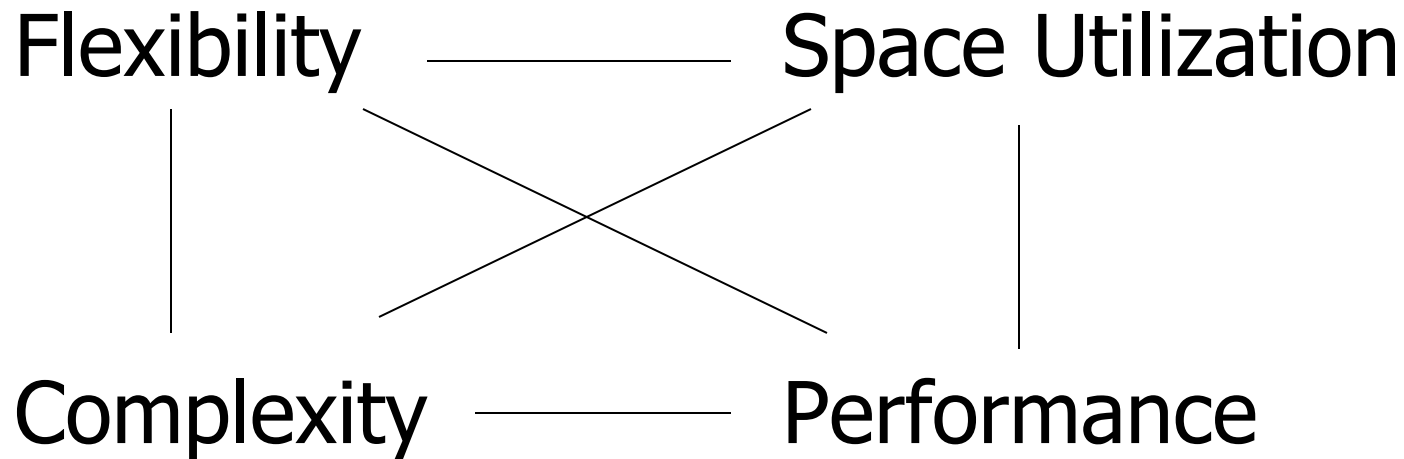
☆ To evaluate a given strategy, compute following parameters:

-> space used for expected data

-> expected time to

- fetch record given key
- fetch record with next key
- insert record
- append record
- delete record
- update record
- read complete file
- reorganize file

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Example

How would you design Megatron 3000 storage system? (for a relational DB, low end)

– Variable length records?
– Spanned?
– What data types?
– Fixed format?
– Record IDs ?
– Sequencing?
– How to handle deletions?

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Summary

- How to lay out data on disk

Data Items

↓

Records

↓

Blocks

↓

Files

↓

Memory

↓

DBMS

Notes 3
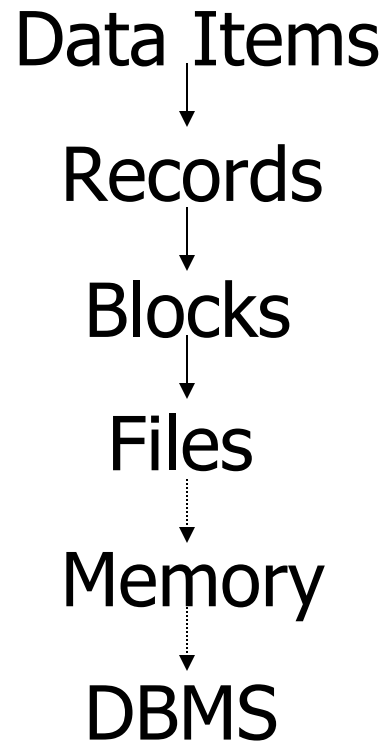
Next

How to find a record quickly,
given a key

CS 525: Advanced Database Organization
04: Indexing

Boris Glavic

Slides: adapted from a course taught by
Hector Garcia-Molina, Stanford InfoLab

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Part 04

Indexing & Hashing

record

value → (?) → | value |

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
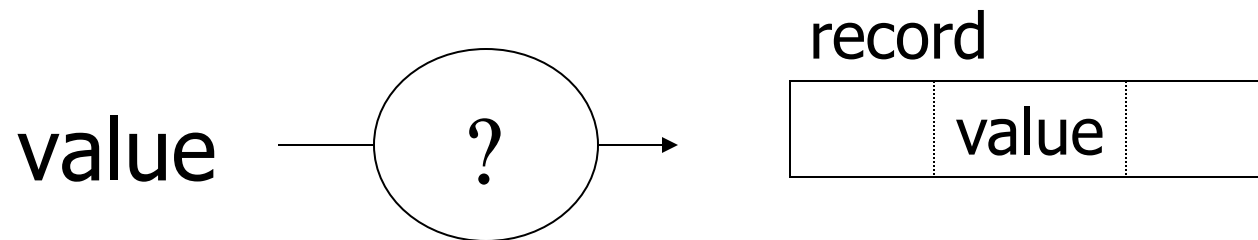
Query Types:

- **Point queries**:
  - *Input:* value **v** of attribute **A**
  - *Output:* all objects (tuples) with that value in attribute **A**

- **Range queries**:
  - *Input:* value interval **[low,high]** of attr **A**
  - Output: all tuples with a value

    low <= v < high in attribute **A**

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Index Considerations:

- Supported Query Types
- Secondary-storage capable
- Storage size
  - Index Size / Data Size
- Complexity of Operations
  - E.g., insert is $O(\log(n))$ worst-case
- Efficient Concurrent Operations?

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Topics

- Conventional indexes
- B-trees
- Hashing schemes
- Advanced Index Techniques

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Sequential File

10	
20	

30	
40	

50	
60	

70	
80	

90	
100	

Dense Index — Sequential File

Sparse Index   Sequential File

10	
30	
50	
70	

90	
110	
130	
150	

170	
190	
210	
230	

| 10 | |
| 20 | |

| 30 | |
| 40 | |

| 50 | |
| 60 | |

| 70 | |
| 80 | |

| 90 | |
| 100 | |

CS 525

Notes 4 - Indexing

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
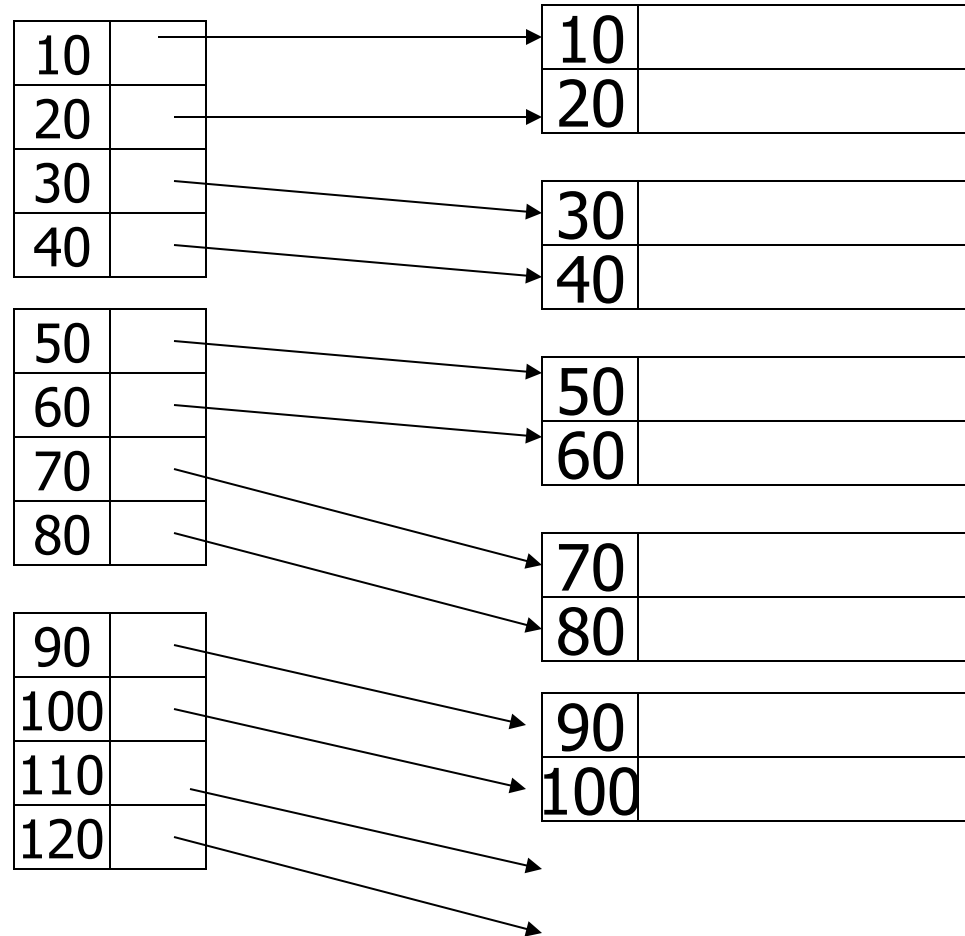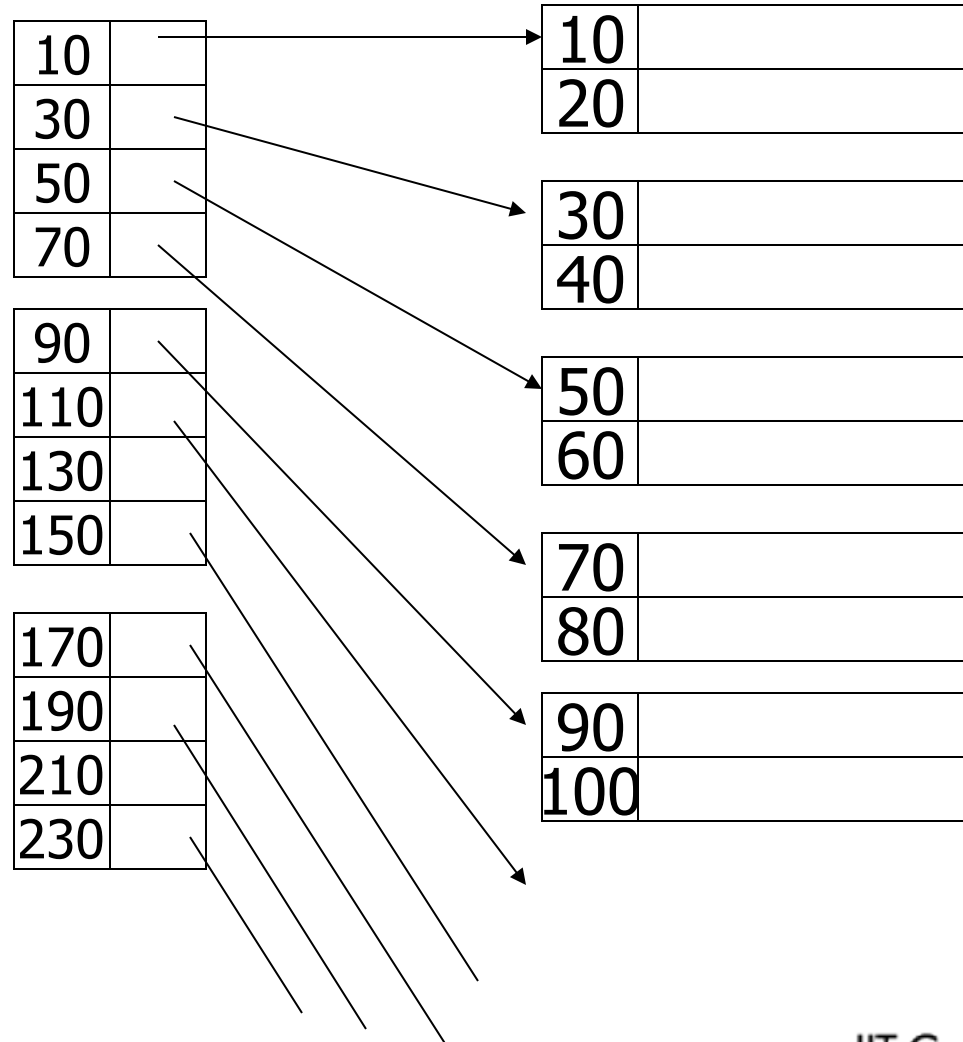
Sparse 2nd level

Sequential File

10	
90	
170	
250	

330	
410	
490	
570	

10	
30	
50	
70	

90	
110	
130	
150	

170	
190	
210	
230	

10	
20	

30	
40	

50	
60	

70	
80	

90	
100	

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
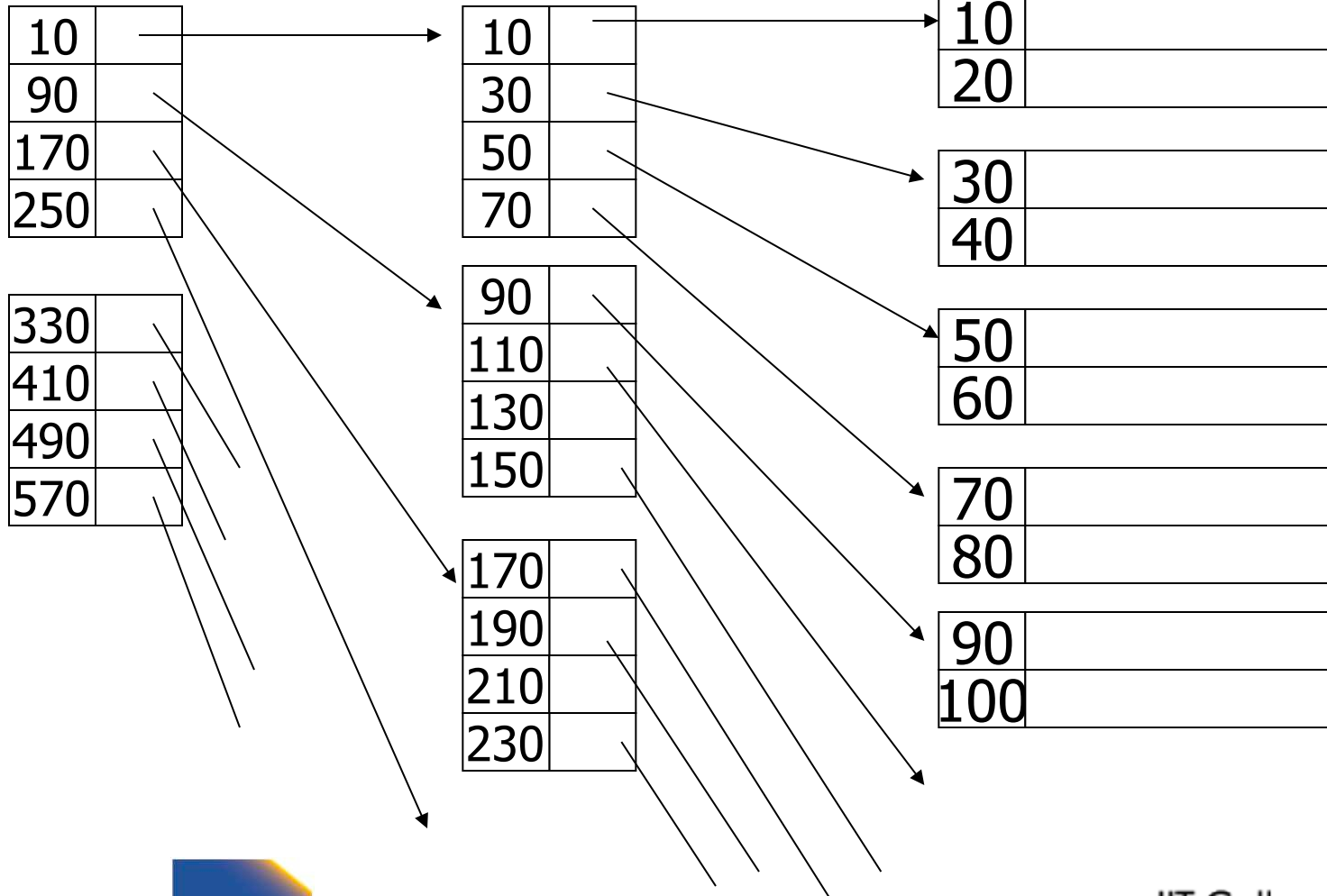
- Comment:

  {FILE,INDEX} may be contiguous

  or not (blocks chained)

Question:

- Can we build a dense, 2nd level index for a dense index?

Notes 4 - Indexing

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Notes on pointers:

(1) Block pointer (sparse index) can be smaller than record pointer

BP

RP

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
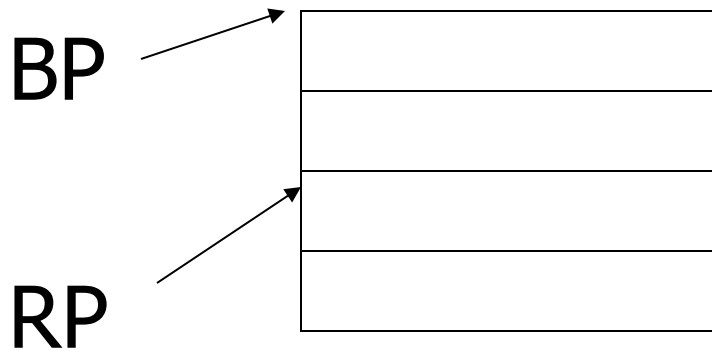
Notes on pointers:

(2) If file is contiguous, then we can omit pointers (i.e., compute them)

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

K1

K2

K3

K4

R1

R2

R3

R4
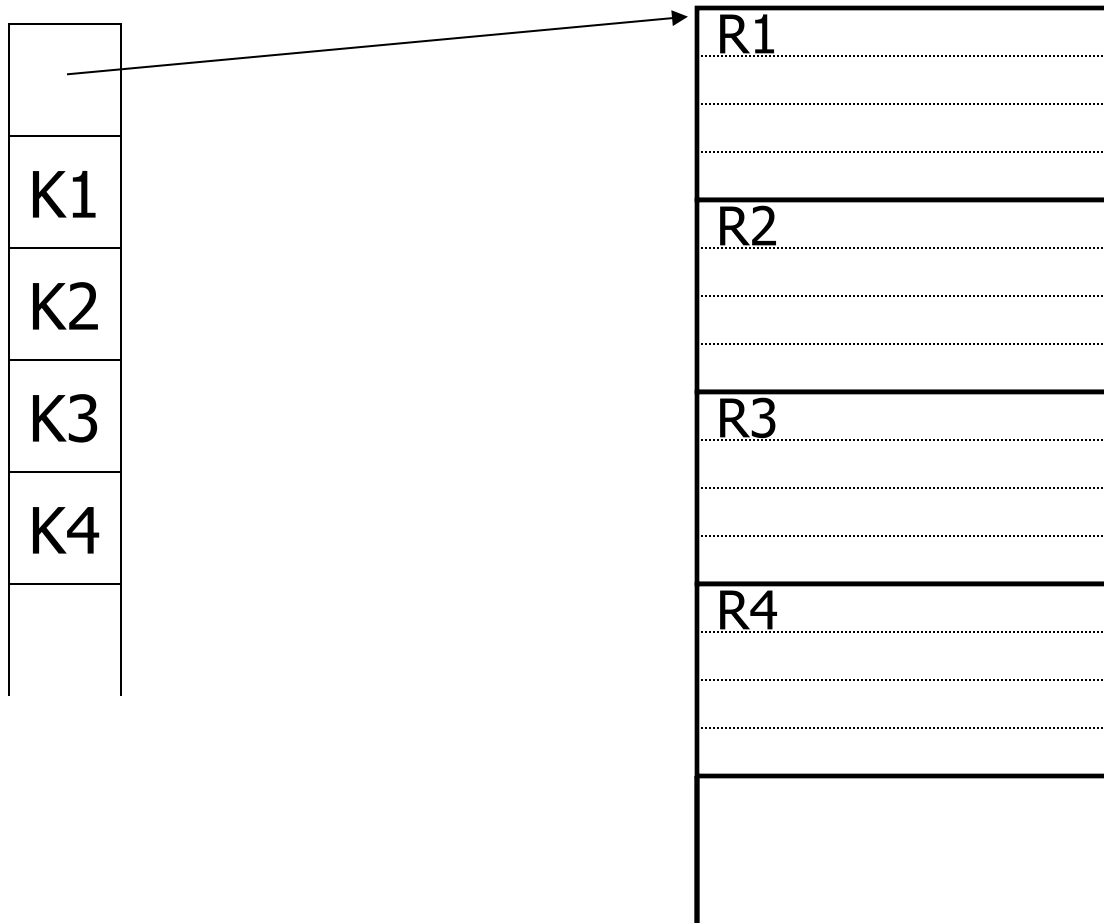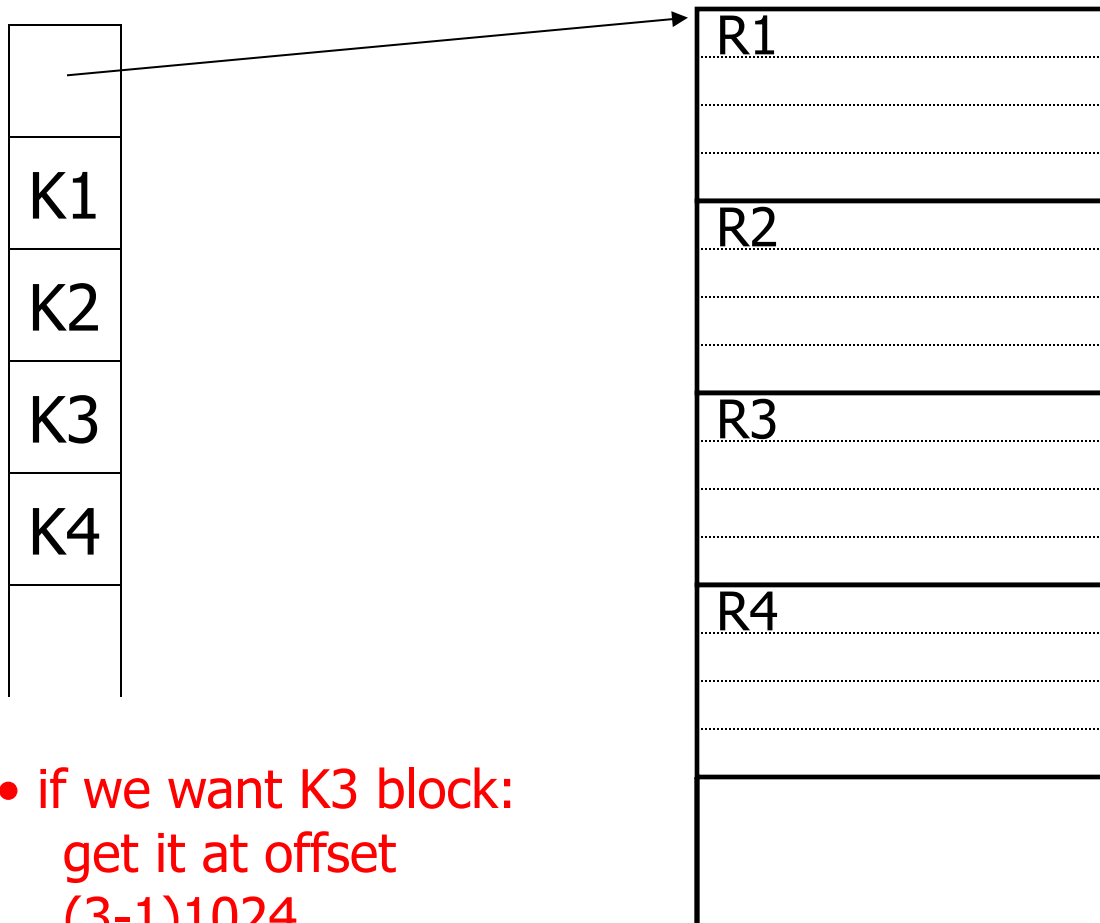
say:
1024 B
per block

- if we want K3 block:
  get it at offset
  (3-1)1024
  = 2048 bytes

COMPUTER
SCIENCE

Notes 4 - Indexing

15

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Sparse vs. Dense Tradeoff

- <u>Sparse:</u> Less index space per record
  can keep more of index in memory

- <u>Dense:</u> Can tell if any record exists
  without accessing file

(Later:
- sparse better for insertions
- dense needed for secondary indexes)

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Terms

- Index sequential file
- Search key ($\neq$ primary key)
- Primary index (on Sequencing field)
- Secondary index
- Dense index (all Search Key values in)
- Sparse index
- Multi-level index

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Next:

- Duplicate keys

- Deletion/Insertion

- Secondary indexes

Duplicate keys

10	
10	

10	
20	

20	
30	

30	
30	

40	
45	

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate keys

Dense index, one way to implement?
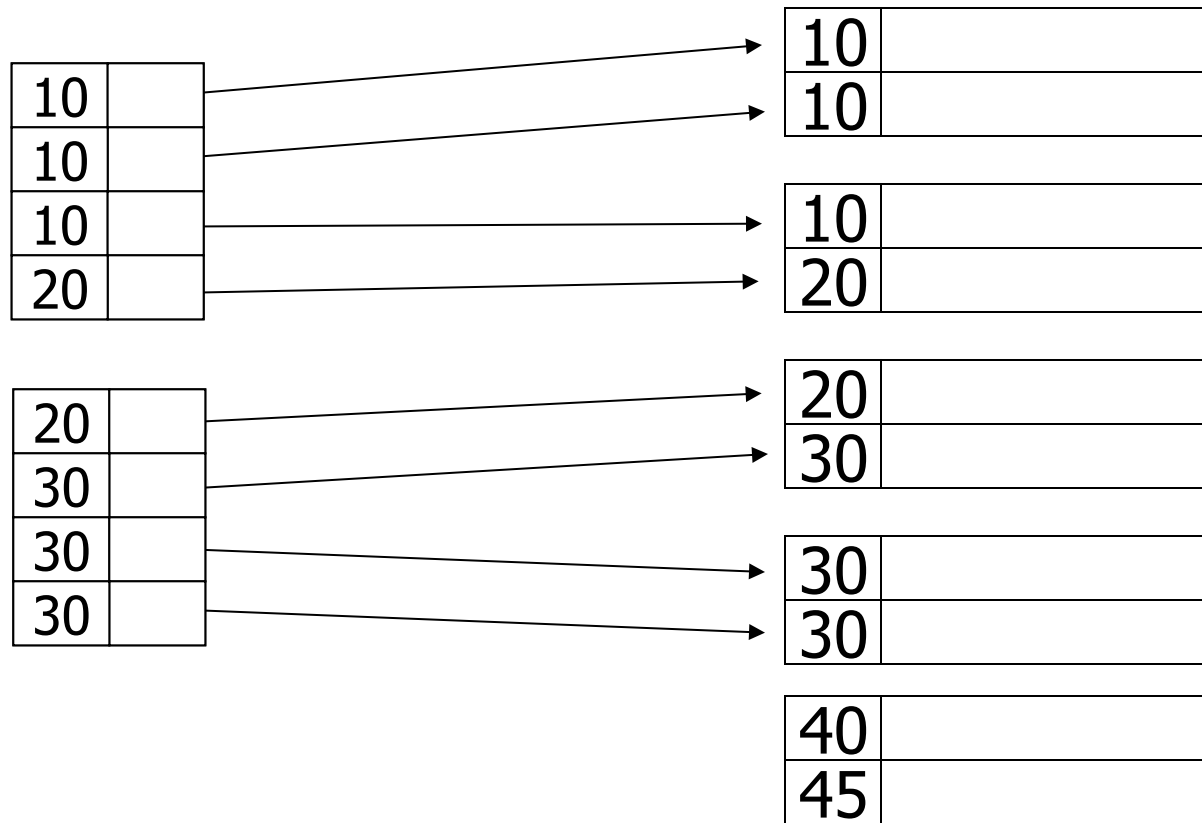
Duplicate keys

Dense index, better way?

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate keys

Sparse index, one way?

Notes 4 - Indexing

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
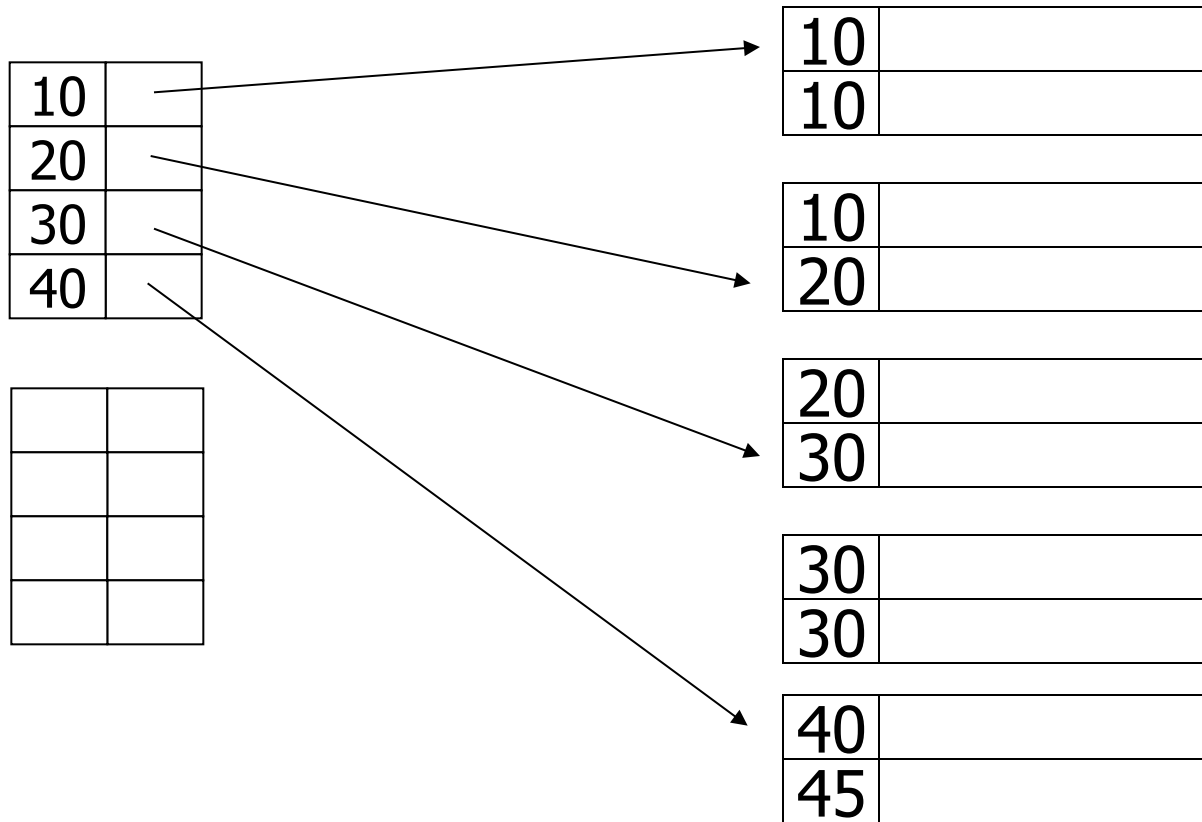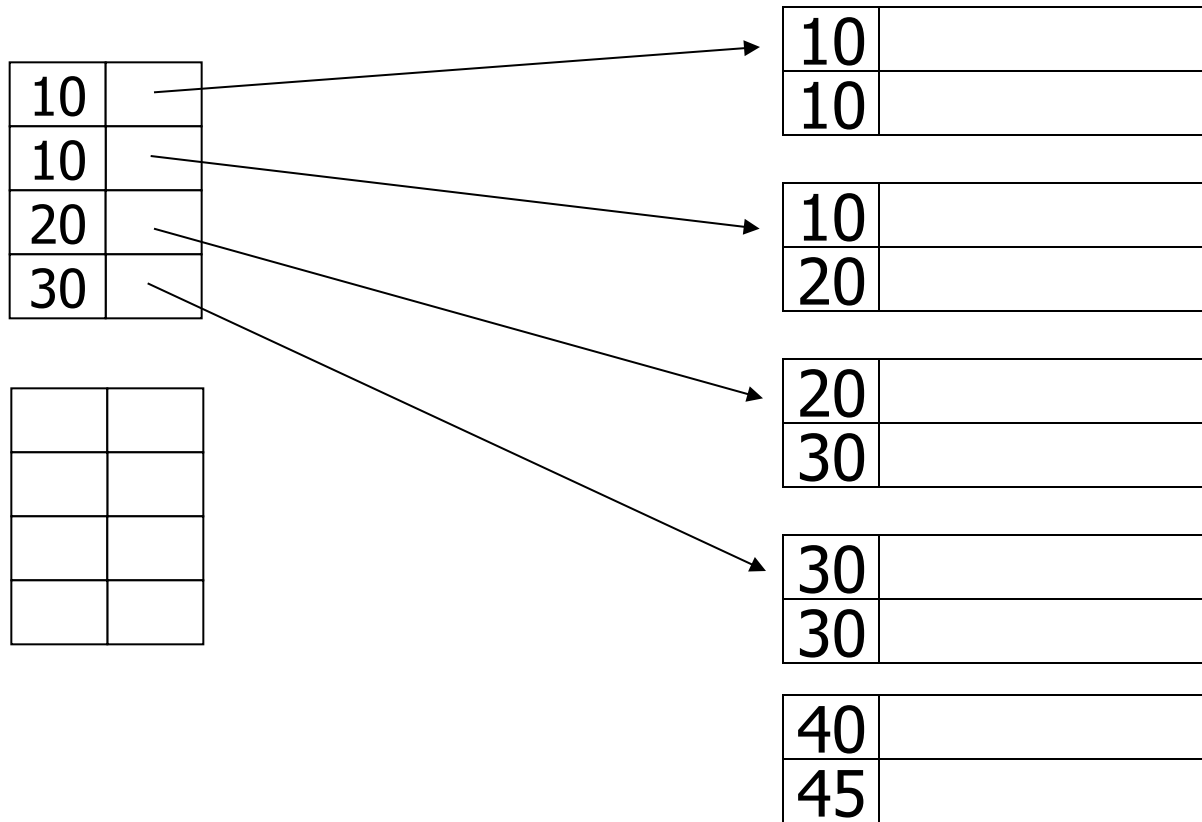
Duplicate keys

Sparse index, one way?

careful if looking for 20 or 30!
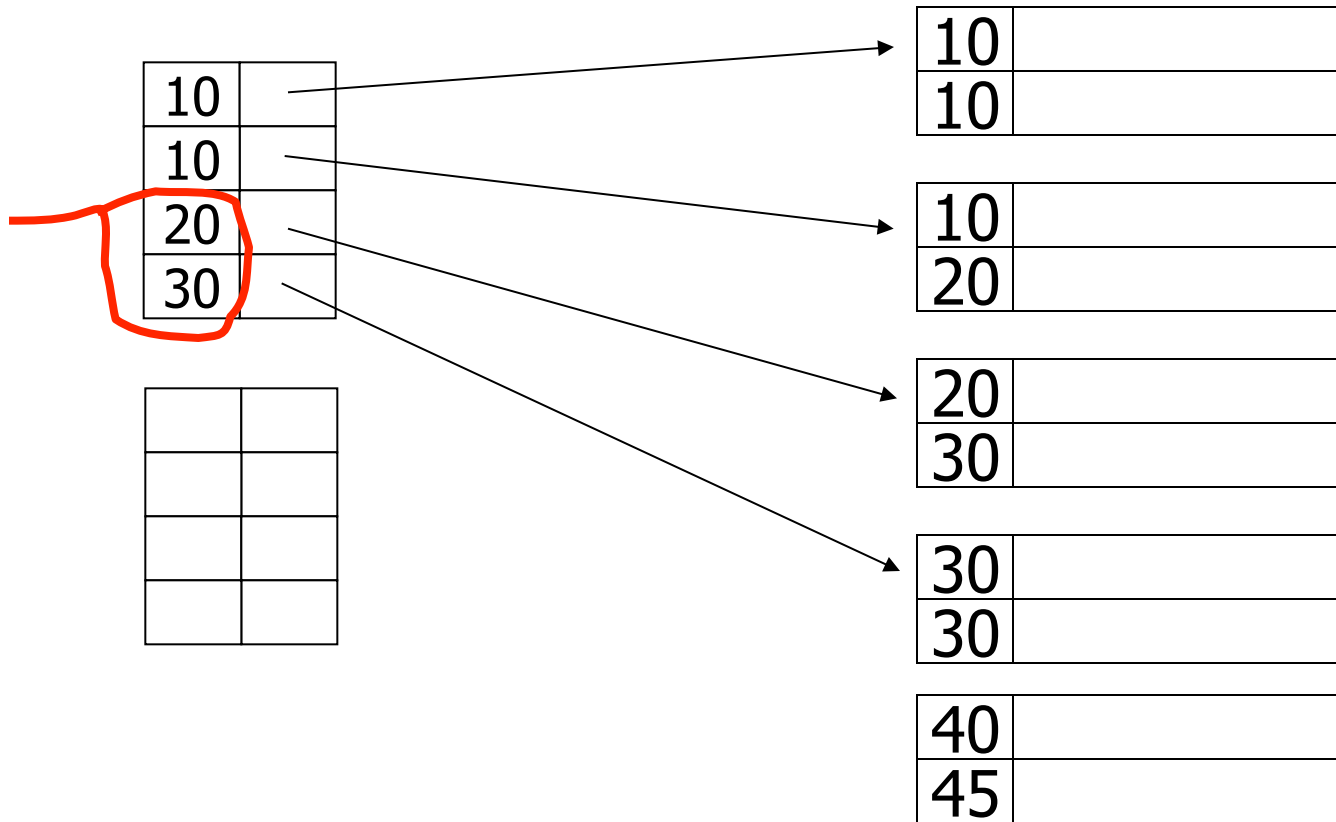
10	
10	
20	
30	

| 10 | |
| 10 | |

| 10 | |
| 20 | |

| 20 | |
| 30 | |

| 30 | |
| 30 | |

| 40 | |
| 45 | |

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate keys

Sparse index, another way?

– place first new key from block

10	
20	
30	
30	

10	
10	

10	
20	

20	
30	

30	
30	

40	
45	

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
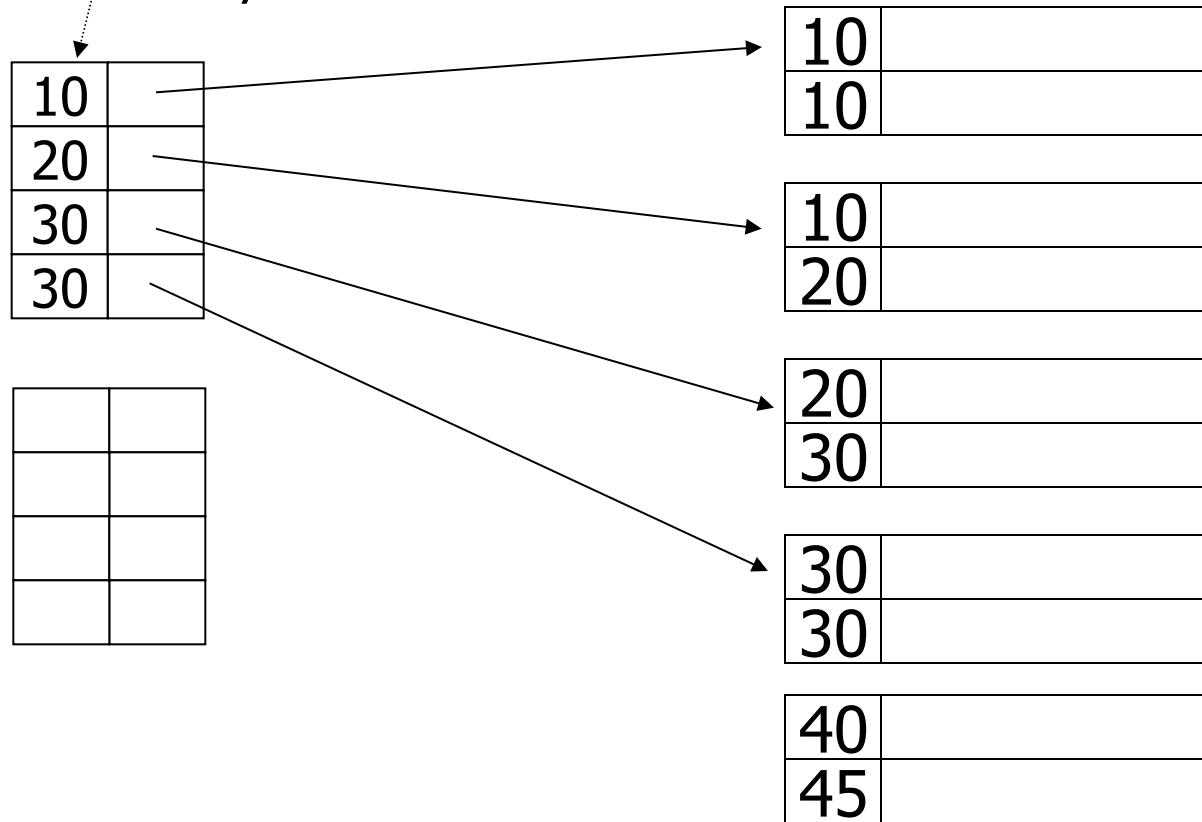
Duplicate keys

Sparse index, another way?

– place first new key from block

should this be 40?

| 10 | |
| 10 | |

| 10 | |
| 20 | |

| 20 | |
| 30 | |

| 30 | |
| 30 | |

| 40 | |
| 45 | |

COMPUTER SCIENCE

Notes 4 - Indexing

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate values, primary index

- Index may point to <u>first</u> instance of each value only

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from sparse index

COMPUTER SCIENCE

Notes 4 - Indexing

27

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from sparse index

– delete record 40

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
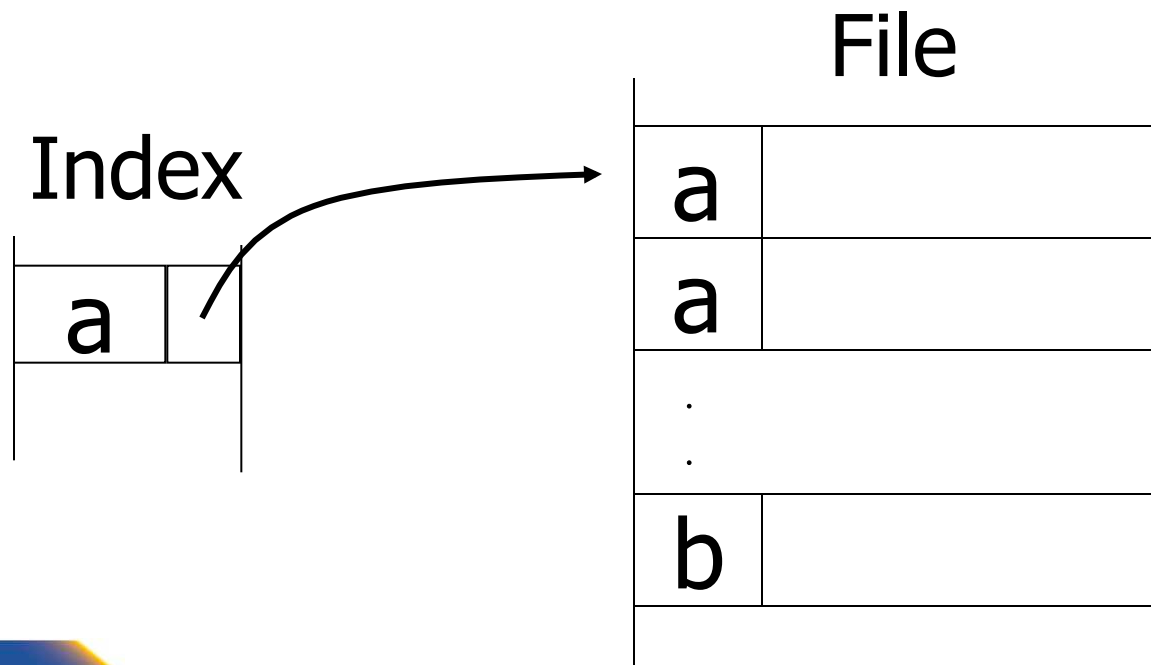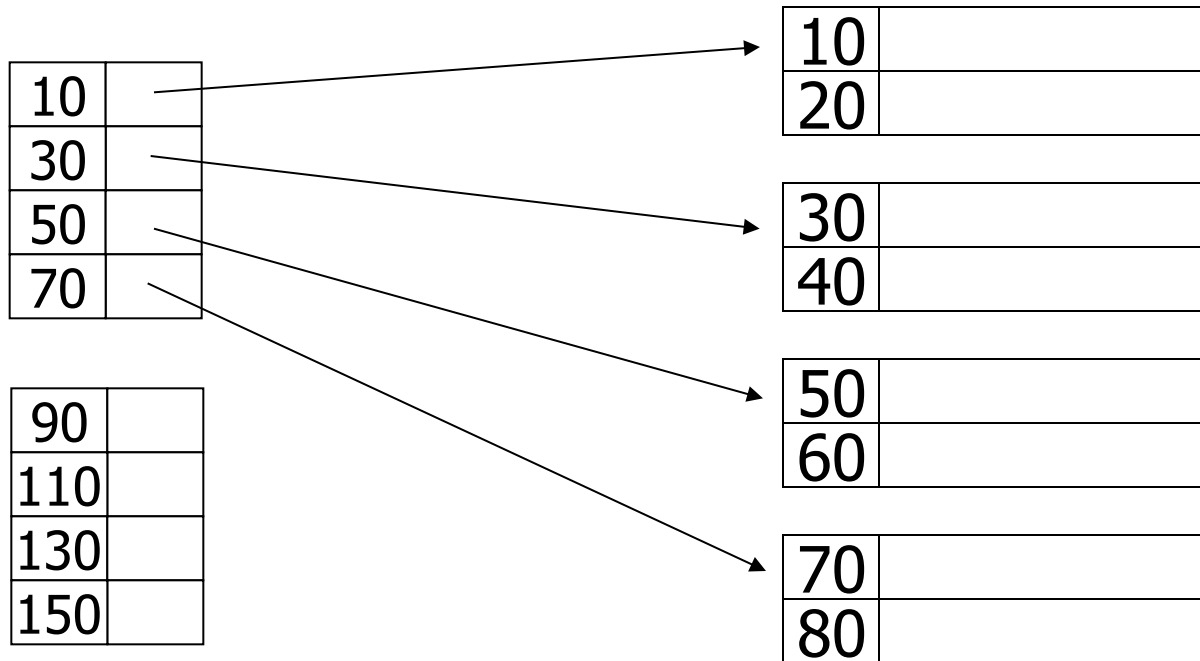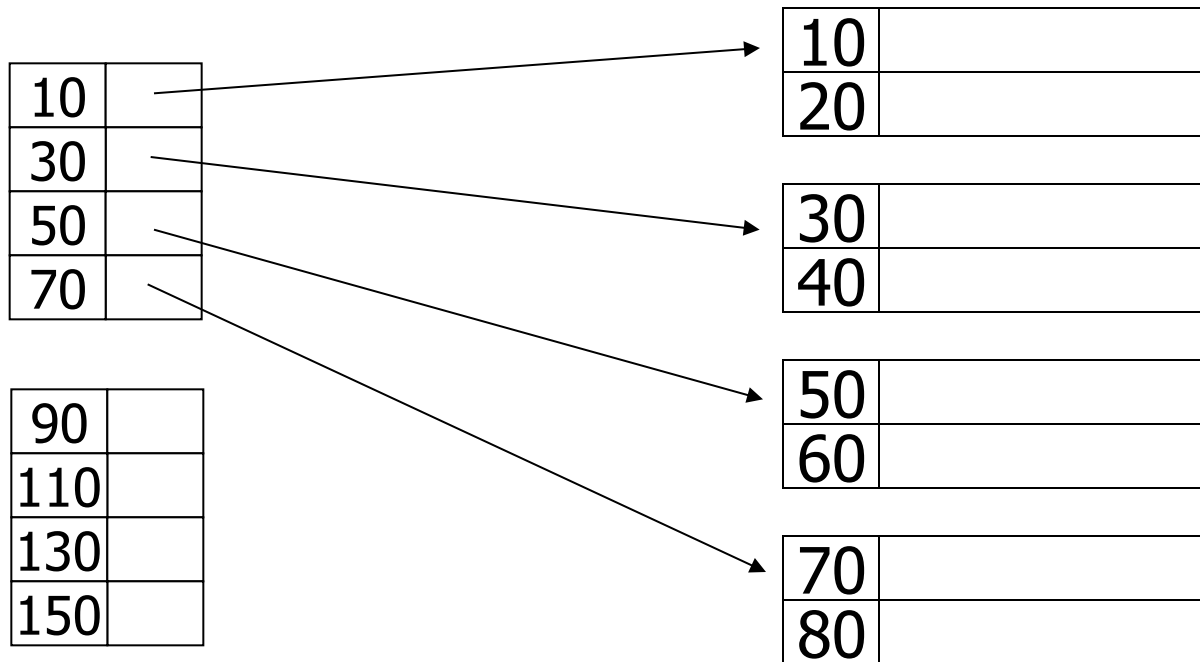
Deletion from sparse index

– delete record 40

Deletion from sparse index
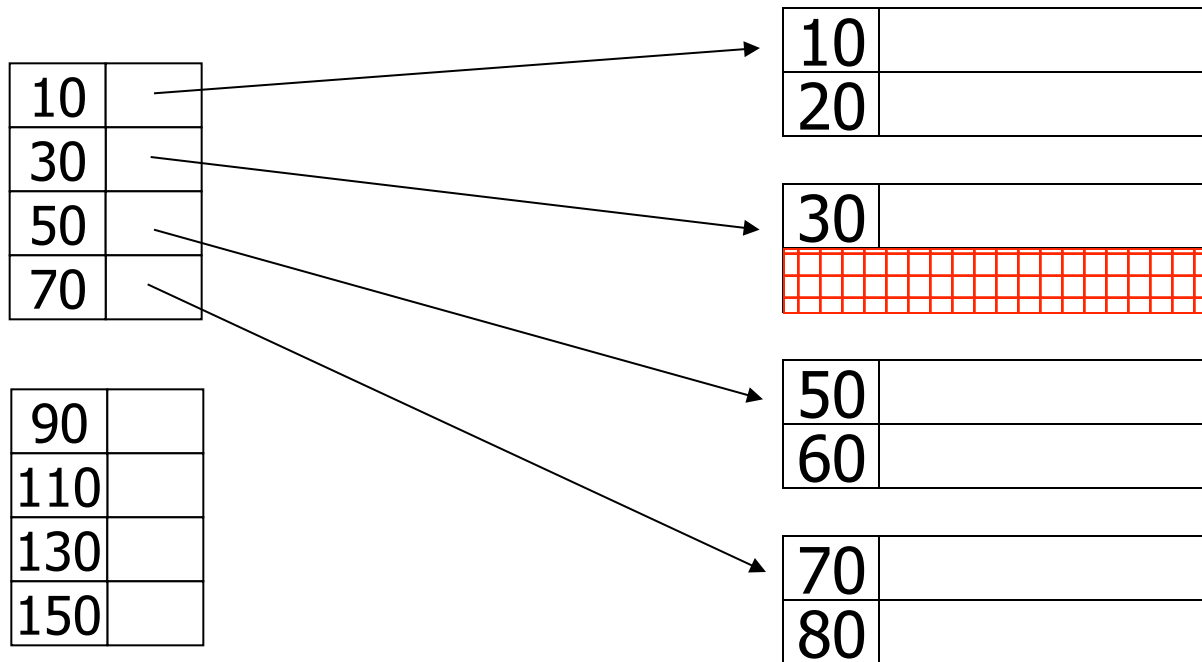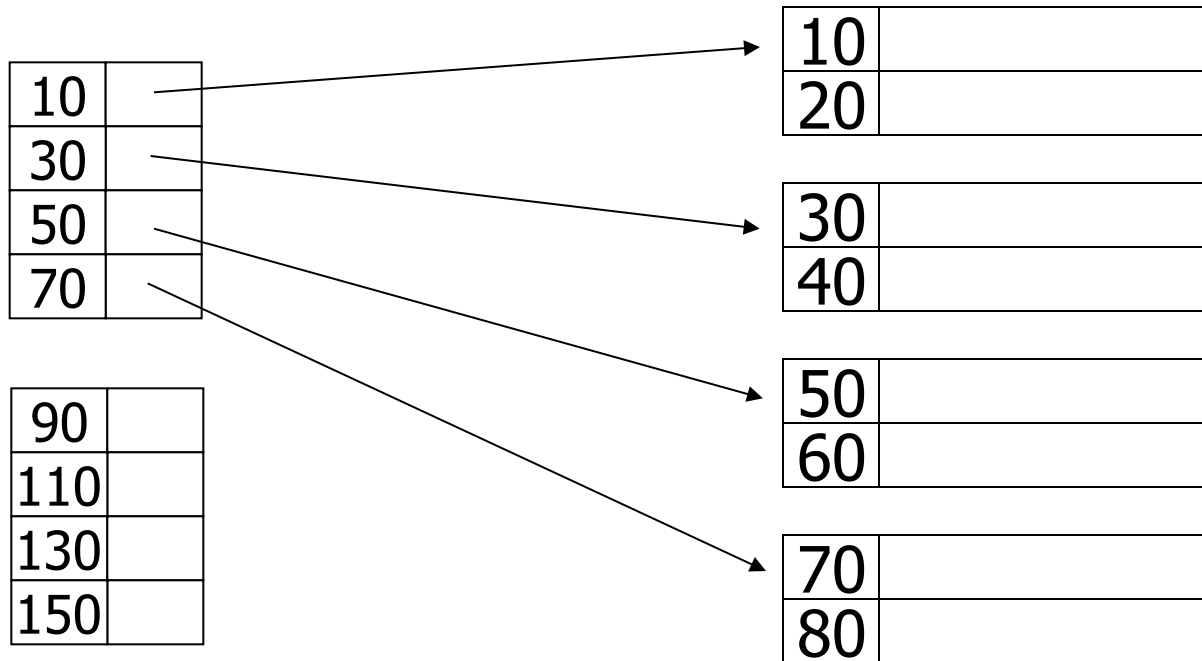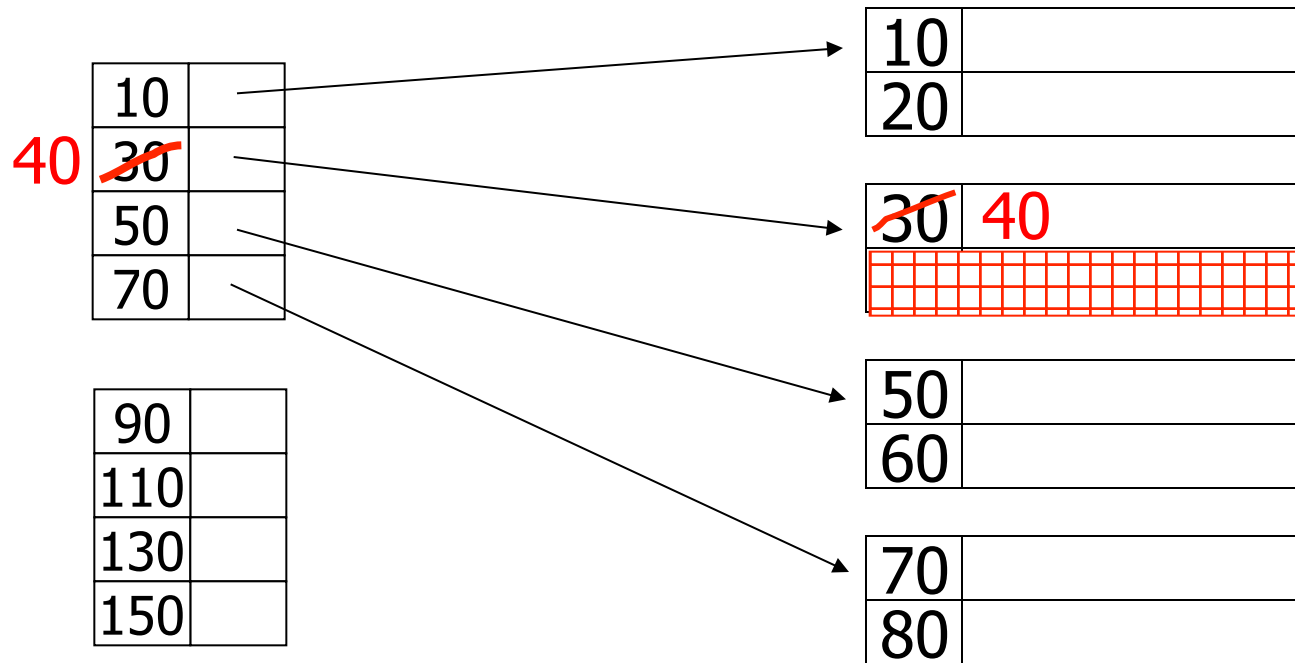
– delete record 30

Notes 4 - Indexing

Deletion from sparse index

– delete record 30

COMPUTER SCIENCE

Notes 4 - Indexing

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from sparse index

– delete records 30 & 40

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Deletion from sparse index

– delete records 30 & 40

10	
30	
50	
70	

90	
110	
130	
150	

10	
20	

50	
60	

70	
80	

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Deletion from sparse index

– delete records 30 & 40

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from dense index

Notes 4 - Indexing

Deletion from dense index

    – delete record 30

Deletion from dense index

– delete record 30

Deletion from dense index

– delete record 30

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
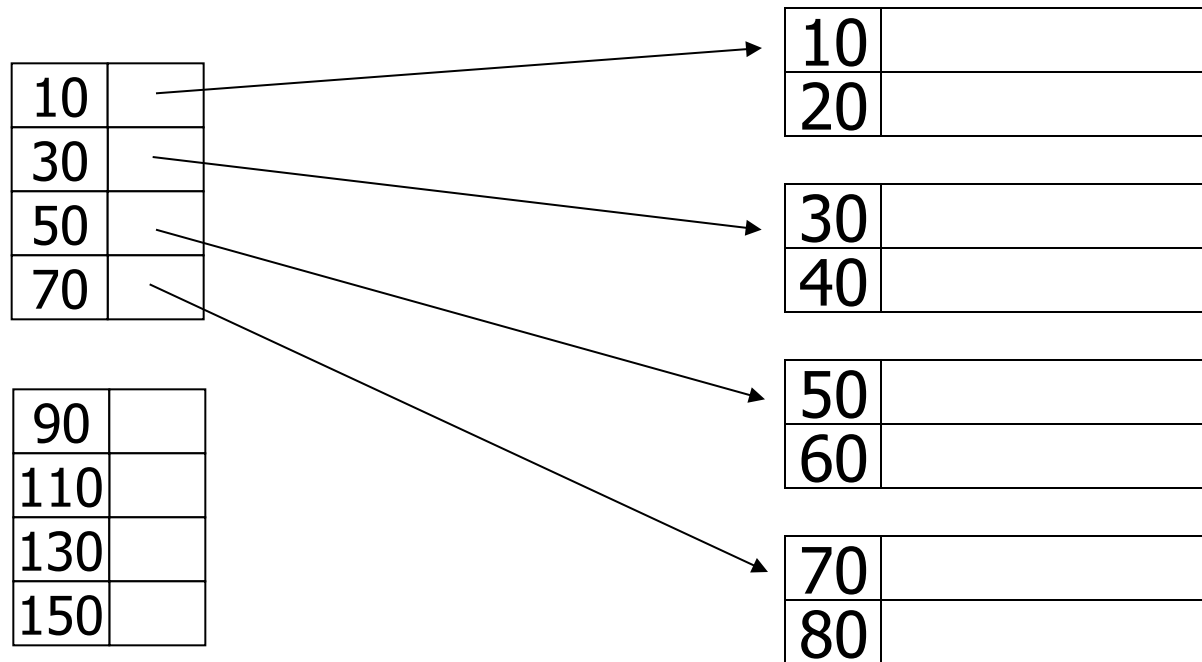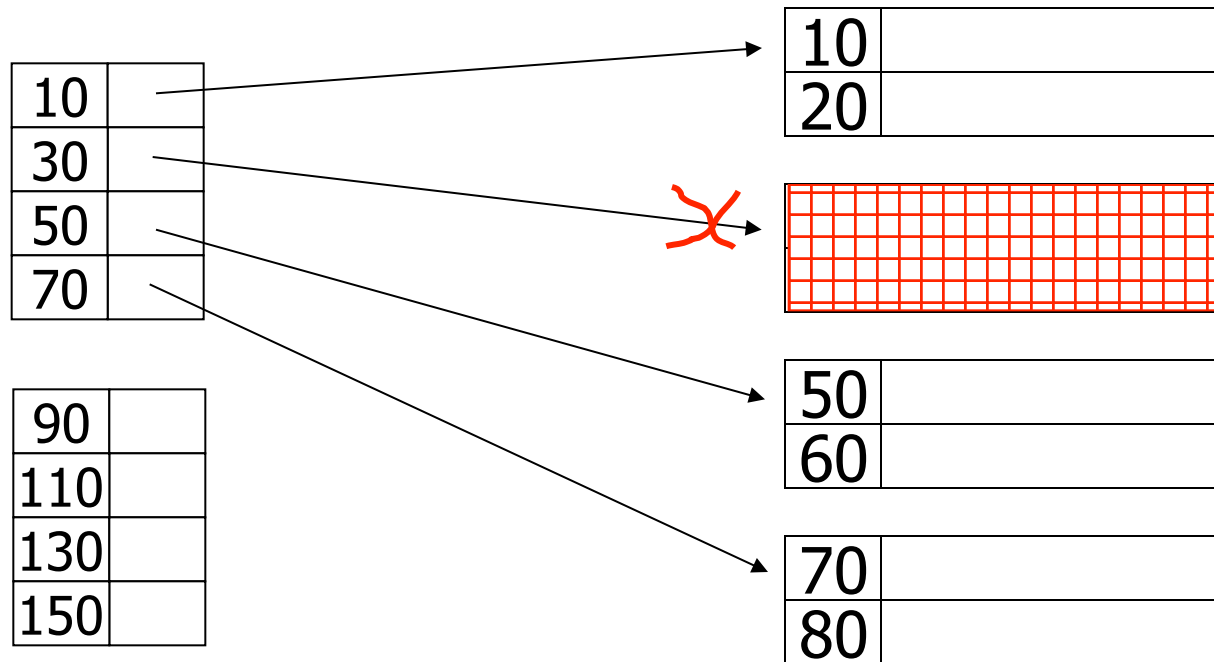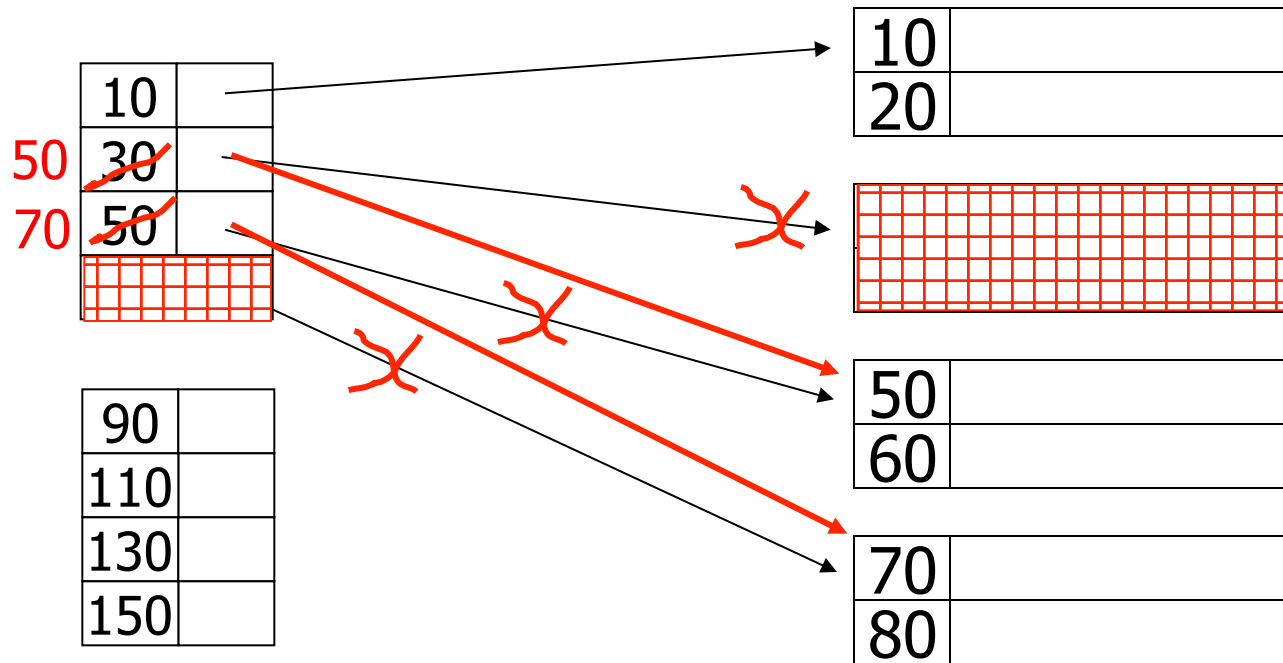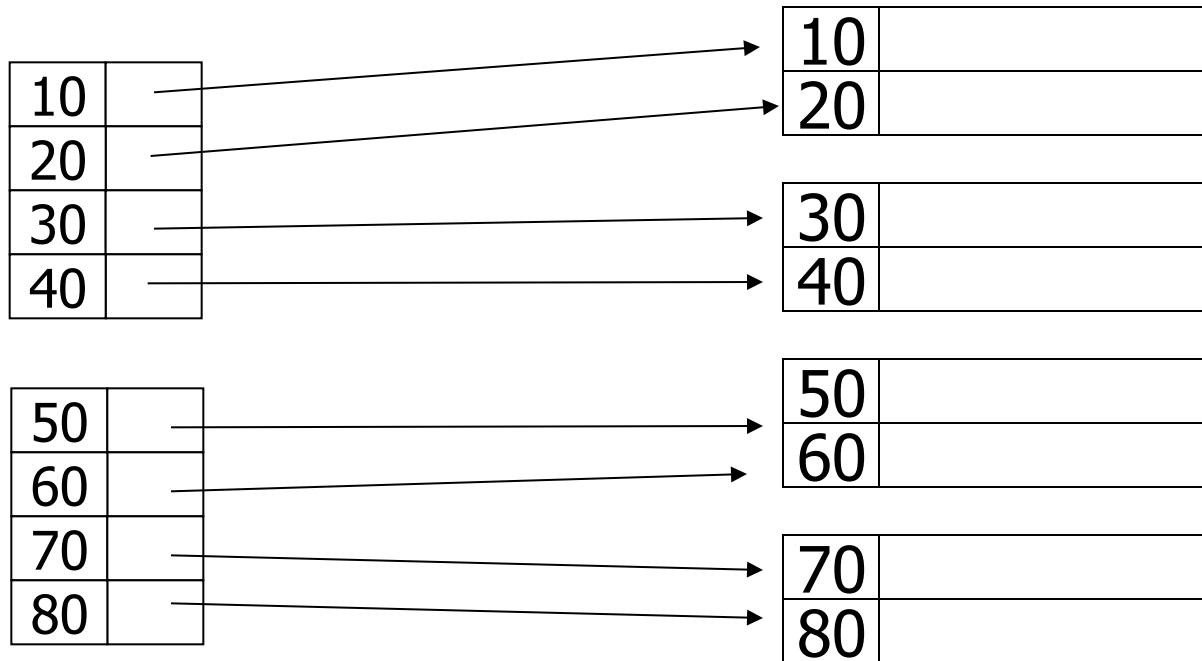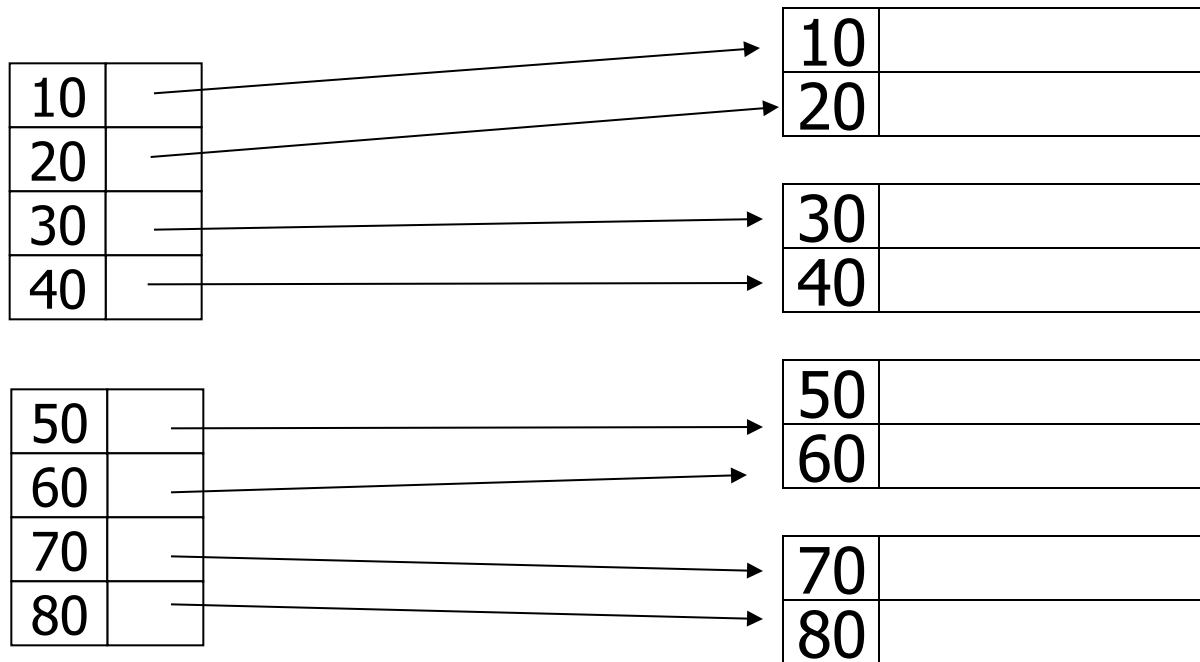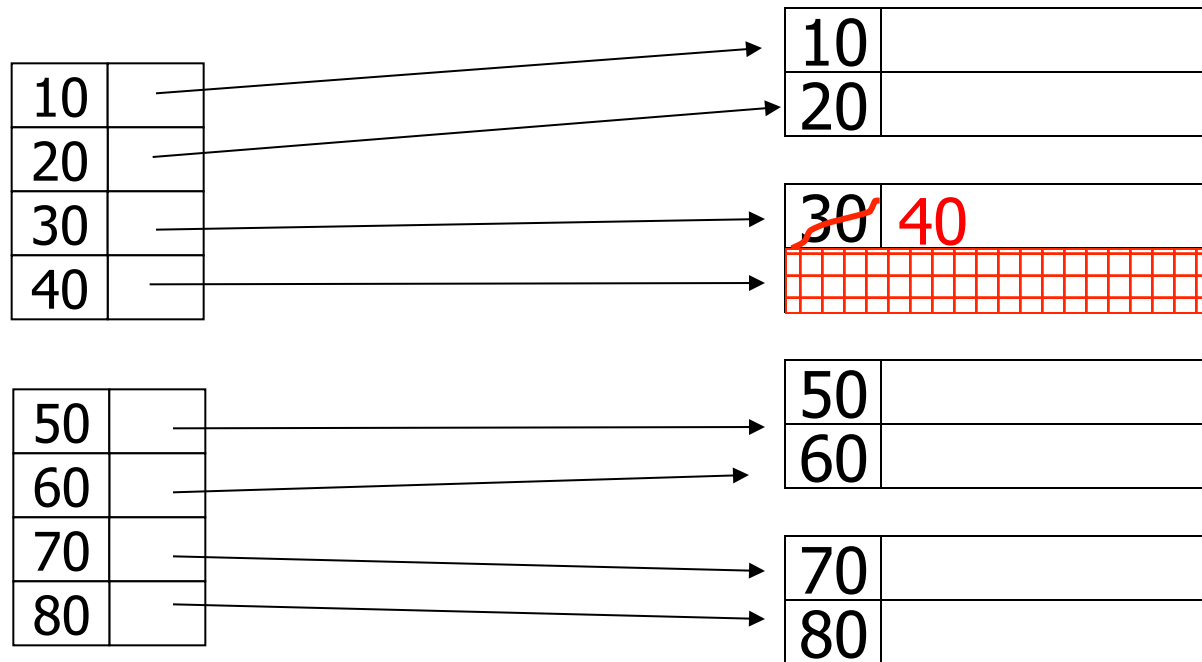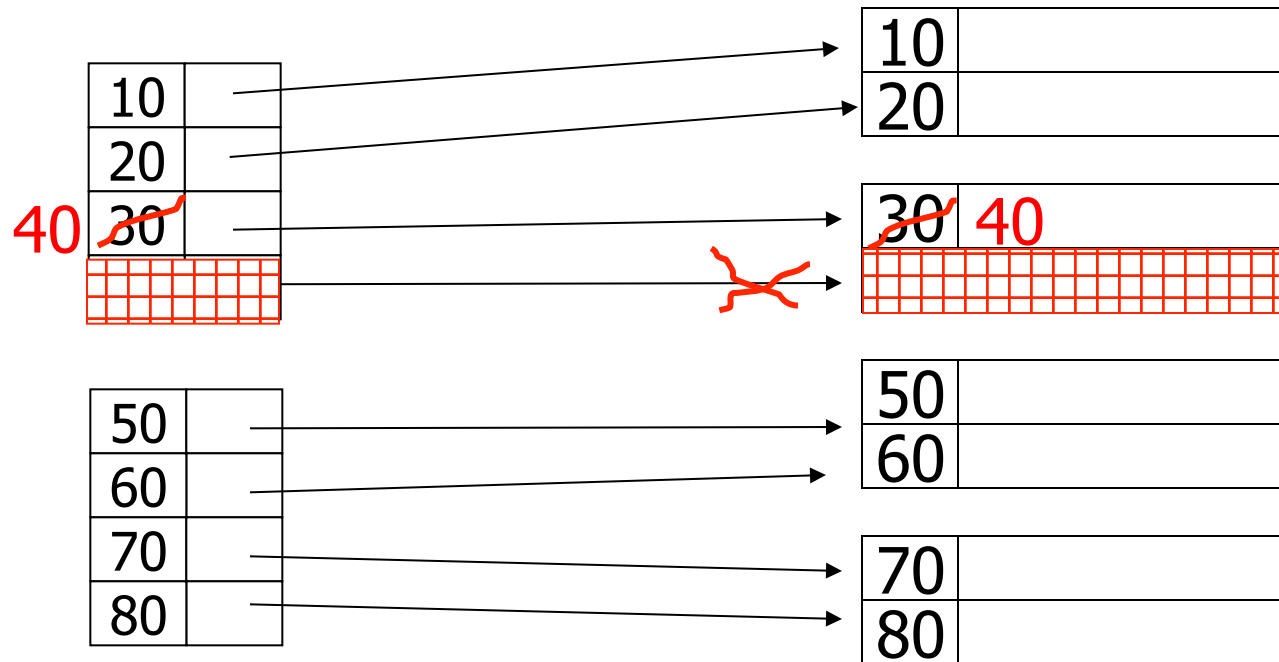
Insertion, sparse index case
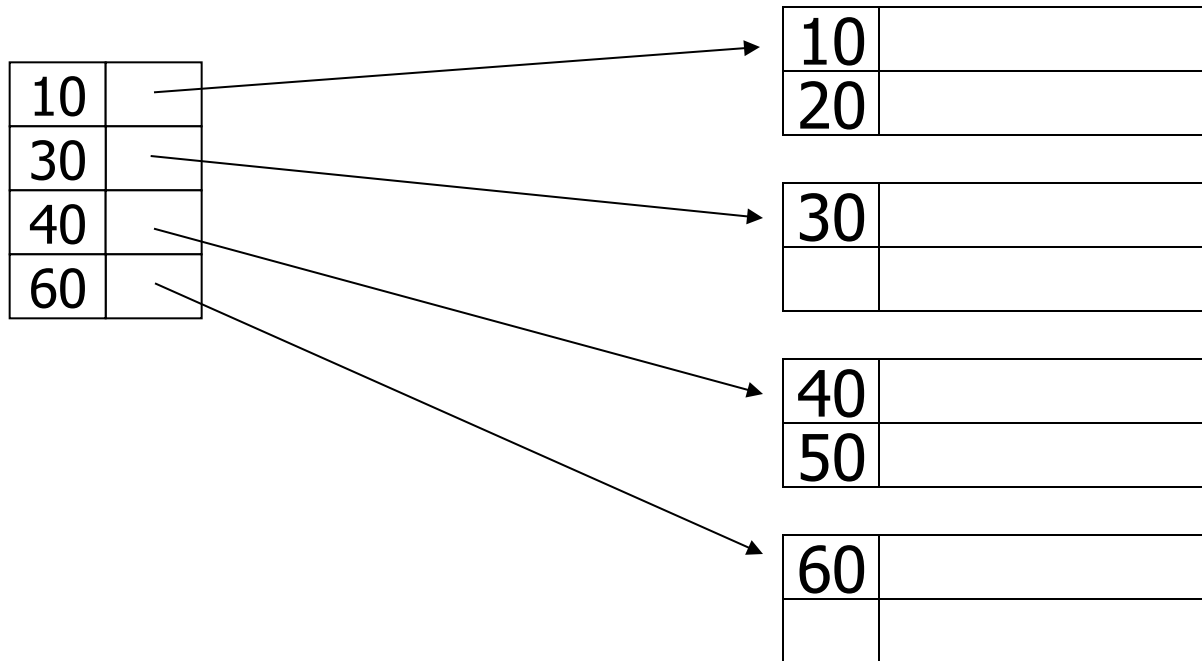
Insertion, sparse index case
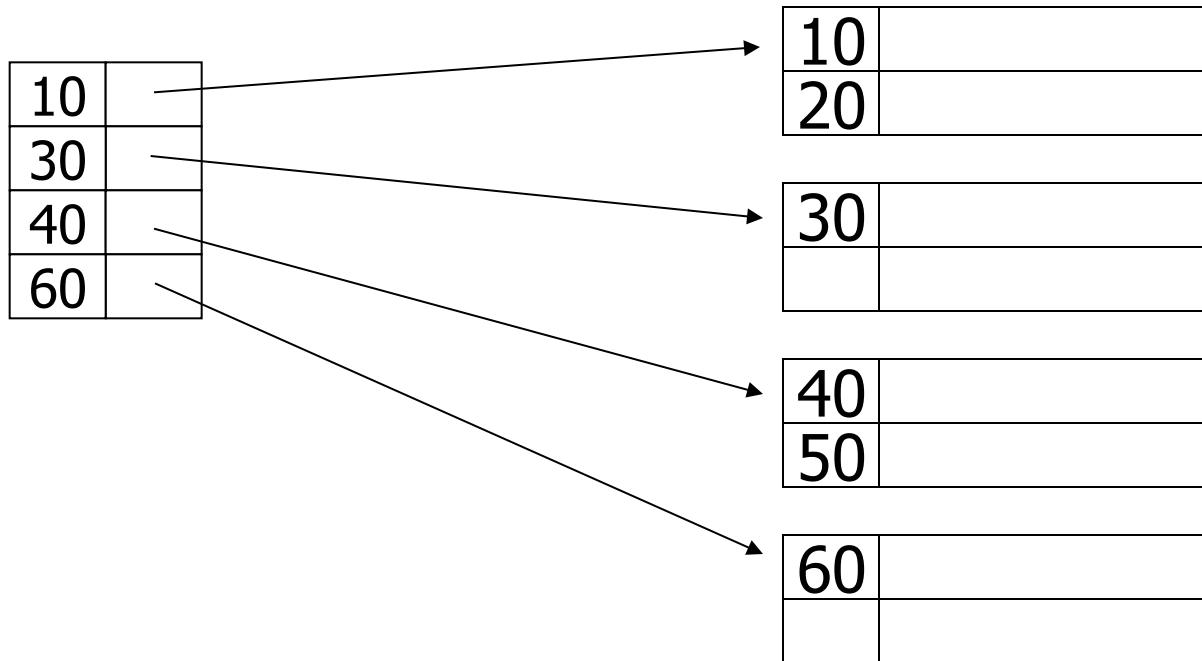
– insert record 34

Insertion, sparse index case
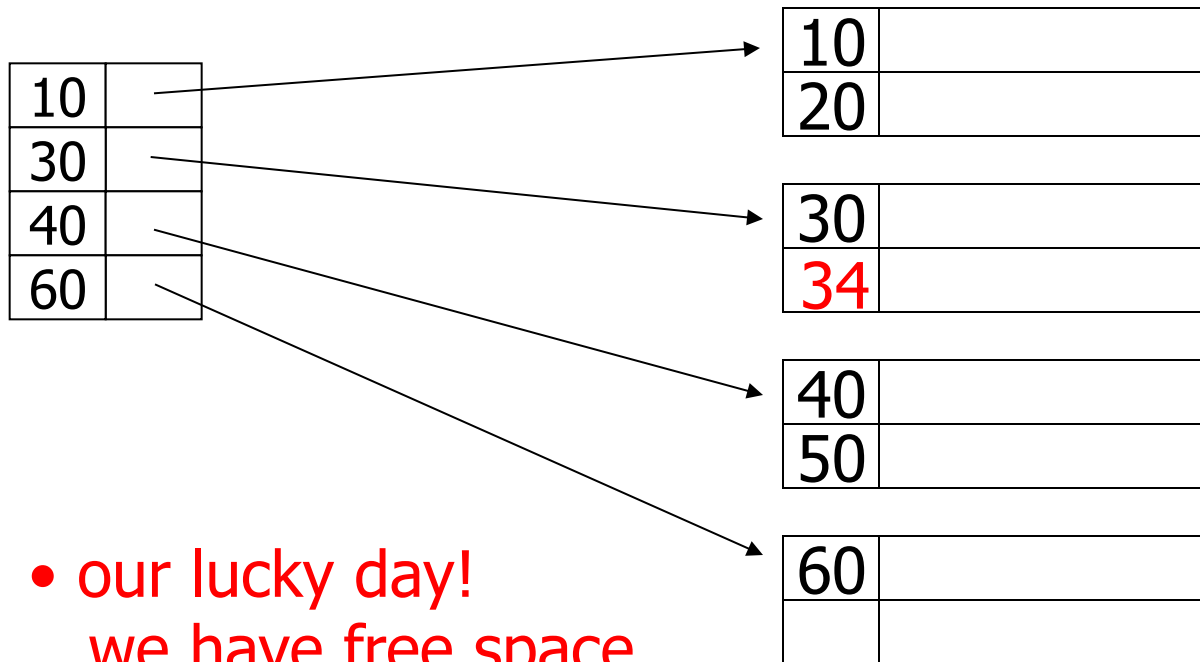
– insert record 34

• our lucky day!
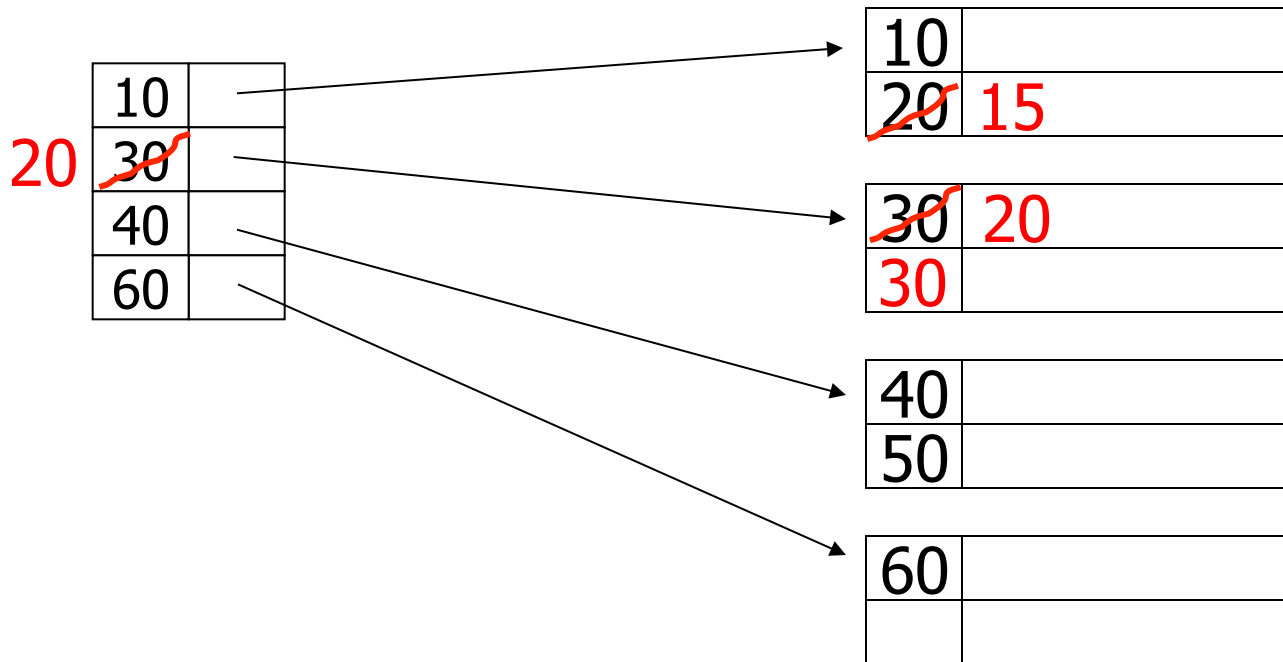 we have free space
  where we need it!

IIT College of
Science and Letters
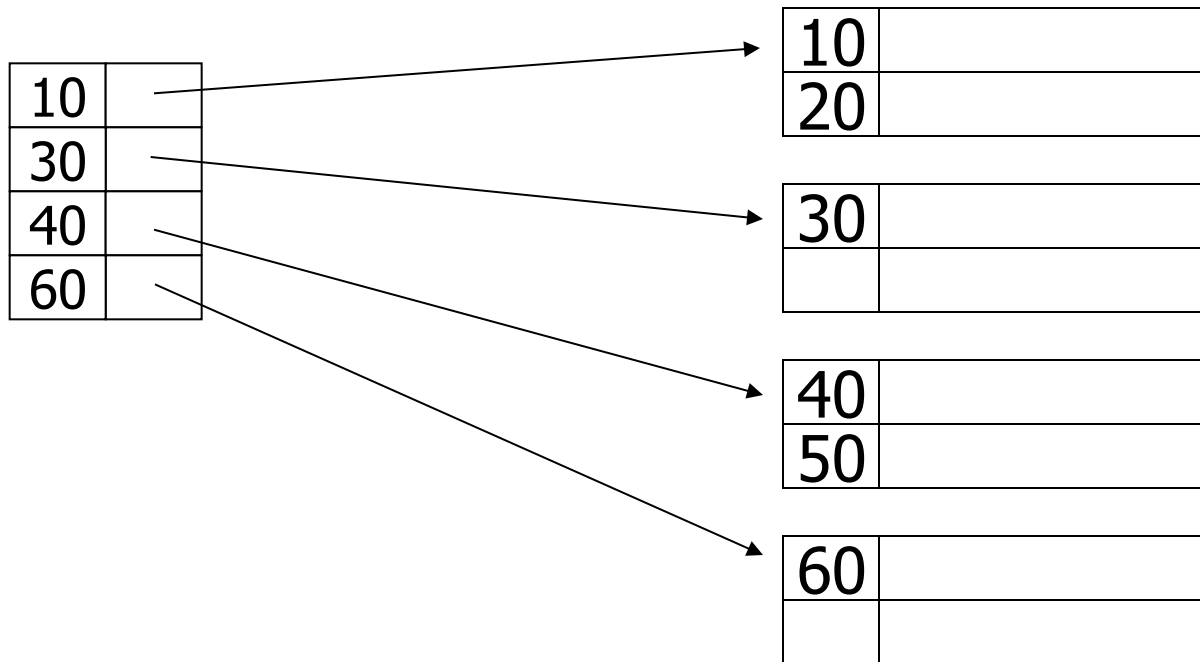
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Insertion, sparse index case

– insert record 15

10	
30	
40	
60	

10	
20	

30	

40	
50	

60	

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Insertion, sparse index case

– insert record 15

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
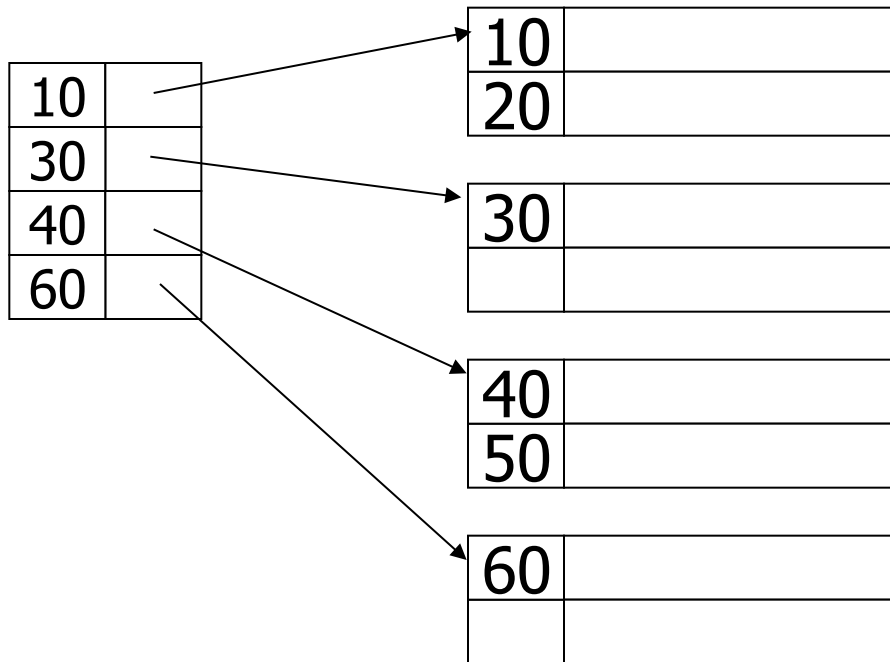
Insertion, sparse index case
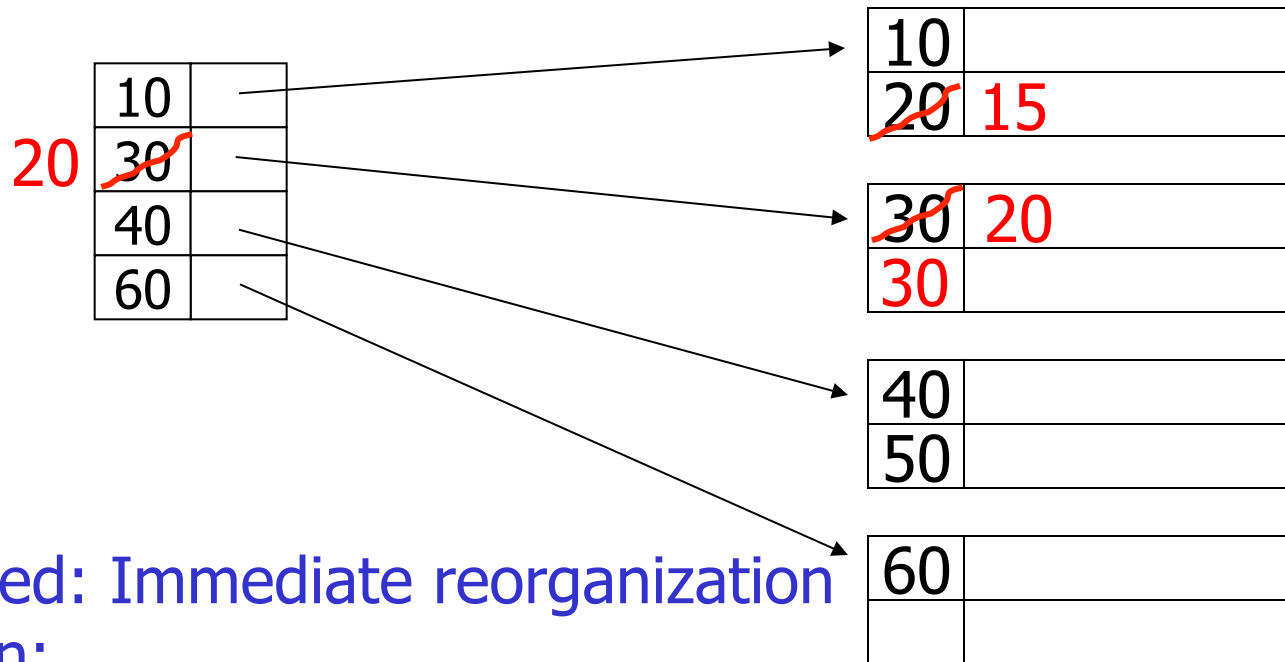
– insert record 15

- Illustrated: Immediate reorganization
- Variation:
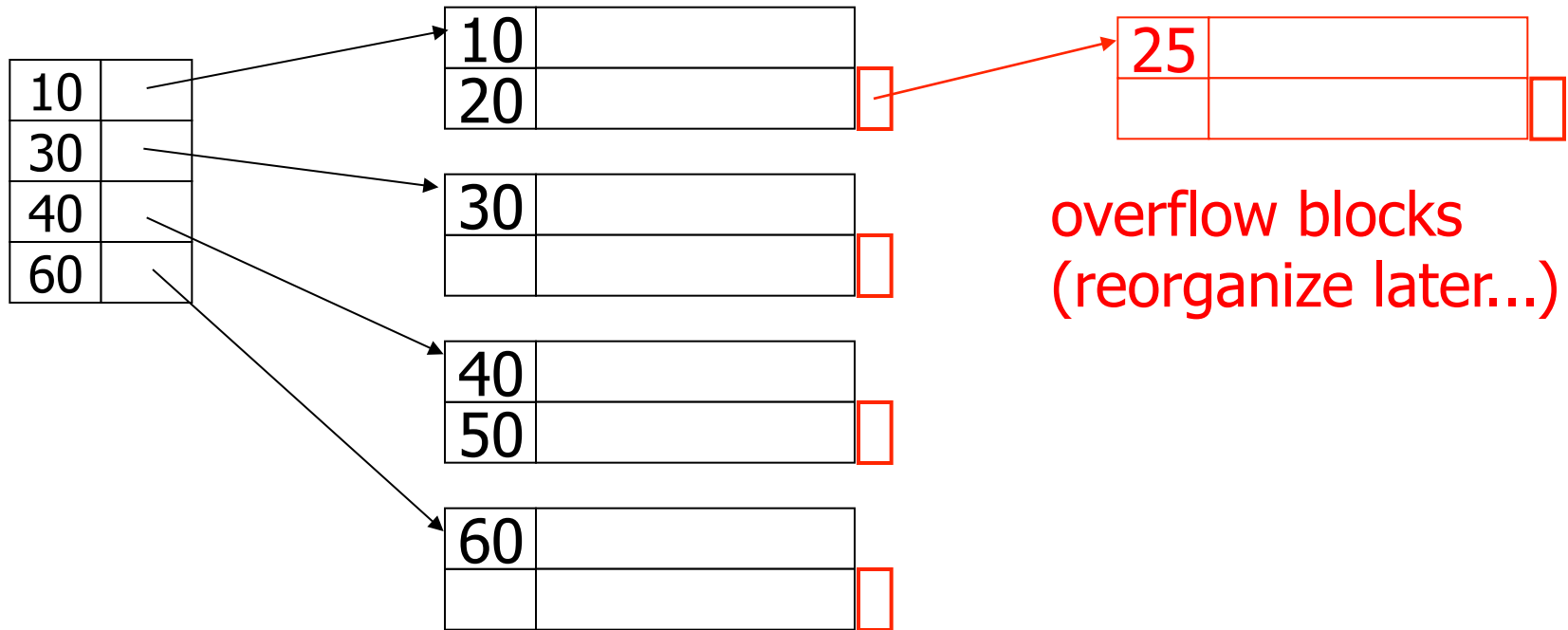 - insert new block (chained file)
  - update index

Insertion, sparse index case

– insert record 25

10	
30	
40	
60	

10	
20	

30	

40	
50	

60	

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Insertion, sparse index case

– insert record 25

overflow blocks
(reorganize later...)

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Insertion, dense index case

- Similar

- Often more expensive . . .

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Secondary indexes

Sequence field

30
50

20
70

80
40

100
10

90
60

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Secondary indexes

- Sparse index

Sequence field

30	
20	
80	
100	

90	
...	

30	
50	

20	
70	

80	
40	

100	
10	

90	
60	

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Secondary indexes

- Sparse index

Sequence field

30	
20	
80	
100	

90	
...	

30	
50	

20	
70	

80	
40	

100	
10	

90	
60	

does not make sense!

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Secondary indexes

- Dense index

Sequence field

| 30 | |
| 50 | |

| 20 | |
| 70 | |

| 80 | |
| 40 | |

| 100 | |
| 10 | |

| 90 | |
| 60 | |

Secondary indexes

- Dense index

Sequence field

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Secondary indexes

- Dense index

Sequence field

10
20
30
40

50
60
70
...

10
50
90
...

sparse high level

30
50

20
70

80
40

100
10

90
60

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
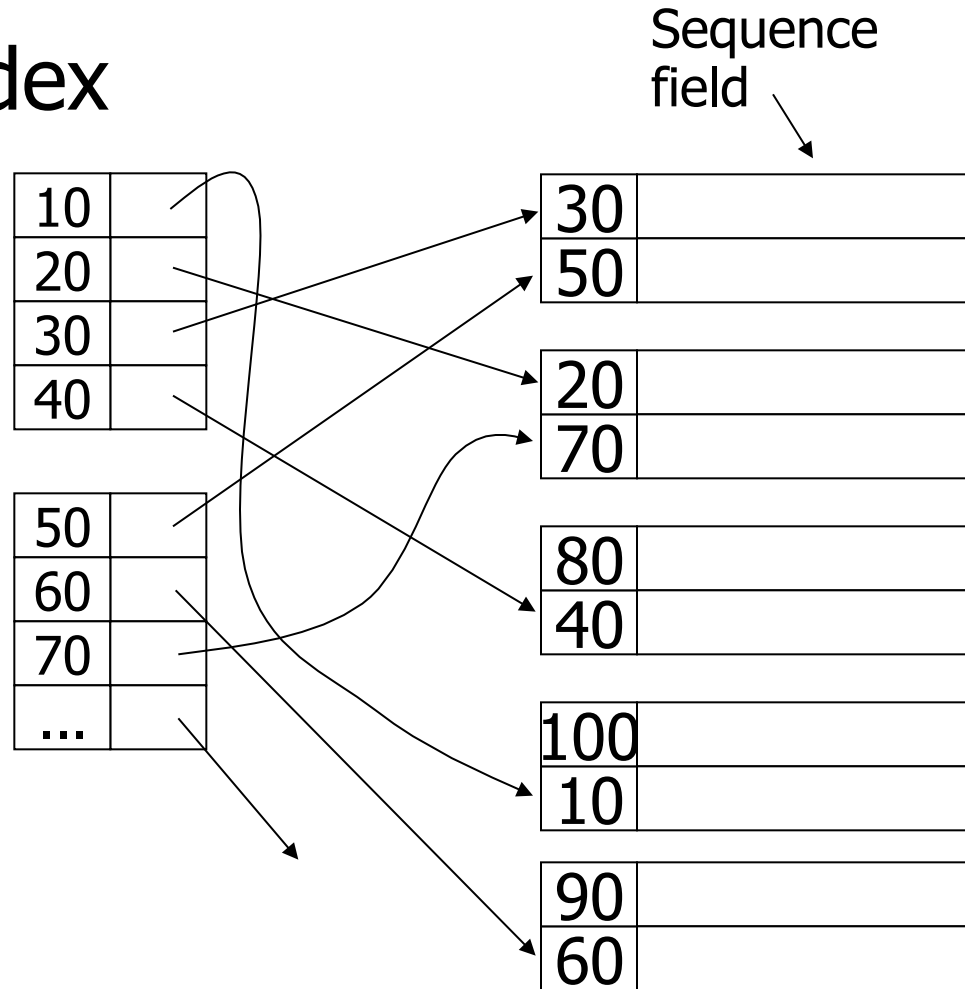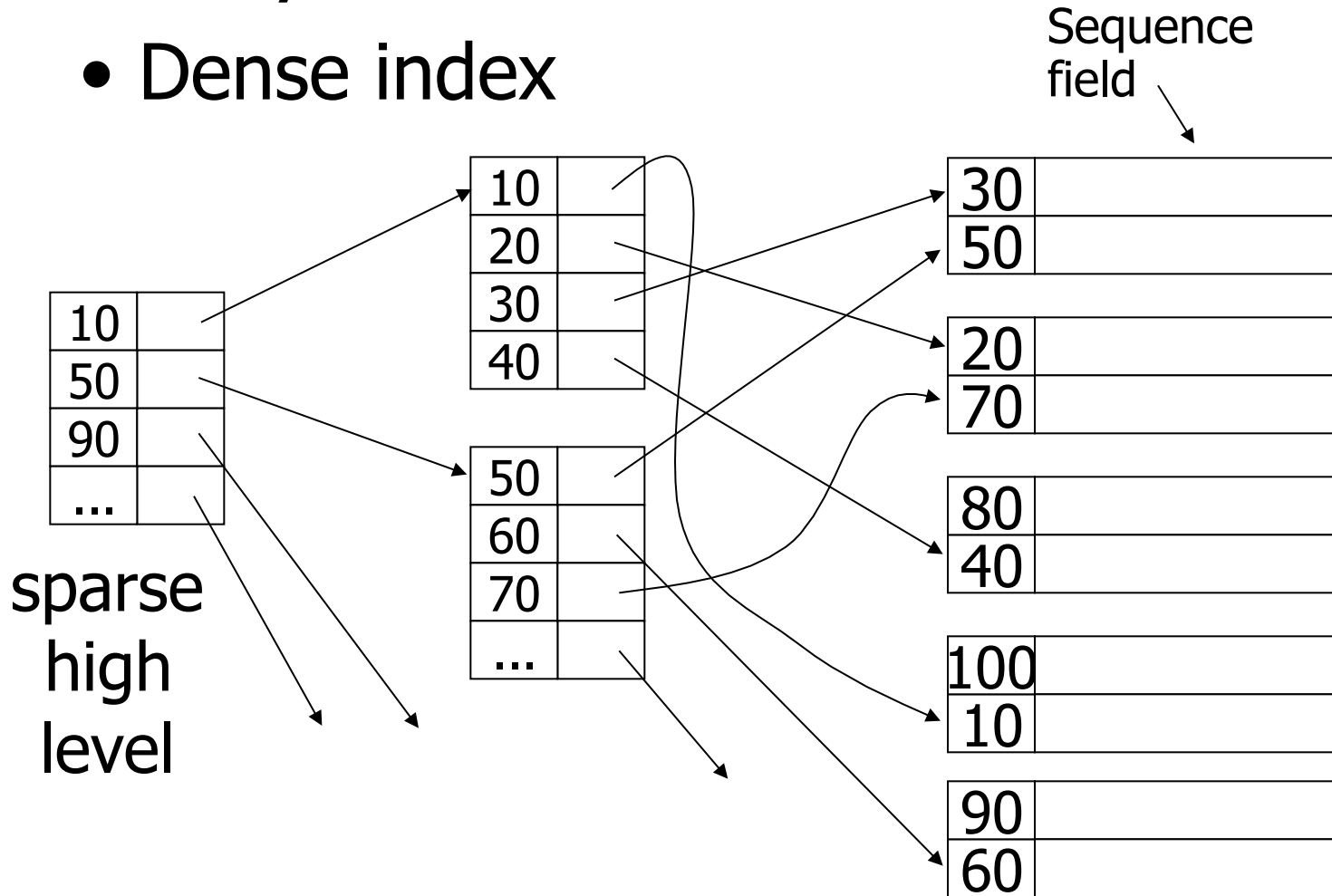
COMPUTER
SCIENCE

With secondary indexes:

- Lowest level is dense
- Other levels are sparse

Also: Pointers are record pointers
(not block pointers; not computed)

Notes 4 - Indexing 54

Duplicate values & secondary indexes

20	
10	

20	
40	

10	
40	

10	
40	

30	
40	

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Duplicate values & secondary indexes

one option…

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
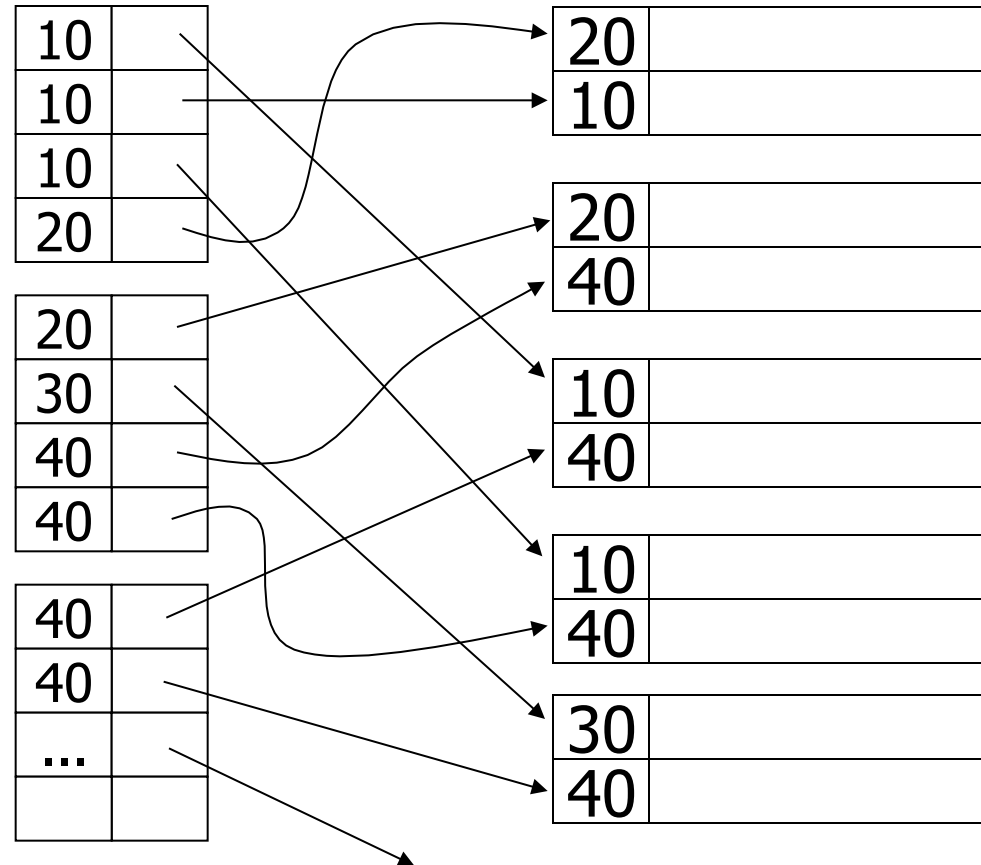
Duplicate values & secondary indexes

one option…

Problem:
excess overhead!
- disk space
- search time

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
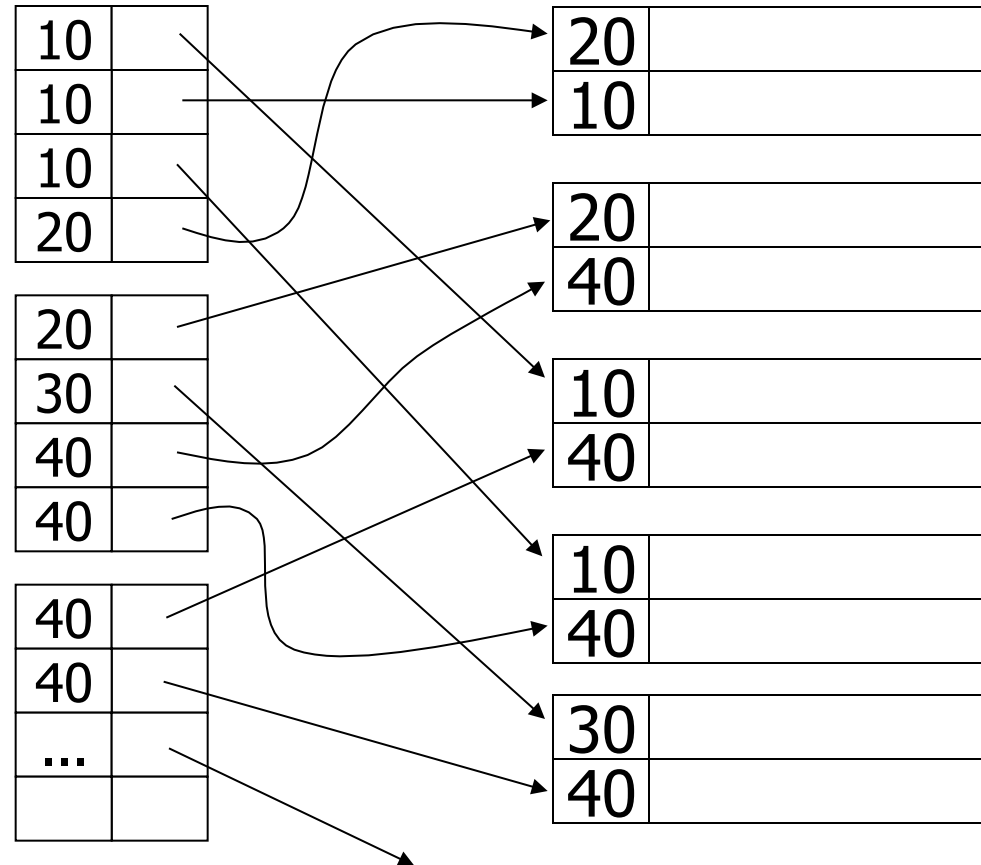
Duplicate values & secondary indexes
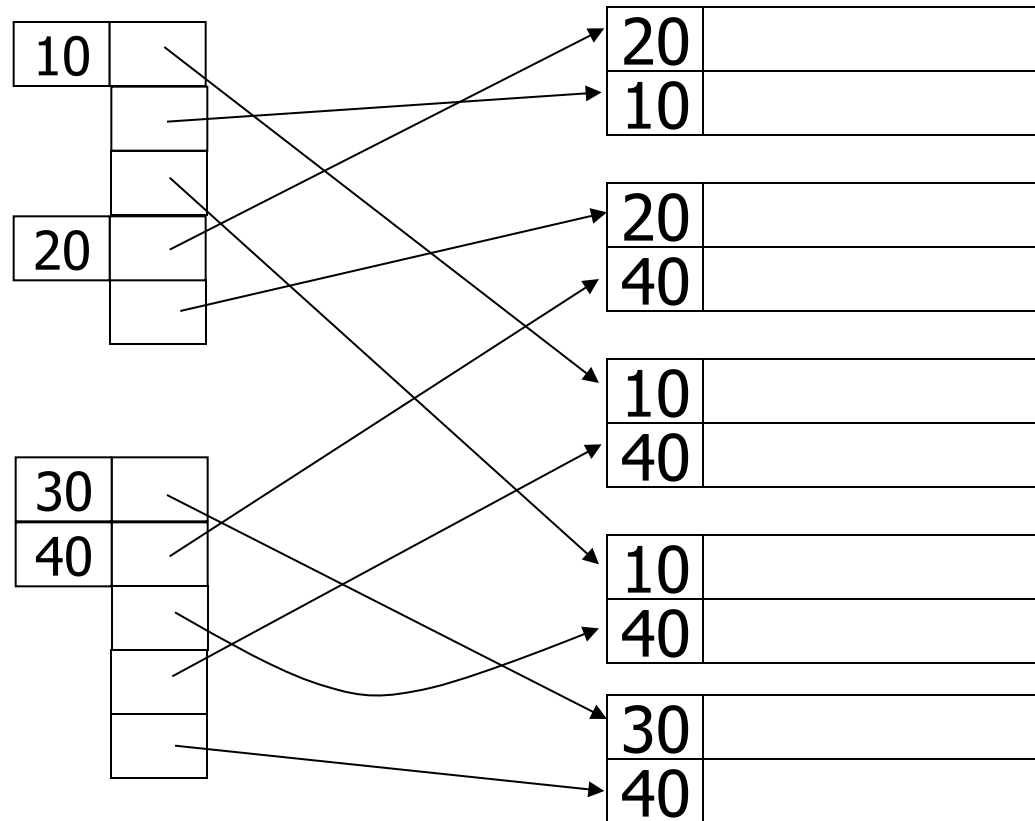
another option…

Duplicate values & secondary indexes

another option…

Problem: variable size records in index!

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate values & secondary indexes

Another idea:
Chain records with same key?

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
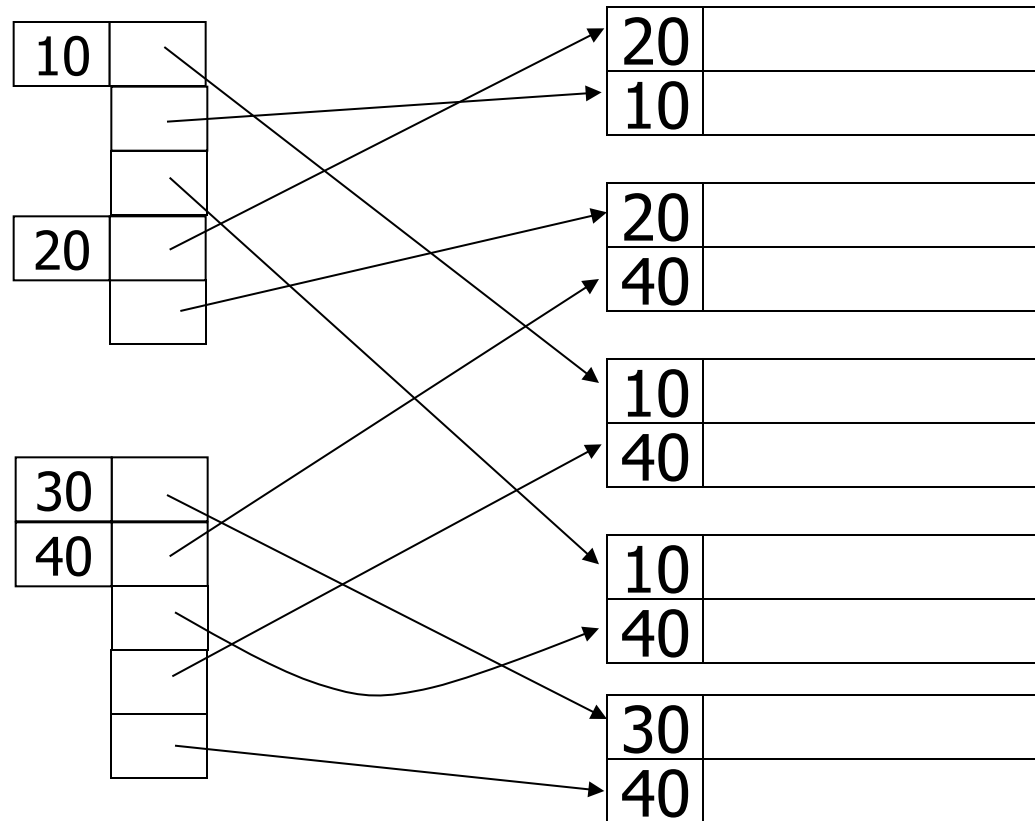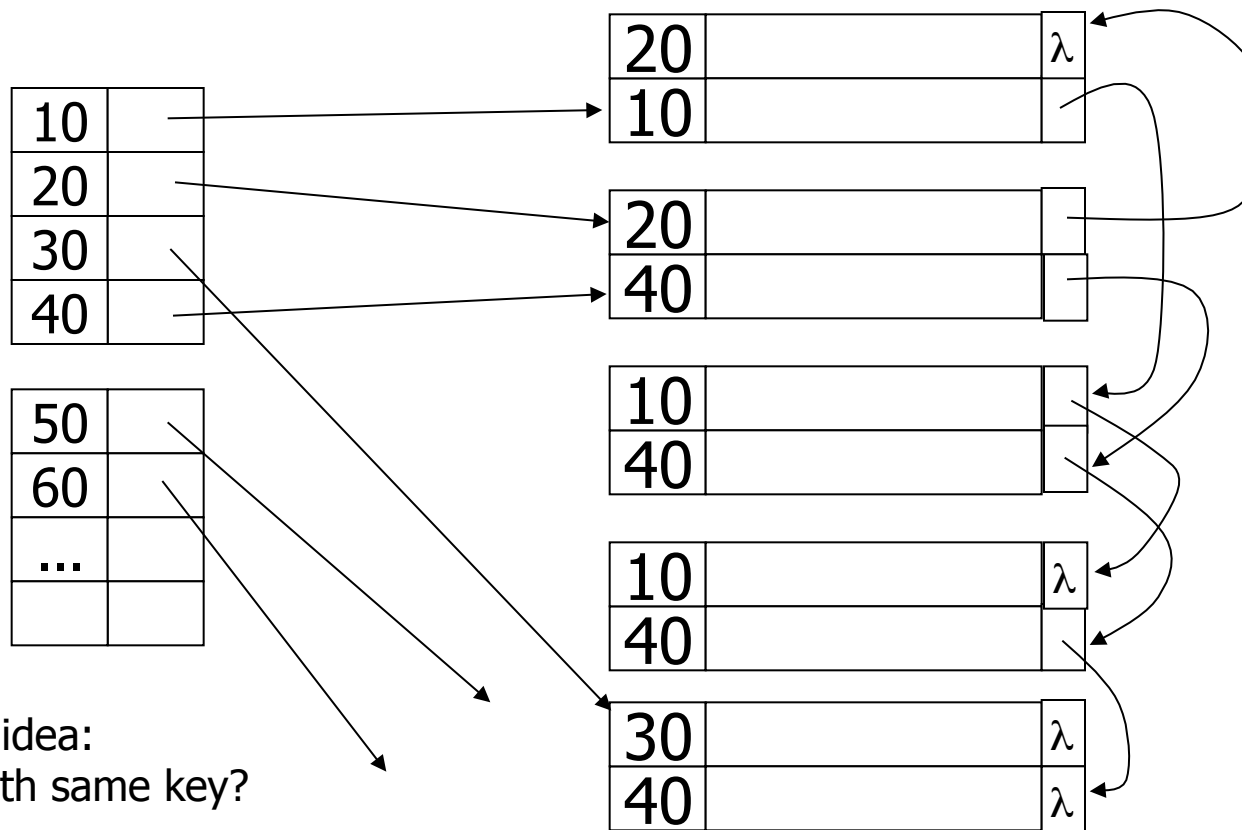
Duplicate values & secondary indexes
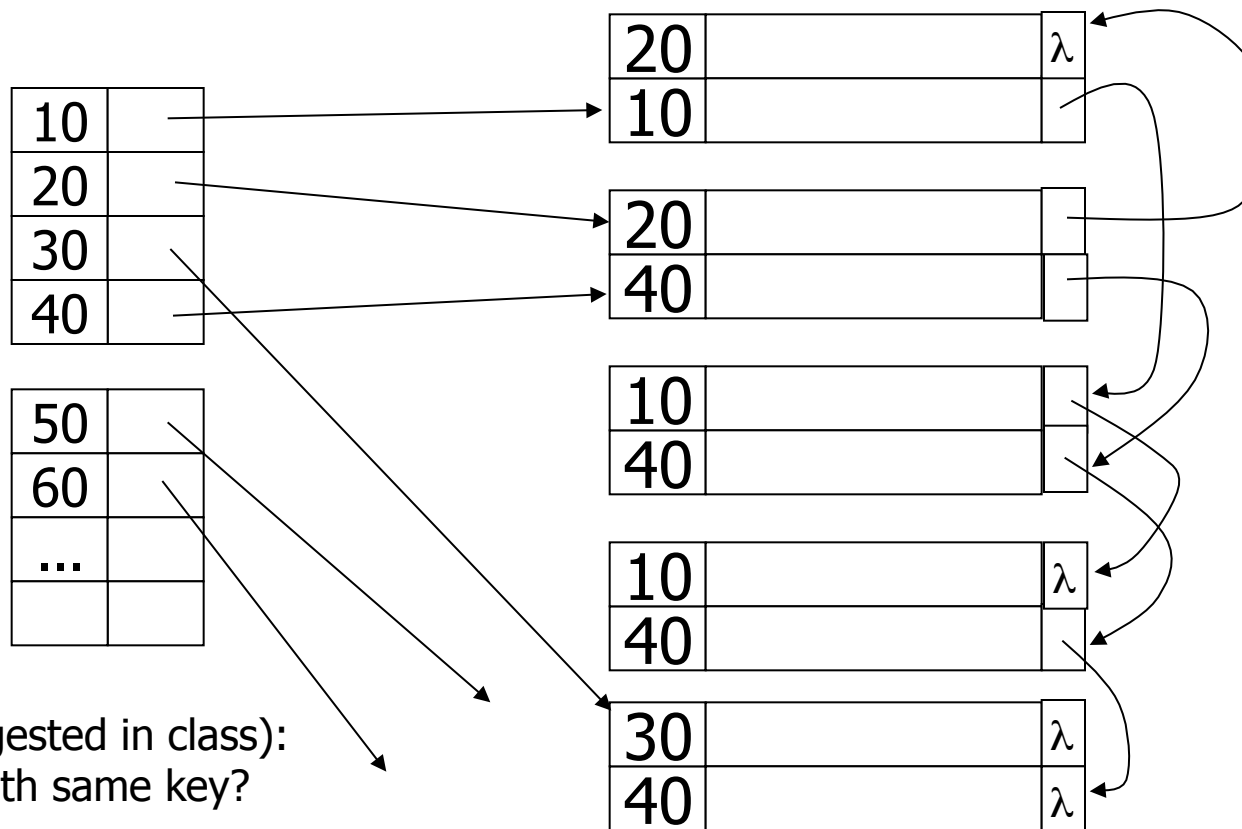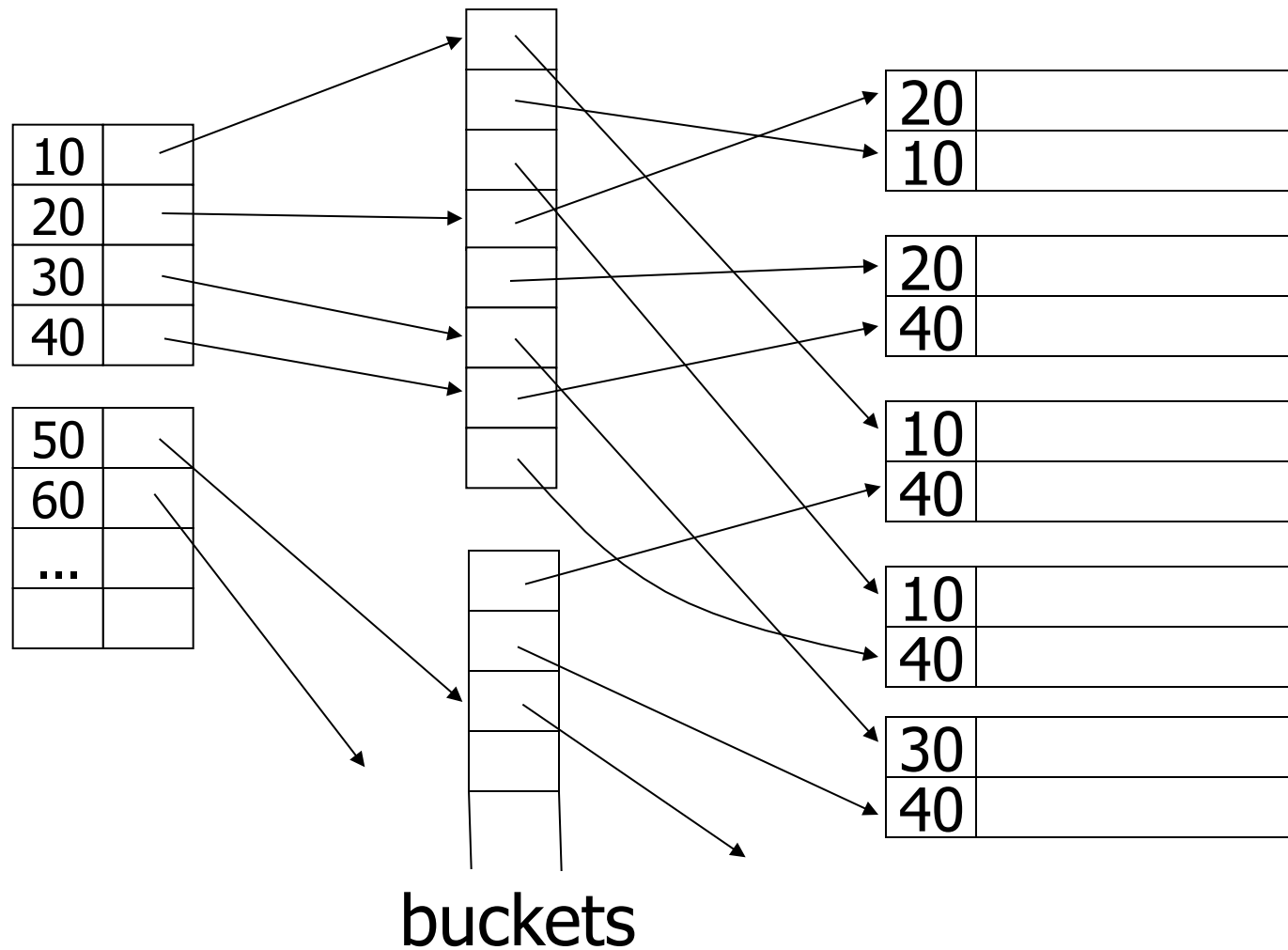

Another idea (suggested in class):
Chain records with same key?

Problems:
• Need to add fields to records
• Need to follow chain to know records

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Duplicate values & secondary indexes

buckets

Why "bucket" idea is useful

Indexes	Records
Name: primary	EMP (name,dept,floor,...)
Dept: secondary	
Floor: secondary	

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Query: Get employees in
(Toy Dept) ∧ (2nd floor)

Dept. index   EMP   Floor index

Toy   2nd

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Query: Get employees in
(Toy Dept) $\wedge$ (2nd floor)

Dept. index   EMP   Floor index

Toy   2nd
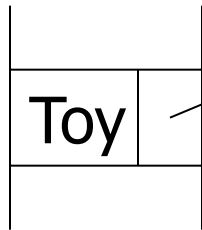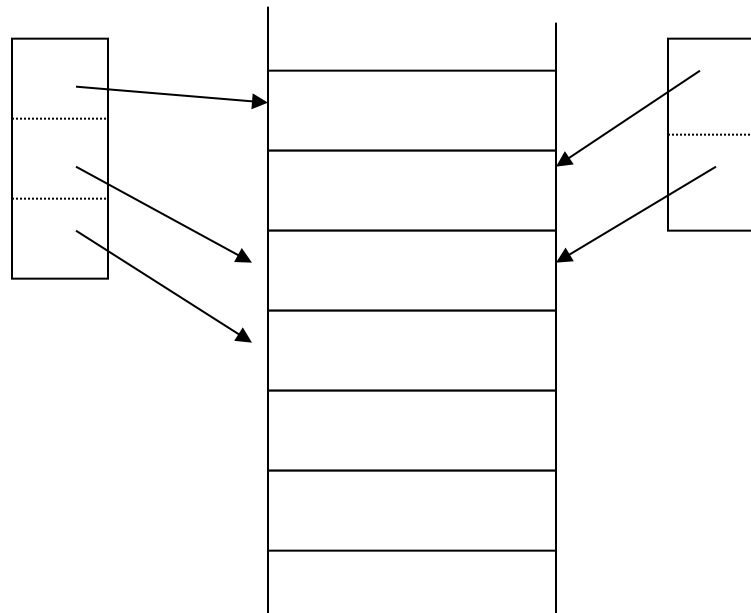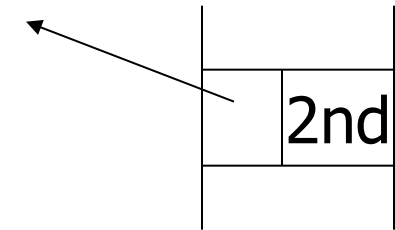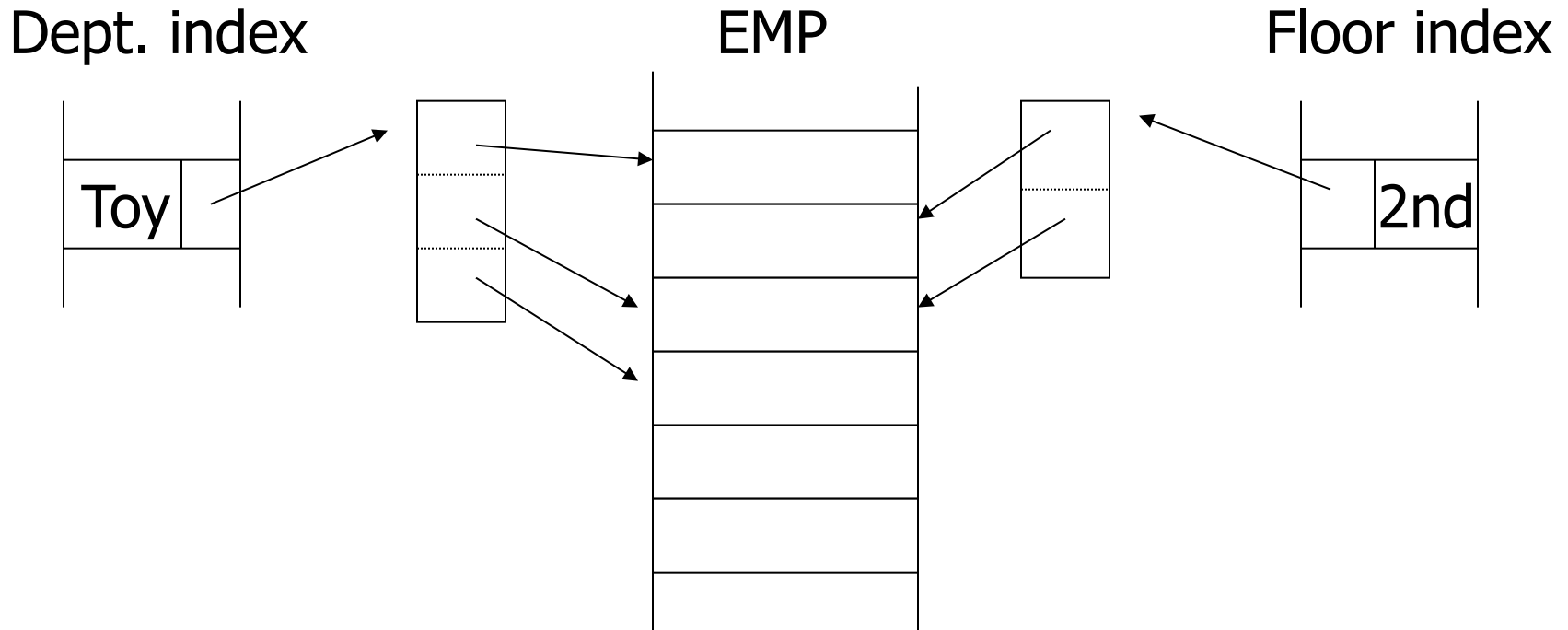
$\rightarrow$ Intersect toy bucket and 2nd Floor bucket to get set of matching EMP's

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

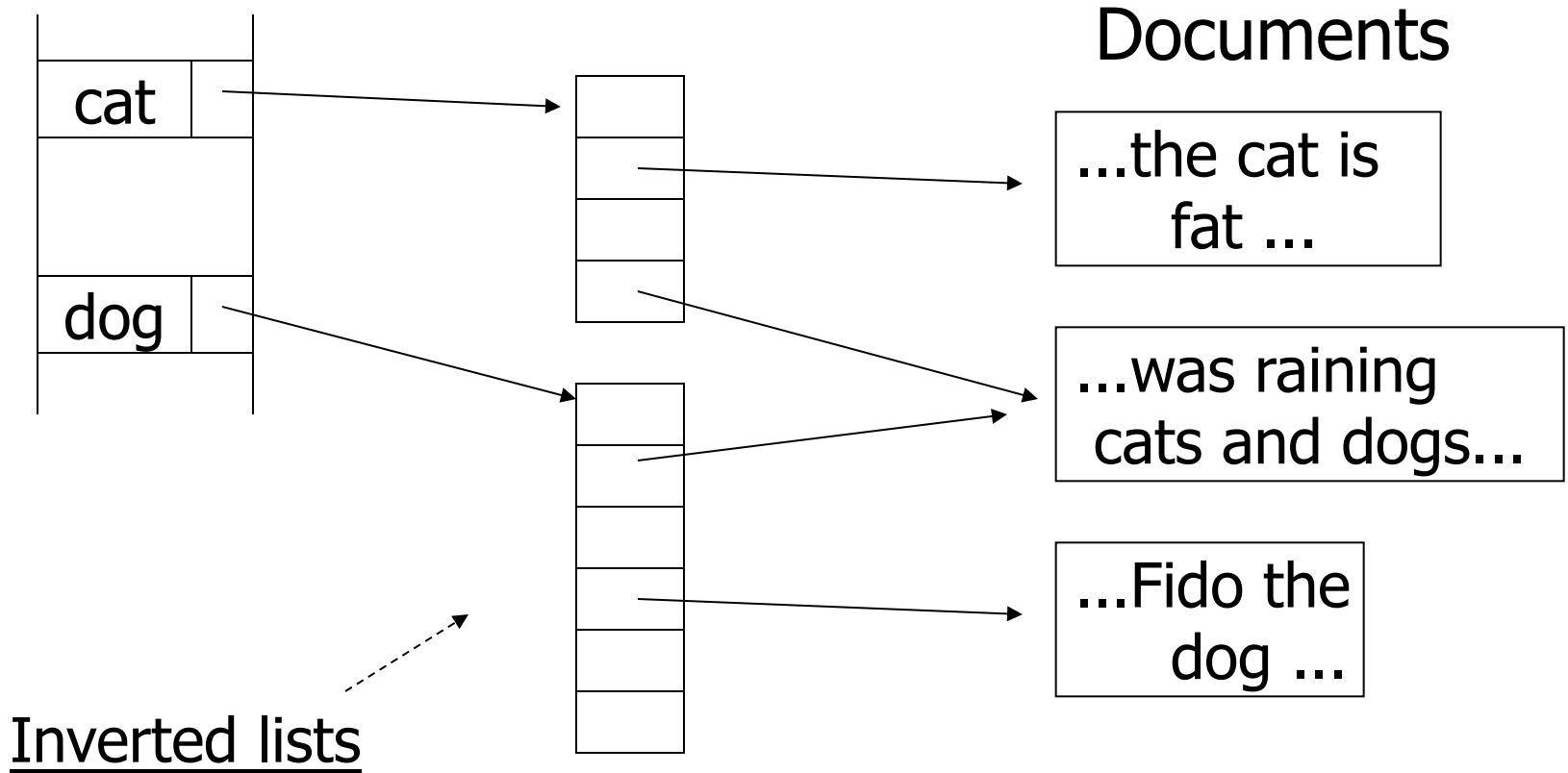# This idea used in text information retrieval

Documents

...the cat is fat ...

...was raining cats and dogs...

...Fido the dog ...

This idea used in
text information retrieval

Documents

cat

dog

Inverted lists

...the cat is fat ...

...was raining cats and dogs...

...Fido the dog ...

COMPUTER SCIENCE

Notes 4 - Indexing

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

IR QUERIES

- Find articles with "cat" and "dog"
- Find articles with "cat" or "dog"
- Find articles with "cat" and not "dog"

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Summary so far

- Conventional index
  - Basic Ideas: sparse, dense, multi-level…
  - Duplicate Keys
  - Deletion/Insertion
  - Secondary indexes
    - Buckets of Postings List

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Conventional indexes

Advantage:
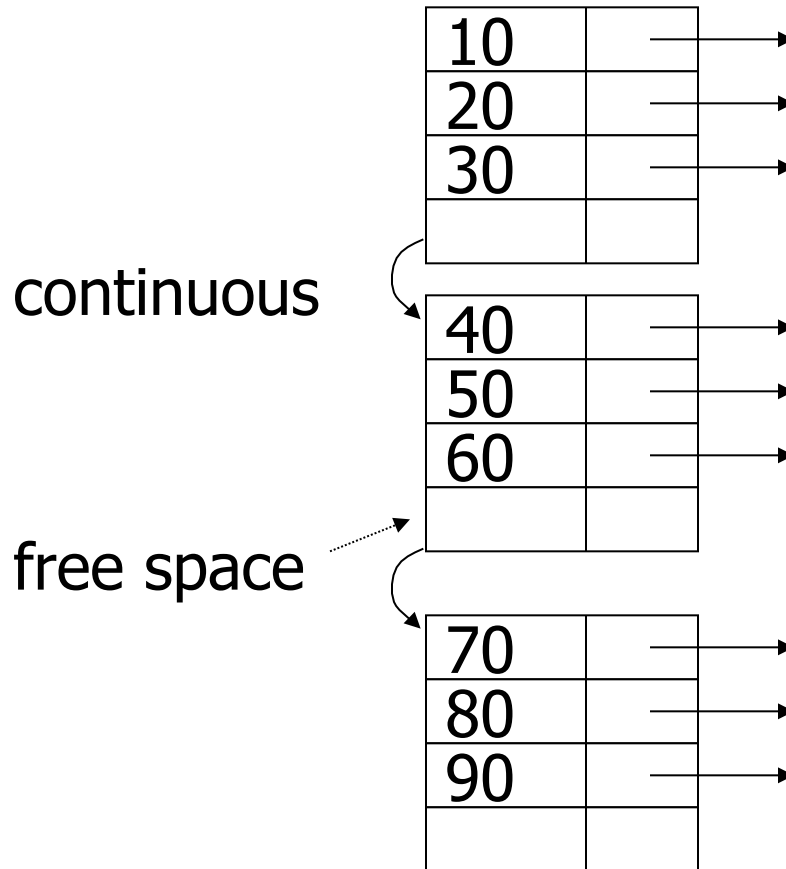
   - Simple
   - Index is sequential file
          good for scans

Disadvantage:

   - Inserts expensive, and/or
   - Lose sequentiality & balance

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example   Index (sequential)

10		→
20		→
30		→

continuous

40		→
50		→
60		→

free space

70		→
80		→
90		→

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example   Index (sequential)

continuous

free space

overflow area
(not sequential)

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
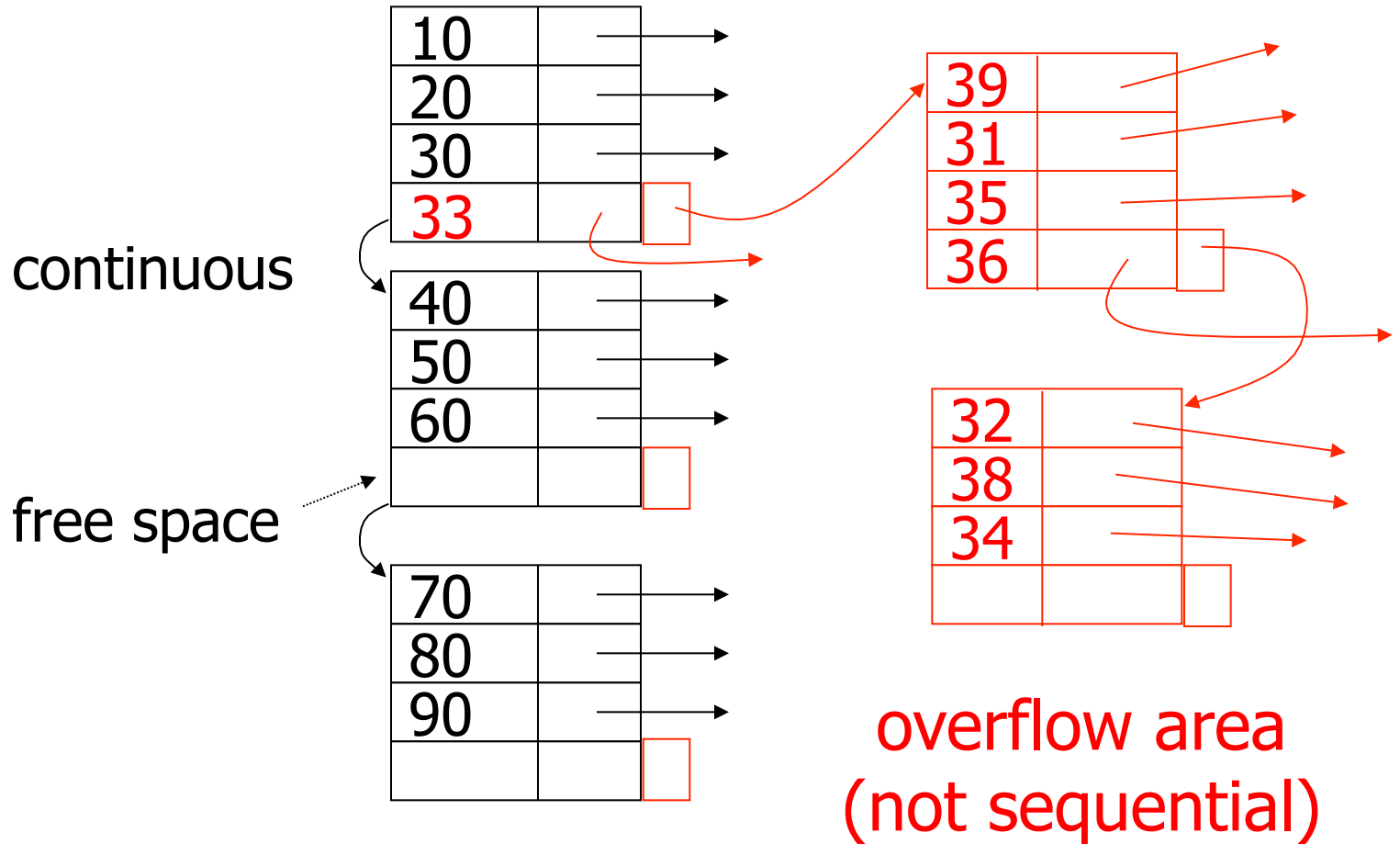
Outline:

- Conventional indexes
- B-Trees $\Rightarrow$ NEXT
- Hashing schemes
- Advanced Index Techniques

- NEXT: Another type of index
  - Give up on sequentiality of index
  - Try to get "balance"

Notes 4 - Indexing

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
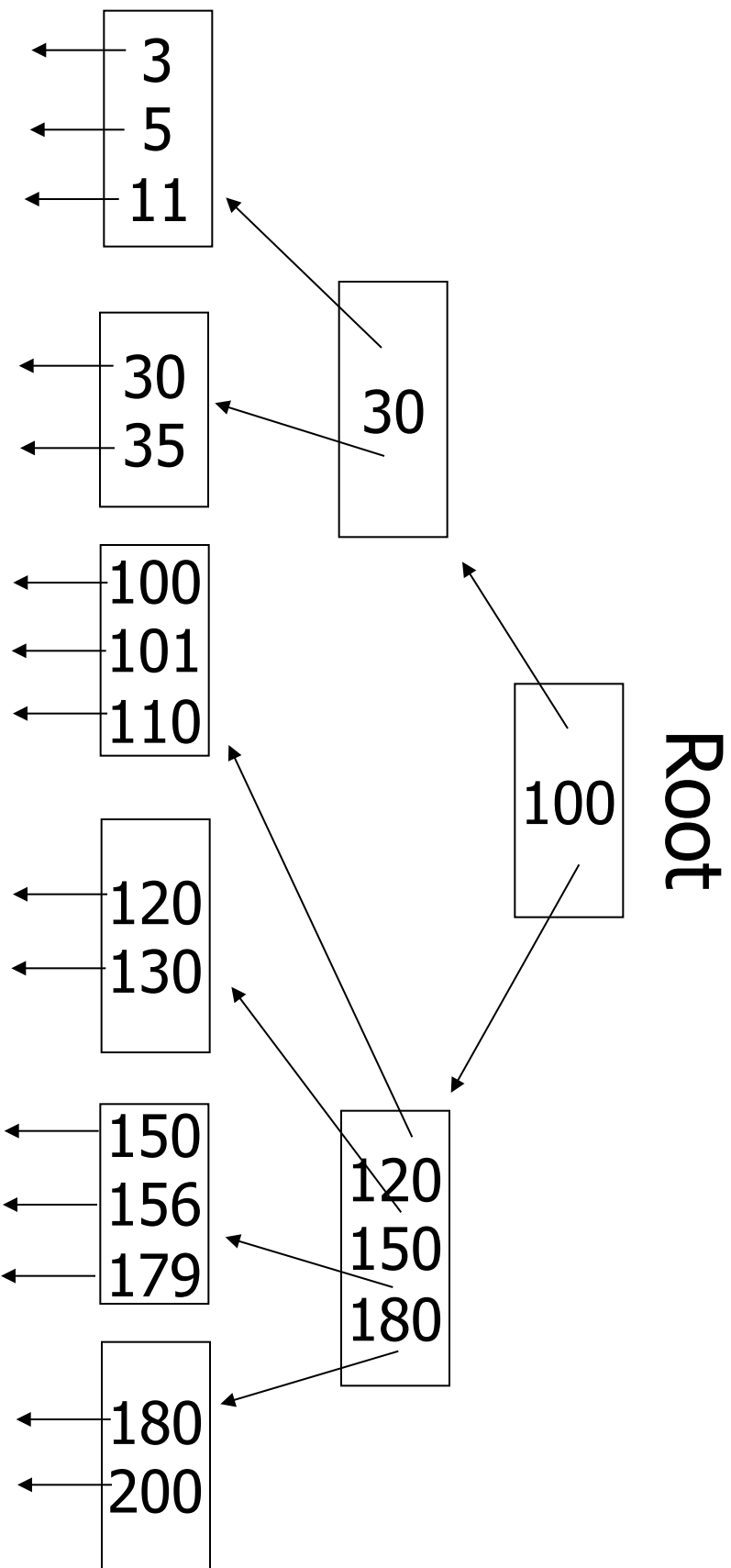
B+-tree Motivation

- Tree indices are pretty efficient
  - E.g., binary search tree
    - Average case $O(\log(n))$ lookup

- However

  - Unclear how to map to disk (index larger than main memory, loading partial index)

  - Worst-case $O(n)$ lookup

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
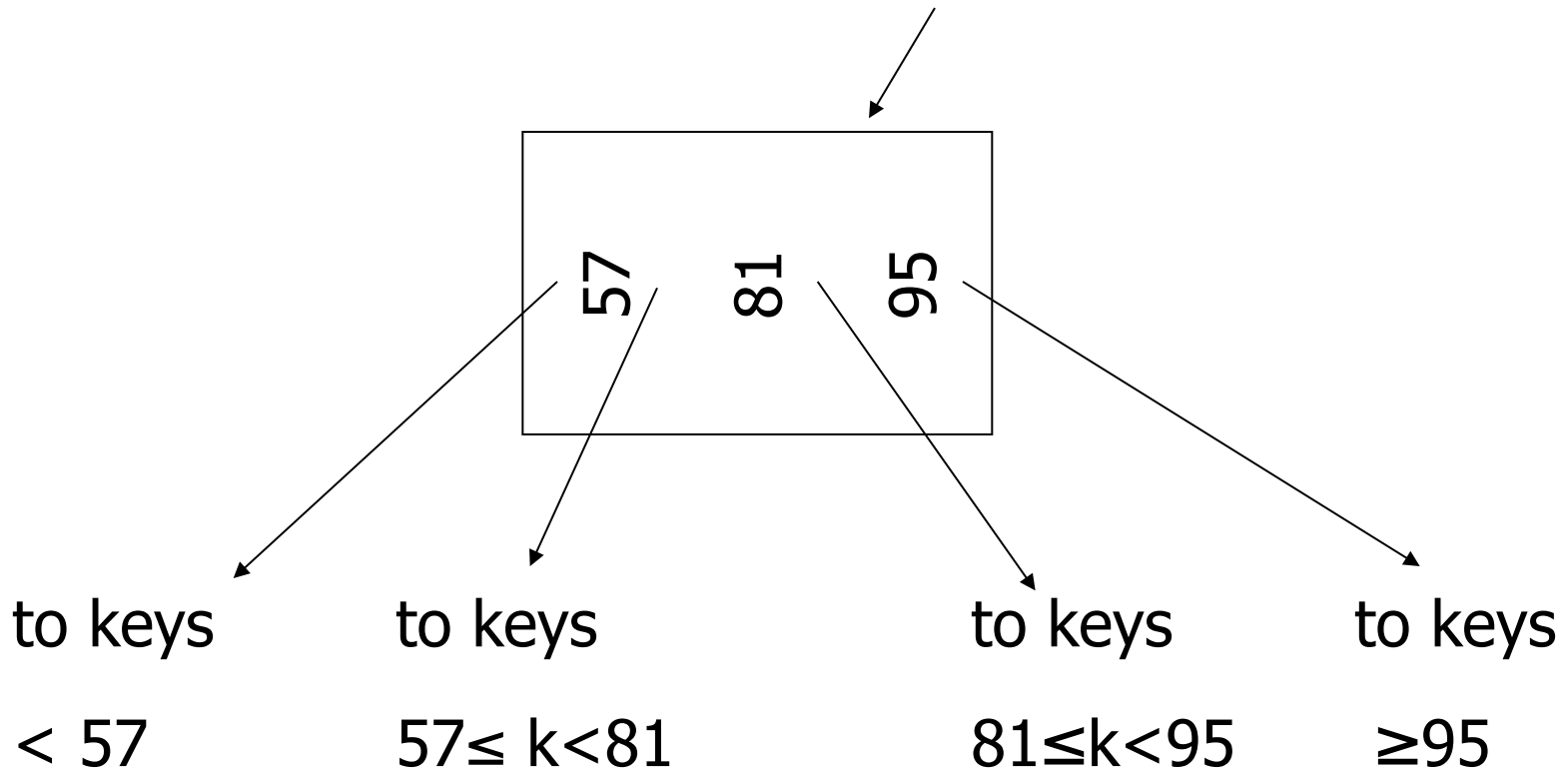
B+-tree Properties

- Large nodes:
  - Node size is multiple of block size
    - -> small number of levels
    - -> simple way to map index to disk
    - -> many keys per node

- Balance:
  - Require all nodes to be more than X% full
  - -> for n records guaranteed only logarithmically many levels
  - -> log(n) worst-case performance

COMPUTER SCIENCE

IIT College of
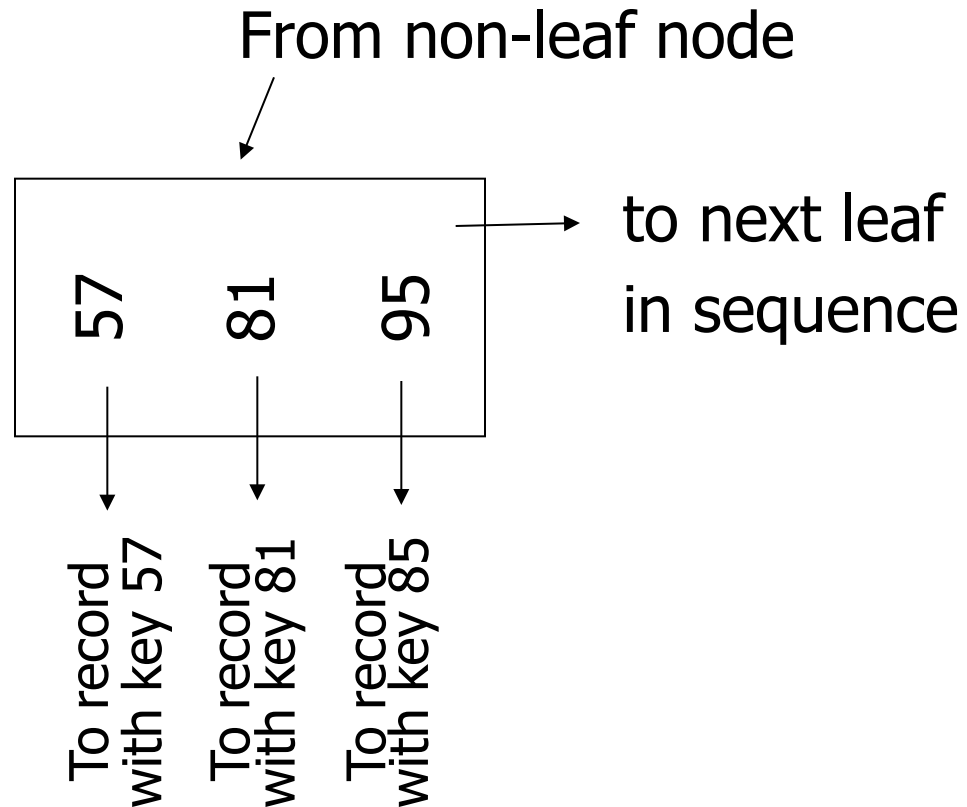Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

B+Tree Example

n=3

3
5
11

30

30
35

Root

100

100
101
110

120
130

120
150
180

150
156
179

180
200

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Sample non-leaf

57 81 95

to keys

< 57

to keys

57≤ k<81

to keys

81≤k<95

to keys

≥95

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Sample leaf node:

From non-leaf node

57 81 95

to next leaf
in sequence

To record with key 57
To record with key 81
To record with key 85

In textbook's notation   n=3

Leaf:

Non-leaf:

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
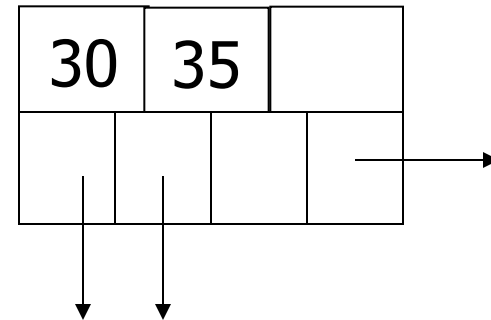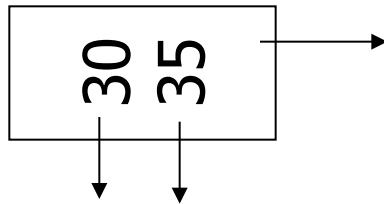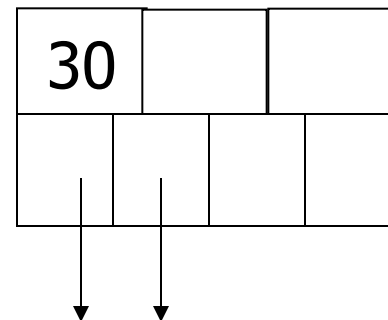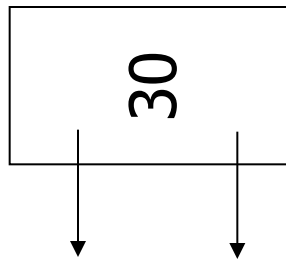
Size of nodes: { n+1 pointers
                  n keys   (fixed)

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Don't want nodes to be too empty

- Use at least (balance)

  Non-leaf: $\lceil (n+1)/2 \rceil$ pointers

  Leaf: $\lfloor (n+1)/2 \rfloor$ pointers to data

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

n=3

Full node   min. node

Non-leaf
| 120 150 180 | | 30 |

Leaf
| 3 5 11 | → | 30 35 | →

counts even if null

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
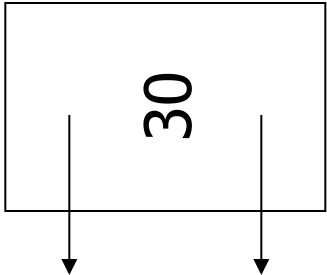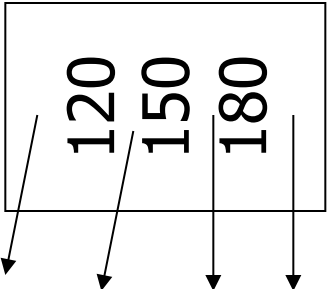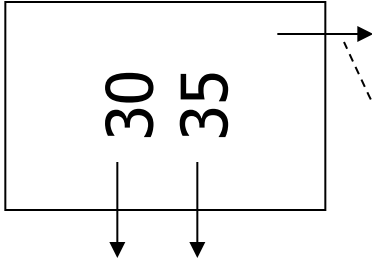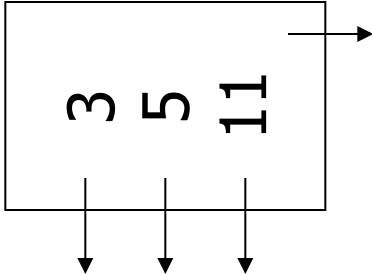
B+tree rules tree of order *n*

(1) All leaves at same lowest level
            (balanced tree)

   -> guaranteed worst-case complexity for operations on the index

(2) Pointers in leaves point to records
            except for "sequence pointer"

(3) Number of pointers/keys for B+tree

	Max ptrs	Max keys	Min ptrs→data	Min keys
Non-leaf (non-root)	$n+1$	n	$\lceil (n+1)/2 \rceil$	$\lceil (n+1)/2 \rceil - 1$
Leaf (non-root)	$n+1$	n	$\lfloor (n+1)/2 \rfloor$	$\lfloor (n+1)/2 \rfloor$
Root	$n+1$	n	1	1

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Search Algorithm

- Search for key **k**
- Start from root until leaf is reached
- For current node find i so that
  - Key[i] <= **k** < Key[i + 1]
  - Follow i+1th pointer
- If current node is leaf return pointer to record or fail (no such record in tree)

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Search Example   k= 120   n=3

Root

| 100 |

| 30 |

| 3 |
| 5 |
| 11 |

| 30 |
| 35 |

| 100 |
| 101 |
| 110 |

| 120 |
| 150 |
| 180 |

| 120 |
| 130 |

| 150 |
| 156 |
| 179 |

| 180 |
| 200 |
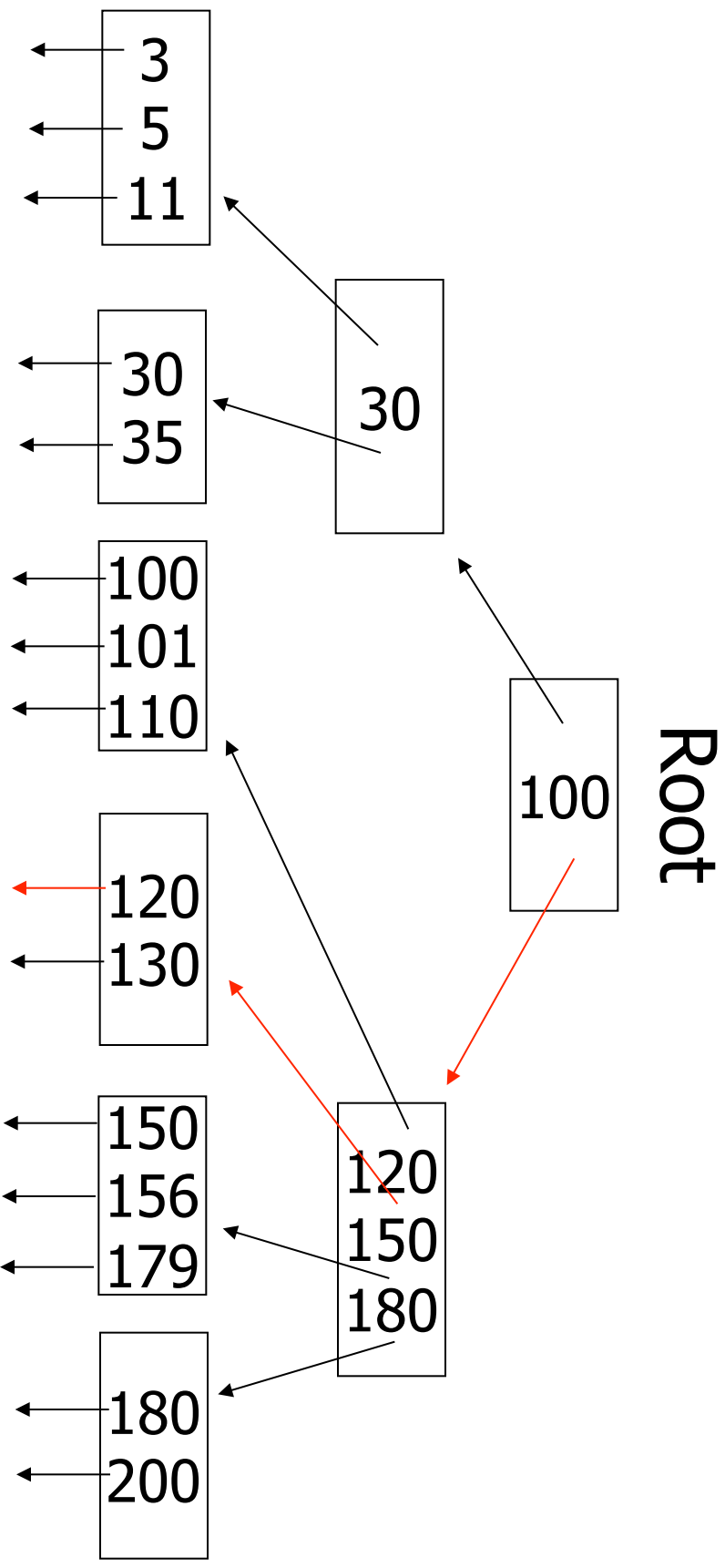
Remarks Search

- If **n** is large, e.g., 500
- Keys inside node are sorted
- -> use binary search to find **I**
- Performance considerations
  - Linear search $O(n)$
  - Binary search $O(\log_2(n))$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Insert into B+tree

(a) simple case

– space available in leaf

(b) leaf overflow

(c) non-leaf overflow

(d) new root

(a) Insert key = 32
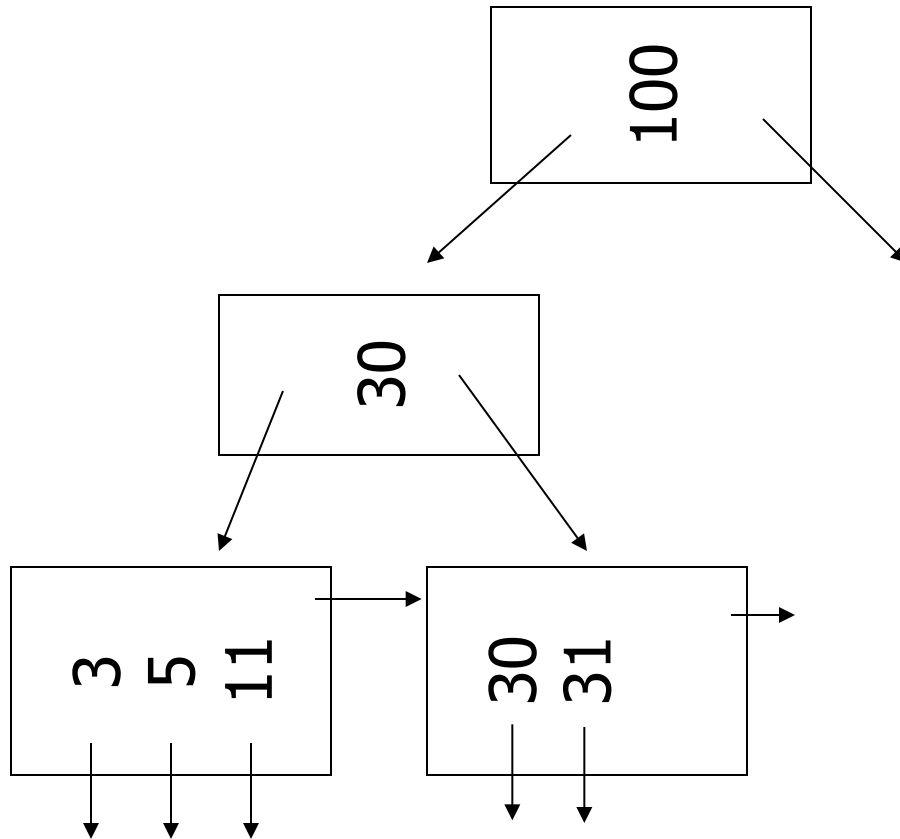
n=3

(a) Insert key = 32

n=3

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

(a) Insert key = 7

n=3

Notes 4 - Indexing
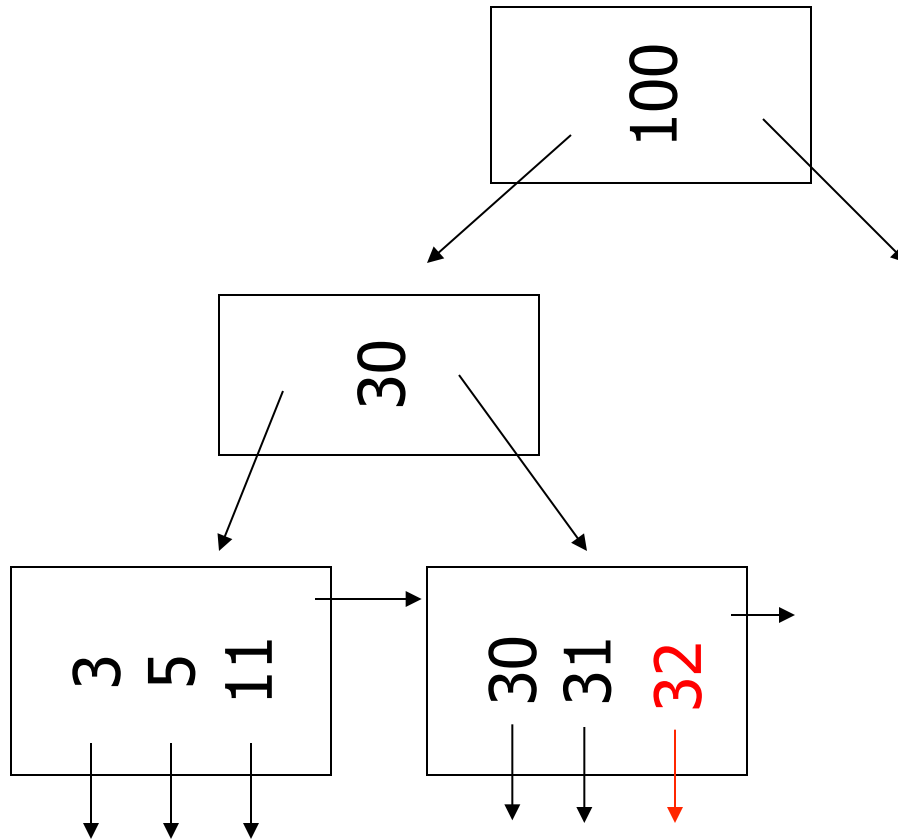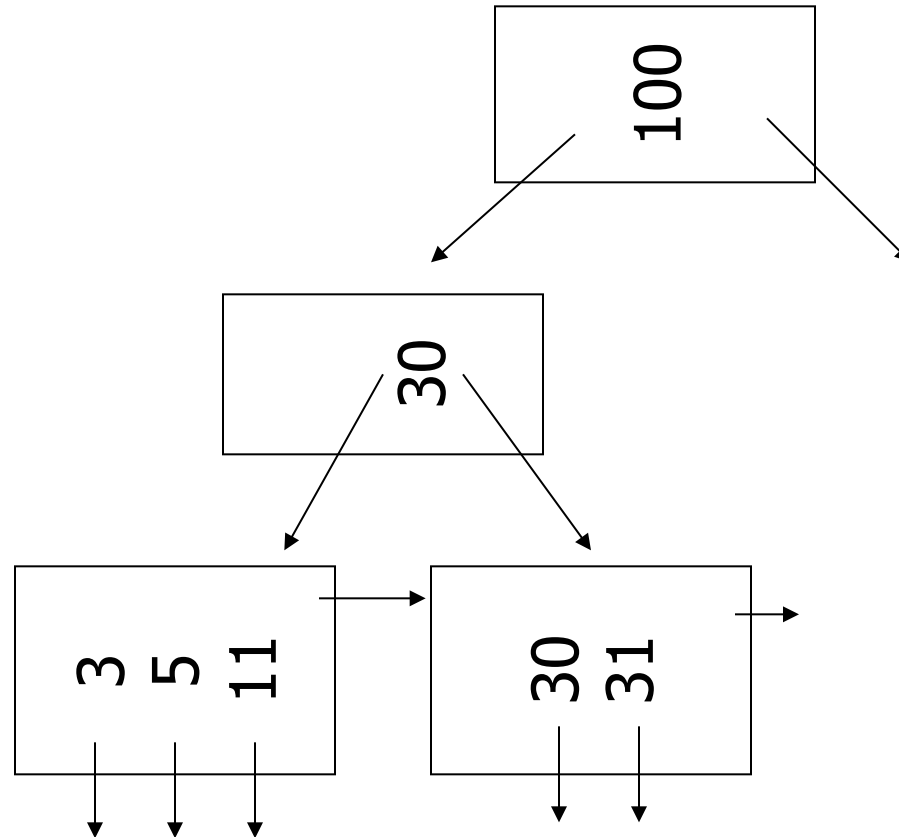
(a) Insert key = 7   n=3

(a) Insert key = 7

n=3

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

(c) Insert key = 160

n=3

100

120 150 180

150 156 179

180 200

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

(c) Insert key = 160

n=3

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

(c) Insert key = 160

n=3

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
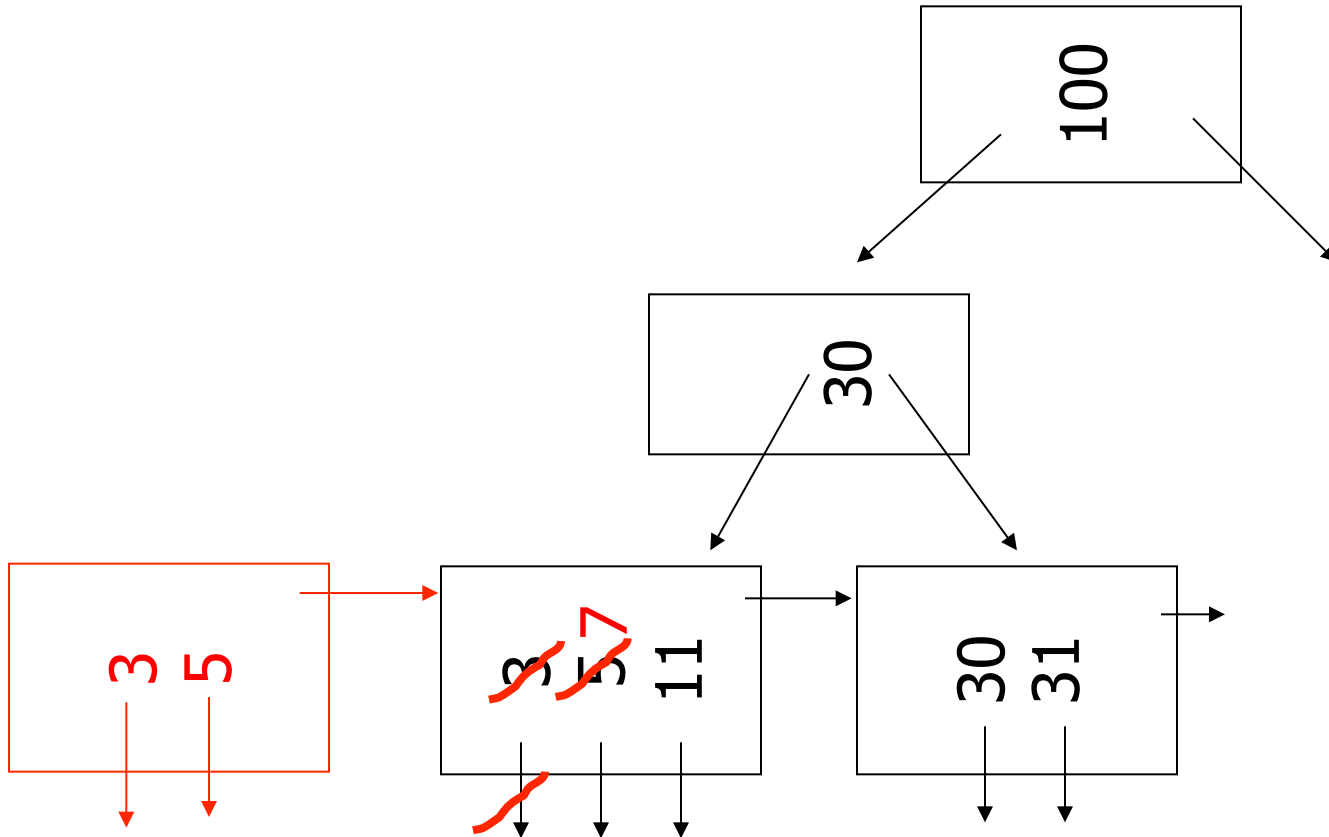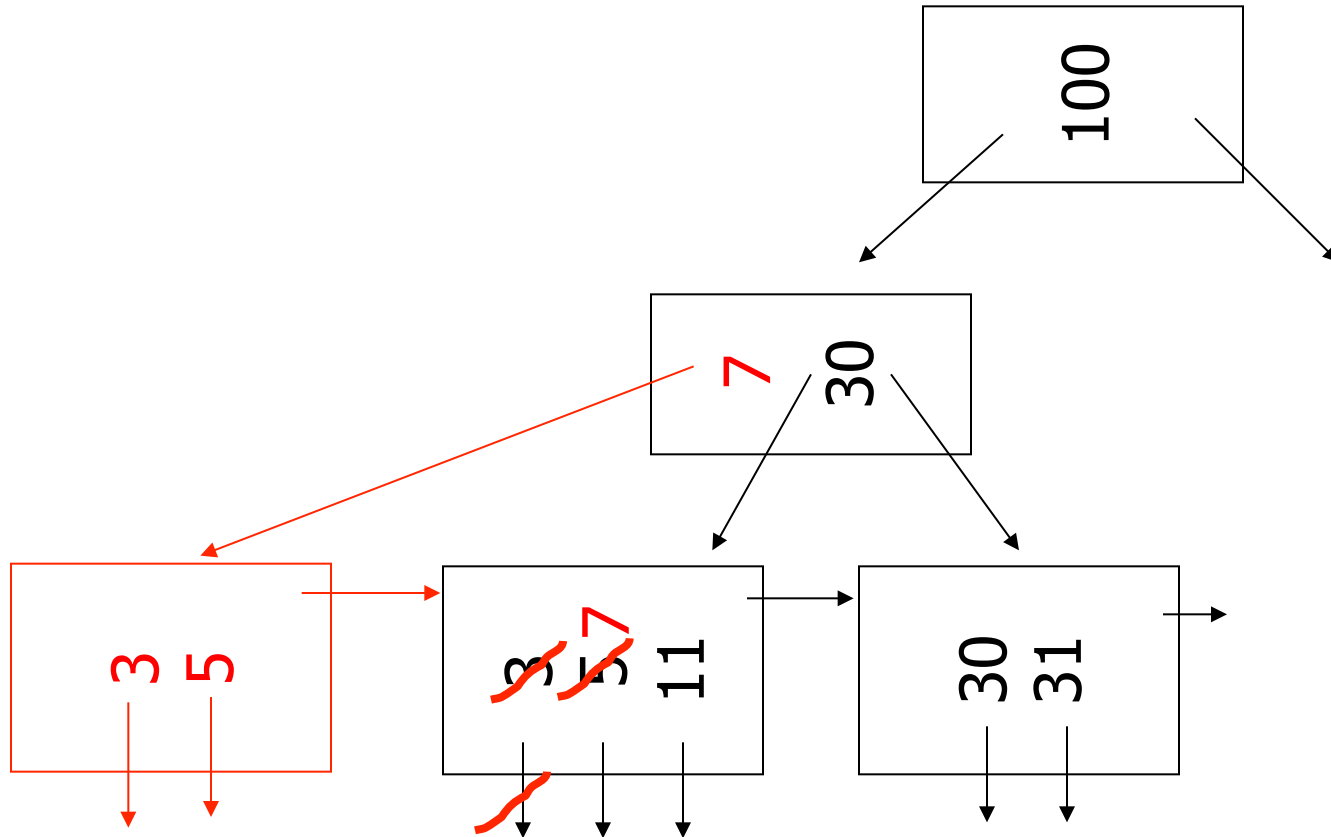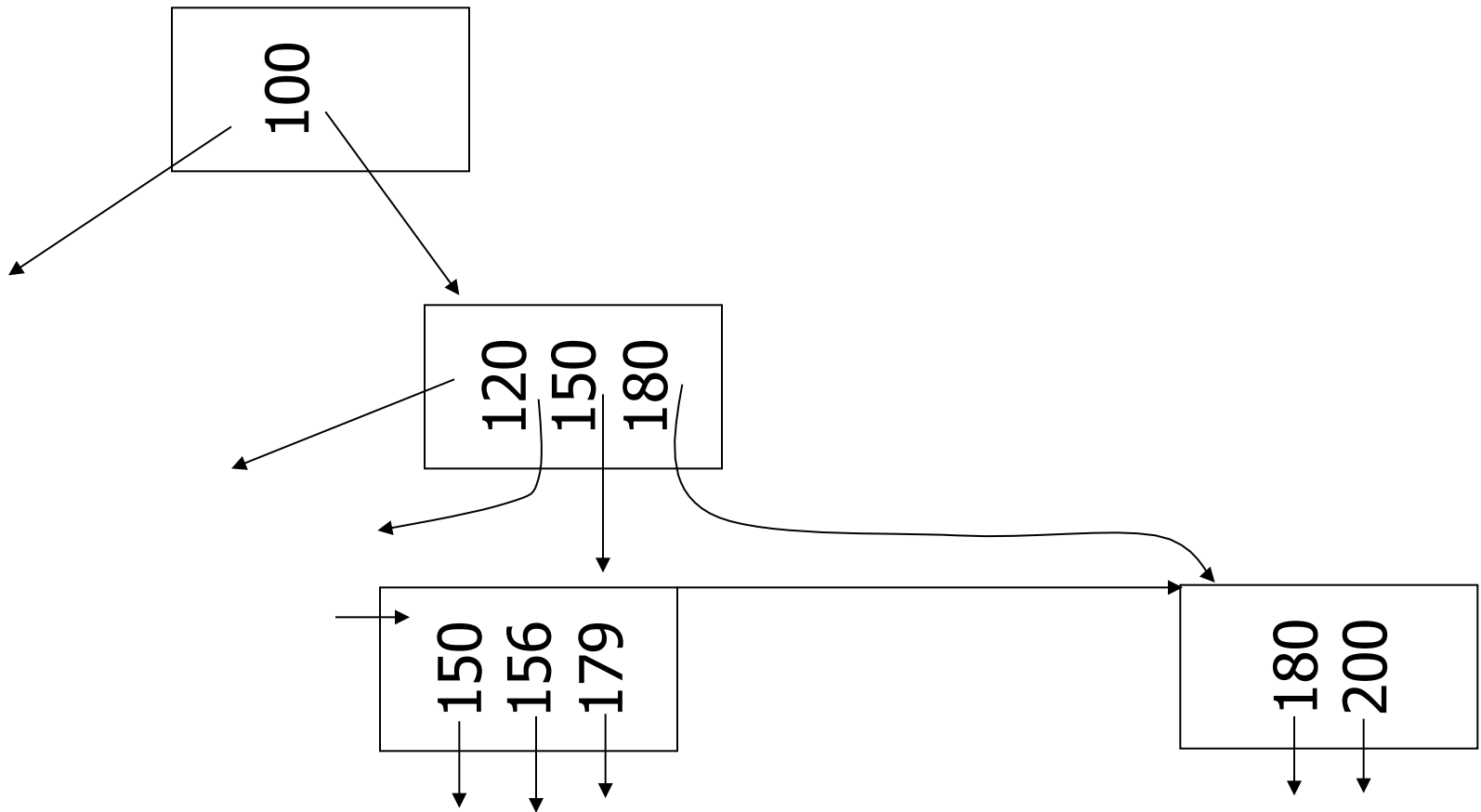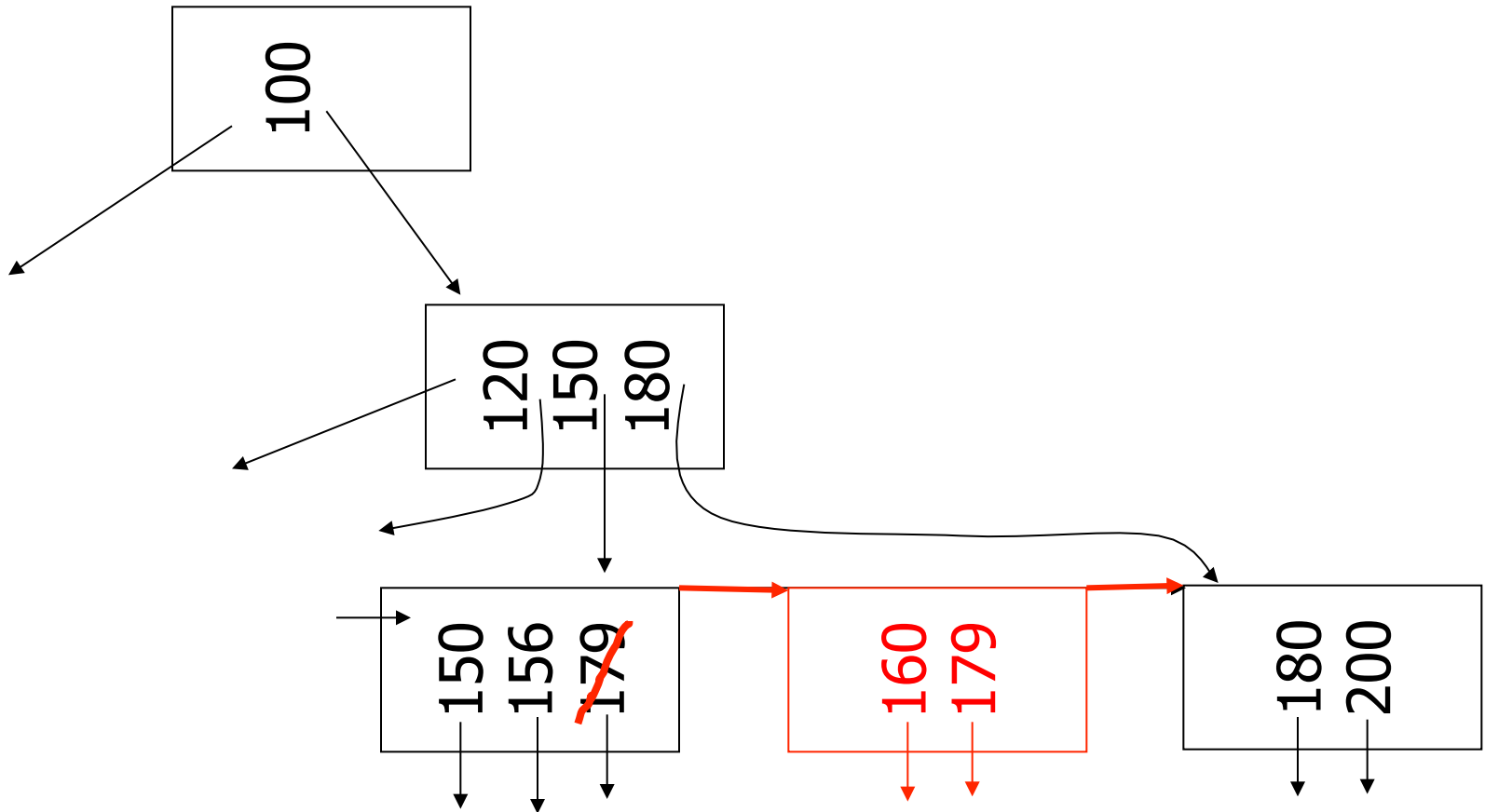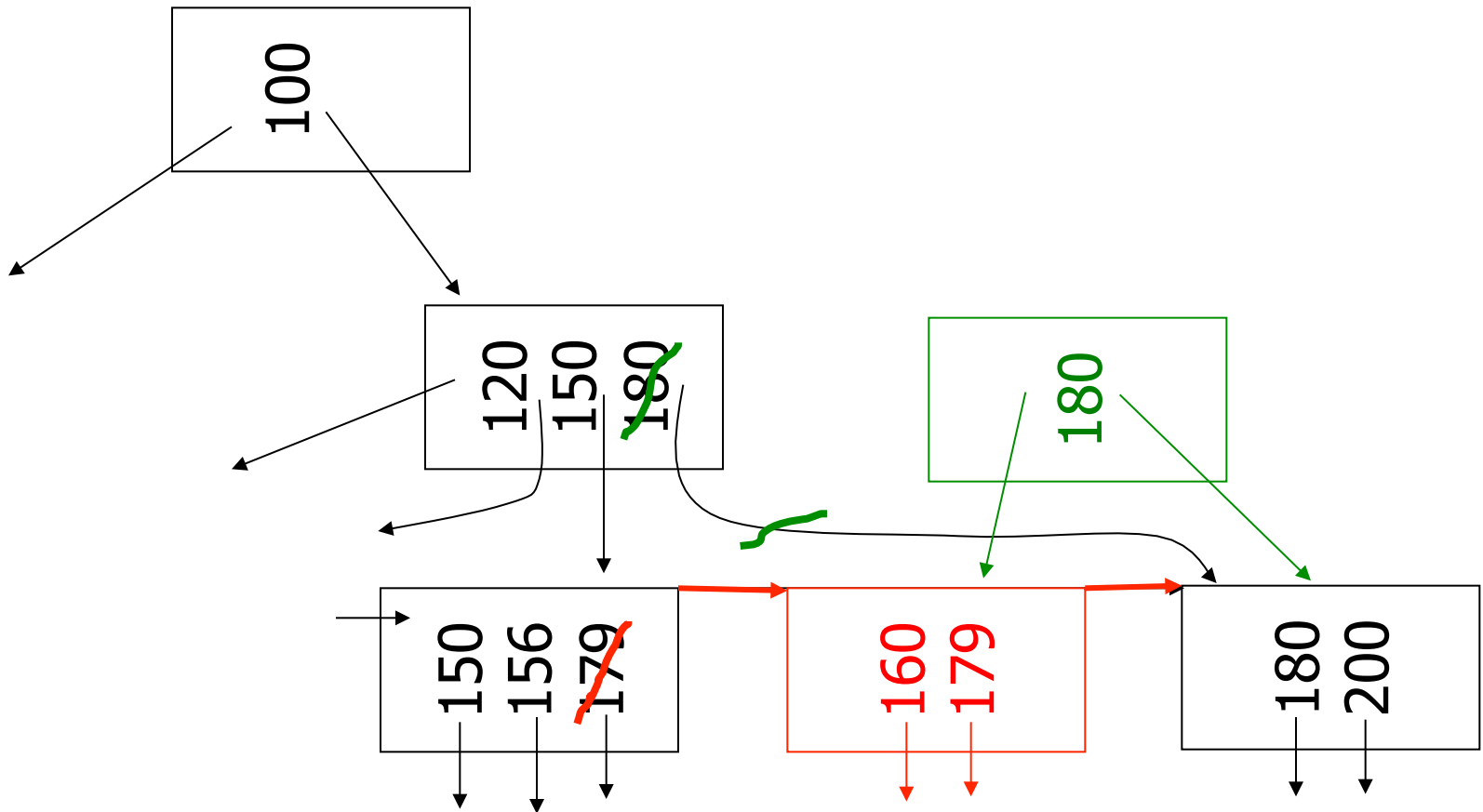
(c) Insert key = 160

n=3

(d) New root, insert 45

n=3

Notes 4 - Indexing

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

(d) New root, insert 45

n=3

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
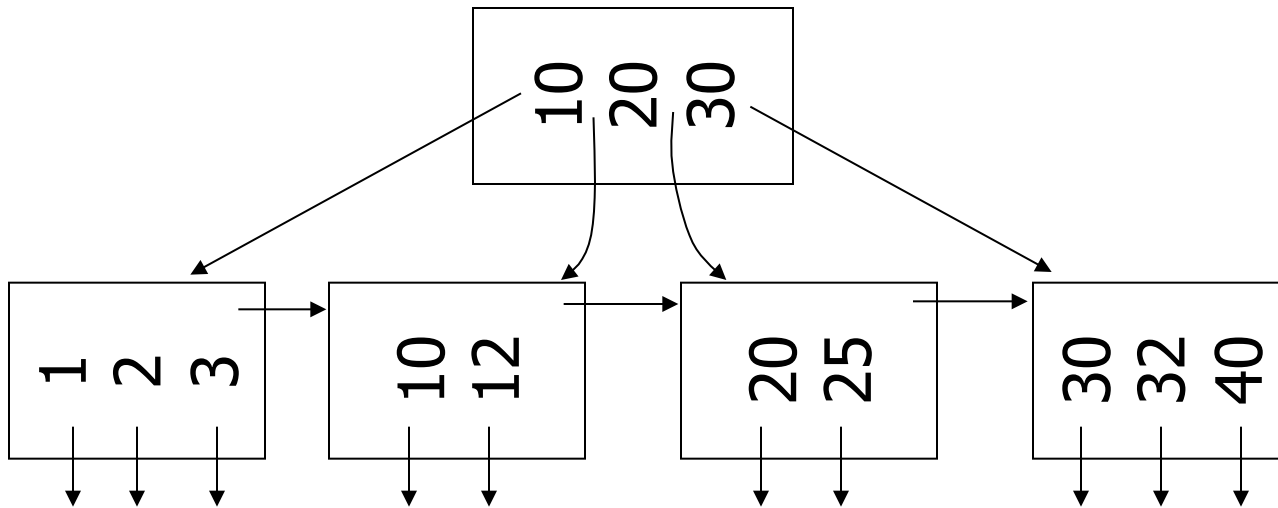
(d) New root, insert 45

n=3

(d) New root, insert 45

n=3

new root

30

10 20 30

40

1 2 3

10 12

20 25

30 32 40

40 45

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
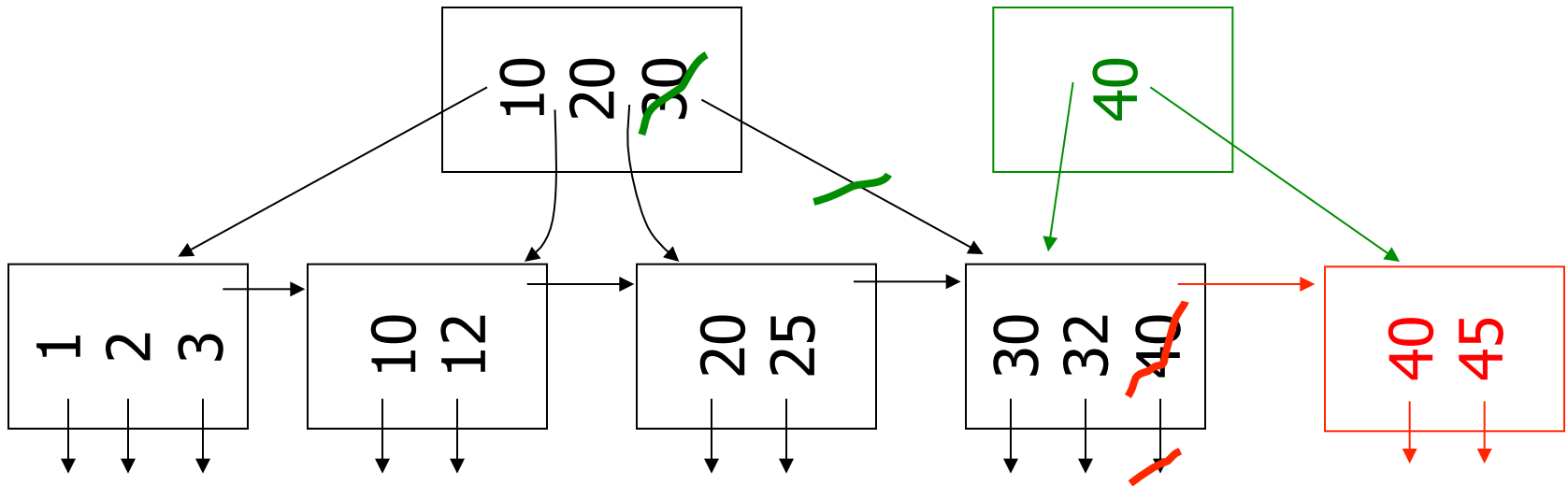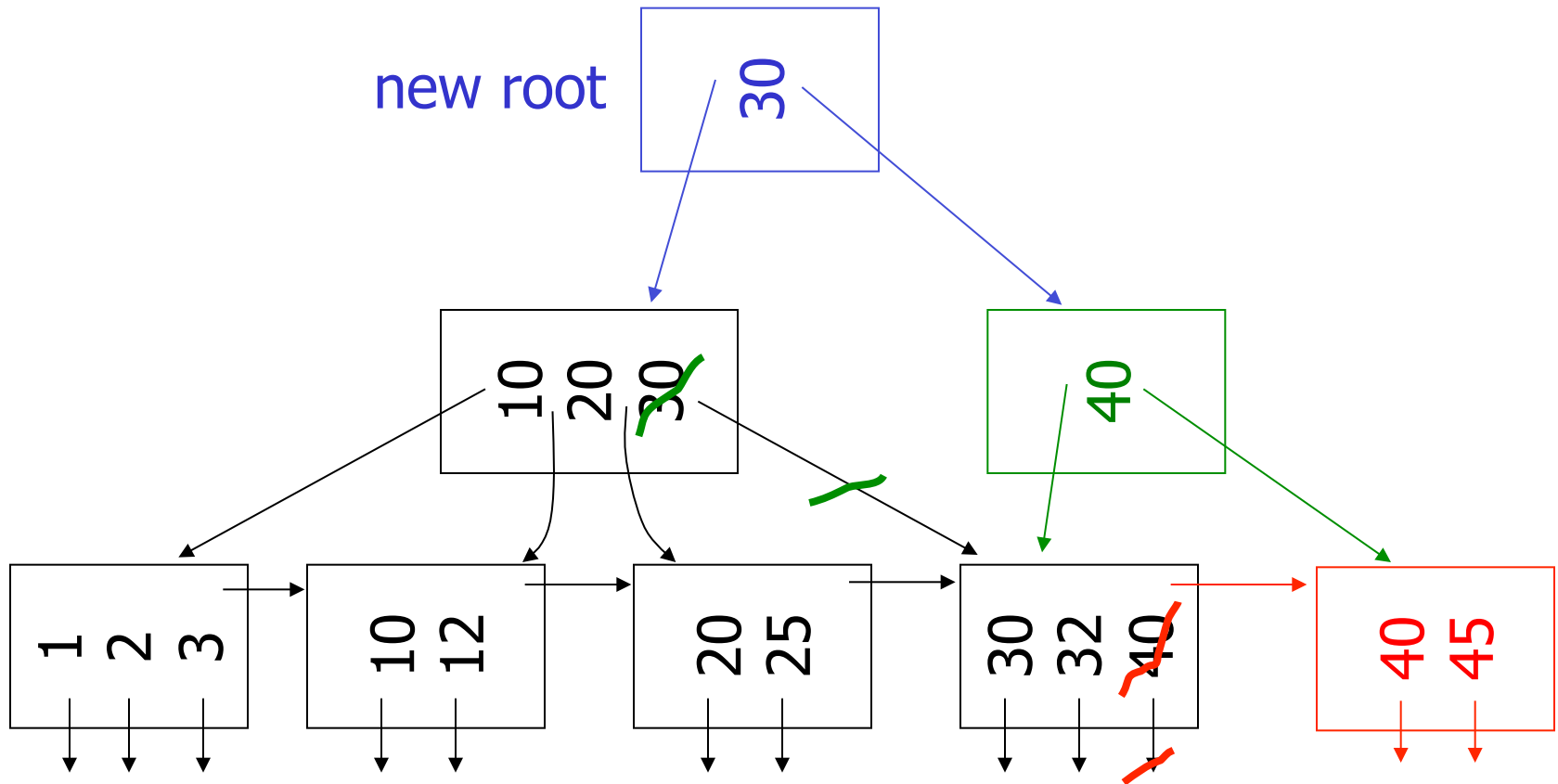
Insertion Algorithm

- Insert Record with key **k**

- Search leaf node for **k**

  - Leaf node has at least one space

    - Insert into leaf

  - Leaf is full

    - Split leaf into two nodes (new leaf)

    - Insert new leaf's smallest key into parent

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Insertion Algorithm cont.

- Non-leaf node is full
  - Split parent
  - Insert median key into parent
- Root is full
  - Split root
  - Create new root with two pointers and single key

- -> B-trees grow at the root

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from B+tree

(a) Simple case - no example
(b) Coalesce with neighbor (sibling)
(c) Re-distribute keys
(d) Cases (b) or (c) at non-leaf

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(b) Coalesce with sibling
– Delete 50

n=4

Notes 4 - Indexing

(b) Coalesce with sibling
– Delete 50

n=4

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
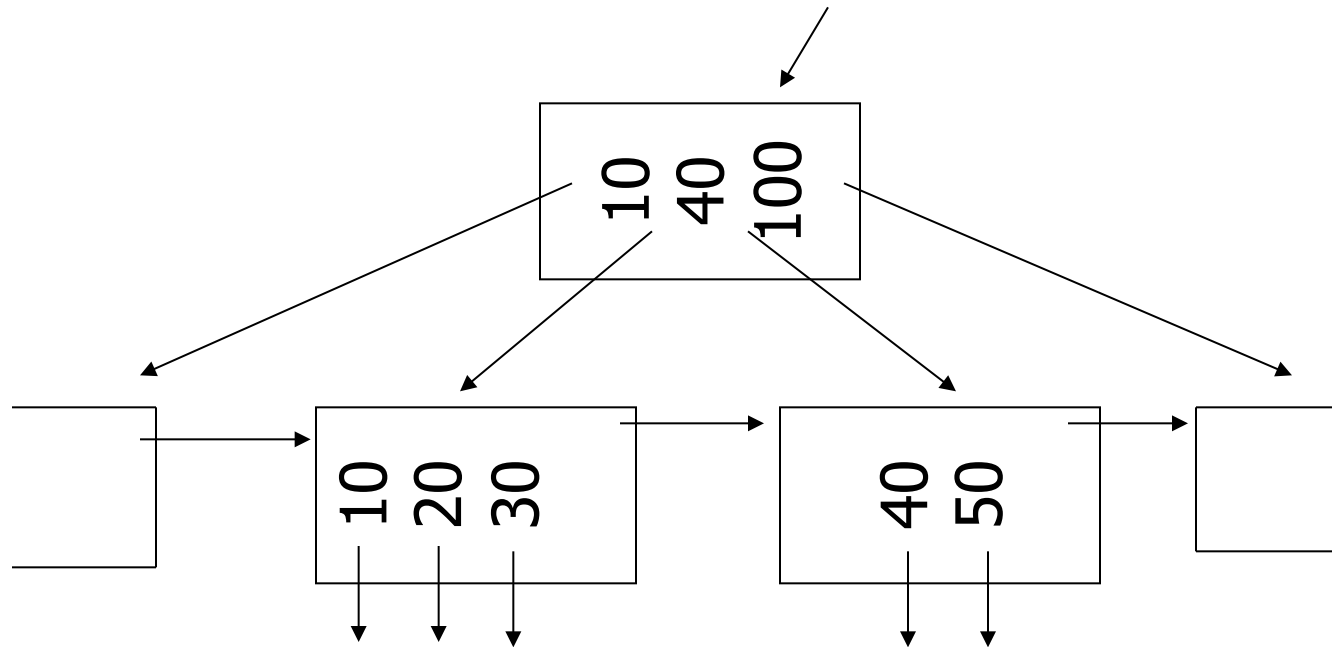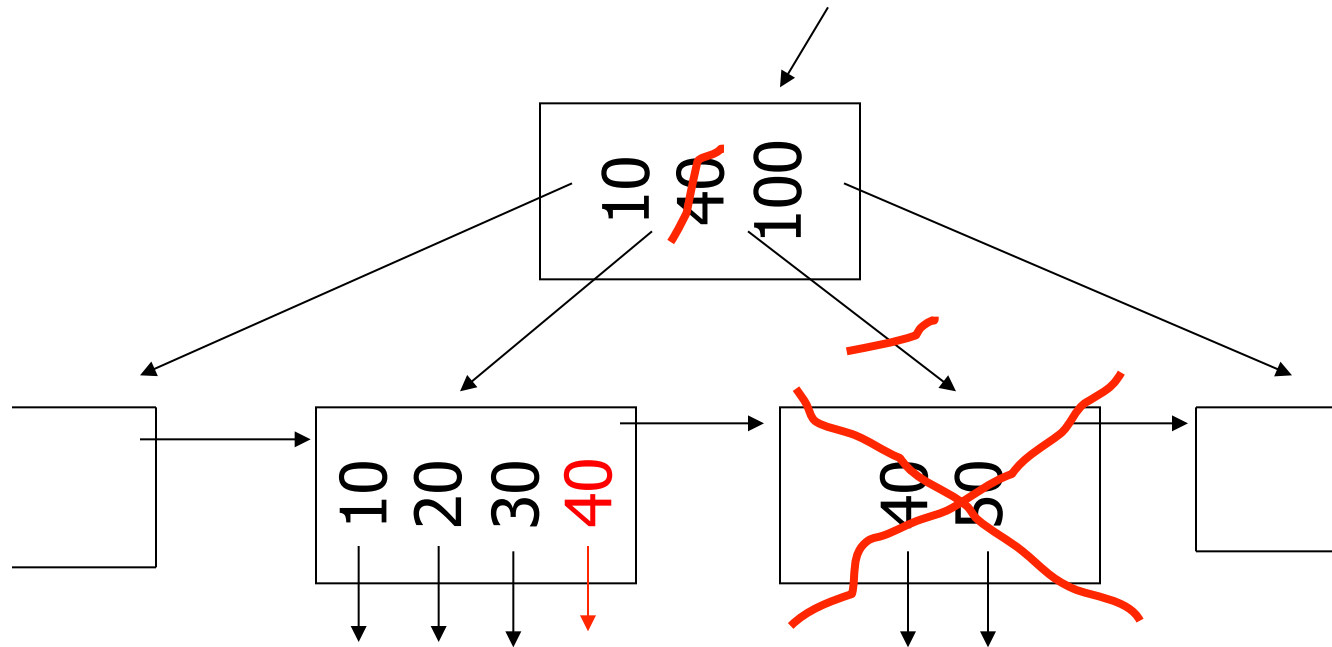
(c) Redistribute keys
  – Delete 50

n=4
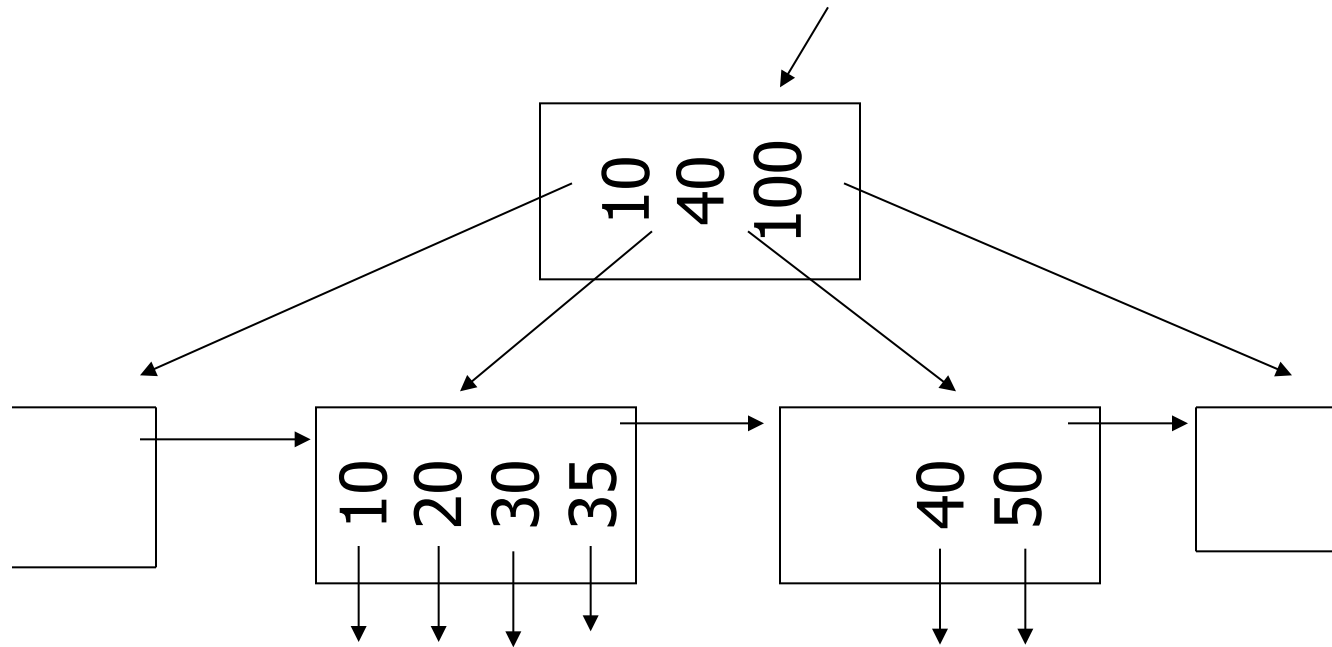
(c) Redistribute keys
 – Delete 50

n=4

(d) Non-leaf coalese
– Delete 37

n=4

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(d) Non-leaf coalese
 – Delete 37

n=4

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
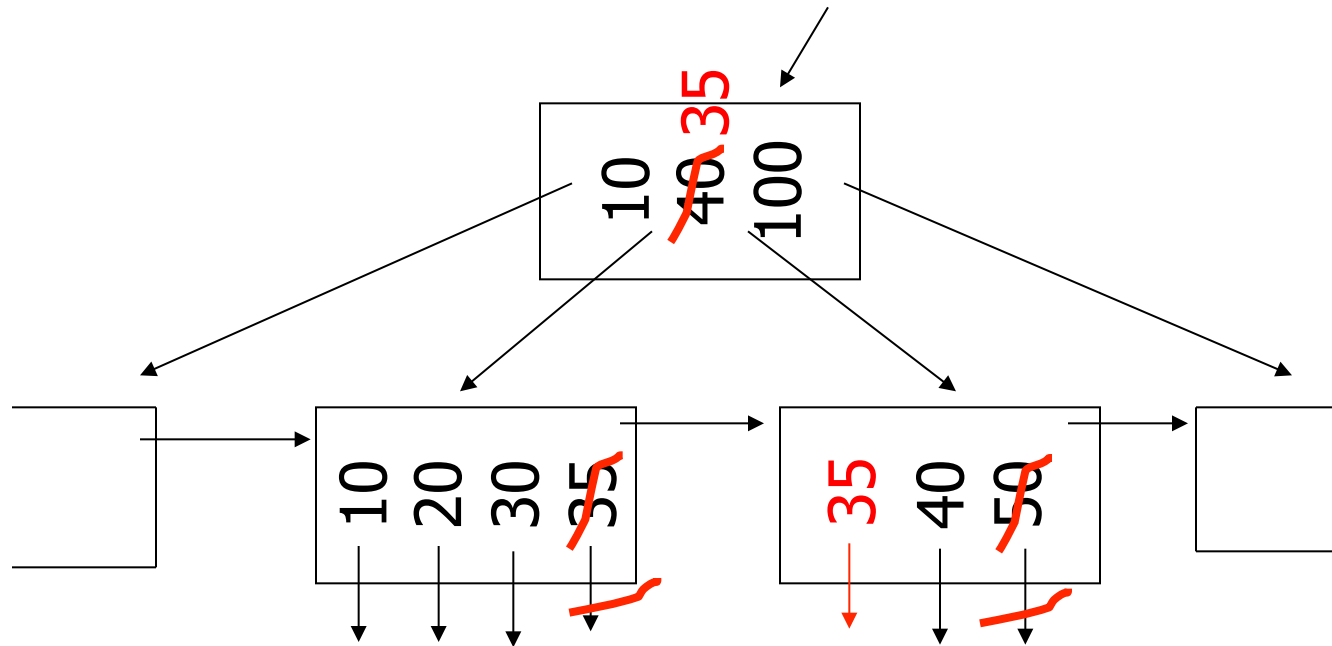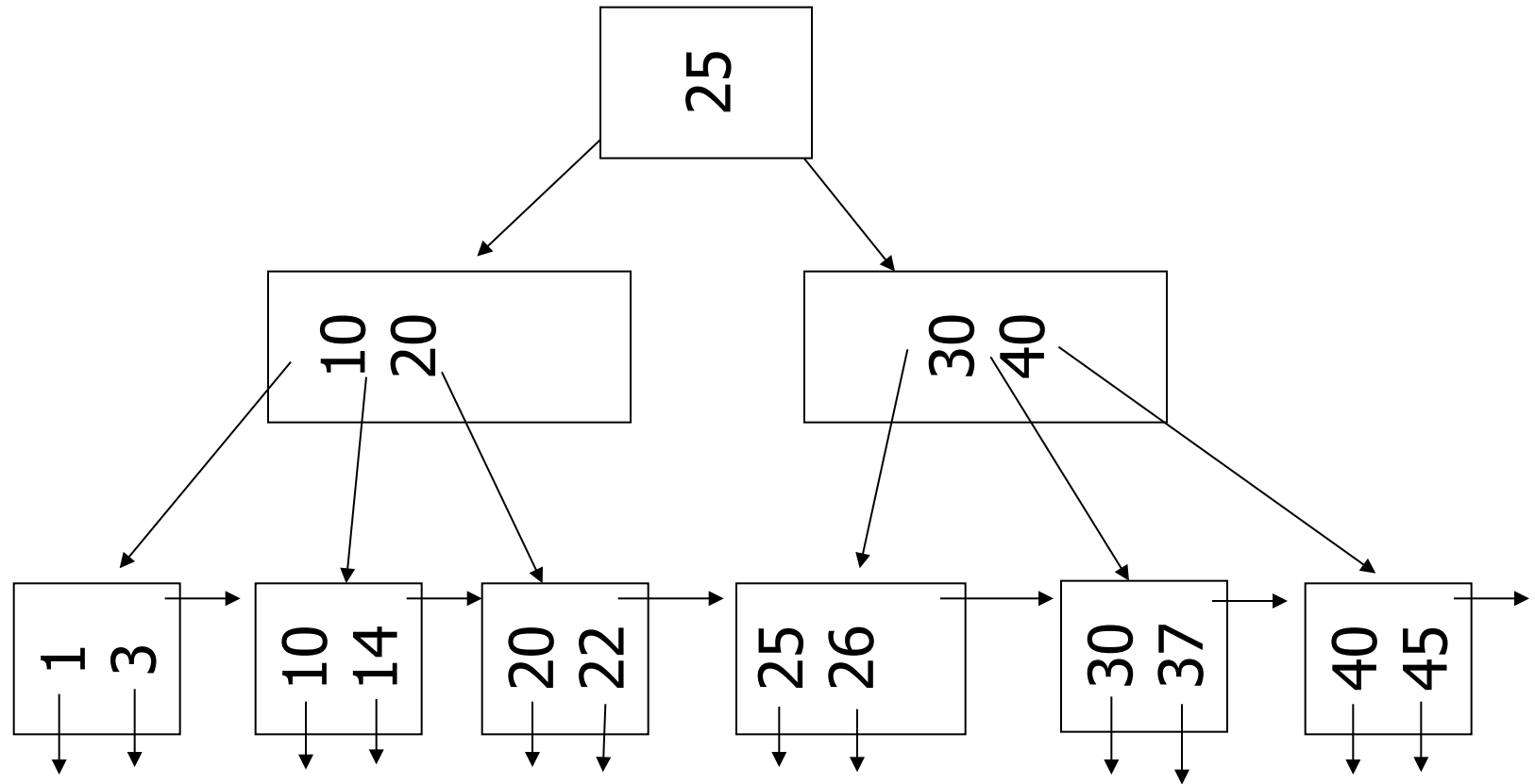
(d) Non-leaf coalese
– Delete 37

n=4

(d) Non-leaf coalese
– Delete 37

n=4

new root

10 20 *25* *40*

25 26 ~~30~~

1 3

10 14

20 22

25 26 ~~30~~

30 ~~37~~

40 45

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
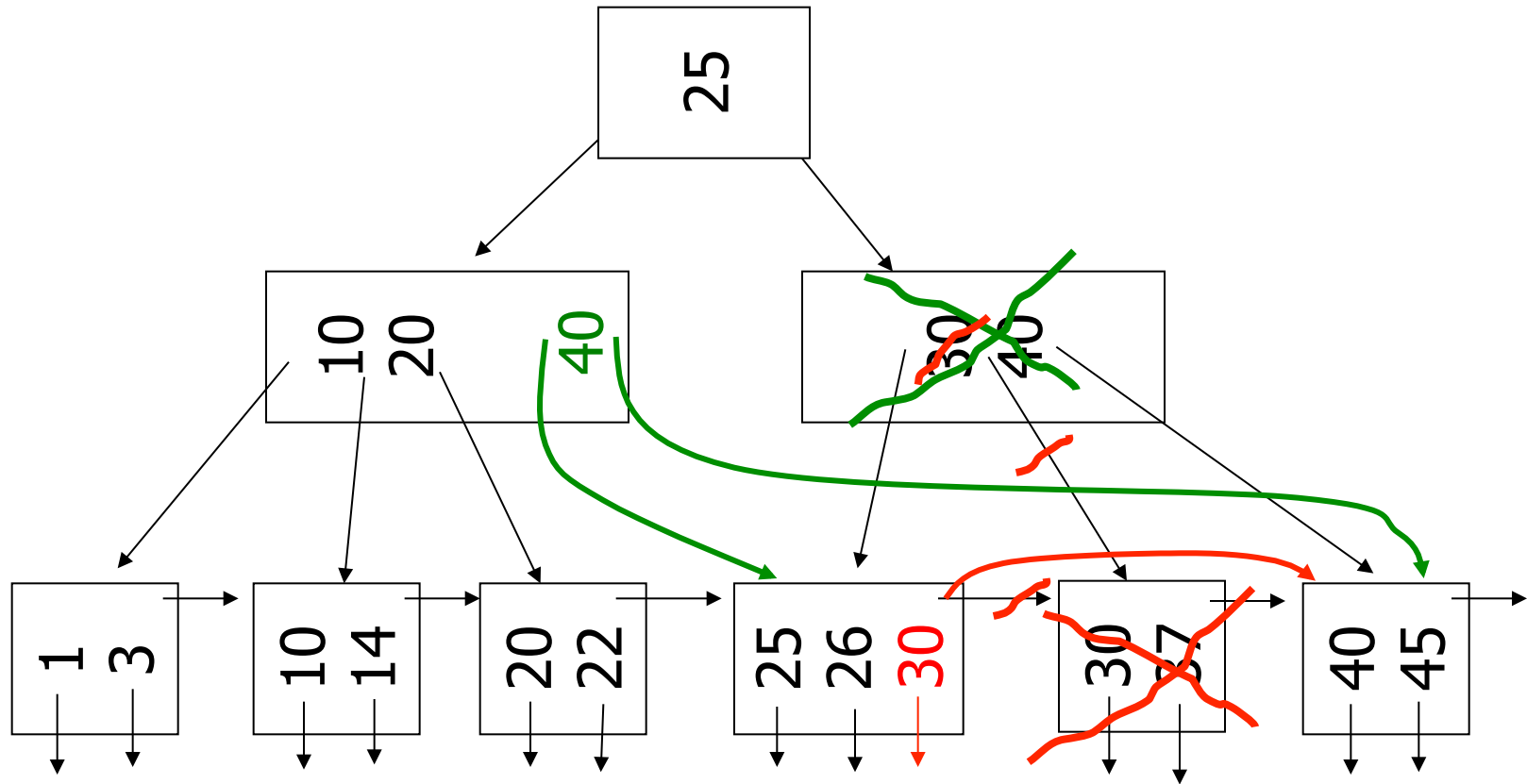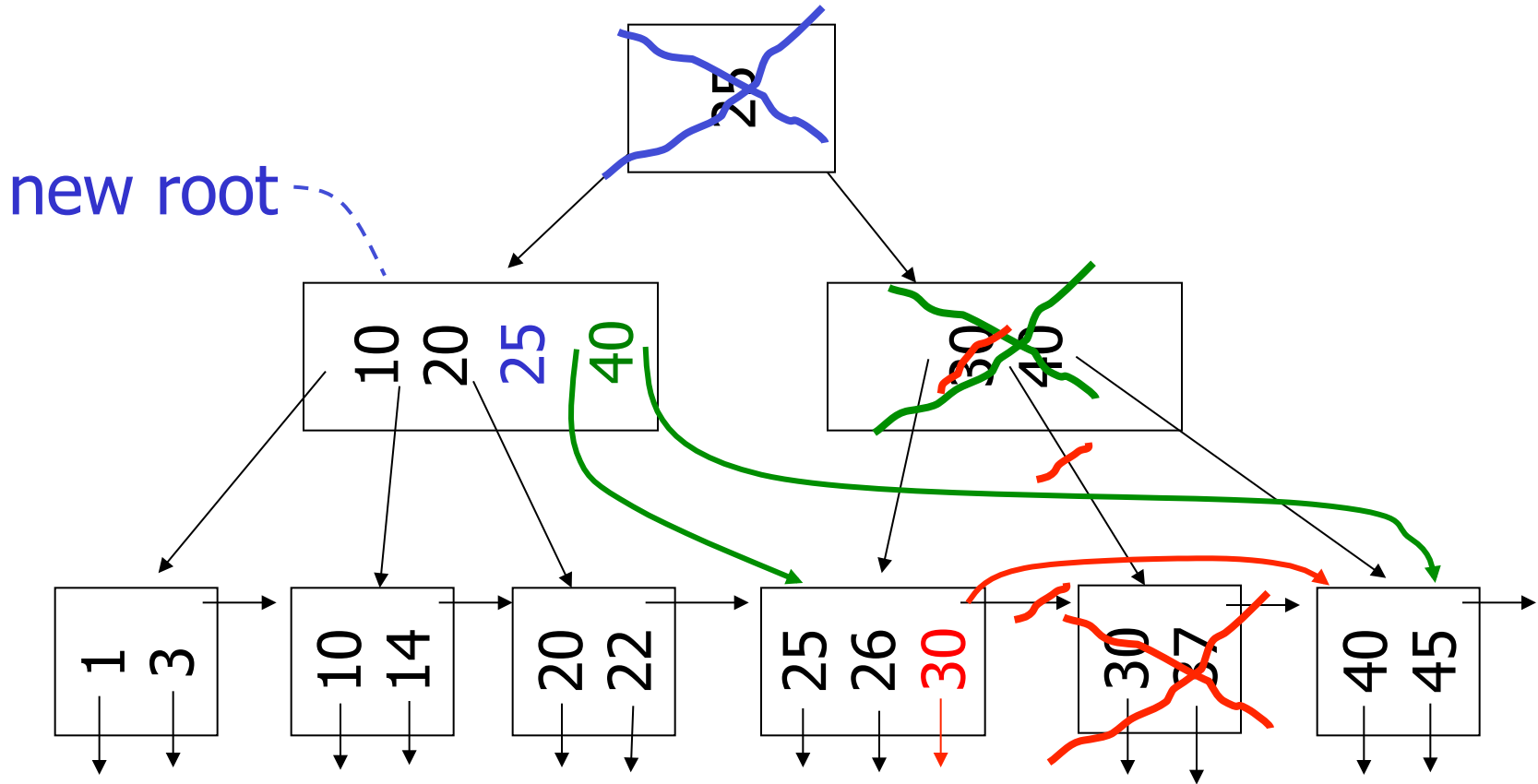
Deletion Algorithm

- Delete record with key **k**
- Search leaf node for **k**
  - Leaf has more than min entries
    - Remove from leaf
  - Leaf has min entries
    - Try to borrow from sibling
  - One direct sibling has more min entries
    - Move entry from sibling and adapt key in parent

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Deletion Algorithm cont.

- Both direct siblings have min entries
  - Merge with one sibling
  - Remove node or sibling from parent
  - ->recursive deletion
- Root has two children that get merged
  - Merged node becomes new root

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

B+tree deletions in practice

– Often, coalescing is <u>not</u> implemented
  – Too hard and not worth it!
  – Assumption: nodes will fill up in time again

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Comparison: B-trees vs. static indexed sequential file

Ref #1:   Held & Stonebraker

"B-Trees Re-examined"

CACM, Feb. 1978

Ref # 1 claims:

 - Concurrency control harder in B-Trees

  - B-tree consumes more space

For their comparison:

block = 512 bytes
key = pointer = 4 bytes
4 data records per block

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Example: 1 block static index

127 keys

k1
k2
k3

k1

k2

k3

1 data block

up to 127 blocks
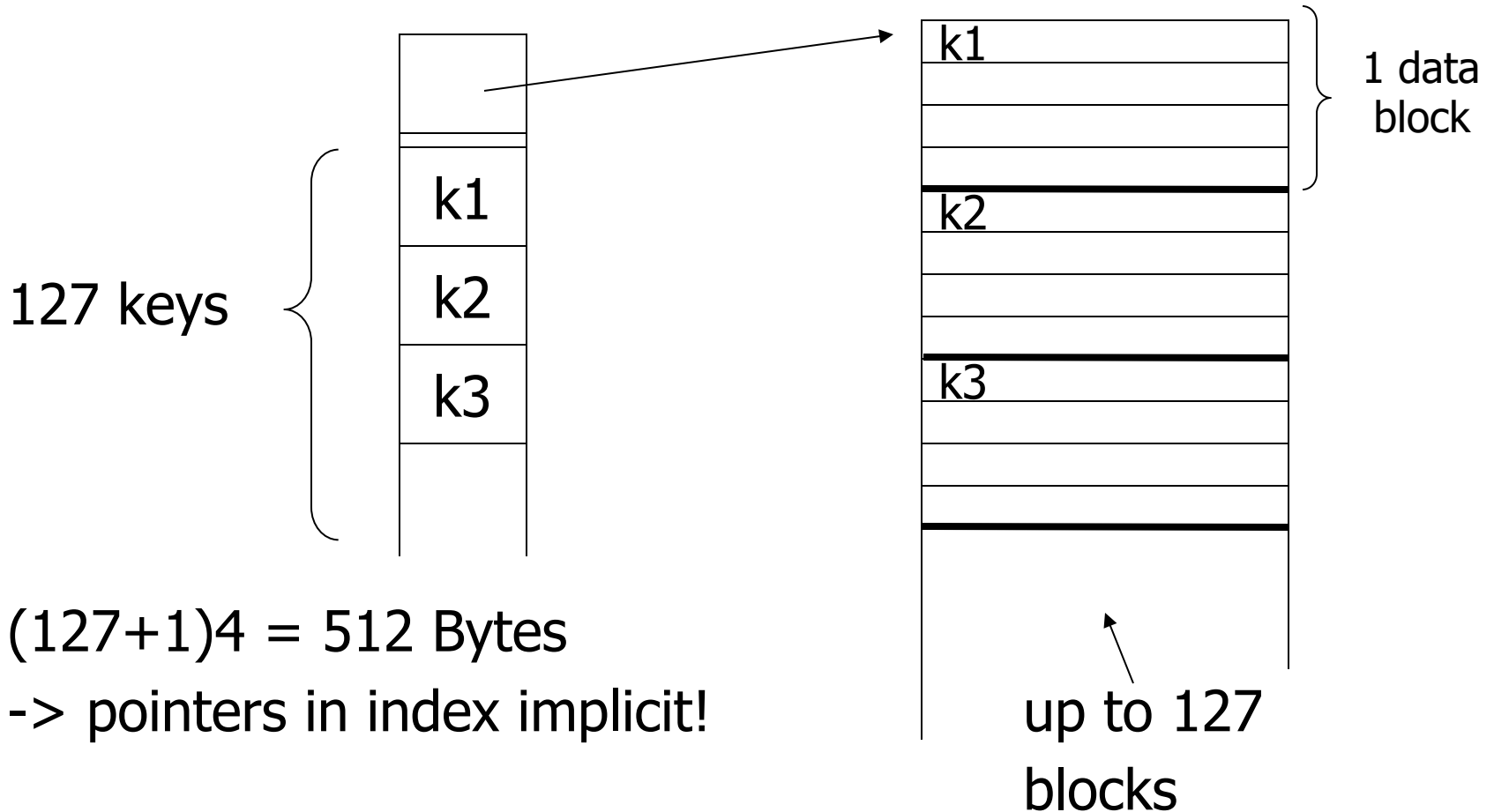
(127+1)4 = 512 Bytes
-> pointers in index implicit!

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example: 1 block B-tree

63 keys

k1

k2

...

k63

-

next

1 data block

k1

k2

k3

up to 63 blocks

63x(4+4)+8 = 512 Bytes

-> pointers needed in B-tree blocks because index is not contiguous

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
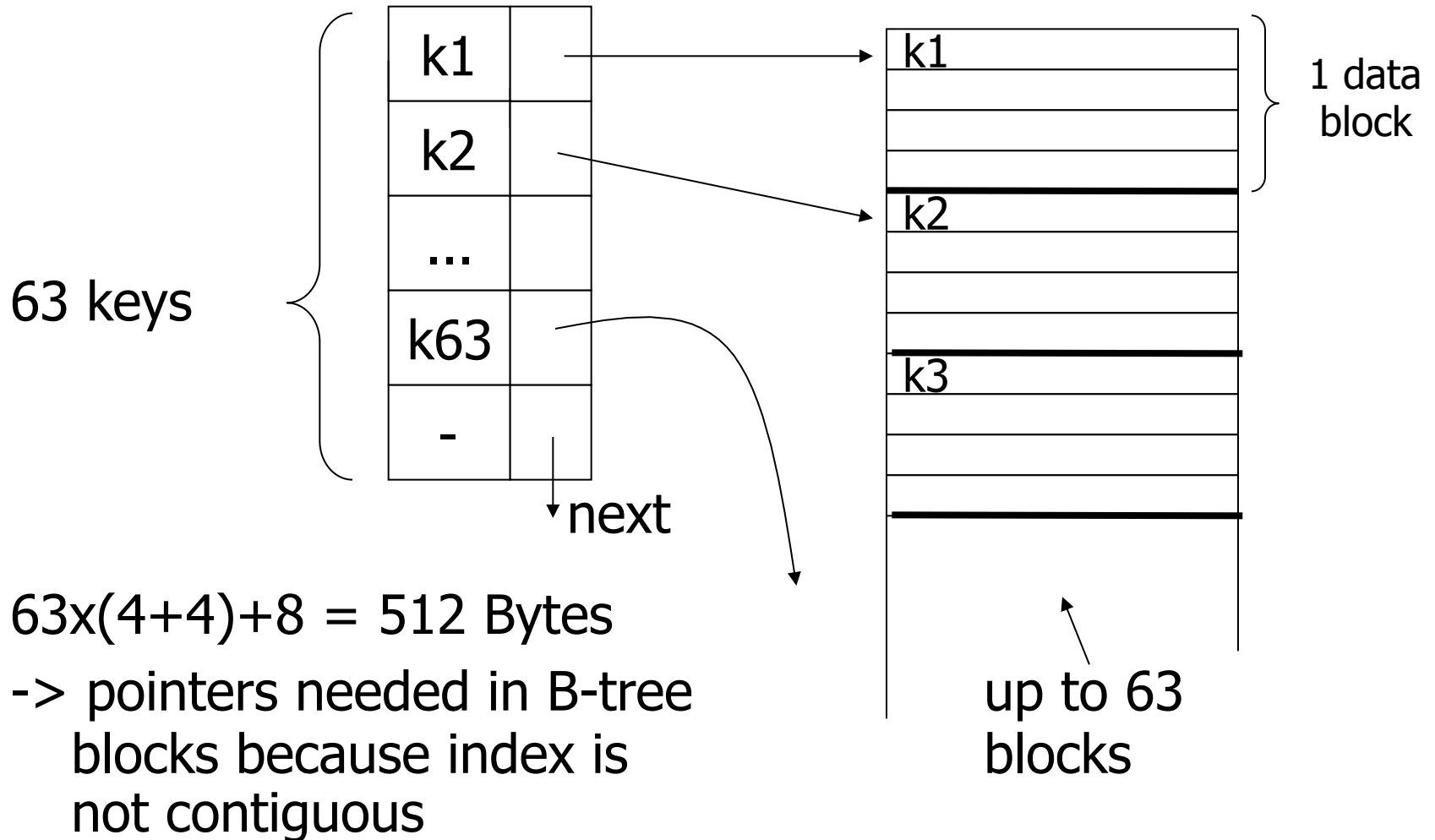
Size comparison   Ref. #1

<u>Static Index</u>		B-tree	
# data blocks	height	# data blocks	height
2 -> 127	2	2 -> 63	2
128 -> 16,129	3	64 -> 3968	3
16,130 -> 2,048,383	4	3969 -> 250,047	4
		250,048 -> 15,752,961	5

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Ref. #1 analysis claims

- For an 8,000 block file,

  $\Big\{$ after 32,000 inserts

  after 16,000 lookups

  $\Rightarrow$ Static index saves enough accesses
  to allow for reorganization

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Ref. #1 analysis claims

- For an 8,000 block file,
    - after 32,000 inserts
    - after 16,000 lookups
  $\Rightarrow$ Static index saves enough accesses
      to allow for reorganization

Ref. #1 conclusion $\Longrightarrow$ Static index better!!

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Ref #2:   M. Stonebraker,

"Retrospective on a database system,"   TODS, June 1980

Ref. #2 conclusion ➔   B-trees better!!

Ref. #2 conclusion ⟹ B-trees better!!

- DBA does not know <u>when</u> to reorganize
- DBA does not know <u>how full</u> to load pages of new index

Ref. #2 conclusion ⟶ B-trees better!!

- Buffering
  - B-tree: has fixed buffer requirements
  - Static index: must read several overflow
    blocks to be efficient
    (large & variable
    size   buffers
    needed for this)

COMPUTER
SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- Speaking of buffering...
    Is LRU a good policy for B+tree buffers?

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

- Speaking of buffering...

    Is LRU a good policy for B+tree buffers?

→ Of course not!

→ Should try to keep root in memory
  at all times

    (and perhaps some nodes from second level)

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Interesting problem:

For B+tree, how large should *n* be?

n is number of keys / node

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
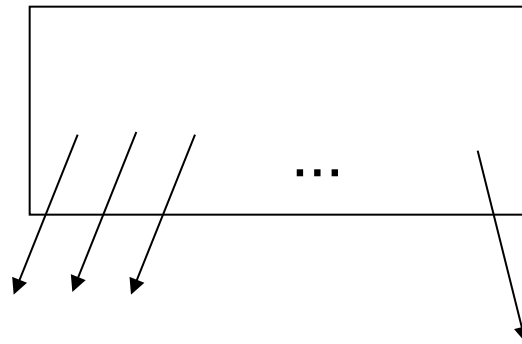
Sample assumptions:

(1) Time to read node from disk is
$(S+Tn)$ msec.

Sample assumptions:

(1) Time to read node from disk is
$(S+Tn)$ msec.

(2) Once block in memory, use binary search to locate key:
$(a + b\ LOG_2\ n)$ msec.

For some constants $a,b;$ Assume a << S

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Sample assumptions:

(1) Time to read node from disk is
$(S+Tn)$ msec.

(2) Once block in memory, use binary search to locate key:
$(a + b \text{ LOG}_2 \, n)$ msec.

For some constants a, b; Assume a << S

(3) Assume B+tree is full, i.e.,
\# nodes to examine is $\text{LOG}_n \, N$
where $N = $ \# records

➻Can get:
f(n) = time to find a record

$f(n)$

n_{opt}

n

Notes 4 - Indexing   133

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
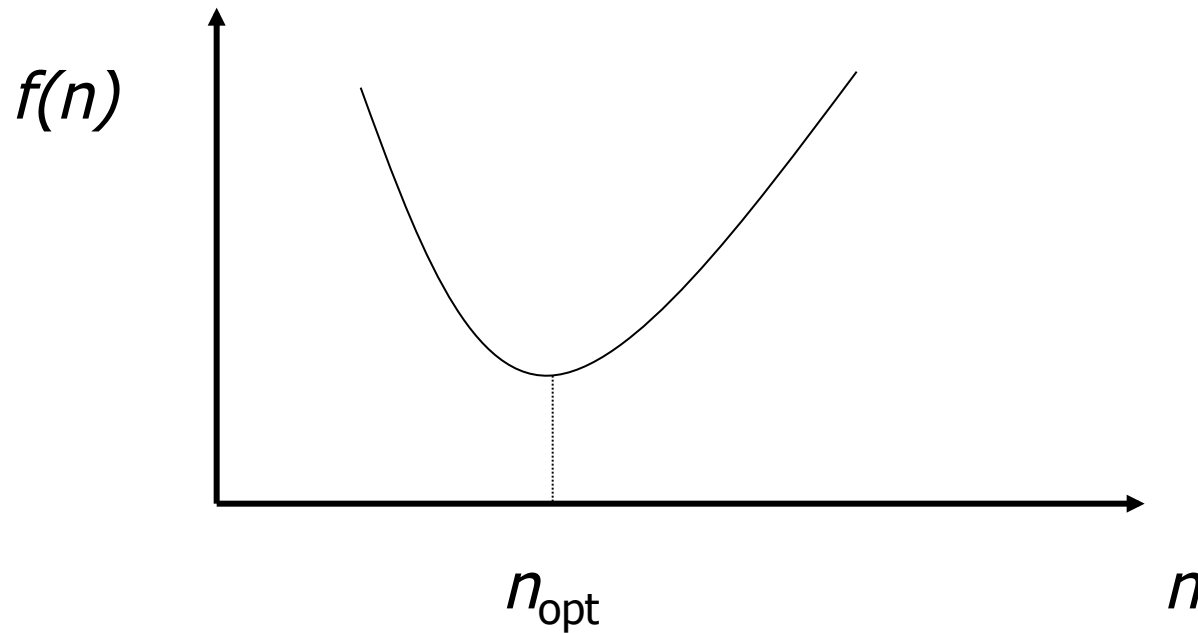
➤➤ FIND n_{opt} by $f'(n) = 0$

Answer is n_{opt} = "few hundred"

➤ FIND n_{opt} by $f'(n) = 0$

Answer is n_{opt} = "few hundred"

➤ What happens to n_{opt} as

- Disk gets faster?
- CPU get faster?
- Memory hierarchy?

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Variation on B+tree: B-tree (no +)

- Idea:
  - Avoid duplicate keys
  - Have record pointers in non-leaf nodes

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

| | K1 | P1 | | K2 | P2 | | K3 | P3 | |

to record
with K1

to record
with K2

to record
with K3

to keys
< K1

to keys
K1<x<K2

to keys
K2<x<k3

to keys
>k3
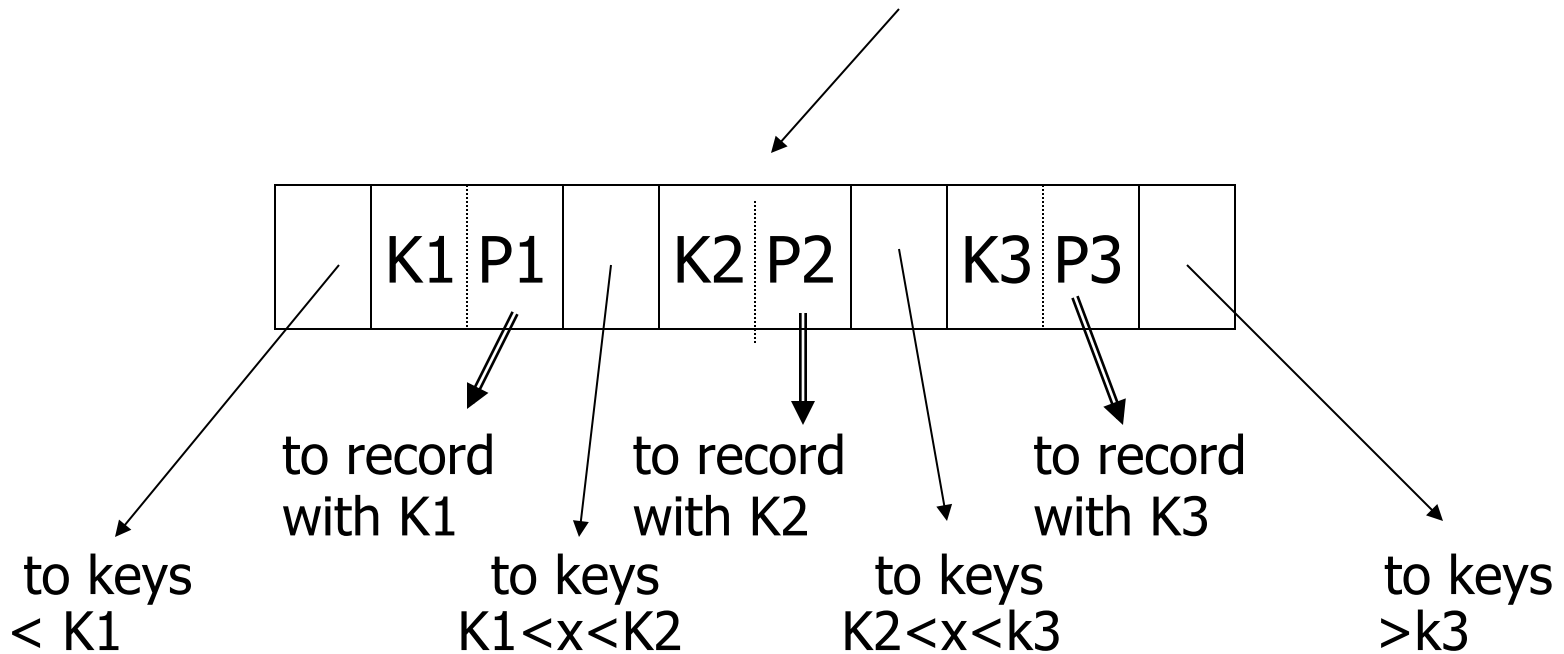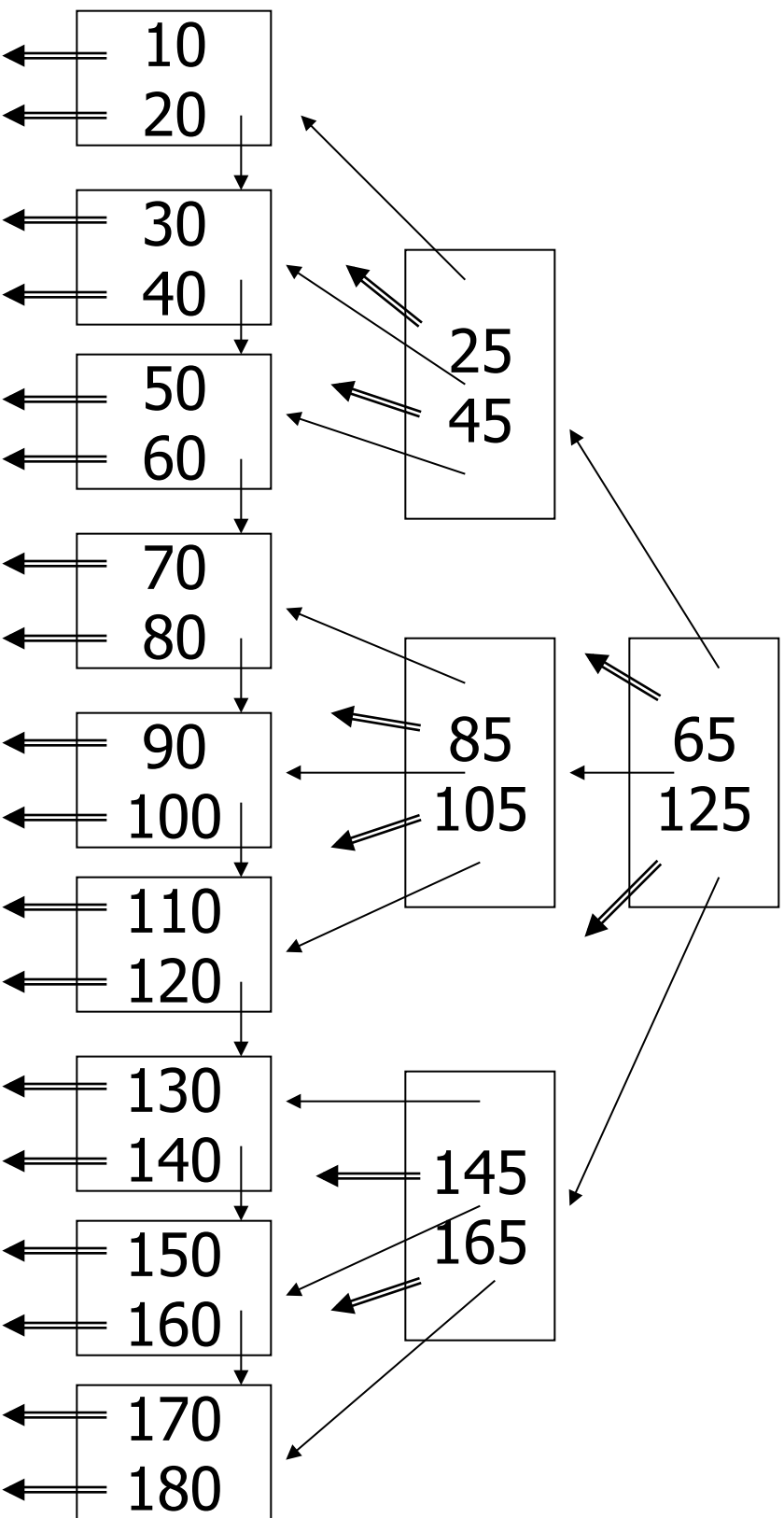
B-tree example   n=2

B-tree example   n=2

- **sequence pointers not useful now!**
  (but keep space for simplicity)
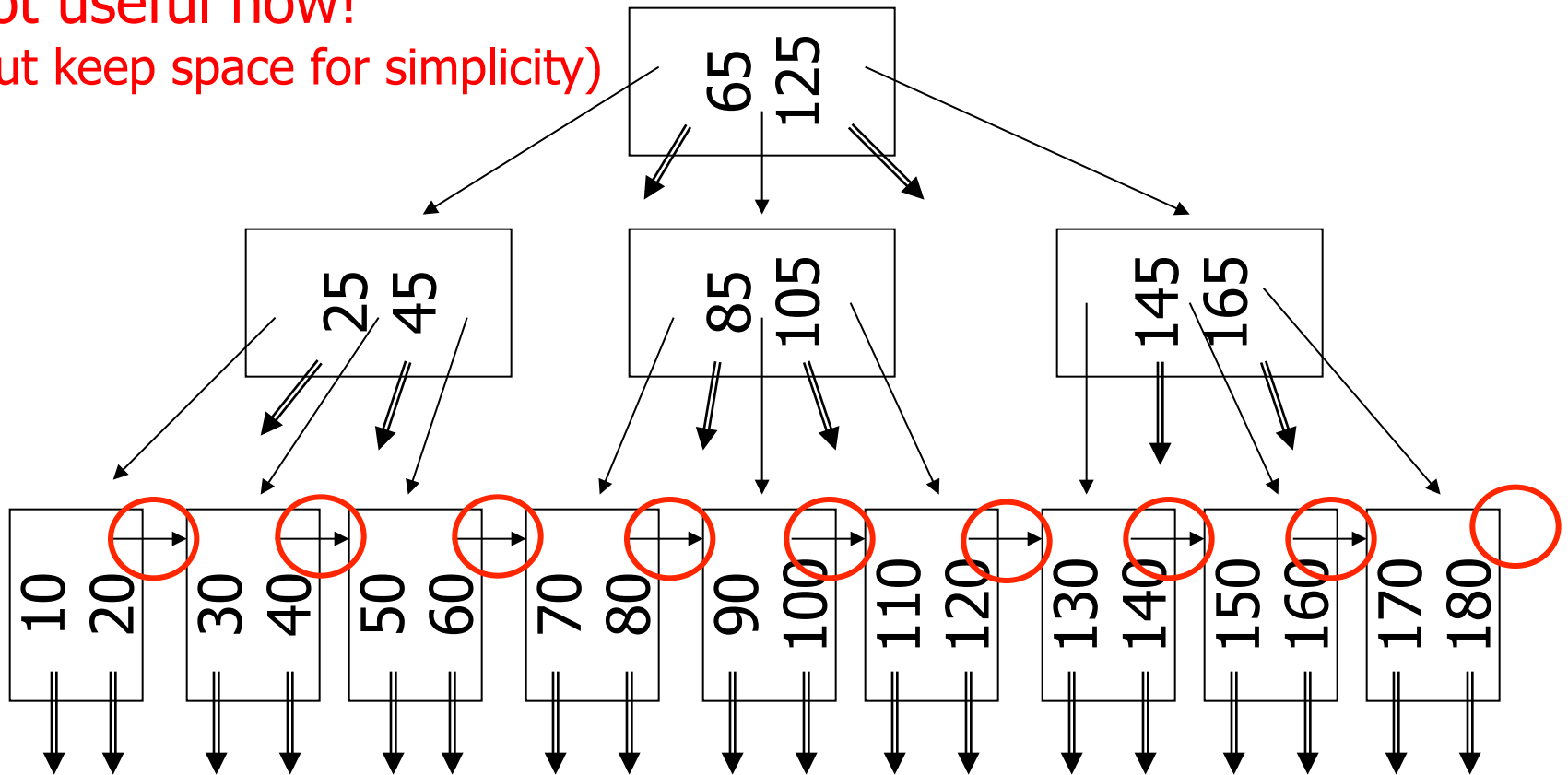
Note on inserts

• Say we insert record with key = 25

leaf [10 20 30 →]   n=3

Note on inserts

- Say we insert record with key = 25

leaf | 10 20 30 | →   n=3

- Afterwards:

| – 20 – |

| 10 | → | 25 30 | →

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
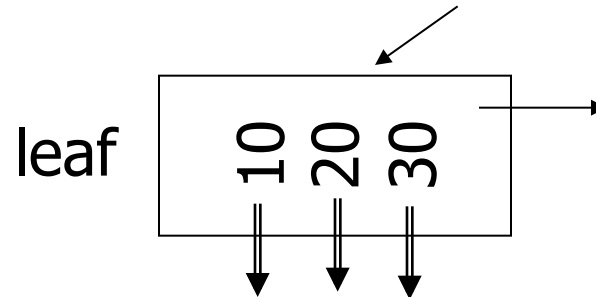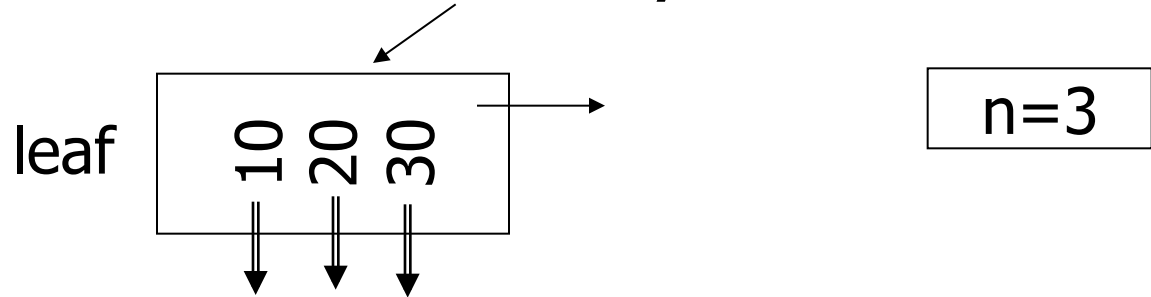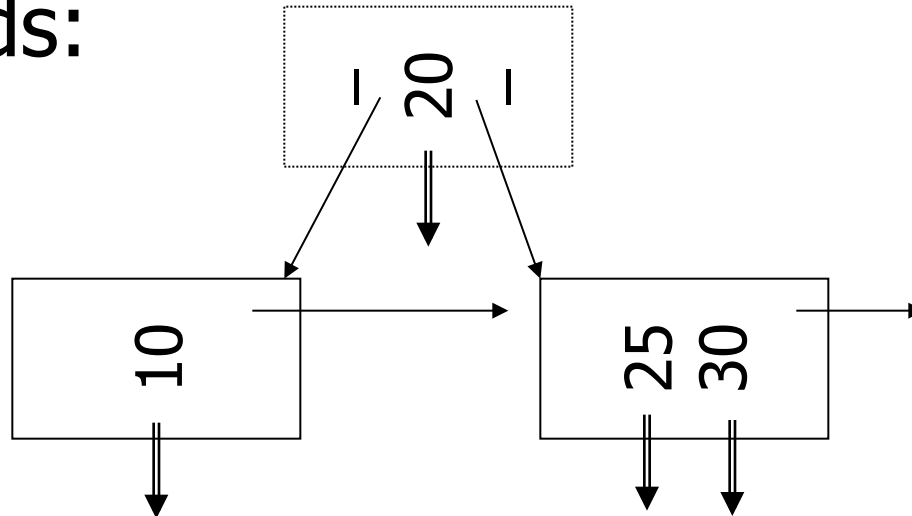
So, for B-trees:

	MAX			MIN		
	Tree Ptrs	Rec Ptrs	Keys	Tree Ptrs	Rec Ptrs	Keys
Non-leaf non-root	n+1	n	n	$\lceil(n+1)/2\rceil$	$\lceil(n+1)/2\rceil$-1	$\lceil(n+1)/2\rceil$-1
Leaf non-root	1	n	n	1	$\lfloor n/2 \rfloor$	$\lfloor n/2 \rfloor$
Root non-leaf	n+1	n	n	2	1	1
Root Leaf	1	n	n	1	1	1

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Tradeoffs:

☺ B-trees have faster lookup than B+trees

☹ in B-tree, non-leaf & leaf different sizes

☹ in B-tree, deletion more complicated

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Tradeoffs:

☺ B-trees have faster lookup than B+trees

☹ in B-tree, non-leaf & leaf different sizes

☹ in B-tree, deletion more complicated

➡ B+trees preferred!

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

But note:

- If blocks are fixed size

   (due to disk and buffering restrictions)

   Then lookup for B+tree is
   <u>actually better!!</u>

Example:

 - Pointers   4 bytes
 - Keys   4 bytes
 - Blocks   100 bytes (just example)
 - Look at full 2 level tree

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

B-tree:

Root has 8 keys + 8 record pointers
                                        + 9 son pointers

       = 8x4 + 8x4 + 9x4 = 100 bytes

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

B-tree:

Root has 8 keys + 8 record pointers
                                + 9 son pointers

= 8x4 + 8x4 + 9x4 = 100 bytes

Each of 9 sons: 12 rec. pointers (+12 keys)

= 12x(4+4) + 4 = 100 bytes

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

B-tree:

Root has 8 keys + 8 record pointers
+ 9 son pointers

$$= 8\text{x}4 + 8\text{x}4 + 9\text{x}4 = 100 \text{ bytes}$$

Each of 9 sons: 12 rec. pointers (+12 keys)

$$= 12\text{x}(4+4) + 4 = 100 \text{ bytes}$$

2-level B-tree, Max # records =

$$12\text{x}9 + 8 = 116$$

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

B+tree:

Root has 12 keys + 13 son pointers

$$= 12x4 + 13x4 = 100 \text{ bytes}$$

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

B+tree:

Root has 12 keys + 13 son pointers
$$= 12 \times 4 + 13 \times 4 = 100 \text{ bytes}$$

Each of 13 sons: 12 rec. ptrs (+12 keys)
$$= 12 \times (4 + 4) + 4 = 100 \text{ bytes}$$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

B+tree:

Root has 12 keys + 13 son pointers
$$= 12 \times 4 + 13 \times 4 = 100 \text{ bytes}$$

Each of 13 sons: 12 rec. ptrs (+12 keys)
$$= 12 \times (4 + 4) + 4 = 100 \text{ bytes}$$

2-level B+tree, Max # records
$$= 13 \times 12 = 156$$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

So...

8 records

B+

OOOOOOOOOOOOOO

156 records

B

OOOOOOOOO

108 records

Total = 116

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

So...

8 records

B+

156 records

B

108 records

Total = 116
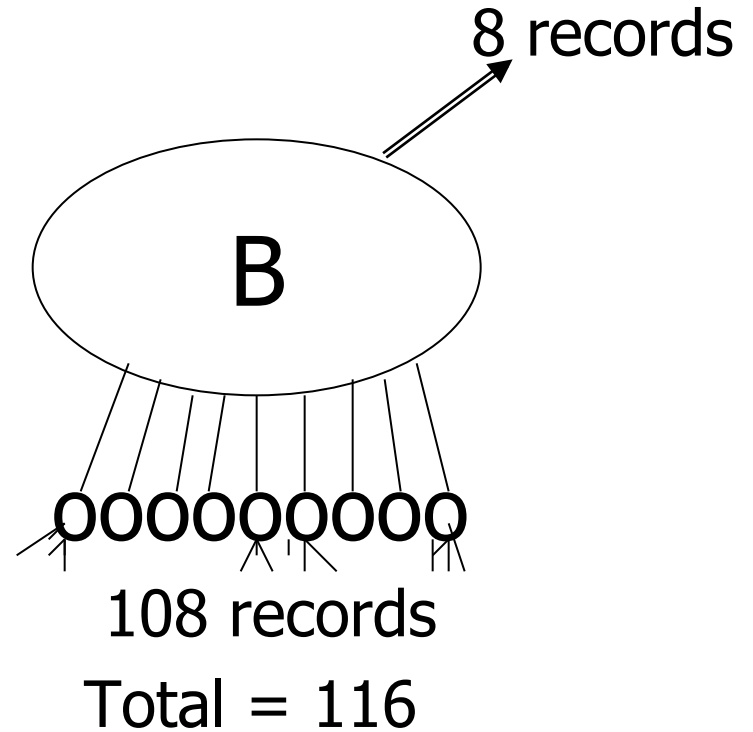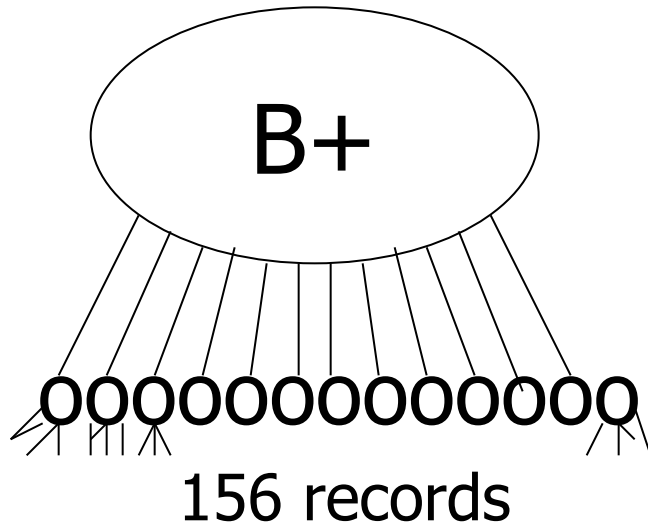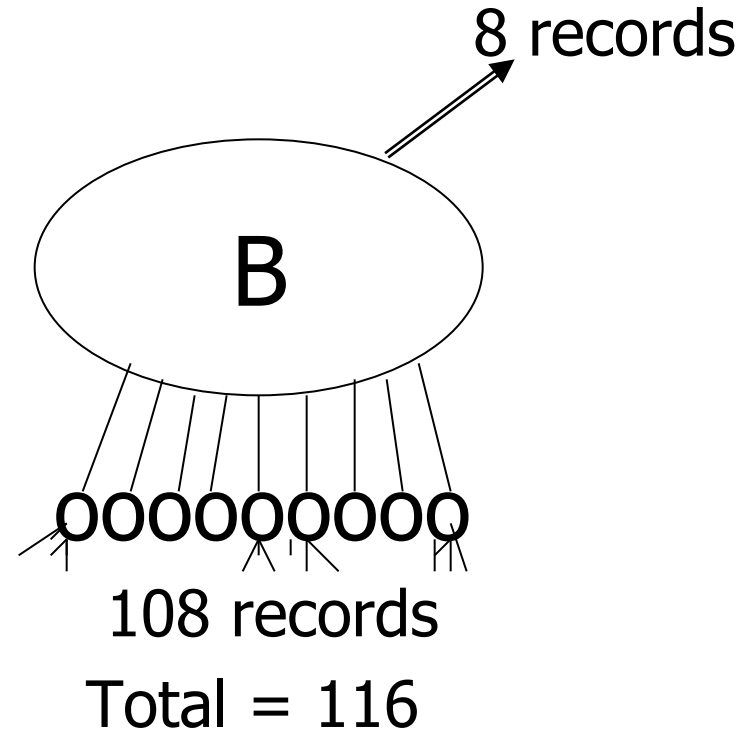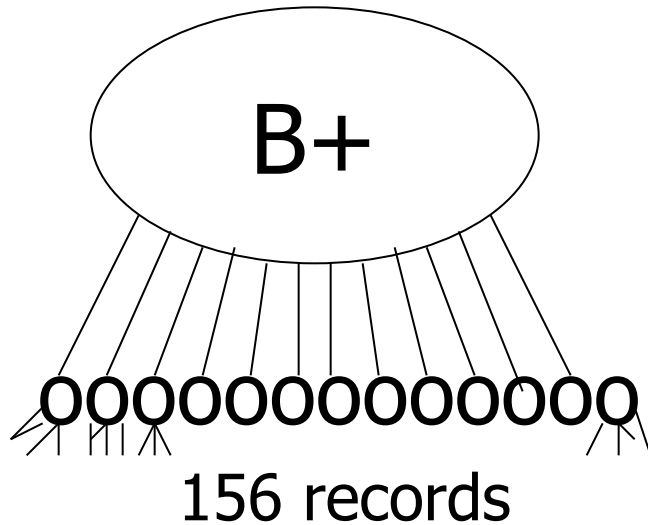
- Conclusion:
  - For fixed block size,
  - B+ tree is better because it is <u>bushier</u>

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Additional B-tree Variants

- B*-tree
  - Internal notes have to be 2/3 full

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

An Interesting Problem...

- What is a good index structure when:
  - records tend to be inserted with keys that are larger than existing values?
    (e.g., banking records with growing data/time)
  - we want to remove older data

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

One Solution: Multiple Indexes

- Example: I1, I2

day	days indexed I1	days indexed I2
10	1,2,3,4,5	6,7,8,9,10
11	11,2,3,4,5	6,7,8,9,10
12	11,12,3,4,5	6,7,8,9,10
13	11,12,13,4,5	6,7,8,9,10

- advantage: deletions/insertions from smaller index
- disadvantage: query multiple indexes

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Another Solution (Wave Indexes)

day	I1	I2	I3	I4
10	1,2,3	4,5,6	7,8,9	10
11	1,2,3	4,5,6	7,8,9	10,11
12	1,2,3	4,5,6	7,8,9	10,11, 12
13	13	4,5,6	7,8,9	10,11, 12
14	13,14	4,5,6	7,8,9	10,11, 12
15	13,14,15	4,5,6	7,8,9	10,11, 12
16	13,14,15	16	7,8,9	10,11, 12

- advantage: no deletions
- disadvantage: approximate windows

Concurrent Access To B-trees

- Multiple processes/threads accessing the B-tree
  - Can lead to corruption
- Serialize access to complete tree for updates
  - Simple
  - Unnecessary restrictive
  - Not feasible for high concurrency

Lock Nodes

	Read	Write
Read	X	-
Write	-	-

- One solution
  - **Read** and **exclusive** locks
  - Safe and unsafe updates of nodes
    - **Safe:** No ancestor of node will be effected by update
    - **Unsafe:** Ancestor may be affected
    - Can be determined locally
      - E.g., deletion is safe is node has more than n/2

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Lock Nodes

- Reading
  - Use standard search algorithm
  - Hold lock on current node
  - Release when navigating to child
- Writing
  - Lock each node on search for key
  - Release all locks on parents of node if the node is safe

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Improvements?

- Try locking only the leaf for update
  - Let update use read locks and only lock leaf node with write lock
  - If leaf node is unsafe then use previous protocol

- Many more locking approaches have been proposed

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Outline/summary

- Conventional Indexes
  - Sparse vs. dense
  - Primary vs. secondary

- B trees
  - B+trees vs. B-trees
  - B+trees vs. indexed sequential

- Hashing schemes   -->   Next

- Advanced Index Techniques

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

CS 525: Advanced Database Organization

05: Hashing and More

Boris Glavic

Slides: adapted from a course taught by Hector Garcia-Molina, Stanford InfoLab

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Hashing
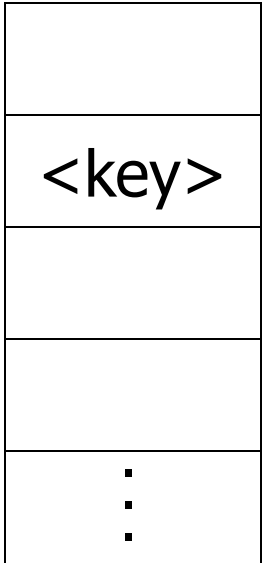
key → h(key) ·······→ | <key> | ← Buckets (typically 1 disk block)

Two alternatives

$$(1) \text{ key} \rightarrow h(\text{key})$$

records

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Two alternatives

$$(2)\ key \rightarrow h(key)$$

Index

key 1

record

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Two alternatives

$$(2)\ key \to h(key)$$

Index
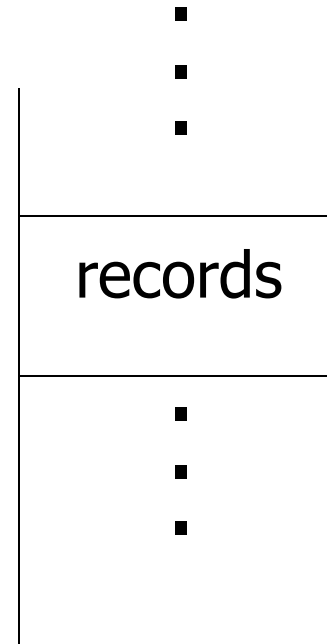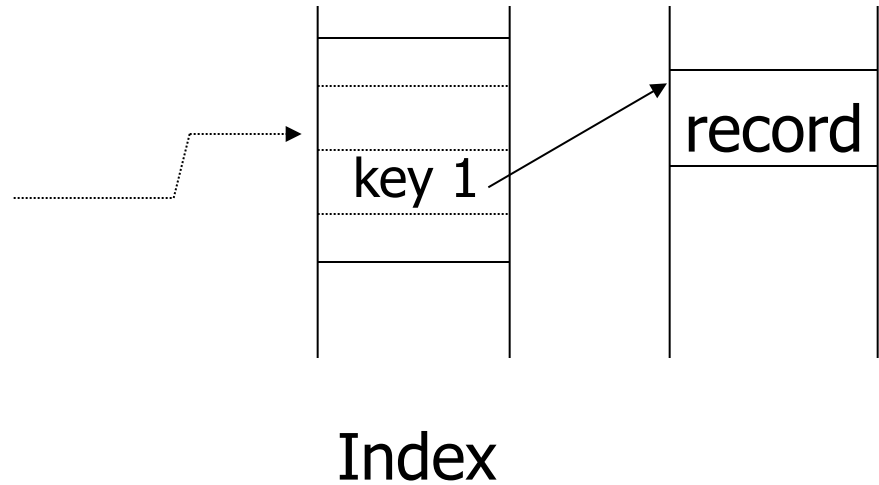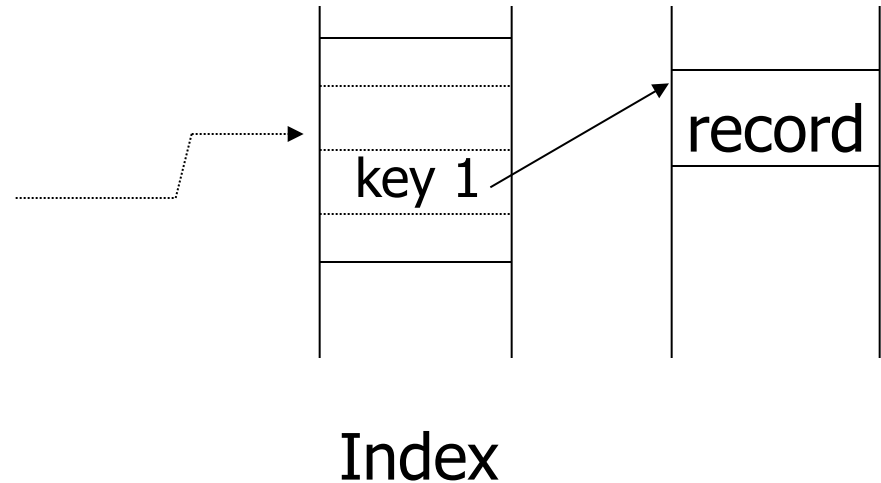
• Alt (2) for "secondary" search key

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example hash function

- Key = '$x_1 \ x_2 \ \dots \ x_n$' n byte character string

- Have b buckets

- h: add $x_1 + x_2 + \ \dots \ x_n$
    - compute sum modulo b

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

➤ This may not be best function …

➤ Read Knuth Vol. 3 if you really
    need to select a good function.

➦ This may not be best function …

➦ Read Knuth Vol. 3 if you really
  need to select a good function.

Good hash
function:

☞ Expected number of
  keys/bucket is the
  same for all buckets

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Within a bucket:

- Do we keep keys sorted?

- Yes, if CPU time critical
    & Inserts/Deletes not too frequent

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Next: example to illustrate inserts, overflows, deletes
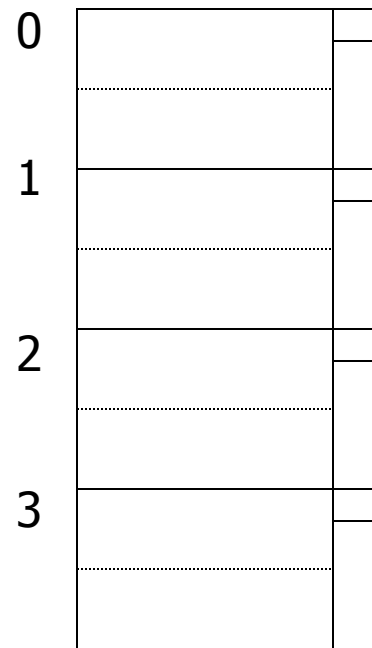
h(K)

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

EXAMPLE 2 records/bucket

INSERT:

h(a) = 1

h(b) = 2

h(c) = 1

h(d) = 0

0

1

2

3

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

EXAMPLE 2 records/bucket

INSERT:

h(a) = 1

h(b) = 2

h(c) = 1

h(d) = 0

h(e) = 1

```
   ┌─────────────┬──┐
0  │      d      │  │
   │·············│  │
   │             │  │
   ├─────────────┼──┤
1  │      a      │  │
   │·············│  │
   │      c      │  │
   ├─────────────┼──┤
2  │      b      │  │
   │·············│  │
   │             │  │
   ├─────────────┼──┤
3  │             │  │
   │·············│  │
   │             │  │
   └─────────────┴──┘
```

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

EXAMPLE 2 records/bucket

INSERT:

h(a) = 1

h(b) = 2

h(c) = 1

h(d) = 0

h(e) = 1

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

EXAMPLE: deletion

Delete:
    e
    f

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

EXAMPLE: deletion

Delete:

e

f

c

maybe move "g" up

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

EXAMPLE: deletion

Delete:
e
f
c

0   a

1   b

    c   d

2   e

3   f

    g

maybe move "g" up

Rule of thumb:

- Try to keep space utilization between 50% and 80%

$$\text{Utilization} = \frac{\text{\# keys used}}{\text{total \# keys that fit}}$$

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
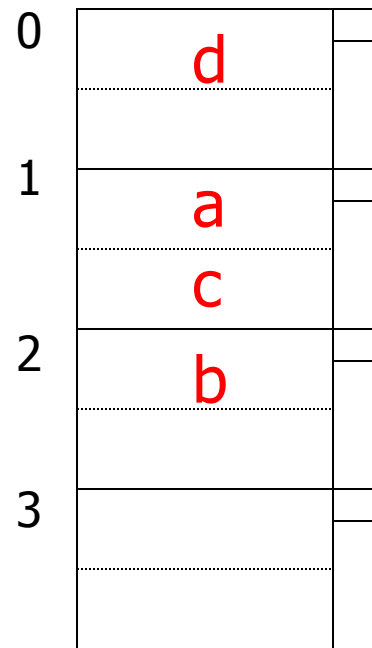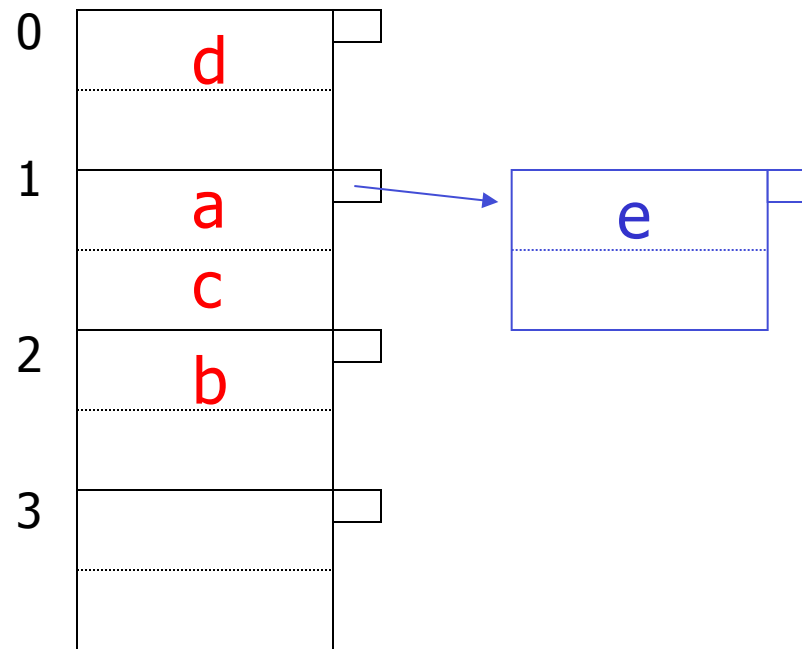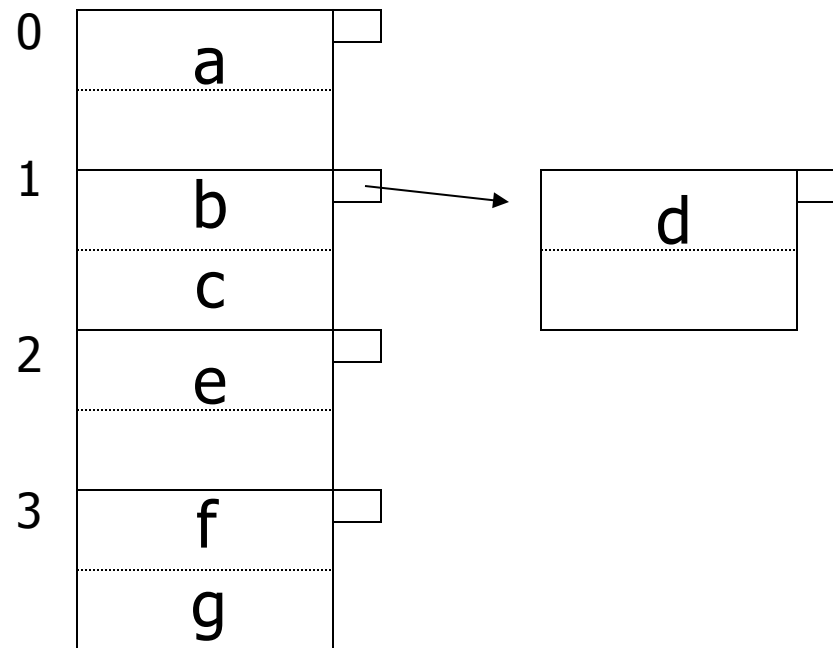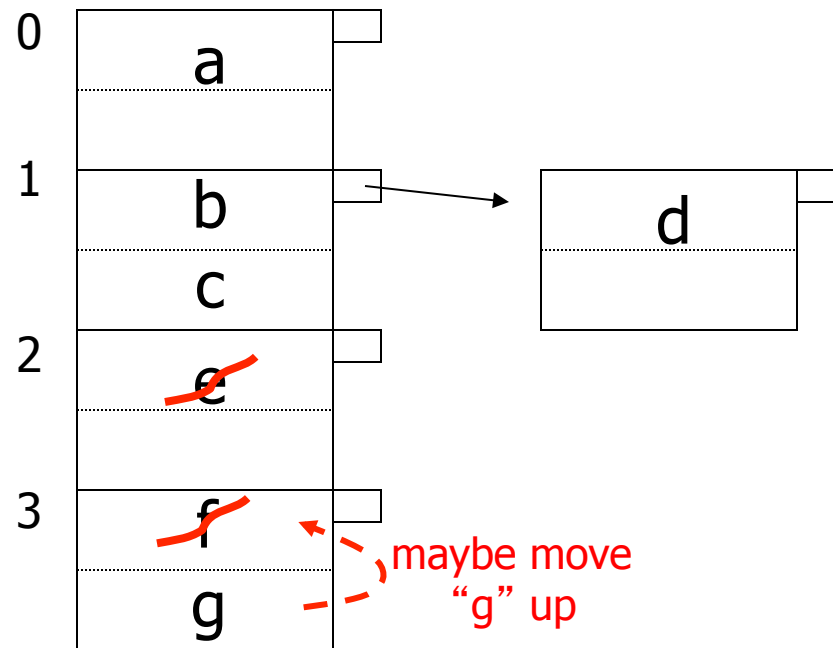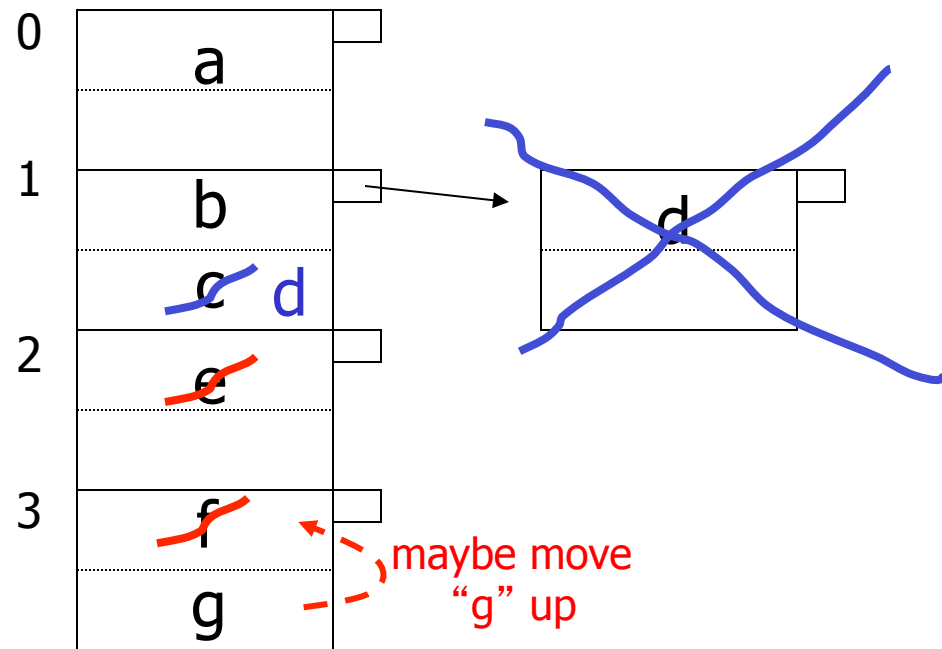
Rule of thumb:

- Try to keep space utilization between 50% and 80%

$$\text{Utilization} = \frac{\#\ \text{keys used}}{\text{total}\ \#\ \text{keys that fit}}$$

- If < 50%, wasting space

- If > 80%, overflows significant
  ↳ depends on how good hash function is & on # keys/bucket

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

How do we cope with growth?

- Overflows and reorganizations
- Dynamic hashing

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

How do we cope with growth?

- Overflows and reorganizations
- Dynamic hashing

  - Extensible
  - Linear

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Extensible hashing: two ideas

(a) Use *i* of *b* bits output by hash function

$$\longleftarrow \quad b \quad \longrightarrow$$

h(K) → | 00110101 |

use *i* → grows over time….

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(b) Use directory

$h(K)[i]$ → to bucket

COMPUTER SCIENCE

IIT College of
Science and Letters
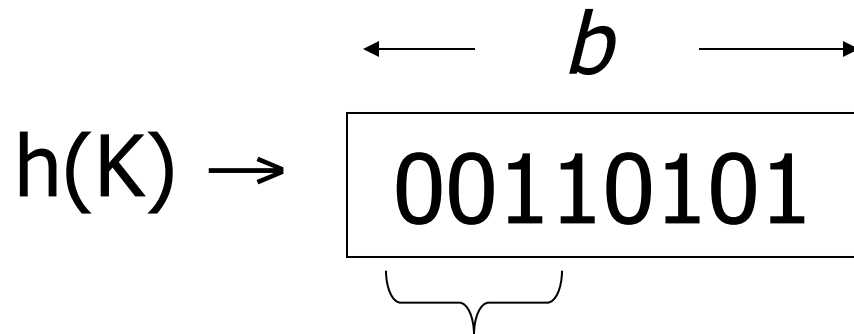
ILLINOIS INSTITUTE OF TECHNOLOGY

Example: h(k) is 4 bits; 2 keys/bucket

$i =$ [1]

<u>1</u>

0001

<u>1</u>

1001
1100
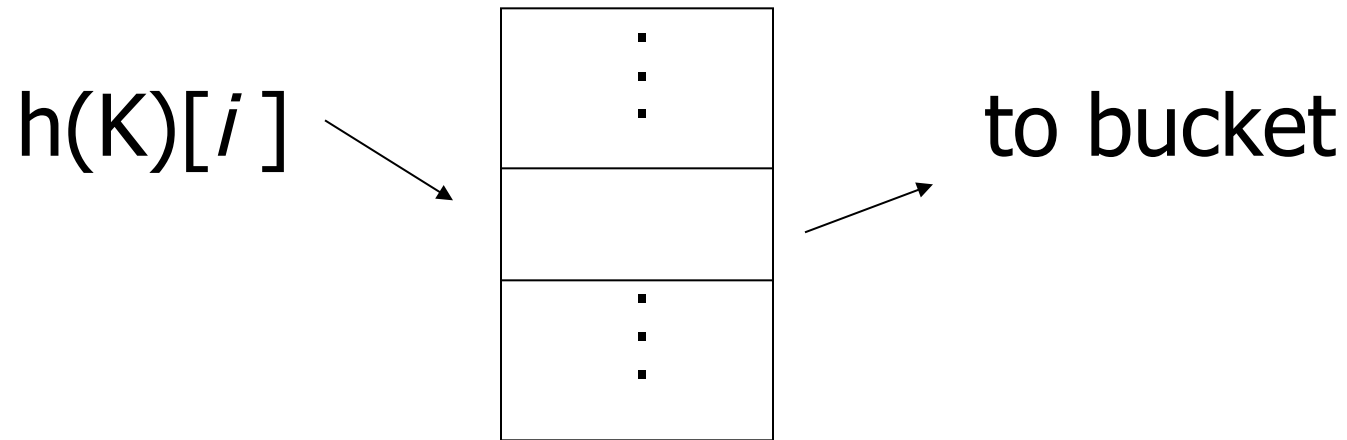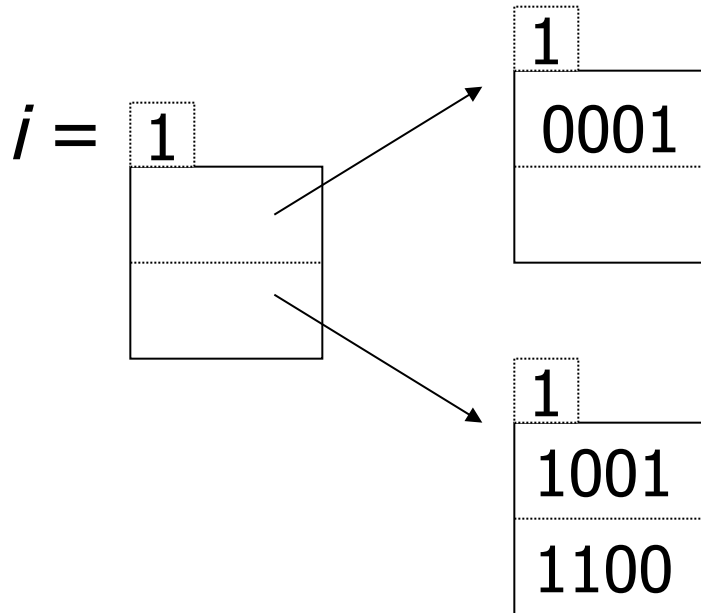
Insert 1010

Example: h(k) is 4 bits; 2 keys/bucket

$i = $ 1

1
| 0001 |
| |

1
| 1001 |
| 1100 |

1010

Insert 1010

1
| 1100 |
| |

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Example: h(k) is 4 bits; 2 keys/bucket

i = 2

l = 1

1

0001

1 2

1001

1010 1100

1 2

1100

Insert 1010

New directory

00

01

10

11

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example continued

i = 2

```
00
01
10
11
```

1
0001

2
1001
1010

2
1100

Insert:

0111

0000

COMPUTER
SCIENCE

Notes 5 - Hashing

26

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example continued

i = 2

00

01

10

11

0000
0001

1
~~0001~~ 0111
~~0111~~

2
1001
1010

2
1100

Insert:

0111

0000

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example continued

2

0000

0001

i = 2

00

01

10

11

1 2

0001 ̶0̶1̶1̶1̶

0̶1̶1̶1̶

2

1001

1010

Insert:

0111

0000

2

1100

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example continued

$i =$ 2

00

01

10

11

Insert:

1001

2 | 0000 | 0001 |

2 | 0111 | |

2 | 1001 | 1010 |

2 | 1100 | |

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example continued

$i =$ 2

00

01

10

11

Insert:

1001

| 0000 | 2 |
| 0001 | |

| 0111 | 2 |
| | |

| 1001 | |
| 1001 | |

1010

| ~~1001~~ | 2 |
| ~~1010~~ | |

| 1100 | 2 |
| | |

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example continued

$i = 2$

00

01

10

11

$$\boxed{0000} \quad 2$$
$$0001$$

$$\boxed{0111} \quad 2$$

$$1001 \quad 3$$
$$1001$$

1010 $\boxed{1001}$ 2 3
$$1010$$

$$\boxed{1100} \quad 2$$

$i = 3$

000

001

010

011

100

101

110

111

Insert:

1001

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Extensible hashing: <u>deletion</u>

- No merging of blocks
- Merge blocks
  and cut directory if possible
  (Reverse insert procedure)

COMPUTER SCIENCE

Notes 5 - Hashing

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion example:

- Run thru insert example in reverse!

COMPUTER SCIENCE

Notes 5 - Hashing

33

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Note: Still need overflow chains

- Example: many records with duplicate keys

insert 1100

if we split:

```
  1
┌────────┐
│  1101  │
├········┤
│  1100  │
└────────┘
```

```
  2
┌────────┐
│        │
├········┤
│        │
└────────┘
```

```
  2
┌────────┐
│  1100  │
├········┤
│  1100  │
└────────┘
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
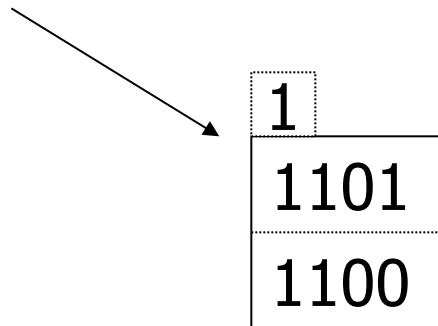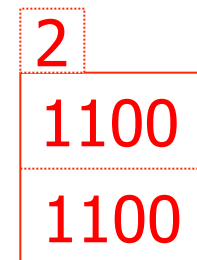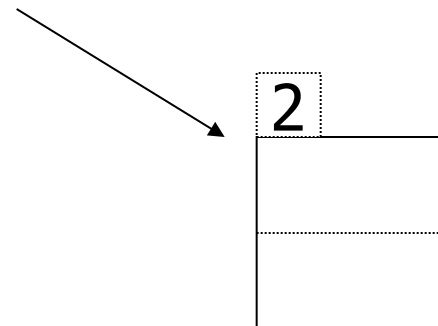
Solution: overflow chains

insert 1100
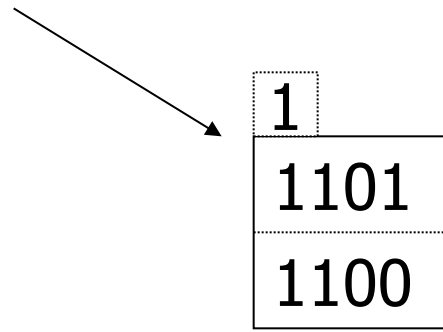
add overflow block:
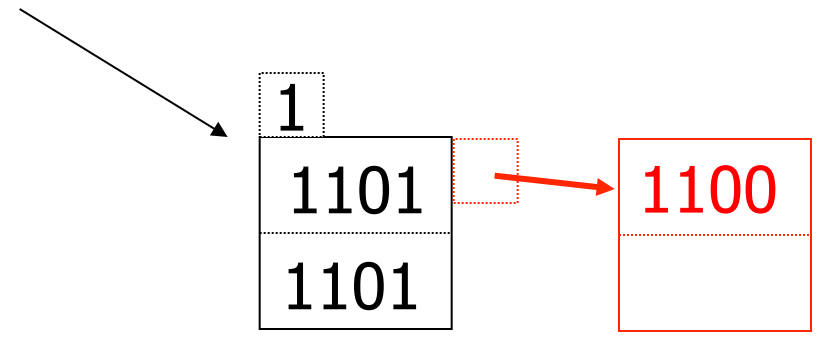
COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Summary | Extensible hashing

(+) Can handle growing files

     - with less wasted space

     - with no full reorganizations

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Summary — Extensible hashing

(+) Can handle growing files

    - with less wasted space

    - with no full reorganizations

(-) Indirection

    (Not bad if directory in memory)

(-) Directory doubles in size

    (Now it fits, now it does not)

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Linear hashing

- Another dynamic hashing scheme

Two ideas:

(a) Use i low order bits of hash

$$\overset{\longleftarrow \quad b \quad \longrightarrow}{\boxed{01110101}}$$

grows $\longleftarrow \underbrace{}_{i}$

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Linear hashing

- Another dynamic hashing scheme

Two ideas:

(a) Use i <u>low</u> order bits of hash

$$\overset{\longleftarrow \quad b \quad \longrightarrow}{\boxed{01110101}}$$

grows $\longleftarrow$ i

(b) File grows linearly

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example b=4 bits, i =2, 2 keys/bucket

0000	0101		Future growth buckets ←
1010	1111		

00   01   10   11

m = 01 (max used block)

Example b=4 bits, i =2, 2 keys/bucket

0000	0101		
1010	1111		
00	01	10	11

Future growth buckets ←

m = 01 (max used block)

Rule	If h(k)[i] ≤ m, then

look at bucket h(k)[i]

else, look at bucket h(k)[i] - 2^{i-1}

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example b=4 bits, i =2,   2 keys/bucket

• insert 0101

Future growth buckets ←

0000	0101		
1010	1111		

00   01   10   11

m = 01 (max used block)

Rule   If h(k)[i] ≤ m, then

　　　　look at bucket h(k)[i]

　　　　else, look at bucket h(k)[i] - 2^{i-1}

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example $b=4$ bits, $i=2$, 2 keys/bucket

0101

• insert 0101

• can have overflow chains!

0000		0101						Future growth buckets
1010		1111						

00   01   10   11

$m = 01$ (max used block)

Rule	If h(k)[i] ≤ m, then

look at bucket h(k)[i]

else, look at bucket h(k)[i] - 2^{i-1}

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
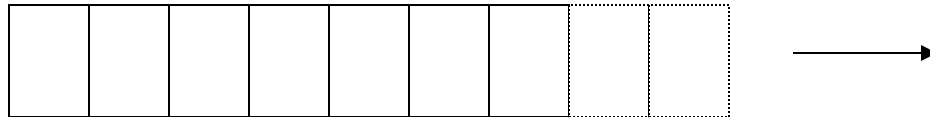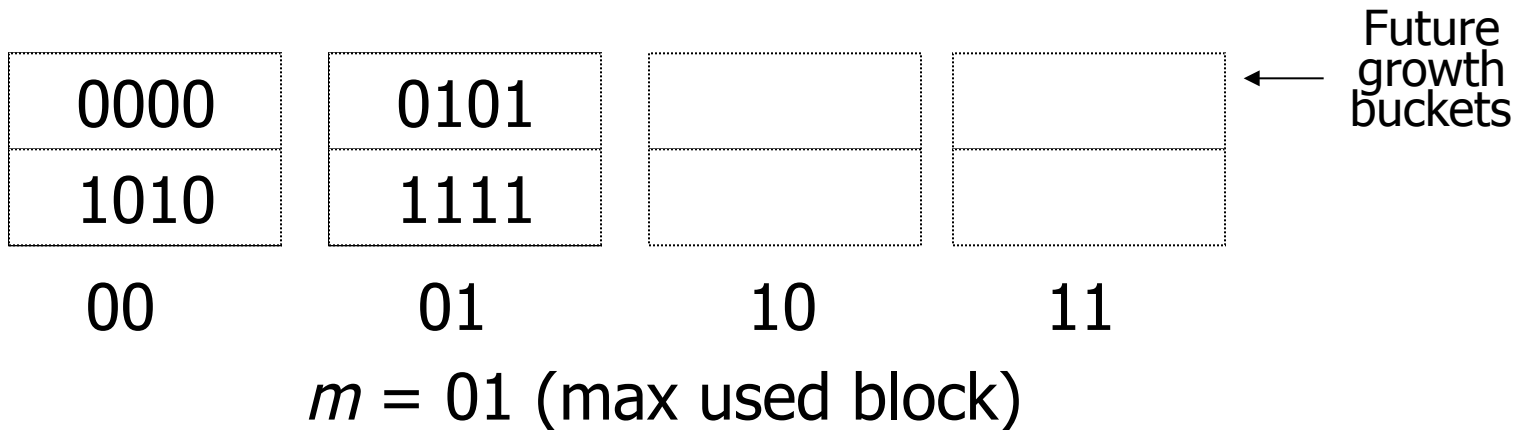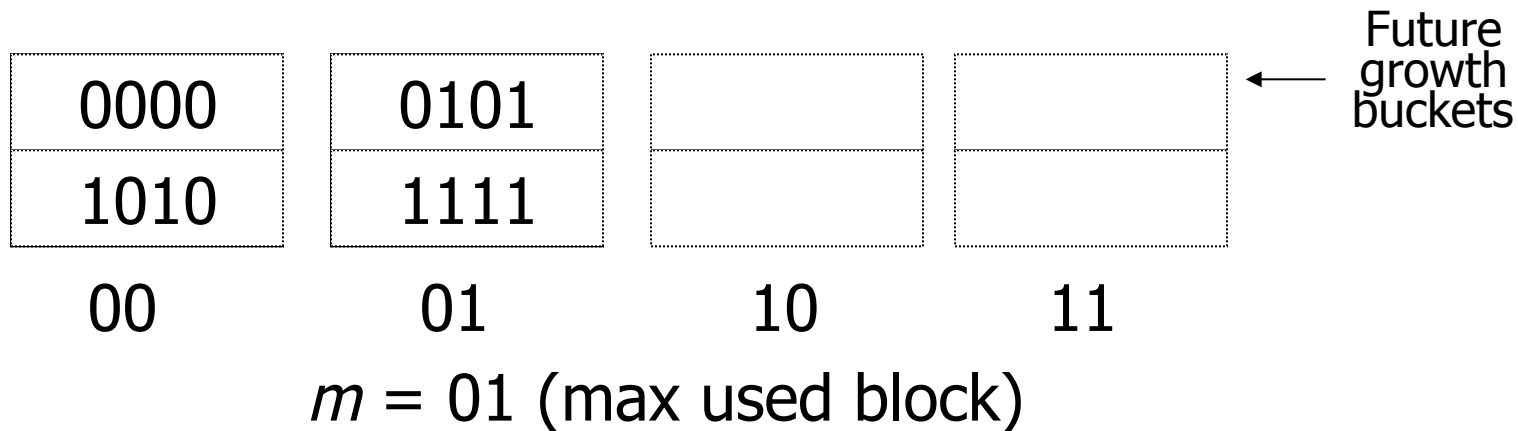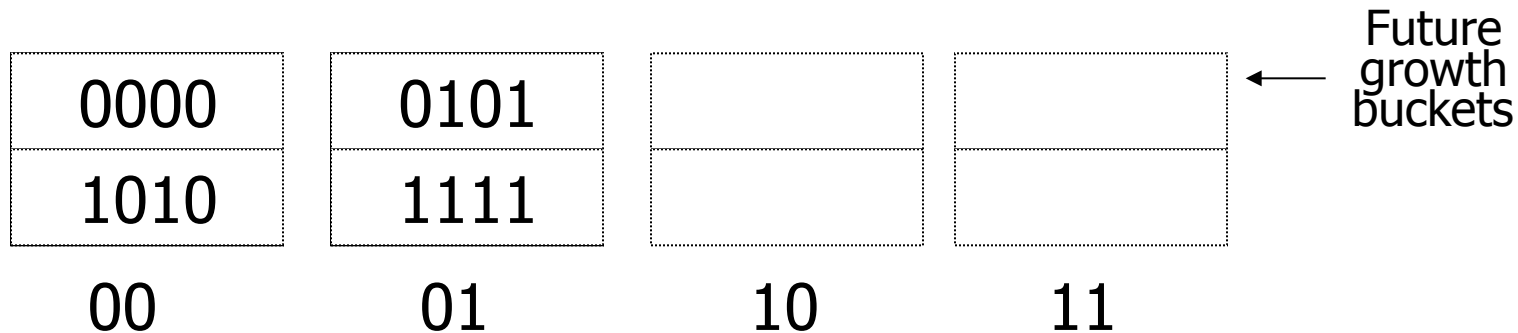
Note

- In textbook, n is used instead of m
- n=m+1

n=10

0000	0101			← Future growth buckets
1010	1111			
00	01	10	11	

m = 01 (max used block)

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example b=4 bits, i =2,   2 keys/bucket

0000	0101		
1010	1111		

00   01   10   11

m = 01 (max used block)

Future growth buckets ←

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example b=4 bits, i =2,   2 keys/bucket

0000		0101		1010			
~~1010~~		1111					

00   01   10   11

m = ~~01~~ (max used block)
10

Future growth buckets

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example b=4 bits, i =2, 2 keys/bucket

0101

• insert 0101

0000		0101		1010			Future growth buckets
~~1010~~		1111				

00   01   10   11

m = ~~01~~ (max used block)
       10

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example b=4 bits, i=2, 2 keys/bucket

0101	

• insert 0101

0000	0101	1010	
~~1010~~	1111		

Future growth buckets

00   01   10   11

$m = $ ~~01~~ (max used block)
   ~~10~~
   11

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example b=4 bits, i =2, 2 keys/bucket

• insert 0101

Future growth buckets

$m = 01$ (max used block)

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example Continued: How to grow beyond this?

$i = 2$

0000	0101	1010	1111
	0101		

  00   01   10   11

. . .

$m = 11$ (max used block)

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example Continued: How to grow beyond this?

$i = \cancel{2}\ 3$

0000	0101	1010	1111		
	0101				

000   001   010   011
100   101   110   111   . . .

$m = 11$ (max used block)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example Continued: How to grow beyond this?

$i = \cancel{2}\ 3$

0000	0101	1010	1111		
	0101				

0 00   0 01   0 10   0 11   100
~~100~~   101   110   111
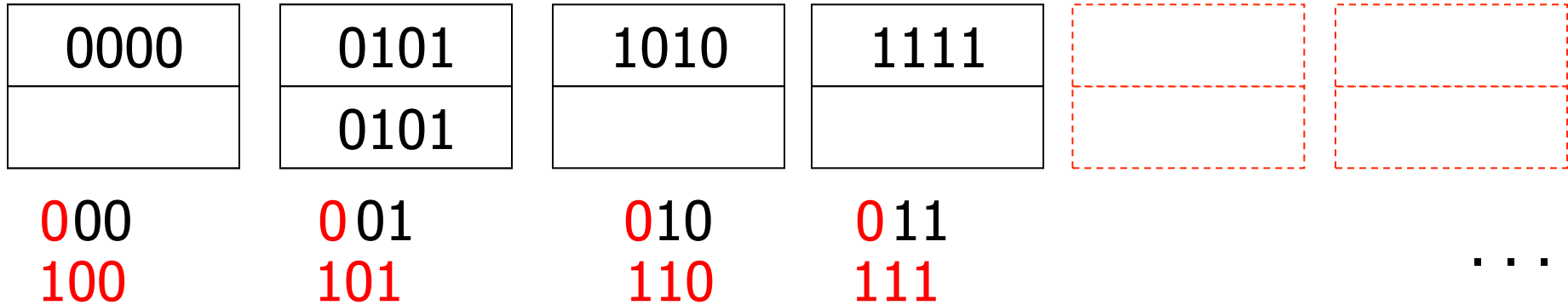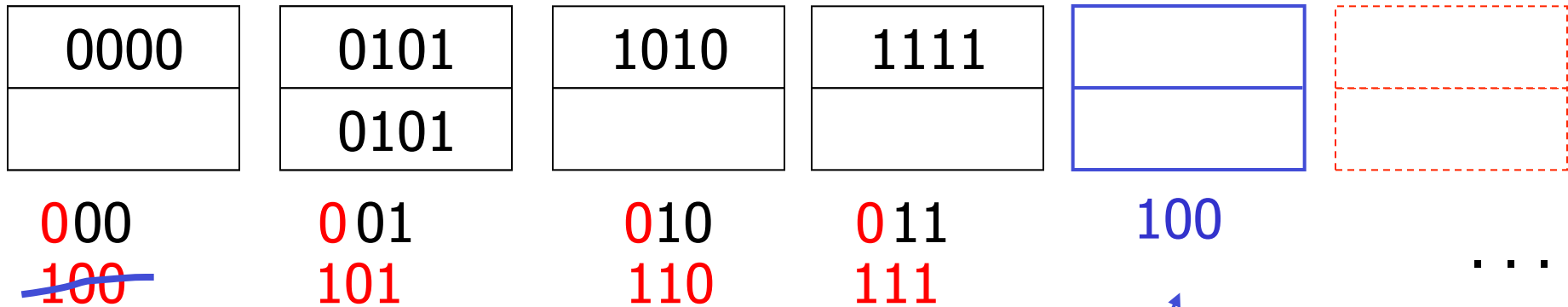
$m = \cancel{11}$ (max used block)

100

Example Continued: How to grow beyond this?
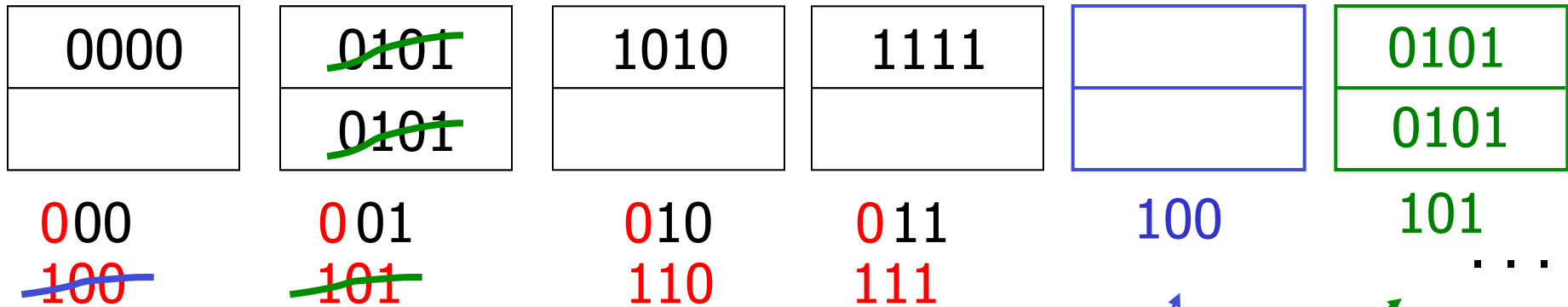
$i = \not{2}\ 3$

0000

0̶1̶0̶1̶
0̶1̶0̶1̶

1010

1111

0101
0101

000   0 01   010   0 11   100   101
1̶0̶0̶   1̶0̶1̶   110   111   · · ·

$m = \not{11}$ (max used block)

1̶0̶0̶

101

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

☛ When do we expand file?

- Keep track of: $\dfrac{\text{\# used slots}}{\text{total \# of slots}} = U$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

☛ When do we expand file?

- Keep track of: $\dfrac{\text{\# used slots}}{\text{total \# of slots}} = U$

- If U > threshold then increase *m* (and maybe *i*)

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Summary — Linear Hashing

+ Can handle growing files
  - with less wasted space
  - with no full reorganizations

+ No indirection like extensible hashing

- Can still have overflow chains

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example: BAD CASE

Very full

Very empty

Need to move

m here…

Would waste

space…

Notes 5 - Hashing

57

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
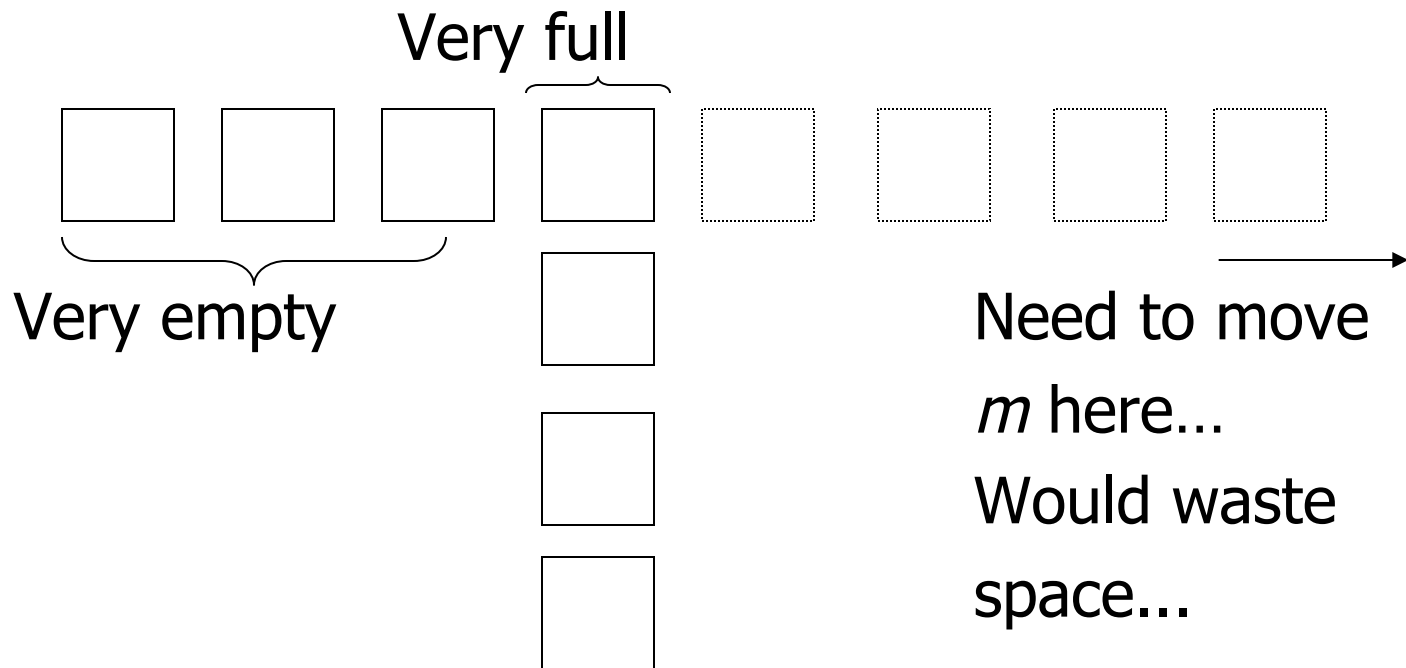
Summary

Hashing

    - How it works

    - Dynamic hashing

        - Extensible

        - Linear

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Next:

- Indexing vs Hashing
- Index definition in SQL
- Multiple key access

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Indexing vs Hashing

- Hashing good for probes given key

  e.g.,   SELECT ...

                 FROM R

                 WHERE R.A = 5

-> **Point Queries**

Indexing vs Hashing

- INDEXING (Including B Trees) good for

  Range Searches:

  e.g.,   SELECT

  FROM R

  WHERE R.A > 5

-> **Range Queries**

Index definition in SQL

- <u>Create</u> <u>index</u> name <u>on</u> rel (attr)
- <u>Create</u> <u>unique</u> <u>index</u> name <u>on</u> rel (attr)

                              → defines candidate key

- <u>Drop</u> INDEX name

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

| Note | CANNOT SPECIFY TYPE OF INDEX

　　　(e.g. B-tree, Hashing, …)

OR PARAMETERS

　　　(e.g. Load Factor, Size of Hash,…)

… at least in standard SQL…

Vendor specific extensions allow that

Note ATTRIBUTE <u>LIST</u> $\Rightarrow$ MULTIKEY INDEX (next)

e.g., <u>CREATE</u> <u>INDEX</u> foo <u>ON</u> R(A,B,C)

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Multi-key Index

Motivation: Find records where

$$DEPT = \text{"Toy"} \text{ AND } SAL > 50k$$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Strategy I:

- Use one index, say Dept.
- Get all Dept = "Toy" records
  and check their salary
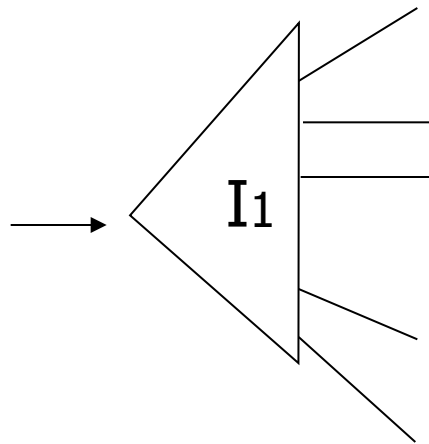
Strategy II:

- Use 2 Indexes; Manipulate Pointers

Toy → ⊔⊔⊔⊔⊔⊔   ⊔⊔⊔⊔⊔⊔⊔⊔←Sal
                              > 50k

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
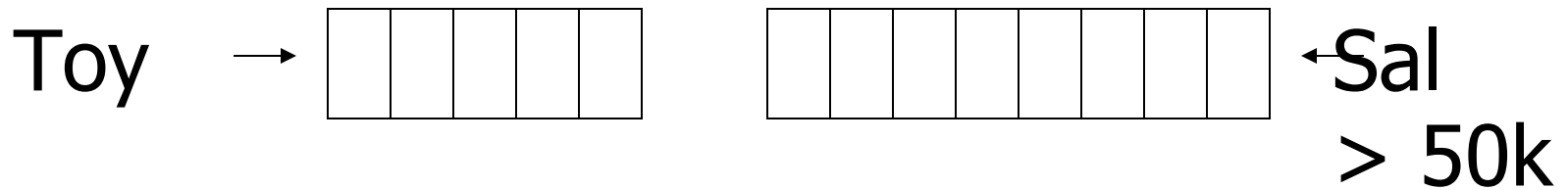
Strategy III:

- Multiple Key Index

One idea:

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

10k	
15k	
17k	
21k	

Art	
Sales	
Toy	

Dept Index

12k	
15k	
15k	
19k	

Salary Index

Example Record

Name=Joe
DEPT=Sales
SAL=15k
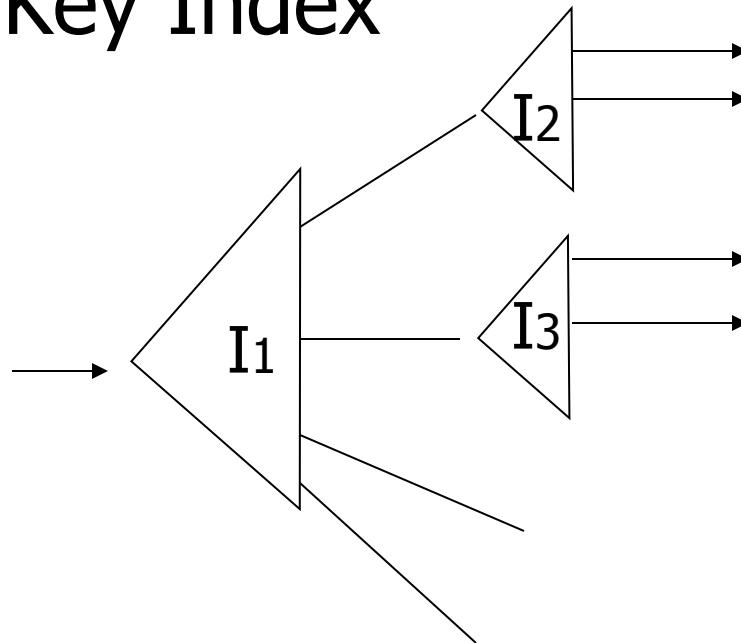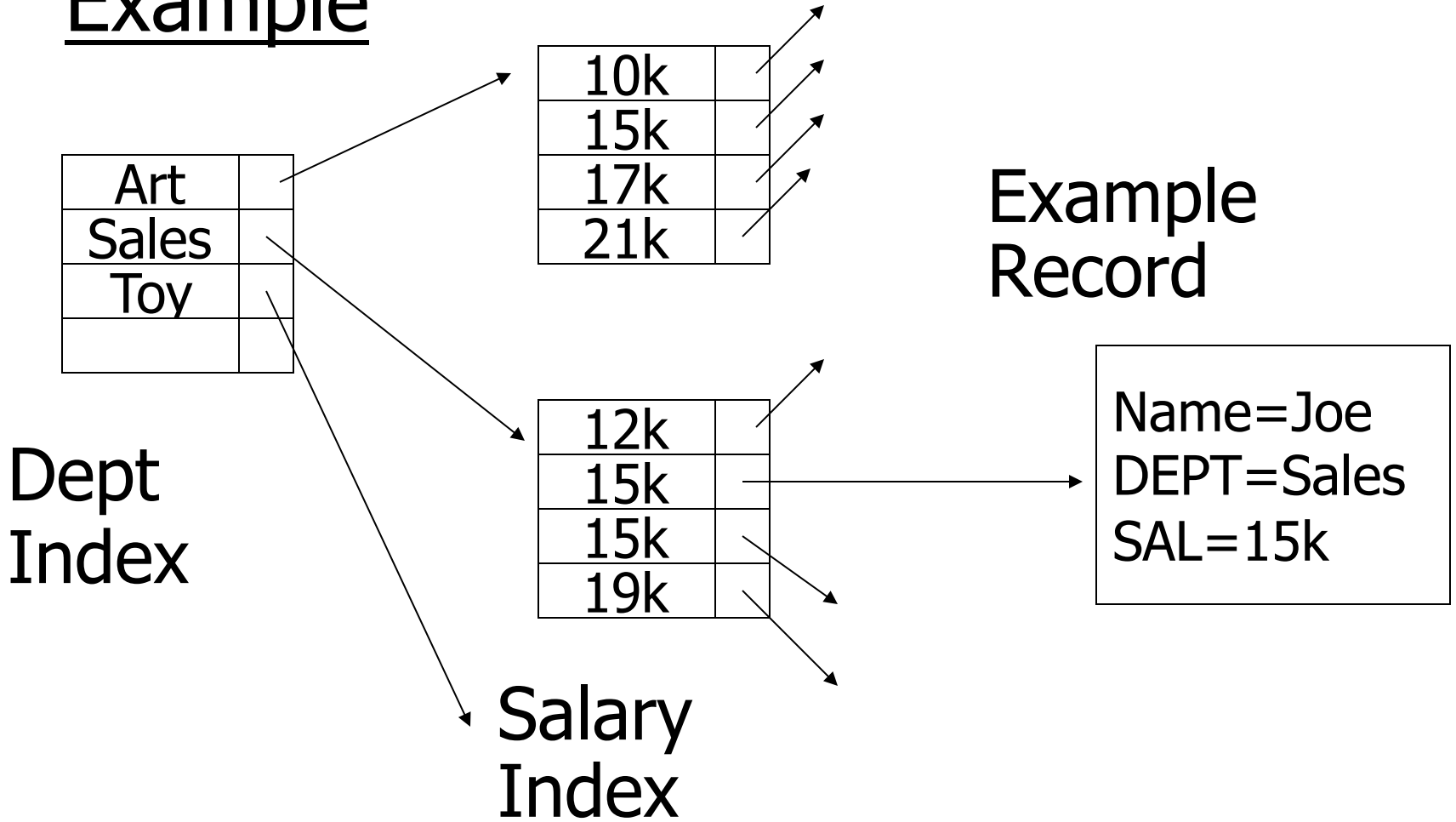
For which queries is this index good?

☐ Find RECs Dept = "Sales" $\wedge$ SAL=20k

☐ Find RECs Dept = "Sales" $\wedge$ SAL $\geq$ 20k

☐ Find RECs Dept = "Sales"

☐ Find RECs SAL = 20k

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Interesting application:

- Geographic Data
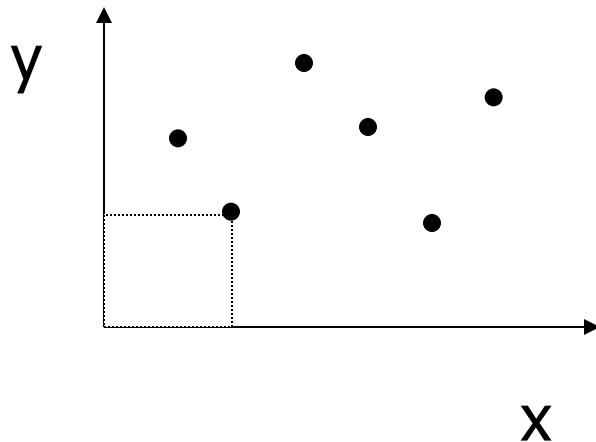
DATA:

$\langle X_1, Y_1,$ Attributes$\rangle$

$\langle X_2, Y_2,$ Attributes$\rangle$

⋮

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Queries:

- What city is at $<X_i, Y_i>$?
- What is within 5 miles from $<X_i, Y_i>$?
- Which is closest point to $<X_i, Y_i>$?

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Notes 5 - Hashing

Example

10   20

Notes 5 - Hashing

74

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example

- Search points near f
- Search points near b

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Queries

- Find points with $Y_i > 20$
- Find points with $X_i < 5$
- Find points "close" to $i = <12,38>$
- Find points "close" to $b = <7,24>$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Next

- Even more index structures ☺

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

CS 525: Advanced Database Organization
06: Even more index structures

Boris Glavic

Slides: adapted from a course taught by
Hector Garcia-Molina, Stanford InfoLab

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Recap

- We have discussed
  - Conventional Indices
  - B-trees
  - Hashing
  - Trade-offs
  - Multi-key indices
  - Multi-dimensional indices
    - ... but no example

Today

- **Multi-dimensional index structures**
  - kd-Trees (very similar to example before)
  - **Grid File (Grid Index)**
  - Quad Trees
  - **R Trees**
  - **Partitioned Hash**
  - …
- **Bitmap-indices**
- **Tries**

Grid Index
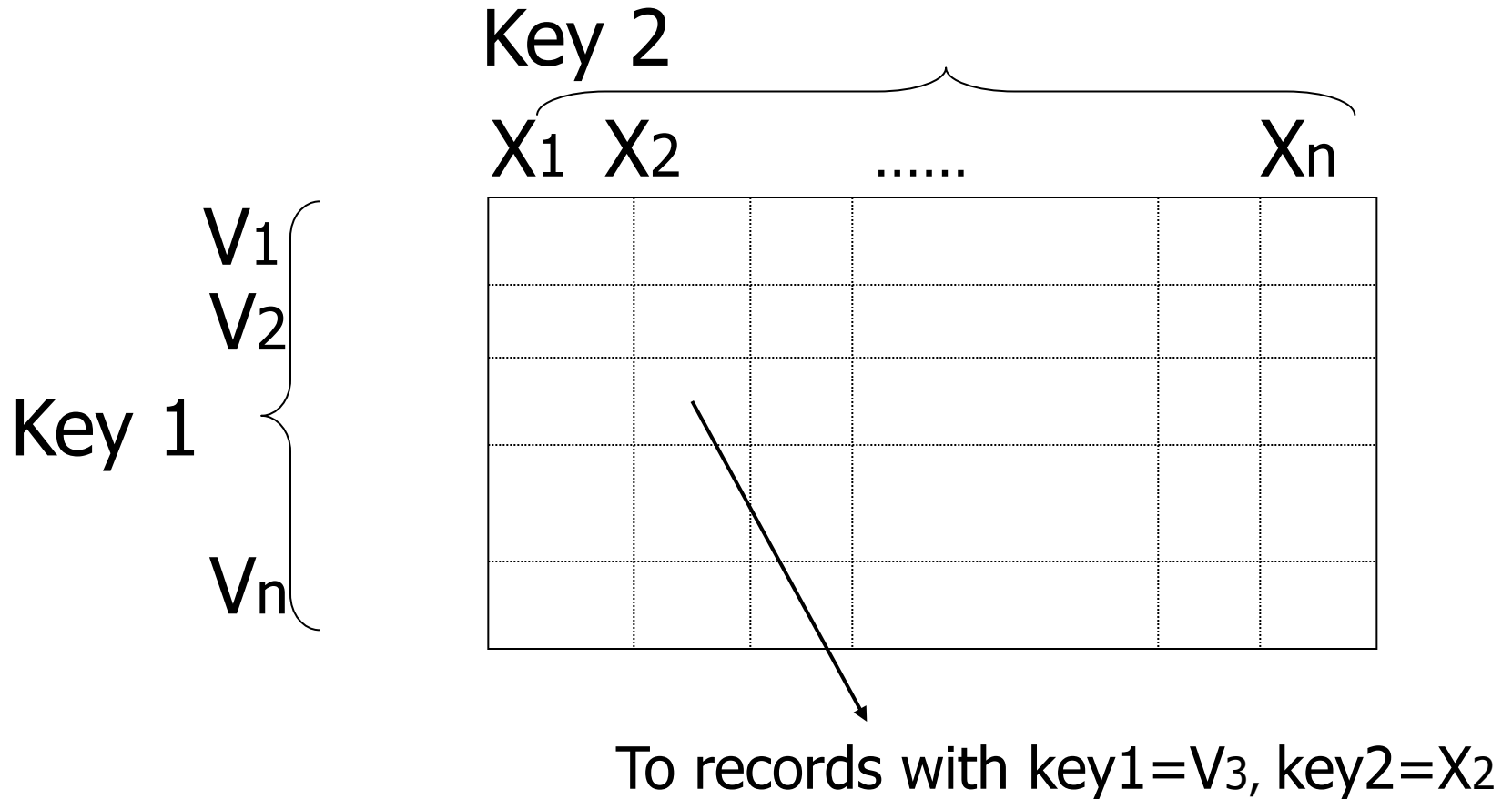
Key 2

X_1 X_2 ...... X_n

V_1
V_2

Key 1

V_n

To records with key1=V_3, key2=X_2

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

CLAIM

- Can quickly find records with
  - key 1 = V_i ∧ Key 2 = X_j
  - key 1 = V_i
  - key 2 = X_j

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

CLAIM

- Can quickly find records with
  - key 1 = V_i $\wedge$ Key 2 = X_j
  - key 1 = V_i
  - key 2 = X_j

- And also ranges….
  - E.g., key 1 $\geq$ V_i $\wedge$ key 2 < X_j

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

- How do we find entry i,j in linear structure?

max number of
i values N=4

i, j

0, 0	← position S+0
0, 1	← position S+1
0, 2	← position S+2
0, 3	← position S+3
1, 0	← position S+4
1, 1
1, 2
1, 3
2, 0
2, 1	← position S+9
2, 2
2, 3
3, 0

pos(i, j) =

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
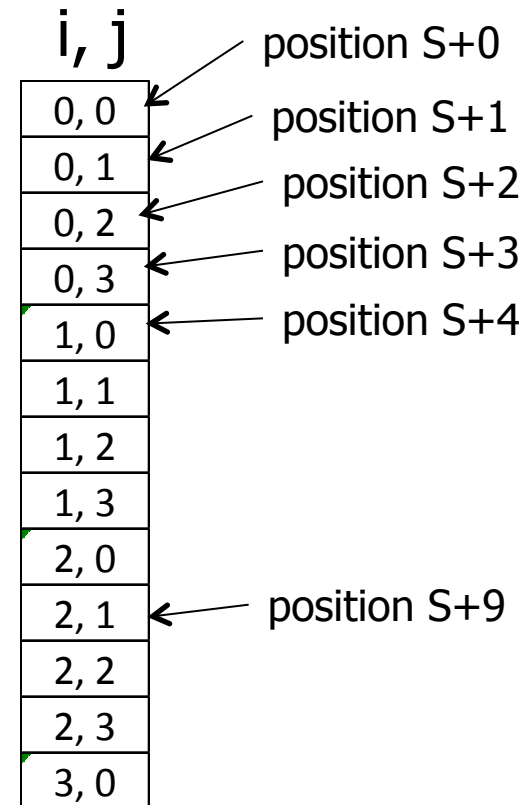
- How do we find entry i,j in linear structure?

max number of
i values N=4

pos(i, j) = S + iN + j

Issue: Cells must be same size,
and N must be constant!

⬇

Issue: Some cells may overflow,
some may be sparse...

i, j

0, 0	← position S+0
0, 1	← position S+1
0, 2	← position S+2
0, 3	← position S+3
1, 0	← position S+4
1, 1	
1, 2	
1, 3	
2, 0	
2, 1	← position S+9
2, 2	
2, 3	
3, 0	

COMPUTER
SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
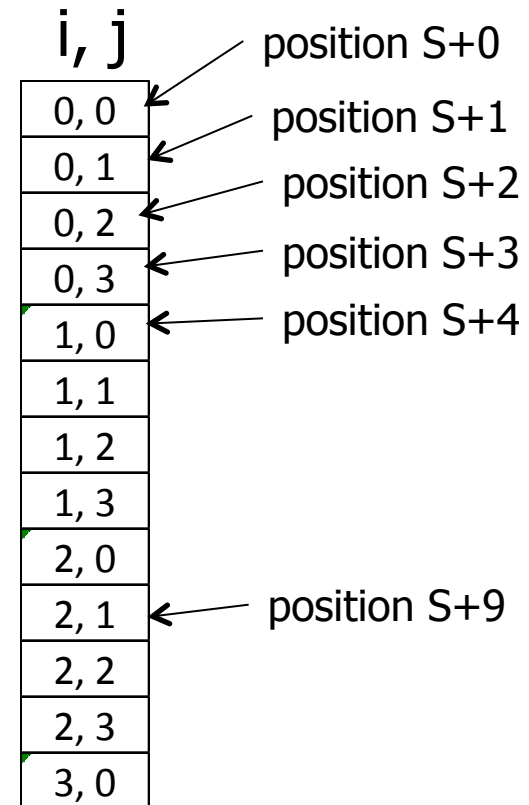
Solution: Use Indirection

Buckets

X1   X2   X3

V_1
V_2
V_3
V_4

*Grid only
   contains
   pointers to
   buckets

Buckets

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
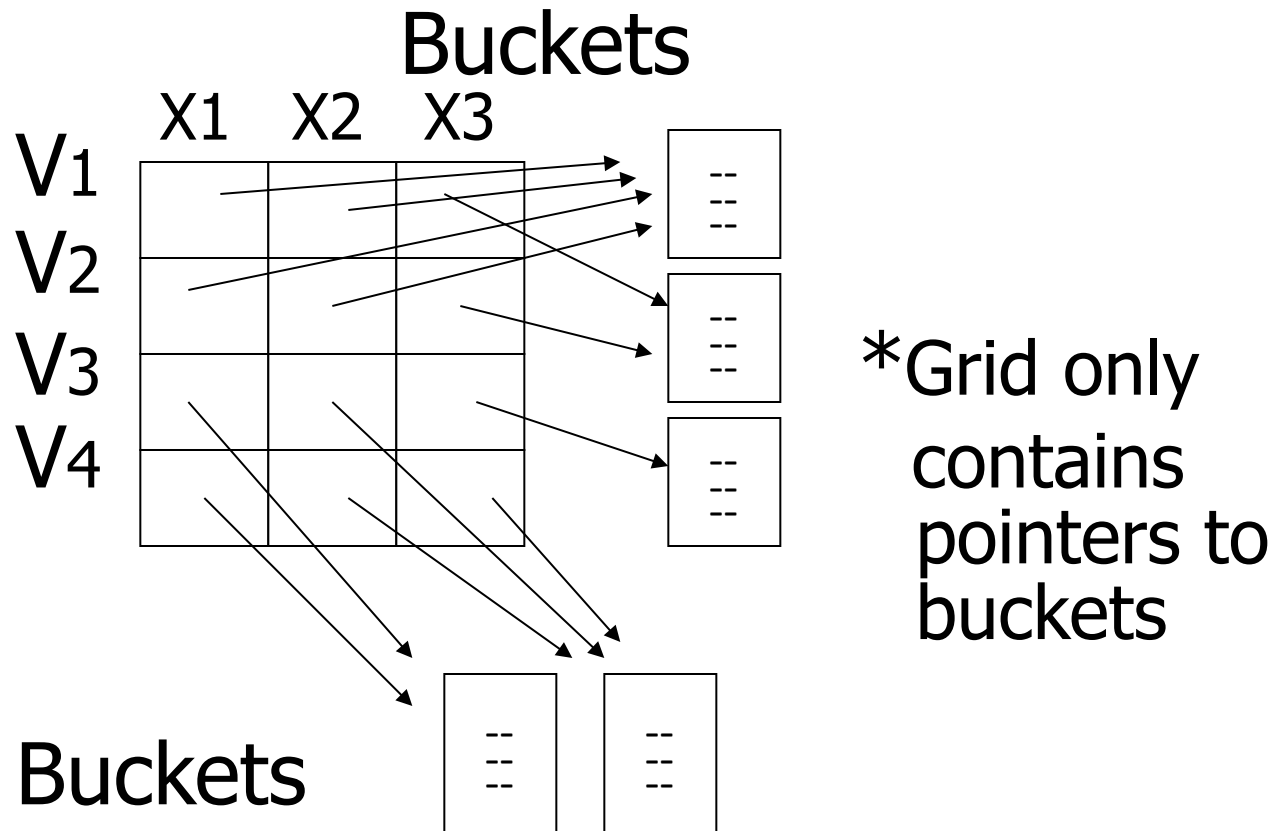
With indirection:

- Grid can be regular without wasting space
- We do have price of indirection

Can also index grid on <u>value ranges</u>

Salary   Grid

0-20K	1
20K-50K	2
50K- ∞	3

```
Linear Scale
```

1	2	3
Toy	Sales	Personnel

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
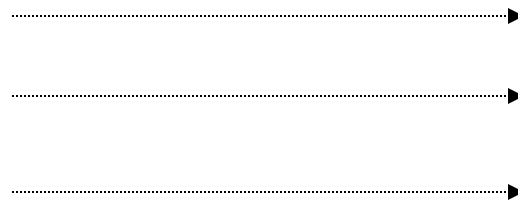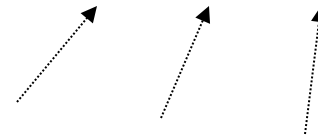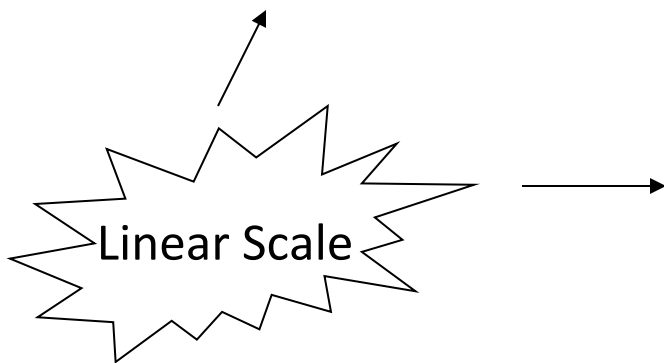
Grid files

⊕ Good for multiple-key search

⊖ Space, management overhead

(nothing is free)

⊖ Need partitioning ranges that evenly split keys

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Partitioned hash function

Idea:

010110 1110010

Key1   Key2

h1   h2

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

EX:

h1(toy)   =0

h1(sales) =1

h1(art)   =1

   .

h2(10k)   =01

h2(20k)   =11

h2(30k)   =01

h2(40k)   =00

   .

000	
001	
010	
011	
100	
101	
110	
111	

Insert ⟹ <Fred,toy,10k>,<Joe,sales,10k>
<Sally,art,30k>

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy)   =0

h1(sales) =1

h1(art)   =1

.

h2(10k)   =01

h2(20k)   =11

h2(30k)   =01

h2(40k)   =00

.

000	
001	<Fred>
010	
011	
100	
101	<Joe><Sally>
110	
111	

Insert ⟹ <Fred,toy,10k>,<Joe,sales,10k>
<Sally,art,30k>

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy)   =0
h1(sales) =1
h1(art)   =1

  .

h2(10k)   =01
h2(20k)   =11
h2(30k)   =01
h2(40k)   =00

  .

000	<Fred>
001	<Joe><Jan>
010	<Mary>
011	
100	<Sally>
101	
110	<Tom><Bill>
111	<Andy>

Find Emp. with Dept. = Sales $\wedge$ Sal=40k

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy)   =0

h1(sales) =1

h1(art)   =1

   .

h2(10k) =01

h2(20k) =11

h2(30k) =01

h2(40k) =00

   .

000	<Fred>
001	<Joe><Jan>
010	<Mary>
011	
100	<Sally>
101	
110	<Tom><Bill>
111	<Andy>
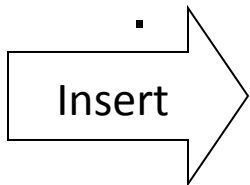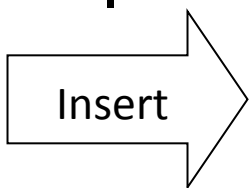
Find Emp. with Dept. = Sales $\wedge$ Sal=40k

EX:

h1(toy)   =0

h1(sales) =1

h1(art)   =1

.

h2(10k)   =01

h2(20k)   =11

h2(30k)   =01

h2(40k)   =00

.

Find Emp. with Sal=30k

000	<Fred>
001	<Joe><Jan>
010	<Mary>
011	
100	<Sally>
101	
110	<Tom><Bill>
111	<Andy>

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy)   =0
h1(sales) =1
h1(art)   =1

    .

h2(10k)   =01
h2(20k)   =11
h2(30k)   =01
h2(40k)   =00

    .

Find Emp. with Sal=30k

000	<Fred>
001	<Joe><Jan>
010	<Mary>
011	
100	<Sally>
101	
110	<Tom><Bill>
111	<Andy>

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy)   =0

h1(sales) =1

h1(art)   =1

   .

h2(10k)   =01

h2(20k)   =11

h2(30k)   =01

h2(40k)   =00

   .

000	<Fred>
001	<Joe><Jan>
010	<Mary>
011	
100	<Sally>
101	
110	<Tom><Bill>
111	<Andy>

Find Emp. with Dept. = Sales

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy)   =0
h1(sales) =1
h1(art)   =1

.

h2(10k)   =01
h2(20k)   =11
h2(30k)   =01
h2(40k)   =00

.

Find Emp. with Dept. = Sales

000	<Fred>
001	<Joe><Jan>
010	<Mary>
011	
100	<Sally>
101	
110	<Tom><Bill>
111	<Andy>

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

R-tree

- Nodes can store up to **M** entries
  - Minimum fill requirement (depends on variant)
- Each node rectangle in **n**-dimensional space
  - Minimum Bounding Rectangle (MBR) of its children
- MBRs of siblings are allowed to overlap
  - Different from B-trees
- balanced

Data Space

[5-7]	[9-15]	[13-19]	
[20-24]	[12-16]	[2-4]	

[5]	[6]	[7]	
[24]	[20]	[24]	

[13]	[14]	[18]	[19]
[4]	[2]	[2]	[3]

[9]	[11]	[15]	
[15]	[16]	[12]	

Notes 6 - More Indices   23

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
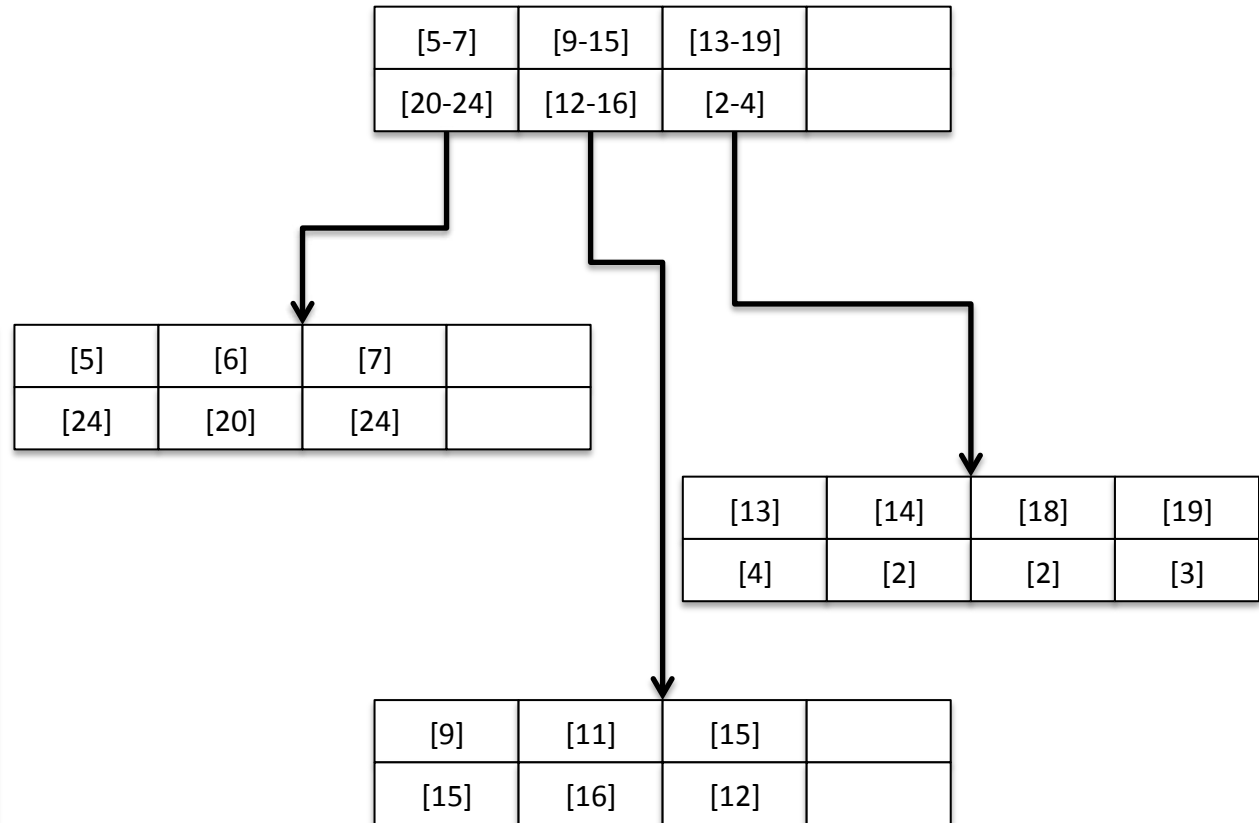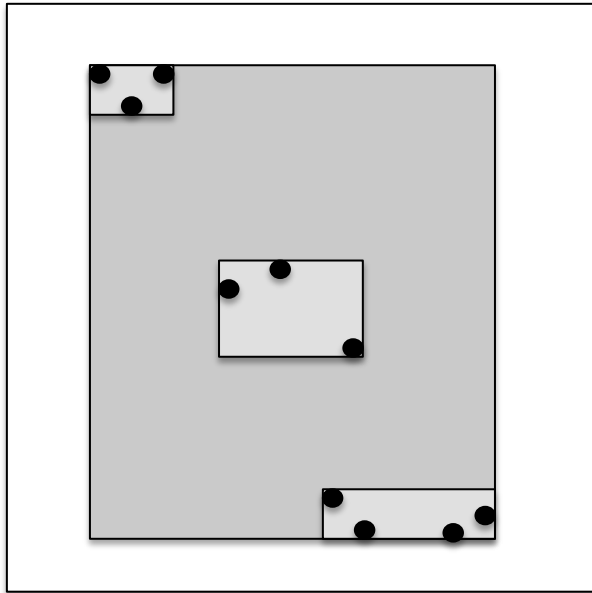
R-tree - Search

- Point Search
  - Search for $p = <x_i, y_i>$
  - Keep list of potential nodes
    - Needed because of overlap
  - Traverse to child if MBR of child contains p

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
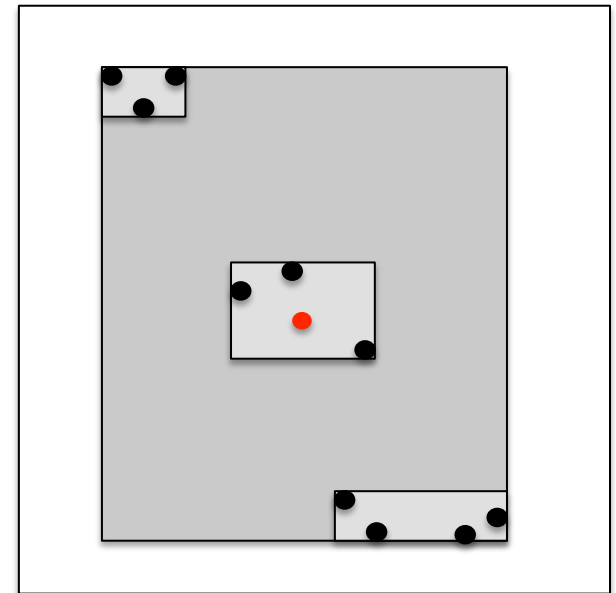
R-tree - Search

- Point Search
  - Search for points in region = $<[x_{min} - x_{max}], [y_{min} - y_{max}]>$
  - Keep list of potential nodes
  - Traverse to child if MBR of child overlaps with query region

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
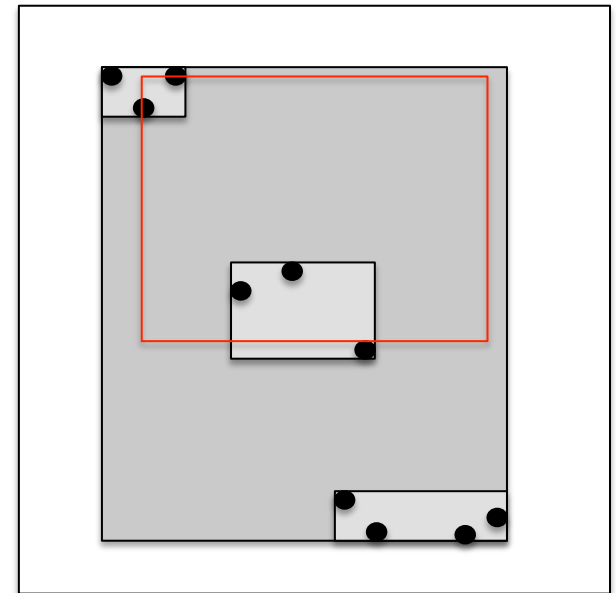
Search <5,24>

Data Space

[5-7]	[9-15]	[13-19]	
[20-24]	[12-16]	[2-4]	

[5]	[6]	[7]	
[24]	[20]	[24]	

[13]	[14]	[18]	[19]
[4]	[2]	[2]	[3]

[9]	[11]	[15]	
[15]	[16]	[12]	

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
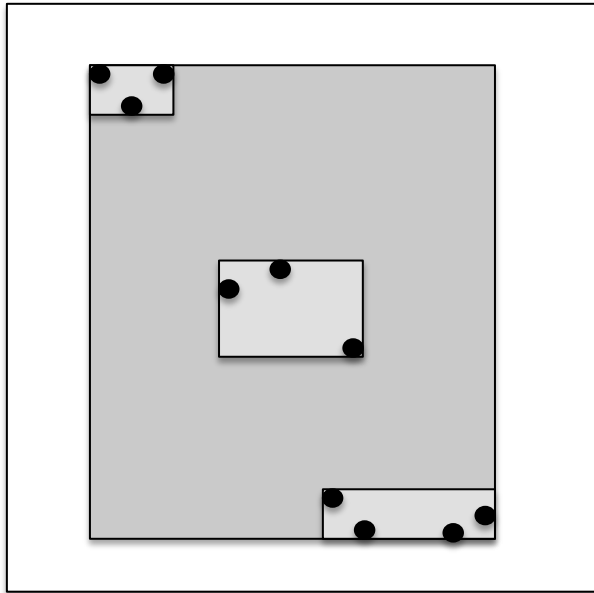
R-tree - Insert

- Similar to B-tree, but more complex
  - Overlap -> multiple choices where to add entry
  - Split harder because more choice how to split node (compare B-tree = 1 choice)
- 1) Find potential subtrees for current node
  - Choose one for insert (heuristic, e.g., the one the would grow the least)
  - Continue until leaf is found

R-tree - Insert

- 2) Insert into leaf
- 3) Leaf is full? -> split
  - Find best split (minimum overlap between new nodes) is hard ($O(2^M)$)
  - Use linear or quadratic heuristics (original paper)
- 4) Adapt parents if necessary

R-tree - Delete

- 1) Find leaf node that contains entry

- 2) Delete entry

- 3) Leaf node underflow?
  - Remove leaf node and cache entries
  - Adapt parents
  - Reinsert deleted entries

Bitmap Index

- Domain of values $D = \{d_1, ..., d_n\}$
  - Gender {male, female}
  - Age {1, ..., 120?}

- Use one vector of bits for each value
  - One bit for each record
    - 0: record has different value in this attribute
    - 1: record has this value

Bitmap Index Example

Age

1	2	3
1	0	0
0	1	0
1	0	0
0	0	1

Todlers

Name	Age	Gender
Peter	1	male
Gertrud	2	female
Joe	1	male
Marry	3	female

Gender

male	female
1	0
0	1
1	0
0	1

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Bitmap Index Example

Age

1	2	3
1	0	0
0	1	0
1	0	0
0	0	1

Todlers

Name	Age	Gender
Peter	1	male
Gertrud	2	female
Joe	1	male
Marry	3	female

Gender

male	female
1	0
0	1
1	0
0	1

Find all todlers with age **2 and** sex **female**:
Bitwise-and between vectors

0
1
0
0

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Bitmap Index Example

Age

1	2	3
1	0	0
0	1	0
1	0	0
0	0	1

Todlers

Name	Age	Gender
Peter	1	male
Gertrud	2	female
Joe	1	male
Marry	3	female

Gender

male	female
1	0
0	1
1	0
0	1

Find all todlers with age **2 or** sex **female**:
Bitwise-or between vectors

0
1
0
1

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Compression

- Observation:
  - Each record has one value in indexed attribute
  - For N records and domain of size $|D|$
    - Only $1/|D|$ bits are 1
  - -> waste of space

- Solution
  - Compress data
  - Need to make sure that **and** and **or** is still fast

Run length encoding (RLE)

- Instead of actual 0-1 sequence encode length of 0 or 1 runs

- One bit to indicate whether 0/1 run + several bits to encode run length

- But how many bits to use to encode a run length?

  - Gamma codes or similar to have variable number of bits

RLE Example

- 0001 0000 1110 1111 **(2 bytes)**
- 3, 1,4, 3, 1,4 **(6 bytes)**
- -> if we use one byte to encode a run we have 7 bits for length = max run length is 128(127)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Elias Gamma Codes

- $X = 2^N + (x \bmod 2^N)$
  - Write N as N zeros followed by one 1
  - Write $(x \bmod 2^N)$ as N bit number
- $18 = 2^4 + 2 = 000010010$

- 0001 0000 1110 1111   **(2 bytes)**
- 3, 1,4,   3, 1,4   **(6 bytes)**
- 0111 0010 0011 1001 00   **(3 bytes)**

Hybrid Encoding

- Run length encoding
  - Can waste space
  - And/or run length not aligned to byte/word boundaries
- Encode some bytes of sequence as is and only store long runs as run length
  - EWAH
  - BBC (that's what Oracle uses)

Extended Word aligned Hybrid (EWAH)

- Segment sequence in machine words (64bit)
- Use two types of words to encode
  - Literal words, taken directly from input sequence
  - Run words
    - ½ word is used to encode a run
    - ½ word is used to encode how many literals follow

0000 0000	0000 0000	0010 1000	1111 1111	1100 0010

0010 0001	0010 1000	1001 0001	1100 0010

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
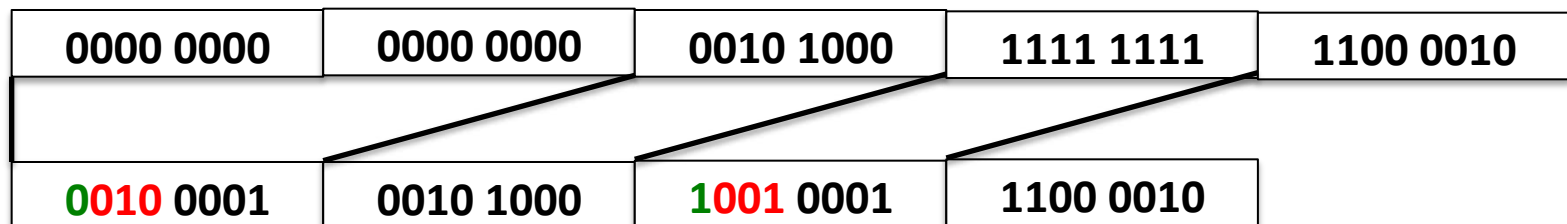
Bitmap Indices

- Fast for read intensive workloads
    - Used a lot in datawarehousing
- Often build on the fly during query processing
    - As we will see later in class

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Trie

- From Retrieval

- Tree index structure

- Keys are sequences of values from a domain D
  - D = {0,1}
  - D = {a,b,c,....,z}

- Key size may or may not be fixed
  - Store 4-byte integers using D = {0,1} (32 elements)
  - Strings using D={a,...,z} (arbitrary length)

Trie

- Each node has pointers to |D| child nodes
  - One for each value of D

- Searching for a key k = $[d_1, ..., d_n]$
  - Start at the root
  - Follow child for value d_i

Trie Example

Words: bar, ball, in

Search for **bald**

Fail !

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
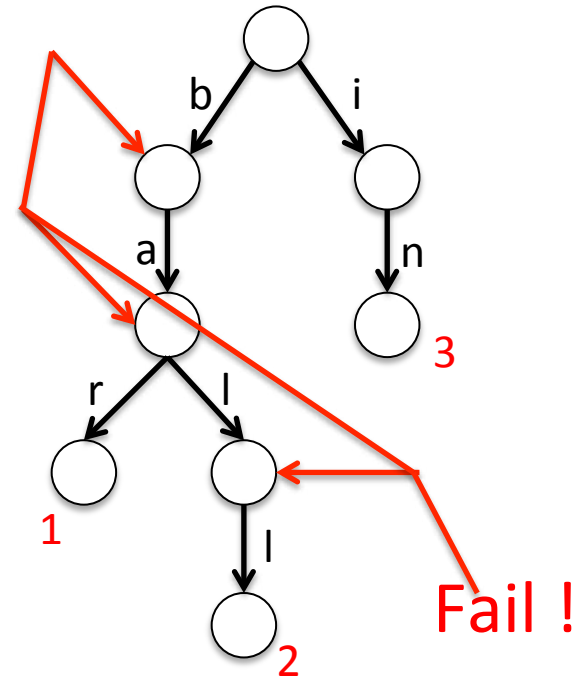
Tries Implementation

- 1) Each node has an array of child pointers
- 2) Each node has a list or hash table of child pointers
- 3) array compression schemes derived from compressed DFA representations

Summary

Discussion:

     - Conventional Indices

     - B-trees

     - Hashing (extensible, linear)

     - SQL Index Definition

     - Index vs. Hash

     - Multiple Key Access

     - Multi Dimensional Indices

          Variations: Grid, R-tree,

     - Partitioned Hash

     - Bitmap indices and compression

     - Tries

Notes 5 - Hashing   45

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

CS 525: Advanced Database Organisation

07: Query Processing Overview

Boris Glavic

Slides: adapted from a course taught by
Hector Garcia-Molina, Stanford InfoLab

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Query Processing

Q $\rightarrow$ Query Plan

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Query Processing

Q → Query Plan

Focus: Relational Systems

- Others?

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Select B,D

From R,S

Where R.A = "c" $\wedge$ S.E = 2 $\wedge$ R.C=S.C

R	A	B	C
	a	1	10
	b	1	20
	c	2	10
	d	2	35
	e	3	45

S	C	D	E
	10	x	2
	20	y	2
	30	z	2
	40	x	1
	50	y	3

R	A	B	C	S	C	D	E
	a	1	10		10	x	2
	b	1	20		20	y	2
	c	2	10		30	z	2
	d	2	35		40	x	1
	e	3	45		50	y	3

Answer

B	D
2	x

Notes 7 - Query Processing

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

- How do we execute query?

One idea ▷

- Do Cartesian product
- Select tuples
- Do projection

Notes 7 - Query Processing

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

RXS

R.A	R.B	R.C	S.C	S.D	S.E
a	1	10	10	x	2
a	1	10	20	y	2
.					
.					
C	2	10	10	x	2
.					
.					

RXS

R.A	R.B	R.C	S.C	S.D	S.E
a	1	10	10	x	2
a	1	10	20	y	2
.					
.					
C	2	10	10	x	2
.					
.					

Bingo! →

Got one...

Relational Algebra - can be used to describe plans...
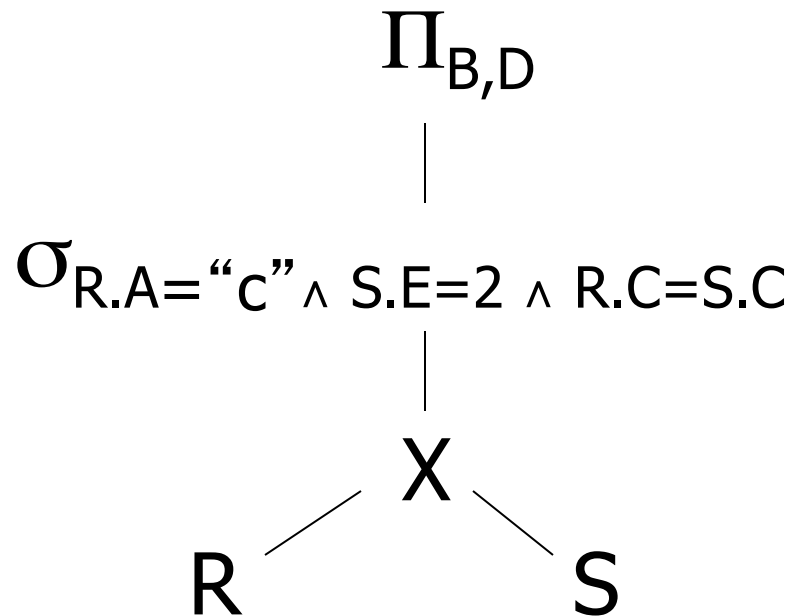
Ex: Plan I

$$\Pi_{B,D}$$

$$\sigma_{R.A=\text{``}c\text{''} \wedge S.E=2 \wedge R.C=S.C}$$

$$X$$

$$R \qquad S$$

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Relational Algebra - can be used to describe plans...

Ex: Plan I

$$\Pi_{B,D}$$

|

$$\sigma_{R.A=\text{``c''} \wedge S.E=2 \wedge R.C=S.C}$$

|

X
/ \
R   S

OR: $\Pi_{B,D} [\sigma_{R.A=\text{``c''} \wedge S.E=2 \wedge R.C = S.C} (RXS)]$

COMPUTER SCIENCE

Another idea:

Plan II

$$\Pi_{B,D}$$

$$\bowtie$$

$$\sigma_{R.A = \text{``c''}} \qquad \sigma_{S.E = 2}$$

R   S

$\bowtie$ natural join

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

R

A	B	C
a	1	10
b	1	20
c	2	10
d	2	35
e	3	45

σ (R)

A	B	C
c	2	10

σ(S)

C	D	E
10	x	2
20	y	2
30	z	2

S

C	D	E
10	x	2
20	y	2
30	z	2
40	x	1
50	y	3

$\bowtie$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Plan III

Use R.A and S.C Indexes

    (1) Use R.A index to select R tuples
        with R.A = "c"

    (2) For each R.C value found, use S.C
        index to find matching tuples

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Plan III

Use R.A and S.C Indexes

(1) Use R.A index to select R tuples
with R.A = "c"

(2) For each R.C value found, use S.C
index to find matching tuples

(3) Eliminate S tuples S.E $\neq$ 2

(4) Join matching R,S tuples, project
B,D attributes and place in result

COMPUTER
SCIENCE

Notes 7 - Query Processing

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

R

A	B	C
a	1	10
b	1	20
c	2	10
d	2	35
e	3	45

A

I₁ ←

S

C I₂ →

C	D	E
10	x	2
20	y	2
30	z	2
40	x	1
50	y	3

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

R

A	B	C
a	1	10
b	1	20
c	2	10
d	2	35
e	3	45

A= "c"

I_1

<c,2,10>

S

C

I_2

C	D	E
10	x	2
20	y	2
30	z	2
40	x	1
50	y	3

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

R

A	B	C
a	1	10
b	1	20
c	2	10
d	2	35
e	3	45

$A=$ "c"   C

I_1 I_2

S

C	D	E
10	x	2
20	y	2
30	z	2
40	x	1
50	y	3

<c,2,10>

<10,x,2>

check=2?

output: <2,x>

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

R

A	B	C
a	1	10
b	1	20
c	2	10
d	2	35
e	3	45

A= "c"   **C**

I₁   I₂

<c,2,10>   <10,x,2>

check=2?

output: <2,x>

next tuple:
<c,7,15>

S

C	D	E
10	x	2
20	y	2
30	z	2
40	x	1
50	y	3

Overview of Query Optimization

CS 525

Notes 7 - Query Processing

21

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

SQL query

parse

parse tree

convert

logical query plan

apply laws

"improved" l.q.p

estimate result sizes

statistics

answer

execute

Pi

pick best

{(P1,C1),(P2,C2)...}

l.q.p. +sizes

consider physical plans

estimate costs

{P1,P2,.....}

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example: SQL query

SELECT title

FROM StarsIn

WHERE starName IN (

      SELECT name

      FROM MovieStar

      WHERE birthdate LIKE '%1960'

);

(Find the movies with stars born in 1960)

Example: Parse Tree


```
                              <Query>
                                 |
                              <SFW>
         /        /       |        |        |          \
   SELECT  <SelList>  FROM  <FromList>  WHERE   <Condition>
              |                 |                  /    |      \
         <Attribute>       <RelName>        <Tuple>  IN  <Query>
              |                 |              /           /      \
            title           StarsIn     <Attribute>  (  <Query>  )
                                              |              |
                                          starName        <SFW>
             /       /       |        |         |          \
       SELECT  <SelList>  FROM  <FromList>  WHERE   <Condition>
                 |                 |                /    |     \
            <Attribute>       <RelName>     <Attribute> LIKE <Pattern>
                 |                 |              |            |
               name           MovieStar      birthDate     '%1960'
```

Notes 7 - Query Processing

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Example: Generating Relational Algebra

$$\Pi_{title}$$

$$\sigma$$

StarsIn   `<condition>`

`<tuple>`   IN Π_{name}

`<attribute>` $\sigma_{birthdate\ LIKE\ '\%1960'}$

starName   MovieStar
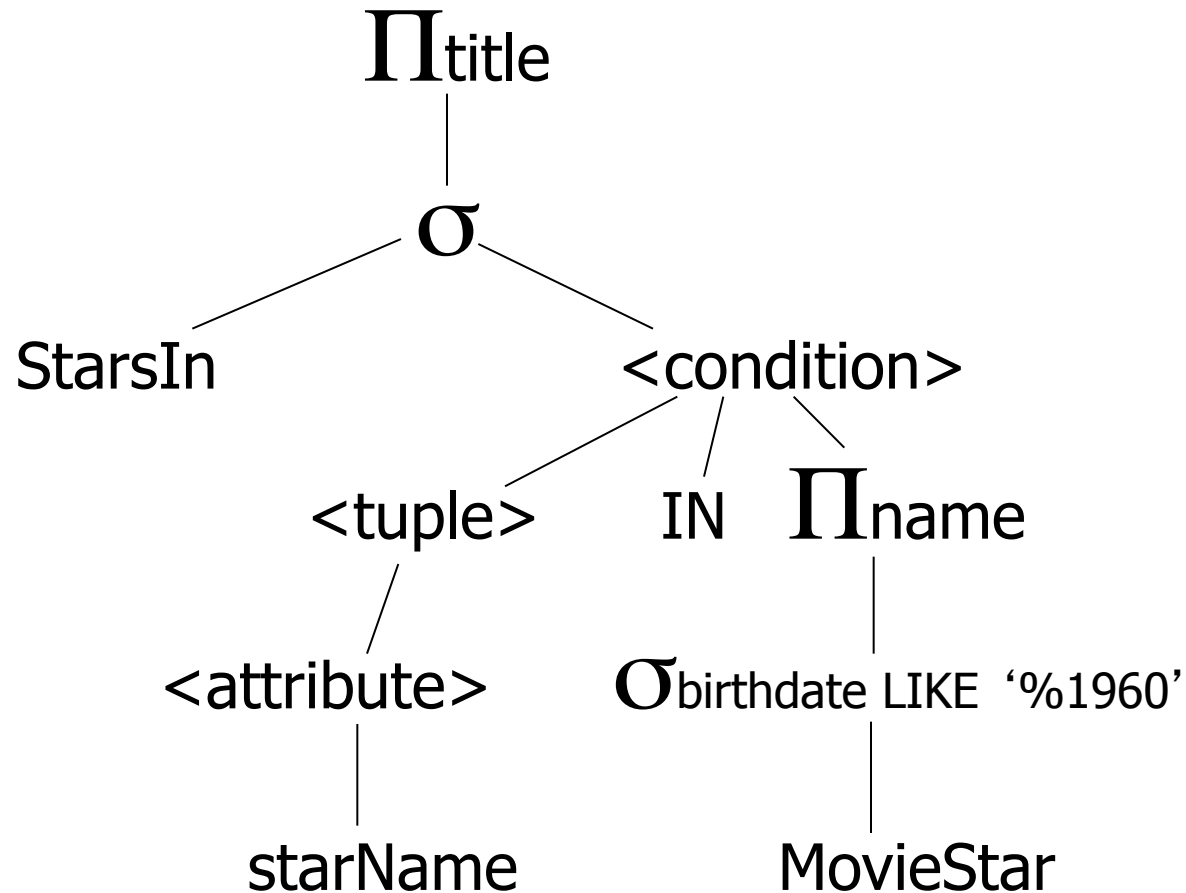
Fig. 7.15: An expression using a two-argument σ, midway between a parse tree
and relational algebra

COMPUTER
SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Logical Query Plan

$$\Pi_{\text{title}}$$

$$\sigma_{\text{starName=name}}$$

$$\times$$

StarsIn Π_{name}

$$\sigma_{\text{birthdate LIKE '\%1960'}}$$

MovieStar
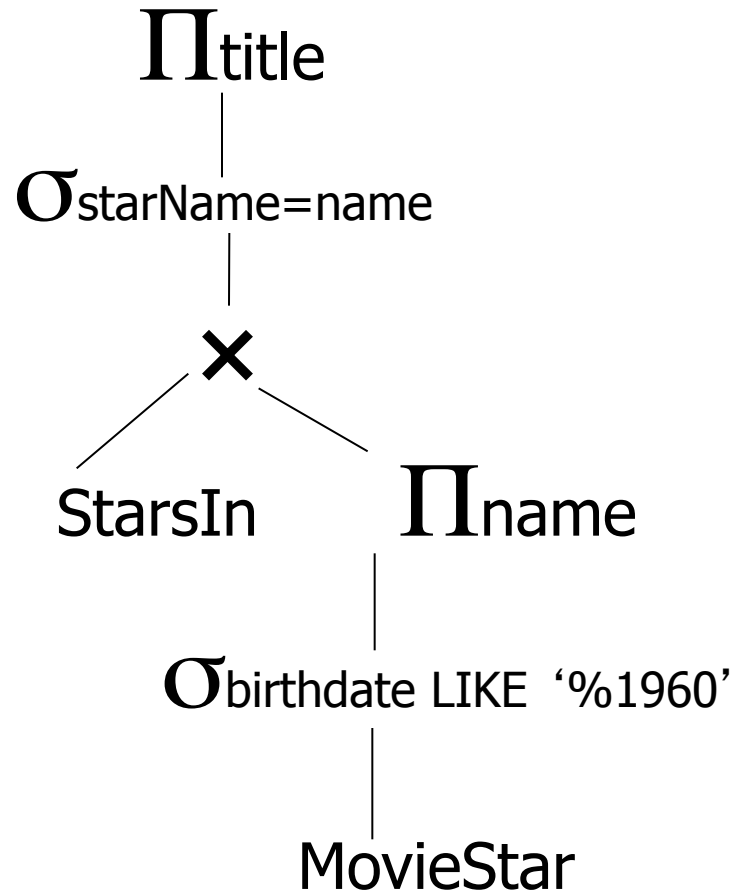
Fig. 7.18: Applying the rule for IN conditions

Example: Improved Logical Query Plan

$$\Pi_{title}$$

$$\bowtie$$
starName=name

StarsIn $$\Pi_{name}$$

$$\sigma_{birthdate \ LIKE \ \text{'%1960'}}$$

MovieStar
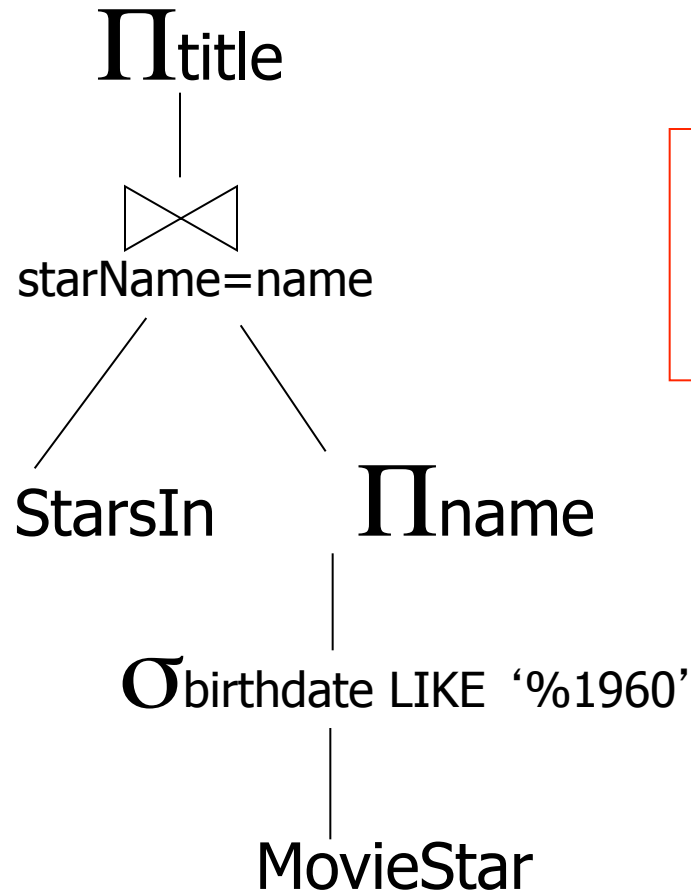
Question:
Push project to
StarsIn?

Fig. 7.20: An improvement on fig. 7.18.

Example: Estimate Result Sizes

⋈

StarsIn   Need expected size

Π

σ

MovieStar

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
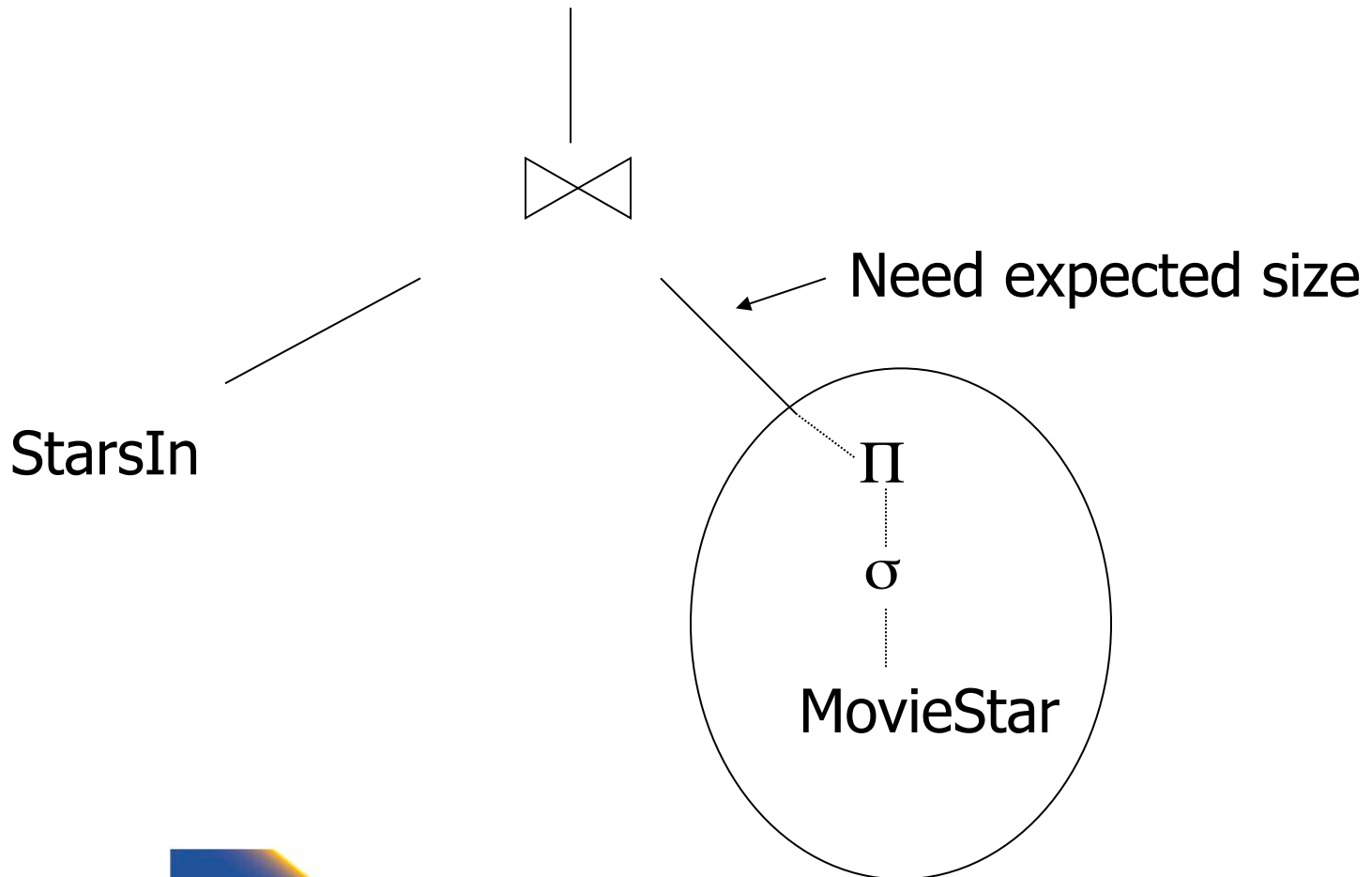
Example: One Physical Plan

Hash join → Parameters: join order, memory size, project attributes,...

SEQ scan

index scan → Parameters: Select Condition,...

StarsIn

MovieStar

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Estimate costs

Notes 7 - Query Processing

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
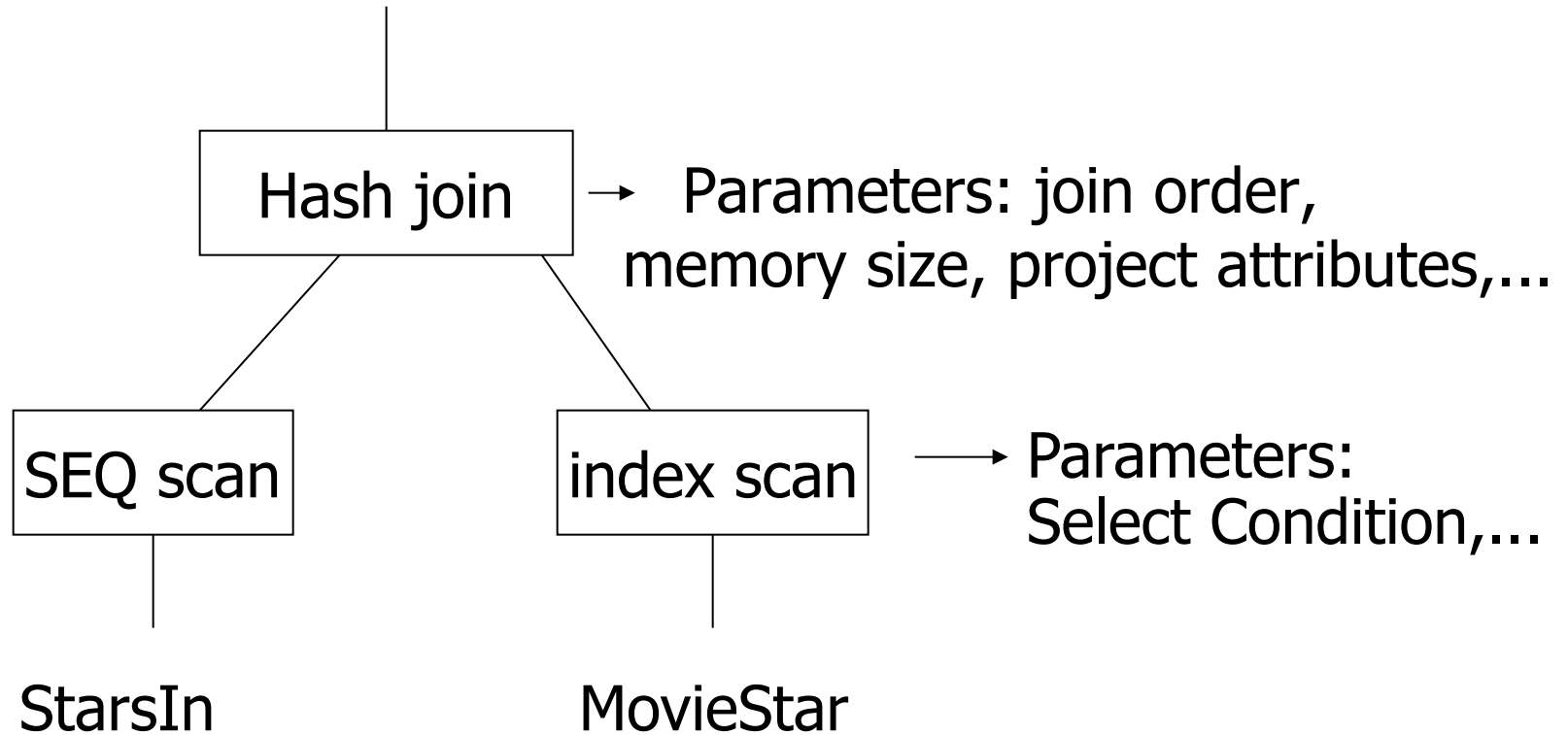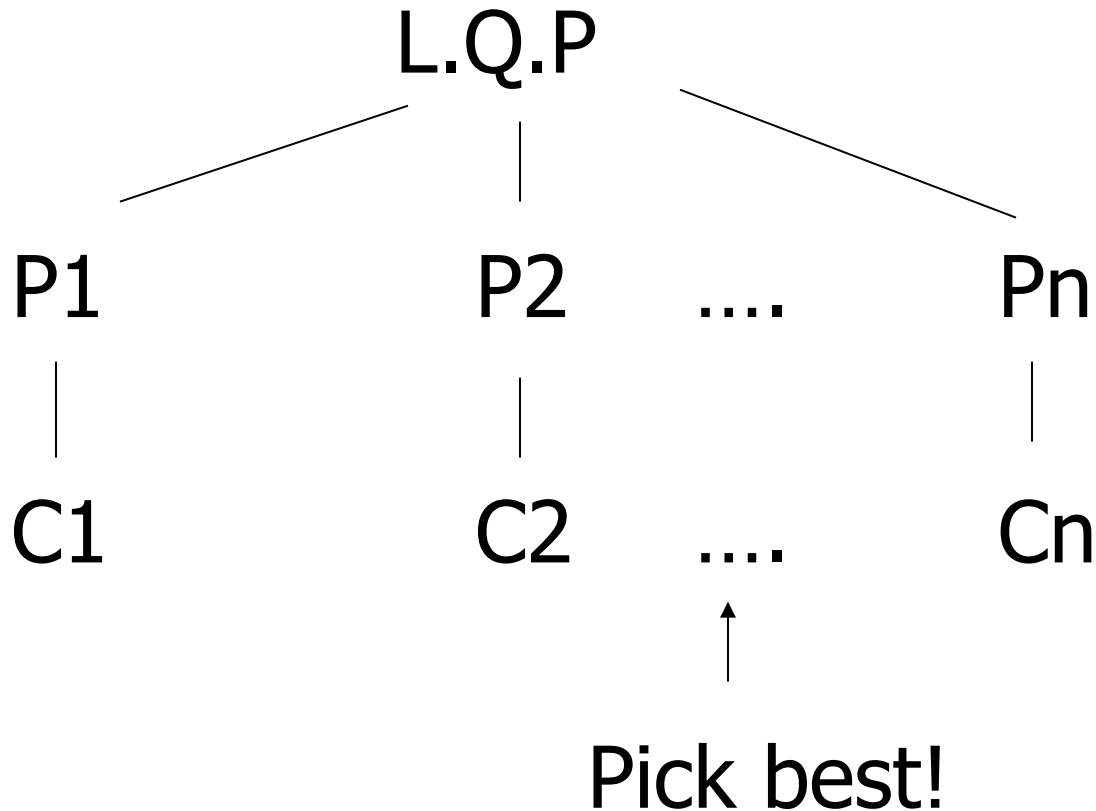
CS 525: Advanced Database Organisation

08: Query Processing Parsing and Analysis

Boris Glavic

Slides: adapted from a course taught by
Hector Garcia-Molina, Stanford InfoLab

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Notes 8 - Parsing and Analysis
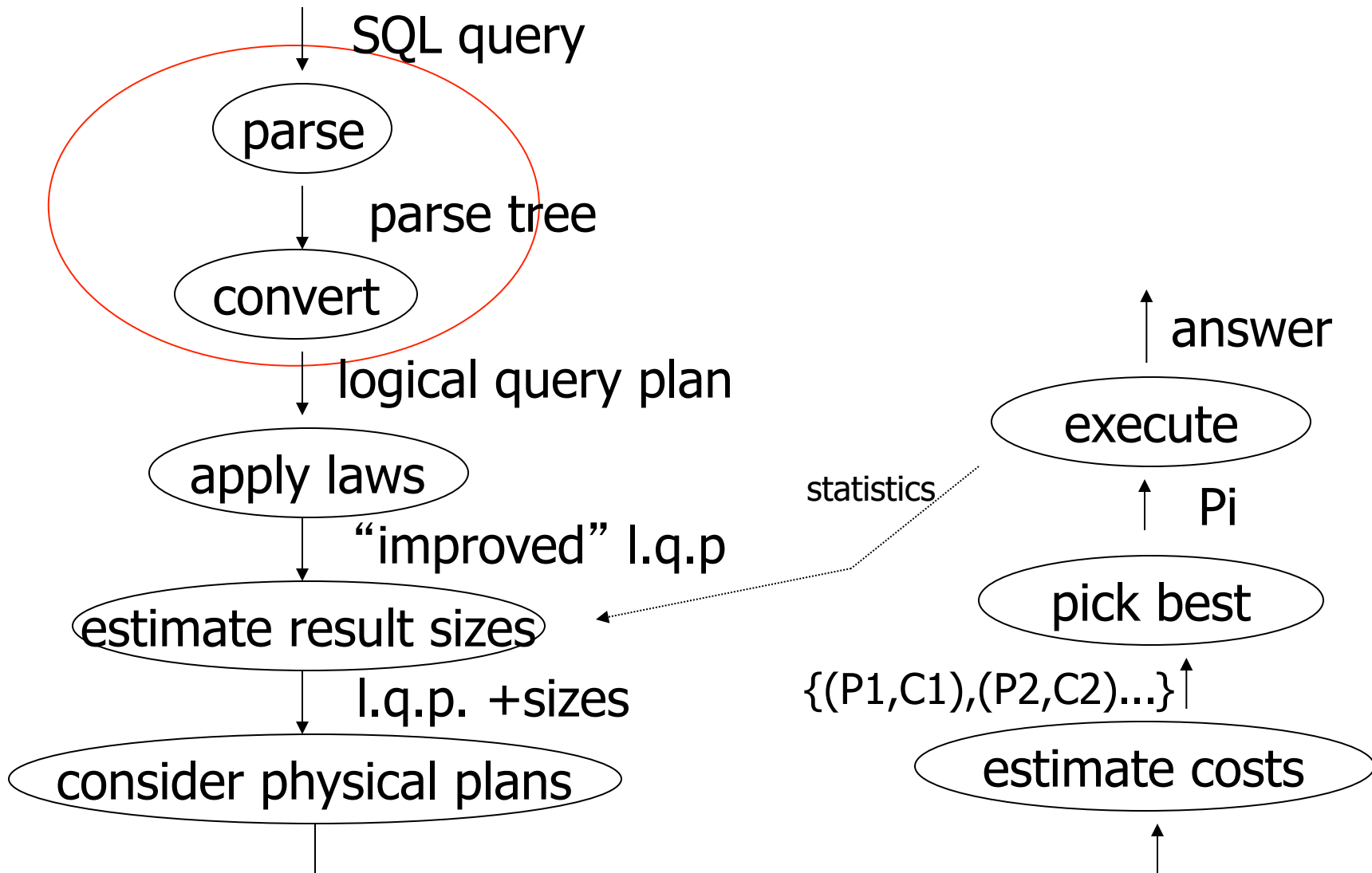
Parsing, Analysis, Conversion

1. ## Parsing
   – Transform SQL text into syntax tree

2. ## Analysis
   – Check for semantic correctness
   – Use database catalog
   – E.g., unfold views, lookup functions and attributes, check scopes

3. ## Conversion
   – Transform into internal representation
   – Relational algebra or QBM

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Analysis and Conversion

- Usually intertwined
- The internal representation is used to store analysis information
- Create an initial representation and complete during analysis

COMPUTER SCIENCE

Notes 8 - Parsing and Analysis

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Parsing, Analysis, Conversion

1. Parsing
2. Analysis
3. Conversion

CS 525

Notes 8 - Parsing and Analysis

5

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Parsing

- SQL -> Parse Tree
- Covered in compiler courses and books
- Here only short overview

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

SQL Standard

- Standardized language
  - 86, 89, 92, 99, 03, 06, 08, 11
- DBMS vendors developed their own dialects

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: SQL query

```
SELECT title
FROM StarsIn
WHERE starName IN (
      SELECT name
      FROM MovieStar
      WHERE birthdate LIKE '%1960'
);
```

(Find the movies with stars born in 1960)

Example: Parse Tree
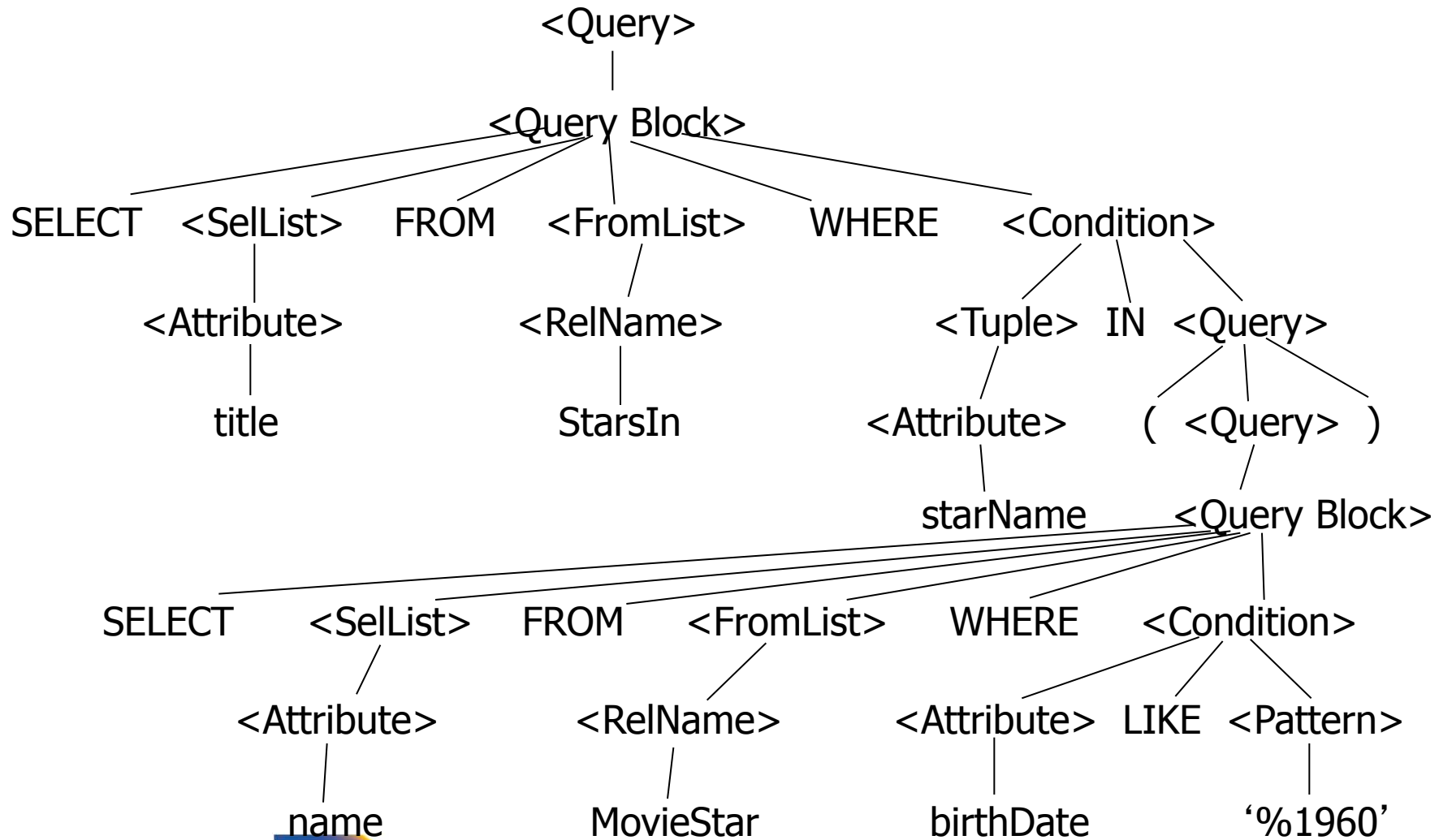
SQL Query Structure

- Organized in Query blocks

SELECT <select_list>

FROM <from_list>

WHERE <where_condition>

GROUP BY <group_by_expressions>

HAVING <having_condition>

ORDER BY <order_by_expressions>

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Query Blocks

- Only **SELECT** clause is mandatory
  - Some DBMS require **FROM**

SELECT (1 + 2) AS result

result
3

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

SELECT clause

- List of expressions and optional name assignment + optional **DISTINCT**
  - Attribute references: R.a, b
  - Constants: 1, 'hello', '2008-01-20'
  - Operators: (R.a + 3) * 2
  - Functions (maybe UDF): substr(R.a, 1,3)
    - Single result or **set functions**
  - Renaming: (R.a + 2) AS x

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

SELECT clause - example

```
SELECT substring(p.name,1,1) AS initial
        p.name
FROM person p
```

person

name	gender
Joe	male
Jim	male

result

initial	name
J	Joe
J	Jim

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

SELECT clause – set functions

- Function `extrChar(string)`

result

n
J
o
e
J
i
m

SELECT `extrChar(p.name) AS n`

FROM `person p`

person

name	gender
Joe	male
Jim	male

Notes 8 - Parsing and Analysis   14

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

SELECT clause – DISTINCT

SELECT DISTINCT gender
FROM person p

person

name	gender
Joe	male
Jim	male

result

gender
male

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

FROM clause

- List of table expressions
  - Access to relations
  - Subqueries (need alias)
  - Join expressions
  - Table functions
  - Renaming of relations and columns

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

FROM clause examples

FROM R

      -access table R

FROM R, S

      -access tables R and S

FROM R JOIN S ON (R.a = S.b)

      -join tables R and S on condition (R.a = S.b)

FROM R x

FROM R AS x

      -Access table R and assign alias 'x'

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

FROM clause examples

```
FROM R x(c,d)
FROM R AS x(c,d)
```
     -using aliases x for R and c,d for its attribues
```
FROM (R JOIN S t ON (R.a = t.b)), T
```
     -join R and S, and access T
```
FROM (R JOIN S ON (R.a = S.b)) JOIN T
```
     -join tables R and S and result with T
```
FROM create_sequence(1,100) AS seq(a)
```
     -call table function

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

FROM clause examples

FROM

      (SELECT count(*) FROM employee)
      AS empcnt(cnt)

-count number of employee in subquery

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

FROM clause examples

```
SELECT *
FROM create_sequence(1,3) AS seq(a)
```

result

a
1
2
3

FROM clause examples

```
SELECT dep, headcnt
FROM (SELECT count(*) AS headcnt, dep
        FROM employee
        GROUP BY dep)
WHERE headcnt > 100
```

employee

name	dep
Joe	IT
Jim	Marketing
…	…

result

dep	headcnt
IT	103
Support	2506
…	…

COMPUTER SCIENCE

Notes 8 - Parsing and Analysis

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

FROM clause - correlation

- Correlation
  - Reference attributes from other FROM clause item
  - Attributes of ith entry only available in $j > i$
  - Semantics:
    - For each row in result of ith entry:
    - Substitute correlated attributes with value from current row and evaluate query

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Correlation - Example

```
SELECT name, chr
FROM employee AS e,
     extrChar(e.name) AS c(chr)
```

result

name	chr
Joe	J
Joe	o
Joe	e
Jim	J
Jim	i
...	...

employee

name	dep
Joe	IT
Jim	Marketing
...	...

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Correlation - Example

```
SELECT name
FROM (SELECT max(salary) maxsal
        FROM employee) AS m,
      (SELECT name
       FROM employee x
       WHERE x.salary = m.maxsal) AS e
```

employee

name	salary
Joe	20,000
Jim	30,000
...	...

result

name
Jim

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

WHERE clause

- A condition
  - Attribute references
  - Constants
  - Operators (boolean)
  - Functions
  - Nested subquery expressions
- Result has to be boolean

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

WHERE clause examples

```
WHERE R.a = 3
```
-comparison between attribute and constant
```
WHERE (R.a > 5) AND (R.a < 10)
```
-range query using boolean AND
```
WHERE R.a = S.b
```
-comparison between two attributes
```
WHERE (R.a * 2) > (S.b − 3)
```
-using operators

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Nested Subqueries

- Nesting a query within an expression
- Correlation allowed
  - Access FROM clause attributes
- Different types of nesting
  - Scalar subquery
  - Existential quantification
  - Universal quantification

Nested Subqueries Semantics

- For each tuple produced by the FROM clause execute the subquery
  - If correlated attributes replace them with tuple values

Scalar subquery

- Subquery that returns one result tuple
  - How to check?
  - -> Runtime error

```
SELECT *
FROM R
WHERE R.a = (SELECT count(*) FROM S)
```

Existential Quantification

- `<expr> IN <subquery>`
  - Evaluates to true if <expr> equal to at least one of the results of the subquery

```
SELECT *
FROM users
WHERE name IN (SELECT name FROM
                blacklist)
```

Notes 8 - Parsing and Analysis

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Existential Quantification

- EXISTS `<subquery>`
  - Evaluates to true if `<subquery>` returns at least one tuple

```
SELECT *
FROM users u
WHERE EXISTS (SELECT * FROM
                  blacklist b
            WHERE b.name = u.name)
```

Existential Quantification

- `<expr> <op> ANY <subquery>`

  - Evaluates to true if `<expr> <op> <tuple>` evaluates to true for **at least one** result tuple

  - Op is any comparison operator: =, <, >, …

```
SELECT *
FROM users
WHERE name = ANY (SELECT name FROM
                  blacklist)
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Universal Quantification

- `<expr> <op> ALL <subquery>`

  - Evaluates to true if <expr> <op> <tuple> evaluates to true for **all** result tuples

  - Op is any comparison operator: =, <, >, …

```
SELECT *
FROM nation
WHERE nname = ALL (SELECT nation FROM
                        blacklist)
```

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Nested Subqueries Example

```
SELECT dep,name
FROM employee e
WHERE salary >= ALL (SELECT salary
                     FROM employee d
                     WHERE e.dep = d.dep)
```

employee

name	dep	salary
Joe	IT	2000
Jim	IT	300
Bob	HR	100
Alice	HR	10000
Patrice	HR	10000

result

dep	Name
IT	Joe
HR	Alice
HR	Patrice

Notes 8 - Parsing and Analysis

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

GROUP BY clause

- A list of expressions
  - Same as WHERE
  - No restriction to boolean
  - DBMS has to know how to compare = for data type

- Results are grouped by values of the expressions

- -> usually used for aggregation

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

GROUP BY restrictions

- If group-by is used then
  - SELECT clause can only use group by expressions or aggregation functions

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

GROUP BY clause examples

```
GROUP BY R.a
```
        -group on single attribute
```
GROUP BY (1+2)
```
        -allowed but useless (single group)
```
GROUP BY salary / 1000
```
        -groups of salary values in buckets of 1000
```
GROUP BY R.a, R.b
```
        -group on two attributes

```
SELECT count(*) AS numP,
     (SELECT count(*)
      FROM friends o
      WHERE o.with = f.name) AS numF
FROM (SELECT DISTINCT name FROM friends) f
GROUP BY (SELECT count(*)
              FROM friends o
              WHERE o.with = f.name)
```

friends

name	with
Joe	Jim
Joe	Peter
Jim	Joe
Jim	Peter
Peter	Joe

result

numP	numF
1	1
2	2

Notes 8 - Parsing and Analysis

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

HAVING clause

- A boolean expression
- Applied after grouping and aggregation
  - Only references aggregation expressions and group by expressions

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

HAVING clause examples

…

HAVING `sum(R.a) > 100`

     -only return tuples with sum bigger than 100

…

GROUP BY dep

HAVING dep = 'IT' AND sum(salary) > 1000000

     -only return group 'IT' and sum threshold

Notes 8 - Parsing and Analysis

ORDER BY clause

- A list of expressions
- Semantics: Order the result on these expressions

ORDER BY clause examples

```
ORDER BY R.a ASC
ORDER BY R.a
```
-order ascending on R.a
```
ORDER BY R.a DESC
```
-order descending on R.a
```
ORDER BY salary + bonus
```
-order by sum of salary and bonus

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

New and Non-standard SQL features (excerpt)

- LIMIT / OFFSET
  - Only return a fix maximum number of rows
  - FETCH FIRST n ROWS ONLY (DB2)
  - row_number() (Oracle)
- Window functions
  - More flexible grouping
  - Return both aggregated results and input values

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Parsing, Analysis, Conversion

1. Parsing

2. Analysis

3. Conversion

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Analysis Goals

- Semantic checks
  - Table column exists
  - Operator, function exists
  - Determine type casts
  - Scope checks
- Rewriting
  - Unfolding views

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Semantic checks

```
SELECT *
FROM R
WHERE R.a + 3 > 5
```

- Table R exists?
- Expand *: which attributes in R?
- R.a is a column?
- Type of constants 3, 5?
- Operator + for types of R.a and 3 exists?
- Operator > for types of result of + and 5 exists?

Notes 8 - Parsing and Analysis

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Database Catalog

- Stores information about database objects

- Aliases:
  - Information Schema
  - System tables
  - Data Dictionary

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Typical Catalog Information

- Tables
  - Name, attributes + data types, constraints
- Schema, DB
  - Hierarchical structuring of data
- Data types
  - Comparison operators
  - physical representation
  - Functions to (de)serialize to string

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Typical Catalog Information

- Functions (including aggregate/set)
  - Build-in
  - User defined (UDF)
- Triggers
- Stored Procedures
- ...

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Type Casts

- Similar to automatic type conversion in programming languages

- Expression: R.a + 3.0

  - Say R.a is of type integer

    - Search for a function +(int,float)

  - Does not exist?

    - Try to find a way to cast R.a, 3.0 or both to new data type

    - So that a function + exists for new types

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Scope checks

- Check that references are in correct scope

- E.g., if GROUP BY is present then SELECT clause expression can only reference group by expressions or aggregated values

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

View Unfolding

- SQL allows for stored queries using CREATE VIEW

- Afterwards a view can be used in queries

- If view is not materialized, then need to replace view with its definition

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

View Unfolding Example

CREATE VIEW totalSalary AS

SELECT name, salary + bonus AS total

FROM employee

SELECT *

FROM totalSalary

WHERE total > 10000

View Unfolding Example

```
CREATE VIEW totalSalary AS
SELECT name, salary + bonus AS total
FROM employee


SELECT *
FROM (SELECT name,
             salary + bonus AS total
      FROM employee) AS totalSalary
WHERE total > 10000
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Analysis Summary

- Perform semantic checks
  - Catalog lookups (tables, functions, types)
  - Scope checks
- View unfolding
- Generate internal representation during analysis

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Parsing, Analysis, Conversion

1. Parsing
2. Analysis
3. Conversion

Conversion

- Create an internal representation
  - Should be useful for analysis
  - Should be useful optimization

- Internal representation
  - Relational algebra
  - Query tree/graph models
    - E.g., QGM (Query Graph Model) in Starburst

Notes 8 - Parsing and Analysis

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Relational Alegbra

- Formal language
- Good for studying logical optimization and query equivalence (containment)
- Not informative enough for analysis
  - No datatype representation in algebra expressions
  - No meta-data

Other Internal Representations

- Practical implementations

  – Mostly following structure of SQL query blocks

  – Store data type and meta-data (where necessary)

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Canonical Translation to Relational Algebra

- TEXTBOOK version of conversion

- Given an SQL query

- Return an equivalent relational algebra expression

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Relational Algebra Recap

- Formal query language
- Consists of operators
  - Input(s): relation
  - Output: relation
  - -> Composable
- Set and Bag semantics version

- Relation Schema
  - A set of attribute name-datatype pairs
- Relation (instance)
  - A (multi-)set of tuples with the same schema
- Tuple
  - List of attribute value pairs (or function from attribute name to value)

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Set- vs. Bag semantics

- Set semantics:
  - Relations are Sets
  - Used in most theoretical work

- Bag semantics
  - Relations are Multi-Sets
    - Each element (tuple) can appear more than once
  - SQL uses bag semantics

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Bag semantics notation

- We use t^m to denote tuple t appears with multiplicity **m**

Notes 8 - Parsing and Analysis

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Set- vs. Bag semantics

Set

Name	Purchase
Peter	Guitar
Joe	Drum
Alice	Bass

Bag

Name	Purchase
Peter	Guitar
Peter	Guitar
Joe	Drum
Alice	Bass
Alice	Bass

Operators

- Selection
- Renaming
- Projection
- Joins
  - Theta, natural, cross-product, outer, anti
- Aggregation
- Duplicate removal
- Set operations

Notes 8 - Parsing and Analysis

Selection

- Syntax: $\sigma_c (R)$
  - R is input
  - C is a condition

- Semantics:
  - Return all tuples that match condition C
  - Set: { t | t ε R AND t fulfills C }
  - Bag: { t^n | $t^n \varepsilon$ R AND t fulfills C }

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Selection Example

- $\sigma_{a>5}$ (R)

R

a	b
1	13
3	12
6	14

Result

a	b
6	14

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Renaming

- Syntax: $\rho_A(R)$
  - R is input
  - A is list of attribute renamings $b \leftarrow a$
- Semantics:
  - Applies renaming from A to inputs
  - Set: $\{ t.A \mid t \in R \}$
  - Bag: $\{ (t.A)^n \mid t^n \in R \}$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Renaming Example

- $\rho_{c \leftarrow a} (R)$

R

a	b
1	13
3	12
6	14

Result

c	b
1	13
3	12
6	14

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Projection

– Syntax: $\Pi_A (R)$
  - R is input
  - A is list of projection expressions
  - Standard: only attributes in A

– Semantics:
  - Project all inputs on projection expressions
  - Set: $\{ t.A \mid t \in R \}$
  - Bag: $\{ (t.A)^n \mid t^n \in R \}$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Projection Example

- $\Pi_b\ (R)$

R

a	b
1	13
3	12
6	14

Result

b
13
12
14

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Cross Product

– Syntax: R X S

- R and S are inputs

– Semantics:

- All combinations of tuples from R and S
- = mathematical definition of cross product
- Set: { (t,s) | t εR AND sεS }
- Bag: { (t,s)^{n*m} | t$^n\varepsilon$R AND s$^m\varepsilon$S }

Notes 8 - Parsing and Analysis

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Cross Product Example

- R X S

R

a	b
1	13
3	12

S

c	d
a	5
b	3
c	4

Result

a	b	c	d
1	13	a	5
1	13	b	3
1	13	c	4
3	12	a	5
3	12	b	3
3	12	c	4

Notes 8 - Parsing and Analysis

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Join

- Syntax: $R \bowtie_C S$
  - R and S are inputs
  - C is a condition

- Semantics:
  - All combinations of tuples from R and S that match C
  - Set: $\{ (t,s) \mid t \in R \text{ AND } s \in S \text{ AND } (t,s) \text{ matches } C\}$
  - Bag: $\{ (t,s)^{n*m} \mid t^n \in R \text{ AND } s^m \in S \text{ AND } (t,s) \text{ matches } C\}$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Join Example

- $R \bowtie_{a=d} S$

R

a	b
1	13
3	12

S

c	d
a	5
b	3
c	4

Result

a	b	c	d
3	12	b	3

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Natural Join

- Syntax: R $\bowtie$ S
  - R and S are inputs

- Semantics:
  - All combinations of tuples from R and S that match on common attributes
  - A = common attributes of R and S
  - C = exclusive attributes of S
  - Set: { (t,s.C) | t εR AND sεS AND t.A=s.A}
  - Bag: { (t,s.C)^{n*m} | t$^n\varepsilon$R AND s$^m\varepsilon$S AND t.A=s.A}

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Natural Join Example

- R ⋈ S

R	
a	**b**
1	13
3	12

S	
c	**a**
a	5
b	3
c	4

Result		
a	**b**	**c**
3	12	b

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Left-outer Join

– Syntax: R $\bowtie_C$ S

- R and S are inputs
- C is condition

– Semantics:

- R join S
- t εR without match, fill S attributes with NULL

{ (t,s) | t εR AND sεS AND (t,s) matches C}

union

{ (t, NULL(S)) | t εR AND NOT exists sεS: (t,s) matches C }

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Left-outer Join Example

- $R \bowtie_{a=d} S$

R

a	b
1	13
3	12

S

c	d
a	5
b	3
c	4

Result

a	b	c	d
1	13	NULL	NULL
3	12	b	3

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Right-outer Join

– Syntax: R $\bowtie_C$ S

- R and S are inputs
- C is condition

– Semantics:

- R join S
- s εS without match, fill R attributes with NULL

{ (t,s) | t εR AND sεS AND (t,s) matches C}

union

{ (NULL(R),s) | s εS AND NOT exists tεR: (t,s) matches C }

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Right-outer Join Example

- R $\bowtie_{a=d}$ S

R

a	b
1	13
3	12

S

c	d
a	5
b	3
c	4

Result

a	b	c	d
NULL	NULL	a	5
3	12	b	3
NULL	NULL	c	4

Notes 8 - Parsing and Analysis

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Full-outer Join

– Syntax: $R \bowtie_C S$

  • R and S are inputs and C is condition

– Semantics:

{ (t,s) | t εR AND sεS AND (t,s) matches C}

union

{ (NULL(R),s) | s εS AND NOT exists tεR: (t,s) matches C }

union

{ (t, NULL(S)) | t εR AND NOT exists sεS: (t,s) matches C }

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Full-outer Join Example

- $R \bowtie_{a=d} S$

R

a	b
1	13
3	12

S

c	d
a	5
b	3
c	4

Result

a	b	c	d
1	13	NULL	NULL
NULL	NULL	a	5
3	12	b	3
NULL	NULL	c	4

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Semijoin

- Syntax: R ⋉ S and R ⋊ S
  - R and S are inputs

- Semantics:
  - All tuples from R that have a matching tuple from relation S on the common attributes A

  { t | t εR AND exists sεS: t.A = s.A}

Notes 8 - Parsing and Analysis

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Semijoin Example

- R ⋉ S

R	
a	**b**
1	13
3	12

S	
c	**a**
a	5
b	3
c	4

Result

a	**b**
3	12

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Antijoin

- – Syntax: R ▷ S
  - • R and S are inputs

- – Semantics:
  - • All tuples from R that have no matching tuple from relation S on the common attributes A

  { t | t εR AND NOT exists sεS: t.A = s.A}

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Antijoin Example

- R ▷ S

R

a	b
1	13
3	12

S

c	a
a	5
b	3
c	4

Result

a	b
1	13

COMPUTER
SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Aggregation

– Syntax: ${}_{G}\alpha_{A}\,(R)$
  - A is list of aggregation functions
  - G is list of group by attributes

– Semantics:
  - Build groups of tuples according G and compute the aggregation functions from each group
  - $\{\ (t.G,\ agg(G(t))\ |\ t\varepsilon R\ \}$
  - $G(t) = \{\ t'\ |\ t'\ \varepsilon R\ AND\ t'.G = t.G\ \}$

Notes 8 - Parsing and Analysis

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Aggregation Example

- $_b\alpha_{sum(a)}$ (R)

R

a	b
1	1
3	1
6	2
3	2

Result

sum(a)	b
4	1
9	2

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Duplicate Removal

– Syntax: $\delta(R)$

   • R is input

– Semantics:

   • Remove duplicates from input

   • Set: N/A

   • Bag: $\{ t^1 \mid t^n \varepsilon R \}$

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Duplicate Removal Example

- $\delta(R)$

R

a	b
1	13
1	13
6	14

Result

a	b
1	13
6	14

COMPUTER SCIENCE

Notes 8 - Parsing and Analysis

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Set operations

– Input: R and S
  - Have to have the same schema
    – Union compatible
  - Modulo attribute names
– Types
  - Union
  - Intersection
  - Set difference

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Union

– Syntax: $R \cup S$

  • R and S are union-compatible inputs

– Semantics:

  • Set: { (t) | t εR OR tεS}

  • Bag: { $(t,s)^{n+m}$ | $t^n\varepsilon$R AND $s^m\varepsilon$S }

    – Assumption t^n with n < 1 for tuple not in relation

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Union Example

- R ∪ S

R
a
1
3

S
b
1
2
3

Result
a
1
2
3
1
3

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Intersection

– Syntax: R ∩ S

  • R and S are union-compatible inputs

– Semantics:

  • Set: $\{ (t) \mid t \in R \text{ AND } t \in S\}$

  • Bag: $\{ (t,s)^{\min(n,m)} \mid t^n \in R \text{ AND } s^m \in S \}$

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Intersection Example

- R ∩ S

R		S		Result

R

a
1
3

S

b
1
2
3

Result

a
1
3

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Set Difference

- Syntax: R - S
  - R and S are union-compatible inputs

- Semantics:
  - Set: $\{ (t) \mid t \in R \text{ AND NOT } t \in S \}$
  - Bag: $\{ (t,s)^{n-m} \mid t^n \in R \text{ AND } s^m \in S \}$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Set Difference Example

- R - S

R

a
1
5

S

b
1
2
3

Result

a
5

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Canonical Translation to Relational Algebra

- TEXTBOOK version of conversion
- Given an SQL query
- Return an equivalent relational algebra expression

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Canonical Translation

- **FROM** clause into joins and cross-products
  - Cross-product between list items
  - Joins into their algebra counter-part
- **WHERE** clause into selection
- **SELECT** clause into projection and renaming
  - If it has aggregation functions use aggreation
  - **DISTINCT** into duplicate removal

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Canonical Translation

- **GROUP BY** clause into aggregation
- **HAVING** clause into selection
- **ORDER BY** – no counter-part

- Then turn joins into crossproducts and selections

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Set Operations

- **UNION ALL** into union
- **UNION** duplicate removal over union
- **INTERSECT ALL** into intersection
- **INTERSECT** add duplicate removal
- **EXCEPT ALL** into set difference
- **EXCEPT** apply duplicate removal to inputs and then apply set difference

Example: Relational Algebra Translation

```
SELECT sum(R.a)
FROM R
GROUP BY b
```

$$\Pi_{\text{sum}(a)}$$

$$_B\alpha_{\text{sum}(a)}$$

$$R$$

Notes 8 - Parsing and Analysis   104

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
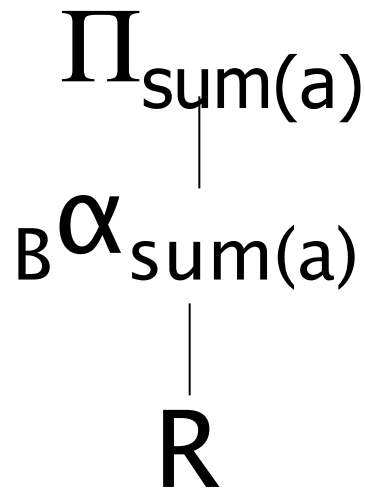
Example: Relational Algebra Translation

```
SELECT dep, headcnt
FROM (SELECT count(*) AS headcnt, dep
      FROM employee
      GROUP BY dep)
WHERE headcnt > 100
```

$$\Pi_{dep,\ headcnt}$$
$$|$$
$$\sigma_{headcnt > 100}$$
$$|$$
$$\rho_{headcnt \leftarrow count(*)}$$
$$|$$
$$_{dep}\alpha_{count(*)}$$
$$|$$
$$Employee$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
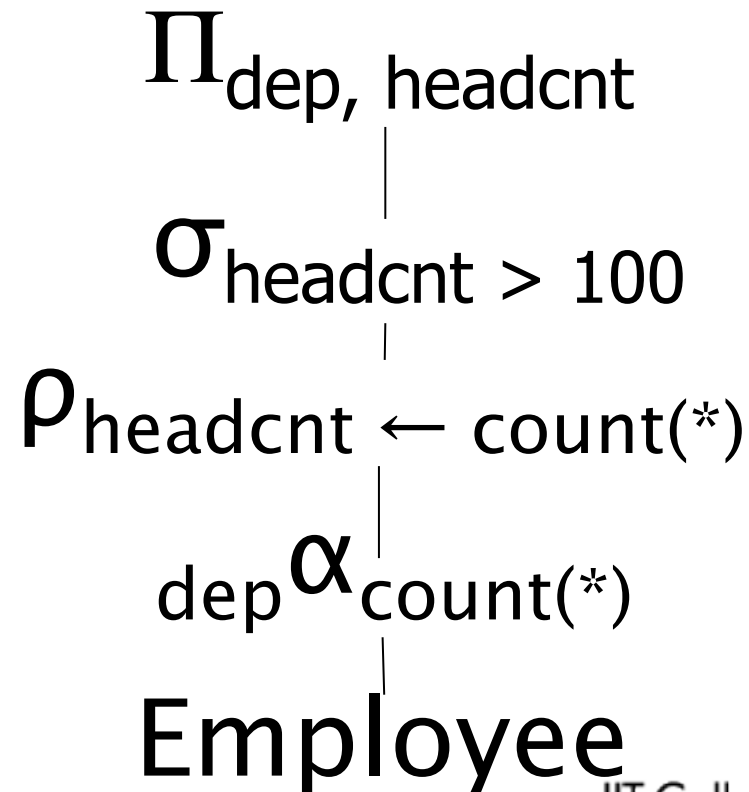
Example: Relational Algebra Translation

```
SELECT *
FROM R JOIN S ON (R.a = S.b)
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
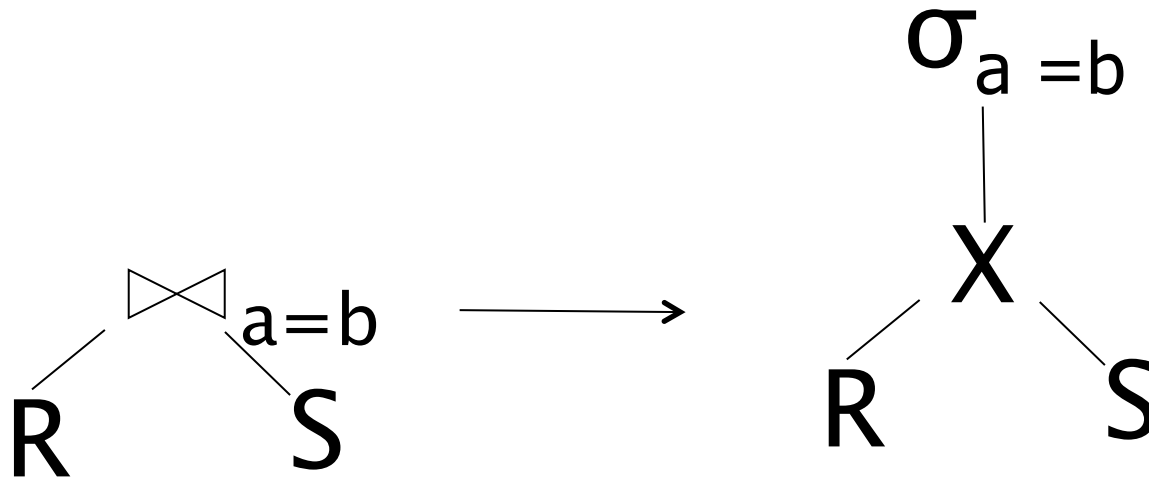
Parsing and Analysis Summary

- SQL text -> Internal representation
- Semantic checks
- Database catalog
- View unfolding

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

CS 525: Advanced Database Organisation

09: Query Optimization - Logical

Boris Glavic

Slides: adapted from a course taught by
Hector Garcia-Molina, Stanford InfoLab

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

SQL query

↓

parse

↓ parse tree

convert

↓ logical query plan

apply laws

↓ "improved" l.q.p

estimate result sizes

↓ l.q.p. +sizes

consider physical plans

{P1,P2,.....}

↑ answer

execute

↑ Pi

pick best

{(P1,C1),(P2,C2)...} ↑

estimate costs

statistics
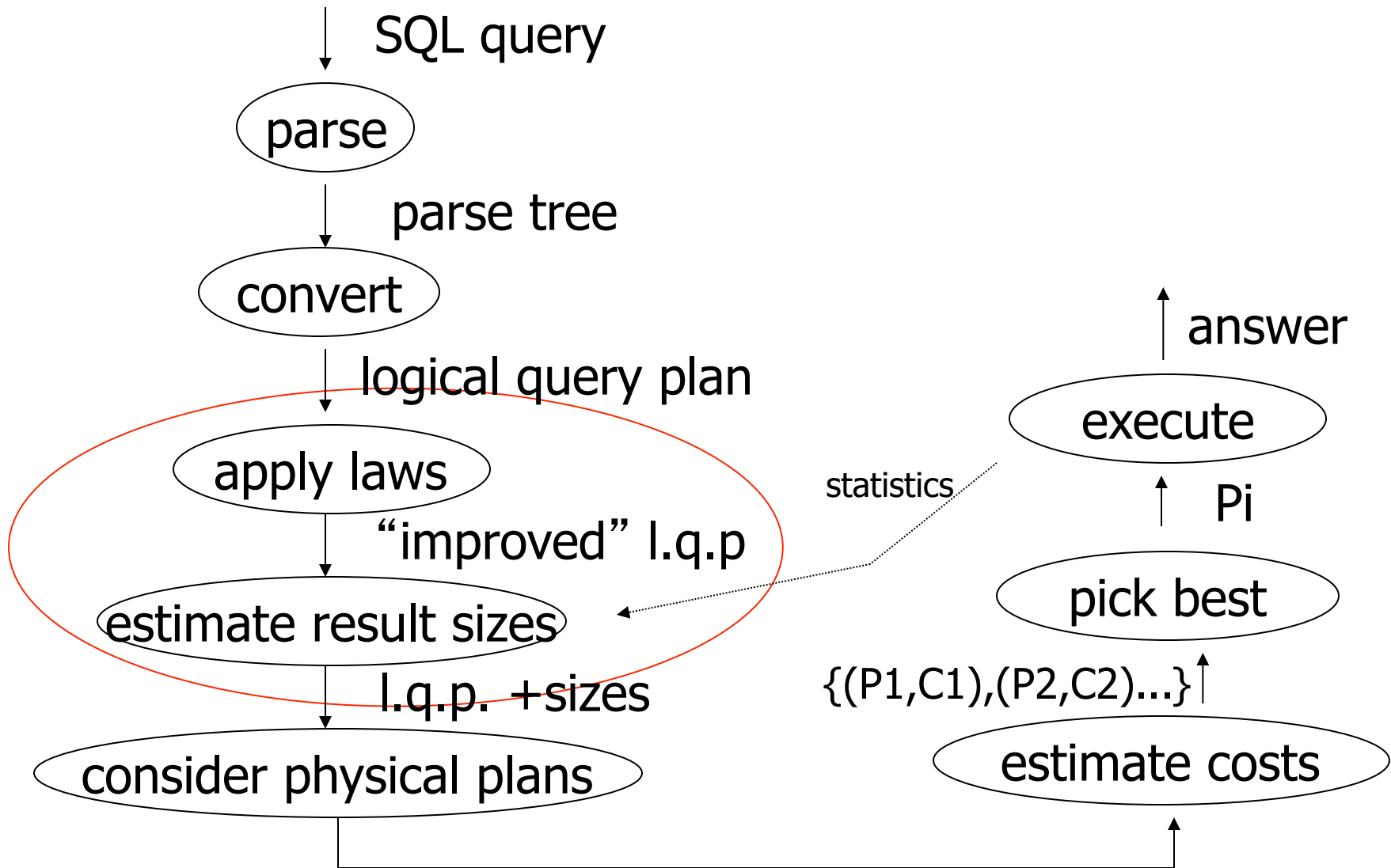
Query Optimization

- Relational algebra level
- Detailed query plan level

Query Optimization

- Relational algebra level

- Detailed query plan level

  - Estimate Costs
    - without indexes
    - with indexes
  - Generate and compare plans

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Relational algebra optimization

- Transformation rules

  (preserve equivalence)

- What are good transformations?
  - Heuristic application of transformations

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Query Equivalence

- Two queries q and q' are equivalent:
  - If for every database instance I
    - Contents of all the tables
  - Both queries have the same result

  $q \equiv q'$ iff $\forall I: q(I) = q'(I)$

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: Natural joins & cross products & union

$$R \bowtie S \quad = \quad S \bowtie R$$

$$(R \bowtie S) \bowtie T \quad = R \bowtie (S \bowtie T)$$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Note:

- Carry attribute names in results, so order is not important

- Can also write as trees, e.g.:
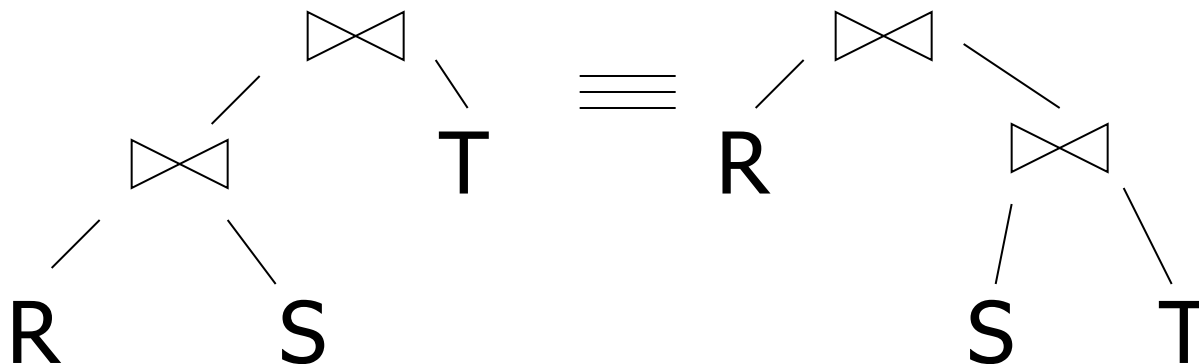
Rules: Natural joins & cross products & union

$R \bowtie S = S \bowtie R$

$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

$R \times S = S \times R$

$(R \times S) \times T = R \times (S \times T)$

$R \cup S = S \cup R$

$R \cup (S \cup T) = (R \cup S) \cup T$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: Selects

$$\sigma_{p1 \wedge p2}(R) =$$

$$\sigma_{p1 \vee p2}(R) =$$

CS 525

COMPUTER SCIENCE

Notes 9 - Logical Optimization   10

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: Selects

$$\sigma_{p1 \wedge p2}(R) = \sigma_{p1}\ [\ \sigma_{p2}\ (R)]$$

$$\sigma_{p1 \vee p2}(R) = [\ \sigma_{p1}\ (R)]\ \cup\ [\ \sigma_{p2}\ (R)]$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Bags vs. Sets

R = {a,a,b,b,b,c}

S = {b,b,c,c,d}

RUS = ?

Bags vs. Sets

R = {a,a,b,b,b,c}

S = {b,b,c,c,d}

RUS = ?

- <u>Option 1</u>   SUM

  RUS = {a,a,b,b,b,b,b,c,c,c,d}

- <u>Option 2</u>   MAX

  RUS = {a,a,b,b,b,c,c,d}

Notes 9 - Logical Optimization   13

Option 2 (MAX) makes this rule work:

$$\sigma_{p1 \lor p2}(R) = \sigma_{p1}(R) \ \cup \ \sigma_{p2}(R)$$

Example: R={a,a,b,b,b,c}

P1 satisfied by a,b; P2 satisfied by b,c

Notes 9 - Logical Optimization

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Option 2 (MAX) makes this rule work:

$$\sigma_{p1 \textbf{v} p2} (R) = \sigma_{p1}(R) \ \cup \ \sigma_{p2}(R)$$

Example: R={a,a,b,b,b,c}

P1 satisfied by a,b;   P2 satisfied by b,c

$$\sigma_{p1 \textbf{v} p2} (R) = \{a,a,b,b,b,c\}$$

$$\sigma_{p1}(R) = \{a,a,b,b,b\}$$

$$\sigma_{p2}(R) = \{b,b,b,c\}$$

$$\sigma_{p1}(R) \cup \sigma_{p2} (R) = \{a,a,b,b,b,c\}$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

"Sum" option makes more sense:

Senators (......)   Rep (......)

T1 = $\pi_{yr,state}$ Senators;   T2 = $\pi_{yr,state}$ Reps

T1

Yr	State
97	CA
99	CA
98	AZ

T2

Yr	State
99	CA
99	CA
98	CA

Union?

Executive Decision

-> Use "SUM" option for bag unions

-> Some rules cannot be used for bags

COMPUTER SCIENCE

Notes 9 - Logical Optimization

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: Project

Let: X = set of attributes

$\qquad$ Y = set of attributes

$\qquad$ XY = X U Y

$\pi_{xy} (R) =$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: Project

Let: X = set of attributes

      Y = set of attributes

      XY = X U Y

$$\pi_{xy}(R) = \pi_x[\pi_y(R)]$$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: Project

Let: X = set of attributes

Y = set of attributes

XY = X ∪ Y

$$\pi_{xy}(R) = \pi_x[\pi_y(R)]$$

Notes 9 - Logical Optimization

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: $\sigma + \bowtie$ combined

Let p = predicate with only R attribs

q = predicate with only S attribs

m = predicate with only R,S attribs

$$\sigma_p (R \bowtie S) =$$

$$\sigma_q (R \bowtie S) =$$

Rules: $\sigma + \bowtie$ combined

Let p = predicate with only R attribs

    q = predicate with only S attribs

    m = predicate with only R,S attribs

$$\sigma_p (R \bowtie S) = [\sigma_p (R)] \bowtie S$$

$$\sigma_q (R \bowtie S) = R \bowtie [\sigma_q (S)]$$

Notes 9 - Logical Optimization

Rules: $\sigma + \bowtie$ combined (continued)

Some Rules can be Derived:

$$\sigma_{p \wedge q} (R \bowtie S) =$$

$$\sigma_{p \wedge q \wedge m} (R \bowtie S) =$$

$$\sigma_{p \vee q} (R \bowtie S) =$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Do one:

$$\sigma_{p \wedge q} (R \bowtie S) = [\sigma_p (R)] \bowtie [\sigma_q (S)]$$

$$\sigma_{p \wedge q \wedge m} (R \bowtie S) =$$
$$\sigma_m \left[(\sigma_p R) \bowtie (\sigma_q S) \right]$$

$$\sigma_{p \vee q} (R \bowtie S) =$$
$$\left[(\sigma_p R) \bowtie S \right] \cup \left[R \bowtie (\sigma_q S) \right]$$

Notes 9 - Logical Optimization   24

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

--> Derivation for first one:

$$\sigma_{p \wedge q} (R \bowtie S) =$$

$$\sigma_p \left[\sigma_q (R \bowtie S) \right] =$$

$$\sigma_p \left[R \bowtie \sigma_q (S) \right] =$$

$$\left[\sigma_p (R) \right] \bowtie \left[\sigma_q (S) \right]$$

Notes 9 - Logical Optimization

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Rules: π,σ combined

Let x = subset of R attributes

z = attributes in predicate P

(subset of R attributes)

$$\pi_x[\sigma_{p\,(R)}] =$$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Rules: π, σ combined

Let x = subset of R attributes

z = attributes in predicate P

(subset of R attributes)

$$\pi_x[\sigma_{p\ (R)}\ ] = \{\sigma_p\ [\ \pi_x\ _{(R)}\ ]\}$$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: π, σ combined

Let x = subset of R attributes

     z = attributes in predicate P

            (subset of R attributes)

$$\pi_x[\sigma_{p\ (R)}] = \pi_x \left\{ \sigma_p \left[\overset{\pi_{xz}}{\cancel{\pi_x}}\ (R) \right] \right\}$$

Rules: π, $\bowtie$ combined

Let   x = subset of R attributes

y = subset of S attributes

z = intersection of R,S attributes

$\pi_{xy} (R \bowtie S) =$

Rules: $\pi, \bowtie$ combined

Let x = subset of R attributes

    y = subset of S attributes

    z = intersection of R,S attributes

$\pi_{xy} (R \bowtie S) =$

$$\pi_{xy}\{[\pi_{xz} (R)] \bowtie [\pi_{yz} (S)]\}$$

Notes 9 - Logical Optimization

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

$$\pi_{xy} \{\sigma_p\ (R \bowtie S)\} \quad =$$

COMPUTER
SCIENCE

$$\pi_{xy} \{\sigma_p \ (R \bowtie S)\} \ =$$

$$\pi_{xy} \{\sigma_p \ [\pi_{xz'} \ (R) \bowtie \pi_{yz'} \ (S)]\}$$

$$z' \ = z \cup \{\text{attributes used in P }\}$$

Rules for σ, π combined with X

similar...

e.g., $\sigma_p (R \times S) = ?$

Notes 9 - Logical Optimization

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Rules σ, U combined:

$$\sigma_p(R \cup S) = \sigma_p(R) \cup \sigma_p(S)$$

$$\sigma_p(R - S) = \sigma_p(R) - S = \sigma_p(R) - \sigma_p(S)$$

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Which are "good" transformations?

☐ $\sigma_{p1 \wedge p2}(R) \rightarrow \sigma_{p1}[\sigma_{p2}(R)]$

☐ $\sigma_p(R \bowtie S) \rightarrow [\sigma_p(R)] \bowtie S$

☐ $R \bowtie S \rightarrow S \bowtie R$

☐ $\pi_x[\sigma_p(R)] \rightarrow \pi_x\{\sigma_p[\pi_{xz}(R)]\}$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Conventional wisdom:
do projects early

<u>Example</u>: R(A,B,C,D,E)   x={E}

P: (A=3) ∧ (B="cat")

$$\pi_x \{\sigma_p (R)\} \quad \text{vs.} \quad \pi_E \{\sigma_p \{\pi_{ABE}(R)\}\}$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

But What if we have A, B indexes?

B = "cat" → A=3 ←
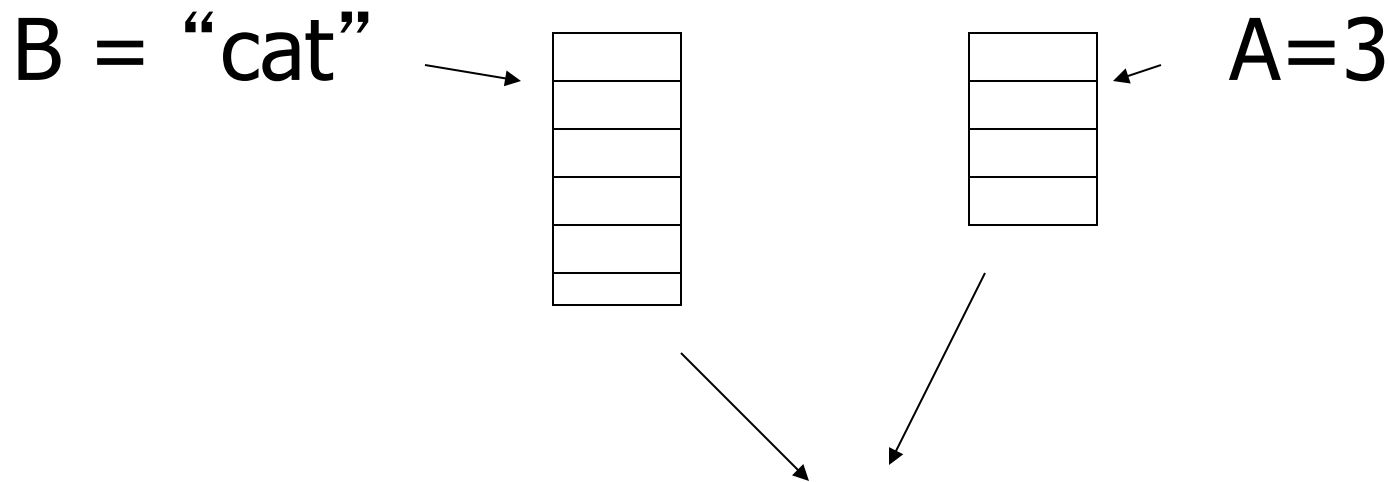
Intersect pointers to get
pointers to matching tuples
e.g., using bitmaps

Bottom line:

- No transformation is <u>always</u> good
- Usually good: early selections
  - Exception: expensive selection conditions
  - E.g., UDFs

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

More transformations

- Eliminate common sub-expressions
- Detect constant expressions
- Other operations: duplicate elimination

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Pushing Selections

- Idea:

  - Join conditions equate attributes
  - For parts of algebra tree (scope) store which attributes have to be the same
    - Called Equivalence classes

- Example: R(a,b), S(c,d)

$$\sigma_{b=3} (R \bowtie_{b=c} S) = \sigma_{b=3} (R) \bowtie_{b=c} \sigma_{c=3} (S)$$

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Outer-Joins

- Not commutative
  - $R \bowtie S \neq S \bowtie R$
- p – condition over attributes in A
- A list of attributes from R

$$\sigma_p (R \bowtie_{A=B} S) \equiv \sigma_p (R) \bowtie_{A=B} S$$

$$\text{Not } \sigma_p (R \bowtie_{A=B} S) \equiv R \bowtie_{A=B} \sigma_p (S)$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Summary Equivalences

- Associativity: $(R \odot S) \odot T \equiv R \odot (S \odot T)$
- Commutativity: $R \odot S \equiv S \odot R$
- Distributivity: $(R \odot S) \otimes T \equiv (R \otimes T) \odot (S \otimes T)$
- Difference between Set and Bag Equivalences
- Only some equivalence are useful

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Outline - Query Processing

- Relational algebra level
  - transformations
  - good transformations
- Detailed query plan level
  - estimate costs
  - generate and compare plans

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- Estimating cost of query plan

(1) Estimating <u>size</u> of results
(2) Estimating # of IOs

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Estimating result size

- Keep statistics for relation R
  - T(R) : # tuples in R
  - S(R) : # of bytes in each R tuple
  - B(R): # of blocks to hold all R tuples
  - V(R, A) : # distinct values in R
                for attribute A

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

A: 20 byte string

B: 4 byte integer

C: 8 byte date

D: 5 byte string

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

A: 20 byte string

B: 4 byte integer

C: 8 byte date

D: 5 byte string

$T(R) = 5$ $S(R) = 37$

$V(R,A) = 3$ $V(R,C) = 5$

$V(R,B) = 1$ $V(R,D) = 4$

Size estimates for W = R1 x R2

T(W) =

S(W) =

COMPUTER
SCIENCE

Notes 9 - Logical Optimization   48

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Size estimates for W = R1 x R2

$T(W) = T(R1) \times T(R2)$

$S(W) = S(R1) + S(R2)$

COMPUTER
SCIENCE

Notes 9 - Logical Optimization

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Size estimate for $W = \sigma_{A=a}(R)$

$S(W) = S(R)$

$T(W) = ?$

Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

$V(R,A)=3$

$V(R,B)=1$

$V(R,C)=5$

$V(R,D)=4$

$W = \sigma_{z=val}(R)$ $T(W) =$

Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

$V(R,A)=3$
$V(R,B)=1$
$V(R,C)=5$
$V(R,D)=4$

$$W = \sigma_{z=val}(R) \qquad T(W) = \frac{T(R)}{V(R,Z)}$$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Assumption:

Values in select expression Z = val
are <u>uniformly distributed</u>
over possible V(R,Z) values.

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Alternate Assumption:

Values in select expression $Z = val$
are <u>uniformly distributed</u>
over domain with DOM(R,Z) values.

Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

Alternate assumption
V(R,A)=3 DOM(R,A)=10
V(R,B)=1 DOM(R,B)=10
V(R,C)=5 DOM(R,C)=10
V(R,D)=4 DOM(R,D)=10

$$W = \sigma_{z=val}(R) \qquad T(W) = ?$$

Notes 9 - Logical Optimization

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

C=val $\Rightarrow$ T(W) = (1/10)1 + (1/10)1 + ...
$$= (5/10) = 0.5$$

B=val $\Rightarrow$ T(W)= (1/10)5 + 0 + 0 = 0.5

A=val $\Rightarrow$ T(W)= (1/10)2 + (1/10)2 + (1/10)1
$$= 0.5$$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

Alternate assumption
V(R,A)=3 DOM(R,A)=10
V(R,B)=1 DOM(R,B)=10
V(R,C)=5 DOM(R,C)=10
V(R,D)=4 DOM(R,D)=10

$$W = \sigma_{z=val}(R) \quad T(W) = \frac{T(R)}{DOM(R,Z)}$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Selection cardinality

SC(R,A) = average # records that satisfy
equality condition on R.A

$$SC(R,A) = \begin{cases} \dfrac{T(R)}{V(R,A)} \\[2em] \dfrac{T(R)}{DOM(R,A)} \end{cases}$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What about W = $\sigma_{z \geq val}(R)$?

T(W) = ?

What about W = $\sigma_{z \geq val}$ (R)   ?

T(W) = ?

- Solution # 1:
  T(W) = T(R)/2

Notes 9 - Logical Optimization

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What about W = $\sigma_{z \geq val}$ (R)   ?

T(W) = ?

- Solution # 1:
    T(W) = T(R)/2

- Solution # 2:
    T(W) = T(R)/3

Notes 9 - Logical Optimization

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

- Solution # 3:   Estimate values in range

Example R

$$\text{Min=1} \qquad V(R,Z)=10$$

$$W= \sigma_{Z \geq 15} (R)$$

Max=20

- Solution # 3:   Estimate values in range

Example R

	Z

Min=1 $V(R,Z)=10$

$W = \sigma_{Z \geq 15}(R)$

Max=20

$$f = \frac{20-15+1}{20-1+1} = \frac{6}{20} \quad \text{(fraction of range)}$$

$$T(W) = f \times T(R)$$

Equivalently:

$$f \times V(R,Z) = \text{fraction of distinct values}$$

$$T(W) = [f \times V(Z,R)] \times \frac{T(R)}{V(Z,R)} = f \times T(R)$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Size estimate for W = R1 ⋈ R2

Let x = attributes of R1

   y = attributes of R2

Size estimate for $W = R1 \bowtie R2$

Let x = attributes of R1

   y = attributes of R2

Case 1 $X \cap Y = \varnothing$

Same as R1 x R2

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Case 2 $W = R1 \bowtie R2$ $X \cap Y = A$

R1 | A | B | C   R2 | A | D

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

| Case 2 | $W = R1 \bowtie R2$ | $X \cap Y = A$ |

$R1$ | A | B | C $R2$ | A | D

Assumption:

$V(R1,A) \leq V(R2,A) \Rightarrow$ Every A value in R1 is in R2

$V(R2,A) \leq V(R1,A) \Rightarrow$ Every A value in R2 is in R1

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Computing T(W) when V(R1,A) ≤ V(R2,A)

R1 | A | B | C   R2 | A | D

Take
1 tuple

Match

Computing T(W) when V(R1,A) ≤ V(R2,A)

R1 | A | B | C   R2 | A | D

Take
1 tuple
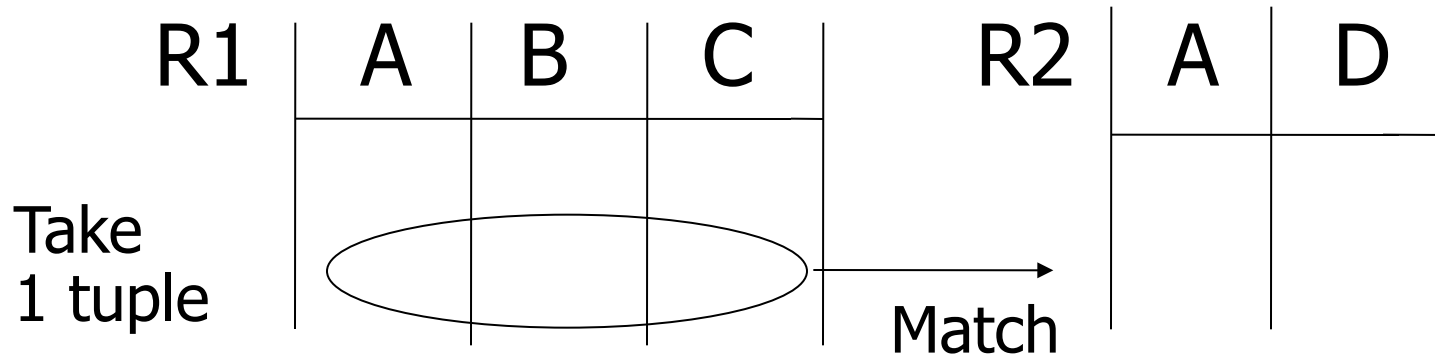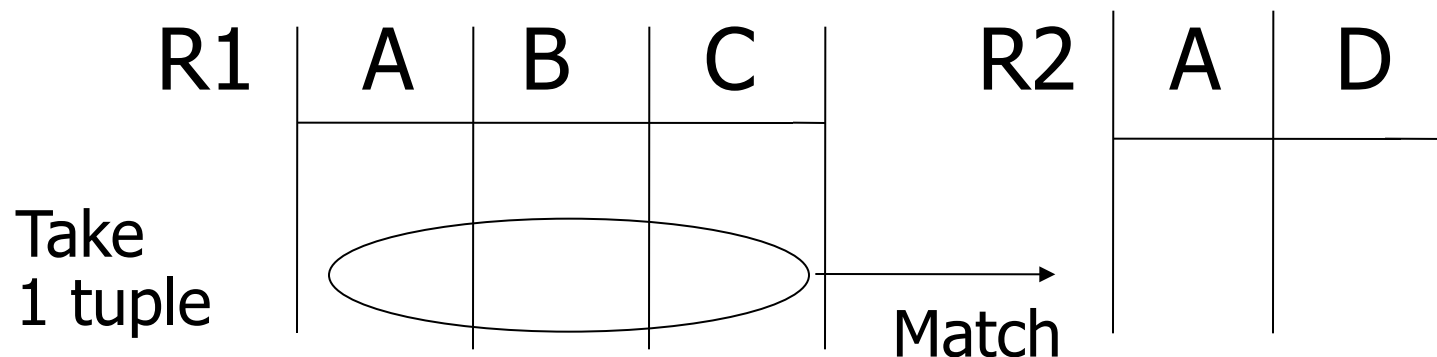
Match

1 tuple matches with $\dfrac{T(R2)}{V(R2,A)}$ tuples...

so $T(W) = \dfrac{T(R2)}{V(R2,\,A)} \times T(R1)$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- $V(R1,A) \leq V(R2,A) \quad T(W) = \dfrac{T(R2)\ T(R1)}{V(R2,A)}$

- $V(R2,A) \leq V(R1,A) \quad T(W) = \dfrac{T(R2)\ T(R1)}{V(R1,A)}$

[A is common attribute]

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

In general $\quad$ W = R1 $\bowtie$ R2

$$T(W) = \frac{T(R2)\, T(R1)}{\max\{\, V(R1,A),\, V(R2,A)\, \}}$$

COMPUTER
SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Case 2 with alternate assumption

Values uniformly distributed over domain

R1 | A | B | C   R2 | A | D
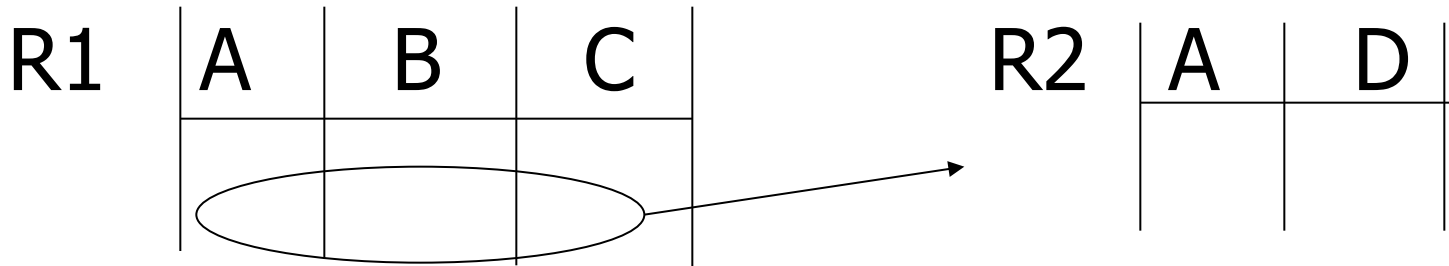
This tuple matches T(R2)/DOM(R2,A) so

$$T(W) = \frac{T(R2)\ T(R1)}{DOM(R2, A)} = \frac{T(R2)\ T(R1)}{DOM(R1, A)}$$

Assume the same

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

In all cases:

$$S(W) = S(R1) + S(R2) - S(A)$$

size of attribute A

Notes 9 - Logical Optimization

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Using similar ideas,
we can estimate sizes of:

$\Pi_{AB}(R)$

$\sigma_{A=a \wedge B=b}(R)$

R ⋈ S with common attribs. A,B,C

Union, intersection, diff,

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Note: for complex expressions, need intermediate T,S,V results.

E.g. $W = [\sigma_{A=a}(R1)] \bowtie R2$

Treat as relation U

$T(U) = T(R1)/V(R1,A)$ $\qquad$ $S(U) = S(R1)$

Also need V (U, *) !!

To estimate Vs

E.g., $U = \sigma_{A=a}(R1)$

        Say R1 has attribs A,B,C,D

   $V(U, A) =$

   $V(U, B) =$

   $V(U, C) =$

   $V(U, D) =$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example

R1

A	B	C	D
cat	1	10	10
cat	1	20	20
dog	1	30	10
dog	1	40	30
bat	1	50	10

$V(R1,A)=3$

$V(R1,B)=1$

$V(R1,C)=5$

$V(R1,D)=3$

$U = \sigma_{A=a}(R1)$

Example

R1

A	B	C	D
cat	1	10	10
cat	1	20	20
dog	1	30	10
dog	1	40	30
bat	1	50	10

$V(R1,A)=3$

$V(R1,B)=1$

$V(R1,C)=5$

$V(R1,D)=3$

$U = \sigma_{A=a}(R1)$

$V(U,A) = 1$ $V(U,B) = 1$ $V(U,C) = \dfrac{T(R1)}{V(R1,A)}$

$V(D,U)$ ... somewhere in between

Notes 9 - Logical Optimization   79

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Possible Guess $\quad U = \sigma_{A=a}(R)$

$V(U,A) \quad = 1$

$V(U,B) \quad = V(R,B)$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

For Joins $U = R1(A,B) \bowtie R2(A,C)$

$V(U,A) = \min \{ V(R1, A), V(R2, A) \}$

$V(U,B) = V(R1, B)$

$V(U,C) = V(R2, C)$

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example:

$$Z = R1(A,B) \bowtie R2(B,C) \bowtie R3(C,D)$$

R1   T(R1) = 1000   V(R1,A)=50   V(R1,B)=100

R2   T(R2) = 2000   V(R2,B)=200   V(R2,C)=300

R3   T(R3) = 3000   V(R3,C)=90   V(R3,D)=500

Partial Result: $U = R1 \bowtie R2$

$$T(U) = \frac{1000 \times 2000}{200}$$

$V(U,A) = 50$

$V(U,B) = 100$

$V(U,C) = 300$

Notes 9 - Logical Optimization

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

$$Z = U \bowtie R3$$

$$T(Z) = \frac{1000 \times 2000 \times 3000}{200 \times 300}$$

$V(Z,A) = 50$

$V(Z,B) = 100$

$V(Z,C) = 90$

$V(Z,D) = 500$

Approximating Distributions

- Summarize the distribution
  - Used to better estimate result sizes
  - Without the need to look at all the data

- Concerns
  - Error metric: How to measure preciseness
  - Memory consumption
  - Computational Complexity

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Approximating Distributions

- Parameterized distribution
  - E.g., gauss distribution
  - Adapt parameters to fit data

- Histograms
  - Divide domain into ranges (buckets)
  - Store the number of tuples per bucket

- Both need to be maintained

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Histograms

Parameterized Distribution

Notes 9 - Logical Optimization

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
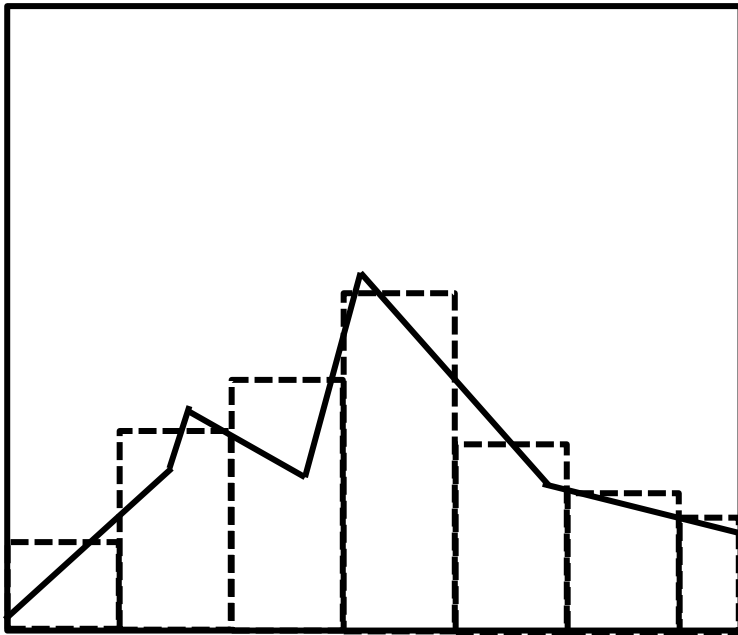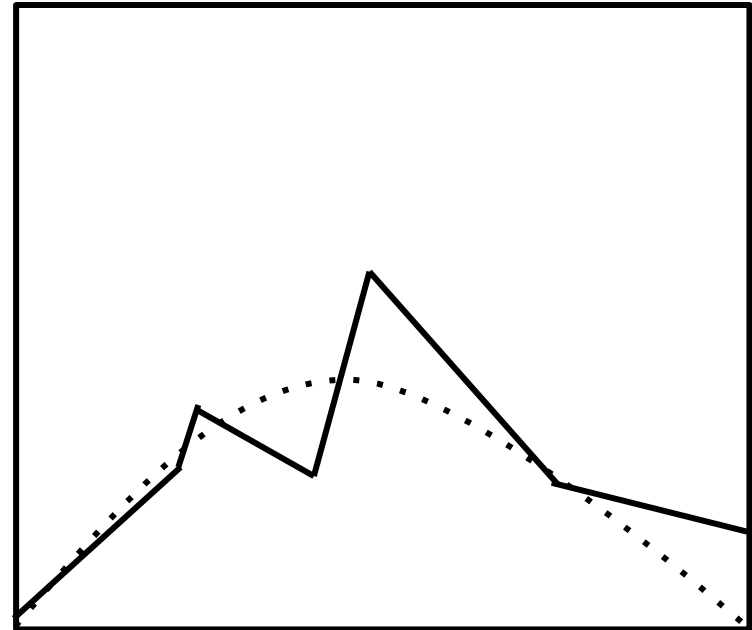
Maintaining Statistics

- Use separate command that triggers statistics collection
  - Postgres: ANALYZE
- During query processing
  - Overhead for queries
- Use Sampling?

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Estimating Result Size using Histograms
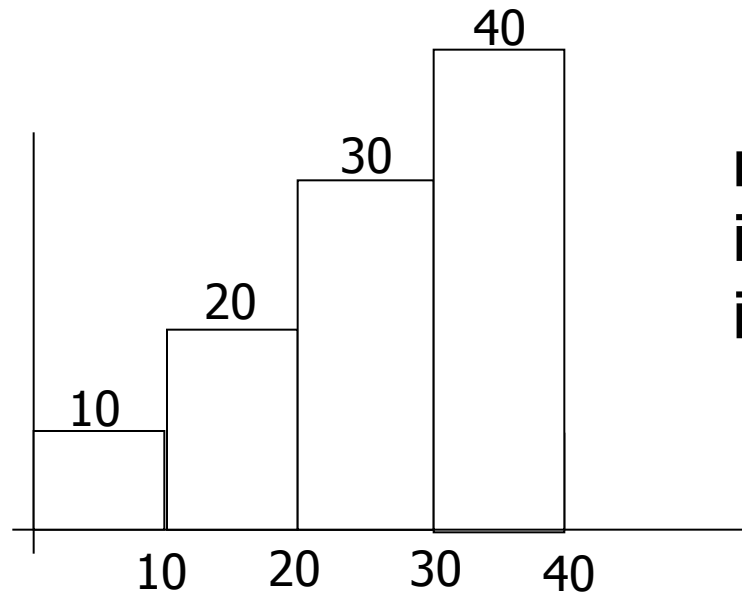
number of tuples in R with A value in given range

$$\sigma_{A=val}(R) = ?$$

IIT College of
Science and Letters

COMPUTER
SCIENCE

ILLINOIS INSTITUTE OF TECHNOLOGY

Estimating Result Size using Histograms

- $\sigma_{A=val}(R) = ?$
- |B| - number of values per bucket
- #B – number of records in bucket

$$\frac{\#B}{|B|}$$

CS 525

COMPUTER SCIENCE

Notes 9 - Logical Optimization   90

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Join Size using Histograms

- R ⋈ S
- Use

$$T(W) = \frac{T(R2)\, T(R1)}{\max\{\, V(R1,A),\, V(R2,A)\, \}}$$

- Apply for each bucket

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Join Size using Histograms

- $V(R1,A) = V(R2,A) =$ bucket size $|B|$

$$T(W) = \sum_{buckets} \frac{\#B(R2)\ \#B(R1)}{|B|}$$

Equi-width vs. Equi-depth

- Equi-width
  - All buckets contain the same number of values
  - Easy, but inaccurate

- Equi-depth (used by most DBMS)
  - All buckets contain the same number of tuples
  - Better accuracy, need to sort data to compute

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Equi-width vs. Equi-depth

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
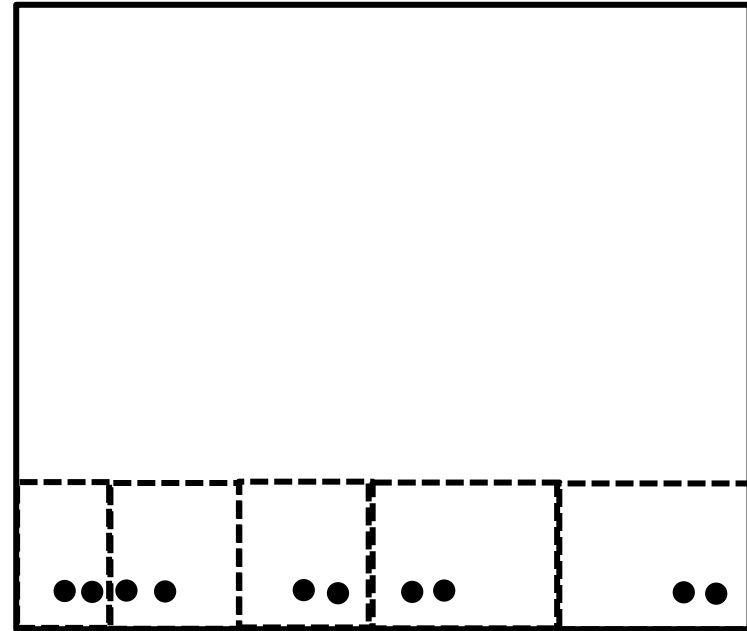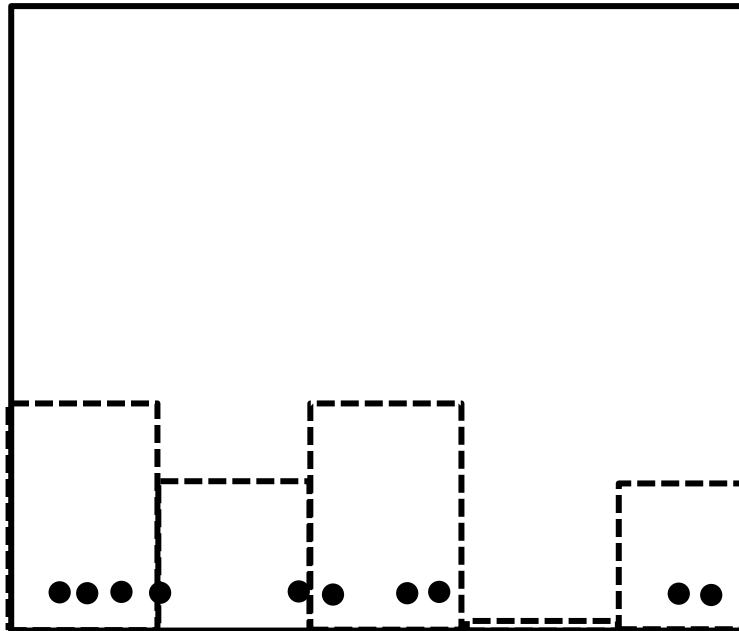
Construct Equi-depth Histograms

- Sort input
- Determine size of buckets
  - #bucket / #tuples
- Example 3 buckets

```
1,  5,44,  6,10,12,  3,  6,  7
1,  3,  5,  6,  6,  7,10,12,44
[1–5] [6–8] [9–44]
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Advanced Techniques

- Wavelets
- Approximate Histograms
- Sampling Techniques
- Compressed Histograms

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Summary

- Estimating size of results is an "art"

- Don't forget:

    Statistics must be kept up to date...
        (cost?)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Outline

- Estimating cost of query plan
  - Estimating size of results ⟵ done!
  - Estimating # of IOs ⟵ next…
  - Operator Implementations

- Generate and compare plans

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

CS 525: Advanced Database Organization

10: Query Execution

Boris Glavic

Slides: adapted from a course taught by
Hector Garcia-Molina, Stanford InfoLab

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

SQL query

parse

parse tree

convert

logical query plan

apply laws

"improved" l.q.p

statistics

answer

execute

Pi

estimate result sizes

pick best

l.q.p. +sizes

$\{(P1,C1),(P2,C2)...\}$

consider physical plans

estimate costs
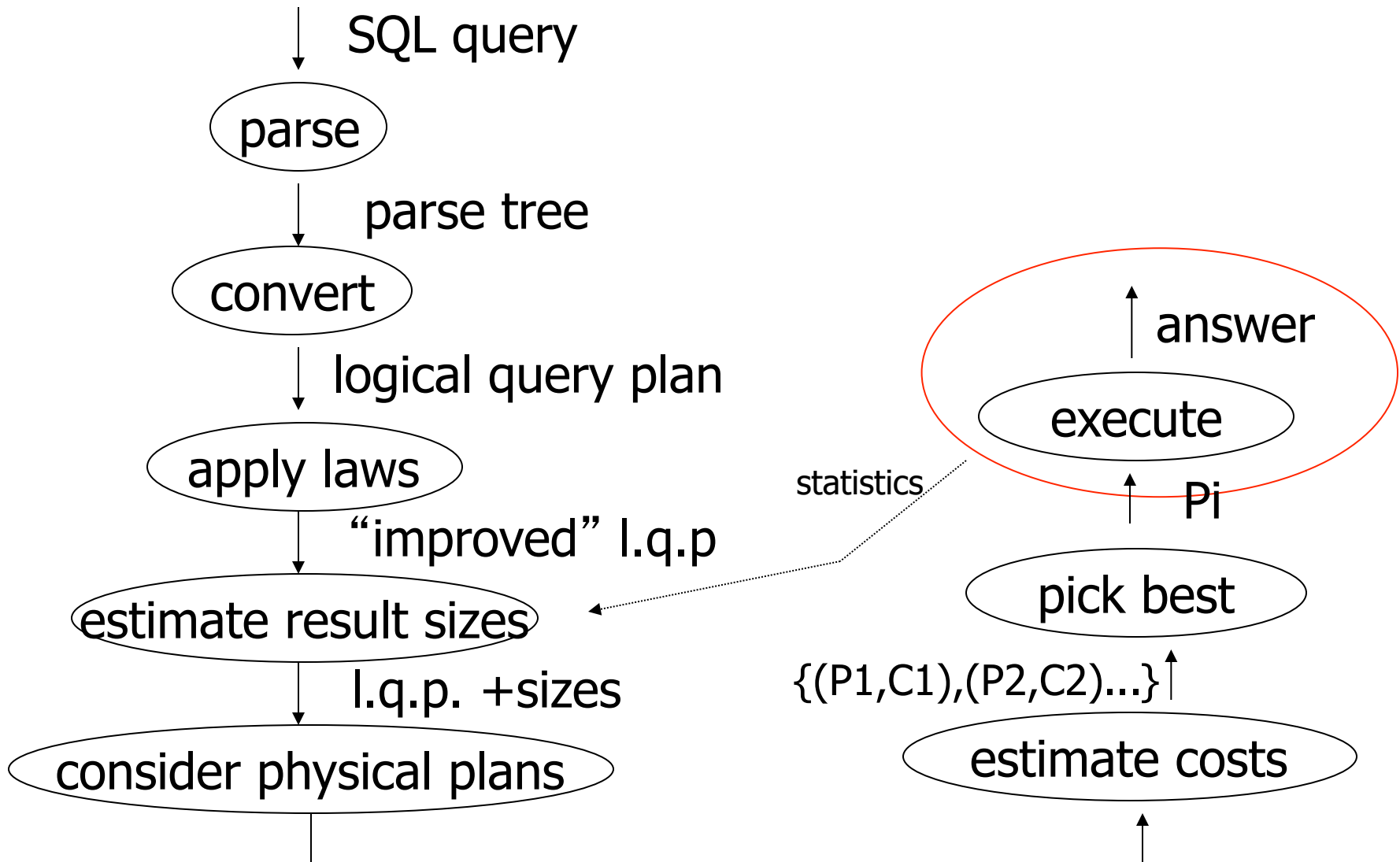
$\{P1,P2,.....\}$

Query Execution

- Here only:
  - how to implement operators
  - what are the costs of implementations
  - how to implement queries
    - Data flow between operators

- Next part:
  - How to choose good plan

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Execution Plan

- A tree (DAG) of physical operators that implement a query

- May use indices

- May create temporary relations

- May create indices on the fly

- May use auxiliary operations such as sorting

How to estimate costs

- If everything fits into memory
  - Standard computational complexity
- If not
  - Assume fixed memory available for buffering pages
  - Count I/O operations
  - Real systems combine this with CPU estimations

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Estimating IOs:

- Count # of disk blocks that must be read (or written) to execute query plan

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

To estimate costs, we may have additional parameters:

B(R) = # of blocks containing R tuples

f(R) = max # of tuples of R per block

M   = # memory blocks available

Notes 10 - Query Execution

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

To estimate costs, we may have additional parameters:

B(R) = # of blocks containing R tuples

f(R) = max # of tuples of R per block

M   = # memory blocks available

HT(i) = # levels in index i

LB(i) = # of leaf blocks in index i

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Clustered index

Index that allows tuples to be read in an order that corresponds to physical order
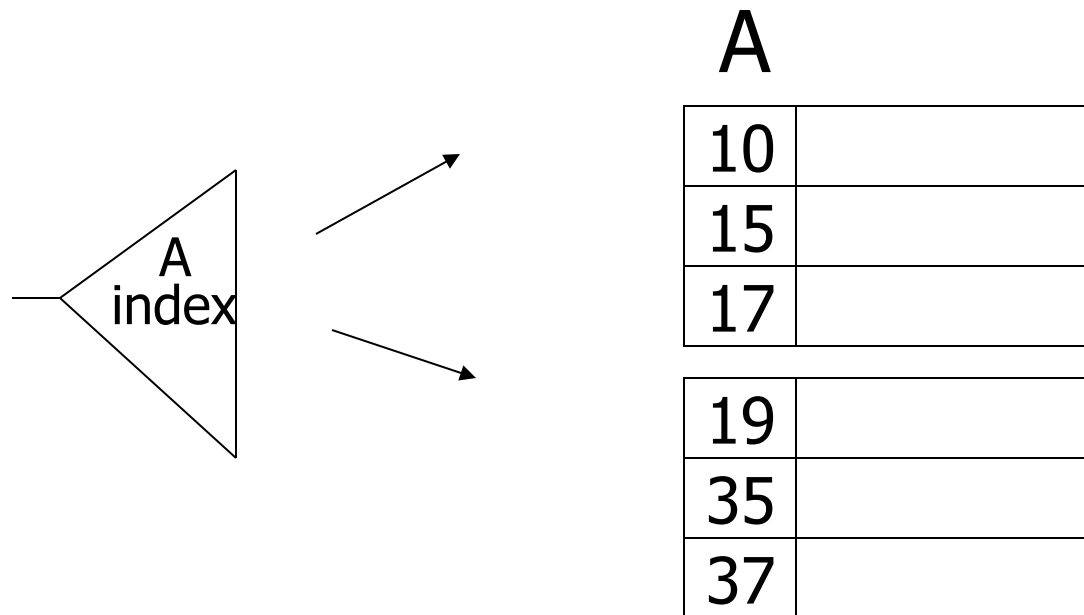
A

10	
15	
17	

19	
35	
37	

A index

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Operators Overview

- (External) Sorting
- Joins (Nested Loop, Merge, Hash, …)
- Aggregation (Sorting, Hash)
- Selection, Projection (Index, Scan)
- Union, Set Difference
- Intersection
- Duplicate Elimination

Notes 10 - Query Execution

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Operator Profiles

- Algorithm

- In-memory complexity: e.g., $O(n^2)$

- Memory requirements

  – Runtime based on available memory

- #I/O if operation needs to go to disk

- Disk space needed

- Prerequisites

  – Conditions under which the operator can be applied

Execution Strategies

- Compiled
  - Translate into C/C++/Assembler code
  - Compile, link, and execute code
- Interpreted
  - Generic operator implementations
  - Generic executor
    - Interprets query plan

Virtual Machine Approach

- Implement virtual machine of low-level DBMS operations

- Compile query into machine-code for that machine

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Iterator Model

- Need to be able to combine operators in different ways
  - E.g., join inputs may be scans, or outputs of other joins, …
  - -> define generic interface for operators
  - be able to arbitrarily compose complex plans from a small set of operators

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Iterator Model - Interface

- **Open**

  – Prepare operator to read inputs

- **Close**

  – Close operator and clean up

- **Next**

  – Return next result tuple

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Query Execution – Iterator Model
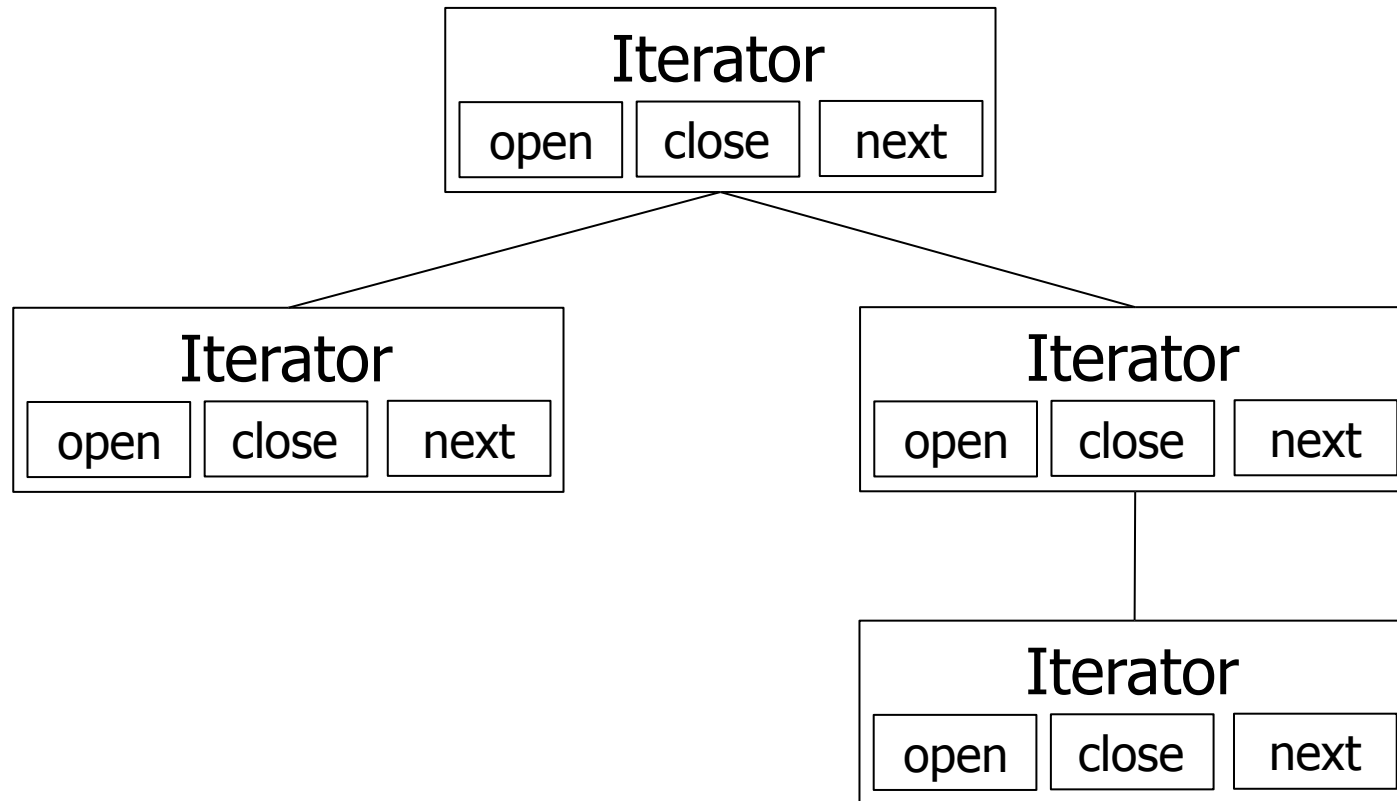
Query Execution – Iterator Model
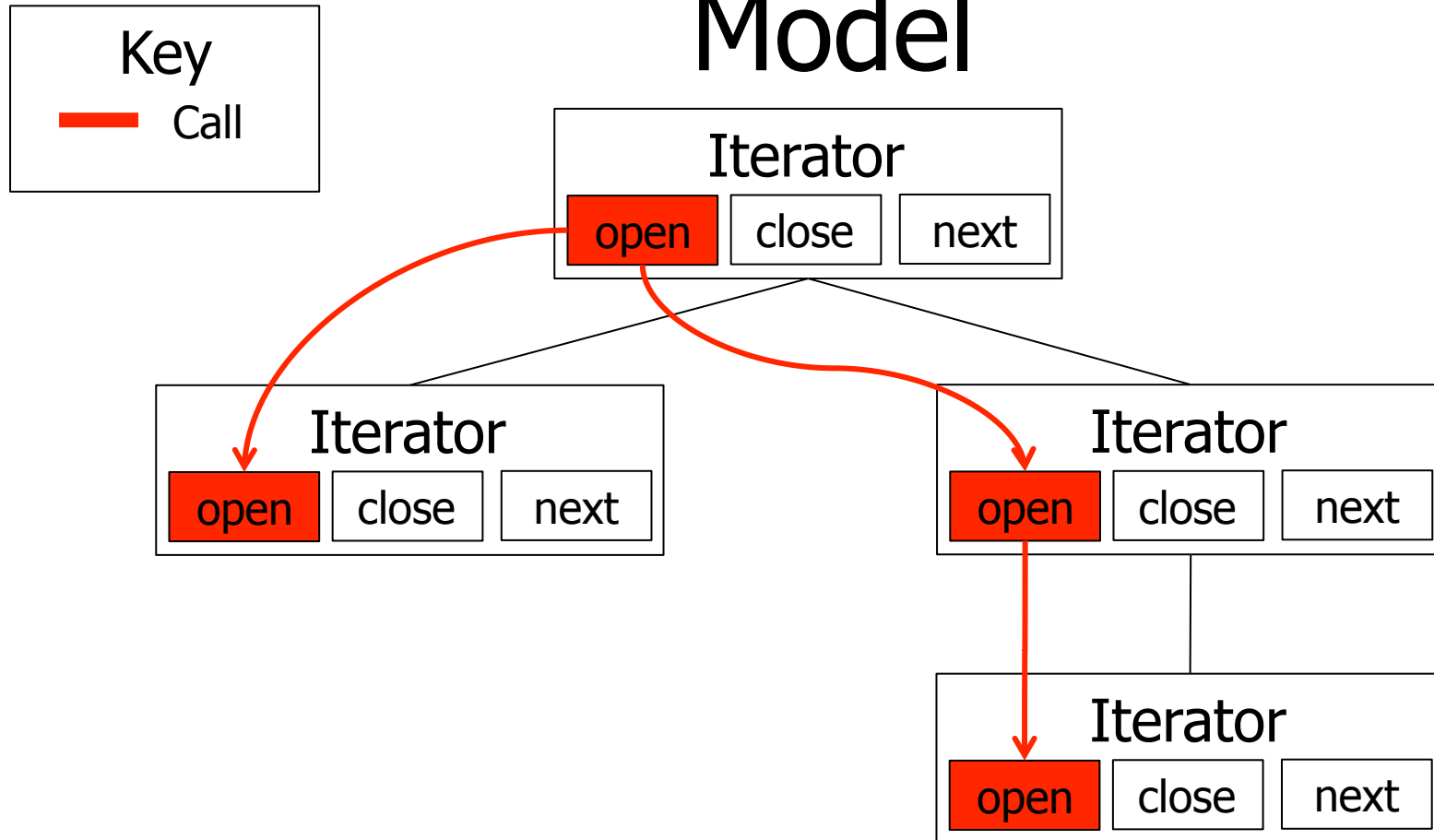
Query Execution – Iterator Model
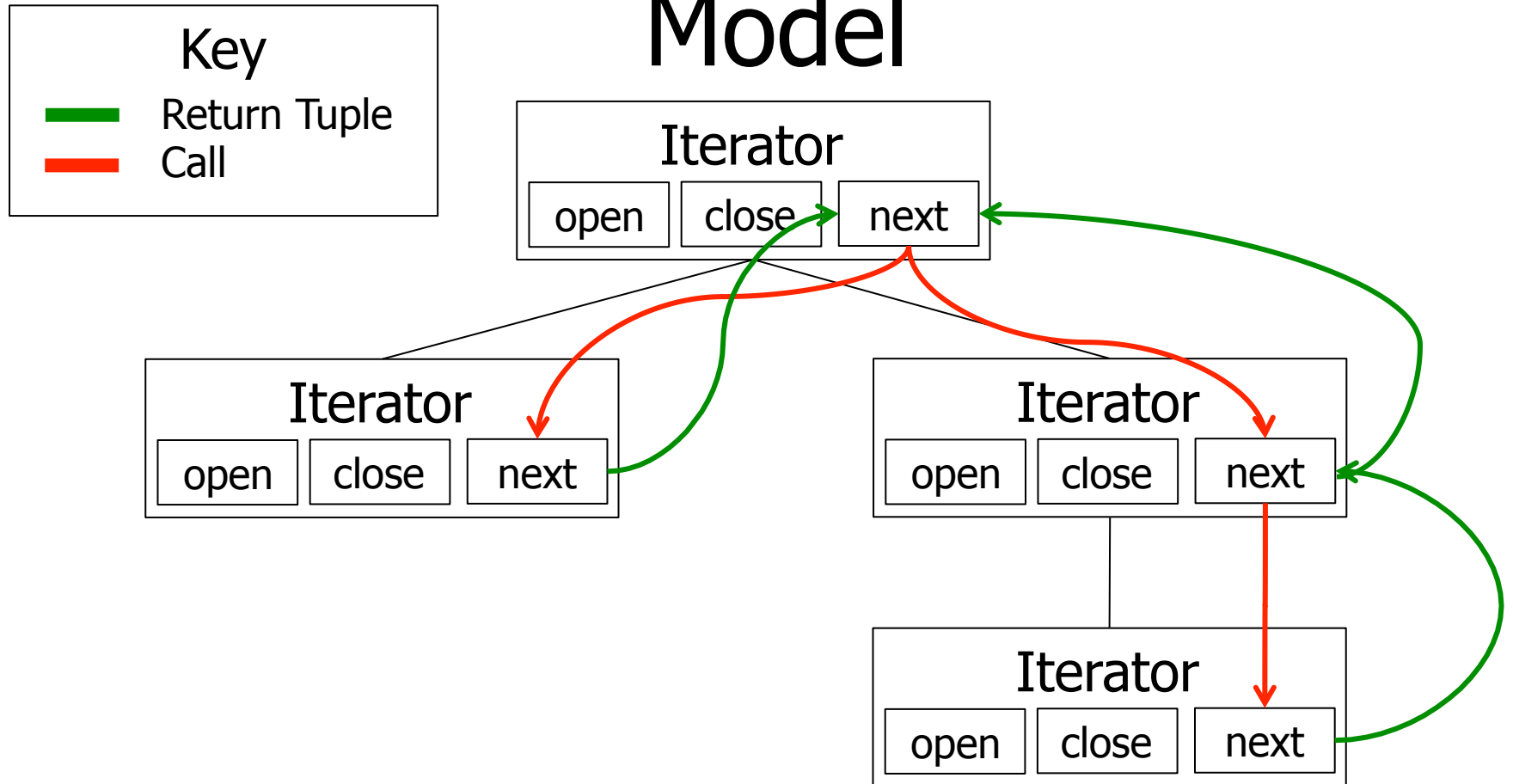
Parallelism

- Iterator Model
  - **Pull-based** query execution
- Potential types of parallelism
  - Inter-query (every multiuser system)
  - Intra-operator
  - Inter-operator

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Intra-Operator Parallelism

- Execute portions of an operator in parallel
  - Merge-Sort
    - Assign a processor to each merge phase
  - Scan
    - Partition tables
    - Each process scans one partition

Notes 10 - Query Execution

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Inter-Operator Parallelism

- Each process executes one or more operators

- **Pipelining**
  - **Push-based** query execution
  - Chain operators to directly produce results
  - Pipeline-breakers
    - Operators that need to consume the whole input (or large parts) before producing outputs

Notes 10 - Query Execution

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Pipelining Communication

- Queues
  - Operators push their results to queues
  - Operators read their inputs from queues
- Direct call
  - Operator calls its parent in the tree with results
  - Within one process

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Pipelines

Key
- Append to queue
- Dequeue
- Direct Call

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
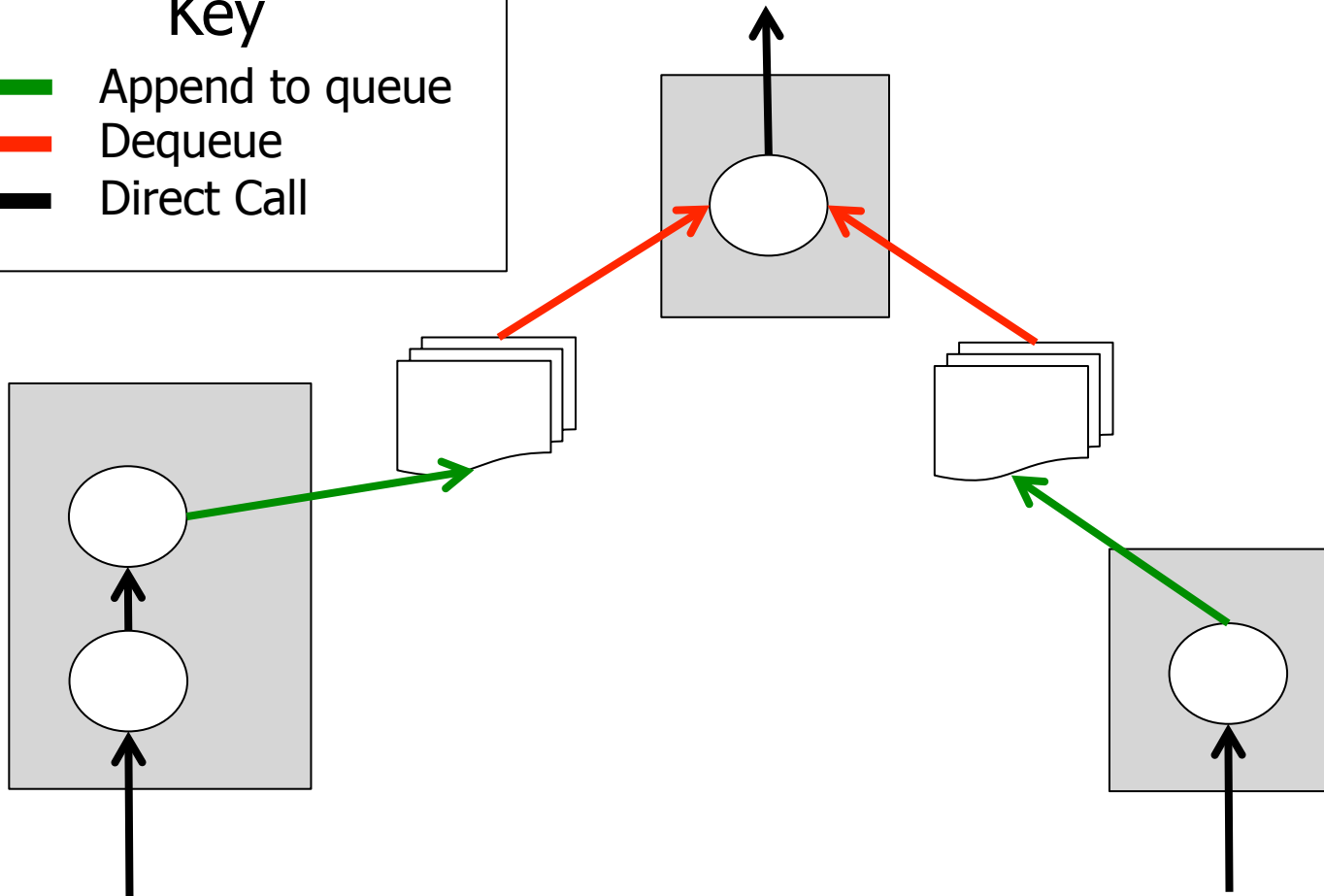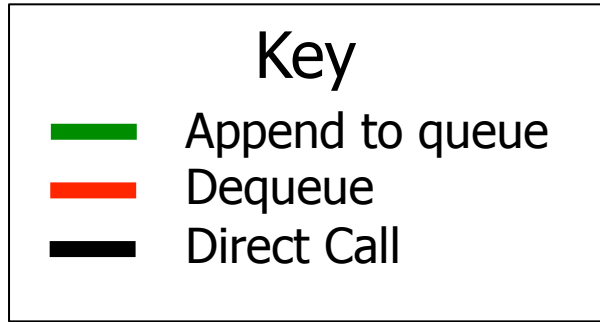
Pipeline-breakers

- Sorting
  - All operators that apply sorting
- Aggregation
- Set Difference
- Some implementations of
  - Join
  - Union

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Operators Overview

- (External) Sorting
- Joins (Nested Loop, Merge, Hash, …)
- Aggregation (Sorting, Hash)
- Selection, Projection (Index, Scan)
- Union, Set Difference
- Intersection
- Duplicate Elimination

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Sorting

- Why do we want/need to sort
  - Query requires sorting (ORDER BY)
  - Operators require sorted input
    - Merge-join
    - Aggregation by sorting
    - Duplicate removal using sorting

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

In-memory sorting

- Algorithms from data structures 101
  - Quick sort
  - Merge sort
  - Heap sort
  - Intro sort
  - ...

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

External sorting

- Problem:
  - Sort **N** pages of data with **M** pages of memory

- Solutions?

First Idea

- Split data into runs of size **M**
- Sort each run in memory and write back to disk
  - ⌈**N/M**⌉ sorted runs of size **M**
- Now what?

M   **M**   **M**
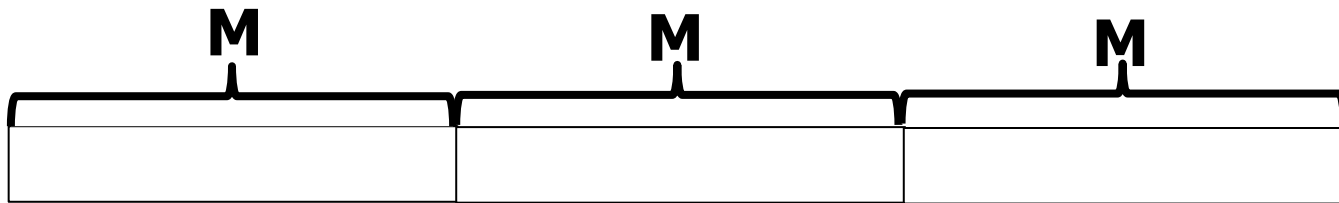
Merging Runs

- Need to create bigger sorted runs out of sorted smaller runs
  - Divide and Conquer
  - Merge Sort?
- How to merge two runs that are bigger than **M?**

COMPUTER
SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merging Runs using 3 pages

- Merging sorted runs R_1 and R_2

- Need 3 pages
  - One page to buffer pages from R_1
  - One page to buffer pages from R_2
  - One page to buffer the result
    - Whenever this buffer is full, write it to disk

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merging Runs
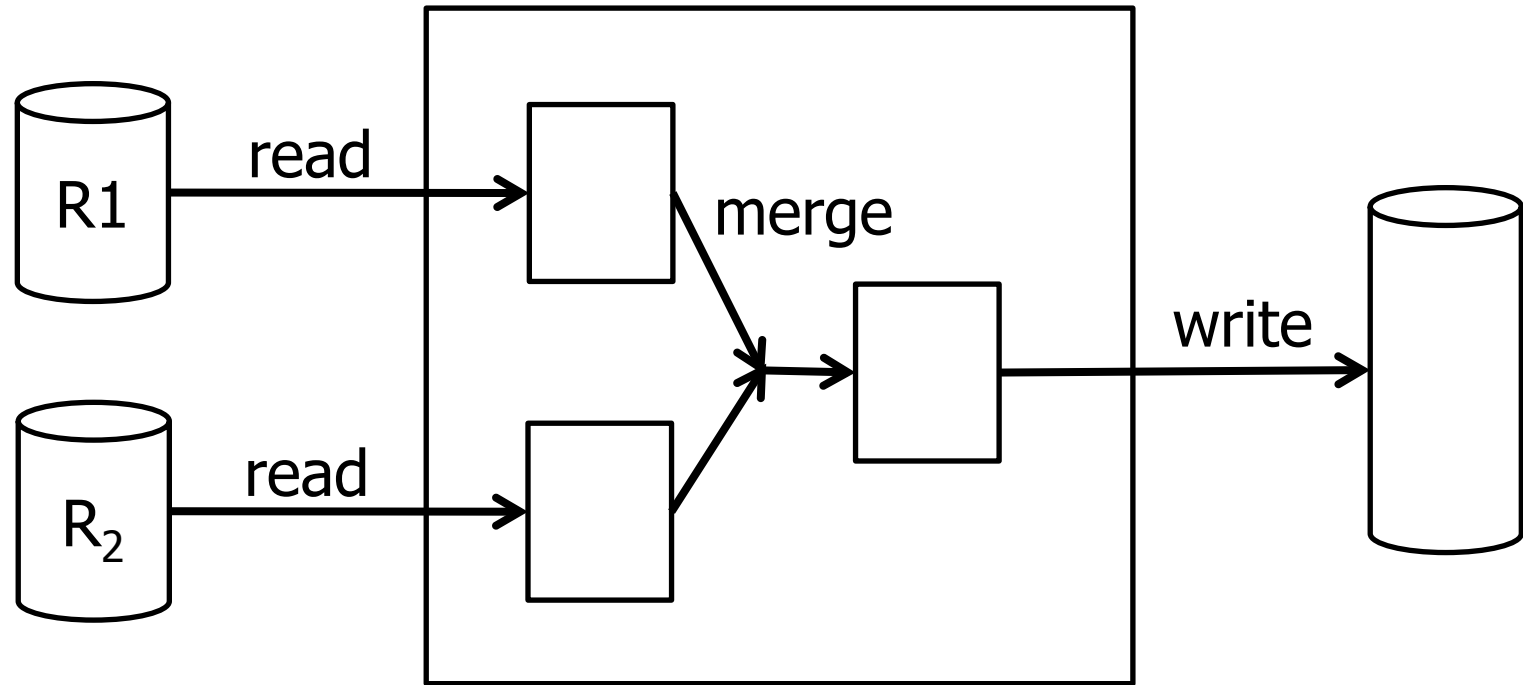
2-Way External Mergesort

- Repeat process until we have one sorted run

- Each iteration (pass) reads and writes the whole table once: **2 B(R)** I/Os

- Each pass doubles the run size
  - **1 + $\lceil$log$_2$ (B(R) / M)$\rceil$** runs
  - **2 B(R) $*$ (1 + $\lceil$log$_2$ (B(R) / M)$\rceil$)** I/Os

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Input

2 3 6 5 7 4 10 1 11 12 13 14 20 40 22 21

Pass 0

2 3 | 5 6 | 4 7 | 10 1 | 11 12 | 13 14 | 20 40 | 21 22

Pass 1

2 3 5 6 | 1 4 7 10 | 11 12 13 14 | 20 21 22 40

Pass 2

1 2 3 4 5 6 7 10 | 11 12 13 14 20 21 22 40

Pass 3

1 2 3 4 5 6 7 10 11 12 13 14 20 21 22 40

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
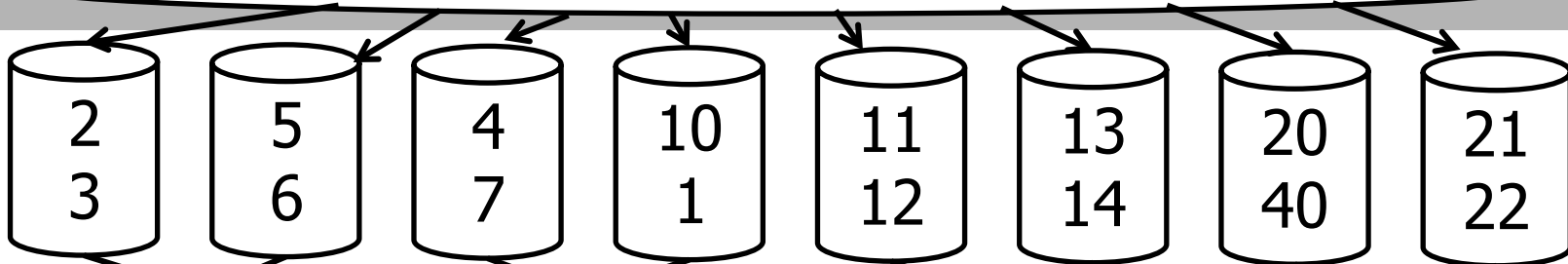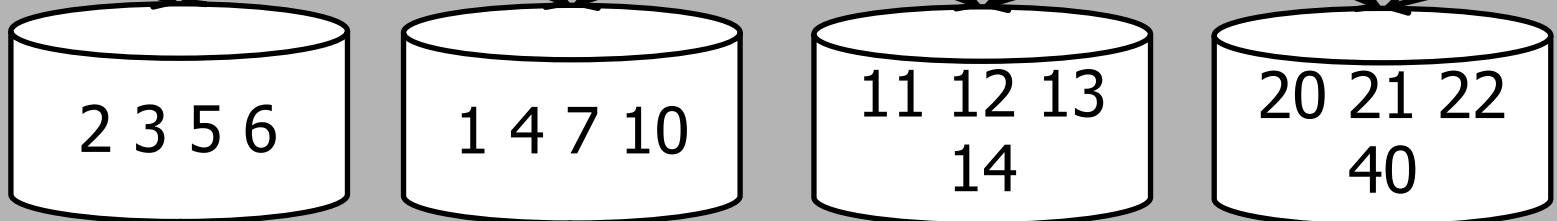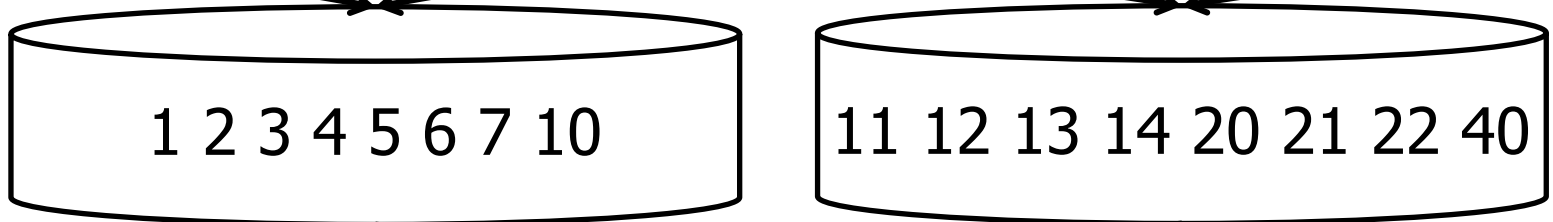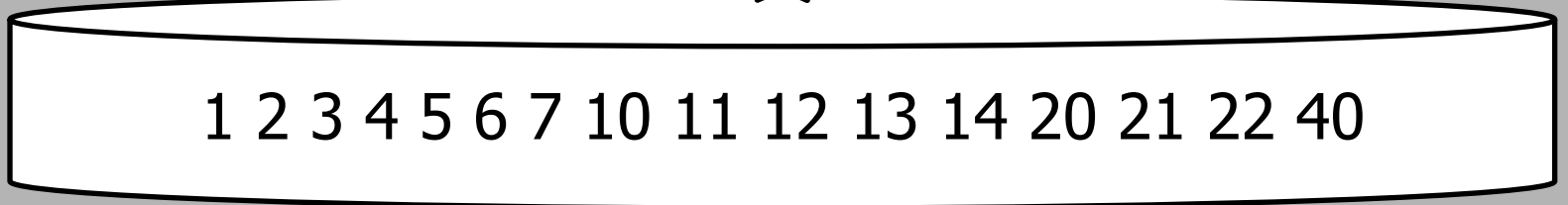
N-Way External Mergesort

- How to utilize **M** buffer during merging?
- Each pass merges **M-1** runs at once
  - One memory page as buffer for each run
- #I/Os

$$1 + \lceil \textbf{log}_{\textbf{M-1}} \textbf{ (B(R) / M)} \rceil \text{ runs}$$
$$2 \textbf{ B(R)} * (\textbf{1 } + \text{ } \lceil \textbf{log}_{\textbf{M-1}} \textbf{ (B(R) / M)} \rceil) \text{ I/Os}$$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Merging Runs

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
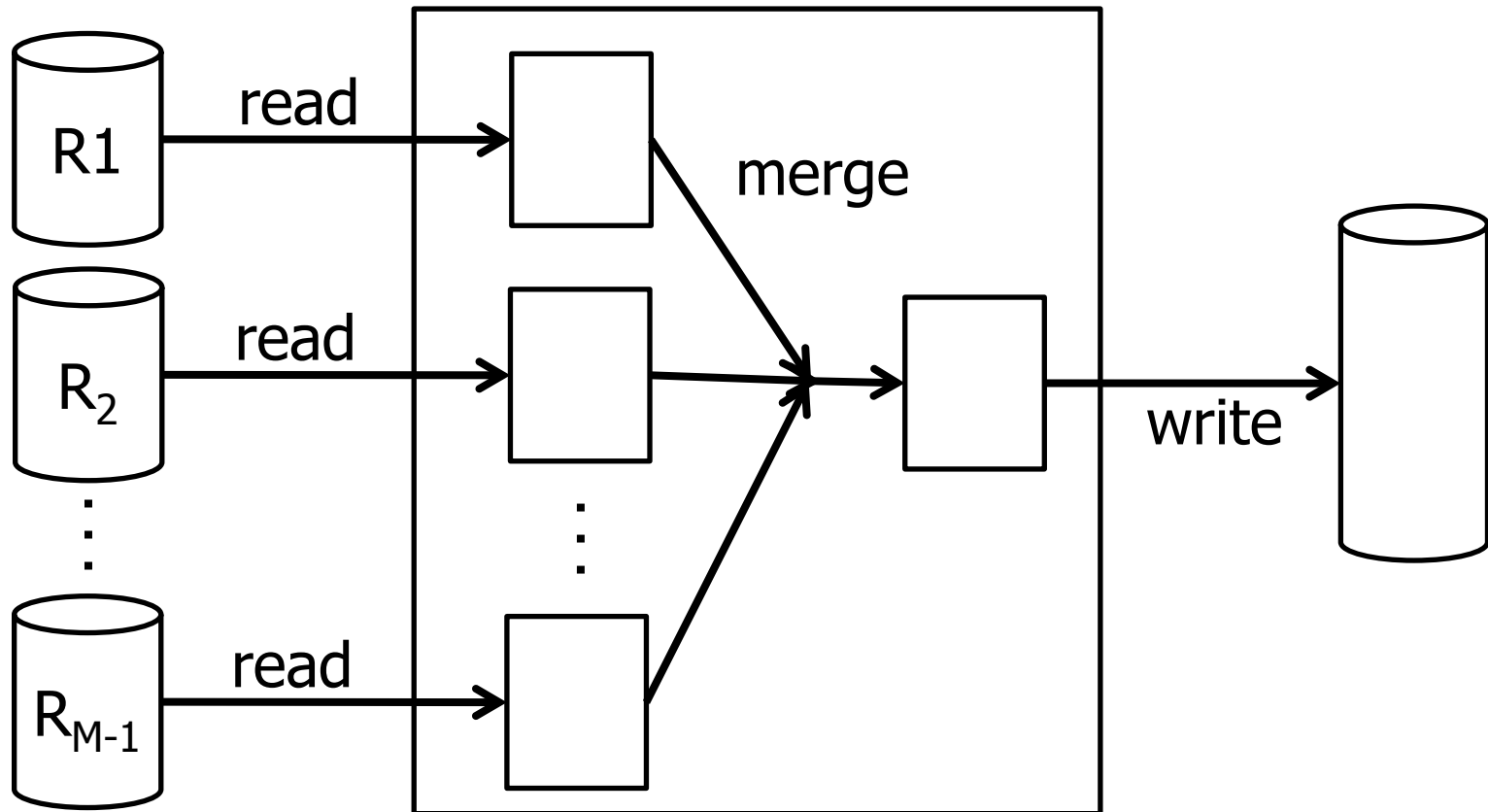
How many passes do we need?

N	M=17	M=129	M=257	M=513	M=1025
100	2	1	1	1	1
1,000	3	2	2	2	1
10,000	4	2	2	2	2
100,000	5	3	3	2	2
1,000,000	5	3	3	3	2
10,000,000	6	4	3	3	3
100,000,000	7	4	4	3	3
1,000,000,000	8	5	4	4	3

Notes 10 - Query Execution

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

To put into perspective

- Scenario
  - Page size 4KB
  - 1TB of data (250,000,000)
  - 10MB of buffer for sorting (250)

- Passes
  - 4 passes

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Merge

- In practice would want larger I/O buffer for each run

- Trade-off between number of runs and efficiency of I/O

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Improving in-memory merging

- Merging **M** runs
  - To choose next element to output
  - Have to compare **M** elements
  - -> complexity linear in **M**: **O(M)**
- How to improve that?
  - Use priority queue to store current element from each run
  - -> **O($\log_2$(M))**

Notes 10 - Query Execution

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Priority Queue

- Queue for accessing elements in some given order
  - **pop-smallest** = return and remove smallest element in set
  - **Insert(e)** = insert element into queue

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Min-Heap

- Implementation of priority queue
  - Store elements in a binary tree
  - All levels are full (except leaf level)
  - Heap property
    - Parent is smaller than child
- Example: { 1, 4, 7, 10 }

COMPUTER
SCIENCE

Notes 10 - Query Execution

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
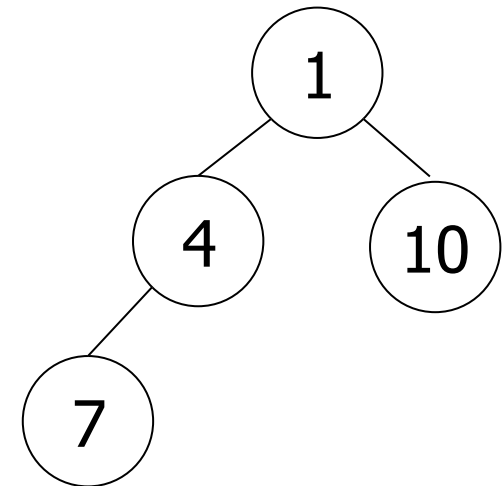
Min-Heap Insertion

- **`insert(e)`**

  1. Add element at next free leaf node
     - This may invalidate heap property
  2. If node smaller than parent then
     - Switch node with parent
  3. Repeat until 2) cannot be applied anymore

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Min-Heap Dequeue

- **pop-smallest**

  1. Return Root and use right-most leaf as new root

     - This may invalidate heap property

  2. If node smaller than child then

     - Switch node with smaller child

  3. Repeat until 2) cannot be applied anymore

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Insertion

- Insert 3

Insert at first free position
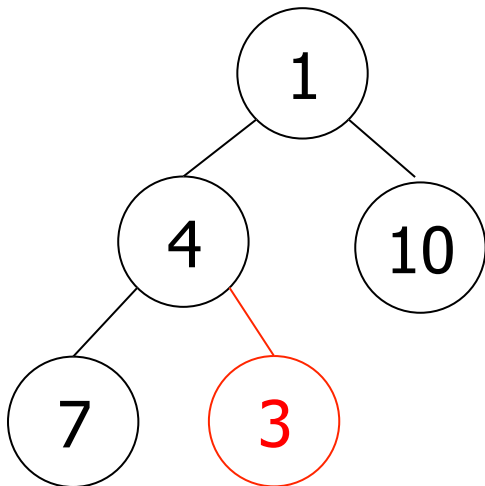
Restore heap property

Dequeue

Notes 10 - Query Execution

46

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Dequeue

COMPUTER SCIENCE

Notes 10 - Query Execution

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
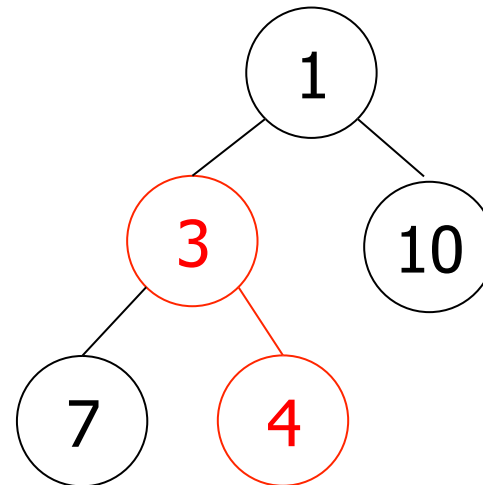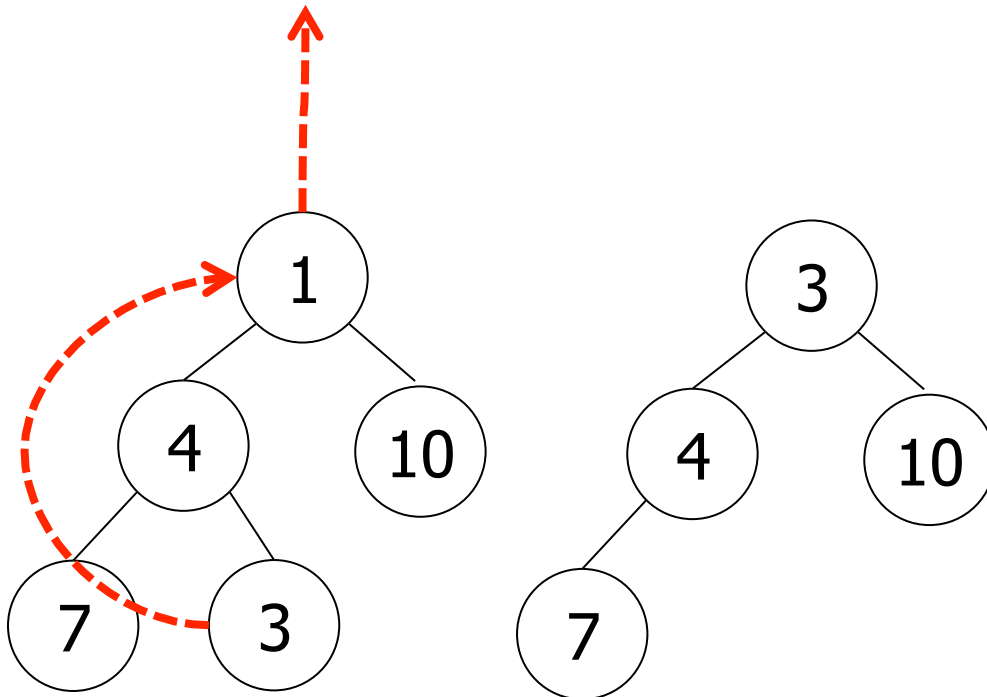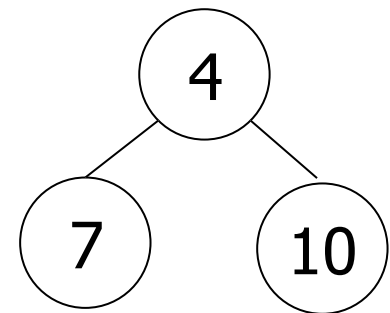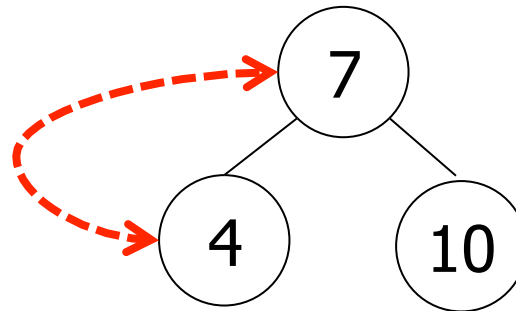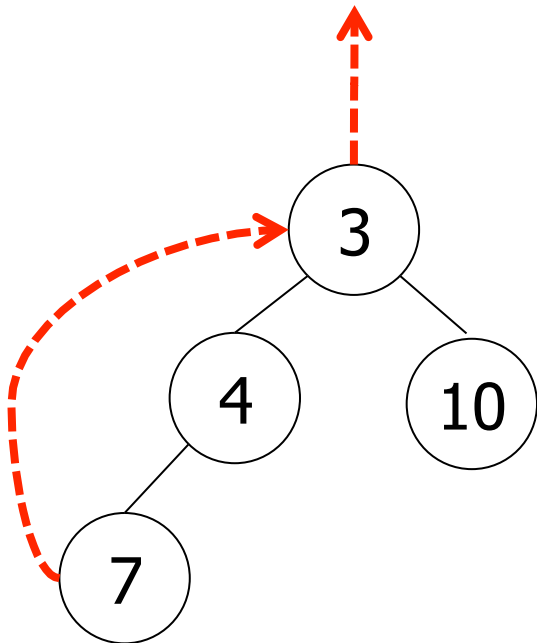
Min/Max-Heap Complexity

- Heap is a complete tree
  - Height is $O(\log_2(n))$
- Insertion
  - Maximal height of the tree switches
  - -> $O(\log_2(n))$
- Dequeue
  - Maximal height of the tree switches
  - -> $O(\log_2(n))$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Min-Heap Implementation

- Full tree
  - Use array to implement tree
- Compute positions
  - Parent(n) = $\lfloor (n-1) / 2 \rfloor$
  - Children(n) = 2n + 1, 2n + 2
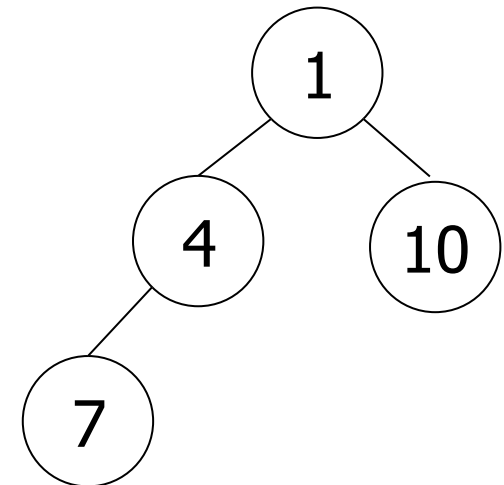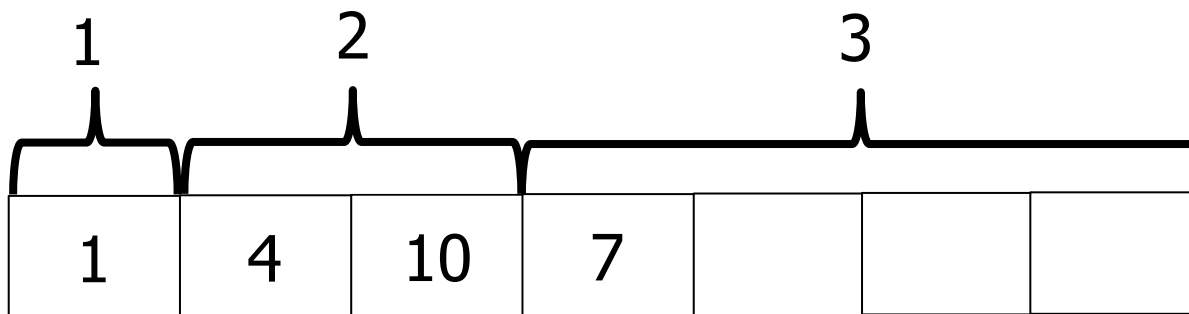
1   2   3

1	4	10	7			

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merging with Priority Queue

Notes 10 - Query Execution

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Merging with Priority Queue

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merging with Priority Queue

Notes 10 - Query Execution

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
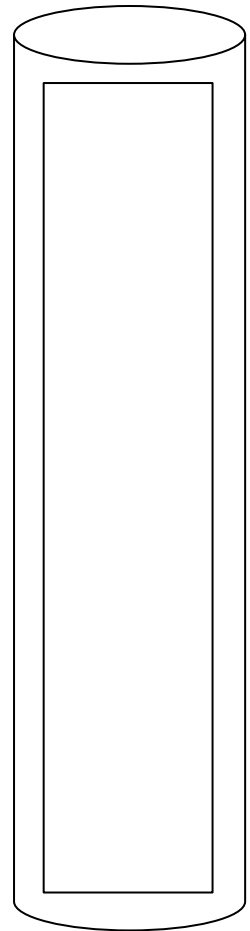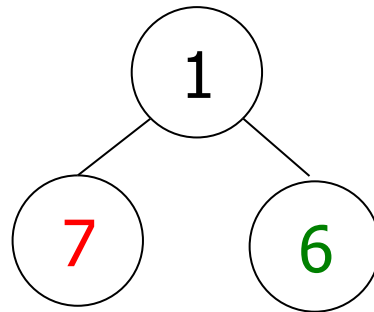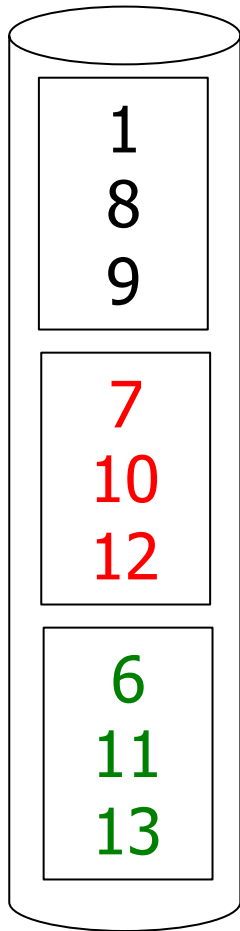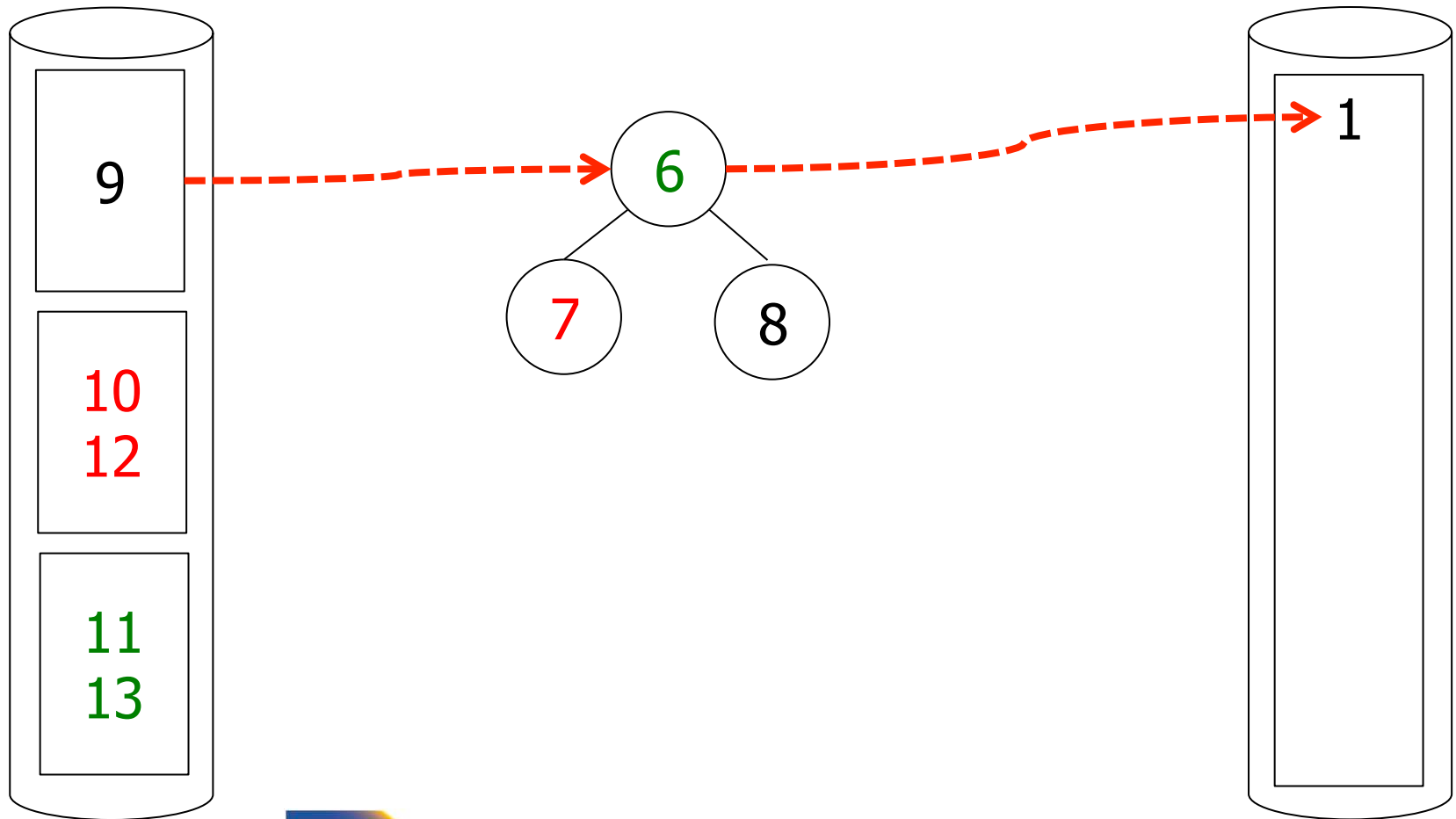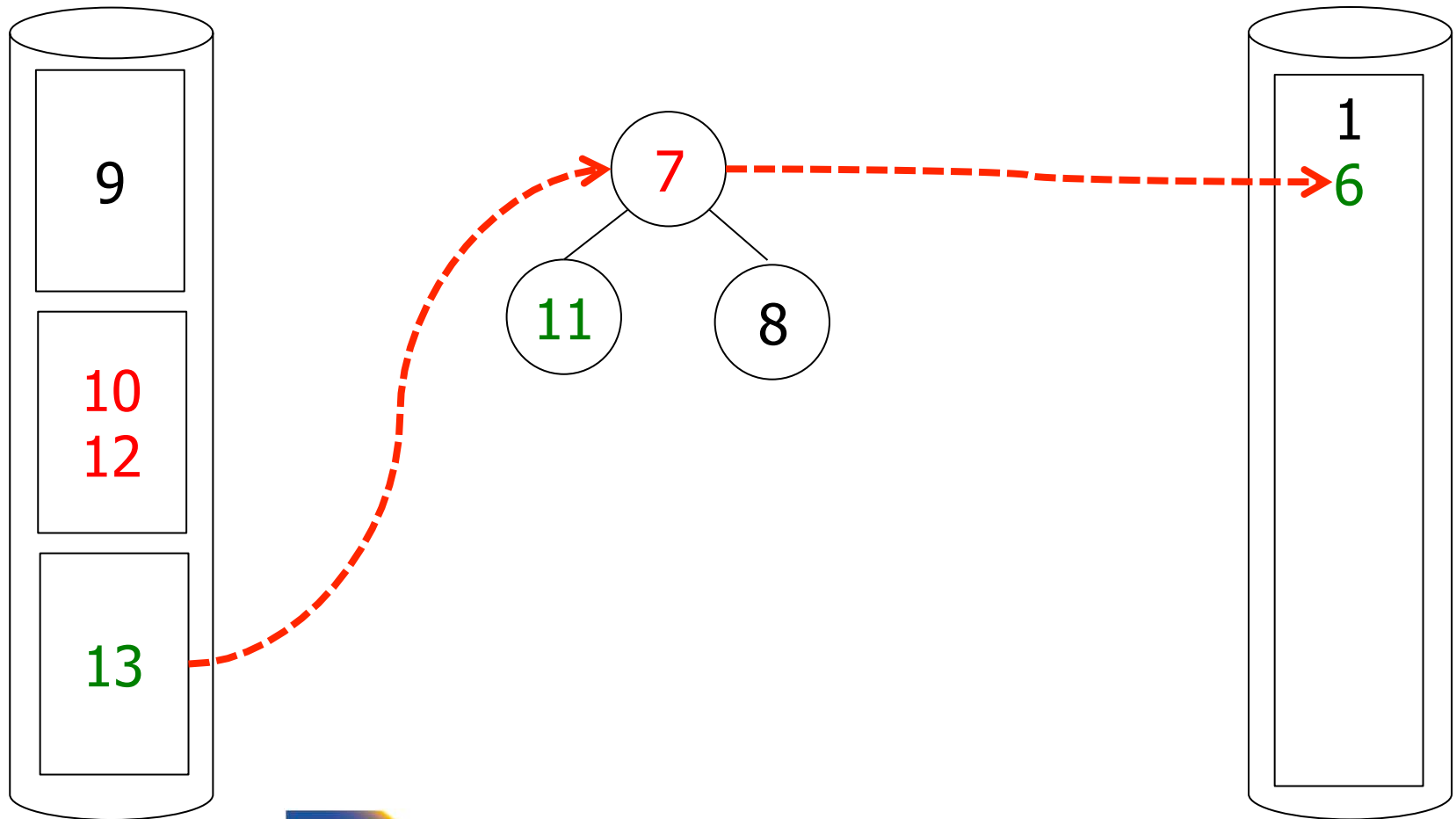
Using a heap to generate runs

- Read inputs into heap
  - Until available memory is full
- Replace elements
  - Remove smallest element from heap
    - If larger then last element written of current run then write to current run
    - Else create a new run
  - Add new element from input to heap

Notes 10 - Query Execution   53

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Using a heap to generate runs

5
7
2
3
4
12
15
1

2

7   5

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Using a heap to generate runs

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Using a heap to generate runs

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Using a heap to generate runs

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Using a heap to generate runs

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Using a heap to generate runs

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Using a heap to generate runs

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Using a heap to generate runs

- Increases the run-length
  - On average by a factor of 2 (see Knuth)

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Use clustered B+-tree

- Keys in the B+-tree **I** are in sort order
  - If B+-tree is clustered traversing the leaf nodes is sequential I/O!
  - **K** = #keys/leaf node

- Approach
  - Traverse from root to first leaf: **HT(I)**
  - Follow sibling pointers: **|R| / K**
  - Read data blocks**: B(R)**

I/O Operations

- **HT(I) + |R| / K + B(R)** I/Os
- Less than **2 B(R)** = 1 pass external mergesort
- ->Better than external merge-sort!

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Unclustered B+-tree?

- Each entry in a leaf node may point to different page of relation R

  - For each leaf page we may read up to **K** pages from relation R

  - Random I/O

- In worst-case we have

  - **K** $*$ **B(R)**

  - **K = 500**

    - **500** $*$ **B(R)** = 250 merge passes

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Sorting Comparison

B(R) = number of block of R
M = number of available memory blocks
#RB = records per page
HT = height of B+-tree (logarithmic)
K = number of keys per leaf node

Property	Ext. Mergesort	B+ (clustered)	B+ (unclustered)
Runtime	$O(N \log_{M-1}(N))$	$O(N)$	$O(N)$
#I/O (random)	$2 B(R) * (1 + \lceil \log_{M-1}(B(R) / M) \rceil)$	HT + |R| / K + B(R)	HT + |R| / K + K * #RB
Memory	M	1 (better HT + X)	1 (better HT + X)
Disk Space	2 B(R)	0	0
Variants	1) Merge with heap 2) Run generation with heap 3) Larger Buffer		

Operators Overview

- (External) Sorting
- Joins (Nested Loop, Merge, Hash, …)
- Aggregation (Sorting, Hash)
- Selection, Projection (Index, Scan)
- Union, Set Difference
- Intersection
- Duplicate Elimination

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Scan

- Implements access to a table
  - Combined with selection
  - Probably projection too

- Variants
  - **Sequential**
    - Scan through all tuples of relation
  - **Index**
    - Use index to find tuples that match selection

Operators Overview

- (External) Sorting
- Joins (Nested Loop, Merge, Hash, …)
- Aggregation (Sorting, Hash)
- Selection, Projection (Index, Scan)
- Union, Set Difference
- Intersection
- Duplicate Elimination

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Options

- Transformations: $R_1 \bowtie_c R_2,\ \ R_2 \bowtie_c R_1$
- Joint algorithms:
  - Nested loop
  - Merge join
  - Join with index
  - Hash join
- Outer join algorithms

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

<u>Nested Loop Join</u> (conceptually)

　　　for each $r \in R_1$ do

　　　　　for each $s \in R_2$ do

　　　　　　　if $(r,s) \vDash C$ then output (r,s)

<u>Applicable to:</u>
- Any join condition C
- Cross-product

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- <u>Merge Join (conceptually)</u>
  (1) if R_1 and R_2 not sorted, sort them
  (2) $i \leftarrow 1; j \leftarrow 1;$
    While $(i \leq T(R_1)) \wedge (j \leq T(R_2))$ do
      if $R_1\{ i \}.C = R_2\{ j \}.C$ then outputTuples
      else if $R_1\{ i \}.C > R_2\{ j \}.C$ then $j \leftarrow j+1$
      else if $R_1\{ i \}.C < R_2\{ j \}.C$ then $i \leftarrow i+1$

<u>Applicable to:</u>

- C is conjunction of equalities or </>:
    $A_1 = B_1$ AND ... AND $A_n = B_n$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Procedure Output-Tuples

While $(R_1\{ i \}.C = R_2\{ j \}.C) \land (i \leq T(R_1))$ do

    $[jj \leftarrow j;$

       while $(R_1\{ i \}.C = R_2\{ jj \}.C) \land (jj \leq T(R_2))$ do

               $[$output pair $R_1\{ i \}, R_2\{ jj \};$

                  $jj \leftarrow jj+1 \quad]$

    $i \leftarrow i+1 \quad]$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example

i	$R_1\{i\}.C$	$R_2\{j\}.C$	j
1	10	5	1
2	20	20	2
3	20	20	3
4	30	30	4
5	40	30	5
		50	6
		52	7

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Index nested loop (Conceptually)

For each $r \in R_1$ do

    [X $\leftarrow$ index (R_2, C, r.C)

        for each $s \in X$ do

            output (r,s) pair]

Assume R_2.C index

Note: X $\leftarrow$ index(rel, attr, value)

    then X = set of rel tuples with attr = value

Hash join (conceptual)

Hash function h, range $0 \to k$

Buckets for R_1: $G_0, G_1, \ldots G_k$

Buckets for R_2: $H_0, H_1, \ldots H_k$

Applicable to:

- C is conjunction of equalities

$$A_1 = B_1 \text{ AND } \ldots \text{ AND } A_n = B_n$$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Hash join (conceptual)

Hash function h, range $0 \rightarrow k$

Buckets for R_1: G_0, G_1, ... G_k

Buckets for R_2: H_0, H_1, ... H_k

Algorithm

(1) Hash R_1 tuples into G buckets

(2) Hash R_2 tuples into H buckets

(3) For i = 0 to k do

      match tuples in G_i, H_i buckets

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Simple example   hash: even/odd

R$_1$ R$_2$ Buckets

Even: | 2 4 8 | | 4 12 8 14 |
             R$_1$ R$_2$

Odd: | 3 5 9 | | 5 3 13 11 |

R$_1$	R$_2$
2	5
4	4
3	12
5	3
8	13
9	8
	11
	14

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Factors that affect performance

(1)   Tuples of relation stored
              physically together?

(2)   Relations sorted by join attribute?

(3)   Indexes exist?

Example 1(a) NL Join $R_1 \bowtie R_2$

- Relations <u>not</u> contiguous

- Recall
  $\begin{cases} T(R_1) = 10,000 \quad T(R_2) = 5,000 \\ S(R_1) = S(R_2) = 1/10 \text{ block} \\ MEM = 101 \text{ blocks} \end{cases}$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example 1(a)

Nested Loop Join $R_1 \bowtie R_2$

- Relations <u>not</u> contiguous

- Recall $\begin{cases} T(R_1) = 10,000 \quad T(R_2) = 5,000 \\ S(R_1) = S(R_2) = 1/10 \text{ block} \\ MEM = 101 \text{ blocks} \end{cases}$

Cost: for each R_1 tuple:

$$[\text{Read tuple} + \text{Read } R_2]$$

Total $= 10,000 \, [1+500] = 5,010,000$ IOs

- Can we do better?

CS 525

COMPUTER
SCIENCE

Notes 10 - Query Execution

81

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

- # Can we do better?

Use our memory

(1)   Read 100 blocks of R_1

(2)   Read all of R_2 (using 1 block) + join

(3)   Repeat until done

COMPUTER
SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Cost: for each R_1 chunk:

$$\text{Read chunk: } 100 \text{ IOs}$$

$$\text{Read } R_2: \quad \underline{500} \text{ IOs}$$

$$600$$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Cost: for each R_1 chunk:

$$\text{Read chunk: } 100 \text{ IOs}$$

$$\text{Read } R_2: \quad \frac{500}{600} \text{ IOs}$$

$$\text{Total} = \frac{1{,}000}{100} \times 600 = 6{,}000 \text{ IOs}$$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- Can we do better?

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- Can we do better?

    ☛ Reverse join order: $R_2 \bowtie R_1$

    Total = $\dfrac{500}{100}$ x (100 + 1,000) =

    $5 \times 1,100 = 5,500$ IOs

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Cost of Block Nested Loop

☛ Reverse join order: $R_1 \bowtie R_2$

$$\text{Total} = \left\lceil \frac{B(R1)}{M-1} \right\rceil \times (\min(B(R1), M-1) + B(R2))$$

COMPUTER
SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Block-Nested Loop Join (conceptual)

for each M-1 blocks of R_1 do

    read M-1 blocks of R_1 into buffer

    for each block of R_2 do

        read next block of R_2

        for each tuple r in R_1 block

            for each tuple s in R_2 block

                if (r,s) $\vDash$ C then output (r,s)

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Note

- How much memory for buffering inner and for outer chunks?
  - 1 for inner would minimize I/O
  - But, larger buffer better for I/O

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

R₁

M - k	M - k	M - k

R₂

k	k	k	k	k	k

Notes 10 - Query Execution

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example 1(b)   Merge Join

- Both R_1, R_2 ordered by C; relations contiguous

Memory

Notes 10 - Query Execution

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
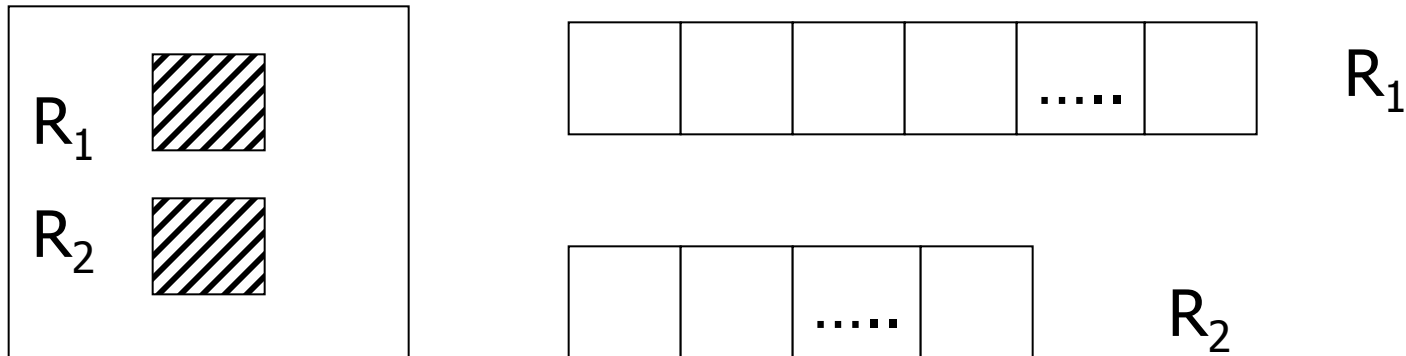
<u>Example 1(b)</u>   Merge Join

- Both R_1, R_2 ordered by C; relations contiguous

<u>Memory</u>
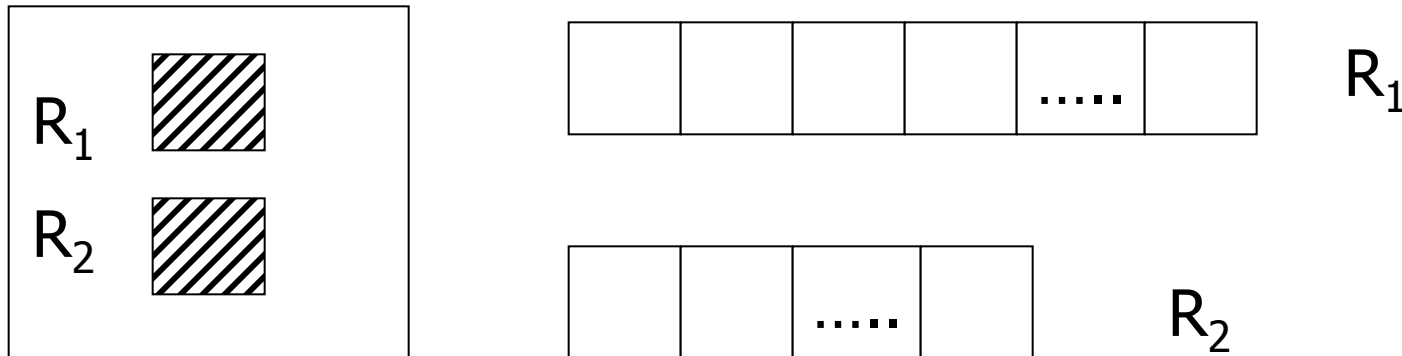
Total cost: Read R_1 cost + read R_2 cost
$$= 1000 + 500 = 1,500 \text{ IOs}$$

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Merge Join Example

$R \bowtie_{B=C} S$

Output: (a,1,1,X)

R

A	B
a	1
b	1
a	2
c	3
d	4
e	5

← Z_R

Z_S →

S

C	D
1	x
2	y
2	e
6	q
7	d

Notes 10 - Query Execution

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
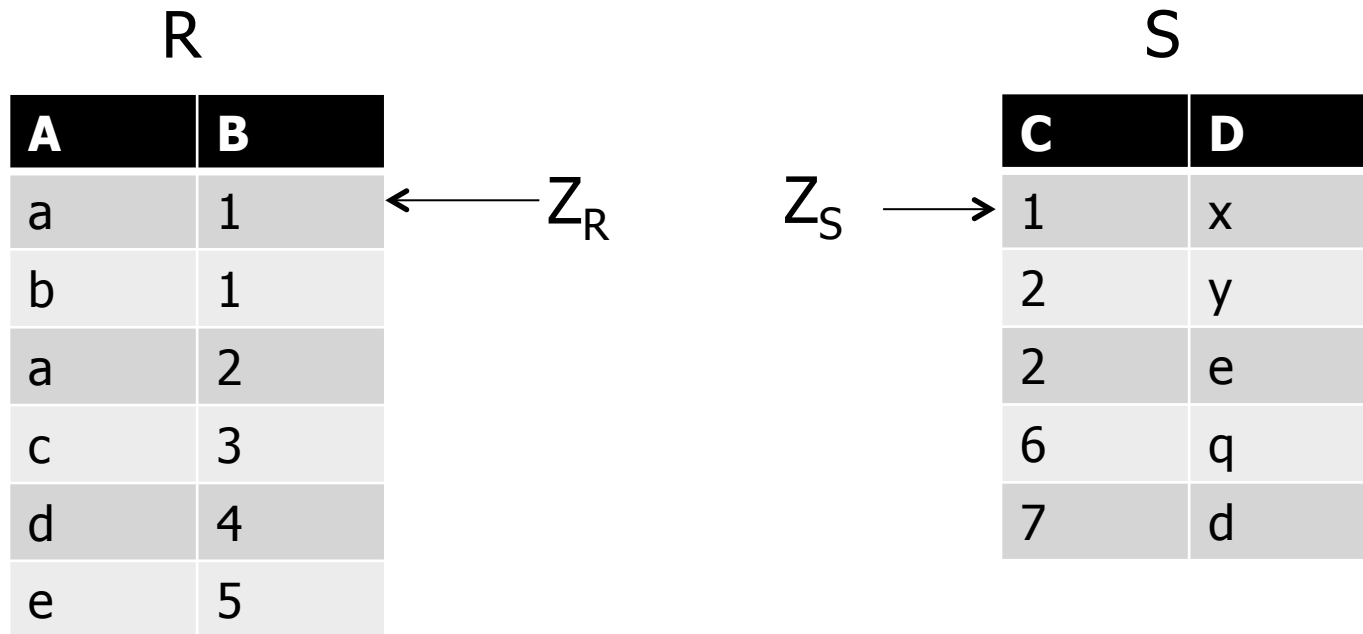
Merge Join Example

$$R \bowtie_{B=C} S$$

Output: (b,1,1,X)

R

A	B
a	1
b	1	← Z_R
a	2
c	3
d	4
e	5

S

C	D
1	x	Z_S →
2	y
2	e
6	q
7	d

Merge Join Example

$$R \bowtie_{B=C} S$$

R.B > S.C: advance Z_S

R

A	B
a	1
b	1
a	2	← Z_R
c	3
d	4
e	5

S

C	D
1	x	Z_S →
2	y
2	e
6	q
7	d

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merge Join Example

$R \bowtie_{B=C} S$

Output: (a,2,2,y)

R

A	B
a	1
b	1
a	2
c	3
d	4
e	5

$\longleftarrow Z_R$

S

C	D
1	x
2	y
2	e
6	q
7	d

$Z_S \longrightarrow$

Merge Join Example

$R \bowtie_{B=C} S$

Output: (a,2,2,e)

R

A	B
a	1
b	1
a	2
c	3
d	4
e	5

$\leftarrow Z_R$

$Z_S \rightarrow$

S

C	D
1	x
2	y
2	e
6	q
7	d

COMPUTER SCIENCE

Notes 10 - Query Execution

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
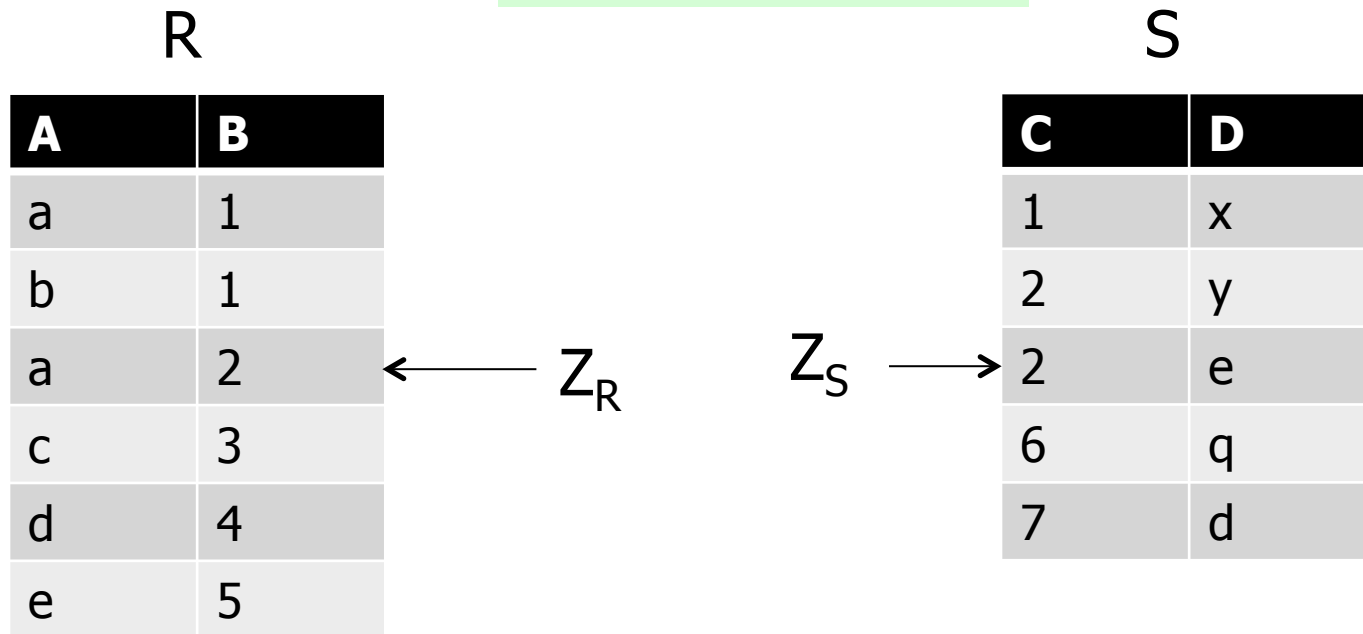
Merge Join Example

$R \bowtie_{B=C} S$

R.B > S.C: advance Z_S

R

A	B
a	1
b	1
a	2
c	3
d	4
e	5

$\longleftarrow Z_R$ (pointing at row c, 3)

S

C	D
1	x
2	y
2	e
6	q
7	d

$Z_S \longrightarrow$ (pointing at row 2, e)

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merge Join Example

$R \bowtie_{B=C} S$

R.B < S.C: advance Z_R

R

A	B
a	1
b	1
a	2
c	3
d	4
e	5

$\longleftarrow Z_R$

$Z_S \longrightarrow$

S

C	D
1	x
2	y
2	e
6	q
7	d

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merge Join Example

$$R \bowtie_{B=C} S$$

R.B < S.C: advance Z_R

R

A	B
a	1
b	1
a	2
c	3
d	4
e	5

$\longleftarrow Z_R$ (pointing to row d, 4)

S

C	D
1	x
2	y
2	e
6	q
7	d

$Z_S \longrightarrow$ (pointing to row 6, q)

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merge Join Example

$$R \bowtie_{B=C} S$$

R.B < S.C: **DONE**

R

A	B
a	1
b	1
a	2
c	3
d	4
e	5

Z_R

S

C	D
1	x
2	y
2	e
6	q
7	d

Z_S

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example 1(c)   Merge Join

- R_1, R_2 <u>not</u> ordered, but contiguous

--> Need to sort R_1, R_2 first

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

One way to sort: Merge Sort

(i) For each 100 blk chunk of R:

- Read chunk
- Sort in memory
- Write to disk

Notes 10 - Query Execution   103

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
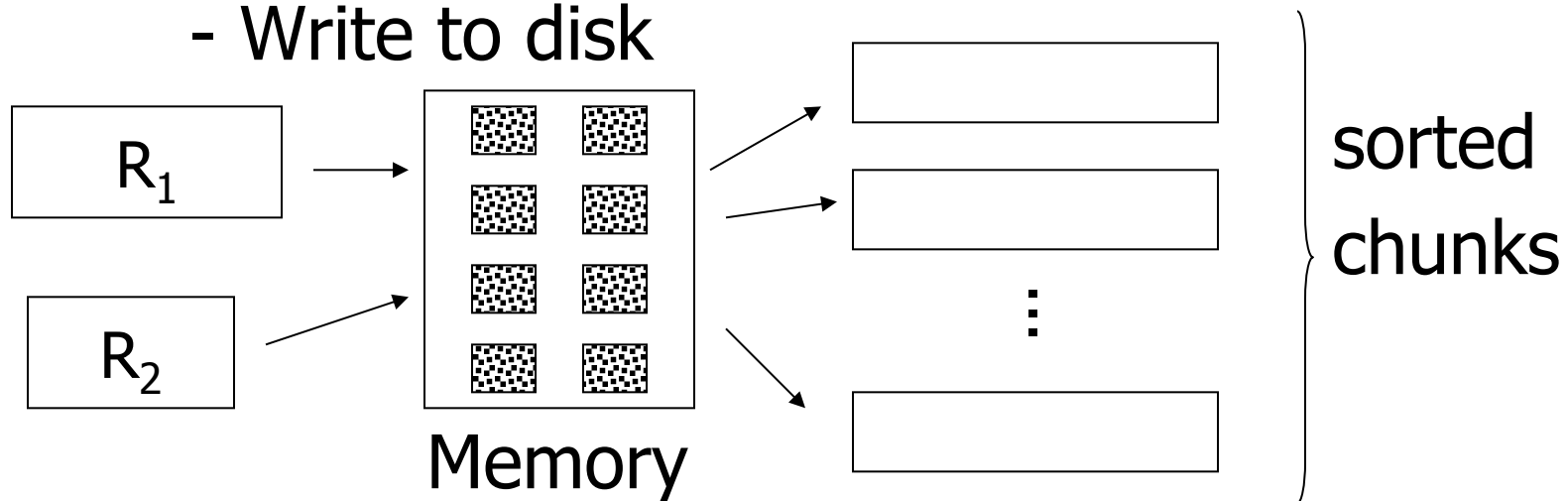
(ii) Read all chunks + merge + write out

Sorted file   Memory   Sorted Chunks

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
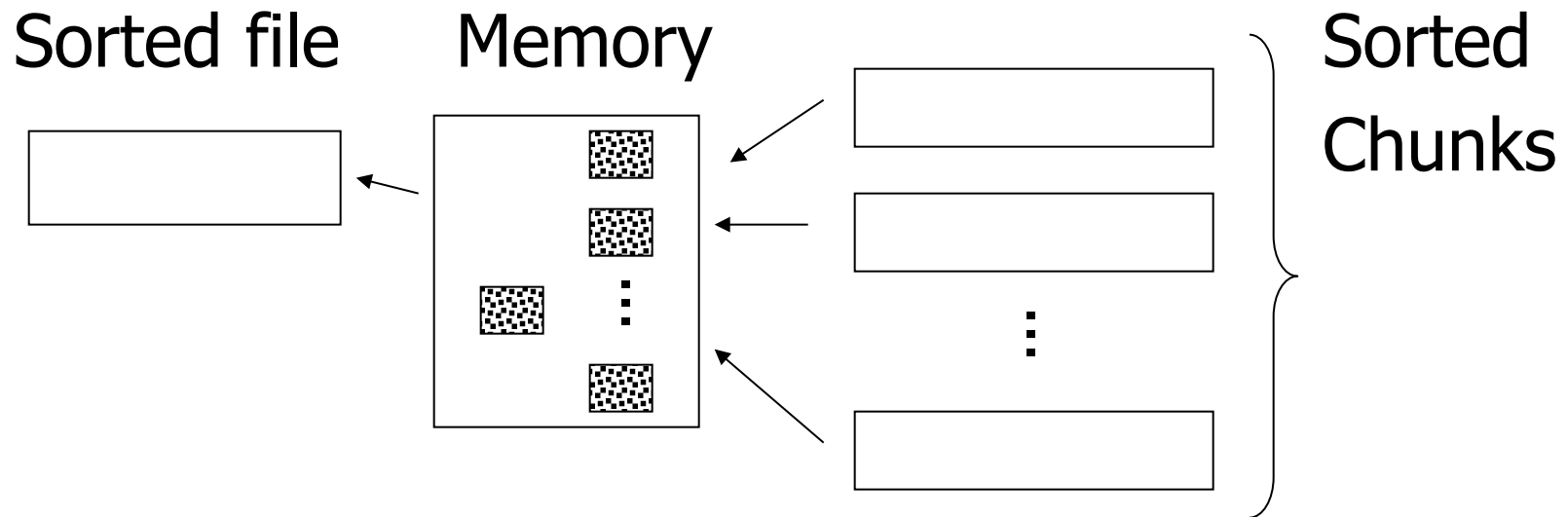
Cost: Sort

Each tuple is read,written,

read, written

so...

Sort cost R_1: 4 x 1,000 = 4,000

Sort cost R_2: 4 x 500   = 2,000

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example 1(d) Merge Join (continued)

R_1, R_2 contiguous, but unordered

Total cost = sort cost + join cost

$$= 6{,}000 + 1{,}500 = 7{,}500 \text{ IOs}$$

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example 1(c) Merge Join (continued)

R_1, R_2 contiguous, but unordered

Total cost = sort cost + join cost

$$= 6,000 + 1,500 = 7,500 \text{ IOs}$$

But: Iteration cost = 5,500
          so merge joint does not pay off!

But say R_1 = 10,000 blocks   contiguous

           R_2 = 5,000 blocks   not ordered

<u>Iterate:</u> $\dfrac{5000}{100}$ x (100+10,000) = 50 x 10,100

$$= 505,000 \text{ IOs}$$

<u>Merge join:</u> 5(10,000+5,000) = 75,000 IOs

Merge Join (with sort) WINS!

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

How much memory do we need for merge sort?

E.g: Say I have 10 memory blocks

$\leftarrow 10 \rightarrow$

R_1
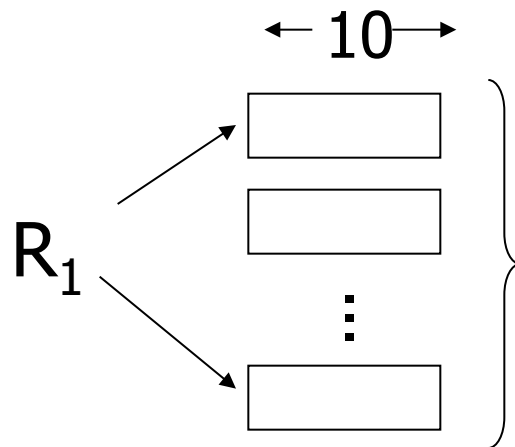
100 chunks $\Rightarrow$ to merge, need 100 blocks!

COMPUTER SCIENCE

Notes 10 - Query Execution

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

In general:

Say k blocks in memory

      x blocks for relation sort

\# chunks = (x/k)   size of chunk = k

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

In general:

Say k blocks in memory

x blocks for relation sort

\# chunks = (x/k)   size of chunk = k

  \# chunks < buffers available for merge

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

In general:

Say k blocks in memory

　　　x blocks for relation sort

chunks = (x/k)　　　size of chunk = k

　# chunks < buffers available for merge

so...　(x/k) ≤ k

or $k^2 \geq x$　　or $k \geq \sqrt{x}$

In our example

R_1 is 1000 blocks, k ≥ 31.62
R_2 is 500 blocks,   k ≥ 22.36

Need at least 32 buffers

Again: in practice we would not want to use only one buffer per run!

Notes 10 - Query Execution   113

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Can we improve on merge join?

Hint: do we really need the fully sorted files?

sorted runs

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
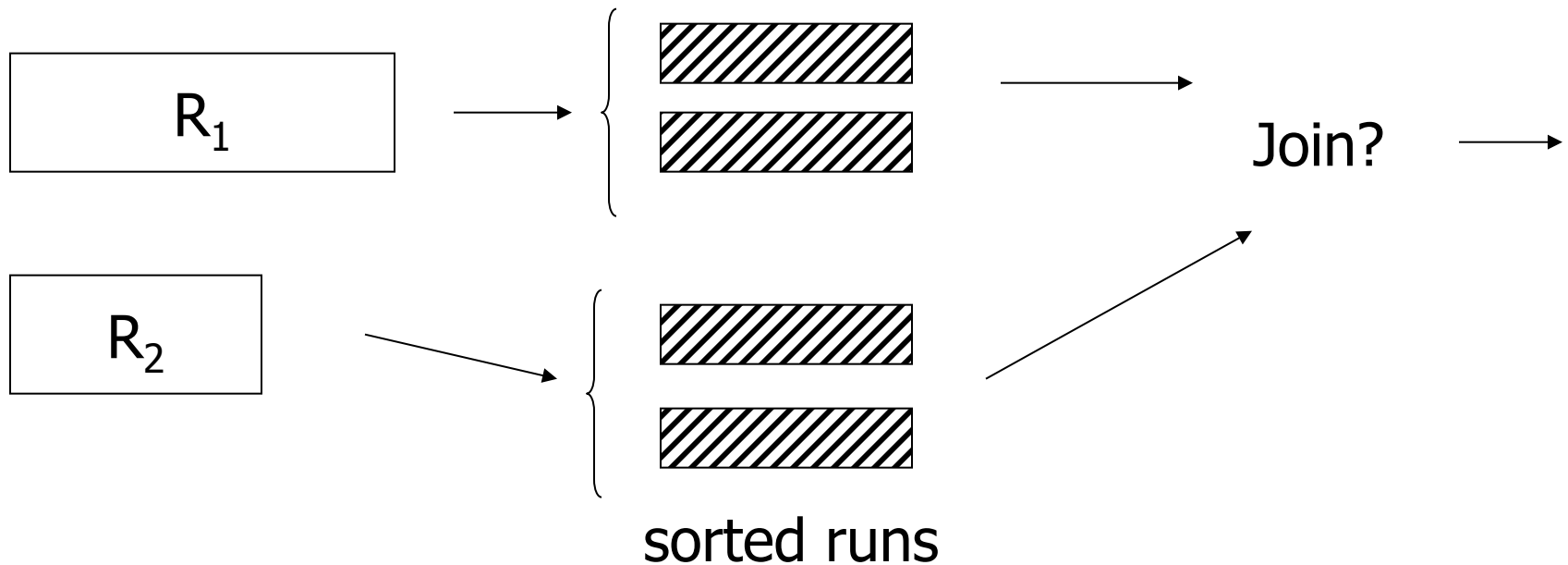
Cost of improved merge join:

C = Read R_1 + write R_1 into runs
  + read R_2 + write R_2 into runs
  + join
  = 2,000 + 1,000 + 1,500 = 4,500

--> Memory requirement?

Notes 10 - Query Execution

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example 1(d) Index Join

- Assume $R_1.C$ index exists; 2 levels
- Assume R_2 contiguous, unordered

- Assume $R_1.C$ index fits in memory

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Cost: Reads: 500 IOs

for each R_2 tuple:

- probe index - free

- if match, read R_1 tuple: 1 IO

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

What is expected # of matching tuples?

(a) say $R_1.C$ is key, $R_2.C$ is foreign key

  then expect = 1

(b) say $V(R_1,C) = 5000$, $T(R_1) = 10,000$
  with uniform assumption
  expect = 10,000/5,000   = 2

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

What is expected # of matching tuples?

(c) Say $DOM(R_1, C) = 1,000,000$

$$T(R_1) = 10,000$$

with alternate assumption

$$\text{Expect} = \frac{10,000}{1,000,000} = \frac{1}{100}$$

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Total cost with index join

(a) Total cost = 500+5000(1)1 = 5,500

(b) Total cost = 500+5000(2)1 = 10,500

(c) Total cost = 500+5000(1/100)1=550

COMPUTER SCIENCE

Notes 10 - Query Execution

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

What if index does not fit in memory?

Example: say R_1.C index is 201 blocks

- Keep root + 99 leaf nodes in memory
- Expected cost of each probe is

$$E = (0)\frac{99}{200} + (1)\frac{101}{200} \approx 0.5$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Total cost (including probes)

$= 500+5000$ [Probe + get records]

$= 500+5000$ [0.5+2]   uniform assumption

$= 500+12{,}500 = 13{,}000$ (case b)

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Total cost (including probes)

= 500+5000 [Probe + get records]

= 500+5000 [0.5+2]   uniform assumption

= 500+12,500 = 13,000   (case b)

For case (c):

= 500+5000[0.5 $\times$ 1 + (1/100) $\times$ 1]

= 500+2500+50 = 3050 IOs

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

So far

$$
\left\{
\begin{array}{ll}
\text{Nested Loop} & 5500 \\
\text{Merge join} & 1500 \\
\text{Sort+Merge Join} & 7500 \rightarrow 4500 \\
R_1.C \text{ Index} & 5500 \rightarrow 3050 \rightarrow 550 \\
R_2.C \text{ Index} & \underline{}
\end{array}
\right.
$$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example 1(e)   Partition Hash Join

- R_1, R_2 contiguous (un-ordered)
- → Use 100 buckets
- → Read R_1, hash, + write buckets

R_1 →

100

10 blocks

COMPUTER SCIENCE

Notes 10 - Query Execution

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
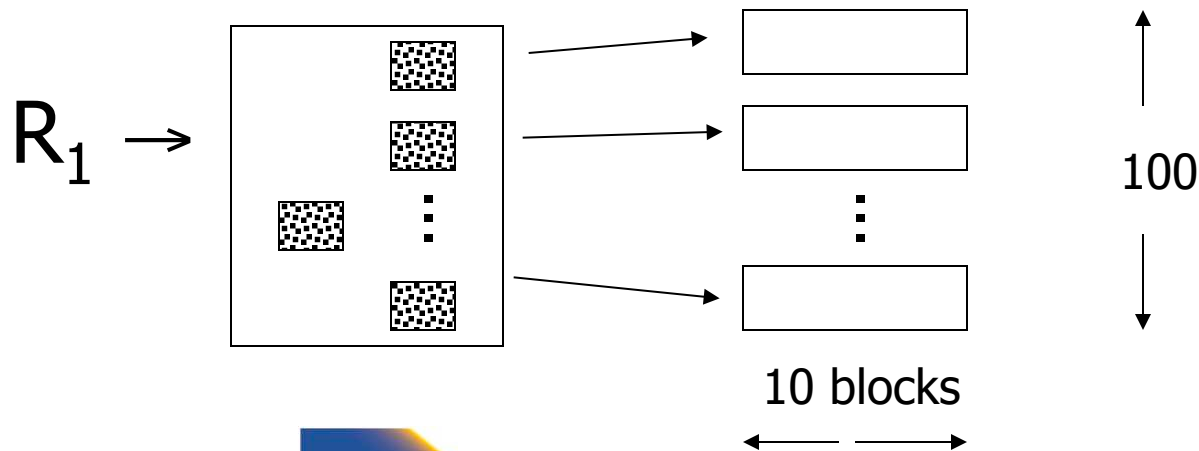
-> Same for R_2

-> Read one R_1 bucket; build memory hash table
  -using different hash function h'

-> Read corresponding R_2 bucket + hash probe

R_1

memory

R_2
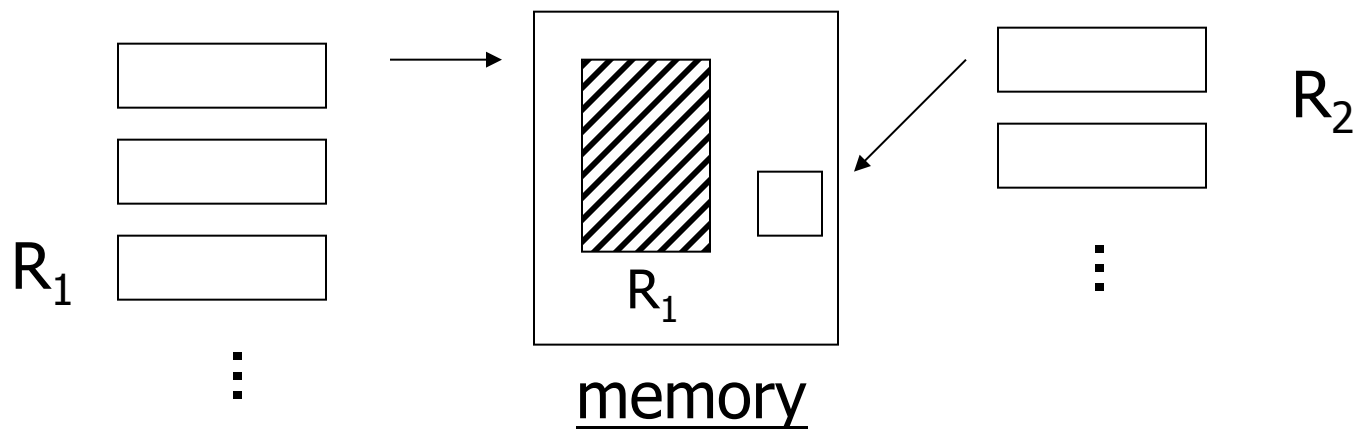
➭ Then repeat for all buckets

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Cost:

"Bucketize:"   Read R_1 + write

Read R_2 + write

Join: Read R_1, R_2

Total cost = 3 x [1000+500] = 4500

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Cost:

"Bucketize:"   Read R_1 + write

Read R_2 + write

Join: Read R_1, R_2

Total cost = 3 x [1000+500] = 4500

> Note: this is an approximation since buckets will vary in size and we have to round up to blocks

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Why is Hash Join good?

Notes 10 - Query Execution

129

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
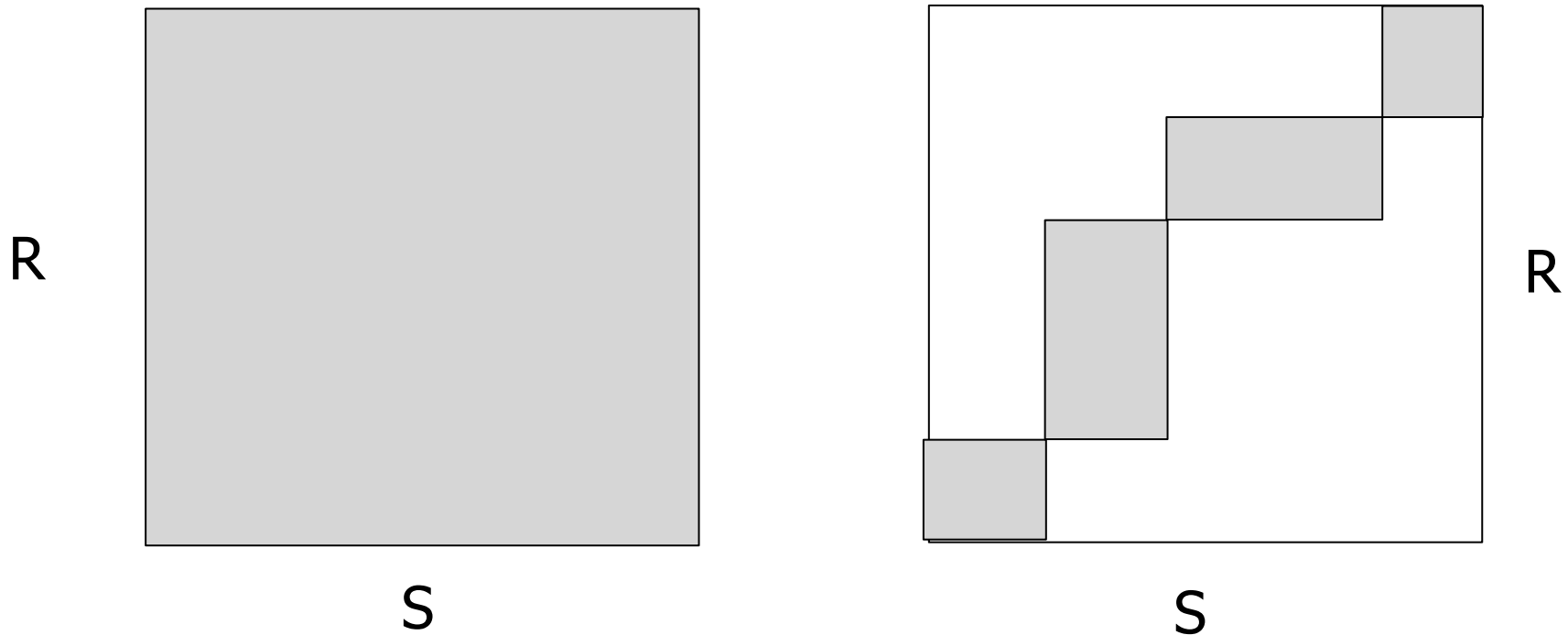
Minimum memory requirements:

Size of R_1 bucket = (x/k)

k = number of memory buffers

x = number of R_1 blocks

So... $(x/k) < k$

$k > \sqrt{x}$ need: k+1 total memory
buffers

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Can we use Hash-join when buckets do not fit into memory?:

- Treat buckets as relations and apply Hash-join recursively

join

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
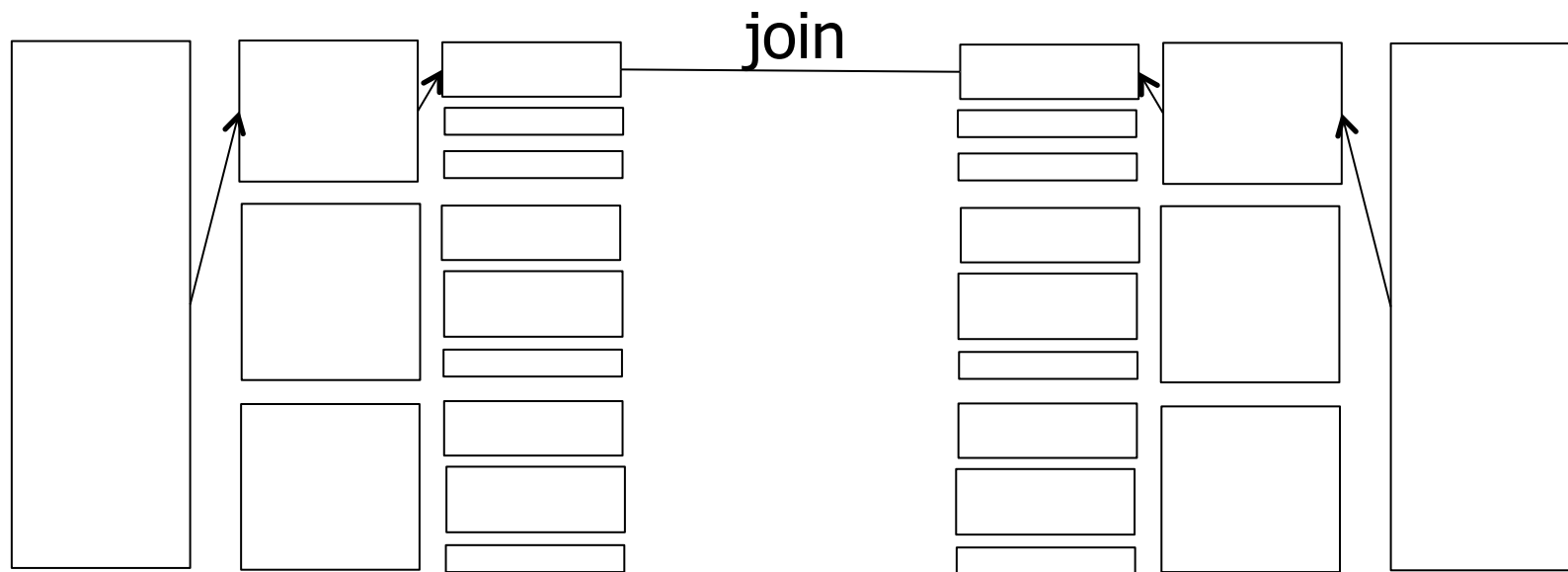
Duality Hashing-Sorting

- Both partition inputs
- Until input fits into memory
- Logarithmic number of phases in memory size

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Trick: keep some buckets in memory

E.g., $k' = 33$ R_1 buckets = 31 blocks
                    keep 2 in memory

called hybrid hash-join

Trick: keep some buckets in memory

E.g., k' =33 R_1 buckets = 31 blocks
keep 2 in memory

memory

R_1

in

G_0

G_1

31

33-2=31

Memory use:
G_0 31 buffers
G_1 31 buffers
Output   33-2 buffers
R_1 input 1
Total   94 buffers
        6 buffers to spare!!

called hybrid hash-join

COMPUTER SCIENCE
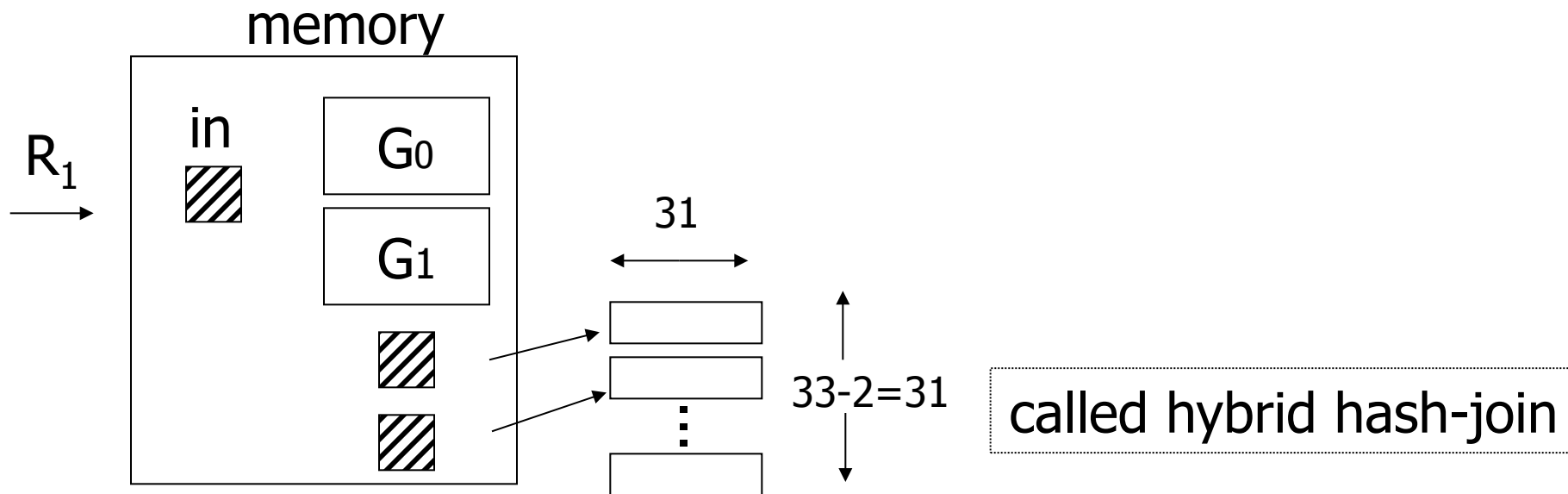
IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
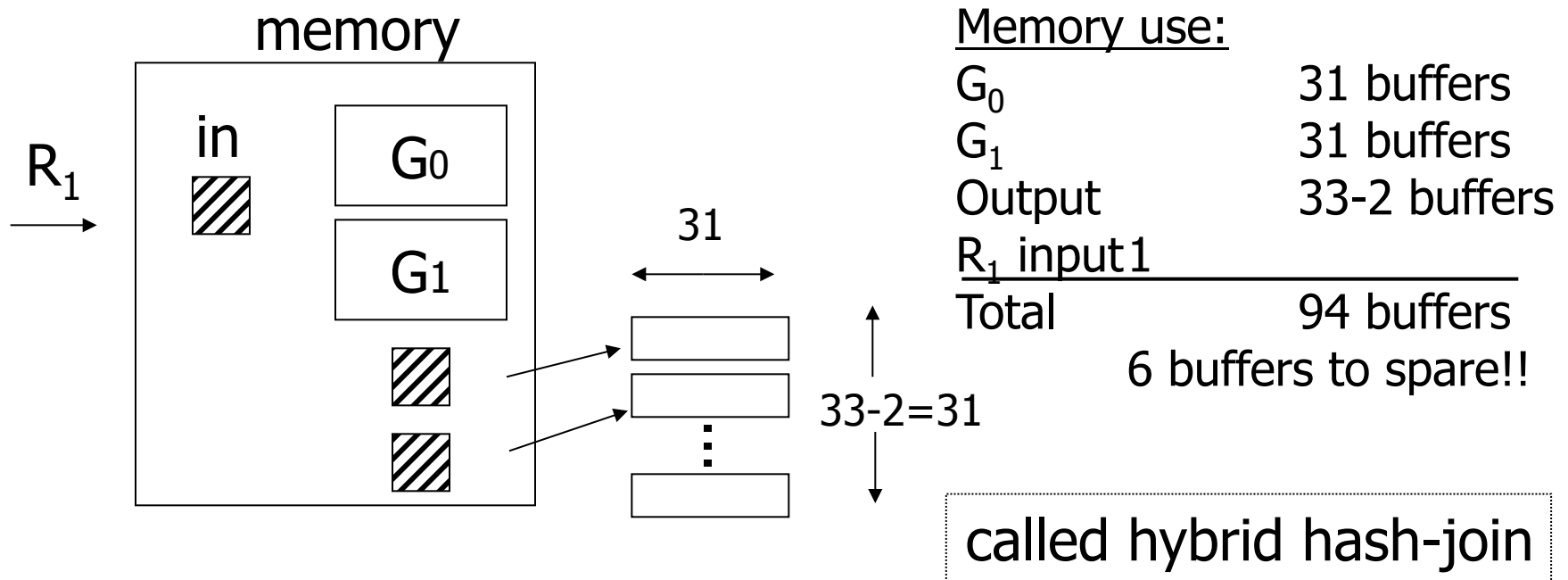
Next: Bucketize R_2

- R_2 buckets $= 500/33 = 16$ blocks
- Two of the R_2 buckets joined immediately with G_0, G_1
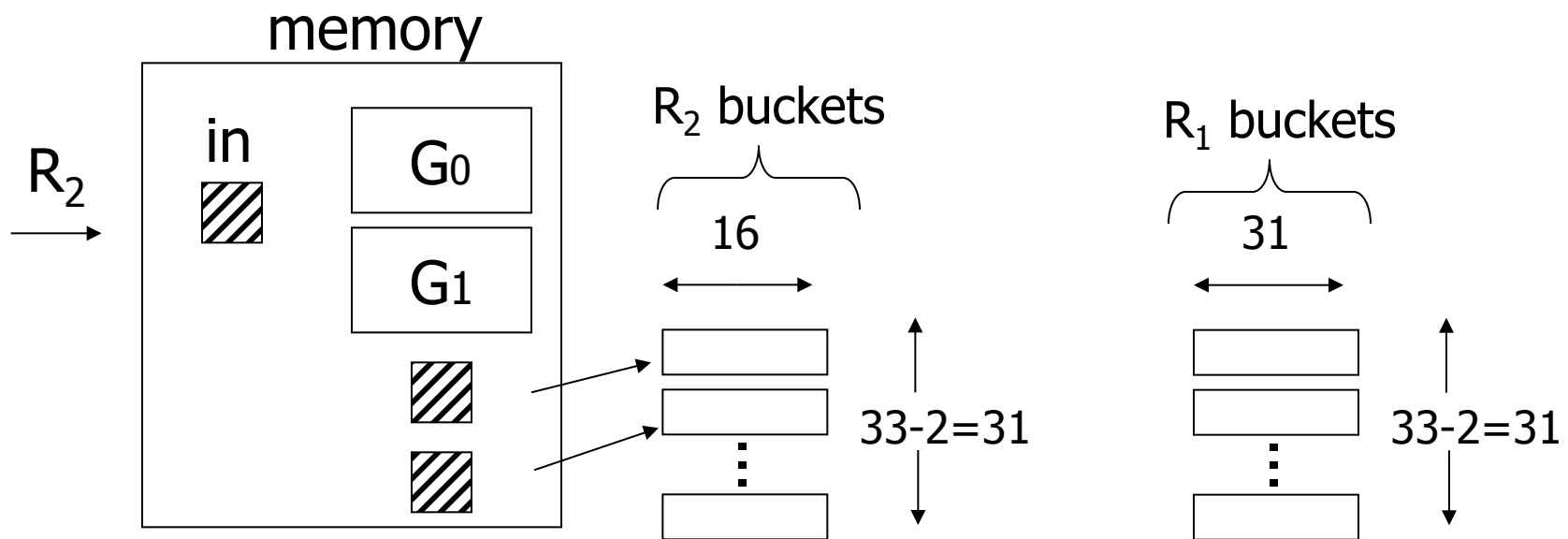
Finally: Join remaining buckets

– for each bucket pair:

- read one of the buckets into memory
- join with second bucket

memory

ans

out

one full R2 bucket

G_i

one R1 buffer

R_2 buckets

16

33-2=31

R_1 buckets

31

33-2=31

COMPUTER SCIENCE

Notes 10 - Query Execution

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
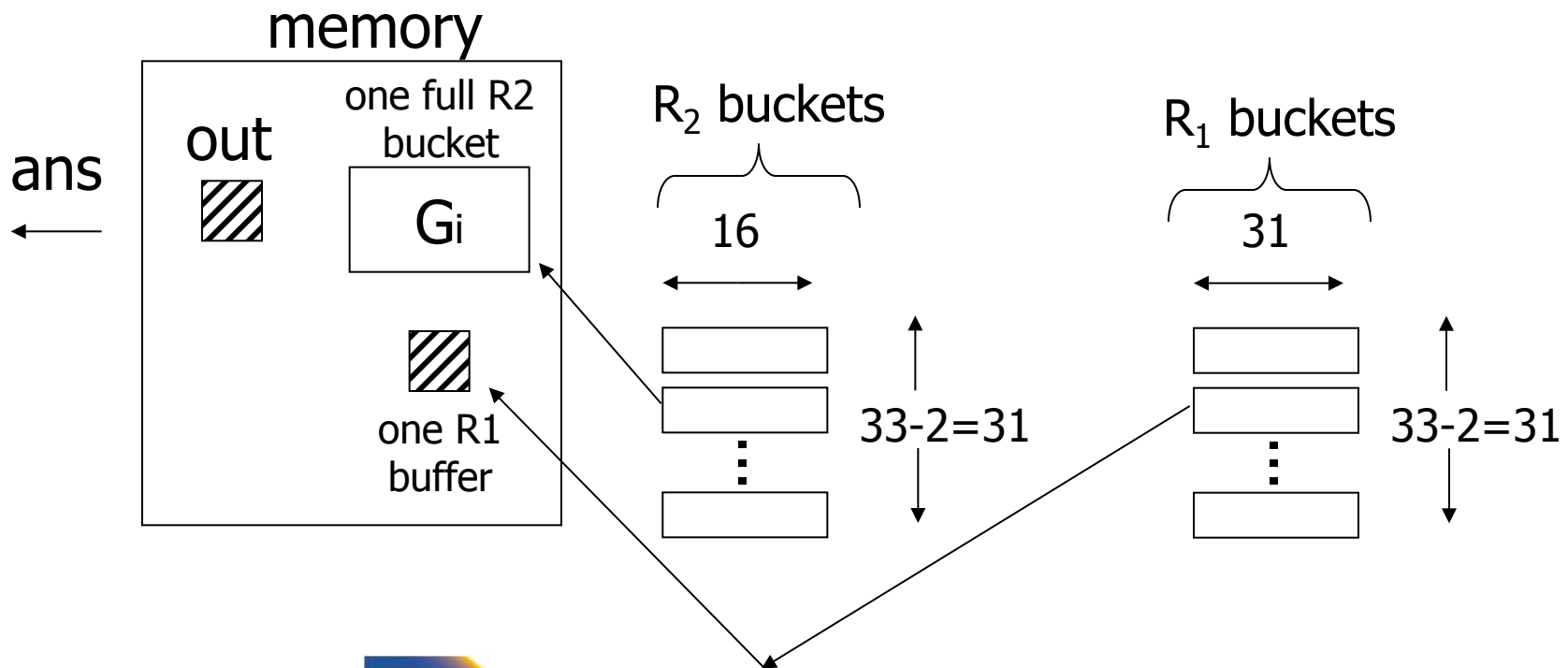
Cost

- Bucketize R_1 = 1000+31×31=1961
- To bucketize R_2, only write 31 buckets:
  so, cost = 500+31×16=996
- To compare join (2 buckets already done)
  read 31×31+31×16=1457

Total cost = 1961+996+1457 = 4414

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

• How many buckets in memory?

OR...

?
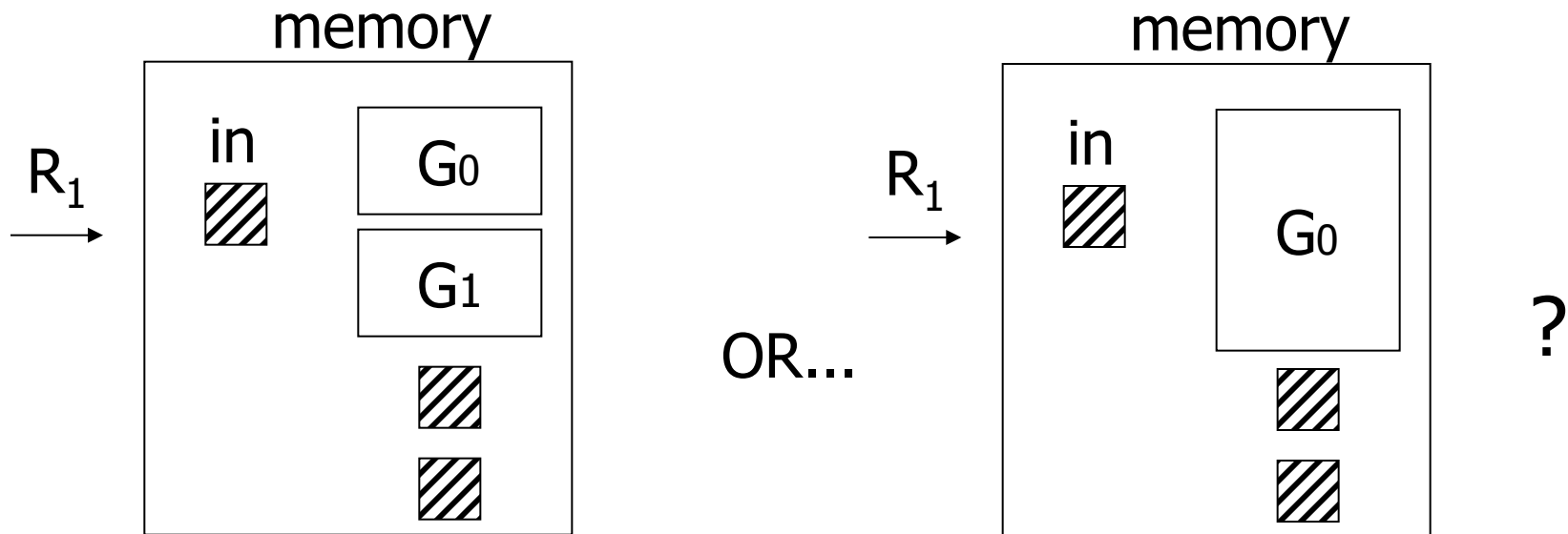
☛ See textbook for answer...

Another hash join trick:

- Only write into buckets
  <val,ptr> pairs

- When we get a match in join phase,
  must fetch tuples

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

- To illustrate cost computation, assume:
  - 100 <val,ptr> pairs/block
  - expected number of result tuples is 100

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

- To illustrate cost computation, assume:
  - 100 <val,ptr> pairs/block
  - expected number of result tuples is 100

- Build hash table for R_2 in memory

  5000 tuples $\rightarrow$ 5000/100 = 50 blocks
- Read R_1 and match
- Read ~ 100 R_2 tuples

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

- To illustrate cost computation, assume:
  - 100 <val,ptr> pairs/block
  - expected number of result tuples is 100

- Build hash table for R_2 in memory

       5000 tuples $\rightarrow$ 5000/100 = 50 blocks

- Read R_1 and match

- Read ~ 100 R_2 tuples

$$
\begin{array}{llr}
\underline{\text{Total cost}} = & \text{Read } R_2: & 500 \\
& \text{Read } R_1: & 1000 \\
& \text{Get tuples:} & \underline{100} \\
& & 1600
\end{array}
$$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

So far:

Iterate	5500
Merge join	1500
Sort+merge joint	7500
R_1.C index	5500 → 550
R_2.C index	_____
Build R_1.C index	_____
Build R_2.C index	_____
Hash join	4500+
with trick,R_1 first	4414
with trick,R_2 first	_____
Hash join, pointers	1600

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Yet another hash join trick:

- Combine the ideas of
  - block nested-loop with hash join
- Use memory to build hash-table for one chunk of relation
- Find join partners in $O(1)$ instead of $O(M)$
- Trade-off
  - Space-overhead of hash-table
  - Time savings from look-up

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Summary

- Nested Loop ok for "small" relations
      (relative to memory size)
  - Need for complex join condition
- For equi-join, where relations not
      sorted and no indexes exist,
                  hash join usually best

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

- Sort + merge join good for non-equi-join (e.g., $R_1.C > R_2.C$)
- If relations already sorted, use merge join
- If index exists, it <u>could</u> be useful

    (depends on expected result size)

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Join Comparison

N_i = number of tuples in R_i

$B(R_i)$ = number of blocks of R_i

$\#P$ = number of partition steps for hash join

P_{ij} = average number of join partners

Algorithm	#I/O	Memory	Disk Space
Nested Loop (block)	$B(R_1) / (M-1) *$ `[min(B(R),M−1)` `+ B(R_2)]`	3	0
Index Nested Loop	$B(R_1) + N_1 * P_{12}$	B(Index) + 2	0
Merge (sorted)	$B(R_1) + B(R_2)$	Max tuples =	0
Merge (unsorted)	$B(R_1) + B(R_2) +$ (sort − 1 pass)	sort	$B(R_1) + B(R_2)$
Hash	$(2\#P + 1)(B(R_1) +$ $B(R_2))$	root(max(B(R_1), B(R_2)), #P + 1)	$\sim B(R_1) + B(R_2)$

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Why do we need nested loop?

- Remember not all join implementations work for all types of join conditions

Algorithm	Type of Condition	Example
Nested Loop	any	a LIKE '%hello%'
Index Nested Loop	Supported by index: Equi-join (hash) Equi or range (B-tree)	a = b a < b
Merge	Equalities and ranges	a < b, a = b AND c = d
Hash	Equi-join	a = b

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Outer Joins

- How to implement (left) outer joins?

- Nested Loop and Merge

  – Use a flag that is set to true if we find a match for an outer tuple

  – If flag is false fill with NULL

- Hash

  – If no matching tuple fill with NULL

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Merge Left Outer Join

$$R \bowtie_{B=C} S$$

Output: (a,1,1,X)

R

A	B
a	1
d	4
e	5

$\longleftarrow Z_R$

S

$Z_S \longrightarrow$

C	D
1	x
2	y
2	e
6	q
7	d

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merge Left Outer Join

$$R \bowtie_{B=C} S$$

R

A	B
a	1
d	4
e	5

$\leftarrow$ Z_R
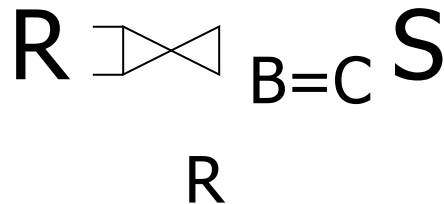
No match for (d,4)
Output: (d,4,NULL,NULL)

C	D
1	x
2	y
2	e
6	q
7	d

$Z_S \longrightarrow$

Merge Left Outer Join

$$R \bowtie_{B=C} S$$
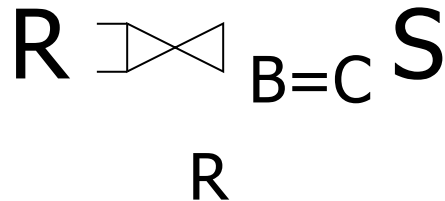
No match for (e,5)
Output: (e,5,NULL,NULL)

R

A	B
a	1
d	4
e	5	← Z_R

C	D
1	x
2	y
2	e
Z_S → 6	q
7	d

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Operators Overview

- (External) Sorting
- Joins (Nested Loop, Merge, Hash, …)
- Aggregation (Sorting, Hash)
- Selection, Projection (Index, Scan)
- Union, Set Difference
- Intersection
- Duplicate Elimination

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Aggregation

- Have to compute aggregation functions
  - for each group of tuples from input
- Groups
  - Determined by equality of group-by attributes

Aggregation Example

```
SELECT sum(a),b
FROM R
GROUP BY b
```

a	b
3	1
4	2
3	1
1	2
1	2

sum(a)	b
6	1
6	2

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Aggregation Function Interface

- `init()`
  - Initialize state
- `update(tuple)`
  - Update state with information from tuple
- `close()`
  - Return result and clean-up

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Implementation SUM(A)

- `init()`
  - `sum := 0`
- `update(tuple)`
  - `sum += tuple.A`
- `close()`
  - **`return`** `sum`

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Aggregation Implementations

- Sorting

  - Sort input on group-by attributes

  - On group boundaries output tuple

- Hashing

  - Store current aggregated values for each group in hash table

  - Update with newly arriving tuples

  - Output result after processing all inputs

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Grouping by sorting

- Similar to Merge join

- Sort R on group-by attribute

- Scan through sorted input
  - If group-by values change
    - Output using close() and call init()
  - Otherwise
    - Call update()

Aggregation Example

SELECT sum(a),b
FROM R
GROUP BY b

sort

init()

a	b
3	1
4	2
3	1
1	2
1	2

a	b
3	1
3	1
4	2
1	2
1	2

0

COMPUTER SCIENCE

Notes 10 - Query Execution

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Aggregation Example

```
SELECT sum(a),b
FROM R
GROUP BY b
```

a	b
3	1
3	1
4	2
1	2
1	2

update(3,1)

3

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Aggregation Example

```
SELECT sum(a),b
FROM R
GROUP BY b
```

a	b
3	1
3	1	←
4	2
1	2
1	2

update(3,1)

6

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Aggregation Example

```
SELECT sum(a),b
FROM R
GROUP BY b
```

a	b
3	1
3	1
4	2
1	2
1	2

Group by changed!
close(), init(), update(4,2)

① 6

② 0

③ 4

output

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Grouping by Hashing

- Create in-memory hash-table
- For each input tuple probe hash table with group by values
  - If no entry exists then call init(), update(), and add entry
  - Otherwise call update() for entry
- Loop through all entries in hash-table and ouput calling close()

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Aggregation Example

```
SELECT sum(a),b
FROM R
GROUP BY b
```

a	b
3	1
4	2
3	1
1	2
1	2
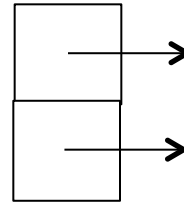
Aggregation Example

```
SELECT sum(a),b
FROM R
GROUP BY b
```

Init() and update(3,1)

a	b
3	1
4	2
3	1
1	2
1	2

3

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
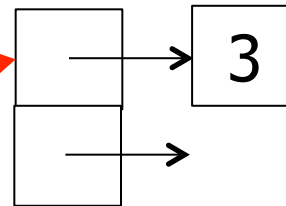
Aggregation Example

```
SELECT sum(a),b
FROM R
GROUP BY b
```

Init() and update(4,2)

a	b
3	1
4	2
3	1
1	2
1	2

Notes 10 - Query Execution

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
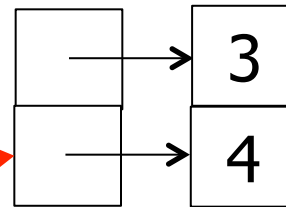
Aggregation Example

SELECT sum(a),b
FROM R
GROUP BY b

update(3,1)

a	b
3	1
4	2
3	1
1	2
1	2

6

4
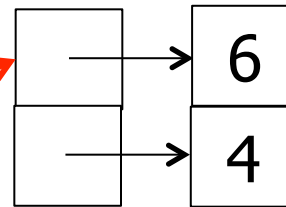
Aggregation Example

SELECT sum(a),b
FROM R
GROUP BY b

- Loop through hash table entries
- Output tuples

a	b
3	1
4	2
3	1
1	2
1	2

	6
	6

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
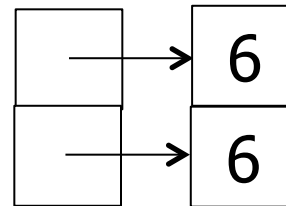
Aggregation Summary

- Hashing

  - No sorting -> no extra I/O

  - Hash table has to fit into memory

  - No outputs before all inputs have been processed

- Sorting

  - No memory required

  - Output one group at a time

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Operators Overview

- (External) Sorting
- Joins (Nested Loop, Merge, Hash, …)
- Aggregation (Sorting, Hash)
- Selection, Projection (Index, Scan)
- Union, Set Difference
- Intersection
- Duplicate Elimination

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate Elimination

- Equivalent to group-by on all attributes

- -> Can use aggregation implementations

- Optimization
  - Hash
    - Directly output tuple and use hash table only to avoid outputting duplicates

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Operators Overview

- (External) Sorting
- Joins (Nested Loop, Merge, Hash, ...)
- Aggregation (Sorting, Hash)
- Selection, Projection (Index, Scan)
- Union, Set Difference
- Intersection
- Duplicate Elimination

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Set Operations

- Can be modeled as join
  - with different output requirements
- As aggregation/group by on all columns
  - with different output requirements

Union

- Bag union
  - Append the two inputs
  - E.g., using three buffers
- Set union
  - Apply duplicate removal to result

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Intersection

- Set version
  - Equivalent to join + project + duplicate removal
  - 3-state aggregate function (found left, found right, found both)

- Bag version
  - Join + project + min(i,j)
  - Aggegate min(count(i),count(j))

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Set Difference

- Using join methods
  - Find matching tuples
  - If no match found, then output
- Using aggregation
  - count(i) − count(j) **(bag)**
  - true(i) AND false(j) **(set)**

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Summary

- Operator implementations
  - Joins!
  - Other operators
- Cost estimations
  - I/O
  - memory
- Query processing architectures

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Next

- Query Optimization Physical
- -> How to **efficiently** choose an **efficient** plan

CS 525: Advanced Database Organization

11: **Query Optimization Physical**

Boris Glavic

Slides: adapted from a course taught by
Hector Garcia-Molina, Stanford InfoLab

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

SQL query

parse

parse tree

convert

logical query plan

apply laws

"improved" l.q.p

estimate result sizes

l.q.p. +sizes

consider physical plans

{P1,P2,.....}

answer

execute

statistics

Pi

pick best

{(P1,C1),(P2,C2)...}

estimate costs

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
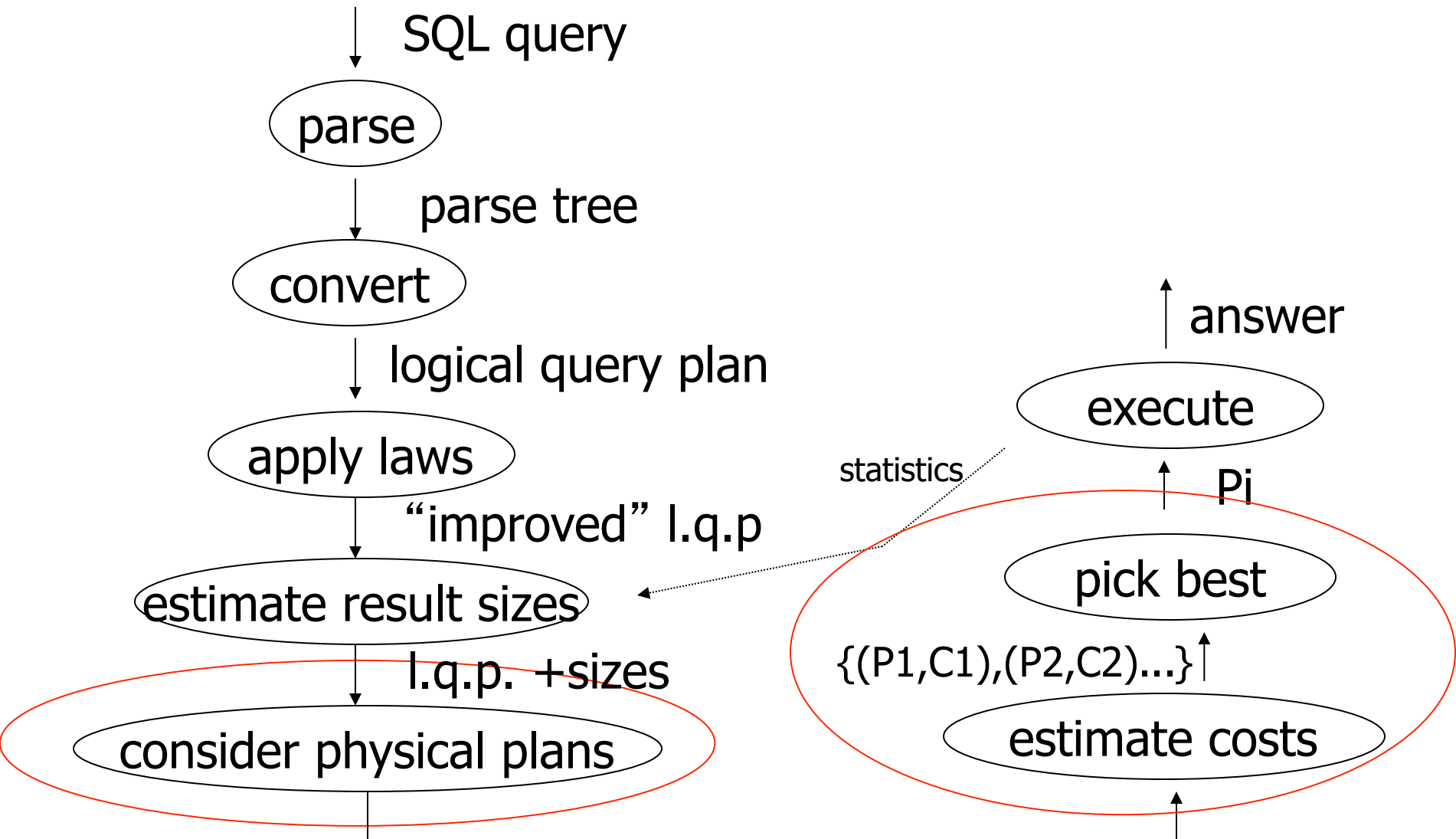
Cost of Query

- Parse + Analyze
- Optimization – Find plan
- Execution
- Return results to client

Cost of Query

- Parse + Analyze
  - Can parse MB of SQL code in milisecs
- **Optimization – Find plan**
  - **Generating plans, costing plans**
- **Execution**
  - **Execute plan**
- Return results to client
  - Can be expensive but not discussed here

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Physical Optimization

- Apply after applying heuristics in logical optimization

- 1) Enumerate potential execution plans
  - All?
  - Subset

- 2) Cost plans
  - What cost function?

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Physical Optimization

- To apply pruning in the search for the best plan
  - Steps 1 and 2 have to be interleaved
  - Prune parts of the search space
    - if we know that it cannot contain any plan that is better than what we found so far

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example Query

```
SELECT e.name
FROM Employee e,
     EmpDep ed,
     Department d
WHERE e.name = ed.emp
      AND ed.dep = d.dep
      AND d.dep = 'CS'
```

π_{name}

$\bowtie_{\text{dep=dep}}$

$\bowtie_{\text{name=emp}}$

$\sigma_{\text{dep=CS}}$

Employee   EmpDep   Department

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
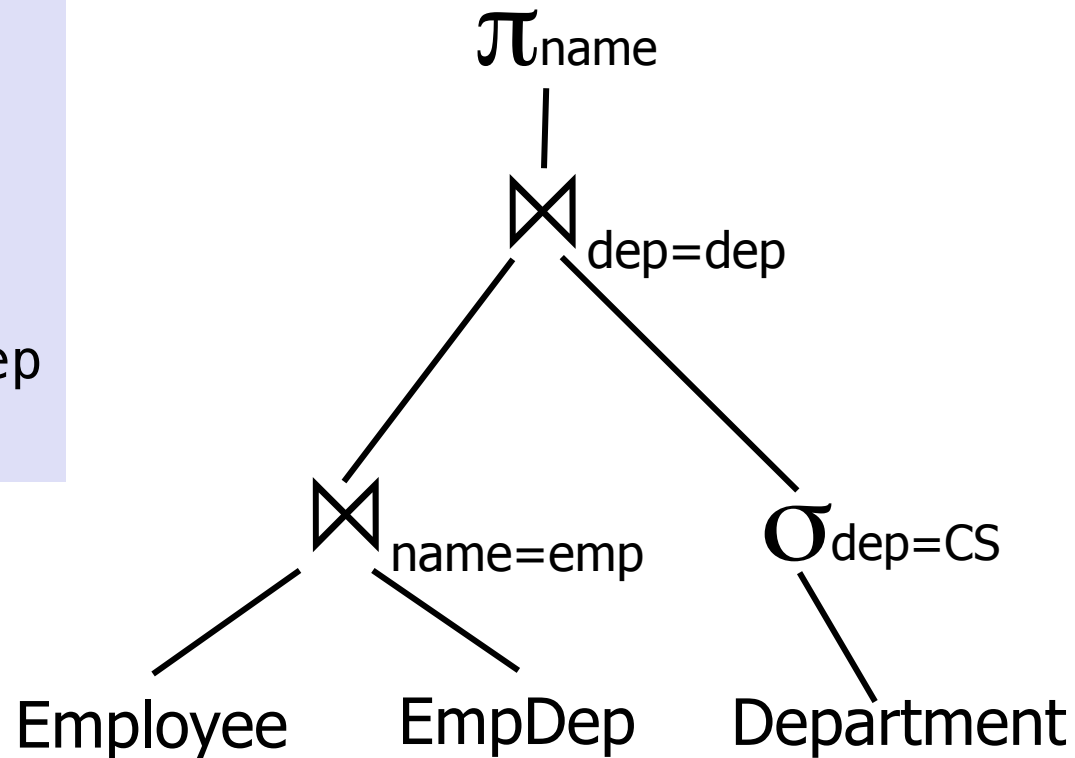
Example Query – Possible Plan

SELECT e.name
FROM Employee e,
     EmpDep ed,
     Department d
WHERE e.name = ed.emp
     AND ed.dep = d.dep
     AND d.dep = 'CS'
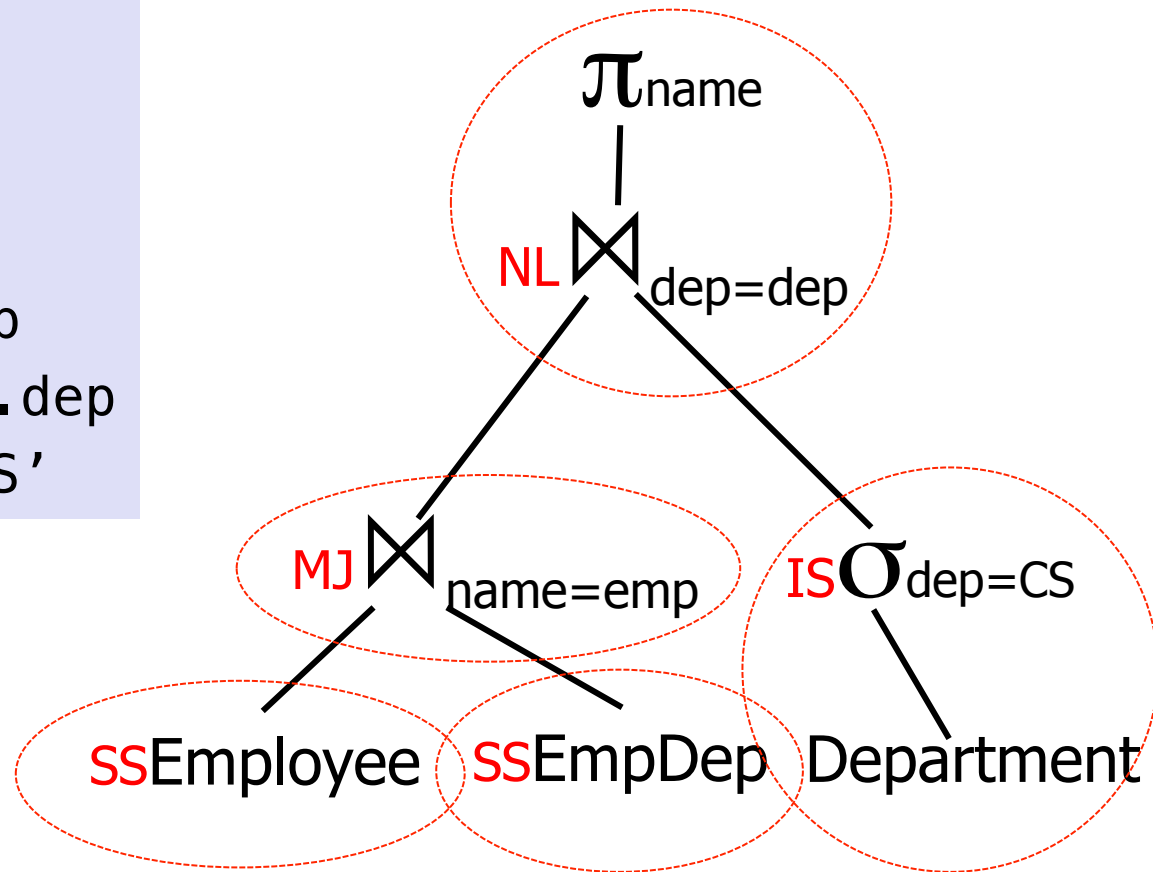
Cost Model

- Cost factors
  - **#disk I/O**
  - **CPU cost**
  - Response time
  - Total **execution time**
- Cost of operators
  - I/O as discussed in query execution (part 10)
  - Need to know **size of intermediate results** (part 09)

COMPUTER SCIENCE

Notes 11 - Physical Optimization

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example Query – Possible Plan

```
SELECT e.name
FROM Employee e,
     EmpDep ed,
     Department d
WHERE e.name = ed.emp
      AND ed.dep = d.dep
      AND d.dep = 'CS'
```

Cost?
Need input size!

π_{name}

NL $\bowtie_{dep=dep}$

MJ $\bowtie_{name=emp}$ IS $\sigma_{dep=CS}$

SSEmployee   SSEmpDep   Department
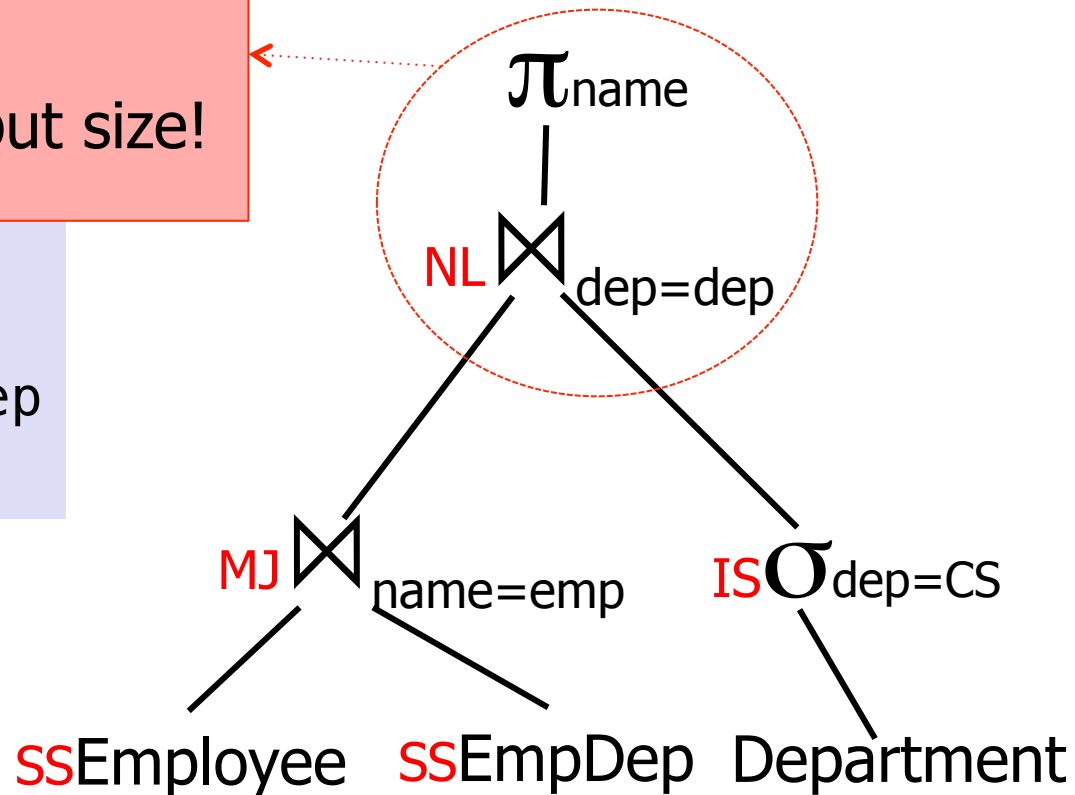
Cost Model Trade-off

- **Precision**
  - Incorrect cost-estimation -> choose suboptimal plan

- **Cost of computing cost**
  - Cost of costing a plan
    - We may have to cost millions or billions of plans
  - Cost of maintaining statistics
    - Occupies resources needed for query processing

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Plan Enumeration

- For each operator in the query
  - Several implementation options
- Binary operators (joins)
  - Changing the order may improve performance a lot!
- -> consider both **different implementations** and **order of operators** in plan enumeration

Example Join Ordering Result Sizes

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
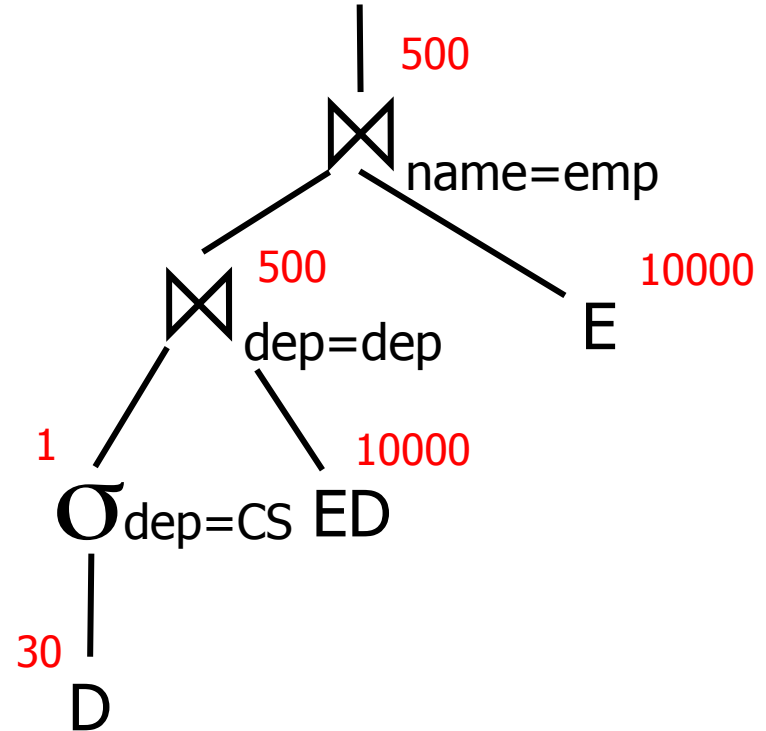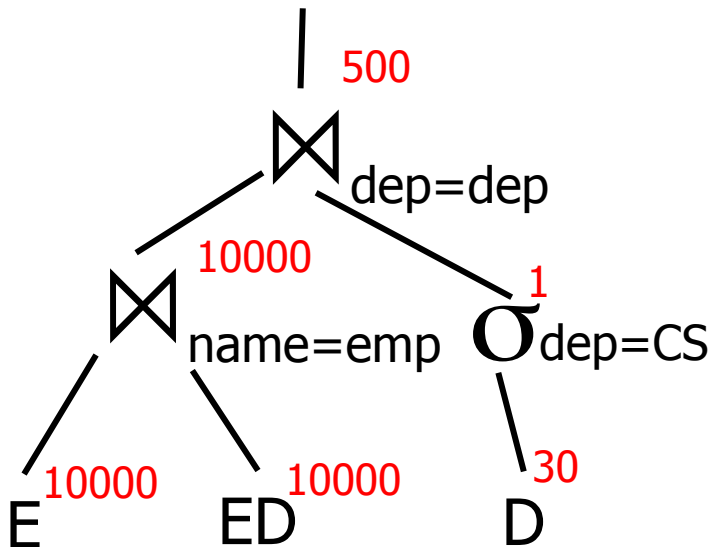
Example Join Ordering Cost (only NL)

S(E) = S(ED) = S(D) = 1/10 block
M = 101
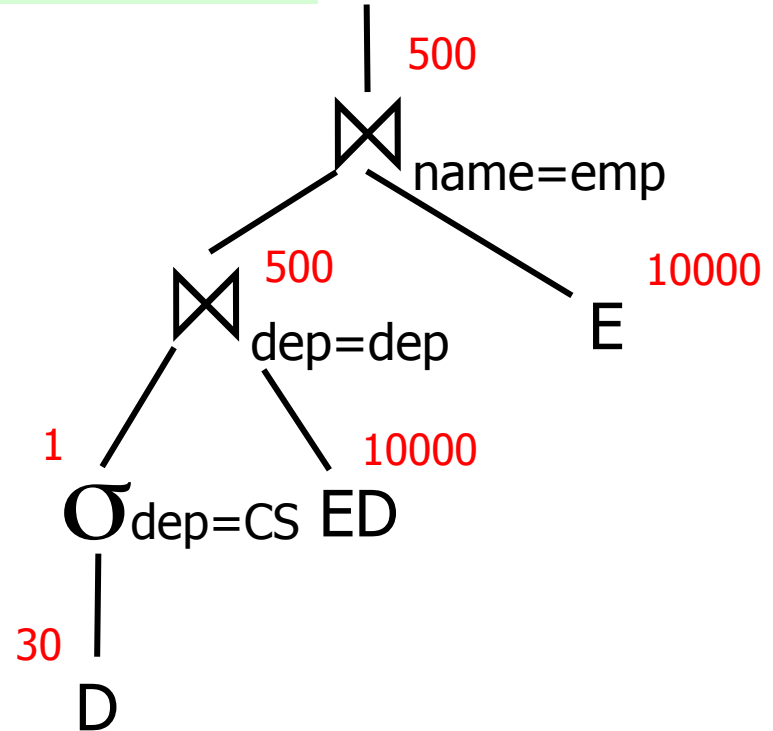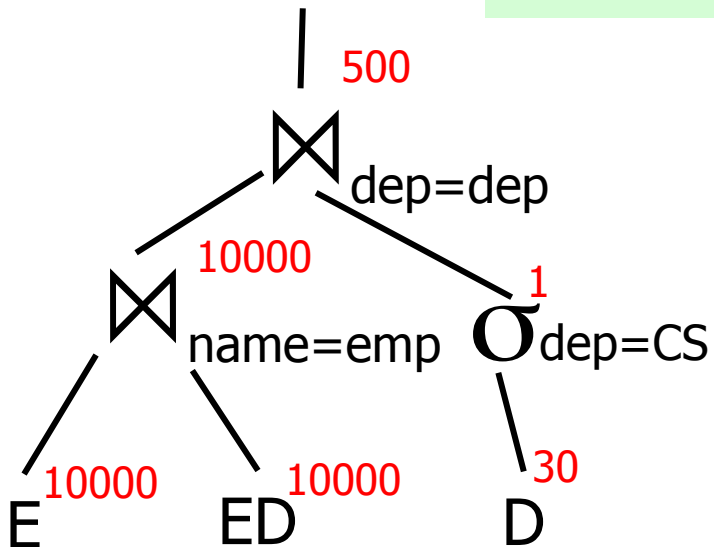
S(E) = S(ED) = S(D) = 1/10 block
M = 101
I/O costs only
No pipelining, write all results to disk

1100 x 10 + 101 x 10 + 3 (operator costs)
+ 1000 + 1 + 50   (write results)
= 13064 I/Os

1001 + 1050 + 3 (operator costs)
+ 1 + 50 + 50
= 2155 I/Os

101 x 10

500

⋈ dep=dep

10000

1100 x 10

1

3

name=emp

σ dep=CS

10000

10000

30

E

ED

D

1050 x 1

500

name=emp

1001 x 1

500

10000

dep=dep

E

1

10000

3

σ dep=CS

ED

30

D
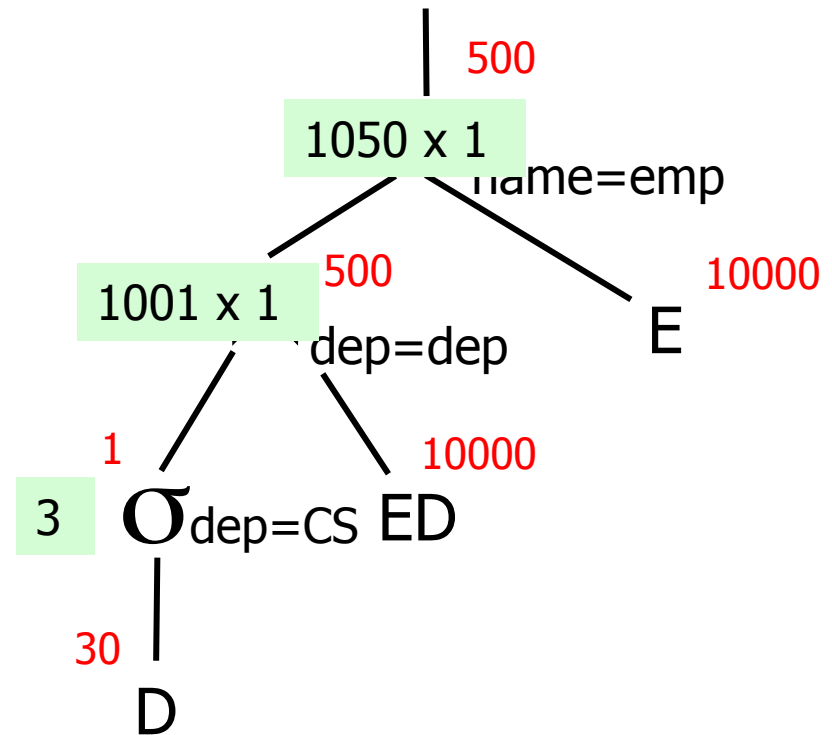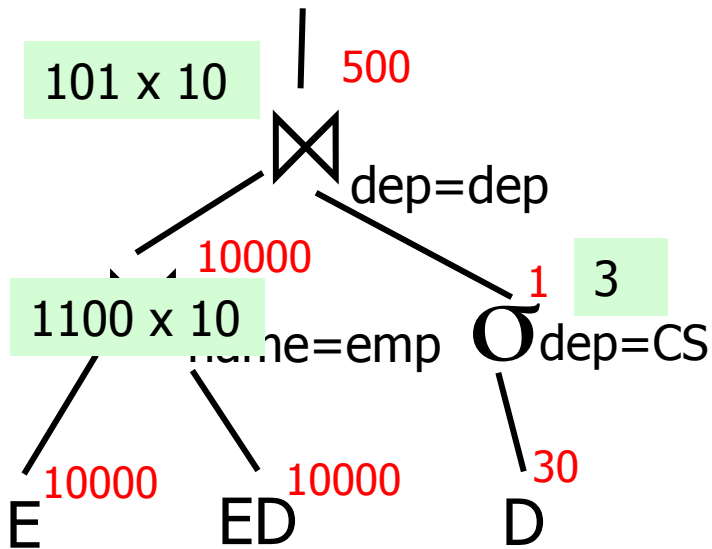
Plan Enumeration

- All
  - Consider all potential plans of a certain type (discussed later)
  - Prune only if sure
- Heuristics
  - Apply heuristics to prune search space
- Randomized Algorithms

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Plan Enumeration Algorithms

- All
  - Dynamic Programming (System R)
  - A* search

- Heuristics
  - Minimum Selectivity, Intermediate result size, …
  - KBZ-Algorithm, AB-Algorithm

- Randomized
  - Genetic Algorithms
  - Simulated Annealing

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Reordering Joins Revisited

- Equivalences (Natural Join)

    1. $R \bowtie S \equiv S \bowtie R$

    2. $(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$

- Equivalences Equi-Join

    1. $R \bowtie_{a=b} S \equiv S \bowtie_{a=b} R$

    2. $(R \bowtie_{a=b} S) \bowtie_{c=d} T \equiv R \bowtie_{a=b} (S \bowtie_{c=d} T)$?

    3. $\sigma_{a=b} (R \times S) \equiv R \bowtie_{a=b} S$?

Notes 11 - Physical Optimization

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Equi-Join Equivalences

- $(R \bowtie_{a=b} S) \bowtie_{c=d} T \equiv R \bowtie_{a=b} (S \bowtie_{c=d} T)$
- What if c is attribute of R?

$(R \bowtie_{a=b} S) \bowtie_{c=d} T \equiv R \bowtie_{a=b \wedge c=d} (S \times T)$

- $\sigma_{a=b} (R \times S) \equiv R \bowtie_{a=b} S$?
- Only useful if a is from R and S from b (vice-versa)

Why Cross-Products are bad

- We discussed efficient join algorithms
  - Merge-join $O(n)$ resp. $O(n \log(n))$
  - Vs. Nested-loop $O(n^2)$

- R X S

  - Result size is $O(n^2)$

    - Cannot be better than $O(n^2)$

  - Surprise, surprise: merge-join doesn't work
    no need to sort, but degrades to nested loop

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Agenda

- Given some query
  - How to enumerate all plans?
- Try to avoid cross-products
- Need way to figure out if equivalences can be applied
  - Data structure: **Join Graph**

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Join Graph

- Assumptions
  - Only equi-joins (a = b)
    - a and b are either constants or attributes
  - Only conjunctive join conditions (AND)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Join Graph

- Nodes: Relations R_1, ... , R_n of query
- Edges: Join conditions
  - Add edge between R_i and R_j labeled with C
    - if there is a join condition C
    - That equates an attribute from R_i with an attribute from R_j
  - Add a self-edge to R_i for each simple predicate

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Join Graph Example

```
SELECT e.name
FROM Employee e,
     EmpDep ed,
     Department d
WHERE e.name = ed.emp
      AND ed.dep = d.dep
      AND d.dep = 'CS'
```

Department

EmpDep

Employee

Join Graph Example

```
SELECT e.name
FROM Employee e,
      EmpDep ed,
      Department d
WHERE e.name = ed.emp
      AND ed.dep = d.dep
      AND d.dep = 'CS'
```

dep='CS'

Department

name=emp

EmpDep

dep=dep

Employee

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
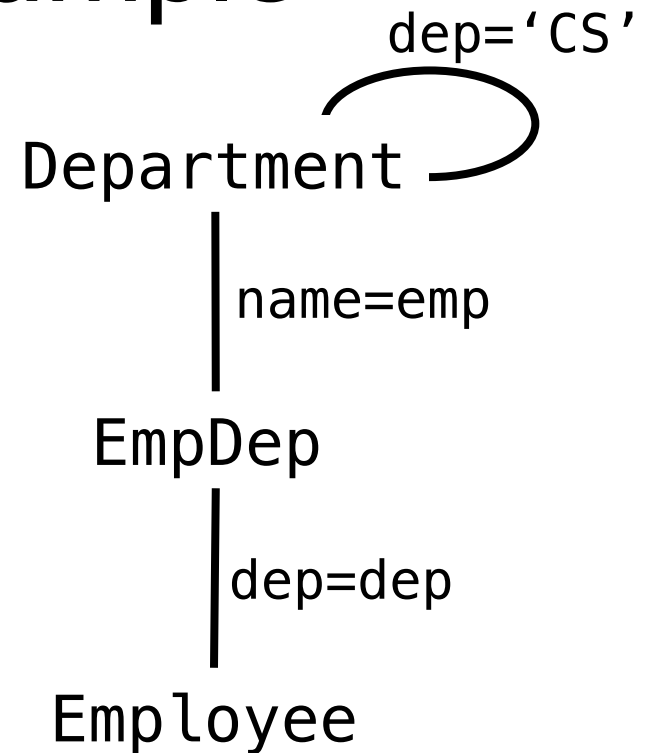
Notes on Join Graph

- Join Graph tells us in which ways we can join without using cross products

- However, …

  – Only if transitivity is considered

Notes 11 - Physical Optimization

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
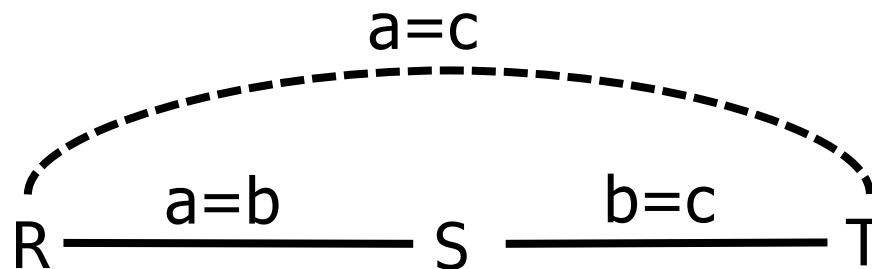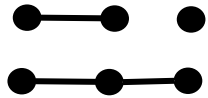
Join Graph Shapes
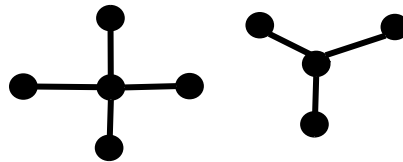
Chain queries

Star queries
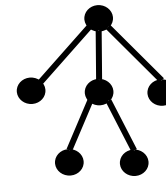
Tree queries
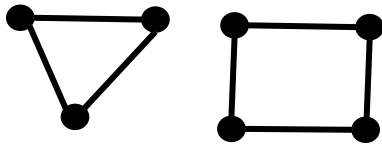
Cycle queries

Clique queries

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
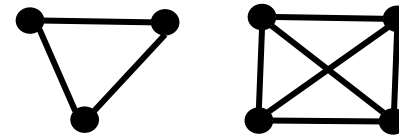
Join Graph Shapes

Chain queries

```
SELECT *
FROM R,S,T
WHERE R.a = S.b
      AND S.c = T.d
```

Join Graph Shapes
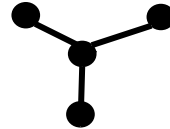
Star queries

```
SELECT *
FROM R,S,T,U
WHERE R.a = S.a
      AND R.b = T.b
      AND R.c = U.c
```

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Join Graph Shapes

Tree queries

```
SELECT *
FROM R,S,T,U,V
WHERE R.a = S.a
       AND R.b = T.b
       AND T.c = U.c
       AND T.d = V.d
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
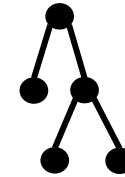
Join Graph Shapes

```
SELECT *
FROM R,S,T
WHERE R.a = S.a
      AND S.b = T.b
      AND T.c = R.c
```

Cycle queries

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
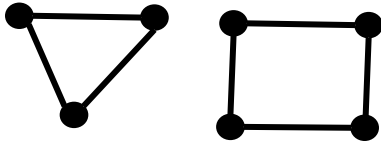
Join Graph Shapes

```
SELECT *
FROM R,S,T
WHERE R.a = S.a
      AND S.b = T.b
      AND T.c = R.c
```
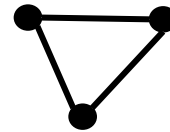
Clique queries

How many join orders?

- Assumption

  - Use cross products (can freely reorder)
  - Joins are binary operations

    - Two inputs
    - Each input either join result or relation access

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How many join orders?

- Example 3 relations R,S,T
  - 12 orders

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
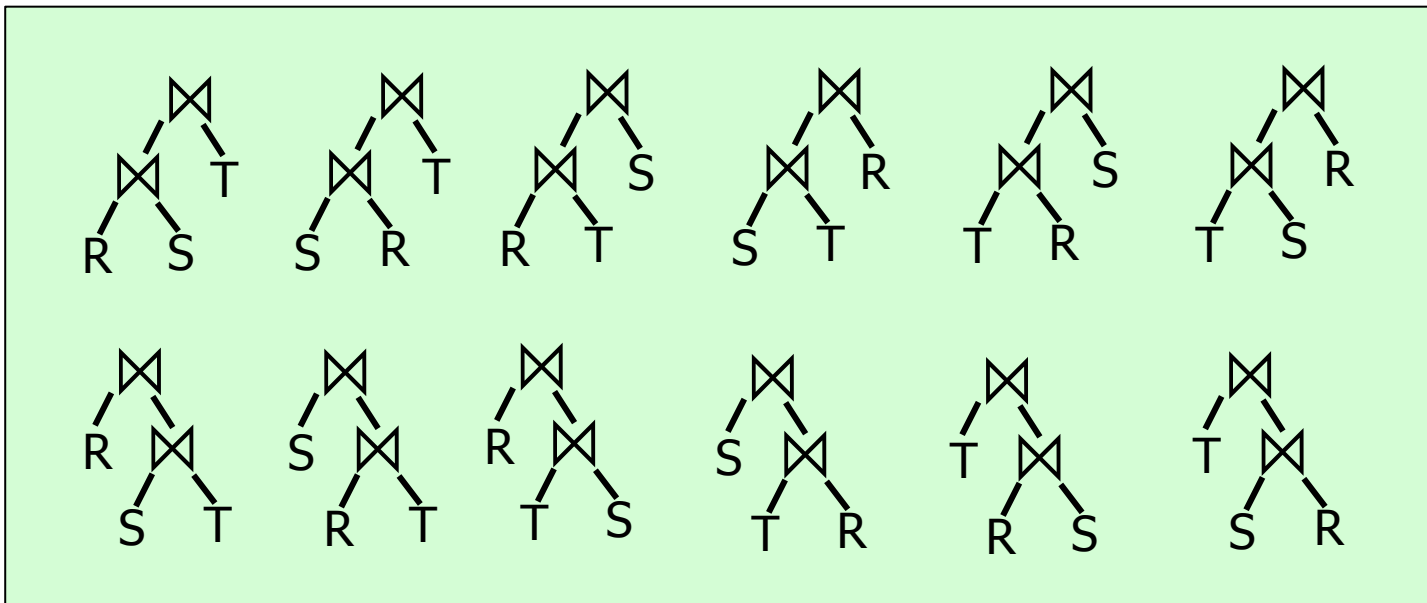
How many join orders?

- A join over **n+1** relations requires **n** binary joins
- The root of the join tree joins **k** with **n – k – 1** join operators (0 <= k <= n-1)

k joins   **n – k - 1** joins

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
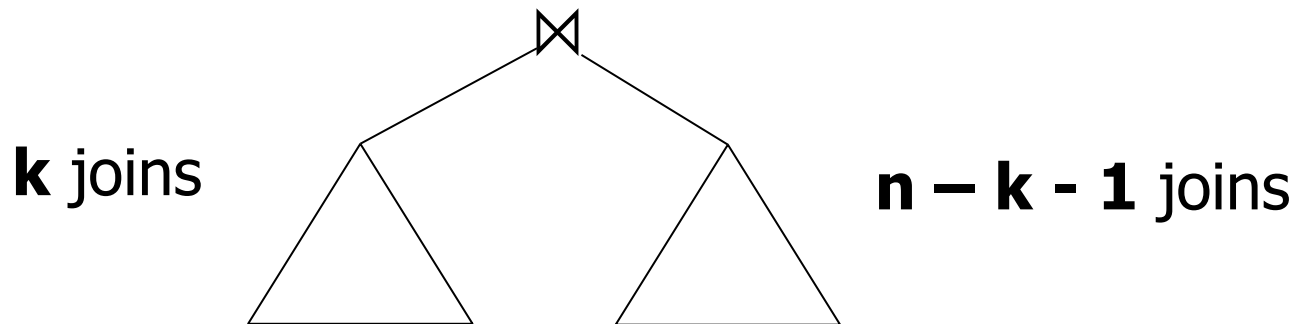
How many join orders?

- This are the **Catalan numbers**

$$C_n = \sum_{k=0}^{n-1} C_k \times C_{n-k-1} = (2n)! / (n+1)!n!$$

$$C_0 = 1$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How many join orders?

- This are the **Catalan numbers**
- For each such tree we can permute the input relations **(n+1)!** Permutations

(2n)! / (n+1)!n! * (n+1)! = (2n)!/n!

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How many join orders?

#relations	#join trees
2	2
3	12
4	120
5	1,680
6	30,240
7	665,280
8	17,297,280
9	17,643,225,600
10	670,442,572,800
11	28,158,588,057,600

How many join orders?

- If for each join we consider **k** join algorithms then for **n** relations we have
  - Multiply with a factor k^{n-1}
- Example consider
  - Nested loop
  - Merge
  - Hash

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

How many join orders?

#relations	#join trees
2	6
3	108
4	3240
5	136,080
6	7,348,320
7	484,989,120
8	37,829,151,360
9	115,757,203,161,600
10	13,196,321,160,422,400
11	1,662,736,466,213,222,400

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Too many join orders?

- Even if costing is cheap
  - Unrealistic assumption 1 CPU cycle
  - Realistic are thousands or millions of instructions
- Cost all join options for 11 relations
  - 3GHz CPU, 8 cores
  - 69,280,686 sec > 2 years

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How to deal with excessive number of combinations?

- Prune parts based on optimality
  - Dynamic programming
  - A*-search
- Only consider certain types of join trees
  - Left-deep, Right-deep, zig-zag, bushy
- Heuristic and random algorithms

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Dynamic Programming

- Assumption: **Principle of Optimality**
  - To compute the **global** optimal plan it is only necessary to consider the optimal solutions for its **sub-queries**
- Does this assumption hold?
  - Depends on cost-function

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What is dynamic programming?

- Recall data structures and algorithms 101!
- Consider a **Divide-and-Conquer** problem
  - Solutions for a problem of size **n** can be build from solutions for sub-problems of smaller size (e.g., **n/2** or **n-1**)

- **Memoize**
  - Store solutions for sub-problems
  - -> Each solution has to be only computed once
  - -> Needs extra memory

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example Fibonacci Numbers

- $F(n) = F(n-1) + F(n-2)$
- $F(0) = F(1) = 1$

```
Fib(n)
{
    if (n = 0) return 0
    else if (n = 1) return 1
    else return Fib(n–1) + Fib(n–2)
}
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example Fibonacci Numbers

Notes 11 - Physical Optimization   46
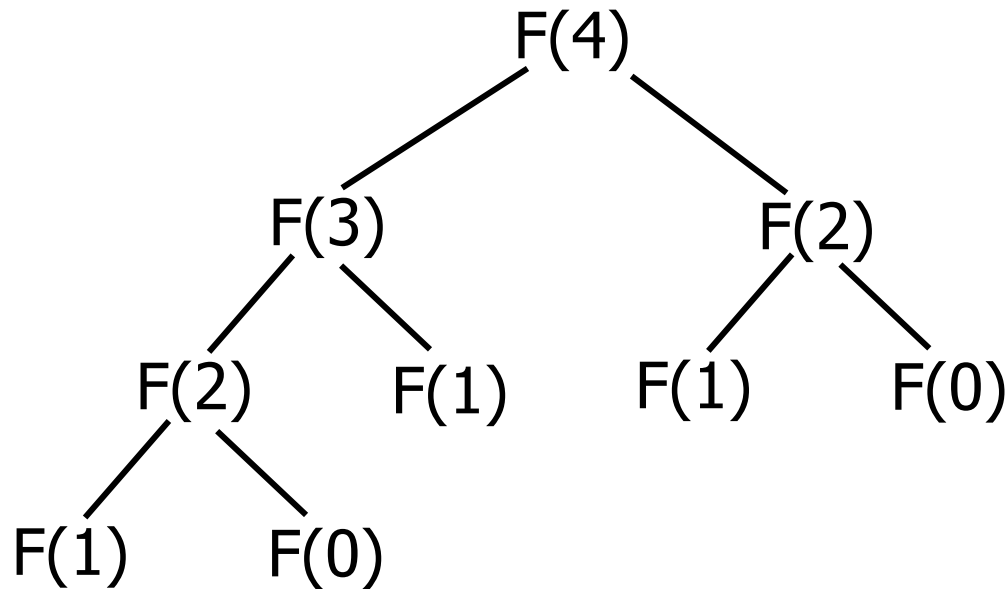
Complexity

- Number of calls
  - $C(n) = C(n-1) + C(n-2) + 1 = Fib(n+2)$
  - $O(2^n)$

Notes 11 - Physical Optimization

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Using dynamic programming

```
Fib(n)
{
    int[] fib;
    fib[0] = 1;
    fib[1] = 1;

    for(i = 2; i < n; i++)
        fib[i] = fib[i-1] + fib[i-2]

    return fib[n];
}
```

Example Fibonacci Numbers

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
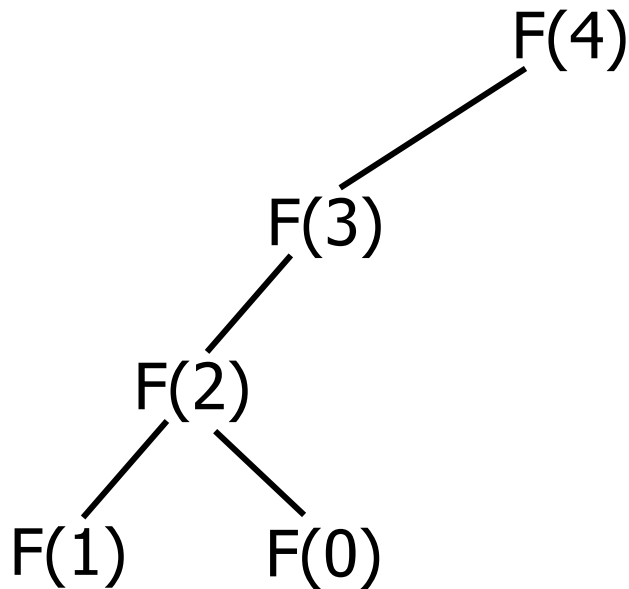
What do we gain?

- $O(n)$ instead of $O(2^n)$

Dynamic Programming for Join Enumeration

- Find cheapest plan for n-relation join in n passes

- For each **i** in **1** … **n**

  - Construct solutions of size **i** from best solutions of size **< i**

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

DP Join Enumeration

```
optPlan ← Map({R},{plan})

find_join_dp(q(R_1,…,R_n))
{
  for i=1 to n
    optPlan[{R_i}] ← access_paths(R_i)
  for i=2 to n
    foreach S ⊆ {R_1,…,R_n} with |S|=i
      optPlan[S] ← ∅
      foreach O ⊂ S with O ≠ ∅
        optPlan[S] ← optPlan[S] ∪
              possible_joins(optPlan(O), optPlan(S\O))
    prune_plans(optPlan[S])
  return optPlan[{R_1,…,R_n}]
}
```

Dynamic Programming for Join Enumeration

- `access_paths (R)`

  – Find cheapest access path for relation R

- `possible_joins(plan, plan)`

  – Enumerate all joins (merge, NL, …) variants between the input plans

- `prune_plans({plan})`

  – Only keep cheapest plan from input set

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

DP-JE Complexity

- Time: $O(3^n)$
- Space: $O(2^n)$
- Still to much for large number of joins (10-20)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Types of join trees

Left-deep

zig-zag

bushy

+left
+ zig-zag
+right

Right-deep

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
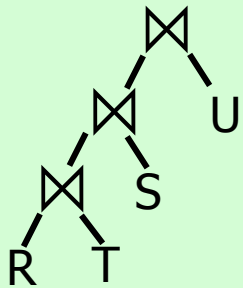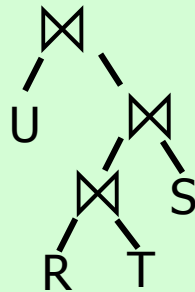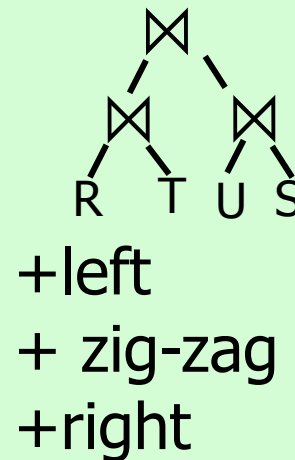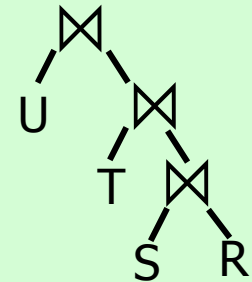
Number of Join-Trees

- Number of join trees for **n** relations
- Left-deep: **n!**
- Right-deep: **n!**
- Zig-zag: $\mathbf{2^{n-2}n!}$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

How many join orders?

#relations	#bushy join trees	#left-deep join trees
2	2	2
3	12	6
4	120	24
5	1,680	120
6	30,240	720
7	665,280	5040
8	17,297,280	40,230
9	17,643,225,600	362,880
10	670,442,572,800	3,628,800
11	28,158,588,057,600	39,916,800

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

DP with Left-deep trees only

- Reduced search-space
- Each join is with input relation
  - ->can use index joins
  - ->easy to pipe-line
- DP with left-deep plans was introduced by system R, the first relational database developed by IBM Research
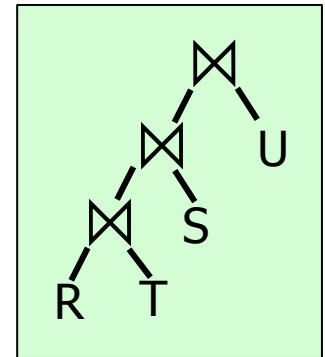
Revisiting the assumption

- Is it really sufficient to only look at the best plan for every sub-query?

- Cost of merge join depends whether the input is already sorted

  - -> A sub-optimal plan may produce results ordered in a way that reduces cost of joining above

  - Keep track of **interesting orders**

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Interesting Orders

- Number of interesting orders is usually small

- ->Extend DP join enumeration to keep track of interesting orders
  - Determine interesting orders
  - For each sub-query store best-plan for each interesting order

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example Interesting Orders

Left-deep best plans: 3-way {R,S,T}
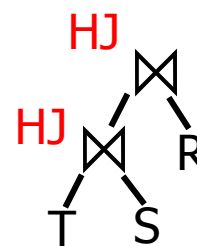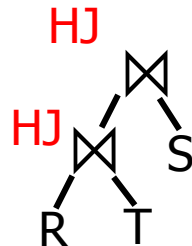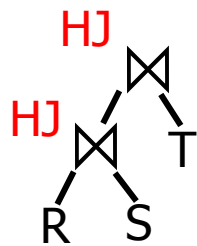
Left-deep best plans: 2-way

{R,S}   {R,T}   {S,T}
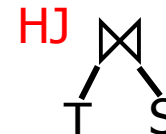
IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example Interesting Orders

Left-deep best plans: 3-way {R,S,T}

Left-deep best plans: 2-way

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
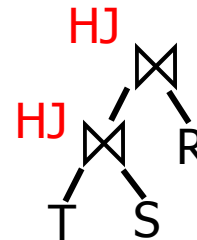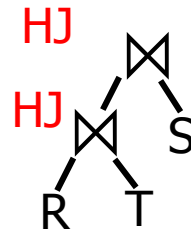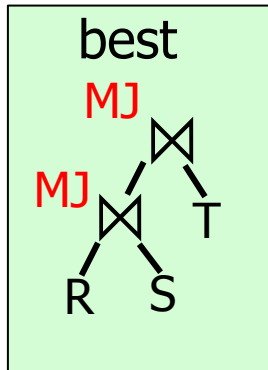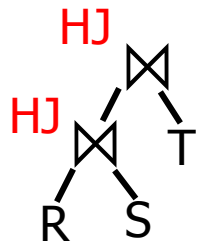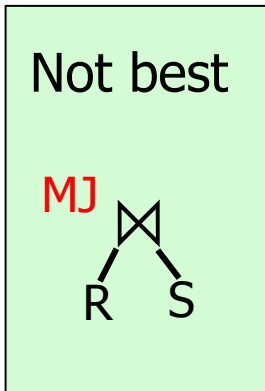
Greedy Join Enumeration

- Heuristic method
    - Not guaranteed that best plan is found
- Start from single relation plans
- In each iteration greedily join to plans with the minimal cost
- Until a plan for the whole query has been generated

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Greedy Join Enumeration

```
plans ← list({plan})

find_join_dp(q(R₁,…,Rₙ))
{
  for i=1 to n
    plans ← plans ∪ access_paths(Rᵢ)
  for i=n to 2
    cheapest = argmin_{j,k∈{1,…,n}} (cost(Pⱼ ⋈ Pₖ))
    plans ← plans \ {Pⱼ,Pₖ} ∪ {Pⱼ ⋈ Pₖ}
  return plans // single plan left
}
```

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Greedy Join Enumeration

- Time: $O(n^3)$
  - Loop iterations: $O(n)$
  - In each iterations looking of pairs of plans in of max size n: $O(n^2)$
- Space: $O(n^2)$
  - Needed to store the current list of plans

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Randomized Join-Algorithms

- Iterative improvement
- Simulated annealing
- Tabu-search
- Genetic algorithms

Transformative Approach

- Start from (random) complete solutions
- Apply transformations to generate new solutions
  - Direct application of equivalences
    - Commutativity
    - Associativity
  - Combined equivalences
    - E.g., $(R \bowtie S) \bowtie T \equiv T \bowtie (S \bowtie R)$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Concern about Transformative Approach

- Need to be able to generate random plans fast

- Need to be able to apply transformations fast

  – Trade-off: space covered by transformations vs. number and complexity of transformation rules

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Iterative Improvement

```
improve(q(R_1,…,R_n))
{
  best ← random_plan(q)
  while (not reached time limit)
    curplan ← random_plan(q)
    do
      prevplan ← curplan
      curplan ← apply_random_trans (prevplan)
    while (cost(curplan) < cost(prevplan))
    if (cost(prevplan) < cost(best)
      best ← prevplan
  return best
}
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Iterative Improvement

- Easy to get stuck in local minimum
- **Idea:** Allow transformations that result in more expensive plans with the hope to move out of local minima
  - ->Simulated Annealing

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Simulated Annealing

```
SA(q(R_1,…,R_n))
{
  best ← random_plan(q)
  curplan ← best
  t ← t_init // "temperature"
  while (t > 0)
    newplan ← apply_random_trans(curplan)
    if cost(newplan) < cost(curplan)
      curplan ← newplan
    else if random() < e^(-(cost(newplan)-cost(curplan))/t)
      curplan ← newplan
    if (cost(curplan) < cost(best)
      best ← curplan
  reduce(t)
  return best
}
```

Simulated Annealing

```
SA(q(R_1,…,R_n))
{
  best ← random_plan(q)
  curplan ← best
  t ← t_init // "temperature"
  while (t > 0)
    newplan ← apply_random_trans(curplan)
    if cost(newplan) < cost(curplan)
      curplan ← newplan
    else if random() < e^{-(cost(newplan)-cost(curplan))/t}
      curplan ← newplan
    if (cost(curplan) < cost(best)
      best ← curplan
    reduce(t)
  return best
}
```

Until "cooled down"

Reduce
Chance
To "jump"

Probability to
Take "bad" plan
Based on temp.

Genetic Algorithms

- Represent solutions as sequences (strings) = genome

- Start with random population of solutions

- Iterations = Generations
  - Mutation = random changes to genomes
  - Cross-over = Mixing two genomes

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Genetic Join Enumeration for Left-deep Plans

- A left-deep plan can be represented as a permutation of the relations
  - Represent each relation by a number
  - E.g., encode this tree as "1243"

Notes 11 - Physical Optimization
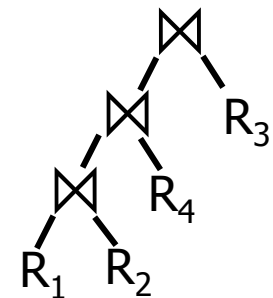
Mutation

- Switch random two random positions
- Is applied with a certain fixed probability
- E.g., "1342" -> "4312"

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Cross-over

- Sub-set exchange
  - For two solutions find subsequence
    - equals length with the same set of relations
  - Exchange these subsequences
- Example
  - J_1 = "5632478" and J_2 = "5674328"
  - Generate J' = "5643278"

COMPUTER SCIENCE

Notes 11 - Physical Optimization

76

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Survival of the fittest

- Probability of survival determined by rank within the current population

- Compute ranks based on costs of solutions

- Assign Probabilities based on rank
  - Higher rank -> higher probability to survive

- Roll a dice for each solution

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Genetic Join Enumeration

- Create an initial population **P** random plans
- Apply crossover and mutation with a fixed rate
    - E.g., crossover 65%, mutation 5%
- Apply selection until size is again **P**
- Stop once no improvement for at least **X** iterations

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Comparison Randomized Join Enumeration

- Iterative Improvement
  - Towards local minima (easy to get stuck)

- Simulated Annealing
  - Probability to "jump" out of local minima

- Genetic Algorithms
  - Random transformation
  - Mixing solutions (crossover)
  - Probabilistic chance to keep solution based on cost

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Join Enumeration Recap

- Hard problem
  - Large problem size
    - Want to reduce search space
  - Large cost differences between solutions
    - Want to consider many solution to increase chance to find a good one.

COMPUTER
SCIENCE

Notes 11 - Physical Optimization

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Join Enumeration Recap

- Tip of the iceberg
  - More algorithms
  - Combinations of algorithms
  - Different representation subspaces of the problem
  - Cross-products / no cross-products
  - ...

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

From Join-Enumeration to Plan Enumeration

- So far we only know how to reorder joins

- What about other operations?

- What if the query does consist of several SQL blocks?

- What if we have nested subqueries?

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

SQL query

parse → parse tree → convert → logical query plan → apply laws → "improved" l.q.p → estimate result sizes → l.q.p. +sizes → consider physical plans

{P1,P2,.....}

estimate costs → {(P1,C1),(P2,C2)...} → pick best → Pi → execute → answer

statistics

COMPUTER
SCIENCE

Notes 11 - Physical Optimization   83

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

From Join-Enumeration to Plan Enumeration

- Lets reconsider the input to plan enumeration!
  - We briefly touched on **Query graph models**
  - We discussed briefly why relational algebra is not sufficient

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Query Graph Models

- Represents an SQL query as query blocks
  - A query block corresponds to the an SQL query block (SELECT FROM WHERE …)
  - Data type/operator/function information
    - Needed for execution and optimization decisions
  - Structured in a way suited for optimization

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

QGM example

```
SELECT name, city
FROM

    (SELECT *
    FROM person) AS p,
    (SELECT *
    FROM address) AS a
WHERE p.addrId = a.id
```

Notes 11 - Physical Optimization

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
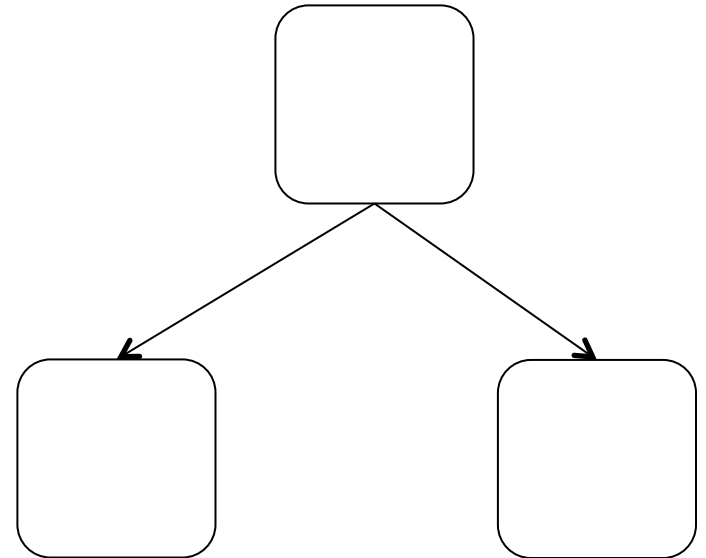
Postgres Example

{QUERY

    :commandType 1
    :querySource 0
    :canSetTag true
    :utilityStmt <>
    :resultRelation 0
    :intoClause <>
    :hasAggs false
    :hasSubLinks false
    :rtable (
     {RTE
     :alias
       {ALIAS
       :aliasname p
       :colnames <>
       }
     :eref
       {ALIAS
       :aliasname p
       :colnames ("name" "addrid")
       }
     :rtekind 1
     :subquery
       {QUERY
       :commandType 1
       :querySource 0
       :canSetTag true

…

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

How to enumerate plans for a QGM query

- Recall the correspondence between SQL query blocks and algebra expressions!

- If block is (A)SPJ
  - Determine join order
  - Decide which aggregation to use (if any)

- If block is set operation
  - Determine order

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

More than one query block

- Recursive create plans for subqueries
  - Start with leaf blocks
- Consider our example
  - Even if blocks are only SPJ we would not consider reordering of joins across blocks
  - -> try to "pull up" subqueries before optimization

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Subquery Pull-up

SELECT name, city
FROM

    (SELECT *
    FROM person) AS p,
    (SELECT *
    FROM address) AS a
WHERE p.addrId = a.id

SELECT name, city
FROM

    person p,
    address a
WHERE p.addrId = a.id

Notes 11 - Physical Optimization

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
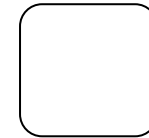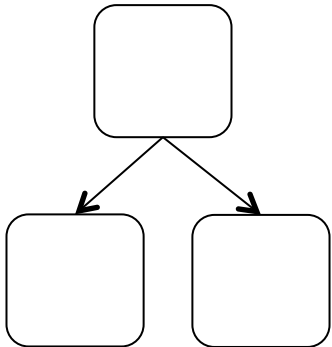
Parameterized Queries

- Problem
  - Repeated executed of similar queries
- Example
  - Webshop
  - Typical operation: Retrieve product with all user comments for that product
  - Same query modulo product id

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Parameterized Queries

- Naïve approach
  - Optimize each version individually
  - Execute each version individually
- Materialized View
  - Store common parts of the query
  - -> Optimizing a query with materialized views
  - -> Separate topic not covered here

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Caching Query Plans

- Caching Query Plans
  - Optimize query once
  - Adapt plan for specific instances
  - **Assumption:** varying values do not effect optimization decisions
  - **Weaker Assumption**: Additional cost of "bad" plan less than cost of repeated planning

Parameterized Queries

- How to represent varying parts of a query
  - Parameters
  - Query planned with parameters assumed to be unknown
  - For execution replace parameters with concrete values

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

PREPARE statement

- In SQL
  - **PREPARE** `name (parameters)` **AS** `query`
  - **EXECUTE** name (parameters)

Nested Subqueries

```
SELECT name
FROM person p
WHERE EXISTS (SELECT newspaper
             FROM hasRead h
             WHERE h.name = p.name
               AND h.newspaper = 'Tribune')
```

COMPUTER
SCIENCE

Notes 11 - Physical Optimization

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

How to evaluate nested subquery?

- If no correlations:
  - Execute once and cache results
- For correlations:
  - Create plan for query with parameters
- -> called nested iteration

Nested Iteration - Correlated

```
q ← outer query
q' ← inner query
result ← execute(q)
foreach tuple t in result
  qt ← q'(t) // parameterize q' with values from t
  result' ← execute (qt)
  evaluate_nested_condition (t,result')
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Nested Iteration - Uncorrelated

```
q ← outer query
q' ← inner query
result ← execute(q)
result' ← execute (q_t)
foreach tuple t in result
    evaluate_nested_condition (t,result')
```

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Nested Iteration - Example

```
SELECT name
FROM person p
WHERE EXISTS (SELECT newspaper
                FROM hasRead h
                WHERE h.name = p.name
                    AND h.newspaper = 'Tribune')
```

person

name	gender
Alice	female
Bob	male
Joe	male

hasRead

name	newspaper
Alice	Tribune
Alice	Courier
Joe	Courier

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Nested Iteration - Example

```
q ← outer query
q' ← inner query
result ← execute(q)
foreach tuple t in result
→ q_t ← q'(t)
   result' ← execute (q_t)
   evaluate_nested_condition (t,result')
```

```
SELECT newspaper
FROM hasRead h
WHERE h.name = p.name
      AND h.newspaper
          = 'Tribune')
```

person

name	gender
→ Alice	female
Bob	male
Joe	male

hasRead

name	newspaper
Alice	Tribune
Alice	Courier
Joe	Courier

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Nested Iteration - Example

```
q ← outer query
q' ← inner query
result ← execute(q)
foreach tuple t in result
→ q_t ← q'(t)
   result' ← execute (q_t)
   evaluate_nested_condition (t,result')
```

```
SELECT newspaper
FROM hasRead h
WHERE h.name = 'Alice'
      AND h.newspaper
          = 'Tribune')
```

person

name	gender
Alice	female
Bob	male
Joe	male

hasRead

name	newspaper
Alice	Tribune
Alice	Courier
Joe	Courier

IIT College of
Science and Letters

COMPUTER
SCIENCE

ILLINOIS INSTITUTE OF TECHNOLOGY

Nested Iteration - Example

```
q ← outer query
q' ← inner query
result ← execute(q)
foreach tuple t in result
  q_t ← q'(t)
→ result' ← execute (q_t)
  evaluate_nested_condition (t,result')
```

```
SELECT newspaper
FROM hasRead h
WHERE h.name = p.name
      AND h.newspaper
          = 'Tribune')
```

person

name	gender
→ Alice	female
Bob	male
Joe	male

hasRead

name	newspaper
Alice	Tribune
Alice	Courier
Joe	Courier

result'

newspaper
Tribune

Notes 11 - Physical Optimization

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Nested Iteration - Example

```
q ← outer query
q' ← inner query
result ← execute(q)
foreach tuple t in result
    q_t ← q'(t)
    result' ← execute (q_t)
→ evaluate_nested_condition (t,result')
```

EXISTS evaluates to true!

Output(Alice)

person

name	gender
Alice	female
Bob	male
Joe	male

hasRead

name	newspaper
Alice	Tribune
Alice	Courier
Joe	Courier

result'

newspaper
Tribune

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Nested Iteration - Example

```
q ← outer query
q' ← inner query
result ← execute(q)
foreach tuple t in result
   q_t ← q'(t)
   result' ← execute (q_t)
   evaluate_nested_condition (t,result')
```

Empty result set –>
EXISTS evaluates to
false

person

name	gender
Alice	female
Bob	male
Joe	male

→ Bob male

hasRead

name	newspaper
Alice	Tribune
Alice	Courier
Joe	Courier

result'

newspaper

CS 525

Notes 11 - Physical Optimization   105

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Nested Iteration - Example

```
q ← outer query
q' ← inner query
result ← execute(q)
foreach tuple t in result
  q_t ← q'(t)
  result' ← execute (q_t)
  evaluate_nested_condition (t,result')
```

Empty result set –>
EXISTS evaluates to
false

person

name	gender
Alice	female
Bob	male
Joe	male

hasRead

name	newspaper
Alice	Tribune
Alice	Courier
Joe	Courier

result'

newspaper

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Nested Iteration - Discussion

- Repeated evaluation of nested subquery
  - If correlated
  - Improve:
    - Plan once and substitute parameters
    - EXISTS: stop processing after first result
    - IN/ANY: stop after first match

- No optimization across nesting boundaries

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Unnesting and Decorrelation

- Apply equivalences to transform nested subqueries into joins
- **Unnesting:**
  - Turn a nested subquery into a join
- **Decorrelation:**
  - Turn correlations into join expressions

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Equivalences

- Classify types of nesting
- Equivalence rules will have preconditions
- Can be applied heuristically before plan enumeration or using a transformative approach

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

N-type Nesting

- Properties
  - Expression ANY comparison (or IN)
  - No Correlations
  - Nested query does not use aggregation

- Example

```
SELECT name
FROM orders o
WHERE o.cust IN (SELECT cId
                 FROM customer
                 WHERE region = 'USA')
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

A-type Nesting

- Properties
  - Expression is ANY comparison (or scalar)
  - No Correlations
  - Nested query uses aggregation
  - No Group By

- Example

```
SELECT name
FROM orders o
WHERE o.amount = (SELECT max(amount)
                  FROM orders i)
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

J-type Nesting

- Properties
  - Expression is ANY comparison (IN)
  - Nested query uses equality comparison with correlated attribute
  - No aggregation in nested query

- Example

```
SELECT name
FROM orders o
WHERE o.amount IN (SELECT amount
                   FROM orders i
                   WHERE i.cust = o.cust
                         AND i.shop = 'New York')
```

ILLINOIS INSTITUTE OF TECHNOLOGY

JA-type Nesting

- Properties
  - Expression equality comparison
  - Nested query uses equality comparison with correlated attribute
  - Nested query uses aggregation and no GROUP BY

- Example

```
SELECT name
FROM orders o
WHERE o.amount = (SELECT max(amount)
                     FROM orders i
                     WHERE i.cust = o.cust)
```

Unnesting A-type

- Move nested query to FROM clause
- Turn nested condition (op ANY, IN) into op with result attribute of nested query

COMPUTER
SCIENCE

Notes 11 - Physical Optimization   114

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Unnesting N/J-type

- Move nested query to FROM clause
- Add DISTINCT to SELECT clause of nested query
- Turn equality comparison with correlated attributes into join conditions
- Turn nested condition (op ANY, IN) into op with result attribute of nested query

COMPUTER SCIENCE

Notes 11 - Physical Optimization   115

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example

1. **To FROM clause**
2. Add DISTINCT
3. Correlation to join
4. Nesting condition to join

```
SELECT name
FROM orders o
WHERE o.amount IN (SELECT amount
                   FROM orders i
                   WHERE i.cust = o.cust
                       AND i.shop = 'New York')
```

```
SELECT name
FROM orders o,
    (SELECT amount
    FROM orders i
    WHERE i.cust = o.cust
         AND i.shop = 'New York') AS sub
```

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example

1. To FROM clause
2. Add DISTINCT
3. Correlation to join
4. Nesting condition to join

```
SELECT name
FROM orders o
WHERE o.amount IN (SELECT amount
                   FROM orders i
                   WHERE i.cust = o.cust
                         AND i.shop = 'New York')
```

```
SELECT name
FROM orders o,
     (SELECT DISTINCT amount
      FROM orders i
      WHERE i.cust = o.cust
            AND i.shop = 'New York') AS sub
```

Notes 11 - Physical Optimization   117

Example

1. To FROM clause
2. Add DISTINCT
3. Correlation to join
4. Nesting condition to join

```
SELECT name
FROM orders o
WHERE o.amount IN (SELECT amount
                   FROM orders i
                   WHERE i.cust = o.cust
                         AND i.shop = 'New York')
```

```
SELECT name
FROM orders o,
     (SELECT DISTINCT amount, cust
      FROM orders i
      WHERE i.shop = 'New York') AS sub
WHERE sub.cust = o.cust
```

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example

1. To FROM clause
2. Add DISTINCT
3. Correlation to join
4. Nesting condition to join

```
SELECT name
FROM orders o
WHERE o.amount IN (SELECT amount
                   FROM orders i
                   WHERE i.cust = o.cust
                        AND i.shop = 'New York')
```

```
SELECT name
FROM orders o,
     (SELECT DISTINCT amount, cust
      FROM orders i
      WHERE i.shop = 'New York') AS sub
WHERE sub.cust = o.cust
      AND o.amount = sub.amount
```

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Unnesting JA-type

- Move nested query to FROM clause
- Turn equality comparison with correlated attributes into
  - GROUP BY
  - Join conditions
- Turn nested condition (op ANY, IN) into op with result attribute of nested query

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

1. To FROM clause
2. Introduce GROUP BY and join conditions
3. Nesting condition to join

```
SELECT name
FROM orders o
WHERE o.amount = (SELECT max(amount)
                         FROM orders i
                         WHERE i.cust = o.cust)
```

```
SELECT name
FROM orders o,
     (SELECT max(amount)
      FROM orders I
      WHERE i.cust = o.cust) sub
```

Notes 11 - Physical Optimization

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Example

1. To FROM clause
2. Introduce GROUP BY and join conditions
3. Nesting condition to join

```
SELECT name
FROM orders o
WHERE o.amount = (SELECT max(amount)
                      FROM orders i
                      WHERE i.cust = o.cust)
```

```
SELECT name
FROM orders o,
     (SELECT max(amount) AS ma, i.cust
      FROM orders i
      GROUP BY i.cust) sub
WHERE i.cust = sub.cust
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example

1. To FROM clause

2. Introduce GROUP BY and join conditions

3. Nesting condition to join

```
SELECT name
FROM orders o
WHERE o.amount = (SELECT max(amount)
                        FROM orders i
                        WHERE i.cust = o.cust)
```

```
SELECT name
FROM orders o,
    (SELECT max(amount) AS ma, i.cust
     FROM orders i
     GROUP BY i.cust) sub
WHERE sub.cust = o.cust
        AND o.amount = sub.ma
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Unnesting Benefits Example

- N(orders) = 1,000,000

- V(cust,orders) = 10,000

- S(orders) = 1/10 block

```
SELECT name
FROM orders o
WHERE o.amount = (SELECT max(amount)
                         FROM orders i
                         WHERE i.cust = o.cust)
```

```
SELECT name
FROM orders o,
    (SELECT max(amount) AS ma, i.cust
     FROM orders i
     GROUP BY i.cust) sub
WHERE sub.cust = o.cust
     AND o.amount = sub.ma
```

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

- N(orders) = 1,000,000
- V(cust,orders) = 10,000
- S(orders) = 1/10 block
- M = 10,000

```
SELECT name
FROM orders o
WHERE o.amount = (SELECT max(amount)
                  FROM orders i
                  WHERE i.cust = o.cust)
```

- Inner query:
  - One scan B(orders) = 100,000 I/Os
- Outer query:
  - One scan B(orders) = 100,000 I/Os
  - 1,000,000 tuples
- Total cost: 1,000,001 x 100,000=~ 10^{11} I/Os

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- N(orders) = 1,000,000
- V(cust,orders) = 10,000
- S(orders) = 1/10 block
- M = 10,000

```
SELECT name
FROM orders o,
     (SELECT max(amount) AS ma, i.cust
      FROM orders i
      GROUP BY i.cust) sub
WHERE sub.cust = o.cust
      AND o.amount = sub.ma
```

- Inner queries:
  - One scan B(orders) = 100,000 I/Os
    - 1,000,000 result tuples
  - Aggregation: Sort (assume 1 pass) = 3 x 100,000 = 300,000 I/Os
    - 10,000 result tuples -> + 1,000 pages to write to disk

- The join: use merge – join during merge
  - 3 x (1,000 + 100,000) I/Os = 303,000 I/Os

- Total cost: 604,000 I/Os

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

CS 525: Advanced Database Organization
Organization
12: Transaction Management

Boris Glavic

Slides: adapted from a course taught by
Hector Garcia-Molina, Stanford InfoLab

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Concurrency and Recovery

- DBMS should enable multiple clients to access the database concurrently
  - This can lead to problems with correctness of data because of interleaving of operations from different clients
  - ->System should ensure correctness (concurrency control)

COMPUTER SCIENCE

Notes 12 - Transaction Management

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Concurrency and Recovery

- DBMS should enable reestablish correctness of data in the presence of failures

  - ->System should restore a correct state after failure (recovery)

COMPUTER SCIENCE

Notes 12 - Transaction Management

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Integrity or correctness of data

- Would like data to be "accurate" or "correct" at all times

EMP

Name	Age
White	52
Green	3421
Gray	1

Integrity or consistency constraints

- Predicates data must satisfy

- Examples:

  - x is key of relation R

  - x → y holds in R

  - Domain(x) = {Red, Blue, Green}

  - α is valid index for attribute x of R

  - no employee should make more than
    twice the average salary

CS 525

COMPUTER
SCIENCE

Notes 12 - Transaction
Management

5

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Definition:

- <u>Consistent state:</u> satisfies all constraints
- <u>Consistent DB:</u> DB in consistent state

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Constraints (as we use here) may not capture "full correctness"

Example 1   Transaction constraints

- When salary is updated,

  new salary > old salary

- When account record is deleted,

  balance = 0

COMPUTER SCIENCE

Notes 12 - Transaction Management

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Note: could be "emulated" by simple constraints, e.g.,

account

Acct #	....	balance	deleted?

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Constraints (as we use here) may not capture "full correctness"

Example 2   Database should reflect real world

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
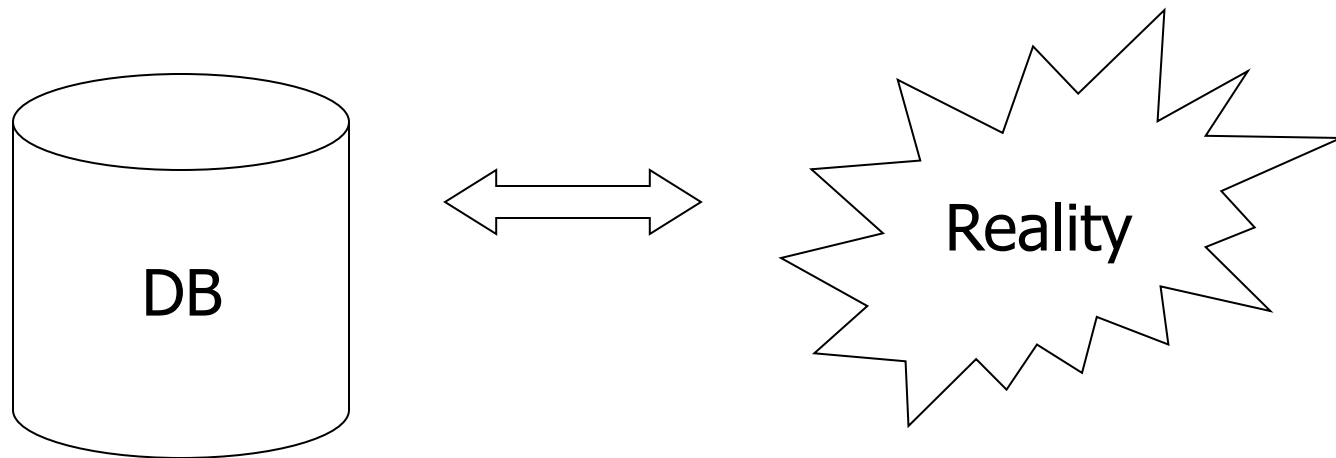
☞in any case, continue with constraints…

Observation: DB <u>cannot</u> be consistent always!

<u>Example:</u> $a_1 + a_2 + \ldots a_n = TOT$ (constraint)

Deposit \$100 in a_2: $\begin{cases} a_2 \leftarrow a_2 + 100 \\ TOT \leftarrow TOT + 100 \end{cases}$

Notes 12 - Transaction Management

10

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: $a_1 + a_2 + \ldots a_n = TOT$ (constraint)

Deposit \$100 in a_2: $a_2 \leftarrow a_2 + 100$

$TOT \leftarrow TOT + 100$

a_2	:
	50
	:
TOT	1000

$\longrightarrow$

:
150
:
1000

$\longrightarrow$

:
150
:
1100
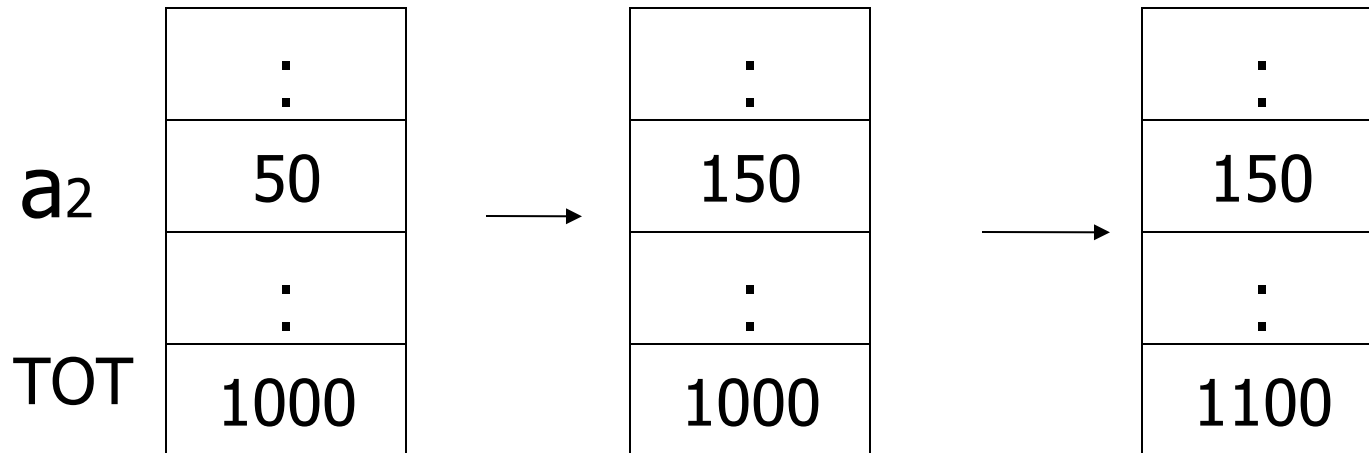
Transactions

- **Transaction:** Sequence of operations executed by one concurrent client that preserve consistency

CS 525

Notes 12 - Transaction Management

12

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Transaction: collection of actions that preserve consistency

Consistent DB — T — Consistent DB'
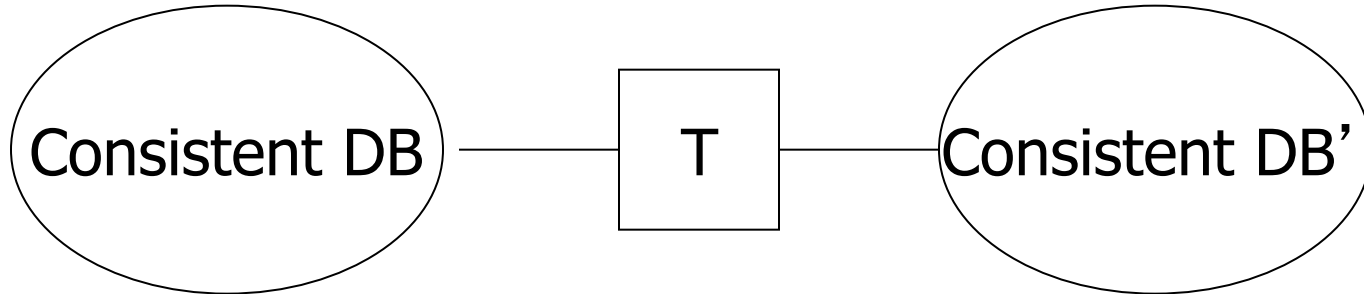
Big assumption:

If T starts with consistent state +

T executes in isolation

$\Rightarrow$ T leaves consistent state

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

<u>Correctness</u> (informally)

- If we stop running transactions,
        DB left consistent

- Each transaction sees a consistent DB

Transactions - ACID

- **A**tomicity
  - Either all or no commands of transaction are executed (their changes are persisted in the DB)

- **C**onsistency
  - After transaction DB is consistent (if before consistent)

- **I**solation
  - Transactions are running isolated from each other

- **D**urability
  - Modifications of transactions are never lost

Notes 12 - Transaction Management

16

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How can constraints be violated?

- Transaction bug
- DBMS bug
- Hardware failure

  e.g., disk crash alters balance of account

- Data sharing

  e.g.: T1: give 10% raise to programmers

  T2: change programmers $\Rightarrow$ systems analysts

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

How can we prevent/fix violations?

- Part 13 (Recovery):
  - due to failures
- Part 14 (Concurrency Control):
  - due to data sharing

Will not consider:

- How to write correct transactions
- How to write correct DBMS
- Constraint checking & repair

That is, solutions studied here do not need to know constraints

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Data Items:

- **Data Item** / **Database Object** / …
- Abstraction that will come in handy when talking about concurrency control and recovery
- Data Item could be
  - Table, Row, Page, Attribute value

Operations:

- Input (x):   block containing x → memory
- Output (x): block containing x → disk

CS 525

Notes 12 - Transaction Management

21

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Operations:

- Input (x):   block containing x $\rightarrow$ memory
- Output (x): block containing x $\rightarrow$ disk

- Read (x,t): do input(x) if necessary
                    $t \leftarrow$ value of x in block
- Write (x,t): do input(x) if necessary
                    value of x in block $\leftarrow t$

CS 525

Notes 12 - Transaction Management

22

IIT College of
Science and Letters

COMPUTER SCIENCE

ILLINOIS INSTITUTE OF TECHNOLOGY

Key problem   Unfinished transaction (**Atomicity**)

Example   Constraint: A=B
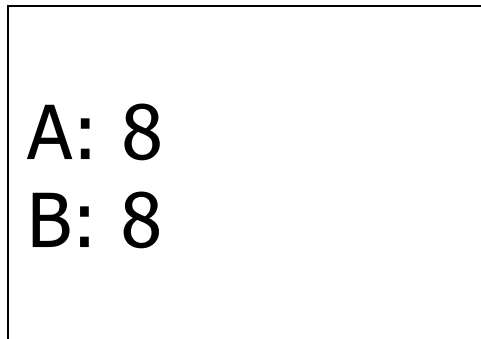
$$T_1: \quad A \leftarrow A \times 2$$
$$B \leftarrow B \times 2$$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

T1: Read (A,t); t ← t×2
    Write (A,t);
    Read (B,t); t ← t×2
    Write (B,t);
    Output (A);
    Output (B);

A: 8
B: 8

memory

A: 8
B: 8

disk

Notes 12 - Transaction
Management

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

T1: Read (A,t); t ← t×2
Write (A,t);
Read (B,t); t ← t×2
Write (B,t);
Output (A);
Output (B);

A: ~~8~~ 16
B: ~~8~~ 16

memory

A: 8
B: 8

disk

T1:   Read (A,t); t ← t×2
      Write (A,t);
      Read (B,t); t ← t×2
      Write (B,t);
      Output (A);
      Output (B);   failure!

memory

A: 8̶ 16
B: 8̶ 16

disk

A: 8̶ 16
B: 8

Notes 12 - Transaction Management

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
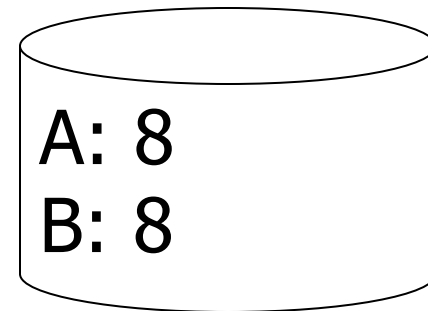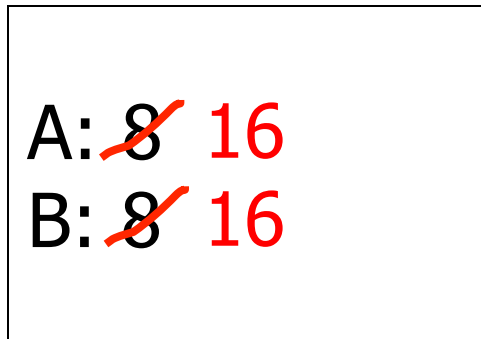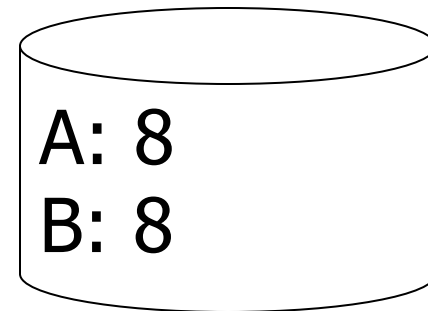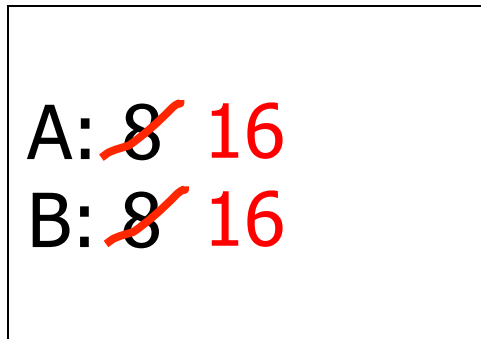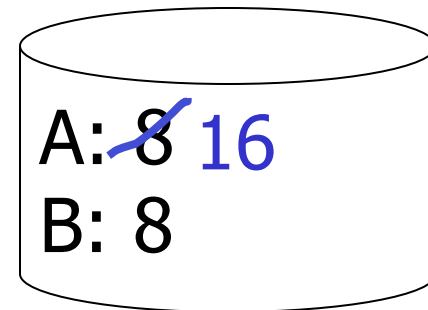
Transactions in SQL

- BEGIN WORK
  - Start new transaction
  - Often implicit

- COMMIT
  - Finish and make all modifications of transactions persistent

- ABORT/ROLLBACK
  - Finish and undo all changes of transaction

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example

time

```
BEGIN WORK;
  UPDATE accounts
    SET bal = bal + 40
    WHERE acc = 10;



  UPDATE accounts
    SET bal = bal - 40
    WHERE acc = 9;
COMMIT;
```

```
BEGIN WORK;
  UPDATE accounts
    SET bal = bal * 1.05;
COMMIT;
```

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Example

time

```
BEGIN WORK;
   UPDATE accounts
      SET bal = bal + 40
      WHERE acc = 10;



   UPDATE accounts
      SET bal = bal - 40
      WHERE acc = 9;
COMMIT;
```

Bank customer transfers money from account 9 to account 10

```
BEGIN WORK;
   UPDATE accounts
      SET bal = bal * 1.05;
COMMIT;
```

CS 525

Notes 12 - Transaction Management

29

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example

Bank adds interest to all accounts

time

```
BEGIN WORK;
   UPDATE accounts
      SET bal = bal + 40
      WHERE acc = 10;



   UPDATE accounts
      SET bal = bal − 40
      WHERE acc = 9;
COMMIT;
```

```
BEGIN WORK;
   UPDATE accounts
      SET bal = bal * 1.05;
COMMIT;
```

COMPUTER SCIENCE

Notes 12 - Transaction Management

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

time

```
BEGIN WORK;
  UPDATE accounts
    SET bal = bal
    WHERE acc = 10


  UPDATE accounts
    SET bal = bal – 40
    WHERE acc = 9;
COMMIT;
```

Potential Problems:
1. Transactions are interrupted
   • No reduction in bal of acc 9
   • Only some accounts got interest
2. Interleaving of Transaction
   • Acc 9 too much interest (before 40 has been deducted)

```
          SET bal = bal * 1.05;
COMMIT;
```

Notes 12 - Transaction Management

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Modeling Transactions and their Interleaving

- Transaction is sequence of operations
  - **read**: $r_i(x)$ = transaction **i** read item **x**
  - **write**: $w_i(x)$ = transaction **i** wrote item **x**
  - **commit**: c_i = transaction **i** committed
  - **abort:** a_i = transaction **i** aborted

$$T_1 = r_1(a_{10}), w_1(a_{10}), r_1(a_9), w_1(a_9), c_1$$

time

```
BEGIN WORK;
  UPDATE accounts
    SET bal = bal + 40
    WHERE acc = 10;




  UPDATE accounts
    SET bal = bal − 40
    WHERE acc = 9;
COMMIT;
```

$$T_1 = r_1(a_{10}), w_1(a_{10}), r_1(a_9), w_1(a_9), c_1$$

$$T_2 = r_2(a_1), w_2(a_1), r_2(a_2), w_2(a_2), r_2(a_9), w_2(a_9), r_2(a_{10}), w_2(a_{10}), c_1$$

```
BEGIN WORK;
   UPDATE accounts
      SET bal = bal + 40
      WHERE acc = 10;



   UPDATE accounts
      SET bal = bal - 40
      WHERE acc = 9;
COMMIT;
```

Assume we have accounts: a_1, a_2, a_9, a_{10}

```
BEGIN WORK;
   UPDATE accounts
      SET bal = bal * 1.05;
COMMIT;
```

Schedules

- A **schedule S** for a set of transactions T = $\{T_1, ..., T_n\}$ is an partial order over operations of T so that

  - **S** contains a prefix of the operations of each T_i

  - Operations of Ti appear in the same order in **S** as in Ti

  - For any two conflicting operations they are ordered

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Note

- For simplicity: We often assume that the schedule is a total order

CS 525

Notes 12 - Transaction Management

36

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

How to model execution order?

- Schedules model the order of the execution for operations of a set of transactions

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Conflicting Operations

- Two operations are conflicting if
  - At least one of them is a write
  - Both are accessing the same data item
- Intuition
  - The order of execution for conflicting operations can influence result!

COMPUTER
SCIENCE

Notes 12 - Transaction
Management

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Conflicting Operations

- Examples
  - $w_1(X)$, $r_2(X)$ are conflicting
  - $w_1(X)$, $w_2(Y)$ are not conflicting
  - $r_1(X)$, $r_2(X)$ are not conflicting
  - $w_1(X)$, $w_1(X)$ are not conflicting

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Complete Schedules = History

- A **schedule S** for T is complete if it contains all operations from each transaction in T

- We will call complete schedules **histories**

Notes 12 - Transaction
Management

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

$$T_1 = r_1(a_{10}), w_1(a_{10}), r_1(a_9), w_1(a_9), c_1$$

$$T_2 = r_2(a_1), w_2(a_1), r_2(a_2), w_2(a_2), r_2(a_9), w_2(a_9), r_2(a_{10}), w_2(a_{10}), c_1$$

Complete Schedule

$$S = r_2(a_1), r_1(a_{10}), w_2(a_1), r_2(a_2), w_1(a_{10}), w_2(a_2), r_2(a_9), w_2(a_9),$$
$$r_1(a_9), w_1(a_9), c_1 \ r_2(a_{10}), w_2(a_{10}), c_1$$

Incomplete Schedule

$$S = r_2(a_1), r_1(a_{10}), w_2(a_1), w_1(a_{10})$$

Not a Schedule

$$S = r_2(a_1), r_1(a_{10}), c_1$$

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

$$T_1 = r_1(a_{10}), w_1(a_{10}), r_1(a_9), w_1(a_9), c_1$$

$$T_2 = r_2(a_1), w_2(a_1), r_2(a_2), w_2(a_2), r_2(a_9), w_2(a_9), r_2(a_{10}), w_2(a_{10}), c_1$$

Conflicting operations

- Conflicting operations $w_1(a_{10})$ and $w_2(a_{10})$
- Order of these operations determines value of a_{10}
- S1 and S2 do not generate the same result

$$S_1 = \ldots w_2(a_1) \ldots w_1(a_{10})$$

$$S_2 = \ldots w_1(a_1) \ldots w_2(a_{10})$$

Notes 12 - Transaction Management

Why Schedules?

- Study properties of different execution orders
  - Easy/Possible to recover after failure
  - Isolation
  - -> preserve ACID properties
- Classes of schedules and protocols to guarantee that only "good" schedules are produced

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

CS 525: Advanced Database Organization
Organization
13: Failure and Recovery

Boris Glavic

Slides: adapted from a course taught by
Hector Garcia-Molina, Stanford InfoLab

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Now

- Crash recovery

Correctness (informally)

- If we stop running transactions,
    DB left consistent

- Each transaction sees a consistent DB

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

How can constraints be violated?

- Transaction bug

- DBMS bug

- Hardware failure

  e.g., disk crash alters balance of account

- Data sharing

  e.g.: T1: give 10% raise to programmers

  T2: change programmers $\Rightarrow$ systems analysts

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Recovery

- First order of business:
  <u>Failure Model</u>
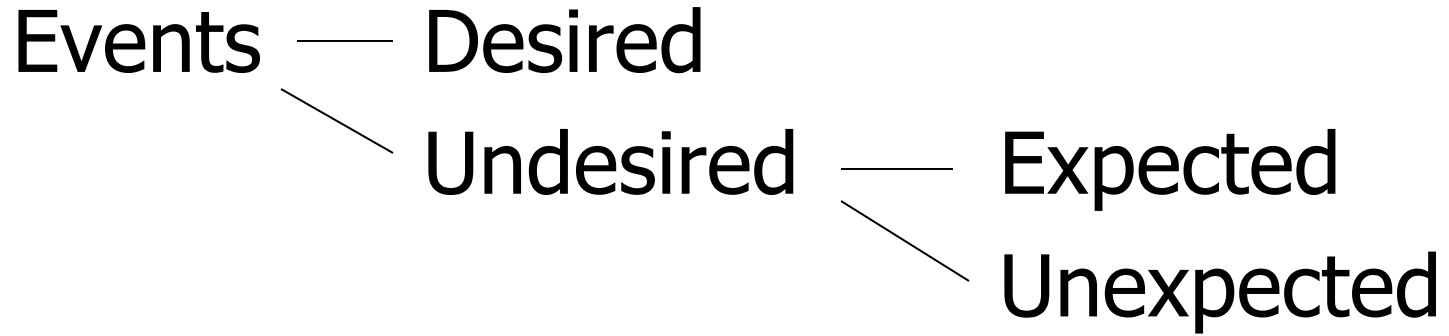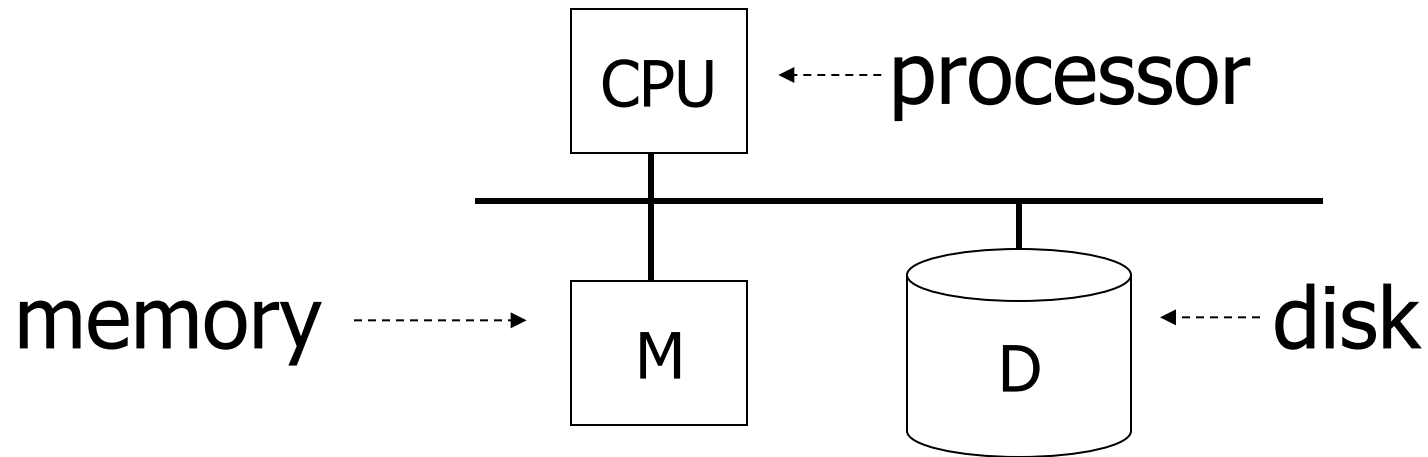
IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Events —— Desired

Undesired —— Expected

Unexpected

COMPUTER SCIENCE

Notes 13 - Failure and Recovery   6

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Our failure model

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Desired events: see product manuals....

Undesired expected events:

System crash

- memory lost

- cpu halts, resets

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Desired events: see product manuals….

Undesired expected events:
System crash

- memory lost

- cpu halts, resets

═══════════════that's it!!═══════════════

Undesired Unexpected: Everything else!

Undesired Unexpected: Everything else!

Examples:

- Disk data is lost
- Memory lost without CPU halt
- CPU implodes wiping out universe....

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Is this model reasonable?

Approach: Add low level checks +
                 redundancy to increase

                probability model holds

E.g., Replicate disk storage (stable store)

      Memory parity

      CPU checks

Second order of business:

Storage hierarchy
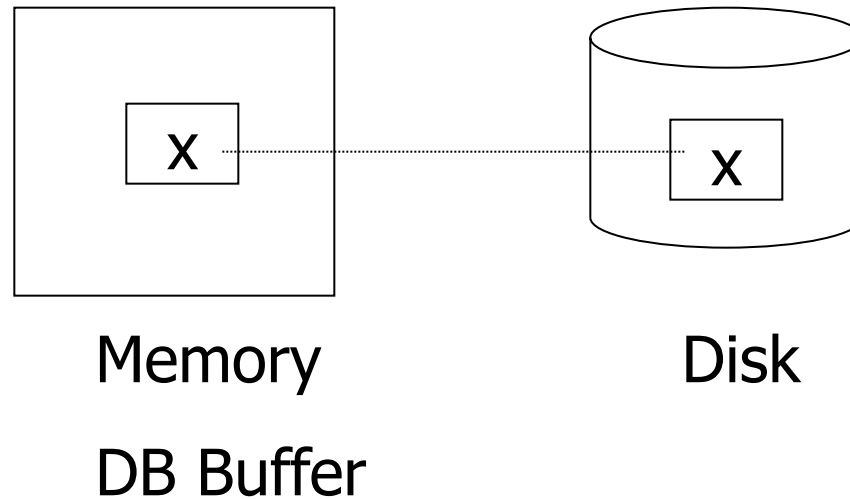
Memory

Disk

DB Buffer

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Operations:

- Input (x):   block containing x $\to$ memory
- Output (x): block containing x $\to$ disk

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Operations:

- Input (x): block containing x $\rightarrow$ memory
- Output (x): block containing x $\rightarrow$ disk

- Read (x,t): do input(x) if necessary

                    t $\leftarrow$ value of x in block
- Write (x,t): do input(x) if necessary

                  value of x in block $\leftarrow$ t

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Key problem — Unfinished transaction

Example   Constraint: A=B

$$T_1: \quad A \leftarrow A \times 2$$

$$B \leftarrow B \times 2$$

T1:   Read (A,t); t ← t×2
        Write (A,t);
        Read (B,t); t ← t×2
        Write (B,t);
        Output (A);
        Output (B);
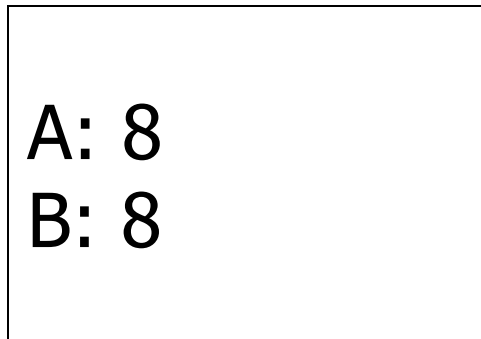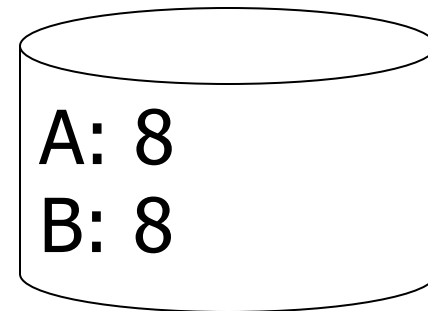
A: 8
B: 8

memory

A: 8
B: 8

disk

Notes 13 - Failure and Recovery   16

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

T1:   Read (A,t); t ← t×2
      Write (A,t);
      Read (B,t); t ← t×2
      Write (B,t);
      Output (A);
      Output (B);

A: ~~8~~ 16
B: ~~8~~ 16

memory

A: 8
B: 8

disk

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

T1: Read (A,t); t ← t×2
     Write (A,t);
     Read (B,t); t ← t×2
     Write (B,t);
     Output (A);
     Output (B);   failure!

memory

A: ~~8~~ 16
B: ~~8~~ 16

disk

A: ~~8~~ 16
B: 8
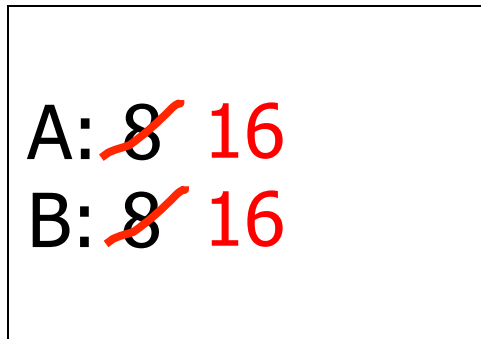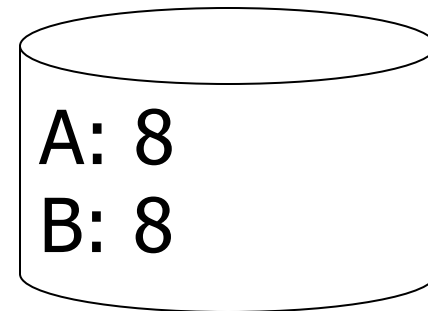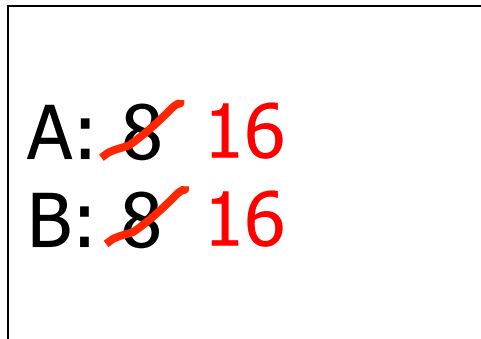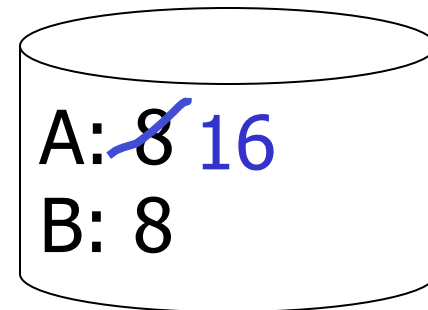
- Need <u>atomicity:</u>
  - execute all actions of a transaction or none at all

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How to restore consistent state after crash?

- Desired state after recovery:
  - Changes of committed transactions are reflected on disk
  - Changes of unfinished transactions are not reflected on disk
- After crash we need to
  - **Undo** changes of unfinished transactions that have been written to disk
  - **Redo** changes of finished transactions that have not been written to disk

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How to restore consistent state after crash?

- After crash we need to
  - **Undo** changes of unfinished transactions that have been written to disk
  - **Redo** changes of finished transactions that have not been written to disk

- We need to either
  - Store additional data to be able to Undo/Redo
  - Avoid ending up in situations where we need to Undo/Redo

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

T1: Read (A,t); t ← t×2
Write (A,t);
Read (B,t); t ← t×2
Write (B,t);
Output (A);
Output (B); ~~~~~ failure!

T_1 is unfinished
-> need to undo the write to A to recover to consistent state

memory

A: ~~8~~ 16
B: 8

disk

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
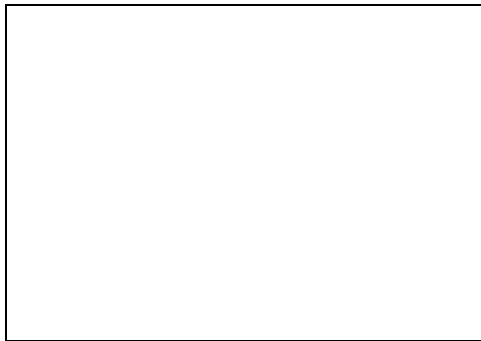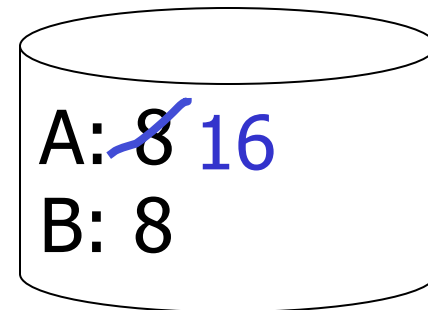
Logging

- After crash need to
  - **Undo**
  - **Redo**

- We need to know
  - Which operations have been executed
  - Which operations are reflected on disk

- ->**Log** upfront what is to be done

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Buffer Replacement Revisited

- Now we are interested in knowing how buffer replacement influences recovery!

COMPUTER SCIENCE

Notes 13 - Failure and Recovery   24

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Buffer Replacement Revisited

- **Steal**: all pages with fix count = 0 are replacement candidates
  - Smaller buffer requirements

- **No steal:** pages that have been modified by active transaction -> not considered for replacement
  - No need to undo operations of unfinished transactions after failure

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Buffer Replacement Revisited

- **Force**: Pages modified by transaction are flushed to disk at end of transaction
  - No redo required

- **No force:** modified (dirty) pages are allowed to remain in buffer after end of transaction
  - Less repeated writes of same page

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Effects of Buffer Replacement

	force	No force
No steal	• No Undo • No Redo	• No Undo • Redo
steal	• Undo • No Redo	• Redo • Undo

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Schedules and Recovery

- Are there certain schedules that are easy/hard/impossible to recover from?

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Recoverable Schedules

- We should never have to rollback an already committed transaction (D in ACID)

- **Recoverable** (**RC**) schedules require that

  – A transaction does not commit before every transaction that is has read from has committed

  – A transaction **T** reads from another transaction **T'** if it reads an item X that has last been written by T' and T' has not aborted before the read

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

$$T_1 = w_1(X), c_1$$

$$T_2 = r_2(X), w_2(X), c_2$$

Recoverable (RC) Schedule

$$S_1 = w_1(X), r_2(X), w_2(X), c_1, c_2$$

Nonrecoverable Schedule

$$S_2 = w_1(X), r_2(X), w_2(X), c_2, c_1$$

Cascading Abort

- Transaction **T** has written an item that is later read by **T'** and **T** aborts after that
  - we have to also abort **T'** because the value it read is no longer valid anymore
  - This is called a **cascading abort**
  - Cascading aborts are complex and should be avoided

$$S = \ldots w_1(X) \ldots r_2(X) \ldots a_1$$

COMPUTER
SCIENCE

Notes 13 - Failure and Recovery   31

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Cascadeless Schedules

- **Cascadeless (CL)** schedules guarantee that there are no cascading aborts
  - Transactions only read values written by already committed transactions

$$T_1 = w_1(X), c_1$$

$$T_2 = r_2(X), w_2(X), c_2$$

Cascadeless (CL) Schedule

$$S_1 = w_1(X), c_1, r_2(X), w_2(X), c_2$$

Recoverable (RC) Schedule

$$S_2 = w_1(X), r_2(X), w_2(X), c_1, c_2$$

Nonrecoverable Schedule

$$S_3 = w_1(X), r_2(X), w_2(X), c_2, c_1$$

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

$T_1 = w_1(X), a_1$

$T_2 = r_2(X), w_2(X), c_2$

Cascadeless (CL) Schedule

$S_1 = w_1(X), a_1, r_2(X), w_2(X), c_2$

Recoverable (RC) Schedule

$S_2 = w_1(X), r_2(X), w_2(X), a_1, a_2$

Nonrecoverable Schedule

$S_3 = w_1(X), r_2(X), w_2(X), c_2, a_1$

Consider what happens if T1 aborts!

COMPUTER SCIENCE

Notes 12 - Transaction Management

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Strict Schedules

- **Strict (ST)** schedules guarantee that to Undo the effect of an transaction we simply have to undo each of its writes
  - Transactions do not read nor write items written by uncommitted transactions

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

$T_1 = w_1(X), c_1$

$T_2 = r_2(X), w_2(X), c_2$

Cascadeless (CL) + Strict Schedule (ST)

$S_1 = w_1(X), c_1, r_2(X), w_2(X), c_2$

Recoverable (RC) Schedule

$S_2 = w_1(X), r_2(X), w_2(X), c_1, c_2$

Nonrecoverable Schedule

$S_3 = w_1(X), r_2(X), w_2(X), c_2, c_1$

Compare Classes

$$ST \subset CL \subset RC \subset ALL$$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

All schedules

Recoverable schedules

Cascadeless schedules

Strict
schedules

COMPUTER
SCIENCE

Notes 13 - Failure and Recovery   38

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
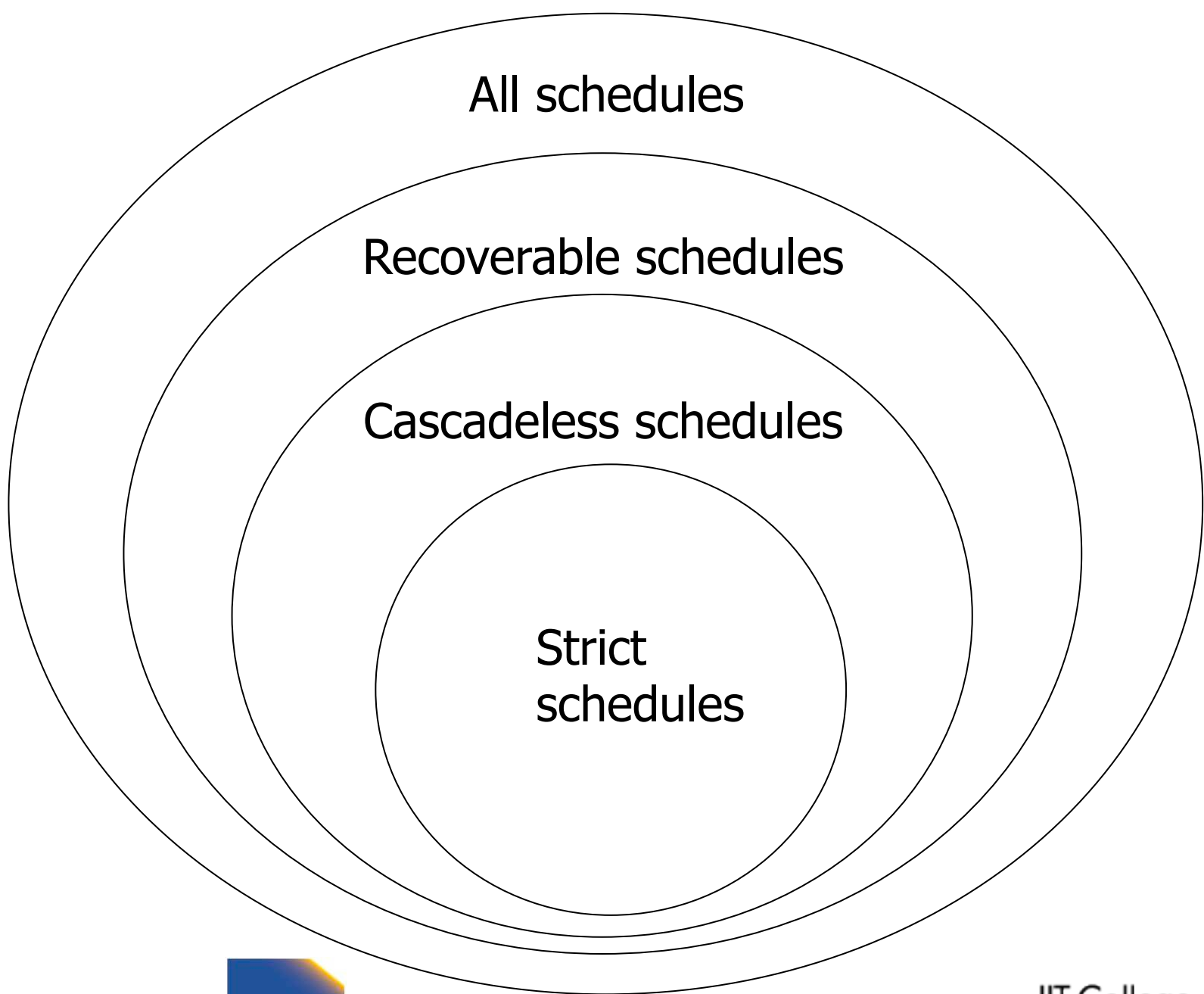
Logging and Recovery

- We now discuss approaches for logging and how to use them in recovery

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

One solution: undo logging (immediate
modification)

due to: Hansel and Gretel, 782 AD

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

One solution: undo logging (immediate modification)

due to: Hansel and Gretel, 782 AD

- Improved in 784 AD to durable undo logging

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Undo logging (Immediate modification)

T1: Read (A,t); t ← t×2   A=B
     Write (A,t);
     Read (B,t); t ← t×2
     Write (B,t);
     Output (A);
     Output (B);

A:8
B:8

memory

A:8
B:8

disk

log

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Undo logging (Immediate modification)

T1: Read (A,t); t ← t×2   A=B
     Write (A,t);
     Read (B,t); t ← t×2
     Write (B,t);
     Output (A);
     Output (B);

A:~~8~~ 16
B:~~8~~ 16

memory

A:8
B:8

disk

<T1, start>
<T1, A, 8>

log

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Undo logging (Immediate modification)

T1:   Read (A,t); t ← t×2   A=B
       Write (A,t);
       Read (B,t); t ← t×2
       Write (B,t);
       Output (A);
       Output (B);

A:~~8~~ 16
B:~~8~~ 16

memory

A:~~8~~ 16
B:8

disk

<T1, start>
<T1, A, 8>
<T1, B, 8>

log

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Undo logging (Immediate modification)

T1: Read (A,t); t ← t×2   A=B
     Write (A,t);
     Read (B,t); t ← t×2
     Write (B,t);
     Output (A);
     Output (B);

A: ~~8~~ 16
B: ~~8~~ 16

memory

A: ~~8~~ 16
B: ~~8~~ 16

disk

<T1, start>
<T1, A, 8>
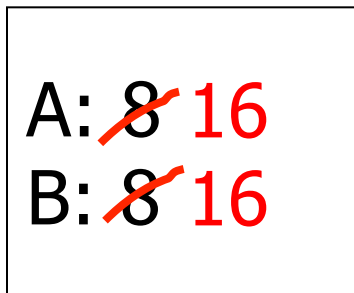<T1, B, 8>

log

Undo logging (Immediate modification)

T1:   Read (A,t); t ← t×2   A=B
      Write (A,t);
      Read (B,t); t ← t×2
      Write (B,t);
      Output (A);
      Output (B);

A:~~8~~ 16
B:~~8~~ 16

memory

A:~~8~~ 16
B:~~8~~ 16

disk

<T1, start>
<T1, A, 8>
<T1, B, 8>
<T1, commit>

log

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

One "complication"

- Log is first written in memory
- Not written to disk on every action

memory

A: ~~8~~ 16
B: ~~8~~ 16
Log:
<T1,start>
<T1, A, 8>
<T1, B, 8>

A: 8
B: 8

DB

Log

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

One "complication"

- Log is first written in memory
- Not written to disk on every action

memory

A: ~~8~~ 16
B: ~~8~~ 16
Log:
<T1,start>
<T1, A, 8>
<T1, B, 8>

A: ~~8~~ 16
B: 8

DB

Log

BAD STATE
1

One "complication"

- Log is first written in memory
- Not written to disk on every action

memory

A: ~~8~~ 16
B: ~~8~~ 16
Log:
<T1,start>
<T1, A, 8>
<T1, B, 8>
<T1, commit>

A: ~~8~~ 16
B: 8

DB

BAD STATE # 2

Log

<T1, B, 8>
<T1, commit>

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
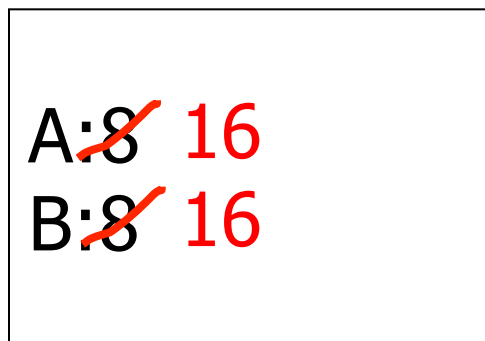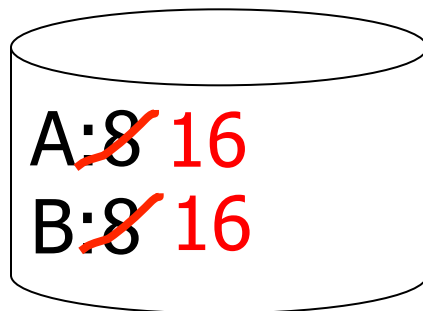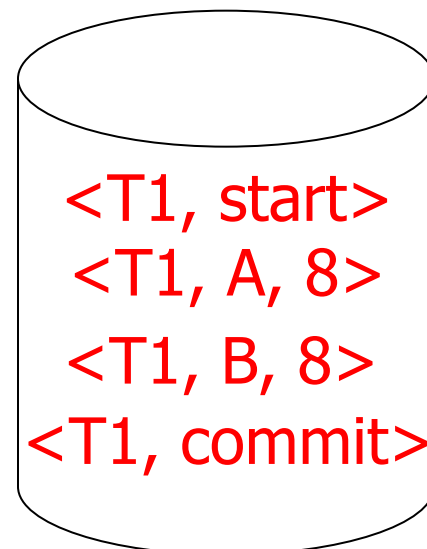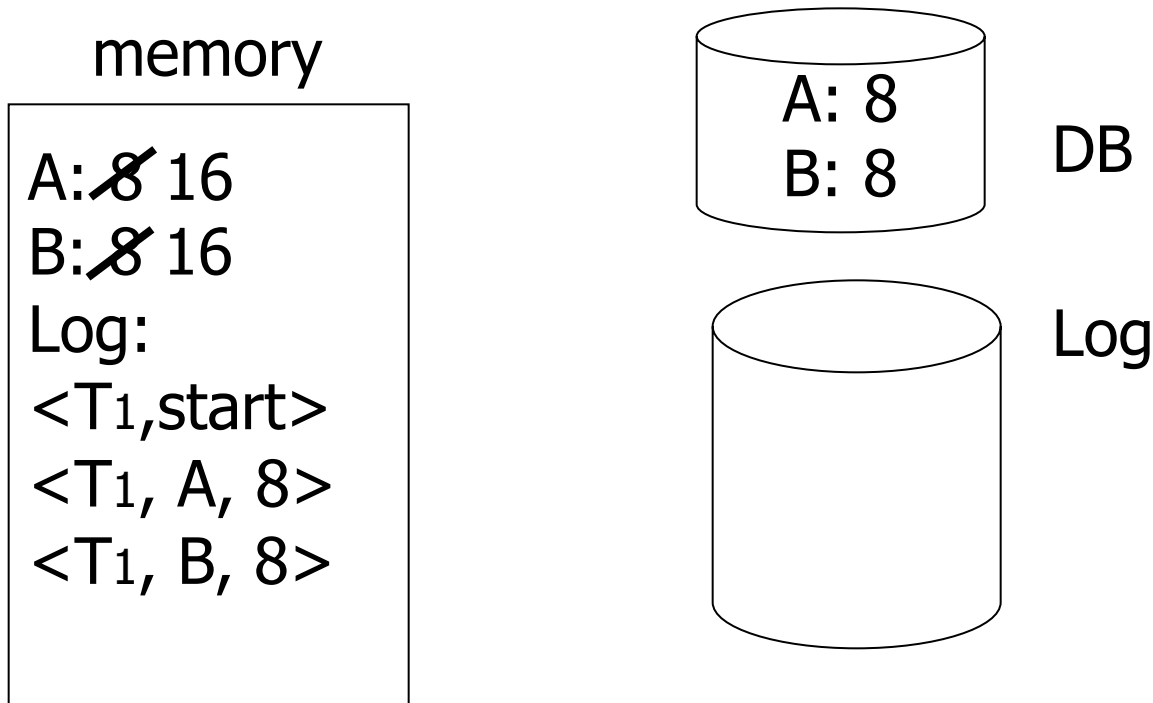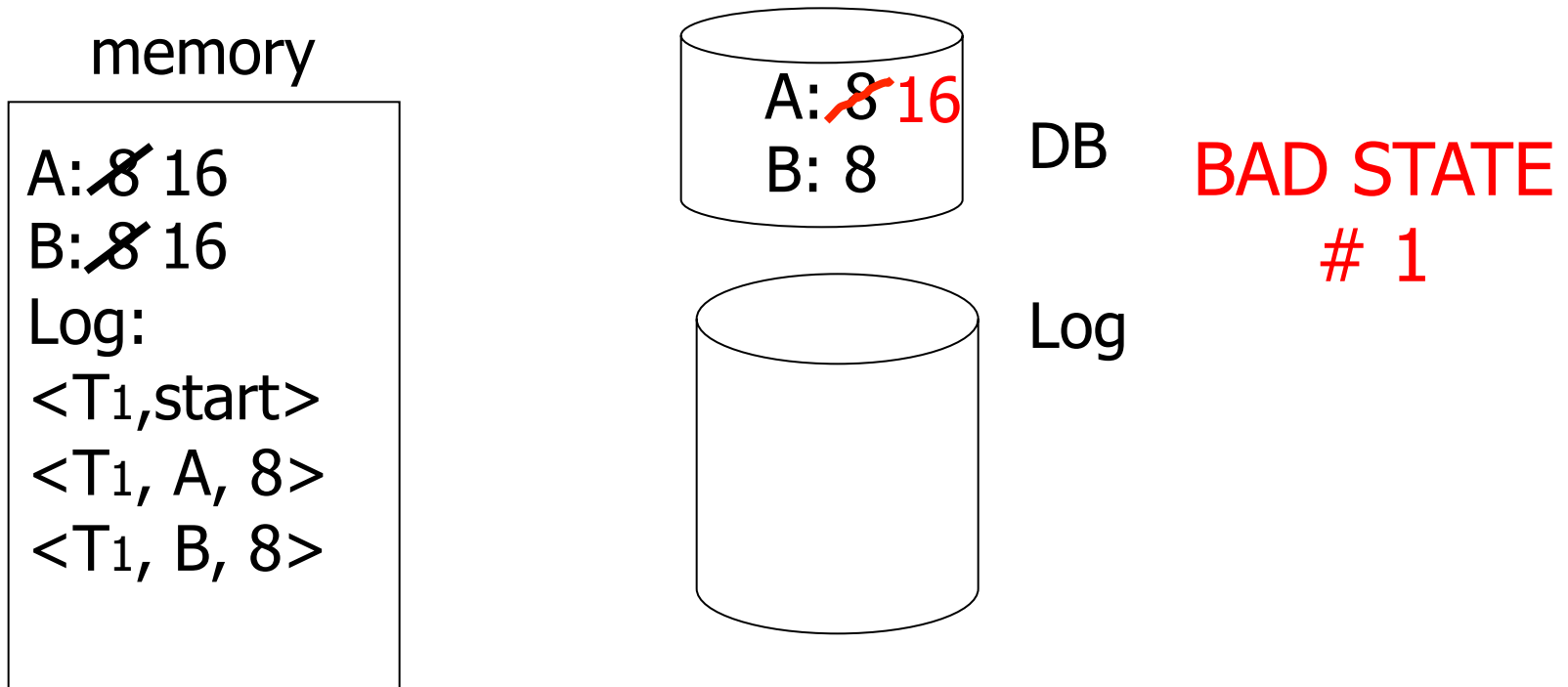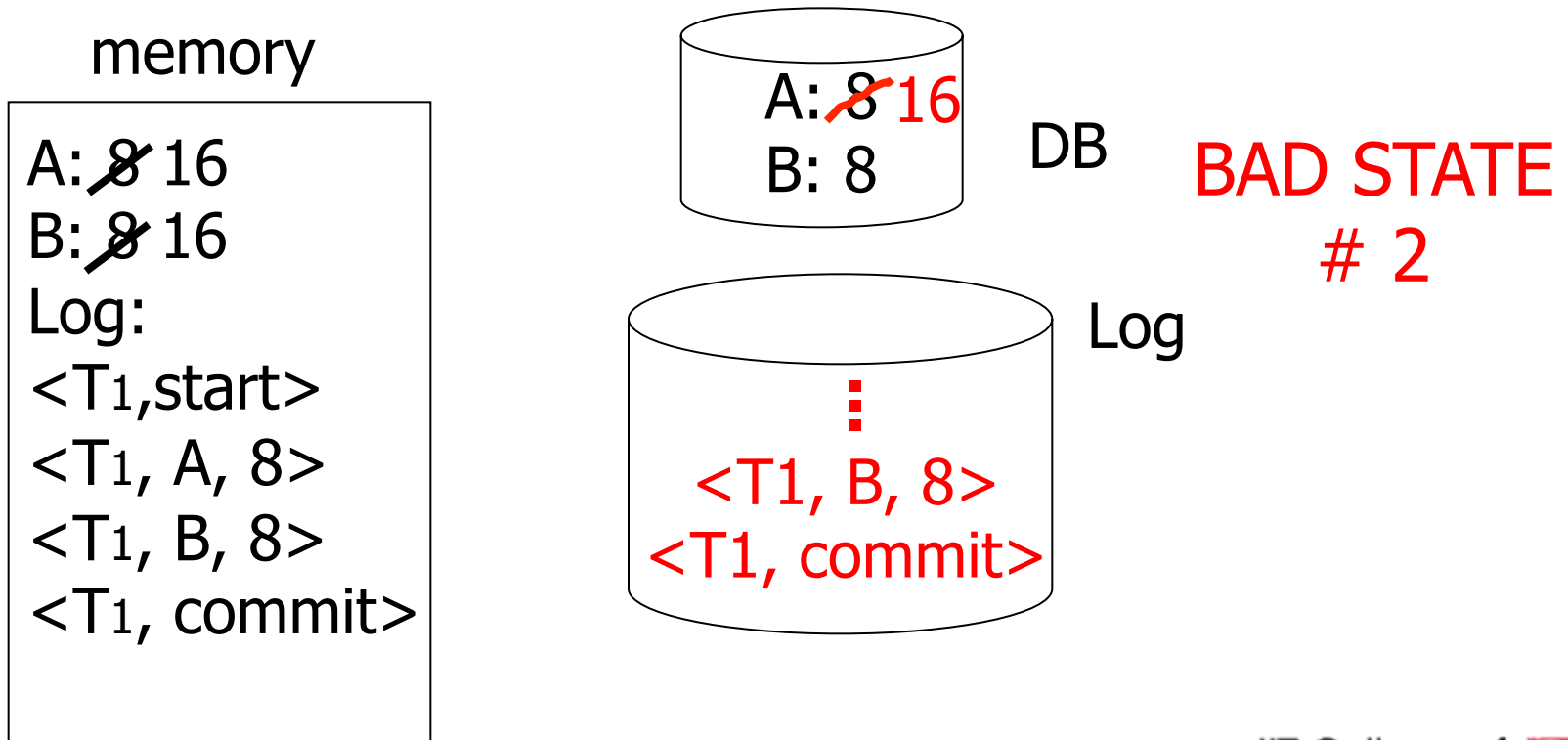
Undo logging rules

(1) For every action generate undo log record (containing old value)

(2) Before x is modified on disk, log records pertaining to x must be

 on disk (write ahead logging: **WAL**)

(3) Before commit is flushed to log, all writes of transaction must be

 reflected on disk

Recovery rules: Undo logging

- For every Ti   with <Ti, start> in log:
  - If <Ti,commit> or <Ti,abort>
    in log, do nothing
  - Else ⎰ For all <Ti, X, v> in log:
    - write (X, v)
    - output (X)
    - Write <Ti, abort> to log

Recovery rules: Undo logging

- For every Ti with <Ti, start> in log:
  - If <Ti,commit> or <Ti,abort> in log, do nothing
  - Else ⎰ For all <Ti, *X*, *v*> in log:
    - { write *(X, v)*
    - output *(X)*
    - Write <Ti, abort> to log

➡️IS THIS CORRECT??

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Recovery rules: Undo logging

(1) Let S = set of transactions with
      <Ti, start> in log, but no

      <Ti, commit> (or <Ti, abort>) record in log

(2) For each <Ti, X, v> in log,

   in reverse order (latest $\rightarrow$ earliest) do:

     - if Ti $\in$ S then   - write (X, v)

                          - output (X)

(3) For each Ti $\in$ S do

     - write <Ti, abort> to log

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Question

- Can writes of <Ti, abort> records be done in any order (in Step 3)?
  - Example: T1 and T2 both write A
  - T1 executed before T2
  - T1 and T2 both rolled-back
  - <T1, abort> written but NOT <T2, abort>?
  - <T2, abort> written but NOT <T1, abort>?

```
    |------------↑---------------------↑-------------→
            T1 write A            T2 write A        time/log
```

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
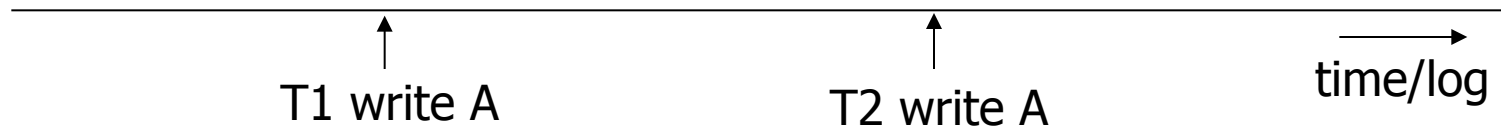
What if failure during recovery?

No problem! ✏ Undo <u>idempotent</u>

- An operation is called **idempotent** if the number of times it is applied do not effect the result
- For Undo:
  - Undo(log) = Undo(Undo(… (Undo(log)) …))

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Undo is idempotent

- We store the values of data items before the operation
- Undo can be executed repeatedly without changing effects
  - idempotent

Physical vs. Logical Logging

- How to represent values in log entries?
- Physical logging
  - Content of pages before and after
- Logical operations
  - Operation to execute for undo/redo
    - E.g., delete record x
- Hybrid (Physiological)
  - Delete record x from page y

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

To discuss:

- Redo logging
- Undo/redo logging, why both?
- Real world actions
- Checkpoints
- Media failures

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Redo logging (deferred modification)

T1:   Read(A,t); t← t×2; write (A,t);
       Read(B,t); t←t×2; write (B,t);
       Output(A); Output(B)

A: 8
B: 8

memory

A: 8
B: 8

DB

LOG

Redo logging (deferred modification)

T1:   Read(A,t); t ← t×2; write (A,t);
      Read(B,t); t ← t×2; write (B,t);
      Output(A); Output(B)

A: 8 16
B: 8 16

memory

A: 8
B: 8

DB

<T1, start>
<T1, A, 16>
<T1, B, 16>
<T1, commit>

LOG

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Redo logging (deferred modification)

T1: Read(A,t); t← t×2; write (A,t);
     Read(B,t); t←t×2; write (B,t);
     Output(A); Output(B)

A: ~~8~~ 16
B: ~~8~~ 16

memory

output →

A: ~~8~~ 16
B: ~~8~~ 16

DB

<T1, start>
<T1, A, 16>
<T1, B, 16>
<T1, commit>

LOG

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Redo logging (deferred modification)

T1:   Read(A,t); t← t×2; write (A,t);
      Read(B,t); t←t×2; write (B,t);
      Output(A); Output(B)

A: ~~8~~ 16
B: ~~8~~ 16

memory

output →

A: ~~8~~ 16
B: ~~8~~ 16

DB

<T1, start>
<T1, A, 16>
<T1, B, 16>
<T1, commit>
<T1, end>

LOG

COMPUTER SCIENCE
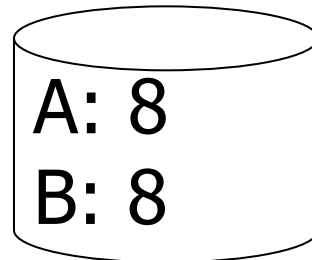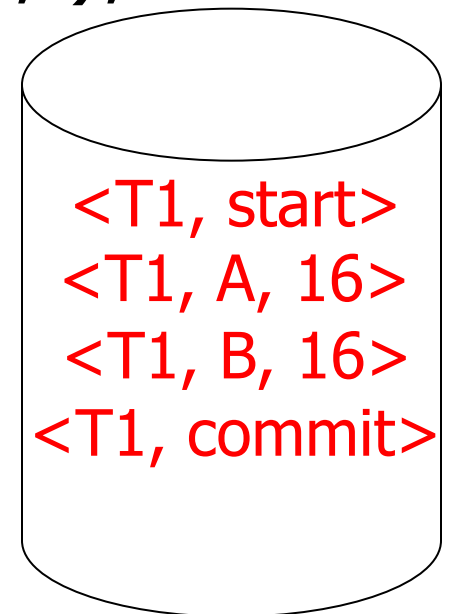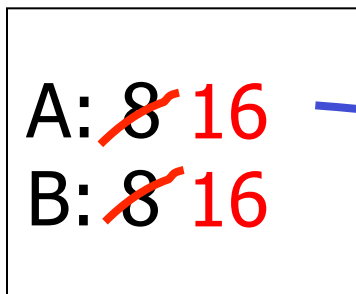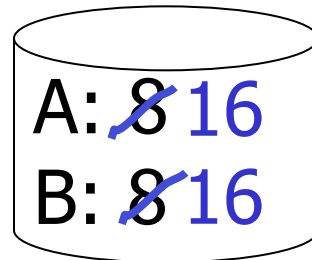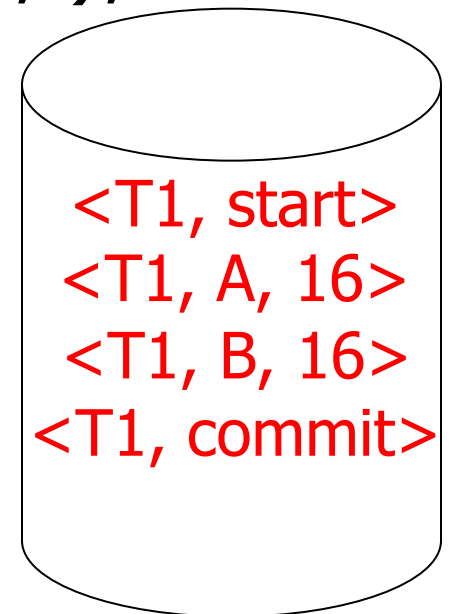
IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Redo logging rules

(1) For every action, generate redo log record (containing new value)

(2) Before X is modified on disk (DB), all log records for transaction that modified X (including commit) must be on disk

(3) Flush log at commit

(4) Write END record after DB updates flushed to disk

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Recovery rules: Redo logging

- For every Ti with <Ti, commit> in log:
  - For all <Ti, X, v> in log:

    $\left\{\begin{array}{l}\text{Write(X, v)} \\ \text{Output(X)}\end{array}\right.$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Recovery rules: Redo logging

- For every Ti with <Ti, commit> in log:
    - For all <Ti, X, v> in log:
        $\Big\{$ Write(X, v)

            Output(X)

➼IS THIS CORRECT??

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Recovery rules: Redo logging

(1) Let S = set of transactions with
   $\langle$Ti, commit$\rangle$ (and no $\langle$Ti, end$\rangle$) in log

(2) For each $\langle$Ti, X, v$\rangle$ in log, in forward
   order (earliest $\rightarrow$ latest) do:
    - if Ti $\in$ S then $\begin{cases} \text{Write(X, v)} \\ \text{Output(X)} \end{cases}$

(3) For each Ti $\in$ S, write $\langle$Ti, end$\rangle$

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Crash During Redo

- Since Redo log contains values after writes, repeated application of a log entry does not change result
  - ->idempotent

Combining <Ti, end> Records
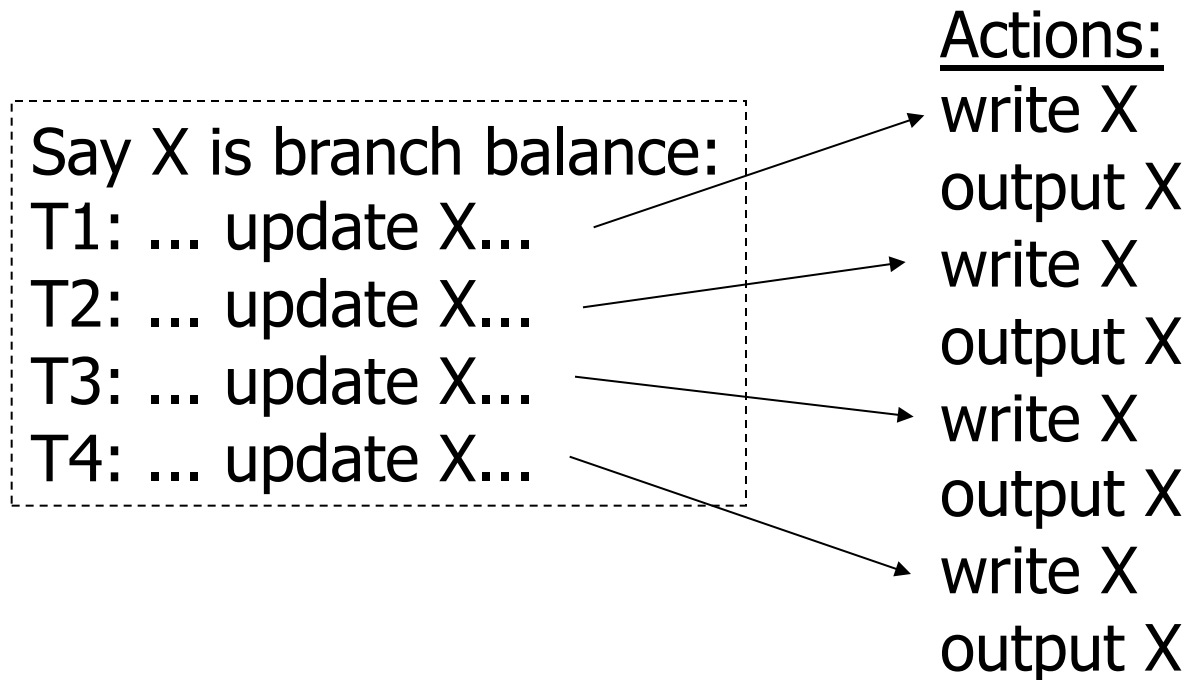
- Want to delay DB flushes for hot objects

Say X is branch balance:
T1: ... update X...
T2: ... update X...
T3: ... update X...
T4: ... update X...

Actions:
write X
output X
write X
output X
write X
output X
write X
output X

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Combining <Ti, end> Records

• Want to delay DB flushes for hot objects

Actions:

write X

~~output X~~

write X

~~output X~~

write X

~~output X~~

write X

output X

Say X is branch balance:
T1: … update X…
T2: … update X…
T3: … update X…
T4: … update X…

combined <end> (checkpoint)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
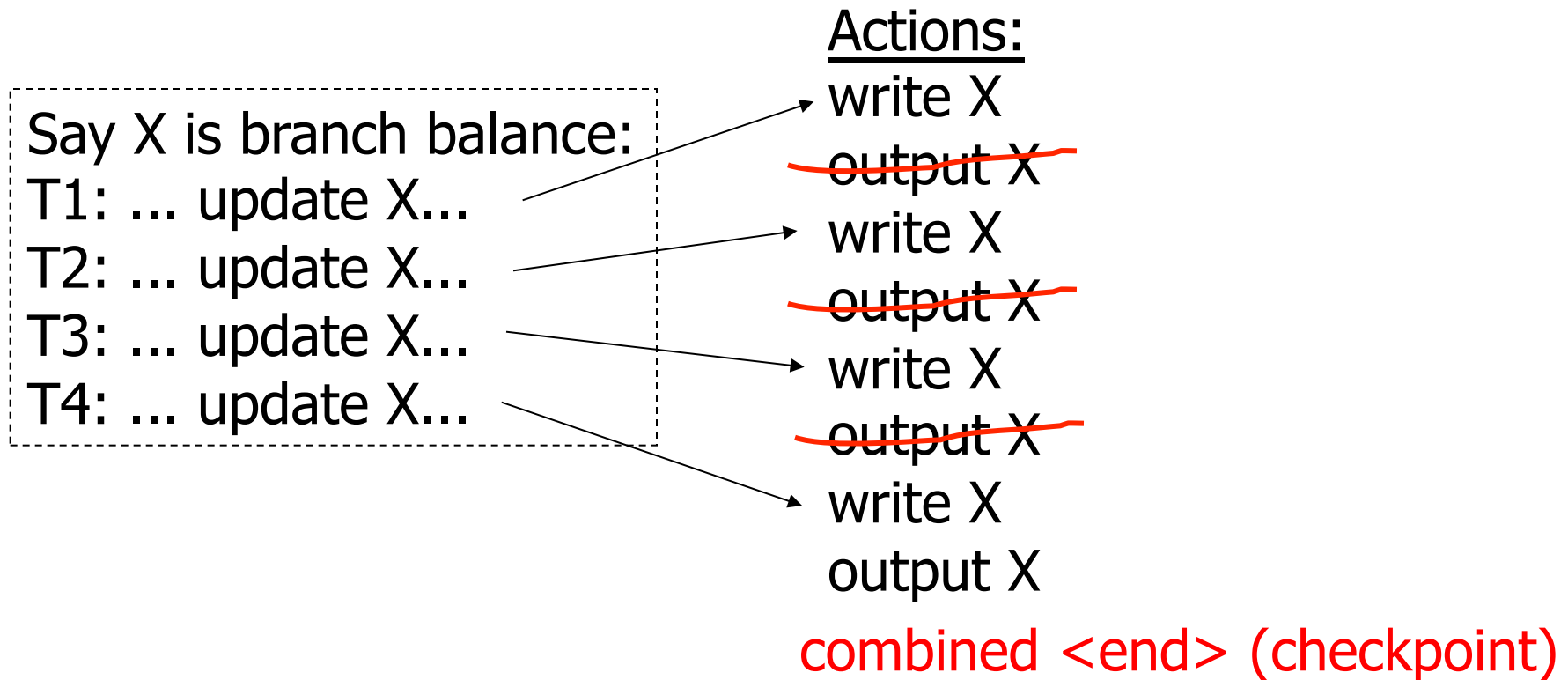
Solution: Checkpoint

Periodically:

(1) Do not accept new transactions

(2) Wait until all transactions finish

(3) Flush all log records to disk (log)

(4) Flush all buffers to disk (DB) (do not discard buffers)

(5) Write "checkpoint" record on disk (log)

(6) Resume transaction processing

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: what to do at recovery?

Redo log (disk):

| ... | <T1,A,16> | ... | <T1,commit> | ... | Checkpoint | ... | <T2,B,17> | ... | <T2,commit> | ... | <T3,C,21> | Crash |

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
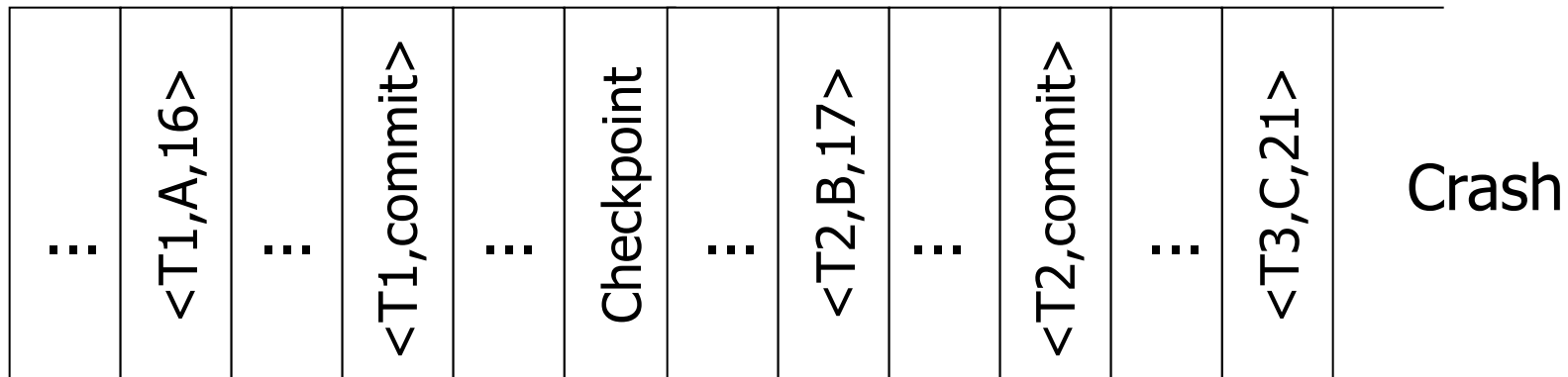
Advantage of Checkpoints

- Limits recovery to parts of the log after the checkpoint
  - Think about system that has been online for months
    - ->Analyzing the whole log is too expensive!

- Source of backups
  - If we backup checkpoints we can use them for media recovery!

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Checkpoints Justification

- Checkpoint should be consistent DB state
  - No active transactions
    - Do not accept new transactions
    - Wait until all transactions finish
  - DB state reflected on disk
    - Flush log
    - Flush buffers

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Key drawbacks:

- *Undo logging:*
  - cannot bring backup DB copies up to date
- *Redo logging:*
  - need to keep all modified blocks in memory until commit

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Solution: undo/redo logging!

Update $\Rightarrow$ <Ti, Xid, New X val, Old X val>
page X

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rules

- Page X can be flushed before or after Ti commit

- Log record flushed before corresponding updated page (WAL)

- Flush at commit (log only)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example: Undo/Redo logging
what to do at recovery?

log (disk):

...	\<checkpoint\>	...	\<T1, A, 10, 15\>	...	\<T1, B, 20, 23\>	...	\<T1, commit\>	...	\<T2, C, 30, 38\>	...	\<T2, D, 40, 41\>	Crash

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Checkpoint Cost

- Checkpoints are expensive
  - No new transactions can start
  - A lot of I/O
    - Flushing the log
    - Flushing dirty buffer pages
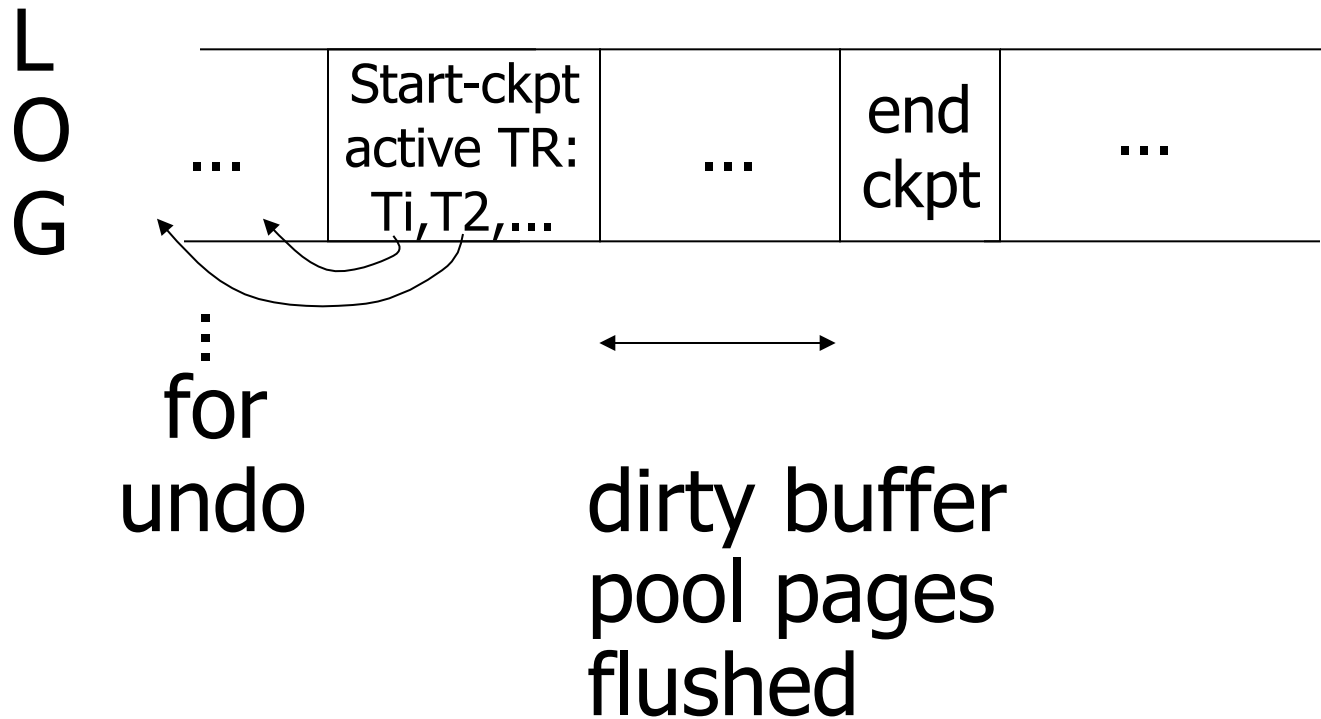
IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Non-quiesce checkpoint

L
O
G

| ... | Start-ckpt active TR: Ti,T2,... | ... | end ckpt | ... |

for undo

dirty buffer pool pages flushed

Notes 13 - Failure and Recovery

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Examples what to do at recovery time?

no T1 commit

L
O
G

| ... | T1,- a | ... | Ckpt T1 | ... | Ckpt end | ... | T1- b | |

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Examples   what to do at recovery time?

no T1 commit

L
O
G

...	T1,-a	...	Ckpt T1	...	Ckpt end	...	T1-b	
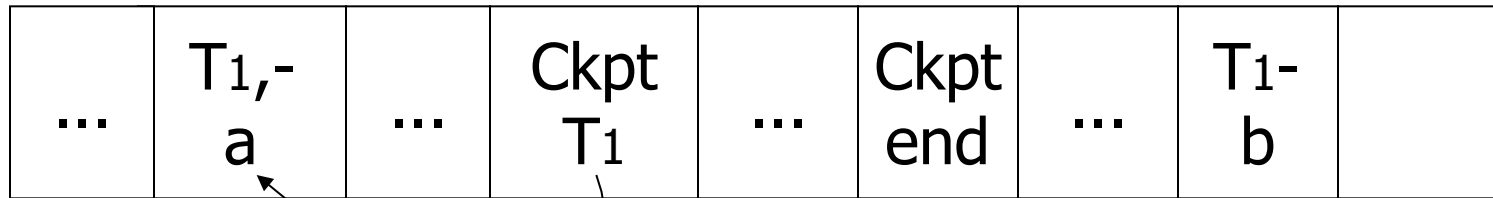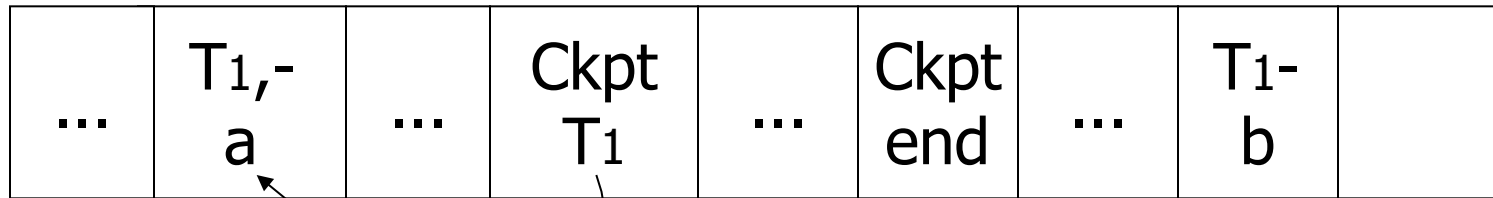
➼ Undo T1 (undo a,b)

Example

LOG

| ... | T1 a | ... | ckpt-s T1 | ... | T1 b | ... | ckpt-end | ... | T1 c | ... | T1 cmt | ... |

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Example

L
O
G

| ... | T1
a | ... | ckpt-s
T1 | ... | T1
b | ... | ckpt-
end | ... | T1
c | ... | T1
cmt | ... |
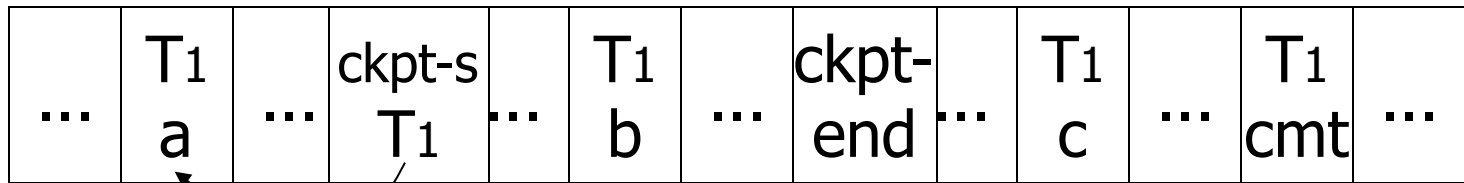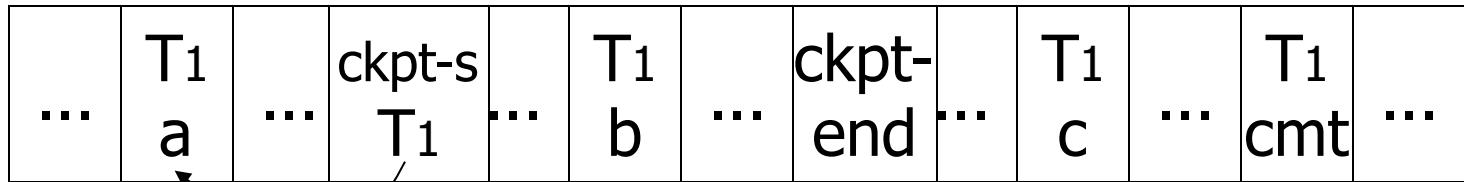
➤ Redo T1: (redo b,c)

Recover From <u>Valid</u> Checkpoint:

L O G	...	ckpt start	...	ckpt end	...	T1 b	...	ckpt- start	...	T1 c	...
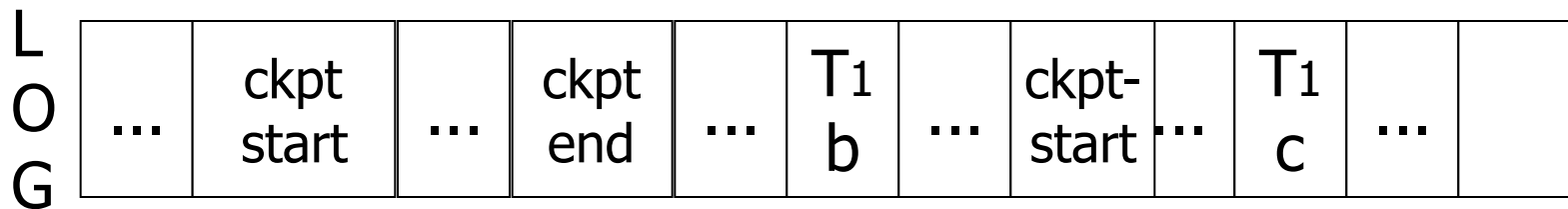
start
of latest
valid
checkpoint

Notes 13 - Failure and Recovery   84

Recovery process:

- Backwards pass (end of log ➜ latest valid checkpoint start)
  - construct set S of committed transactions
  - undo actions of transactions not in S

- Undo pending transactions
  - follow undo chains for transactions in (checkpoint active list) - S

- Forward pass (latest checkpoint start ➜ end of log)
  - redo actions of S transactions

start check-point

backward pass

forward pass

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
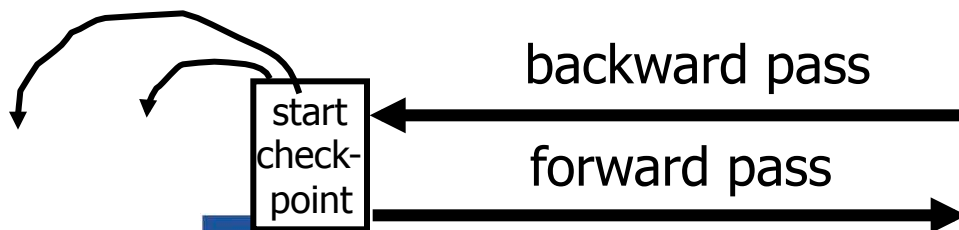
Real world actions

E.g., dispense cash at ATM

$$T_i = a_1 \, a_2 \, \ldots\ldots \, a_j \, \ldots\ldots \, a_n$$

$\downarrow$
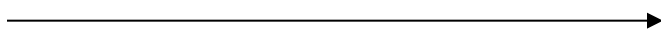
$\$$

Solution

(1) execute real-world actions after commit
(2) try to make idempotent

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

ATM

Give$$
(amt, Tid, time)
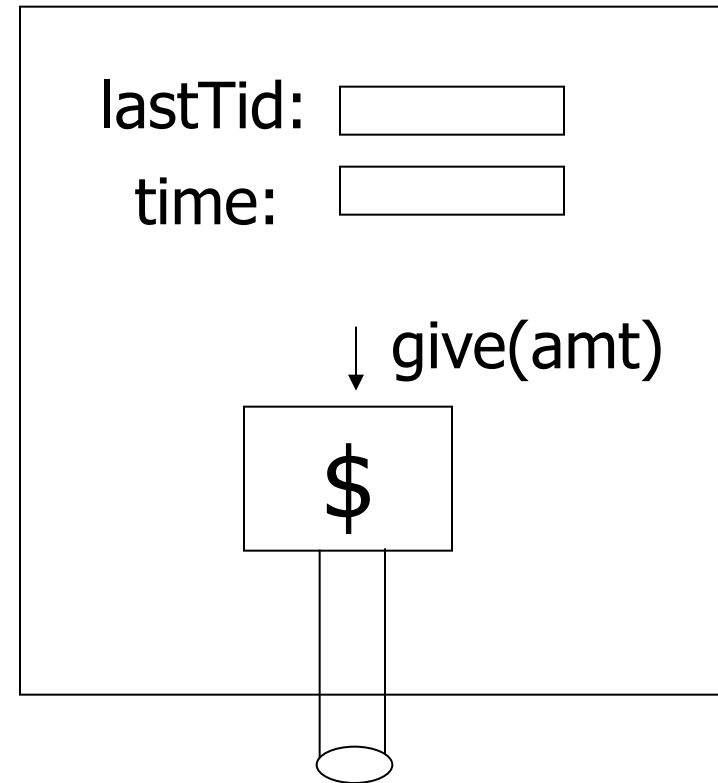
→

lastTid: ☐

time: ☐

↓ give(amt)

$

Notes 13 - Failure and Recovery   88

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Media failure (loss of non-volatile storage)

A: 16

Media failure (loss of non-volatile storage)

A: 16
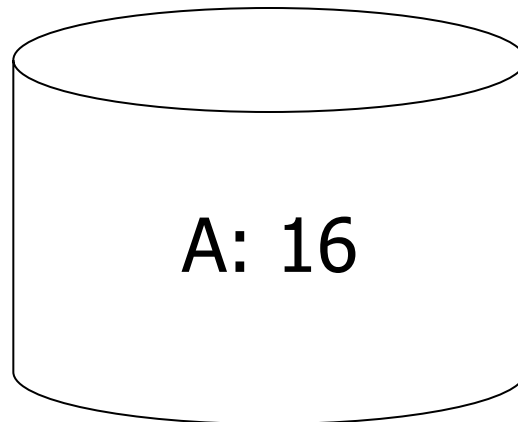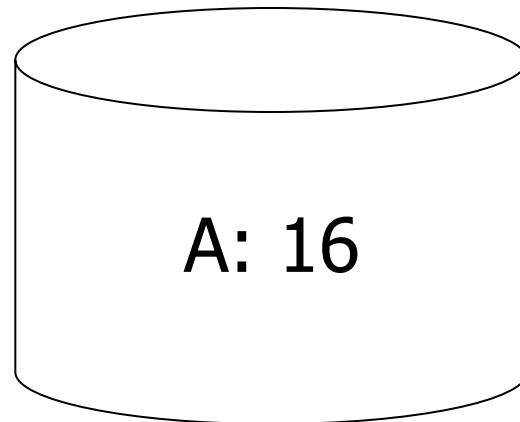
Solution: Make copies of data!

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example 1 Triple modular redundancy

- Keep 3 copies on separate disks
- Output(X) --> three outputs
- Input(X) --> three inputs + vote

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
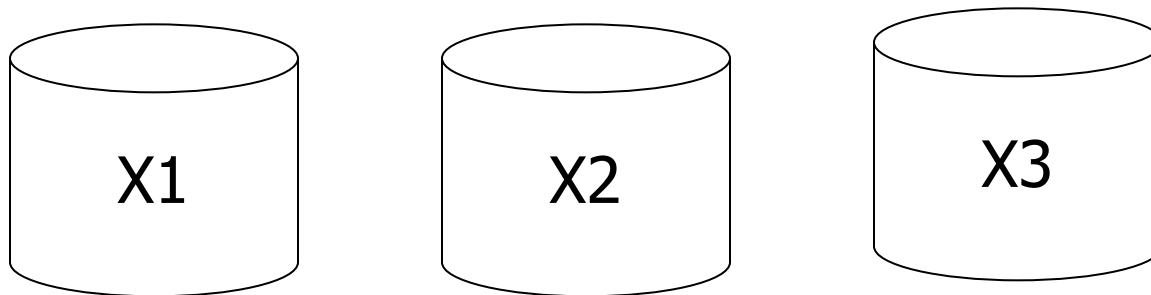
Example #2   Redundant writes, Single reads

- Keep N copies on separate disks
- Output(X) --> N outputs
- Input(X) --> Input one copy
  - if ok, done
  - else try another one

➠ Assumes bad data can be detected

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example #3: DB Dump + Log

backup database   log   active database
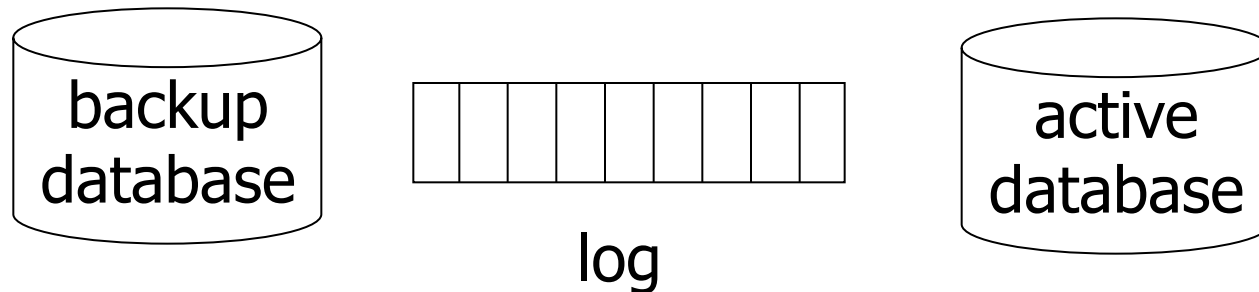
- If active database is lost,
    - restore active database from backup
    - bring up-to-date using redo entries in log

When can log be discarded?

COMPUTER
SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
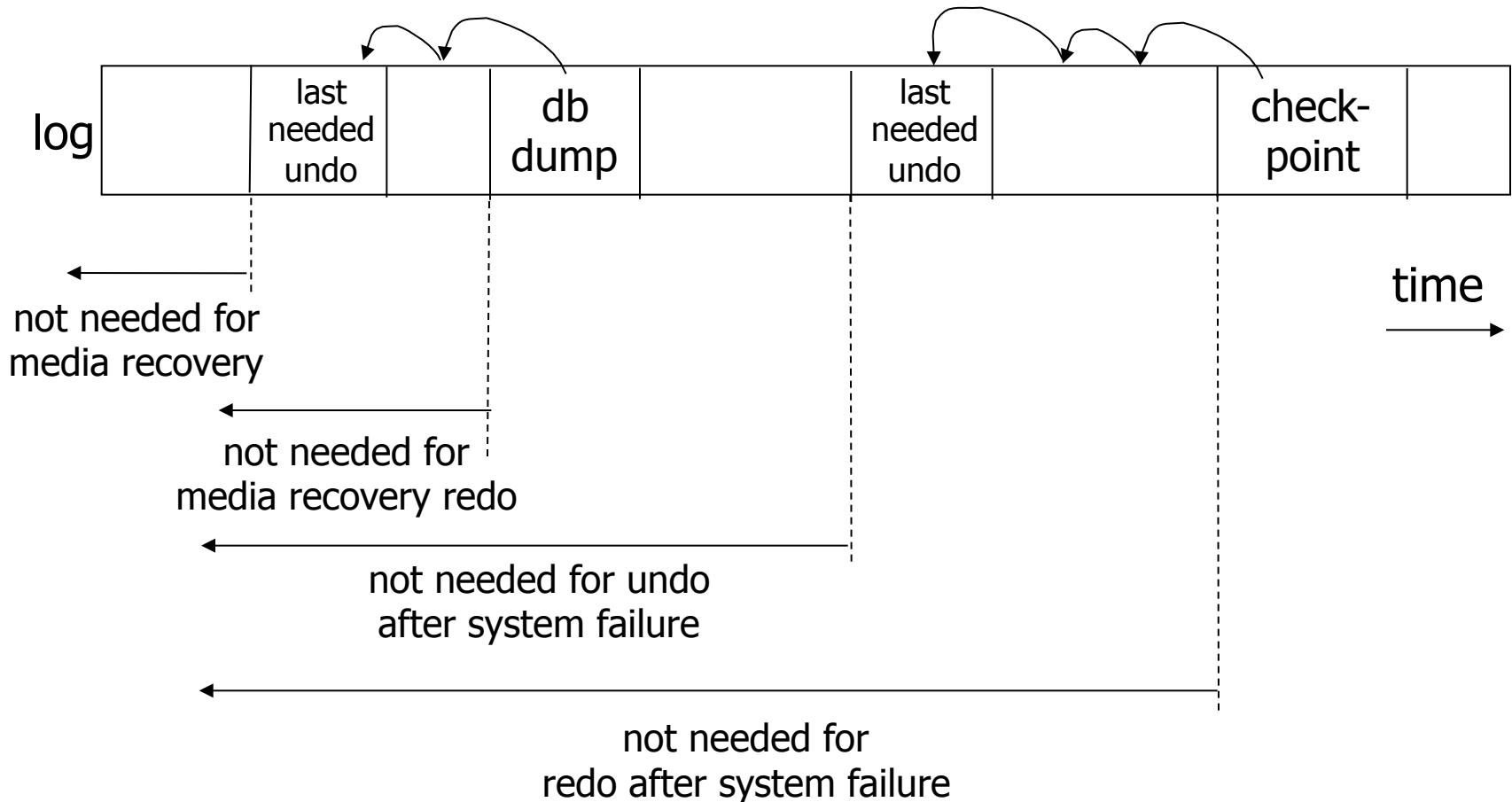
Practical Recovery with ARIES

- **ARIES**
  - **A**lgorithms for **R**ecovery and **I**solation **E**xploiting **S**emantics
- Implemented in, e.g.,
  - DB2
  - MSSQL

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Underlying Ideas

- Keep track of state of pages by relating them to entries in the log

- **WAL**

- Recovery in **three phases**
  - Analysis, Redo, Undo

- Log entries to track state of Undo for repeated failures

- **Redo**: page-oriented -> efficient

- **Undo**: logical -> permits higher level of concurrency

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Log Entry Structure

- **LSN**
  - **L**og **s**equence **n**umber
  - Order of entries in the log
  - Usually **log file id** and **offset** for direct access

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

- **LSN**
- **Entry type**
  - Update, compensation, commit, …
- **TID**
  - Transaction identifier
- **PrevLSN**
  - LSN of previous log record for same transaction
- **UndoNxtLSN**
  - Next undo operation for CLR (later!)
- **Undo/Redo data**
  - Data needed to undo/redo the update

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Page Header Additions

- **PageLSN**
  - **LSN** of the last update that modified the page
  - Used to know which changes have been applied to a page

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Forward Processing

- Normal operations when no ROLLBACK is required

  – WAL: write redo/undo log record for each action of a transaction

- Buffer manager has to ensure that

  – changes to pages are not persisted before the corresponding log record has been persisted

  – Transactions are not considered committed before all their log records have been flushed

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Dirty Page Table

- **PageLSN**
  - Entries **<PageID,RecLSN>**
  - Whenever a page is first fixed in the buffer pool with indention to modify
    - Insert **<PageId,RecLSN>** with **RecLSN** being the current end of the log
  - Flushing a page removes it from the Dirty page table

Dirty Page Table

- Used for checkpointing
- Used for recovery to figure out what to redo

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Transaction Table

- ## TransID
  - Identifier of the transaction

- ## State
  - Commit state

- ## LastLSN
  - LSN of the last update of the transaction

- ## UndoNxtLSN
  - If last log entry is a CLR then UndoNxtLSN from that record
  - Otherwise = LastLSN

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Transaction Table:
<1, U, -, ->

Dirty Page Table:

Page_LSN:
LSN of last modification to page
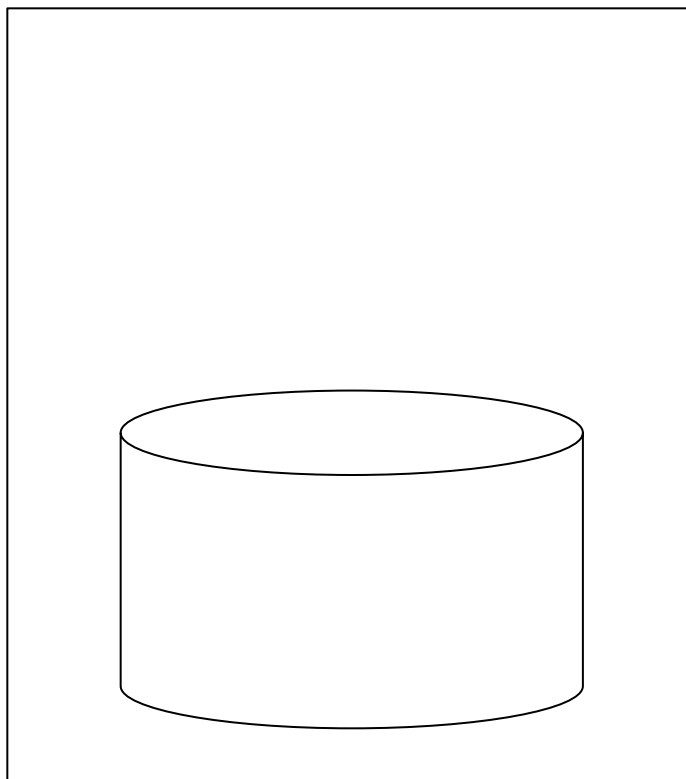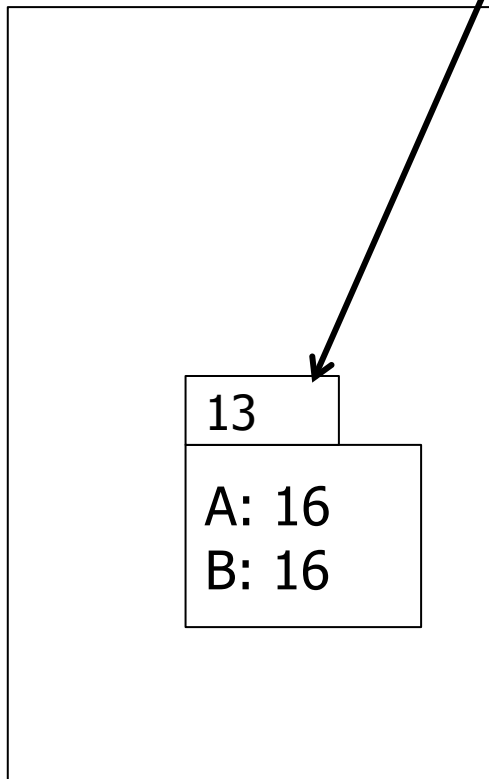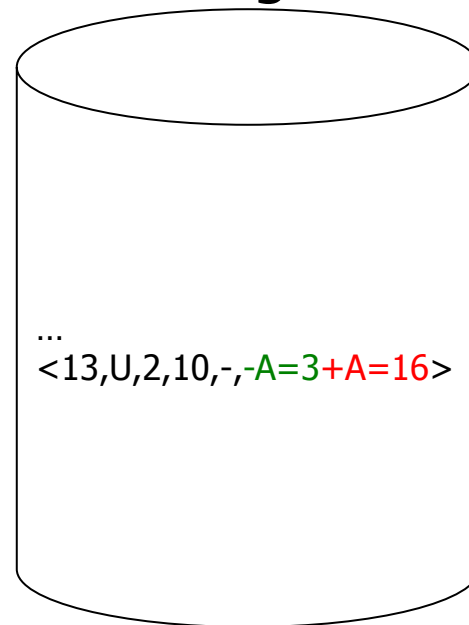
$T_1=r_1(A), A=A*2, w_1(A)$

buffer

disk

Persistent log

13

A: 16
B: 16

...
<13,U,2,10,-,-A=3+A=16>

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

$T_1 = r_1(A), A=A*2, w_1(A)$

buffer

disk

Persistent log

13

A: 16
B: 16

13

A: 16
B: 16

...
<13,U,2,10,-,-A=3+A=16>

Notes 13 - Failure and Recovery   105

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
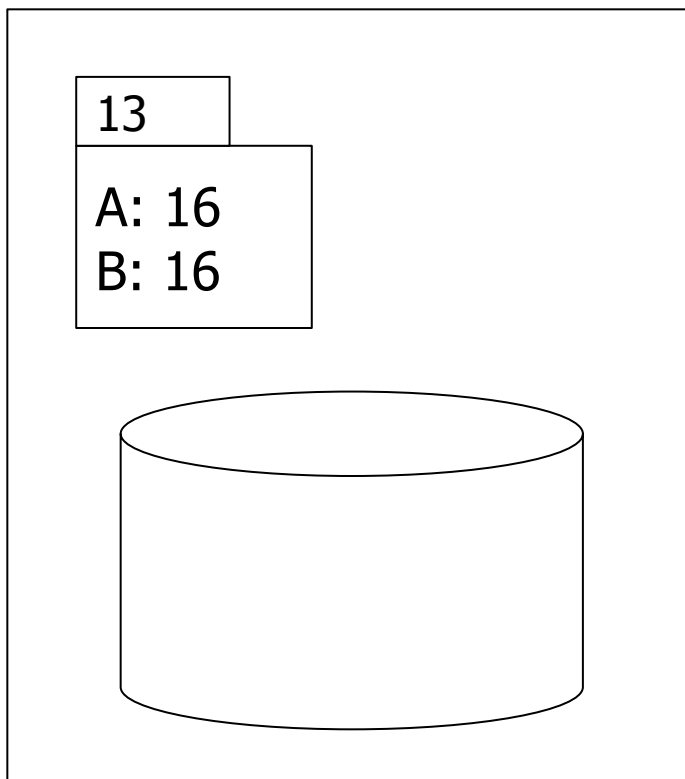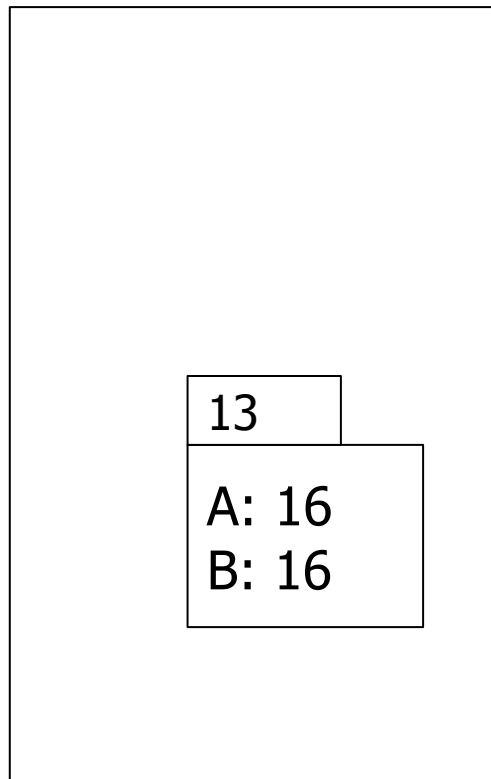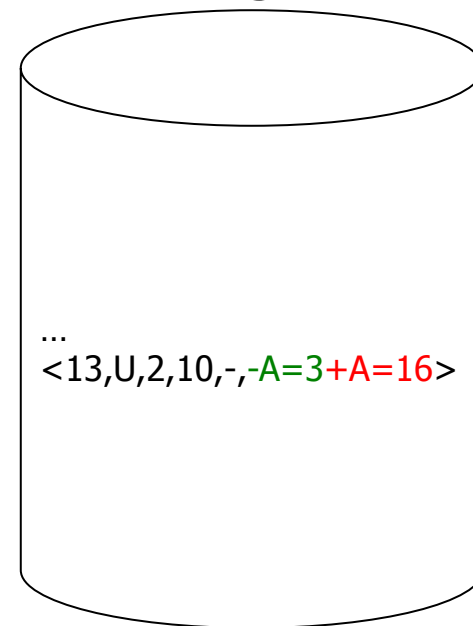
Transaction Table:
<1, U, -, ->

Dirty Page Table:
<100, 14>

Write log entry

$T_1 = r_1(A), A = A * 2, w_1(A)$

buffer

disk

Persistent log

13

A: 16
B: 16

<14,U,1,-,-,-A=16+A=32>

13

A: 16
B: 16

...
<13,U,2,10,-,-A=3+A=16>

Notes 13 - Failure and Recovery   106

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
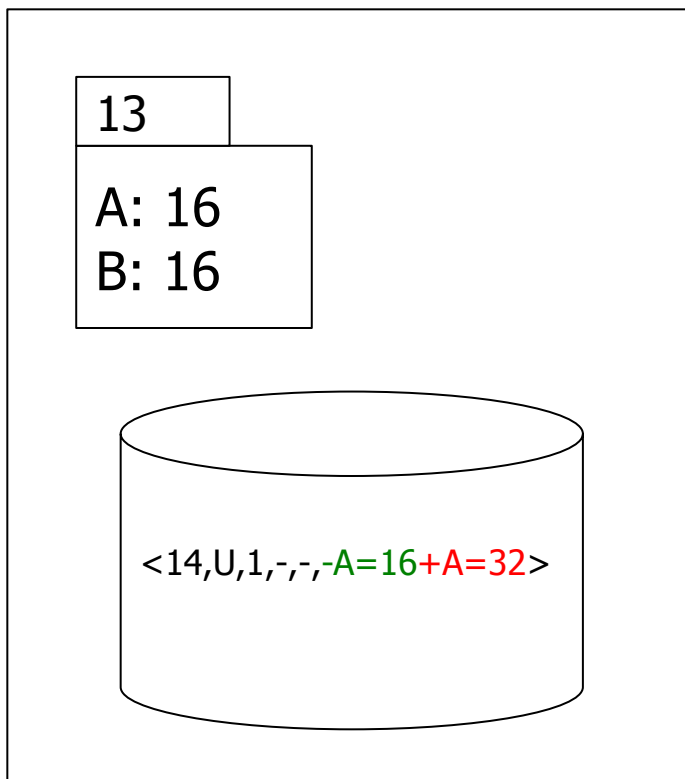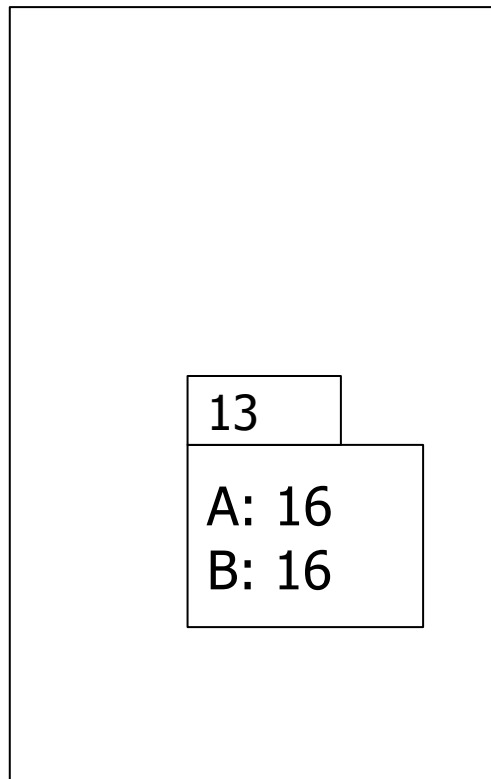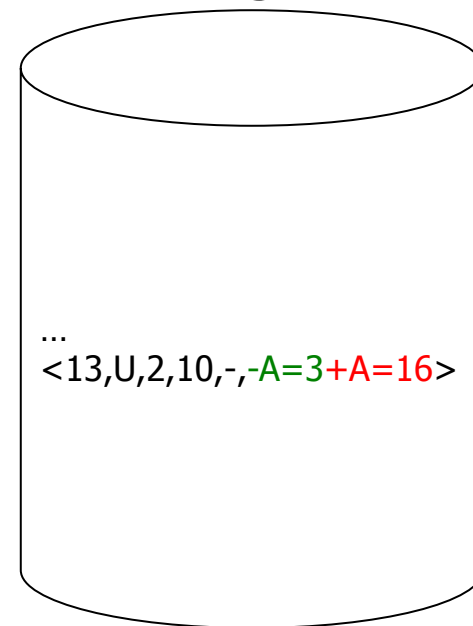
Transaction Table:

<1, U, 14, 14>

Dirty Page Table:

<100, 14>

Update page

$T_1=r_1(A),A=A*2,w_1(A)$

buffer

14

A: 32
B: 16

<14,U,1,-,-,-A=16+A=32>

disk

13

A: 16
B: 16

Persistent log

...
<13,U,2,10,-,-A=3+A=16>

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
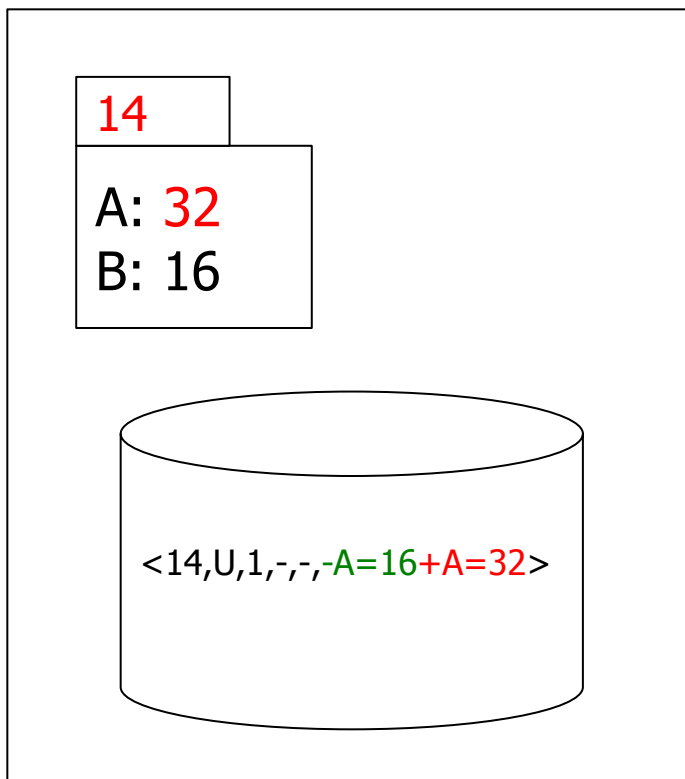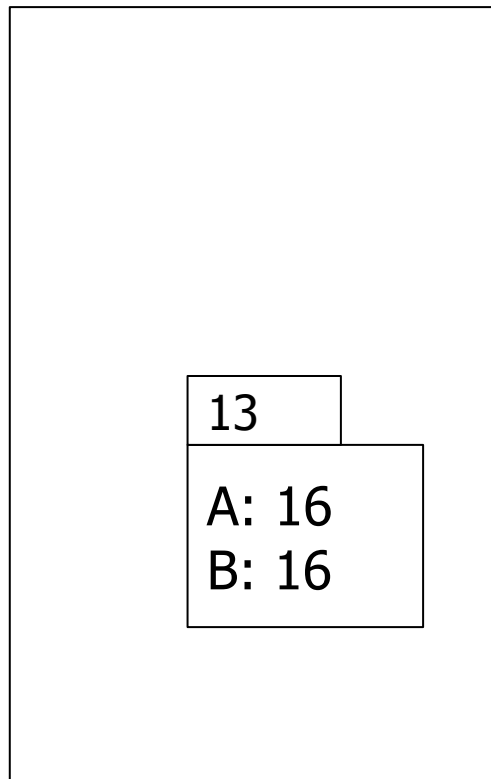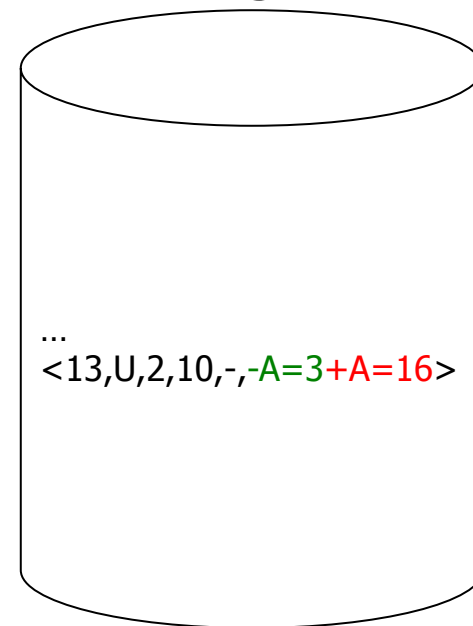
Transaction Table:
<1, U, 14, 14>

Dirty Page Table:
<100, 14>

Can wait with flushing page, but log has to be flushed first!

$T_1 = r_1(A), A = A*2, w_1(A)$

buffer

disk

Persistent log

14

A: 32
B: 16

<14,U,1,-,-,-A=16+A=32>

13

A: 16
B: 16

...
<13,U,2,10,-,-A=3+A=16>

2

1

COMPUTER
SCIENCE

Notes 13 - Failure and Recovery   108

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
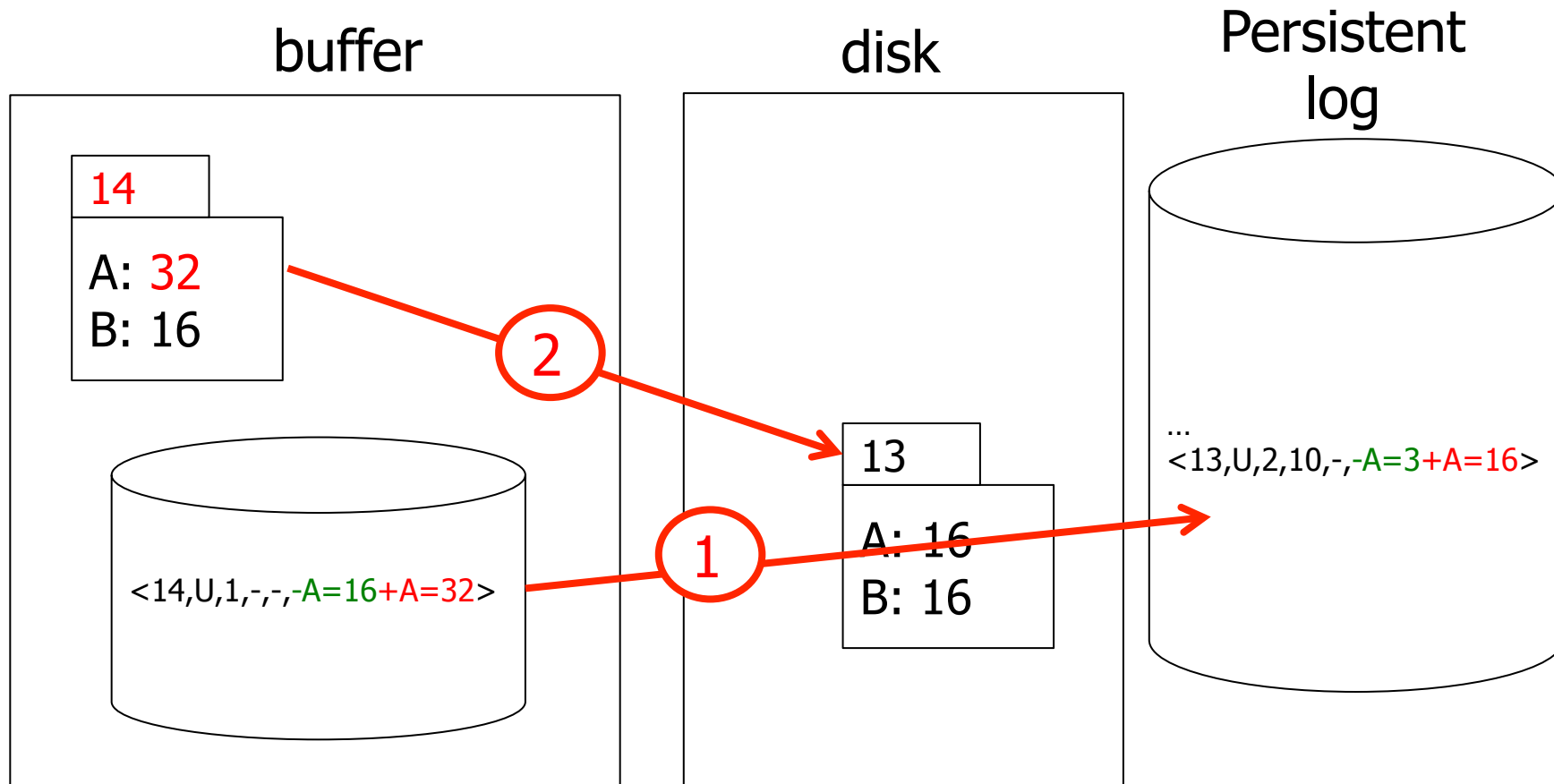
Undo during forward processing

- Transaction was rolled back
  - User aborted, aborted because of error, …
- Need to undo operations of transaction
- During Undo
  - Write log entries for every undo
  - **Compensation Log Records (CLR)**
  - Used to avoid repeated undo when failures occur

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Undo during forward processing

- Starting with the LastLSN of transaction from transaction table
  - Traverse log entries of transaction last to first using PrevLSN pointers
  - For each log entry use undo information to undo action
    - **<LSN, Type, TID, PrevLSN, -, Undo/Redo data>**
  - Before modifying data write an CLR that stores redo-information for the undo operation
    - **UndoNxtLSN** = **PrevLSN** of log entry we are undoing
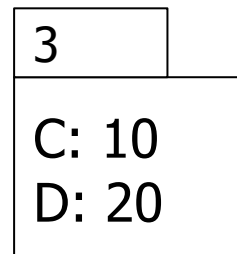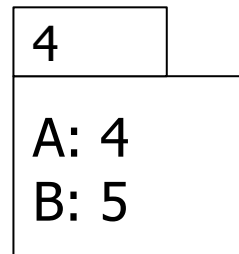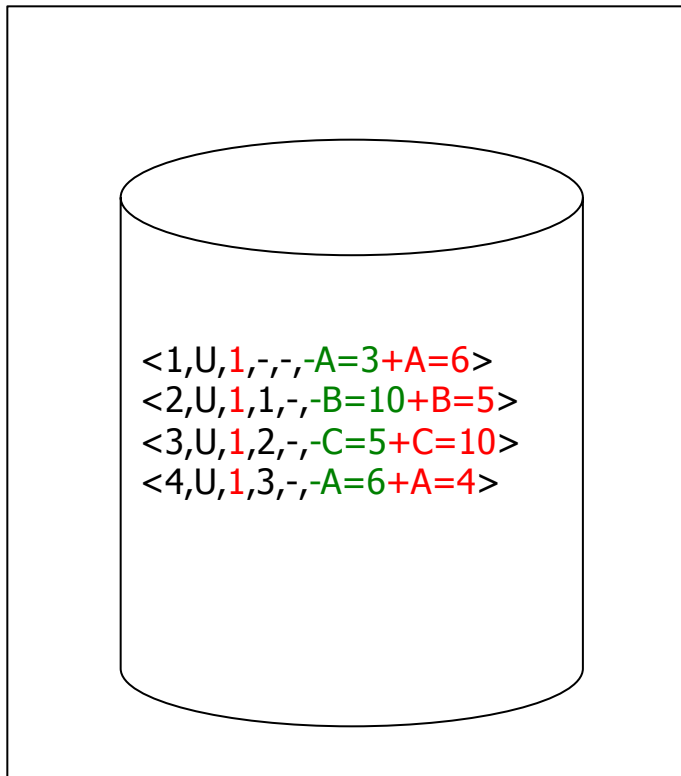   - **Redo data** = How to redo the undo

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Transaction Table:

<1, U, 4, 4>

Undo T$_1$

$T_1 = w_1(A), w_1(B), w_1(C), w_1(A), a_1$

buffer

<1,U,1,-,-,-A=3+A=6>
<2,U,1,1,-,-B=10+B=5>
<3,U,1,2,-,-C=5+C=10>
<4,U,1,3,-,-A=6+A=4>

4

A: 4
B: 5

3

C: 10
D: 20

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Transaction Table:

<1, U, 5, 3>

Undo T$_1$
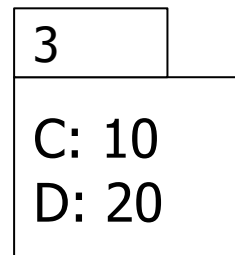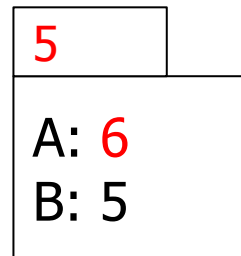
$T_1= w_1(A), w_1(B), w_1(C), w_1(A), a_1$

buffer


```
<1,U,1,-,-,-A=3+A=6>
<2,U,1,1,-,-B=10+B=5>
<3,U,1,2,-,-C=5+C=10>
<4,U,1,3,-,-A=6+A=4>
<5,CLR,1,-,3,+A=6>
```

5

A: 6
B: 5

3

C: 10
D: 20

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Transaction Table:

<1, U, 6, 2>

Undo T$_1$

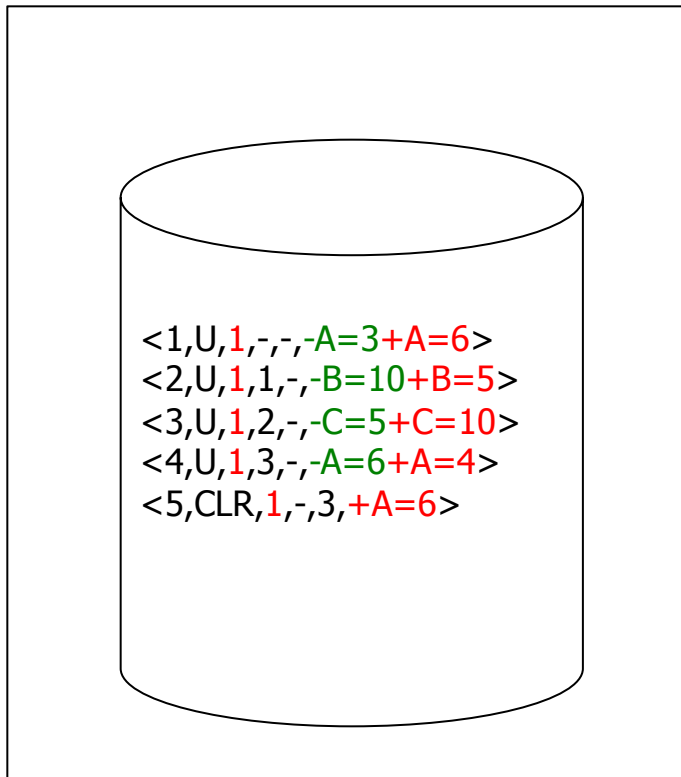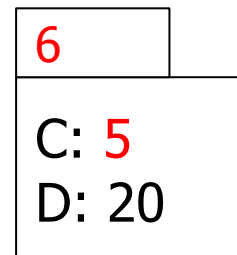$$T_1 = w_1(A), w_1(B), w_1(C), w_1(A), a_1$$

buffer

<1,U,1,-,-,-A=3+A=6>
<2,U,1,1,-,-B=10+B=5>
<3,U,1,2,-,-C=5+C=10>
<4,U,1,3,-,-A=6+A=4>
<5,CLR,1,-,3,+A=6>
<6,CLR,1,-,2,+C=5>

5

A: 6
B: 5

6

C: 5
D: 20

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Transaction Table:

<1, U, 7, 1>

Undo T$_1$

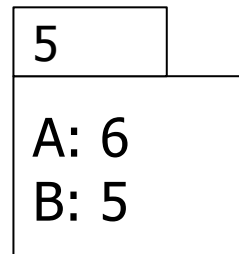$T_1 = w_1(A), w_1(B), w_1(C), w_1(A), a_1$

buffer

<1,U,1,-,-,-A=3+A=6>
<2,U,1,1,-,-B=10+B=5>
<3,U,1,2,-,-C=5+C=10>
<4,U,1,3,-,-A=6+A=4>
<5,CLR,1,-,3,+A=6>
<6,CLR,1,-,2,+C=5>
<7,CLR,1,-,1,+B=10>

7

A: 6
B: 10

6

C: 5
D: 20

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Undo T$_1$

$$T_1 = w_1(A), w_1(B), w_1(C), w_1(A), a_1$$

buffer

<1,U,1,-,-,-A=3+A=6>
<2,U,1,1,-,-B=10+B=5>
<3,U,1,2,-,-C=5+C=10>
<4,U,1,3,-,-A=6+A=4>
<5,CLR,1,-,3,+A=6>
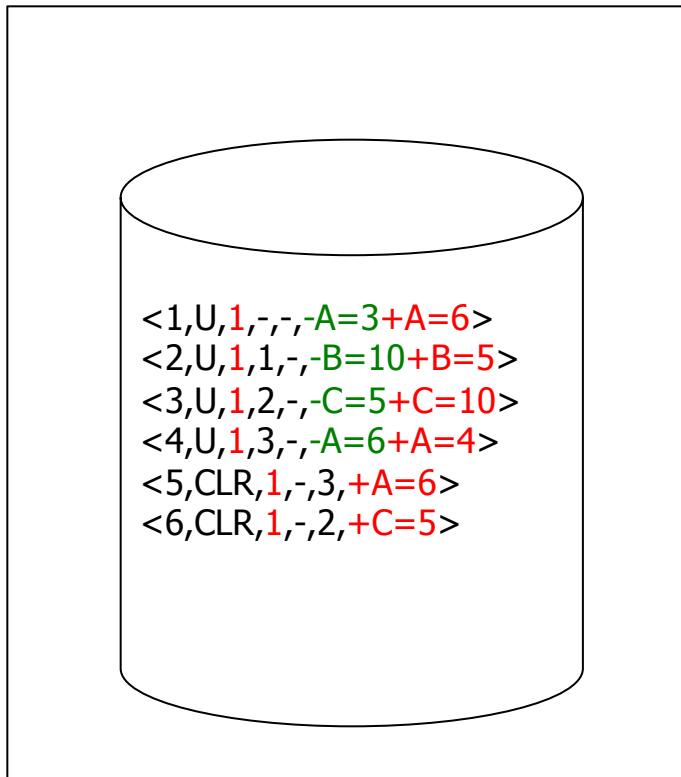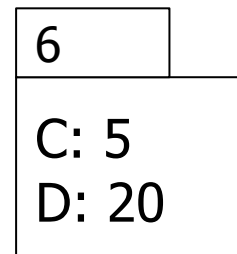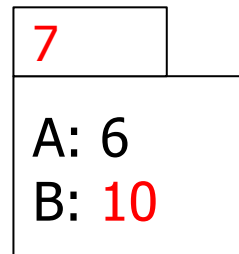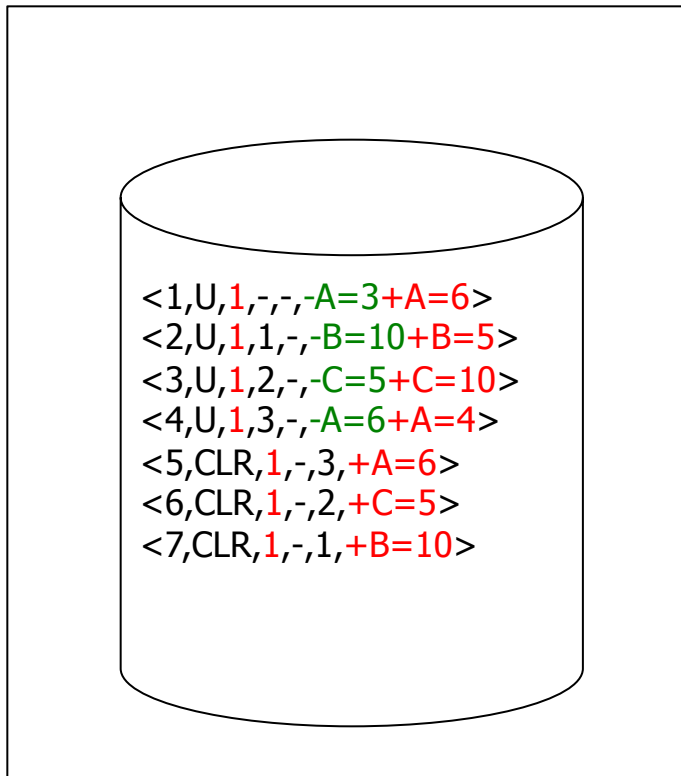<6,CLR,1,-,2,+C=5>
<7,CLR,1,-,1,+B=10>
<8,CLR,1,-,-,+A=3>

8

A: 3
B: 10

6

C: 5
D: 20

CS 525

COMPUTER SCIENCE

Notes 13 - Failure and Recovery   115

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
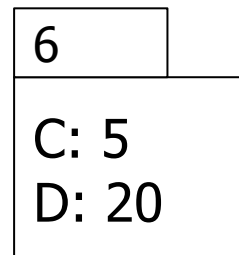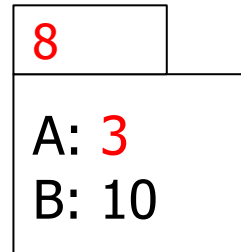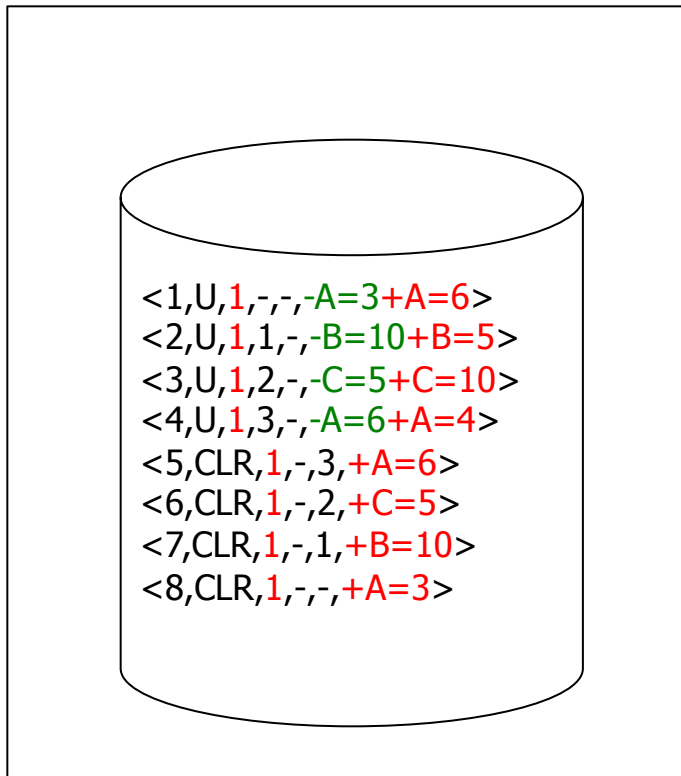
Fuzzy Checkpointing in ARIES

- Begin of checkpoint
  - Write **begin_cp** log entry
  - Write **end_cp** log entry with
    - Dirty page table
    - Transaction table

- **Master Record**
  - LSN of begin_cp log entry of last complete checkpoint

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Restart Recovery

1. Analysis Phase
2. Redo Phase
3. Undo Phase

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Analysis Phase

1) Determine LSN of last checkpoint using Master Record

2) Get Dirty Page Table and Transaction Table from checkpoint end record

3) RedoLSN = min(RecLSN) from Dirty Page Table or checkpoint LSN if no dirty page

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Analysis Phase

4) Scan log forward starting from RedoLSN

- Update log entry from transaction
  - If necessary: Add Page to Dirty Page Table
  - Add Transaction to Transaction Table or update LastLSN

- Transaction end entry
  - Remove transaction from Transaction Table

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Analysis Phase

- Result
  - Transaction Table
    - Transactions to be later undone
  - RedoLSN
    - Log entry to start Redo Phase
  - Dirty Page Table
    - Pages that may not have been written back to disk

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Redo Phase

- Start at RedoLSN scan log forward
- Unconditional Redo
  - Even redo actions of transactions that will be undone later
- Only redo once
  - Only redo operations that have not been reflected on disk (PageLSN)

Redo Phase

- For each update log entry
  - If affected page is not in Dirty Page Table or RecLSN > LSN
    - skip log entry
  - Fix page in buffer
    - If PageLSN >= LSN then operation already reflected on disk
      - Skip log entry
    - Otherwise apply update

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Redo Phase

- Result
  - State of DB before Failure

Undo Phase

- Scan log backwards from end using Transaction Table

  - Repeatedly take log entry with max LSN from all the current actions to be undone for each transaction

    - Write CLR

    - Update Transaction Table

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Undo Phase

- All unfinished transactions have been rolled back

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Idempotence?

- Redo
  - We are not logging during Redo so repeated Redo will result in the same state
- Undo
  - If we see CLRs we do not undo this action again

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Avoiding Repeated Work

- Redo
  - If operation has been reflected on disk (PageLSN) we do not need to redo it again

- Undo
  - If we see CLRs we do not undo this action again

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

$T_1 = w_1(A), \quad w_1(B), w_1(C), w_1(A), c_1$

$T_2 = \quad w_1(X), \quad\quad\quad\quad\quad\quad r(A), w(A)$

Log

<1,begin(T_1), ->
<2,begin(T_2), ->
<3,write(A,T_1),1>
<4,write(X,T_2),2>
<5,write(B,T_1),3>
<6,write(C,T_1),5>
<7,write(A,T_1),6>
<8,commit(T_1),7>
<9,write(A,T_2),4>

Notes 13 - Failure and Recovery   128

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
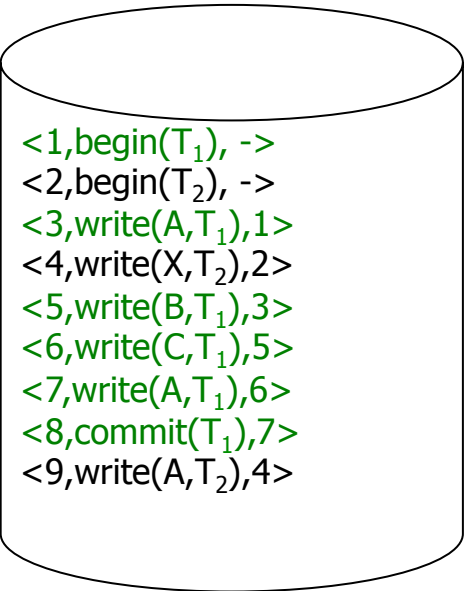
$T_1 = w_1(A), \quad w_1(B), w_1(C), w_1(A), c_1$

$T_2 = \quad w_1(X), \quad\quad\quad\quad r(A), w(A)$

Analysis Phase:

- start at log entry 1
- add T_1 to transaction table (rec. 1)
- add T_2 to transaction table (rec. 2)
- add A to dirty page table (RecLSN 3)
- add X to dirty page table (RecLSN 4)
- add B to dirty page table (RecLSN 5)
- add C to dirtypage table (RecLSN 6)
- remove T1 from Transaction Table (rec. 8)

Log

<1,begin(T_1), ->
<2,begin(T_2), ->
<3,write(A,T_1),1>
<4,write(X,T_2),2>
<5,write(B,T_1),3>
<6,write(C,T_1),5>
<7,write(A,T_1),6>
<8,commit(T_1),7>
<9,write(A,T_2),4>

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

$T_1 = w_1(A), \qquad w_1(B), w_1(C), w_1(A), c_1$

$T_2 = \qquad w_1(X), \qquad\qquad\qquad r(A), w(A)$

Analysis Phase Result:
  - Transaction Table:
    $\langle T_2, 9 \rangle$
  - Dirty Page Table:
    $\langle A, 3 \rangle$, $\langle B, 5 \rangle$, $\langle C, 6 \rangle$, $\langle X, 4 \rangle$
  - RedoLSN = min(3,5,6,4) = 3

Log

<1,begin(T_1), ->
<2,begin(T_2), ->
<3,write(A,T_1),1>
<4,write(X,T_2),2>
<5,write(B,T_1),3>
<6,write(C,T_1),5>
<7,write(A,T_1),6>
<8,commit(T_1),7>
<9,write(A,T_2),4>

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

$T_1 = w_1(A), \quad w_1(B), w_1(C), w_1(A), c_1$

$T_2 = \quad w_1(X), \quad\quad\quad\quad r(A), w(A)$

Redo Phase (RedoLSN 3):
- Read A if PageLSN < 3 apply write
- Read X if PageLSN < 4 apply write
- Read B if PageLSN < 5 apply write
- Read C if PageLSN < 6 apply write
- Read A if PageLSN < 7 apply write
- Read A if PageLSN < 9 apply write

Log

$<1, \text{begin}(T_1), ->$
$<2, \text{begin}(T_2), ->$
$<3, \text{write}(A, T_1), 1>$
$<4, \text{write}(X, T_2), 2>$
$<5, \text{write}(B, T_1), 3>$
$<6, \text{write}(C, T_1), 5>$
$<7, \text{write}(A, T_1), 6>$
$<8, \text{commit}(T_1), 7>$
$<9, \text{write}(A, T_2), 4>$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
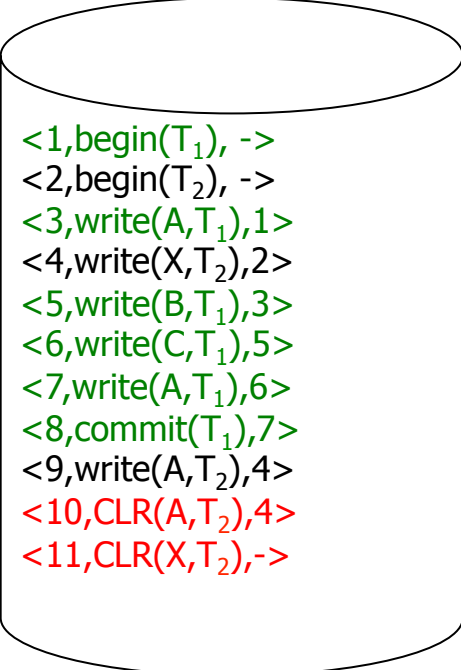
$T_1 = w_1(A),\quad w_1(B),\ w_1(C),\ w_1(A),\ c_1$

$T_2 =\quad w_1(X),\quad\quad\quad\quad r(A),\ w(A)$

Undo Phase (T_2):
- Undo entry 9
  - write CLR with UndoNxtLSN = 4
  - modify page A
- Undo entry 4
  - write CLR with UndoNxtLSN = 2
  - modify page X
- Done

Log

<1,begin(T_1), ->
<2,begin(T_2), ->
<3,write(A,T_1),1>
<4,write(X,T_2),2>
<5,write(B,T_1),3>
<6,write(C,T_1),5>
<7,write(A,T_1),6>
<8,commit(T_1),7>
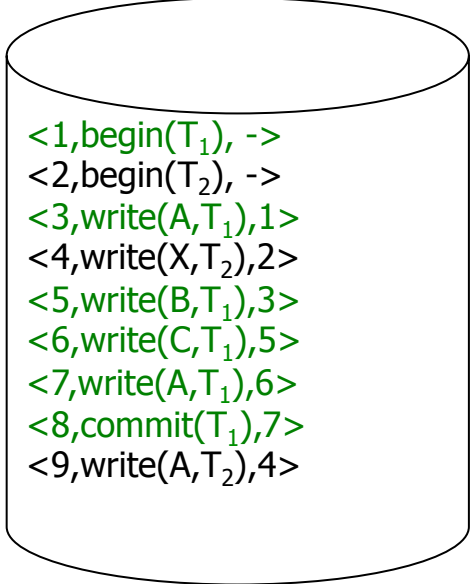<9,write(A,T_2),4>
<10,CLR(A,T_2),4>
<11,CLR(X,T_2),->

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

ARIES take away messages

- Provide good performance by
  - Not requiring complete checkpoints
  - Linking of log records
  - Not restricting buffer operations (no-force/steal is ok)

- Logical Undo and Physical (Physiological) Redo

- Idempotent Redo and Undo
  - Avoid undoing the same operation twice

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Media Recovery

- What if disks where log or DB is stored failes
  - ->keep backups of log + DB state

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Log Backup

- Split log into several files
- Is append only, backup of old files cannot interfere with current log operations

Backup DB state

- Copy current DB state directly from disk
- May be inconsistent
- ->Use log to know which pages are up-to-date and redo operations not yet reflected

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Summary

- Consistency of data
- One source of problems: failures

    - Logging

    - Redundancy

- Another source of problems:
        Data Sharing….. next

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

CS 525: Advanced Database Organization
14: **Concurrency Control**

Boris Glavic

Slides: adapted from a course taught by

Hector Garcia-Molina, Stanford InfoLab

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Chapter 18 [18] Concurrency Control
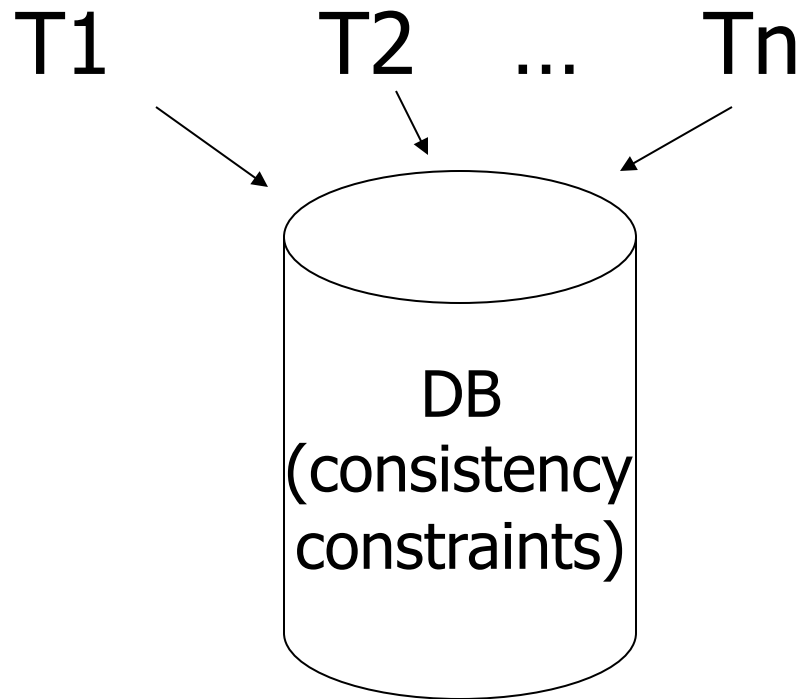
T1   T2   ...   Tn

DB
(consistency
constraints)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Example:

T1: Read(A)   T2: Read(A)
     A ← A+100   A ← A×2
     Write(A)   Write(A)
     Read(B)   Read(B)
     B ← B+100   B ← B×2
     Write(B)   Write(B)

Constraint: A=B

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule A

T1	T2
Read(A); A ← A+100	
Write(A);	
Read(B); B ← B+100;	
Write(B);	
	Read(A);A ← A×2;
	Write(A);
	Read(B);B ← B×2;
	Write(B);

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule A

	A	B
	25	25

T1	T2	A	B
Read(A); A ← A+100			
Write(A);		125	
Read(B); B ← B+100;			
Write(B);			125
	Read(A);A ← A×2;		
	Write(A);	250	
	Read(B);B ← B×2;		
	Write(B);		250
		250	250

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule B

T1	T2
	Read(A);A ← A×2;
	Write(A);
	Read(B);B ← B×2;
	Write(B);
Read(A); A ← A+100	
Write(A);	
Read(B); B ← B+100;	
Write(B);	

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule B

		A	B
		25	25

T1	T2
	Read(A);A ← A×2;
	Write(A);
	Read(B);B ← B×2;
	Write(B);
Read(A); A ← A+100	
Write(A);	
Read(B); B ← B+100;	
Write(B);	

A	B
25	25
50	
	50
150	
	150
150	150

Notes 14 - Concurrency Control

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Schedule C

T1	T2
Read(A); A ← A+100	
Write(A);	
	Read(A);A ← A×2;
	Write(A);
Read(B); B ← B+100;	
Write(B);	
	Read(B);B ← B×2;
	Write(B);

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule C

		A	B
		25	25

T1	T2		
Read(A); A ← A+100			
Write(A);		125	
	Read(A);A ← A×2;		
	Write(A);	250	
Read(B); B ← B+100;			
Write(B);			125
	Read(B);B ← B×2;		
	Write(B);		250
		250	250

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule D

T1	T2
Read(A); A ← A+100	
Write(A);	
	Read(A);A ← A×2;
	Write(A);
	Read(B);B ← B×2;
	Write(B);
Read(B); B ← B+100;	
Write(B);	

COMPUTER SCIENCE

Notes 14 - Concurrency Control

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule D

		A	B
		25	25

T1	T2
Read(A); A ← A+100	
Write(A);	
	Read(A); A ← A×2;
	Write(A);
	Read(B); B ← B×2;
	Write(B);
Read(B); B ← B+100;	
Write(B);	

A: 25, 125, 250, 250
B: 25, 50, 150, 150

Schedule E

T1	T2'
Read(A); A ← A+100	
Write(A);	
	Read(A);A ← A×1;
	Write(A);
	Read(B);B ← B×1;
	Write(B);
Read(B); B ← B+100;	
Write(B);	

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule E

Same as Schedule D but with new T2'

		A	B
		25	25

T1	T2'
Read(A); A ← A+100	
Write(A);	
	Read(A);A ← A×1;
	Write(A);
	Read(B);B ← B×1;
	Write(B);
Read(B); B ← B+100;	
Write(B);	

A	B
25	25
125	
125	
	25
	125
125	125

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Serial Schedules

- As long as we do not execute transactions in parallel and each transaction does not violate the constraints we are good
  - All schedules with no interleaving of transaction operations are called **serial** schedules

COMPUTER SCIENCE

Notes 14 - Concurrency Control

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Definition: Serial Schedule

- No transactions are interleaved
  - There exists no two operations from transactions Ti and Tj so that both operations are executed before either transaction commits

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

$$T_1 = r_1(A), w_1(A), r_1(B), w_1(B), c_1$$

$$T_2 = r_2(A), w_2(A), r_2(B), w_2(B), c_2$$

Serial Schedule

$$S_1 = r_2(A), w_2(A), r_2(B), w_2(B), c_2, r_1(A), w_1(A), r_1(B), w_1(B), c_1$$

Nonserial Schedule

$$S_2 = r_2(A), w_2(A), r_1(A), w_1(A), r_2(B), w_2(B), c_2, r_1(B), w_1(B), c_1$$

Compare Classes

$$S \subset ST \subset CL \subset RC \subset ALL$$

- Abbreviations
  - S = Serial
  - ST = Strict
  - CL = Cascadeless
  - RC = Recoverable
  - ALL = all possible schedules

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

All schedules (**ALL**)

Recoverable (**RC**)

Cascadeless (**CL**)

Strict (**ST**)

Serial (**S**)

Why not serial schedules?

- No concurrency! ☹

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

- Want schedules that are "good", regardless of
  - initial state and
  - transaction semantics
- Only look at order of read and writes

Example:

$Sc = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$

IIT College of
Science and Letters
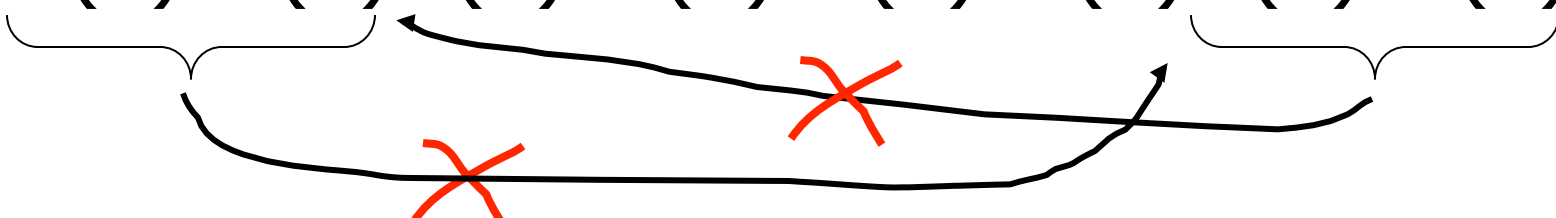ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Outline

- Since serial schedules have good properties we would like our schedules to behave like (be **equivalent** to) serial schedules

  1. Need to define equivalence based solely on order of operations

  2. Need to define class of schedules which is equivalent to serial schedule

  3. Need to design scheduler that guarantees that we only get these good schedules

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example:

$Sc = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$

$Sc' = r_1(A)w_1(A)\ r_1(B)w_1(B)r_2(A)w_2(A)r_2(B)w_2(B)$

T_1 T_2

However, for Sd:

$Sd = r_1(A)w_1(A)r_2(A)w_2(A) \; r_2(B)w_2(B)r_1(B)w_1(B)$

- as a matter of fact,

  T_2 must precede T_1

  in any equivalent schedule,

  i.e., $T_2 \to T_1$

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
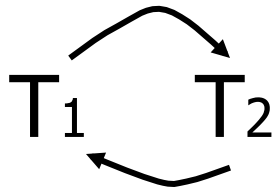
- $T_2 \rightarrow T_1$
- Also, $T_1 \rightarrow T_2$

$T_1 \quad T_2$ $\Rightarrow$ Sd cannot be rearranged
                        into a serial schedule

$\Rightarrow$ Sd is not "equivalent" to
                  any serial schedule

$\Rightarrow$ Sd is "bad"

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Returning to Sc

$$Sc=r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$$

$$T_1 \rightarrow T_2 \qquad\qquad T_1 \rightarrow T_2$$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Returning to Sc

$$Sc = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$$

$$T_1 \rightarrow T_2 \qquad\qquad T_1 \rightarrow T_2$$

☛ no cycles $\Rightarrow$ Sc is "equivalent" to a

serial schedule

(in this case T_1, T_2)

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Concepts

Transaction: sequence of $r_i(x)$, $w_i(x)$ actions

Conflicting actions:

$$r_1(A) \qquad w_2(A) \qquad w_1(A)$$
$$w_2(A) \qquad r_1(A) \qquad w_2(A)$$

Schedule: represents chronological order
in which actions are executed

Serial schedule: no interleaving of actions
or transactions

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

What about concurrent actions?

Ti issues   System   Input(X)   t ← x
read(x,t)   issues   completes
           input(x)

                                                time

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

What about concurrent actions?

Ti issues read(x,t)   System issues input(x)   Input(X) completes   t ← x

                                                                                 time

T2 issues write(B,S)   input(B) completes   System issues output(B)   output(B) completes

System issues input(B)   B ← S

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

So net effect is either

- $S = \ldots r_1(x) \ldots w_2(b) \ldots$ or
- $S = \ldots w_2(B) \ldots r_1(x) \ldots$
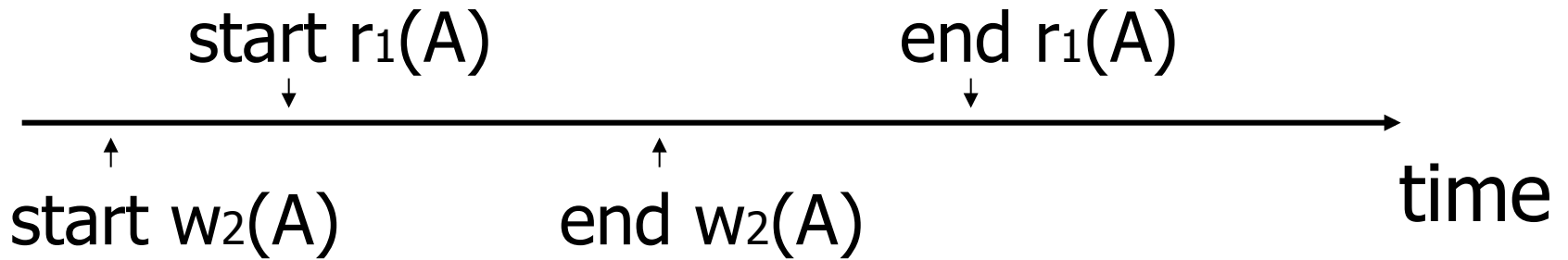
Notes 14 - Concurrency Control

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

What about conflicting, concurrent actions on same object?

start $r_1(A)$ end $r_1(A)$

————————————————————————————————→

start $w_2(A)$ end $w_2(A)$ time

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

What about conflicting, concurrent actions on same object?

start $r_1(A)$ end $r_1(A)$

start $w_2(A)$ end $w_2(A)$ time

- Assume equivalent to either $r_1(A)$ $w_2(A)$

  or $w_2(A)$ $r_1(A)$

- $\Rightarrow$ low level synchronization mechanism

- Assumption called "atomic actions"

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

- Since serial schedules have good properties we would like our schedules to behave like (be **equivalent** to) serial schedules

  1. Need to define equivalence based solely on order of operations

  2. Need to define class of schedules which is equivalent to serial schedule

  3. Need to design scheduler that guarantees that we only get these good schedules

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Conflict Equivalence

- Define equivalence based on the order of conflicting actions

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Definition

S$_1$, S$_2$ are <u>conflict equivalent</u> schedules
   if S$_1$ can be transformed into S$_2$ by a
   series of swaps on non-conflicting
   actions.

<u>Alternatively:</u>

If the order of conflicting actions in S$_1$
   and S$_2$ is the same

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

- Since serial schedules have good properties we would like our schedules to behave like (be **equivalent** to) serial schedules

  1. Need to define equivalence based solely on order of operations

  2. Need to define class of schedules which is equivalent to serial schedule

  3. Need to design scheduler that guarantees that we only get these good schedules

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Definition

A schedule is <u>conflict serializable</u> (**CSR**) if it is conflict equivalent to some serial schedule.

How to check?

- Compare orders of all conflicting operations

- Can be simplified because there is some redundant information here, e.g.,

$$S_1 = w_2(A), w_2(B), r_1(A), w_1(B)$$

- W2(A) conflicts with R1(A)
- W2(B) conflicts with W1(B)
- Both imply that T2 has to be executed before T1 in any equivalent serial schedule

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Conflict graph P(S) (S is schedule)

Nodes: transactions in S

Arcs: $T_i \rightarrow T_j$ whenever

        - $p_i(A)$, $q_j(A)$ are actions in S

        - $p_i(A) <_S q_j(A)$
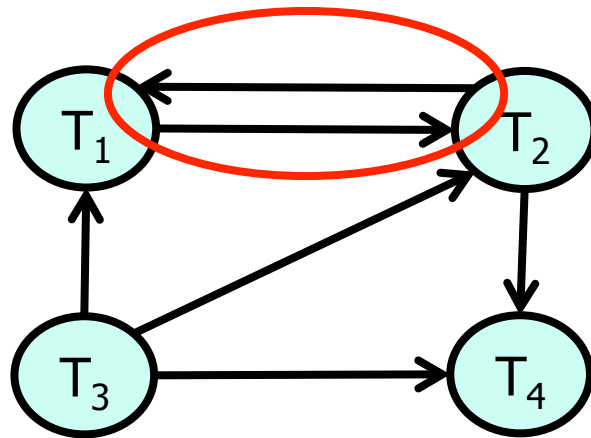
        - at least one of p_i, q_j is a write
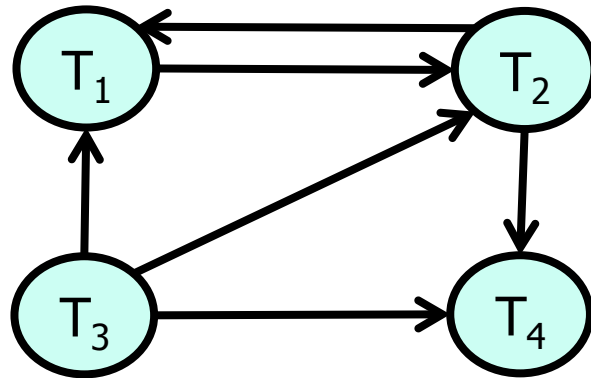
COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Exercise:

- What is P(S) for
  S = $w_3(A)$ $w_2(C)$ $r_1(A)$ $w_1(B)$ $r_1(C)$ $w_2(A)$ $r_4(A)$ $w_4(D)$

- Is S serializable?

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Exercise:

- What is P(S) for
S = w3(A) w2(C) r1(A) w1(B) r1(C) w2(A) r4(A) w4(D)

- Is S serializable?

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Exercise:

- What is P(S) for
  S = $w_3(A)$ $w_2(C)$ $r_1(A)$ $w_1(B)$ $r_1(C)$ $w_2(A)$ $r_4(A)$ $w_4(D)$

- Is S serializable?

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Another Exercise:

- What is P(S) for
  $S = w_1(A) \ r_2(A) \ r_3(A) \ w_4(A)$?

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Another Exercise:

- What is P(S) for
S = $w_1(A)$ $r_2(A)$ $r_3(A)$ $w_4(A)$?

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Lemma

S_1, S_2 conflict equivalent $\Rightarrow P(S_1) = P(S_2)$

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
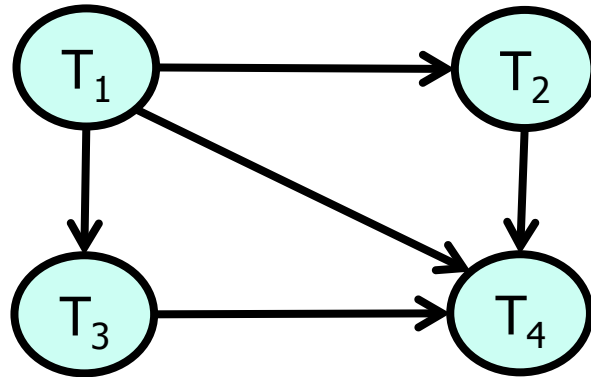
Lemma

S$_1$, S$_2$ conflict equivalent $\Rightarrow$ P(S$_1$)=P(S$_2$)

Proof: (a $\rightarrow$ b same as ¬b $\rightarrow$ ¬a)

Assume P(S$_1$) $\neq$ P(S$_2$)

$\Rightarrow$ ∃ T$_i$: T$_i$ $\rightarrow$ T$_j$ in S$_1$ and not in S$_2$

$\Rightarrow$ S$_1$ = ...p$_i$(A)... q$_j$(A)...   p$_i$, q$_j$

   S$_2$ = ...q$_j$(A)...p$_i$(A)...   conflict

$\Rightarrow$ S$_1$, S$_2$ not conflict equivalent

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Note: $P(S_1)=P(S_2) \not\Rightarrow S_1, S_2$ conflict equivalent

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Note: $P(S_1)=P(S_2) \not\Rightarrow S_1, S_2$ conflict equivalent

Counter example:

$S_1 = w_1(A)\ r_2(A) \qquad w_2(B)\ r_1(B)$

$S_2 = r_2(A)\ w_1(A) \qquad r_1(B)\ w_2(B)$

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Theorem

$P(S_1)$ acyclic $\Longleftrightarrow$ S_1 conflict serializable

($\Longleftarrow$) Assume S_1 is conflict serializable

$\Rightarrow \exists S_s$: S_s, S_1 conflict equivalent

$\Rightarrow P(S_s) = P(S_1)$

$\Rightarrow P(S_1)$ acyclic since $P(S_s)$ is acyclic

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Theorem

P(S_1) acyclic $\Longleftrightarrow$ S_1 conflict serializable

($\Rightarrow$) Assume P(S_1) is acyclic

Transform S_1 as follows:

(1) Take T_1 to be transaction with no incident arcs

(2) Move all T_1 actions to the front

$$S_1 = ……. \; q_j(A)…….p_1(A)…..$$

(3) we now have $S_1 = <\,T_1 \text{ actions }><… \text{ rest } …>$

(4) repeat above steps to serialize rest!

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
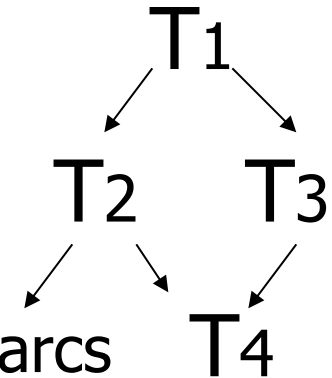
What's the damage?

- Classification of "bad" things that can happen in "bad" schedules
  - Dirty reads
  - Non-repeatable reads
  - Phantom reads (later)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Dirty Read

- A transaction T_1 read a value that has been updated by an uncommitted transaction T_2

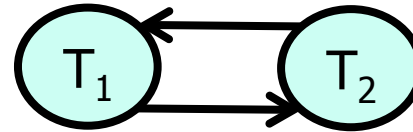- If T_2 aborts then the value read by T_1 is invalid
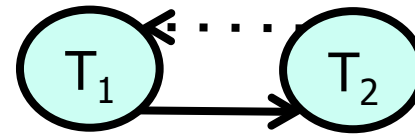
IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Dirty Read

T_1	T_2	A=50
Read(A), A += 100		T_1: A = 150
Write(A);		A = 150
	Read(A), A +=200	T_2: A = 350
Abort		
	Write(A);	

$S_1 = r_1(A), w_1(A), r_2(A), a_1, w_2(A)$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Non-repeatable Read

- A transaction T_1 reads items; some before and some after an update of these item by a transaction T_2

- Problem

  - Repeated reads of the same item see different values

  - Some values are modified and some are not

Inconsistent Read

T_1	T_2	A = 100
Read(A)		
	Read(A), A /= 2	A = 50
	Write(A)	
	Commit	
Read(A)		A = 50
Commit		

$S_1 = r_1(A), r_2(A), w_2(A), c_2, r_1(A), c_1$

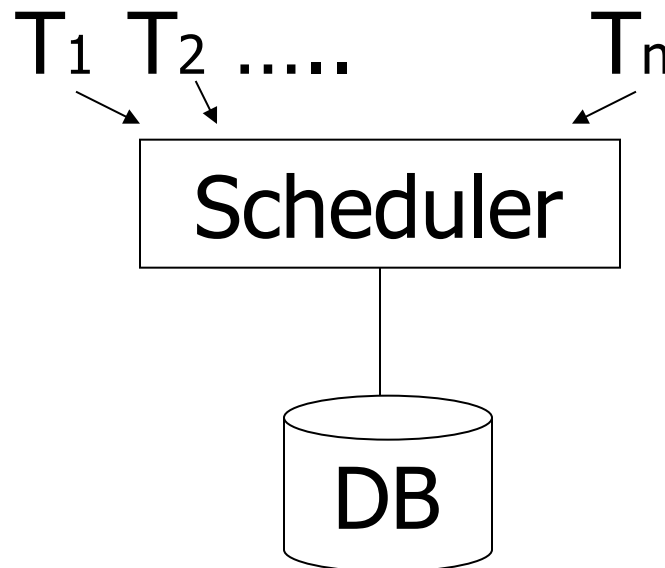IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

How to enforce serializable schedules?

Option 1:   run system, recording P(S);
at end of day, check for P(S)
cycles and declare if execution
was good

COMPUTER
SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

How to enforce serializable schedules?

Option 1: run system, recording P(S); at end of day, check for P(S) cycles and declare if execution was good

This is called **optimistic concurrency control**

COMPUTER SCIENCE

Notes 14 - Concurrency Control

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

How to enforce serializable schedules?

Option 2: prevent P(S) cycles from occurring

$$T_1 \; T_2 \; ..... \qquad\qquad T_n$$

```
        Scheduler

          DB
```

COMPUTER
SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How to enforce serializable schedules?

Option 2: prevent P(S) cycles from occurring

This is called **pessimistic concurrency control**

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

A locking protocol

Two new actions:

    lock (exclusive): $l_i (A)$

    unlock: $u_i (A)$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
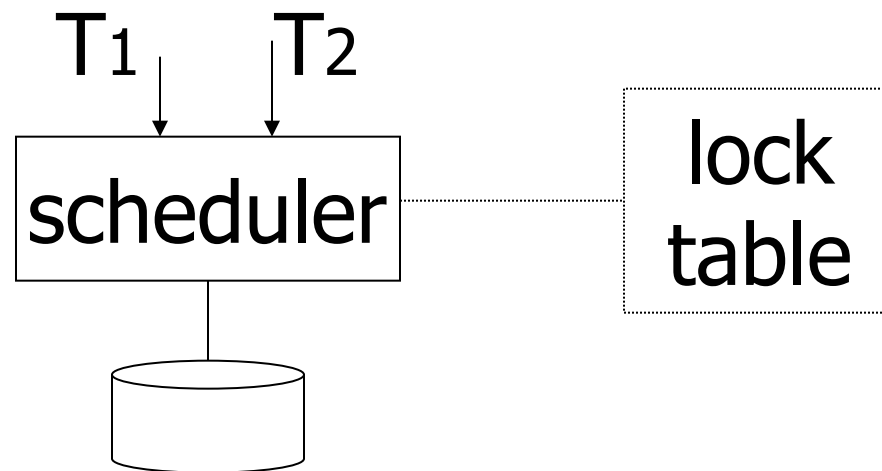
Rule #1: Well-formed transactions

T_i: ... $l_i(A)$ ... $p_i(A)$ ... $u_i(A)$ ...

1) Transaction has to lock A before it can access A

2) Transaction has to unlock A eventually

3) Transaction cannot access A after unlock

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rule #2 Legal scheduler

$$S = \text{........ } l_i(A) \text{ .... } u_i(A) \text{ ....}$$

$$\xleftrightarrow{\hspace{3cm}}$$

no $l_j(A)$

4) Only one transaction can hold a lock
   on A at the same time

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Exercise:

- What schedules are legal?
  What transactions are well-formed?

$S_1 = l_1(A)l_1(B)r_1(A)w_1(B)l_2(B)u_1(A)u_1(B)$
$r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)$

$S_2 = l_1(A)r_1(A)w_1(B)u_1(A)u_1(B)$
$l_2(B)r_2(B)w_2(B)l_3(B)r_3(B)u_3(B)$

$S_3 = l_1(A)r_1(A)u_1(A)l_1(B)w_1(B)u_1(B)$
$l_2(B)r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Exercise:

- What schedules are legal?
  What transactions are well-formed?

  S1 = l$_1$(A)l$_1$(B)r$_1$(A)w$_1$(B)l$_2$(B)u$_1$(A)u$_1$(B)

  r$_2$(B)w$_2$(B)u$_2$(B)l$_3$(B)r$_3$(B)u$_3$(B)

  S2 = l$_1$(A)r$_1$(A)w$_1$(B)u$_1$(A)u$_1$(B)

  l$_2$(B)r$_2$(B)w$_2$(B)l$_3$(B)r$_3$(B)u$_3$(B)

  S3 = l$_1$(A)r$_1$(A)u$_1$(A)l$_1$(B)w$_1$(B)u$_1$(B)

  l$_2$(B)r$_2$(B)w$_2$(B)u$_2$(B)l$_3$(B)r$_3$(B)u$_3$(B)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule F

T1	T2
$l_1(A)$;Read(A)	
$A \leftarrow A+100$;Write(A);$u_1(A)$	
	$l_2(A)$;Read(A)
	$A \leftarrow A \times 2$;Write(A);$u_2(A)$
	$l_2(B)$;Read(B)
	$B \leftarrow B \times 2$;Write(B);$u_2(B)$
$l_1(B)$;Read(B)	
$B \leftarrow B+100$;Write(B);$u_1(B)$	

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule F

		A	B
		25	25

T1	T2		
$l_1(A)$;Read(A)			
$A \leftarrow A+100$;Write(A);$u_1(A)$		125	
	$l_2(A)$;Read(A)		
	$A \leftarrow A \times 2$;Write(A);$u_2(A)$	250	
	$l_2(B)$;Read(B)		
	$B \leftarrow B \times 2$;Write(B);$u_2(B)$		50
$l_1(B)$;Read(B)			
$B \leftarrow B+100$;Write(B);$u_1(B)$			150
		250	150

COMPUTER
SCIENCE

Notes 14 - Concurrency Control

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Rule #3 Two phase locking (**2PL**)
for transactions

T_i = ……. $l_i(A)$ ………… $u_i(A)$ ………

← no unlocks   → no locks
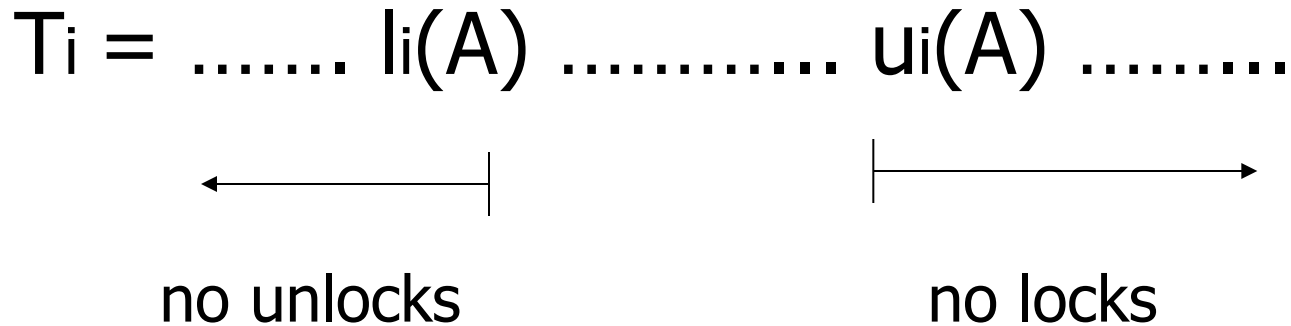
5) A transaction does not require new locks after its first unlock operation

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

locks held by Ti

Time

Growing Phase

Shrinking Phase

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Schedule G

T1	T2
$l_1(A)$;Read(A)	
A←A+100;Write(A)	
$l_1(B)$; $u_1(A)$	
	$l_2(A)$;Read(A) delayed
	A←Ax2;Write(A); $l_2(B)$

Schedule G

T1	T2
$l_1(A)$;Read(A)	
A←A+100;Write(A)	
$l_1(B)$; $u_1(A)$	
	$l_2(A)$;Read(A) delayed
	A←Ax2;Write(A); $l_2(B)$
Read(B);B← B+100	
Write(B); $u_1(B)$	

Schedule G

T1	T2
$l_1(A)$;Read(A)	
A←A+100;Write(A)	
$l_1(B)$; $u_1(A)$	
	$l_2(A)$;Read(A) delayed
	A←Ax2;Write(A); $l_2(B)$
Read(B);B← B+100	
Write(B); $u_1(B)$	
	$l_2(B)$; $u_2(A)$;Read(B)
	B← Bx2;Write(B);$u_2(B)$;

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Schedule H (T2 reversed)

T1	T2
l₁(A); Read(A)	l₂(B);Read(B)
A←A+100;Write(A)	B←Bx2;Write(B)
l₁(B)	l₂(A)
delayed	delayed

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deadlock

- Two or more transactions are waiting for each other to release a lock

- In the example
  - T_1 is waiting for T_2 and is making no progress
  - T_2 is waiting for T_1 and is making no progress
  - -> if we do not do anything they would wait forever

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

- Assume deadlocked transactions are rolled back
  - They have no effect
  - They do not appear in schedule
  - **Come back to that later**

E.g., Schedule H =

This space intentionally

left blank!

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Next step:

Show that rules #1,2,3 $\Rightarrow$ conflict-
serializable
schedules

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Conflict rules for $l_i(A)$, $u_i(A)$:

- $l_i(A)$, $l_j(A)$ conflict
- $l_i(A)$, $u_j(A)$ conflict

Note: no conflict $< u_i(A), u_j(A)>$, $< l_i(A), r_j(A)>$,...

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Theorem Rules #1,2,3 $\Rightarrow$ conflict
(2PL)   serializable
        schedule

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Theorem Rules #1,2,3 $\Rightarrow$ conflict
(2PL) serializable
schedule

To help in proof:

Definition Shrink(Ti) = SH(Ti) =
first unlock
action of Ti

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Lemma

$T_i \rightarrow T_j$ in $S \Rightarrow SH(T_i) <_S SH(T_j)$

Lemma

$T_i \rightarrow T_j$ in $S \Rightarrow SH(T_i) <_S SH(T_j)$

Proof of lemma:

$T_i \rightarrow T_j$ means that

$\quad$ $S = \ldots p_i(A) \ldots q_j(A) \ldots;$ $\quad$ p,q conflict

By rules 1,2:

$\quad$ $S = \ldots p_i(A) \ldots u_i(A) \ldots l_j(A) \ldots q_j(A) \ldots$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Lemma

$Ti \rightarrow Tj$ in $S \Rightarrow SH(Ti) <_S SH(Tj)$

Proof of lemma:

$Ti \rightarrow Tj$ means that

$\quad S = ... \, p_i(A) \, ... \, q_j(A) \, ...; \quad$ p,q conflict

By rules 1,2:

$\quad S = ... \, p_i(A) \, ... \, u_i(A) \, ... \, l_j(A) \, ... \, q_j(A) \, ...$

By rule 3: $\quad SH(Ti) \qquad SH(Tj)$

So, $SH(Ti) <_S SH(Tj)$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Theorem Rules #1,2,3 $\Rightarrow$ conflict
(2PL)   serializable schedule

Proof:

(1) Assume P(S) has cycle

$$T_1 \to T_2 \to \dots T_n \to T_1$$

(2) By lemma: $SH(T_1) < SH(T_2) < \dots < SH(T_1)$

(3) Impossible, so P(S) acyclic

(4) $\Rightarrow$ S is conflict serializable

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

2PL subset of Serializable

$$S \subset 2PL \subset CSR \subset ALL$$

All schedules (**ALL**)

Conflict Serializable (**CSR**)

2PL (**2PL**)

Serial (**S**)

S1: w1(x) w3(x) w2(y) w1(y)

- S1 cannot be achieved via 2PL:
  The lock by T1 for y must occur after w2(y), so the unlock by T1 for x must occur after this point (and before w1(x)). Thus, w3(x) cannot occur under 2PL where shown in S1 because T1 holds the x lock at that point.

- However, S1 is serializable (equivalent to T2, T1, T3).

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

If you need a bit more practice:

Are our schedules S_C and S_D 2PL schedules?

S_C: w1(A) w2(A) w1(B) w2(B)

S_D: w1(A) w2(A) w2(B) w1(B)

Notes 14 - Concurrency Control

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

- Beyond this simple **2PL** protocol, it is all a matter of improving performance and allowing more concurrency….
  - Shared locks
  - Multiple granularity
  - Avoid Deadlocks
  - Inserts, deletes and phantoms
  - Other types of C.C. mechanisms
    - Multiversioning concurrency control

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Shared locks

So far:

$$S = \ldots l_1(A)\ r_1(A)\ u_1(A)\ \ldots\ l_2(A)\ r_2(A)\ u_2(A)\ \ldots$$

Do not conflict

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Shared locks

So far:

$$S = \ldots l_1(A)\ r_1(A)\ u_1(A)\ \ldots\ l_2(A)\ r_2(A)\ u_2(A)\ \ldots$$

Do not conflict

Instead:

$$S = \ldots\ ls_1(A)\ r_1(A)\ ls_2(A)\ r_2(A)\ \ldots\ us_1(A)\ us_2(A)$$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Lock actions

l-t_i(A): lock A in t mode (t is S or X)

u-t_i(A): unlock t mode (t is S or X)

Shorthand:

u_i(A): unlock whatever modes

        T_i has locked A

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Rule #1 Well formed transactions

$T_i = ...$ l-$S_1(A)$ ... $r_1(A)$ ... $u_1(A)$ ...

$T_i = ...$ l-$X_1(A)$ ... $w_1(A)$ ... $u_1(A)$ ...

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- What about transactions that read and write same object?

Option 1: Request exclusive lock

$T_i = ...l\text{-}X_1(A) ... r_1(A) ... w_1(A) ... u(A) ...$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

- What about transactions that read and write same object?

Option 2: Upgrade

(E.g., need to read, but don't know if will write...)

$$T_i = \ldots\ l\text{-}S_1(A)\ \ldots\ r_1(A)\ \ldots\ l\text{-}X_1(A)\ \ldots w_1(A)\ \ldots u(A)\ldots$$

Think of
- Get 2nd lock on A, or
- Drop S, get X lock

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rule #2 Legal scheduler

$S = \dots l\text{-}S_i(A) \dots \quad \dots u_i(A) \dots$

no $l\text{-}X_j(A)$

$S = \dots l\text{-}X_i(A) \dots \quad \dots u_i(A) \dots$

no $l\text{-}X_j(A)$
no $l\text{-}S_j(A)$

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
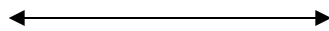
A way to summarize Rule #2

Compatibility matrix

Comp

	S	X
S	true	false
X	false	false

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Rule # 3   2PL transactions

No change except for upgrades:

(I) If upgrade gets more locks

(e.g., S → {S, X}) then no change!

(II) If upgrade releases read (shared) lock (e.g., S → X)

- can be allowed in growing phase

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Theorem Rules 1,2,3 $\Rightarrow$ Conf.serializable
for S/X locks   schedules

Proof: similar to X locks case

Detail:
l-t_i(A), l-r_j(A) do not conflict if comp(t,r)
l-t_i(A), u-r_j(A) do not conflict if comp(t,r)

Lock types beyond S/X

Examples:

       (1) increment lock

       (2) update lock

Notes 14 - Concurrency Control

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example (1): increment lock

- Atomic increment action: $IN_i(A)$

$$\{Read(A); A \leftarrow A+k; Write(A)\}$$

- $IN_i(A)$, $IN_j(A)$ do not conflict!

Notes 14 - Concurrency Control

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Comp

	S	X	I
S			
X			
I			

COMPUTER SCIENCE

Notes 14 - Concurrency Control

100

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Comp

	S	X	I
S	T	F	F
X	F	F	F
I	F	F	T

Notes 14 - Concurrency Control

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
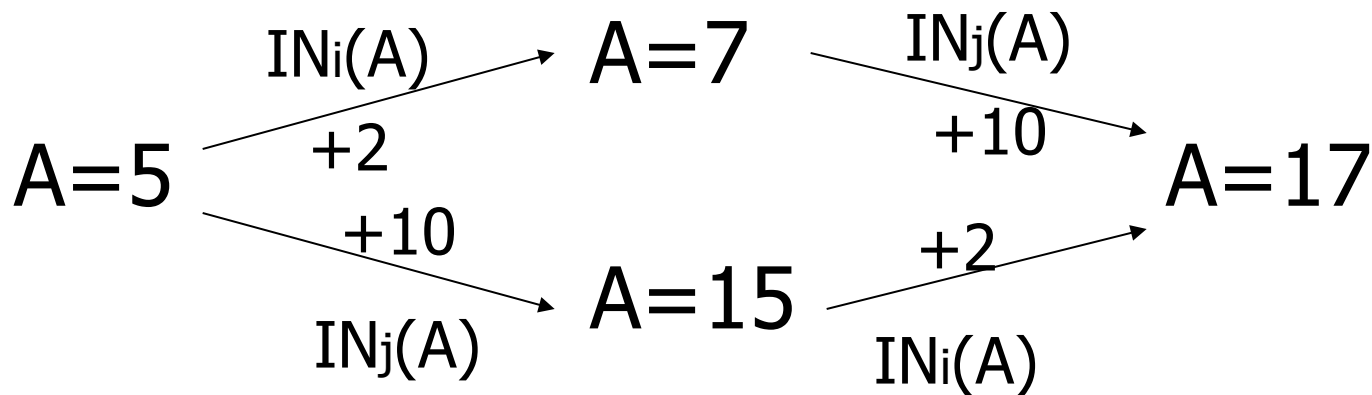
Update locks

A common deadlock problem with upgrades:

T1	T2
l-S_1(A)	
	l-S_2(A)
l-X_1(A)	
	l-X_2(A)

--- Deadlock ---
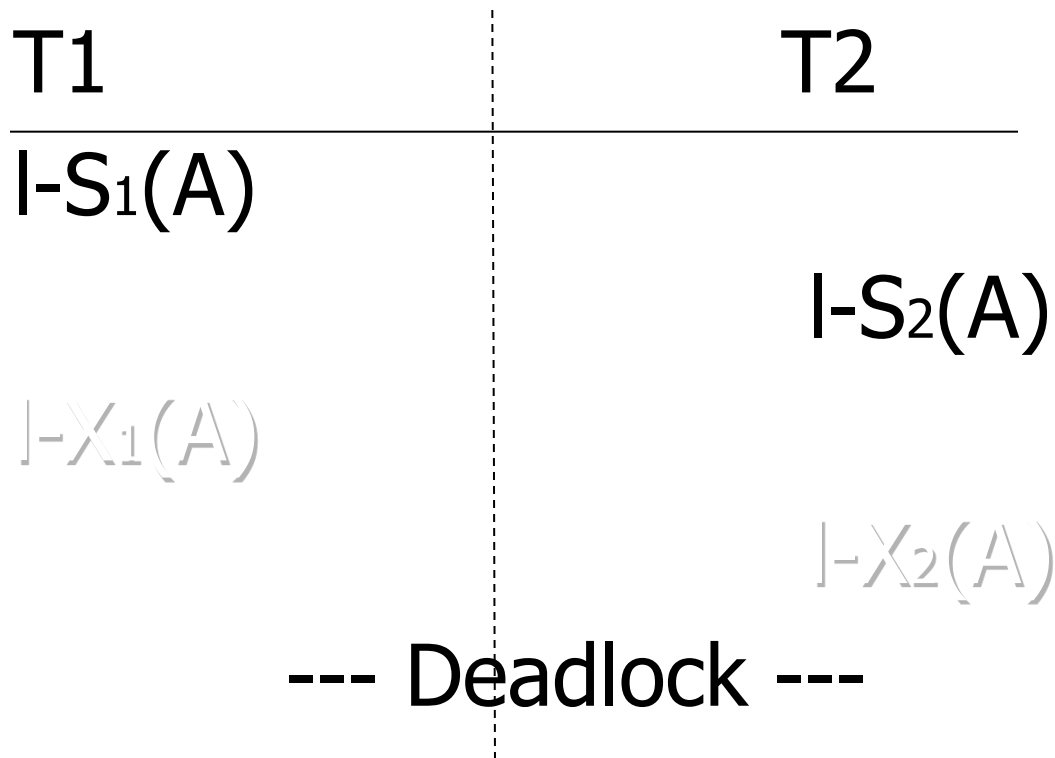
Solution

If T_i wants to read A and knows it may later want to write A, it requests <u>update</u> lock (not shared)

Notes 14 - Concurrency Control

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

New request

Comp

	S	X	U
S			
X			
U			

Lock already held in

COMPUTER SCIENCE

Notes 14 - Concurrency Control

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

New request

Comp

Lock already held in

	S	X	U
S	T	F	T
X	F	F	F
U	TorF	F	F

-> symmetric table?

Note: object A may be locked in different modes at the same time...

$$S_1 = ...l\text{-}S_1(A)...l\text{-}S_2(A)...l\text{-}U_3(A)... \begin{cases} l\text{-}S_4(A)...? \\ l\text{-}U_4(A)...? \end{cases}$$

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Note: object A may be locked in different modes at the same time...

$$S_1 = \ldots l\text{-}S_1(A)\ldots l\text{-}S_2(A)\ldots l\text{-}U_3(A)\ldots \begin{cases} l\text{-}S_4(A)\ldots? \\ l\text{-}U_4(A)\ldots? \end{cases}$$

- To grant a lock in mode t, mode t must be compatible with all currently held locks on object

How does locking work in practice?

- Every system is different

    (E.g., may not even provide
    CONFLICT-SERIALIZABLE schedules)

- But here is one (simplified) way ...

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Sample Locking System:

(1) Don't trust transactions to request/release locks

(2) Hold all locks until transaction commits

\# locks

time

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
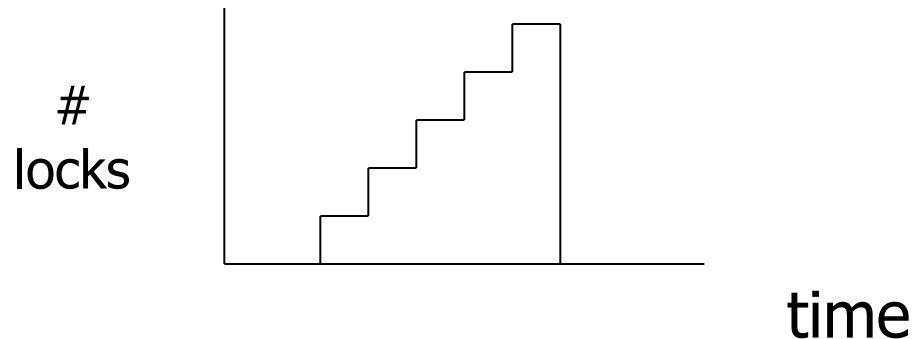
Strict Strong 2PL (**SS2PL**)

- 2PL + (2) from the last slide
- All locks are held until transaction end
- Compare with schedule class **strict** (**ST**) we defined for recovery
  - A transaction never reads or writes items written by an uncommitted transactions
- **SS2PL** = (**ST** ∩ **2PL**)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

All schedules (**ALL**)

Conflict Serializable (**CSR**)

2PL (**2PL**)

SS2PL (**SS2PL**)

Serial (**S**)
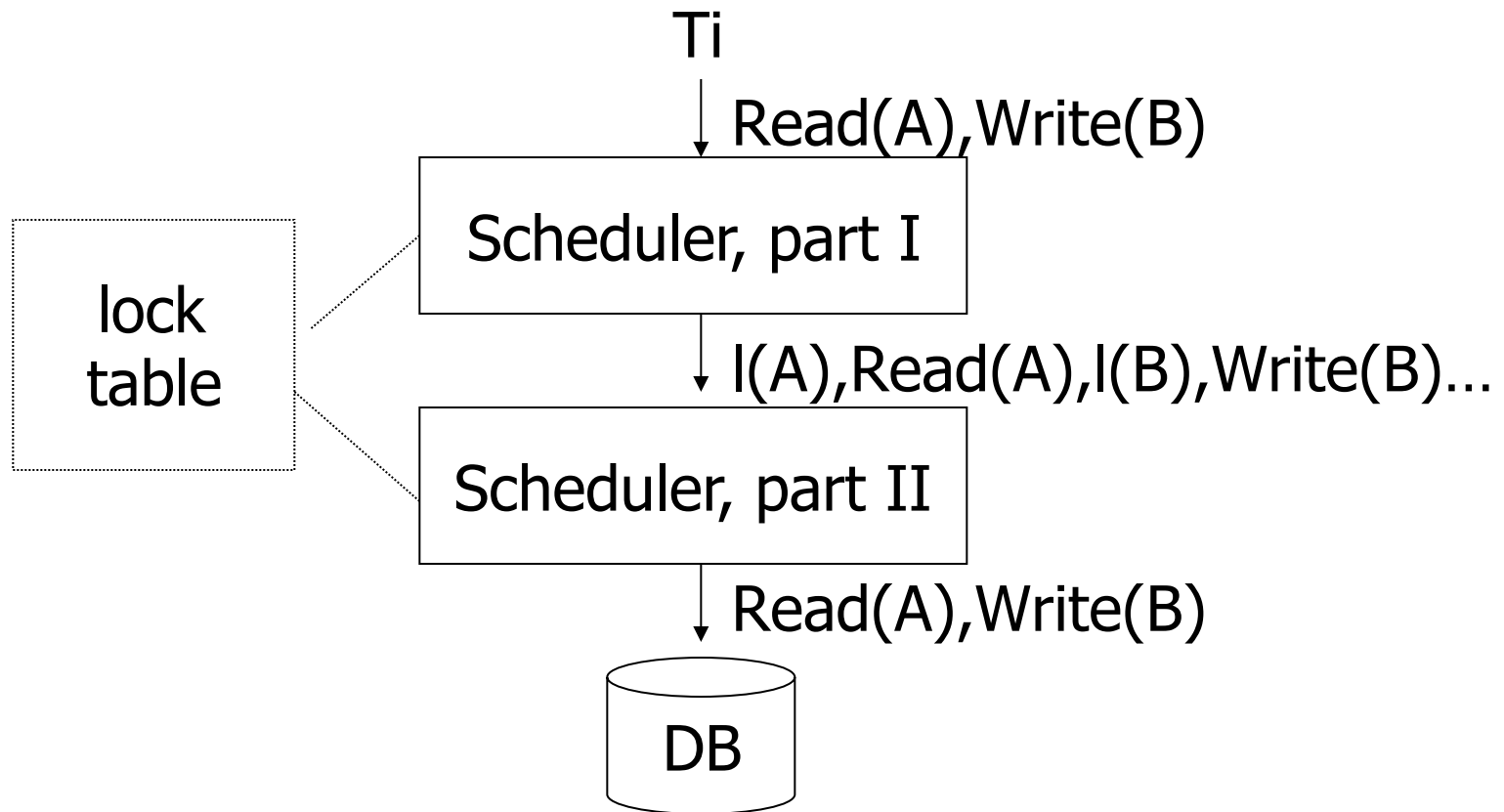
Lock table   Conceptually

If null, object is unlocked

A	Λ
B	
C	
	Λ
⋮	

Every possible object

Lock info for B

Lock info for C

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

But use hash table:

A
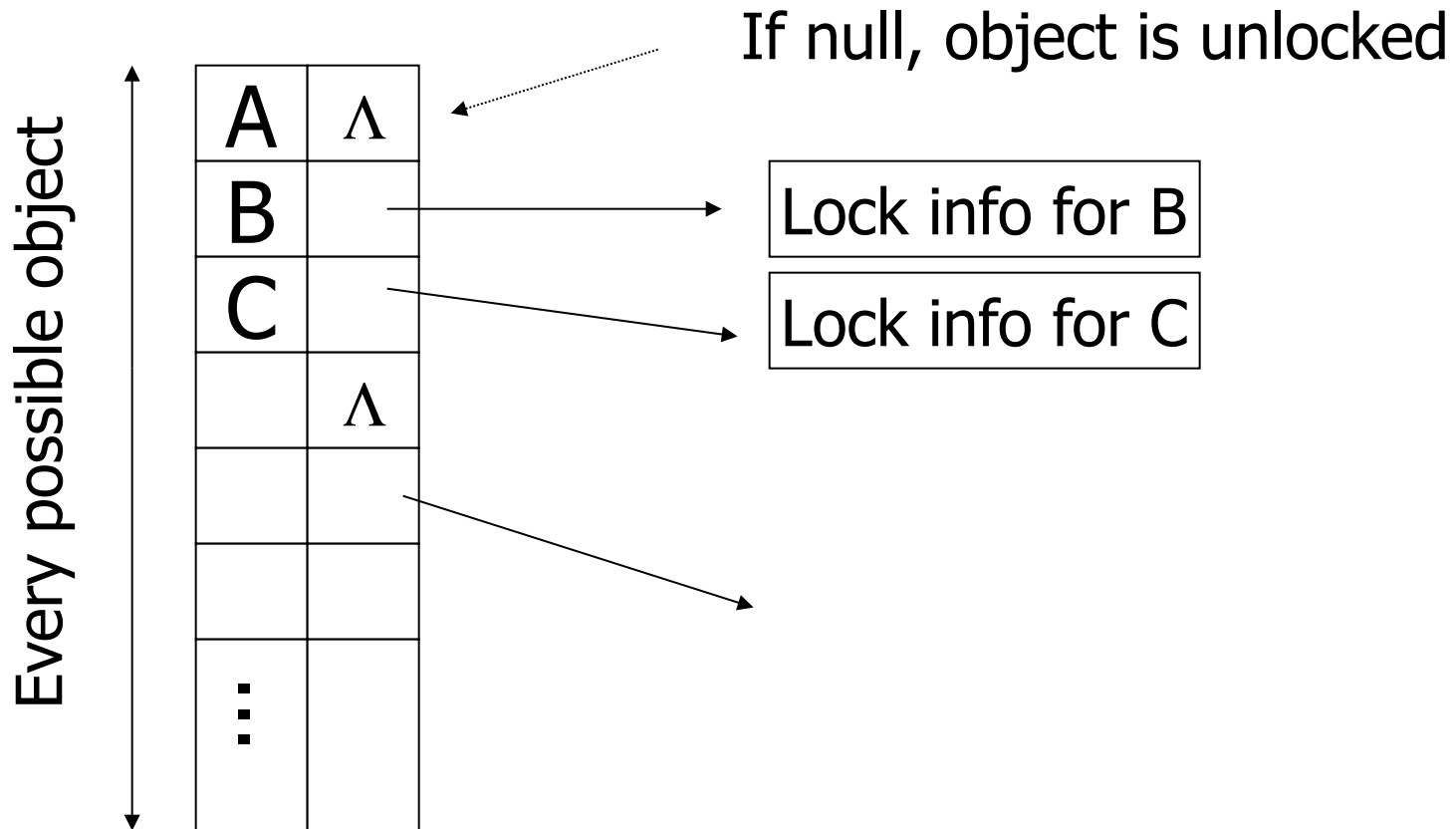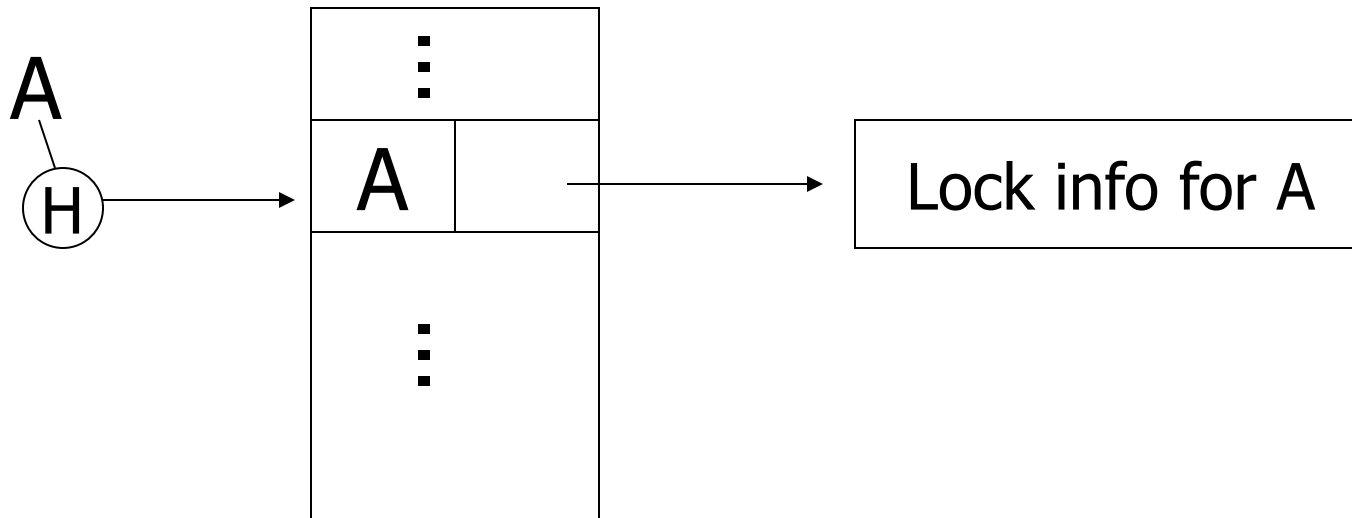
(H) ⟶ [... / A |] ⟶ [Lock info for A]

If object not found in hash table, it is unlocked

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Lock info for A - example

Object:A
Group mode:U
Waiting:yes
List:

tran mode wait? Nxt T_link

T1	S	no			
T2	U	no			
T3	X	yes	Λ		

To other T3
records
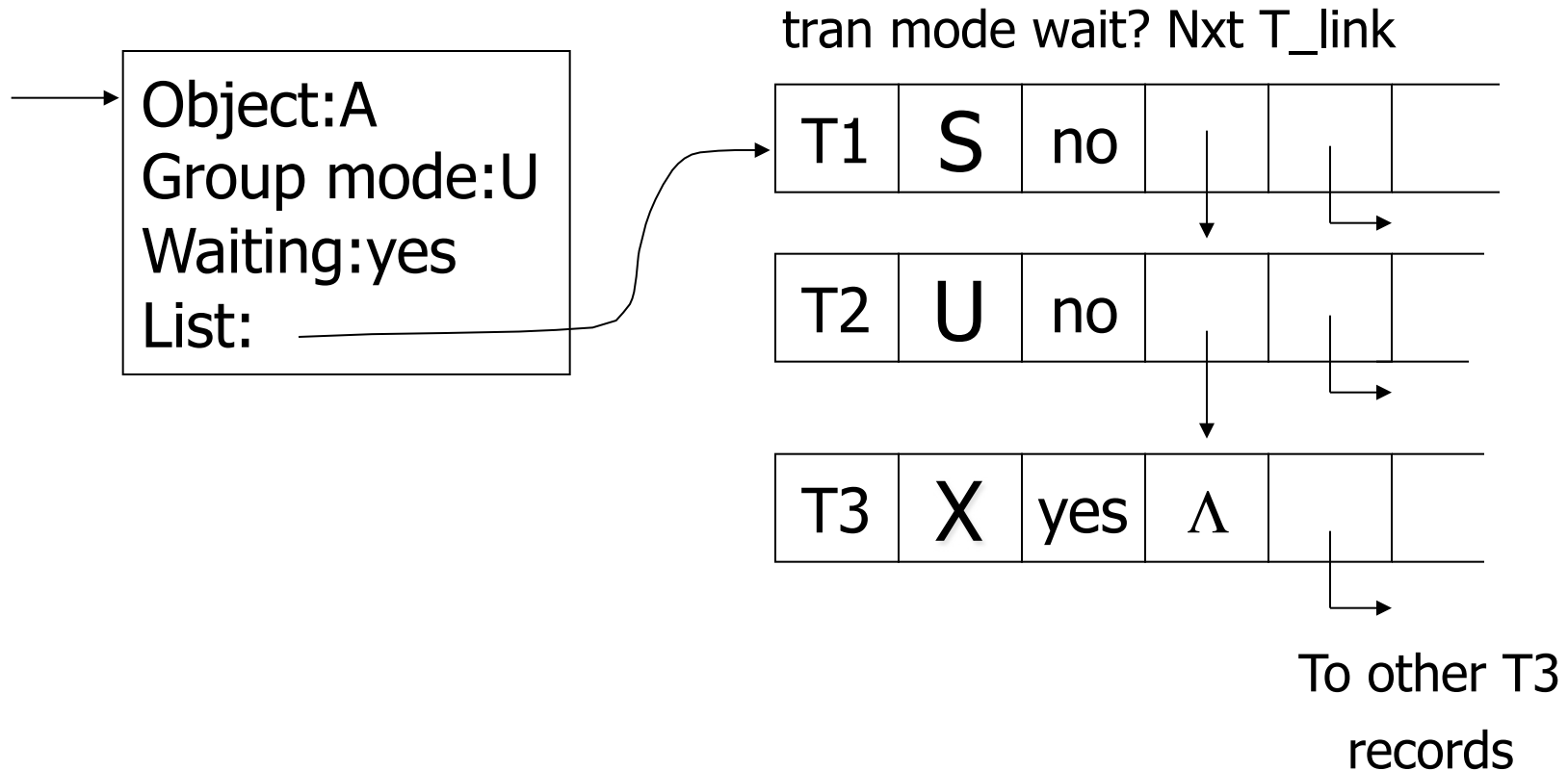
What are the objects we lock?
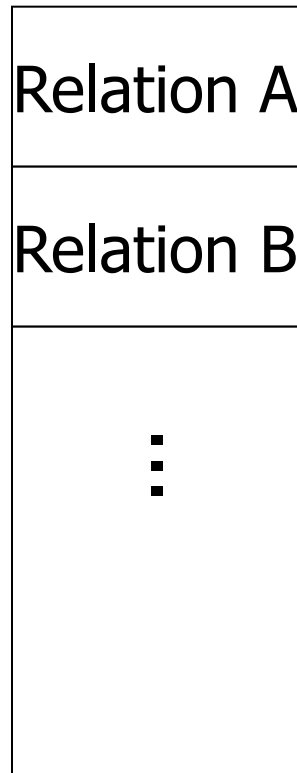
DB	DB	DB
Relation A	Tuple A	Disk block A
Relation B	Tuple B	Disk block B
⋮	Tuple C	⋮
	⋮	
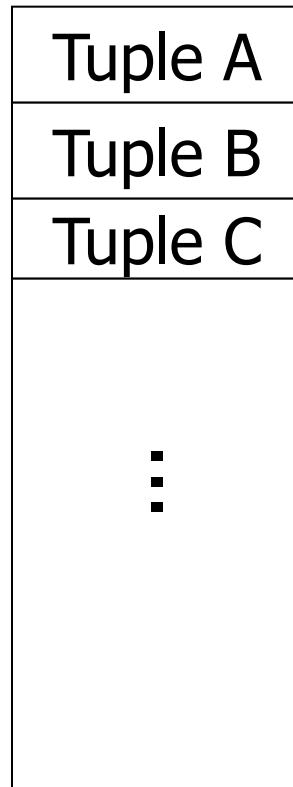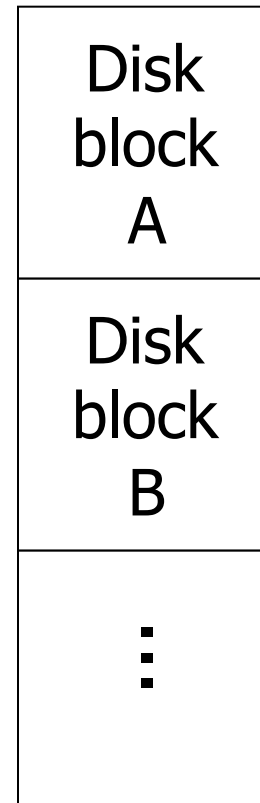
?

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

- Locking works in any case, but should we choose <u>small</u> or <u>large objects?</u>

COMPUTER SCIENCE

IIT College of Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

- Locking works in any case, but should we choose <u>small</u> or <u>large objects?</u>

- If we lock <u>large</u> objects (e.g., Relations)
  - Need few locks
  - Low concurrency
- If we lock small objects (e.g., tuples,fields)
  - Need more locks
  - More concurrency

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

We <u>can</u> have it both ways!!

Ask any janitor to give you the solution...

Stall 1	Stall 2	Stall 3	Stall 4

restroom

hall

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Notes 14 - Concurrency Control

120

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
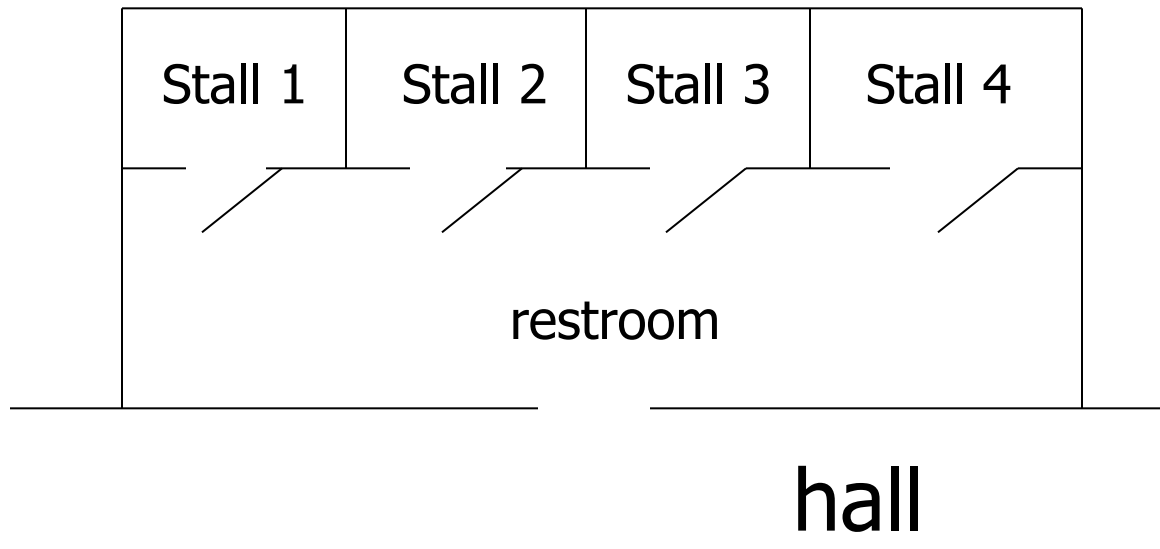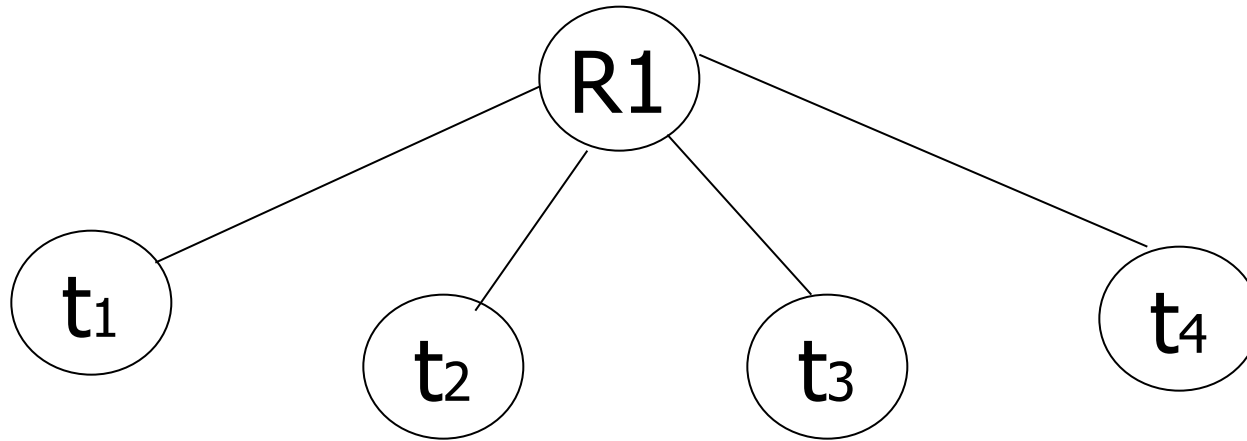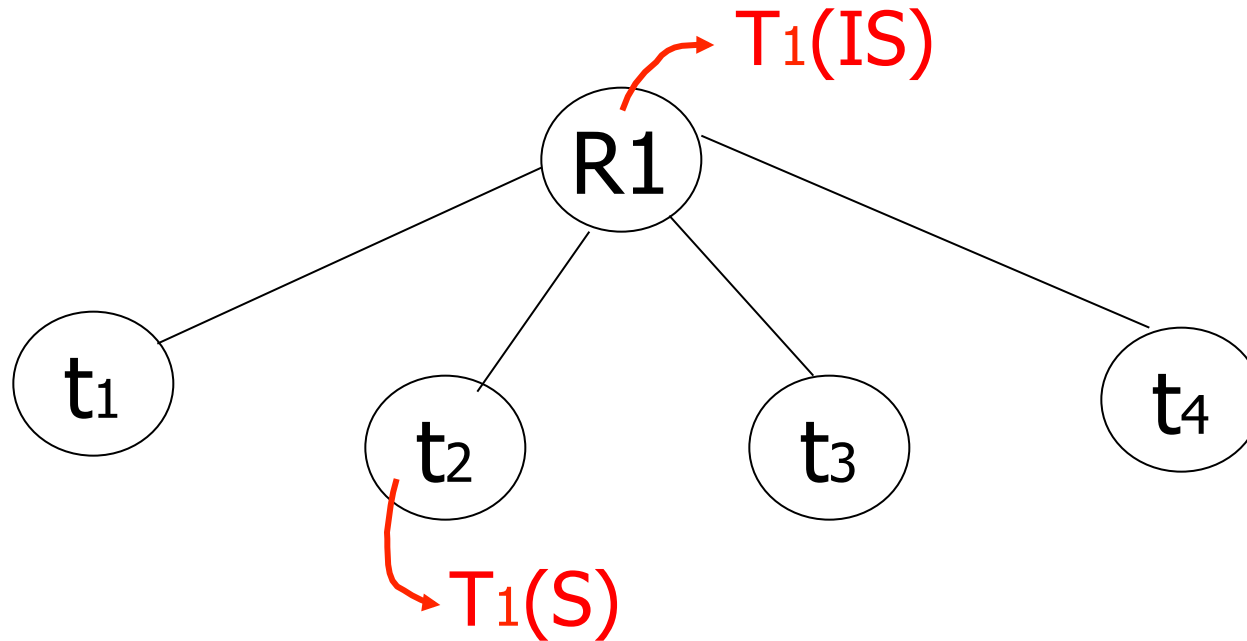
Example

Example (b)

Example
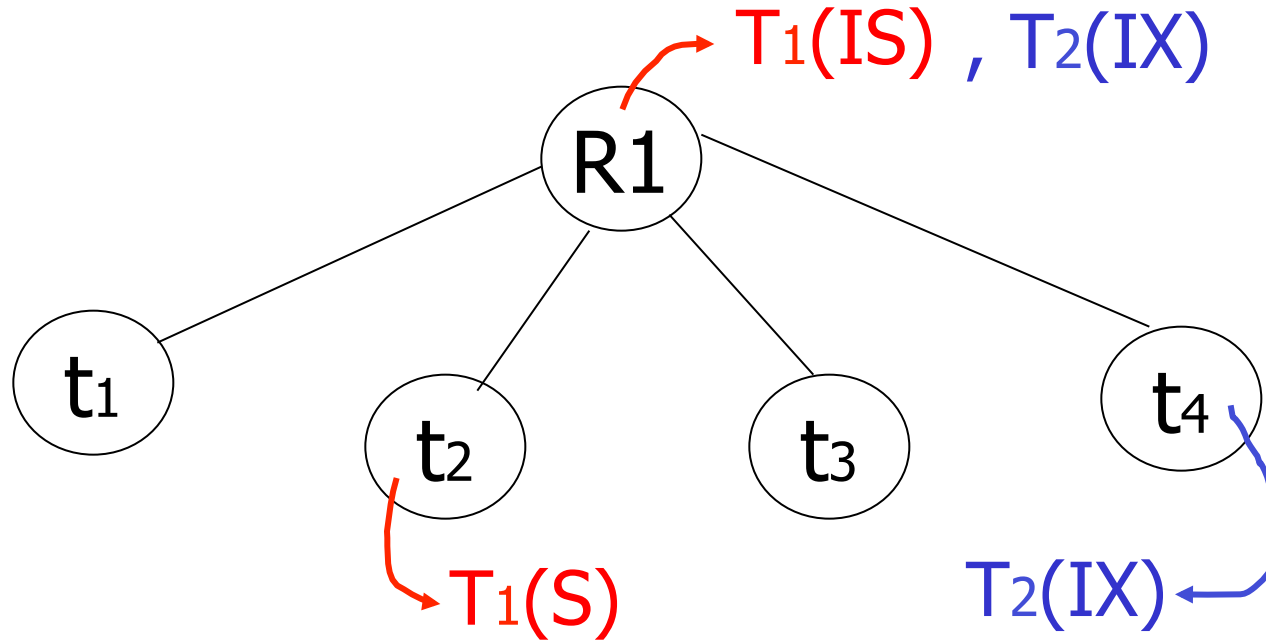
T₁(IS) , T₂(IX)
R1
t₁   t₂   t₃   t₄
T₁(S)
T₂(IX)

Multiple granularity

Comp   Requestor

		IS	IX	S	SIX	X
	IS					
Holder	IX					
	S					
	SIX					
	X					

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Multiple granularity

Comp   Requestor

		IS	IX	S	SIX	X
	IS	T	T	T	T	F
Holder	IX	T	T	F	F	F
	S	T	F	T	F	F
	SIX	T	F	F	F	F
	X	F	F	F	F	F

Notes 14 - Concurrency Control

126

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Parent locked in	Child can be locked in
IS	
IX	
S	
SIX	
X	

P

C
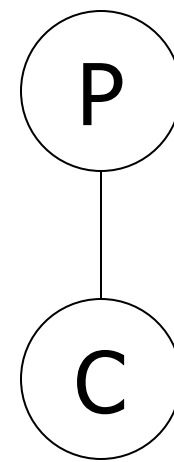
Parent locked in	Child can be locked by same transaction in
IS	IS, S
IX	IS, S, IX, X, SIX
S	none
SIX	X, IX, [SIX]
X	none

not necessary

P

C
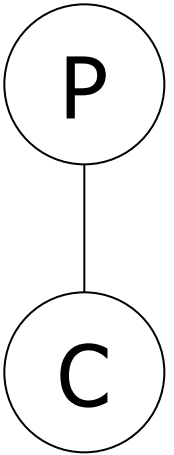
Rules

(1) Follow multiple granularity comp function

(2) Lock root of tree first, any mode

(3) Node Q can be locked by Ti in S or IS only if
    parent(Q) locked by Ti in IX or IS

(4) Node Q can be locked by Ti in X,SIX,IX only
    if parent(Q) locked by Ti in IX,SIX

(5) Ti is two-phase

(6) Ti can unlock node Q only if none of Q's
    children are locked by Ti

Notes 14 - Concurrency Control

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Exercise:

- Can T2 access object f2.2 in X mode? What locks will T2 get?

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
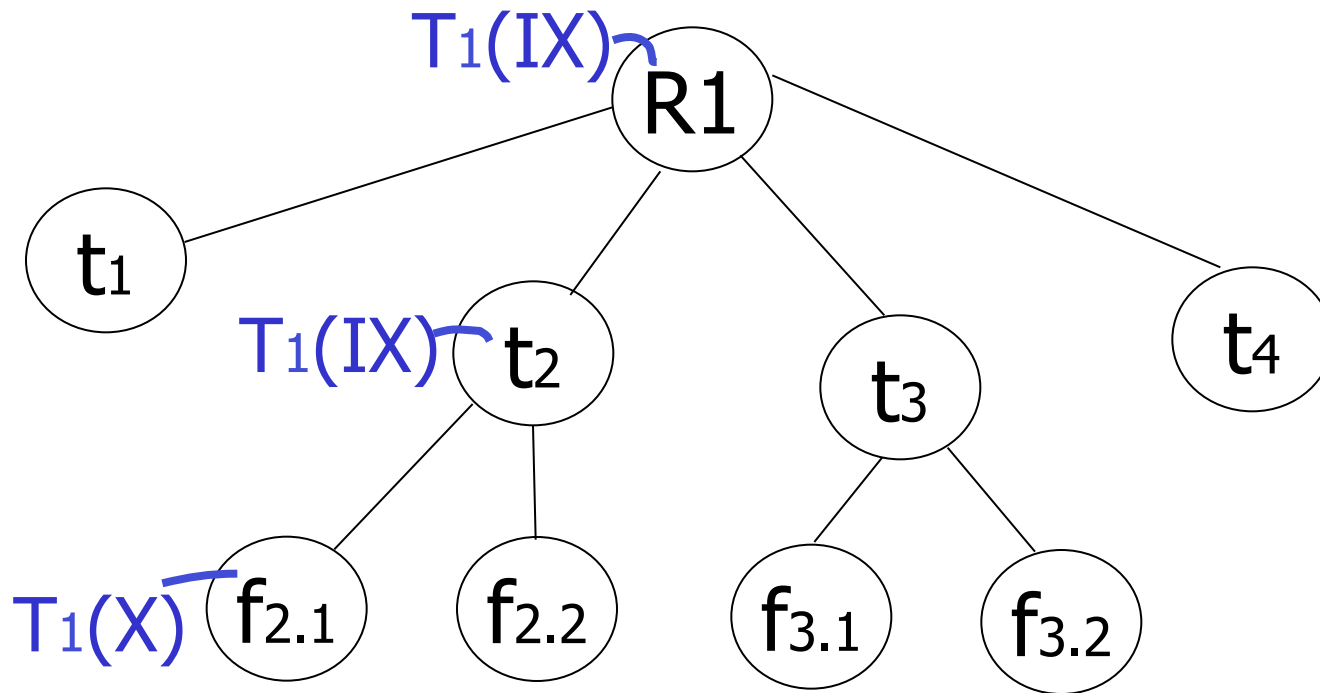
Exercise:

- Can T2 access object f2.2 in X mode? What locks will T2 get?

Exercise:
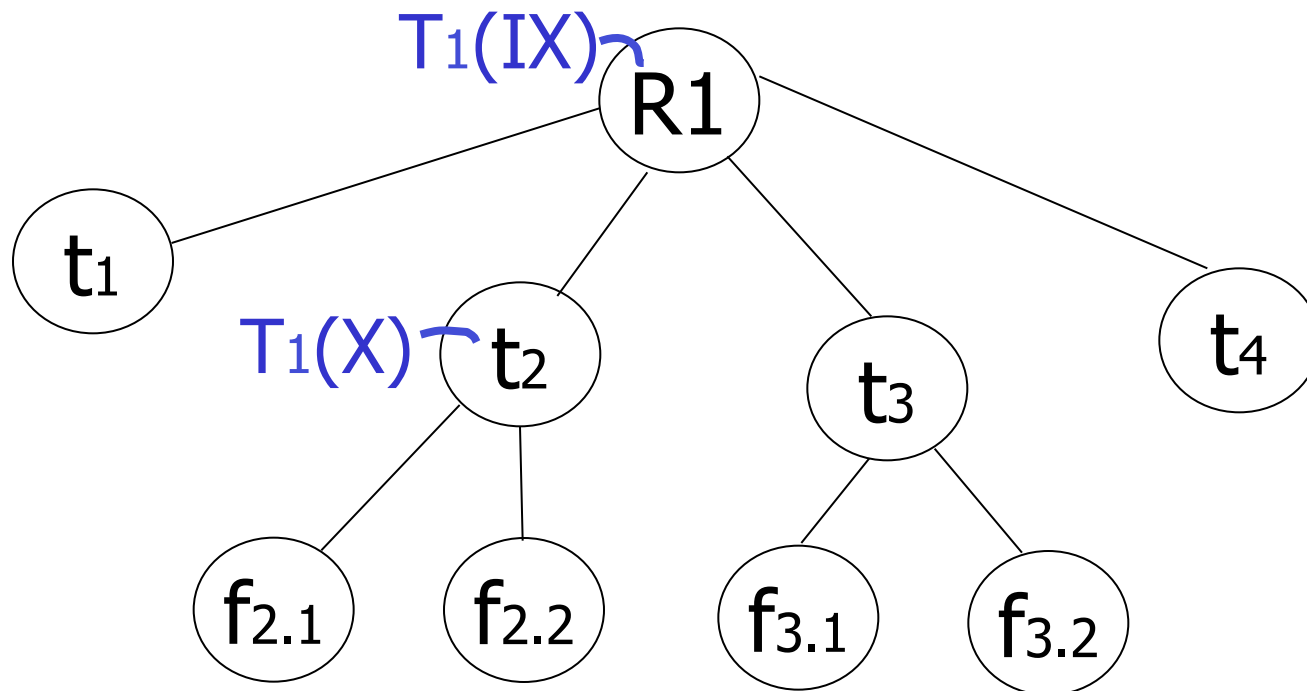
- Can T2 access object f3.1 in X mode? What locks will T2 get?

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
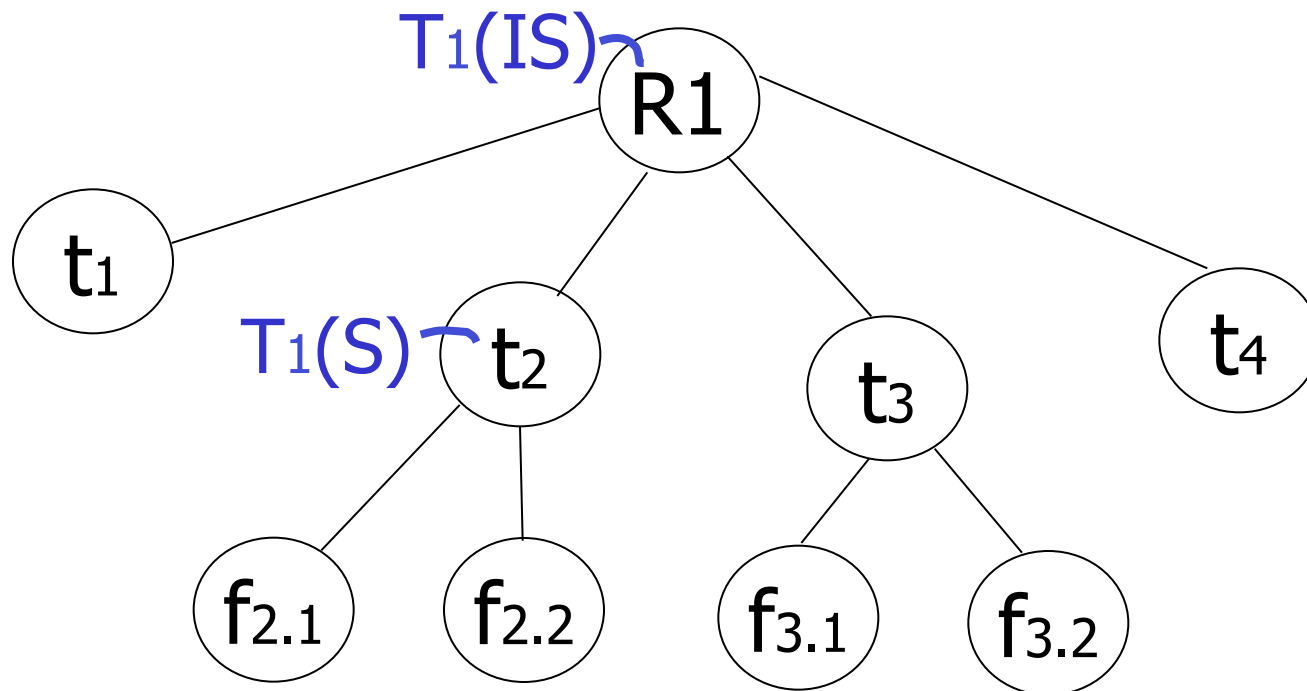
Exercise:

- Can T2 access object f2.2 in S mode? What locks will T2 get?

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Exercise:

- Can T2 access object f2.2 in X mode? What locks will T2 get?

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
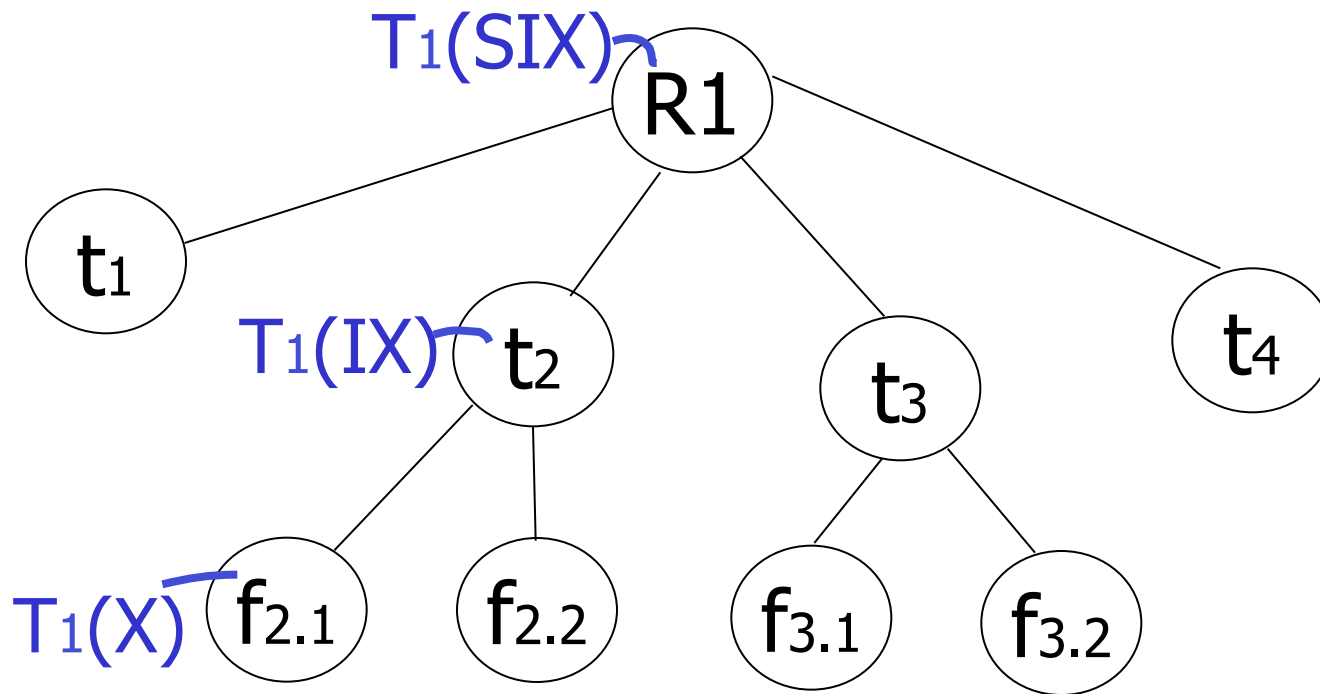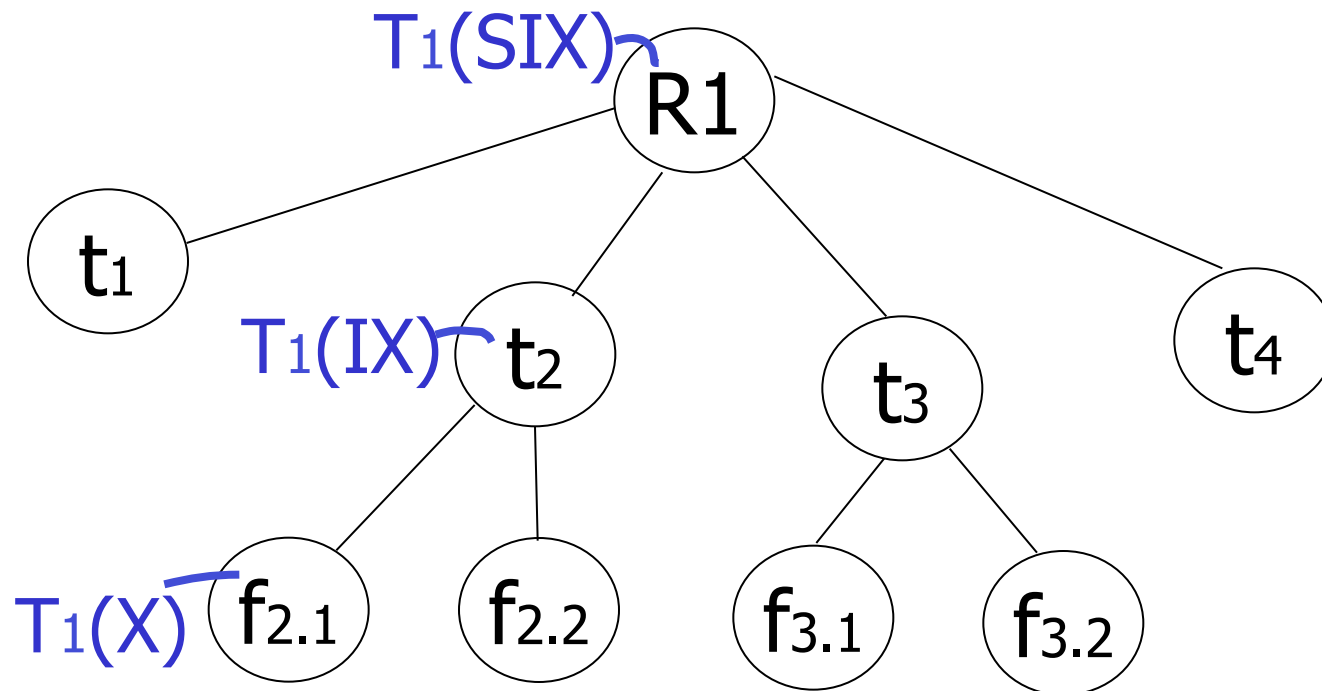
Insert + delete operations

$$
\begin{array}{|c|}
\hline
A \\
\hline
\vdots \\
\hline
Z \\
\hline
\alpha \\
\hline
\end{array}
\qquad \longleftarrow \text{Insert}
$$

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Modifications to locking rules:

(1) Get exclusive lock on A before deleting A

(2) At insert A operation by Ti, Ti is given exclusive lock on A

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Still have a problem: **Phantoms**

Example: relation R (E#,name,...)
constraint: E# is key
use tuple locking

R   E#   Name   ....

	E#	Name	....
o1	55	Smith	
o2	75	Jones	

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

T1: Insert <08,Obama,...> into R
T2: Insert <08,McCain,...> into R

T1	T2
$S_1(o_1)$	$S_2(o_1)$
$S_1(o_2)$	$S_2(o_2)$
Check Constraint	Check Constraint
$\vdots$	$\vdots$
Insert o_3[08,Obama,..]	
	Insert o_4[08,McCain,..]

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
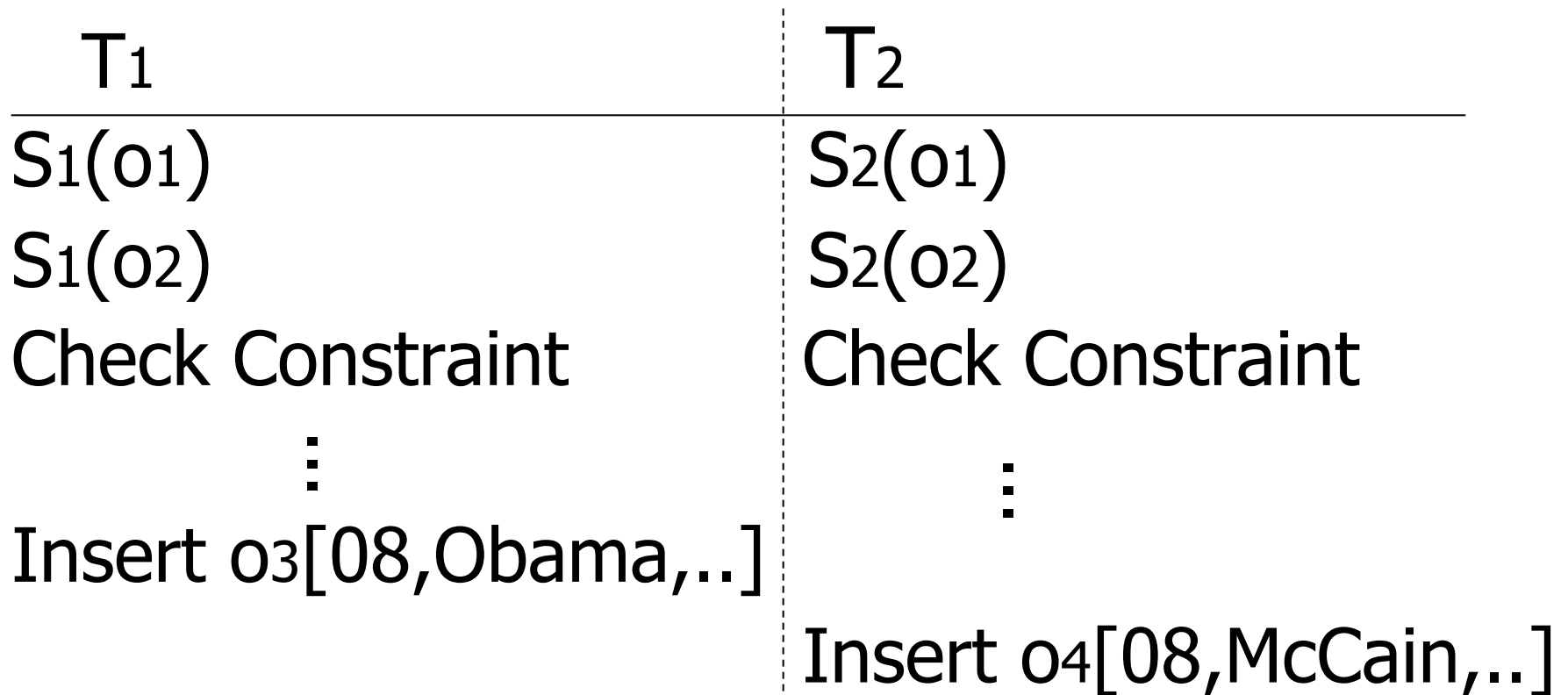
Solution

- Use multiple granularity tree
- Before insert of node Q,
  lock parent(Q) in
  X mode

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
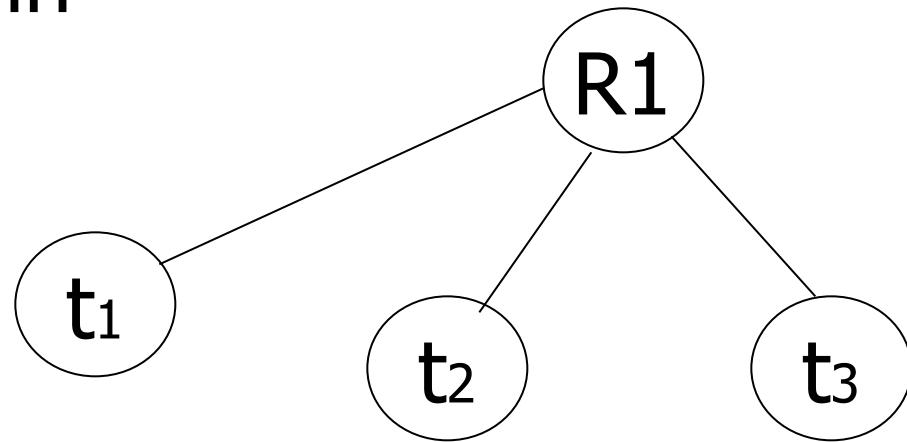
Back to example

T1: Insert<04,Kerry>	T2: Insert<04,Bush>
T1	T2

$X_1(R)$

$X_2(R)$ — *delayed*

Check constraint
Insert<04,Kerry>
$U(R)$

$X_2(R)$
Check constraint
Oops! e# = 04 already in R!

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Instead of using R, can use index on R:

Example:

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
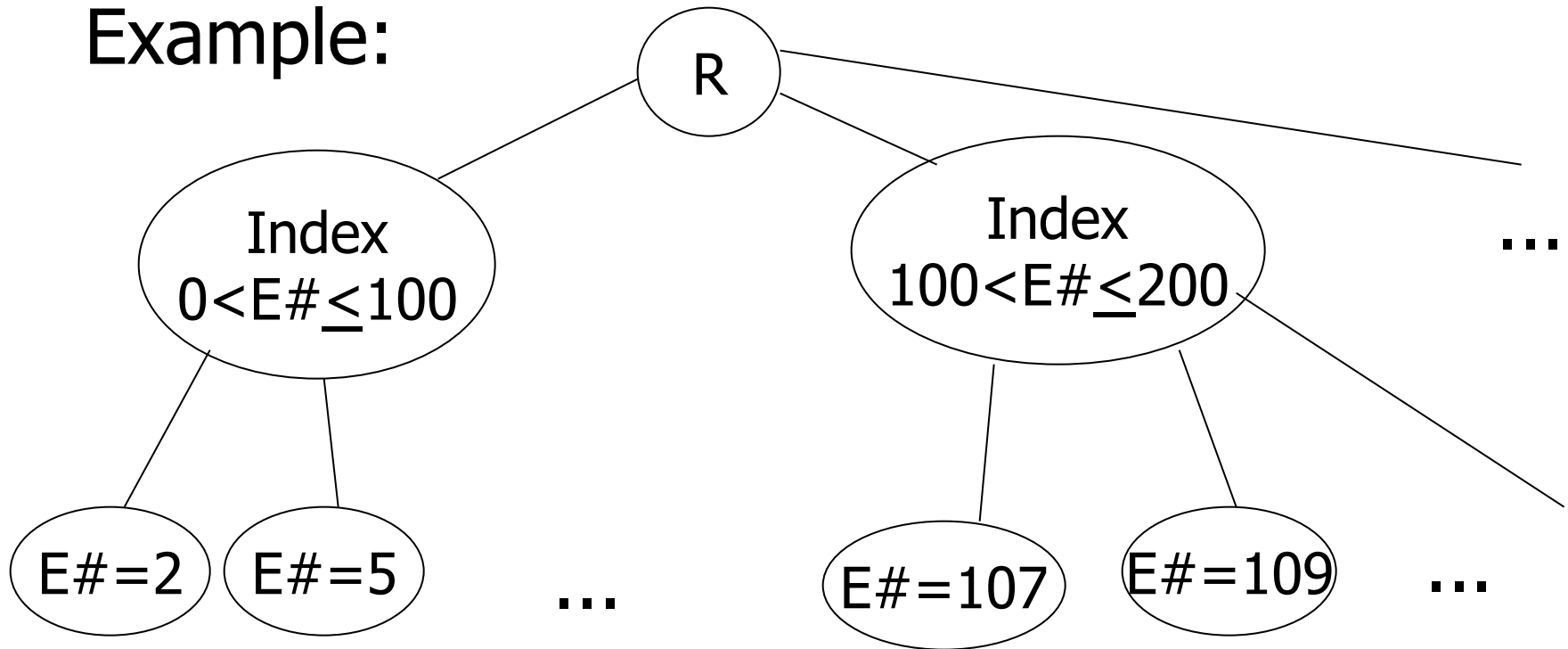
- This approach can be generalized to multiple indexes…

Notes 14 - Concurrency Control

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Next:

- Tree-based concurrency control
- Validation concurrency control
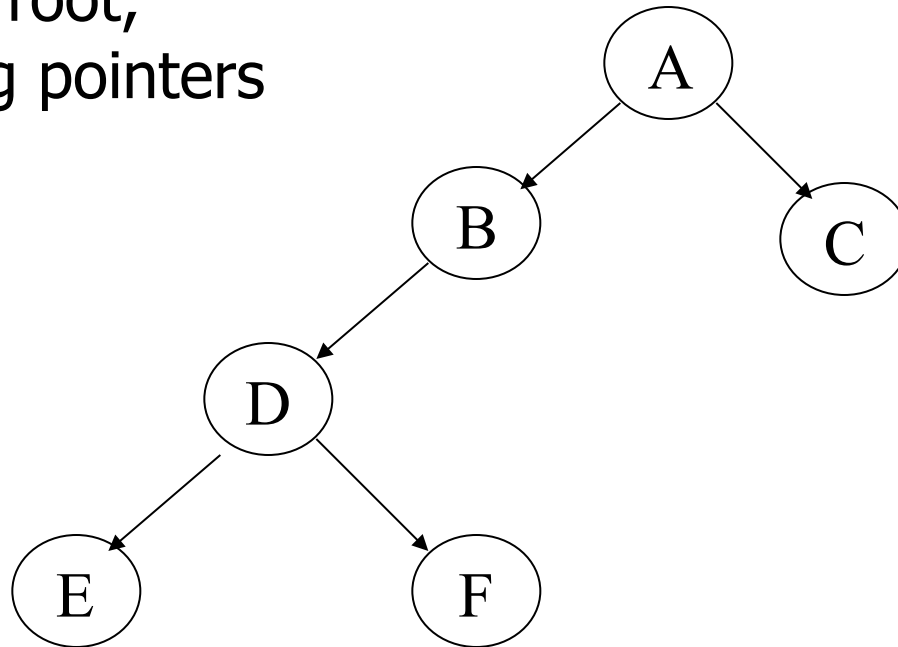
COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

- all objects accessed through root, following pointers

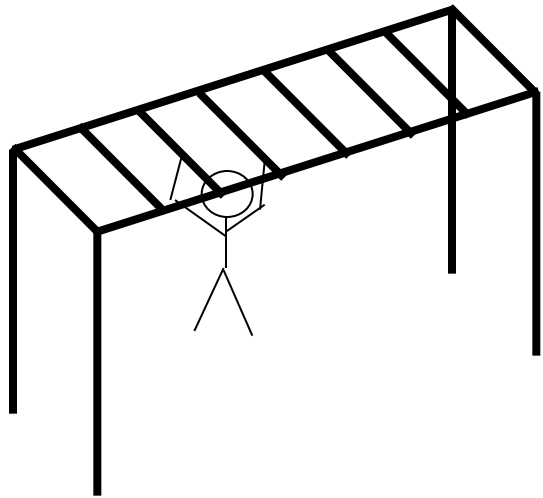Notes 14 - Concurrency Control

IIT College of
Science and Letters

COMPUTER SCIENCE

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

• all objects accessed through root, following pointers

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

- all objects accessed through root, following pointers

T1 lock → A

B   C

T1 lock → D ← T1 lock

E   F
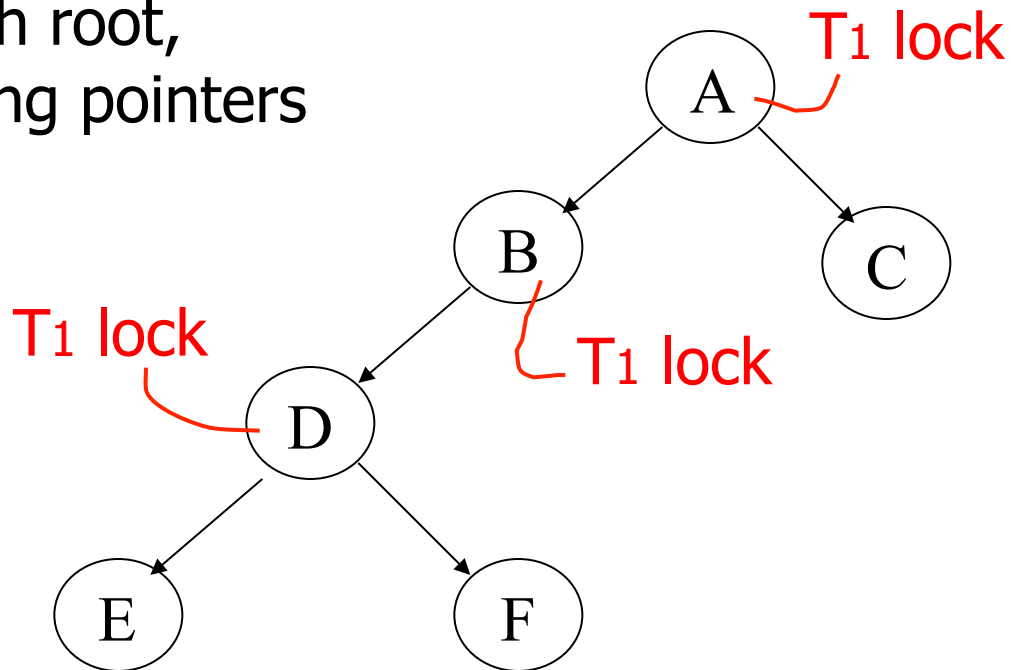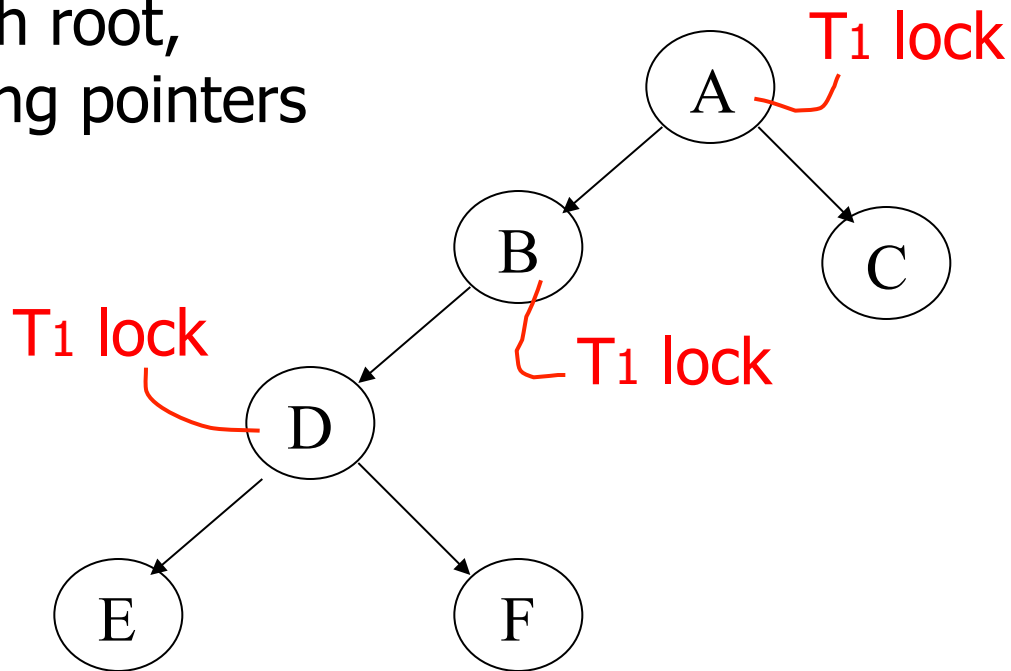
☛ can we release A lock
   if we no longer need A??

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE
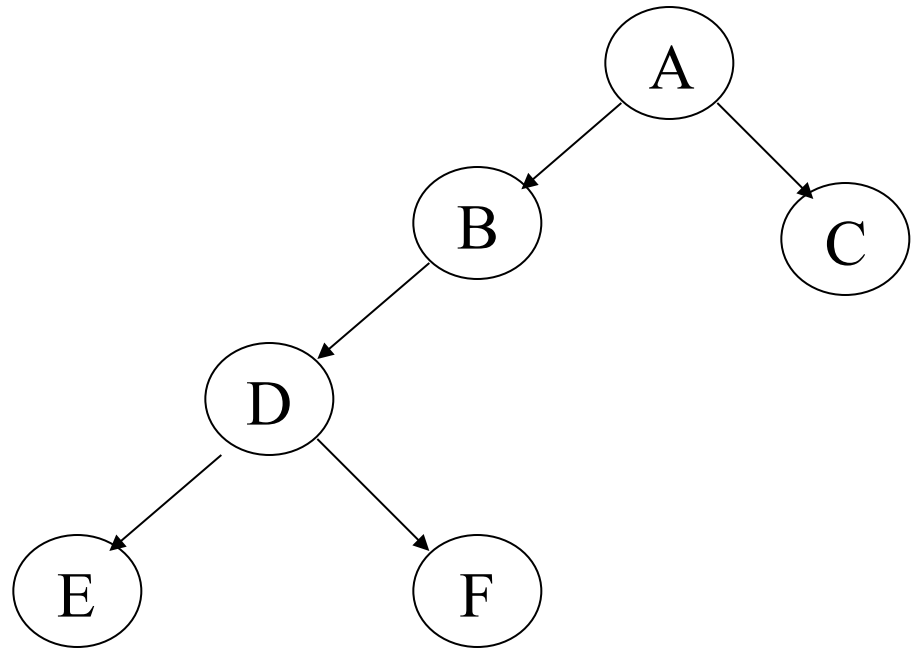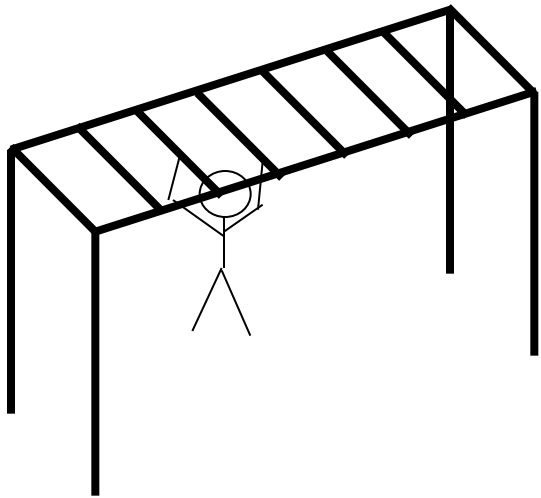
Idea: traverse like "Monkey Bars"

Idea: traverse like "Monkey Bars"

T1 lock

A

B   C

T1 lock

D

E   F

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Idea: traverse like "Monkey Bars"

T1 lock

T1 lock

Notes 14 - Concurrency Control
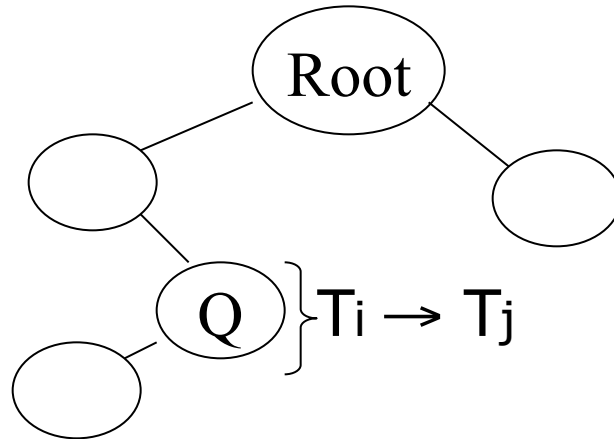
IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE
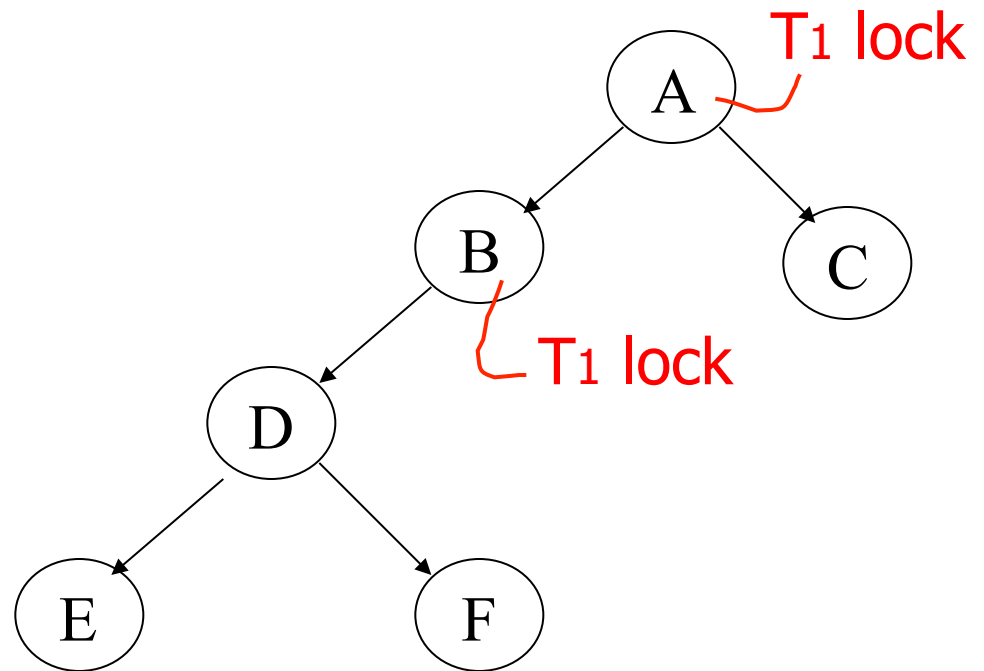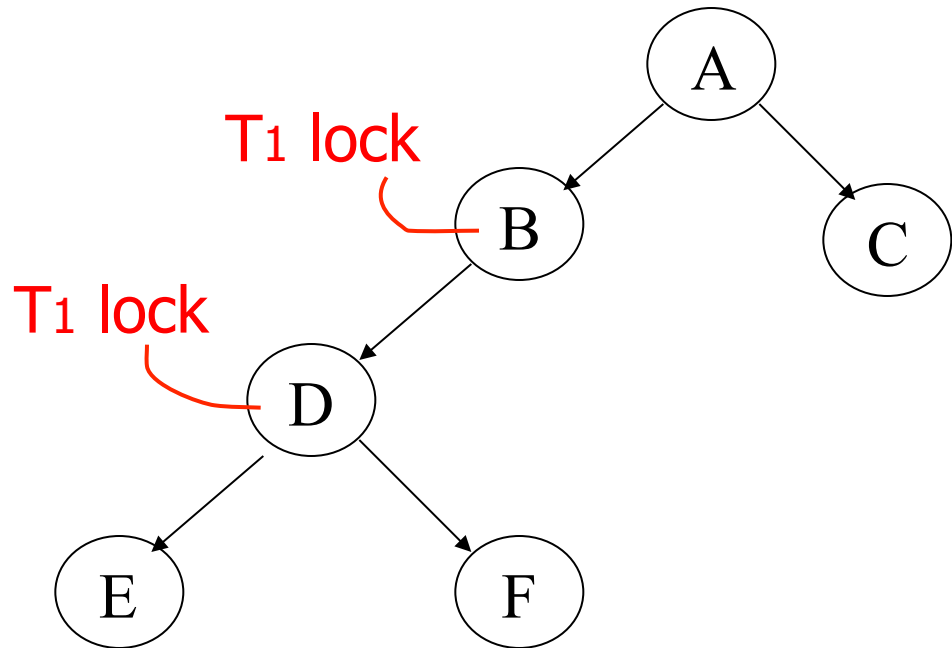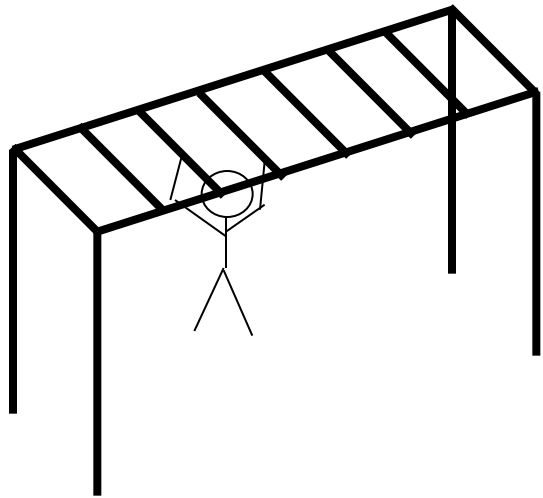
Why does this work?

- Assume all T_i start at root; exclusive lock
- $T_i \rightarrow T_j \Rightarrow T_i$ locks root before T_j

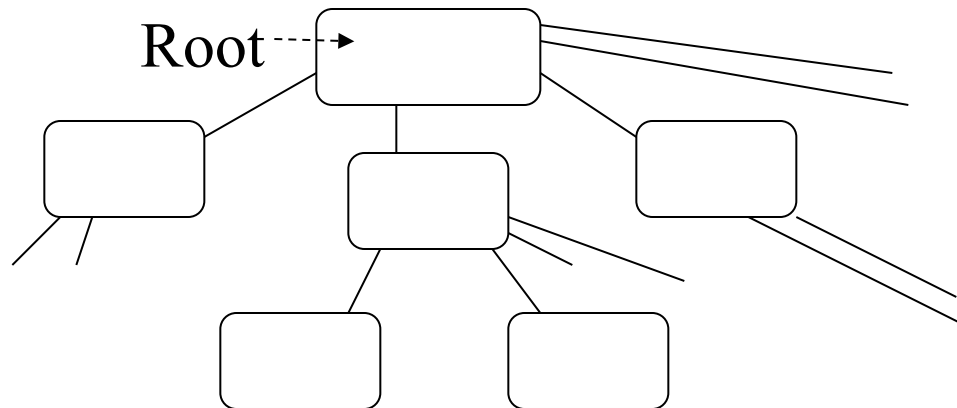- Actually works if we don't always start at root

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: tree protocol (exclusive locks)

(1) First lock by T_i may be on any item

(2) After that, item Q can be locked by T_i only if parent(Q) locked by T_i

(3) Items may be unlocked at any time

(4) After T_i unlocks Q, it cannot relock Q

COMPUTER SCIENCE

IIT College of Science and Letters
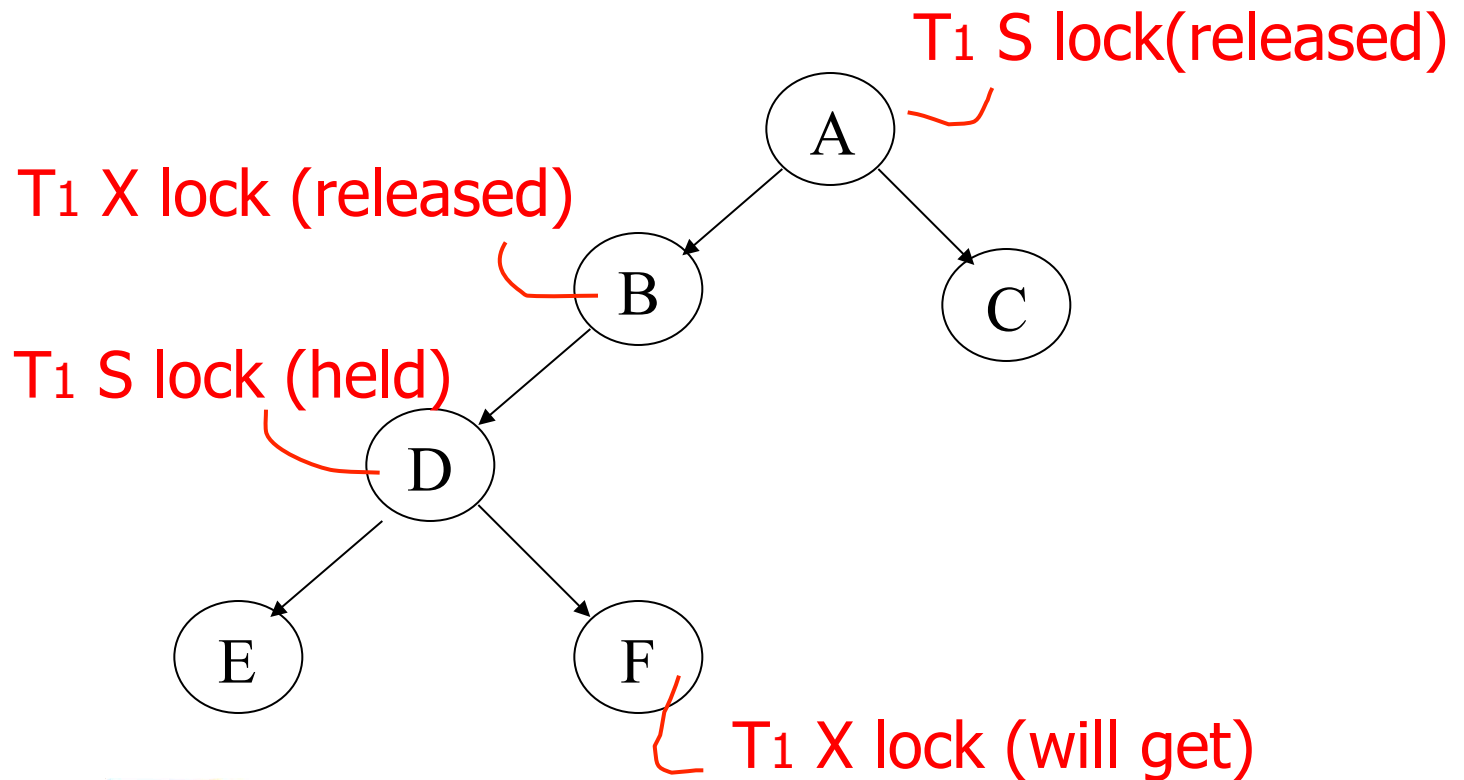ILLINOIS INSTITUTE OF TECHNOLOGY

- Tree-like protocols are used typically for B-tree concurrency control

Root

E.g., during insert, do not release parent lock, until you are certain child does not have to split

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

Tree Protocol with Shared Locks

- Rules for shared & exclusive locks?

T1 S lock(released)

T1 X lock (released)

T1 S lock (held)

T1 X lock (will get)

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Tree Protocol with Shared Locks

- Rules for shared & exclusive locks?

T1 S lock(released)

T1 X lock (released)

T1 S lock (held)

T2 reads:
- B modified by T1
- F not yet modified by T1

T1 X lock (will get)

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
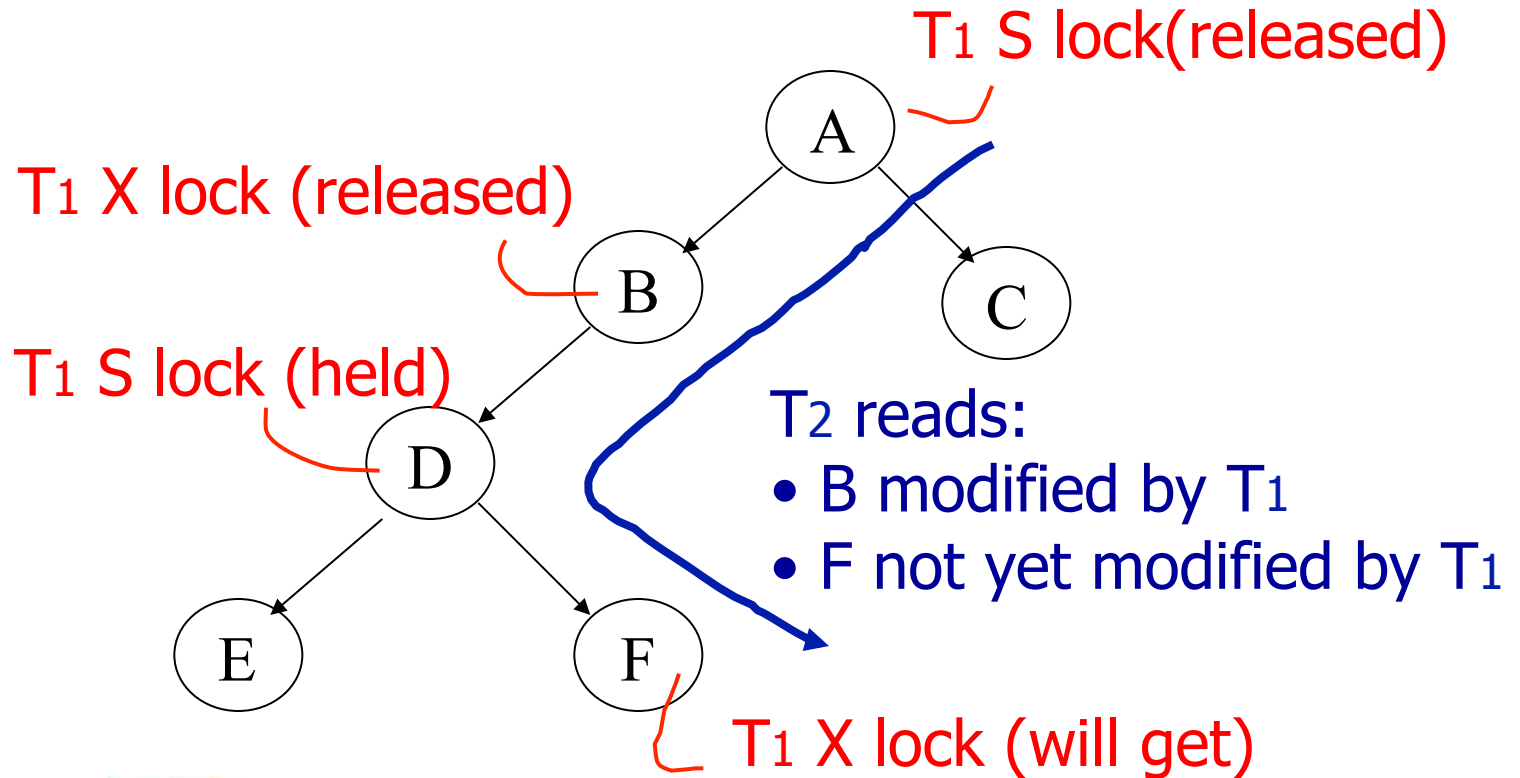
Tree Protocol with Shared Locks

- Need more restrictive protocol

- Will this work??

  - Once T_1 locks one object in X mode, all further locks down the tree must be in X mode

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Deadlocks (again)

- Before we assumed that we are able to detect deadlocks and resolve them
- Now two options
  - (1) Deadlock detection (and resolving)
  - (2) Deadlock prevention

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Deadlock Prevention

- Option 1:
  - 2PL + transaction has to acquire all locks at transaction start following a global order

locks

time

Notes 14 - Concurrency Control
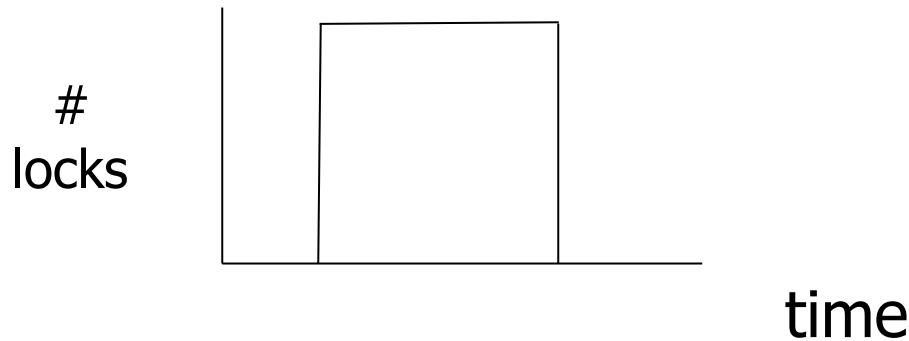
Deadlock Prevention

- Option 1:
  - Long lock durations ☹
  - Transaction has to know upfront what data items it will access ☹
    - E.g.,
    UPDATE R **SET** a = a + 1 **WHERE** b < 15
    - We don't know what tuples are in R!

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Deadlock Prevention

- Option 2:
  - Define some global order of data items O
  - Transactions have to acquire locks according to this order

- Example ($X < Y < Z$)
  $l_1(X), l_1(Z)$ (OK)
  $l_1(Y), l_1(X)$ (NOT OK)

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Deadlock Prevention

- Option 2:
  - Accessed data items have to be known upfront ☹
  - or access to data has to follow the order ☹

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deadlock Prevention

- Option 3 (**Preemption**)
  - Roll-back transactions that wait for locks under certain conditions
  - 3 a) **wait-die**
    - Assign timestamp to each transaction
    - If transaction T_i waits for T_j to release a lock
      - Timestamp $T_i < T_j$ -> wait
      - Timestamp $T_i > T_j$ -> roll-back T_i

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deadlock Prevention

- Option 3 (**Preemption**)
  - Roll-back transactions that wait for locks under certain conditions
  - 3 a) **wound-wait**
    - Assign timestamp to each transaction
    - If transaction T_i waits for T_j to release a lock
      - Timestamp $T_i < T_j$ -> roll-back T_j
      - Timestamp $T_i > T_j$ -> wait

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Deadlock Prevention

- Option 3:
  - Additional transaction roll-backs ☹

Timeout-based Scheme

- Option 4:
  - After waiting for a lock longer than X, a transaction is rolled back

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

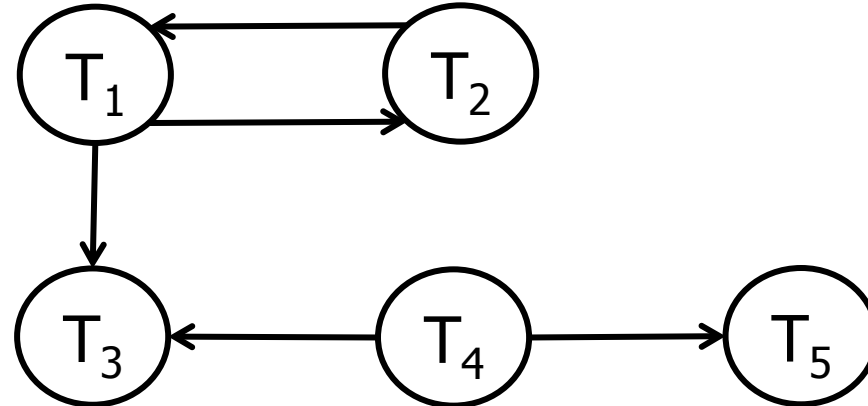# Timeout-based Scheme

- Option 4:
  - Simple scheme ☺
  - Hard to find a good value of X
    - To high: long wait times for a transaction before it gets eventually aborted
    - To low: to many transaction that are not deadlock get aborted

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Deadlock Detection and Resolution

- Data structure to detect deadlocks: **wait-for** graph
  - One node for each transaction
  - Edge $T_i \rightarrow T_j$ if T_i is waiting for T_j
  - Cycle -> Deadlock
    - Abort one of the transaction in cycle to resolve deadlock

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deadlock Detection and Resolution

- When do we run the detection?
- How to choose the victim?

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Optimistic Concurrency Control: Validation

Transactions have 3 phases:

(1) <u>Read</u>
- all DB values read
- writes to temporary storage
- no locking

(2) <u>Validate</u>
- check if schedule so far is serializable

(3) <u>Write</u>
- if validate ok, write to DB
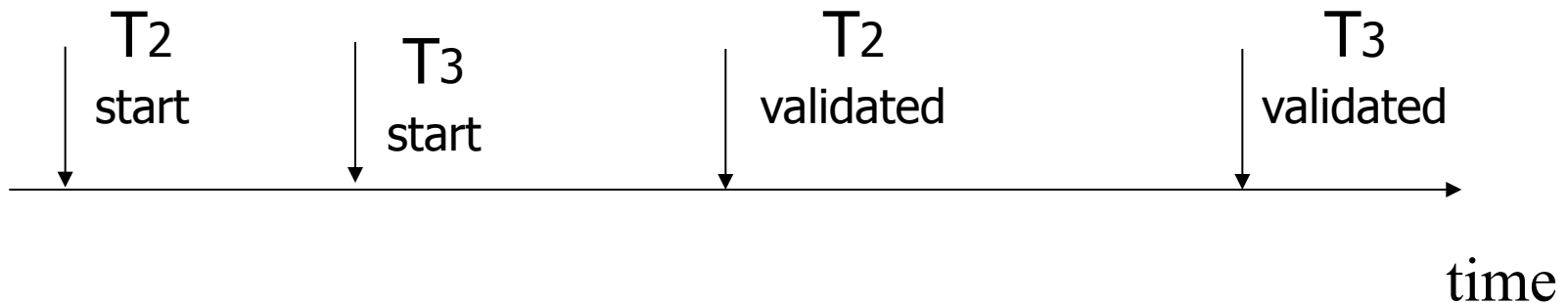
IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Key idea

- Make validation atomic

- If $T_1, T_2, T_3, \ldots$ is validation order, then resulting schedule will be conflict equivalent to $S_s = T_1 \, T_2 \, T_3 \ldots$

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER
SCIENCE

To implement validation, system keeps <u>two sets:</u>

- <u>FIN</u> = transactions that have finished phase 3 (and are all done)

- <u>VAL</u> = transactions that have successfully finished phase 2 (validation)

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example of what validation must prevent:

$$RS(T_2)=\{B\} \quad \cap \quad RS(T_3)=\{A,B\} \neq \phi$$
$$WS(T_2)=\{B,D\} \qquad WS(T_3)=\{C\}$$
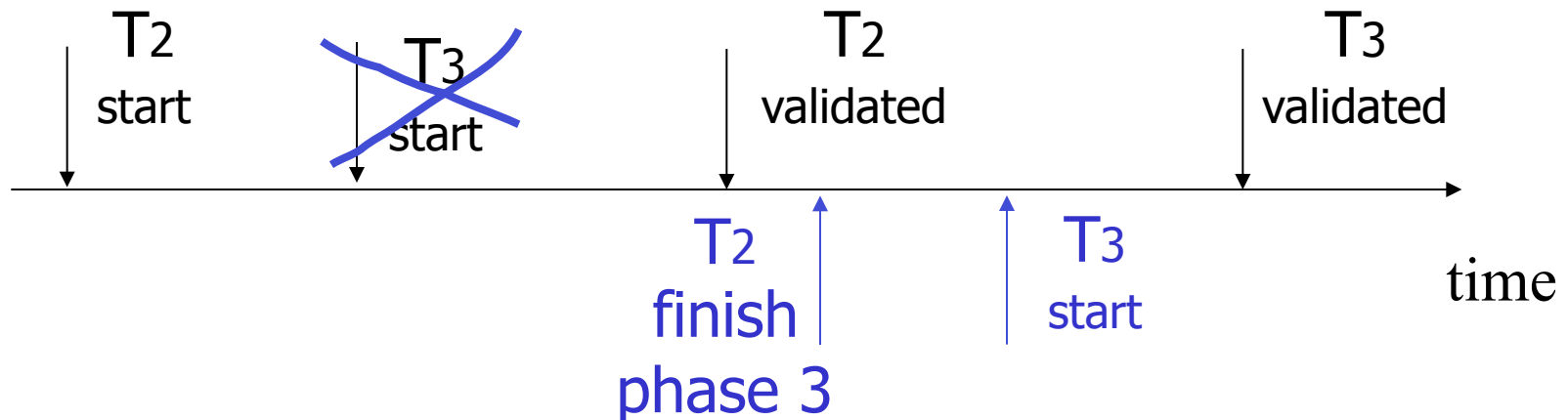
T2
start

T3
start

T2
validated

T3
validated

time

COMPUTER
SCIENCE

Notes 14 - Concurrency Control

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
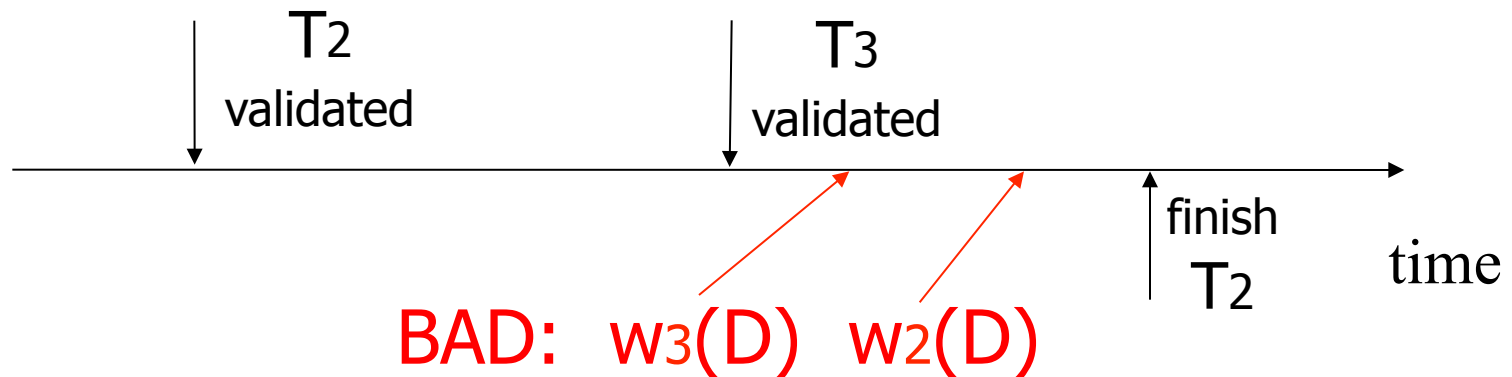
Example of what validation must prevent:

allow

$$RS(T_2)=\{B\} \quad \cap \quad RS(T_3)=\{A,B\} \neq \phi$$
$$WS(T_2)=\{B,D\} \qquad WS(T_3)=\{C\}$$

T2 start

~~T3 start~~

T2 validated

T3 validated

T2 finish phase 3

T3 start

time

Another thing validation must prevent:

$$RS(T_2)=\{A\} \qquad RS(T_3)=\{A,B\}$$
$$WS(T_2)=\{D,E\} \qquad WS(T_3)=\{C,D\}$$

T2
validated

T3
validated

finish
T2
time

Another thing validation must prevent:

RS(T2)={A}   RS(T3)={A,B}

WS(T2)={D,E}   WS(T3)={C,D}

T2 validated

T3 validated

finish T2

time

BAD: w3(D) w2(D)

Another thing validation must prevent: ~~prevent~~ allow

$$RS(T_2)=\{A\} \qquad RS(T_3)=\{A,B\}$$
$$WS(T_2)=\{D,E\} \qquad WS(T_3)=\{C,D\}$$

T2 validated

T3 validated

finish T2

~~finish~~ T2

time

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY
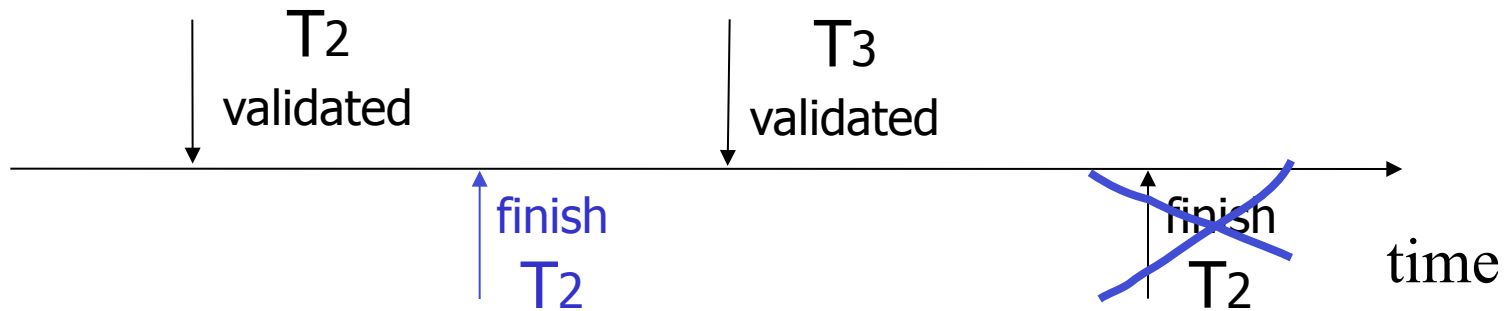
Validation rules for Tj:

(1) When Tj starts phase 1:

ignore(Tj) ← FIN

(2) at Tj Validation:

if check (Tj) then

[VAL ← VAL U {Tj};

do write phase;

FIN ← FIN U {Tj}]

Check (T_j):

$\quad$ For $T_i \in$ VAL - IGNORE (T_j) DO

$\qquad$ IF [WS(T_i) $\cap$ RS(T_j) $\neq \varnothing$ OR

$\quad$ $T_i \notin$ FIN] THEN RETURN false;

RETURN true;

Check (T$_j$):

For T$_i$ $\in$ VAL - IGNORE (T$_j$) DO

IF [WS(T$_i$) $\cap$ RS(T$_j$) $\neq$ $\varnothing$ OR

T$_i$ $\notin$ FIN] THEN RETURN false;

RETURN true;

Is this check too restrictive ?

IIT College of
Science and Letters

COMPUTER
SCIENCE

ILLINOIS INSTITUTE OF TECHNOLOGY

Improving Check(T_j)

For $T_i \in$ VAL - IGNORE (T_j) DO

  IF [WS(T_i) $\cap$ RS(T_j) $\neq \varnothing$ OR

    ($T_i \notin$ FIN AND WS(T_i) $\cap$ WS(T_j) $\neq \varnothing$)]

      THEN RETURN false;

RETURN true;

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Exercise:

Legend: △ start, ⊕ validate, ☆ finish
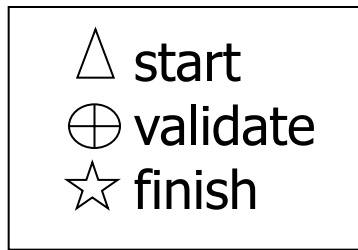
U: RS(U)={B}
    WS(U)={D}

W: RS(W)={A,D}
    WS(W)={A,C}
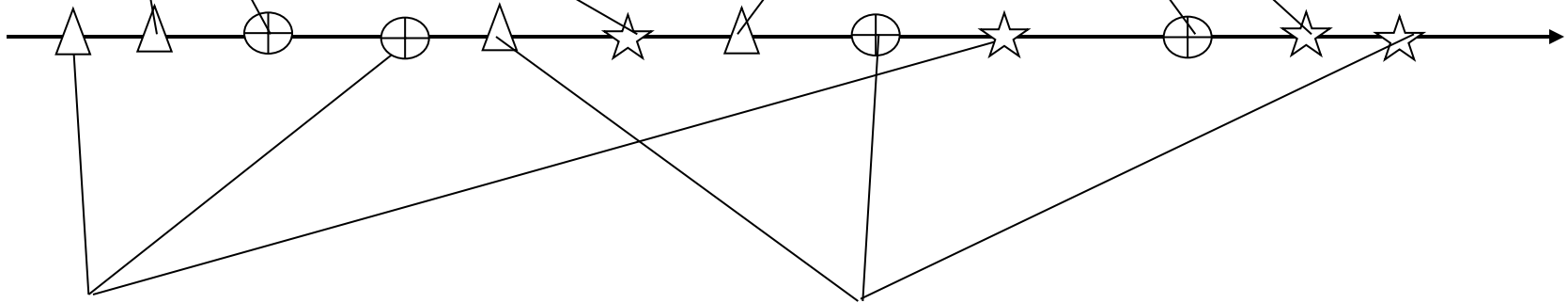
T: RS(T)={A,B}
    WS(T)={A,C}

V: RS(V)={B}
    WS(V)={D,E}

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Is Validation = 2PL?

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY
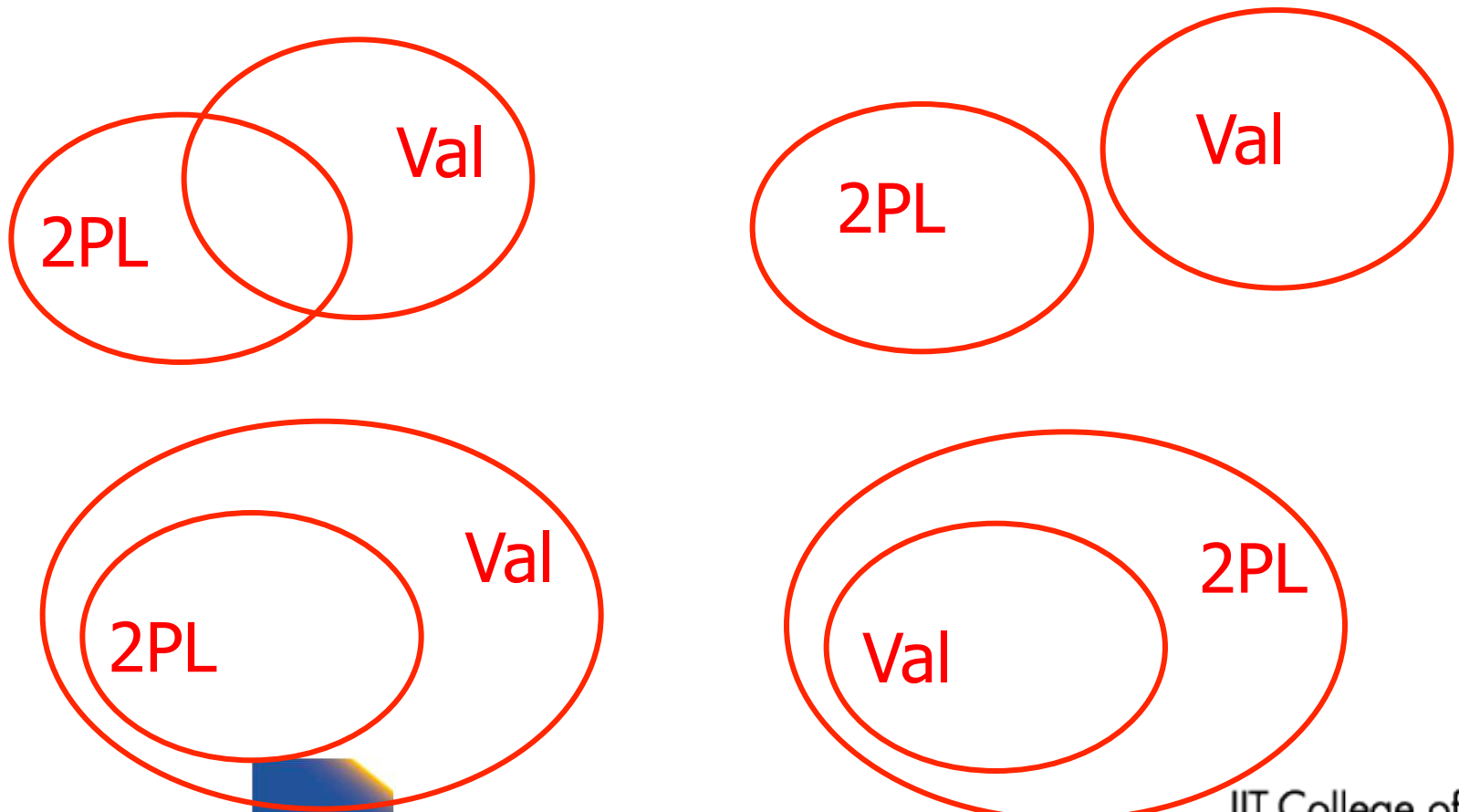
S2: w2(y) w1(x) w2(x)

- S2 can be achieved with 2PL:
  l2(y) w2(y) l1(x) w1(x) u1(x) l2(x) w2(x) u2(y) u2(x)

- S2 cannot be achieved by validation:
  The validation point of T2, val2 must occur before w2(y) since transactions do not write to the database until after validation. Because of the conflict on x, val1 < val2, so we must have something like

     S2: val1 val2 w2(y) w1(x) w2(x)

  With the validation protocol, the writes of T2 should not start until T1 is all done with its writes, which is not the case.

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Validation subset of 2PL?

- Possible proof (Check!):
  - Let S be validation schedule
  - For each T in S insert lock/unlocks, get S':
    - At T start: request read locks for all of RS(T)
    - At T validation: request write locks for WS(T); release read locks for read-only objects
    - At T end: release all write locks
  - Clearly transactions well-formed and 2PL
  - Must show S' is legal (next page)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- Say S' not legal:
  S': ... l1(x)   w2(x) r1(x)   val1 u2(x) ...
  - At val1: T2 not in Ignore(T1); T2 in VAL
  - T1 does not validate: $WS(T2) \cap RS(T1) \neq \varnothing$
  - contradiction!
- Say S' not legal:
  S': ... val1 l1(x)   w2(x) w1(x)   u2(x) ...
  - Say T2 validates first (proof similar in other case)
  - At val1: T2 not in Ignore(T1); T2 in VAL
  - T1 does not validate:
    $T2 \notin FIN$ AND $WS(T1) \cap WS(T2) \neq \varnothing)$
  - contradiction!

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Validation (also called **optimistic concurrency control**) is useful in some cases:

- Conflicts rare

- System resources plentiful

- Have real time constraints

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Multiversioning Concurrency Control (MVCC)

- Keep old versions of data item and use this to increase concurrency

- Each write creates a new version of the written data item

- Use version numbers of timestamps to identify versions

Multiversioning Concurrency Control (MVCC)

- **Different transactions** operate over **different versions** of data items

- -> readers never have to wait for writers

- -> great for combined workloads
  - **OLTP** workload (writes, only access small number of tuples, short)
  - **OLAP** workload (reads, access large portions of database, long running)

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

MVCC schemes

- MVCC timestamp ordering
- MVCC 2PL
- Snapshot isolation (SI)
  - We will only cover this one

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Snapshot Isolation (SI)

- Each transaction **T** is assigned a timestamp **S(T)** when it starts

- Each write creates a new data item version timestamped with the current timestamp

- When a transaction commits, then the latest versions created by the transaction get a timestamp **C(T)** as of the commit

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Snapshot Isolation (SI)

- Under snapshot isolation each transaction T sees a consistent snapshot of the database as of S(T)
  - It only sees data item versions of transactions that committed before T started
  - It also sees its own changes

COMPUTER
SCIENCE

Notes 14 - Concurrency Control

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

First Updater Wins Rule (FUW)

- Two transactions Ti and Tj may update the same data item A
  - To avoid lost updates only one of the two can be safely committed

- **First Updater Wins Rules**
  - The transaction that updated A first is allowed to commit
  - The other transaction is aborted

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

First Committer Wins Rule (FCW)

- Two transactions Ti and Tj may update the same data item A
  - To avoid lost updates only one of the two can be safely committed

- **First Committer Wins Rules**
  - The transaction that attempts to commit first is allowed to commit
  - The other transaction is aborted

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

	X	Y	Z		T1	T2	T3
Update not visible outside of T1 →	0	1	5	1	W(Y := 1)		
Update becomes visible to →	0			2	Commit		
future transactions				3		Start	
				4		R(X) → 0	
				5		R(Y) → 1	
				6			
	2		3	7			W(X:=2)
	2		3	8			W(Z:=3)
				9			Commit
Concurrent updates not visible				10		R(Z) → 5	
				11		R(Y) → 1	
	3			12		W(X:=3)	
Not first-committer of X				13		Commit-Req	
Serialization error, T2 is rolled back				14		Abort	

COMPUTER SCIENCE

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Why does that work?

- Since all transactions see a consistent snapshot and their changes are only made "public" once they commit
  - It looks like the transactions have been executed in the order of their commits*

* Recall the writes to the same data item are disallowed for concurrent transactions

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Is that serializable?

- Almost ;-)
- There is still one type of conflict which cannot occur in serialize schedules called **write-skew**

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

Write Skew

- Consider two data items A and B
  - A = 5, B = 5

- Concurrent Transactions T1 and T2
  - T1: A = A + B
  - T2: B = A + B

- Final result under SI
  - A = 10, B = 10

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Write Skew

- Consider serial schedules:
  - T1, T2: A=10, B=15
  - T2, T1: A=15, B=10

- What is the problem
  - Under SI both T1 and T2 do not see each others changes
  - In any serial schedule one of the two would see the others changes

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Oracle

- Tuples are updated in place
- Old versions in separate ROLLBACK segment
  - GC once nobody needs them anymore
- How to implement the FCW or FUW?
  - Oracle uses write locks to block concurrent writes
  - Transaction waiting for a write lock aborts if transaction holding the lock commits

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE

SI Discussion

- Advantages
  - Readers and writers do not block each other
  - If we do not GC old row versions we can go back to previous versions of the database -> Time travel
    - E.g., show me the customer table as it was yesterday

- Disadvantages
  - Storage overhead to keep old row versions
  - GC overhead
  - Not strictly serializable

COMPUTER SCIENCE

IIT College of
Science and Letters

ILLINOIS INSTITUTE OF TECHNOLOGY

Summary

Have studied CC mechanisms used in practice

- 2 PL variants

- Multiple lock granularity

- Deadlocks

- Tree (index) protocols

- Optimistic CC (Validation)

- Multiversioning Concurrency Control (MVCC)

COMPUTER SCIENCE

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY