

CS 525: Advanced Database Organization



01: Introduction

Boris Glavic

Slides: adapted from a [course](#) taught by
[Hector Garcia-Molina](#), Stanford InfoLab

CS 525



Notes 1 - Introduction

1

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 ;-)

CS 525



Notes 1 - Introduction

2

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Isn't Implementing a Database System Simple?

Relations \Rightarrow Statements \Rightarrow Results

CS 525



Notes 1 - Introduction

3

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Introducing the MEGATRON 3000 Database Management System

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

CS 525



Notes 1 - Introduction

4

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Megatron 3000 Implementation Details

! First sign non-disclosure agreement !

CS 525



Notes 1 - Introduction

5

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
:
```

CS 525



Notes 1 - Introduction

6

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 ...
.
.
.
    
```

CS 525
Notes 1 - Introduction
7
IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Megatron 3000 Sample Sessions

```

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

CS 525
Notes 1 - Introduction
8
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
&
    
```

CS 525
Notes 1 - Introduction
9
IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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
&
    
```

CS 525
Notes 1 - Introduction
10
IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Megatron 3000 Sample Sessions

```

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

Result sent to LPR (printer).

CS 525
Notes 1 - Introduction
11
IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Megatron 3000 Sample Sessions

```

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

New relation T created.

CS 525
Notes 1 - Introduction
12
IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 1 - Introduction

13

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 1 - Introduction

14

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

CS 525



Notes 1 - Introduction

15

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's wrong with the Megatron 3000 DBMS?

CS 525



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

CS 525



Notes 1 - Introduction

17

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 1 - Introduction

18

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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?

CS 525



Notes 1 - Introduction

19

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's wrong with the Megatron 3000 DBMS?

- No buffer manager

e.g., Need caching

CS 525



Notes 1 - Introduction

20

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's wrong with the Megatron 3000 DBMS?

- No concurrency control

CS 525



Notes 1 - Introduction

21

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

CS 525



Notes 1 - Introduction

22

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's wrong with the Megatron 3000 DBMS?

- No security

e.g., - File system insecure

- File system security is coarse

CS 525



Notes 1 - Introduction

23

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?

CS 525



Notes 1 - Introduction

24

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's wrong with the Megatron 3000 DBMS?

- Cannot interact with other DBMSs.

CS 525



Notes 1 - Introduction

25

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's wrong with the Megatron 3000 DBMS?

- Poor dictionary facilities

CS 525



Notes 1 - Introduction

26

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's wrong with the Megatron 3000 DBMS?

- No GUI

CS 525



Notes 1 - Introduction

27

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's wrong with the Megatron 3000 DBMS?

- Lousy salesman!!

CS 525



Notes 1 - Introduction

28

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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,...

CS 525



Notes 1 - Introduction

29

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Course Overview

- **Concurrency Control**
Correctness, locks,...
- **Transaction Processing**
Logs, deadlocks,...
- **Security & Integrity**
Authorization, encryption,...
- **Advanced Topics**
Distribution, More Fancy Optimizations, ...

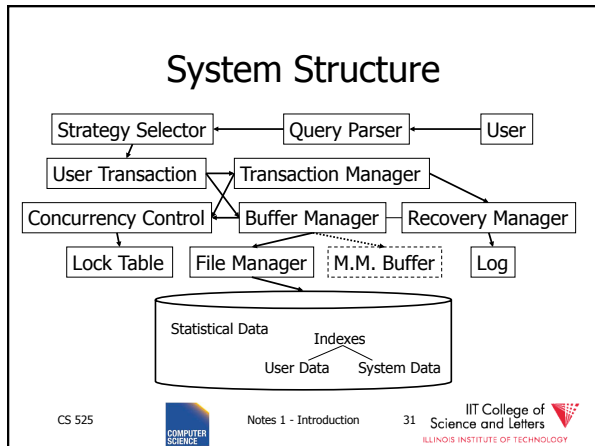
CS 525



Notes 1 - Introduction

30

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY



- ## Some Terms
- Database system
 - Transaction processing system
 - File access system
 - Information retrieval system
- CS 525 Notes 1 - Introduction 32 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- ## 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: Thursdays, 1pm-2pm**
 - **Office:** Stuart Building, Room 226 C
 - **TA:** Ma Di (dma2@hawk.iit.edu)
 - **Time:** Mon + Wed 3:15pm – 4:30pm
- CS 525 Notes 1 - Introduction 33 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- ## Google Group
- <https://groups.google.com/forum/#!forum/cs525-2013-spring-group>
 - Mailing-list for announcements
 - Discussion forum
 - Student - Instructor/TA
 - Student - Student
 - ->please accept invite to keep up to date
- CS 525 Notes 1 - Introduction 34 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- ## 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%)
- CS 525 Notes 1 - Introduction 35 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**, 3rd Edition , McGraw-Hill , 2002
 - Silberschatz, Korth, and Sudarshan , **Database System Concepts**, 6th Edition , McGraw Hill , 2010
- CS 525 Notes 1 - Introduction 36 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 = commit to repository
 - One repository per student
 - You should have gotten an invitation (if not, contact me/TA)
 - Git tutorials linked on course webpage!

CS 525



Notes 1 - Introduction

37

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Next time:

- Hardware

CS 525



Notes 1 - Introduction

38

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

CS 525



Notes 2 - Hardware

1



Outline

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

CS 525

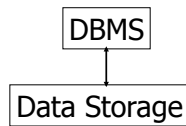


Notes 2 - Hardware

2



Hardware

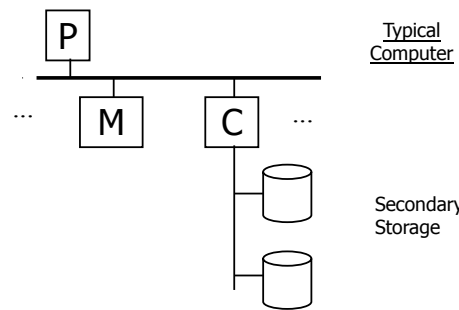


CS 525



Notes 2 - Hardware

3



CS 525



Notes 2 - Hardware

4



Processor

Fast, slow, reduced instruction set, with cache, pipelined...
Speed: 100 → 500 → 1000 MIPS

Memory

Fast, slow, non-volatile, read-only...
Access time: 10^{-6} → 10^{-9} sec.
 $1 \mu\text{s}$ → 1 ns

CS 525



Notes 2 - Hardware

5



Secondary storage

Many flavors:

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

CS 525

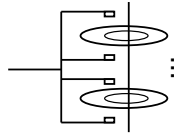


Notes 2 - Hardware

6



Focus on: "Typical Disk"



Terms: Platter, Head, Actuator
Cylinder, Track
Sector (physical),
Block (logical), Gap

CS 525

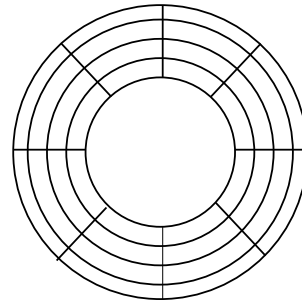


Notes 2 - Hardware

7

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Top View



CS 525



Notes 2 - Hardware

8

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)
→ 500 GB (I use)

CS 525

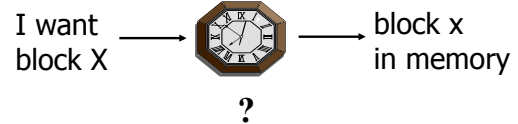


Notes 2 - Hardware

9

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Disk Access Time



CS 525



Notes 2 - Hardware

10

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Time = Seek Time +
Rotational Delay +
Transfer Time +
Other

CS 525

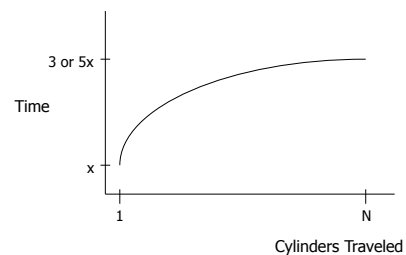


Notes 2 - Hardware

11

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Seek Time



CS 525



Notes 2 - Hardware

12

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Average Random Seek Time

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

CS 525



Notes 2 - Hardware

13

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Average Random Seek Time

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

“Typical” S: 10 ms → 40 ms

CS 525

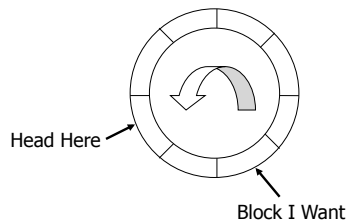


Notes 2 - Hardware

14

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rotational Delay



CS 525



Notes 2 - Hardware

15

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Average Rotational Delay

$R = 1/2$ revolution

“typical” R = 8.33 ms (3600 RPM)

CS 525



Notes 2 - Hardware

16

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Transfer Rate: t

- “typical” t: 1 → 3 MB/second
- transfer time: $\frac{\text{block size}}{t}$

CS 525



Notes 2 - Hardware

17

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

CS 525



Notes 2 - Hardware

18

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

CS 525



Notes 2 - Hardware

19

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

CS 525



Notes 2 - Hardware

20

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 2 - Hardware

21

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

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

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

CS 525



Notes 2 - Hardware

22

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.

CS 525



Notes 2 - Hardware

23

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Cost for Writing similar to Reading

... unless we want to verify!
need to add (full) rotation + $\frac{\text{Block size}}{t}$

CS 525



Notes 2 - Hardware

24

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- To Modify a Block?

CS 525



Notes 2 - Hardware

25

- To Modify a Block?

To Modify Block:

- (a) Read Block
- (b) Modify in Memory
- (c) Write Block
- [(d) Verify?]

CS 525



Notes 2 - Hardware

26

Block Address:

- Physical Device
- Cylinder #
- Surface #
- Sector

CS 525

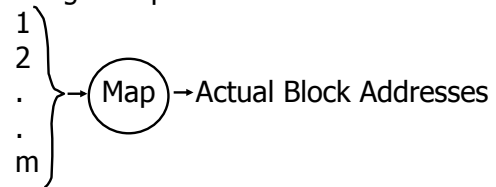


Notes 2 - Hardware

27

Complication: Bad Blocks

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



CS 525



Notes 2 - Hardware

28

An Example Megatron 747 Disk (old)

- 3.5 in diameter
- 3600 RPM
- 1 surface
- 16 MB usable capacity (16×2^{20})
- 128 cylinders
- seek time: average = 25 ms.
adjacent cyl = 5 ms.

CS 525



Notes 2 - Hardware

29

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

CS 525



Notes 2 - Hardware

30

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

One track:



CS 525



Notes 2 - Hardware

31

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

3600 RPM → 60 revolutions / sec
 → 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 = $16.66/128 = 0.13$ ms.

CS 525



Notes 2 - Hardware

32

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Burst Bandwidth

1 KB in 0.117 ms.

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

or

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

CS 525



Notes 2 - Hardware

33

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Sustained bandwidth (over track)

128 KB in 16.66 ms.

$$SB = 128/16.66 = 7.68 \text{ KB/ms}$$

or

$$SB = 7.68 \times 1000/1024 = 7.50 \text{ MB/sec.}$$

CS 525



Notes 2 - Hardware

34

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

T_1 = Time to read one random block

$$T_1 = \text{seek} + \text{rotational delay} + TT$$

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

CS 525

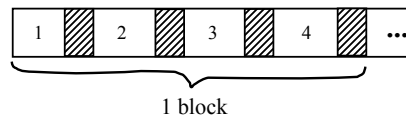


Notes 2 - Hardware

35

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Suppose OS deals with 4 KB blocks



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

[Compare to $T_1 = 33.45$ ms]

CS 525



Notes 2 - Hardware

36

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

$$T_T = 25 + (0.130/2) + 16.66^* = 41.73 \text{ ms}$$

↑
to get to first block

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

CS 525



Notes 2 - Hardware

37

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

The NEW 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

CS 525

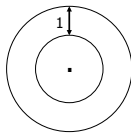


Notes 2 - Hardware

38

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



CS 525



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

CS 525



Notes 2 - Hardware

40

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Timing for new Megatron 747 (Ex 2.3)

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

CS 525



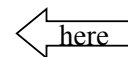
Notes 2 - Hardware

41

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



CS 525



Notes 2 - Hardware

42

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

CS 525



Notes 2 - Hardware

43

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

⋮

CS 525



Notes 2 - Hardware

44

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 ...

CS 525



Notes 2 - Hardware

45

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 = $n(P+R)$

CS 525

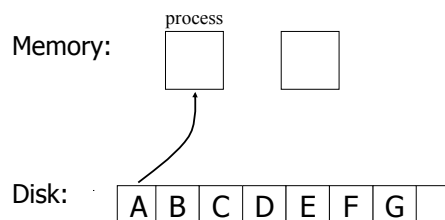


Notes 2 - Hardware

46

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Double Buffering



CS 525

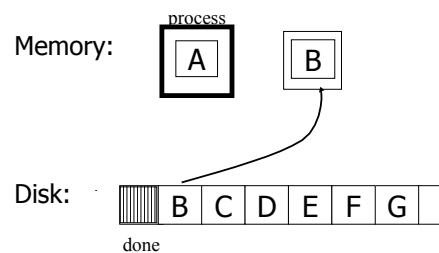


Notes 2 - Hardware

47

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Double Buffering



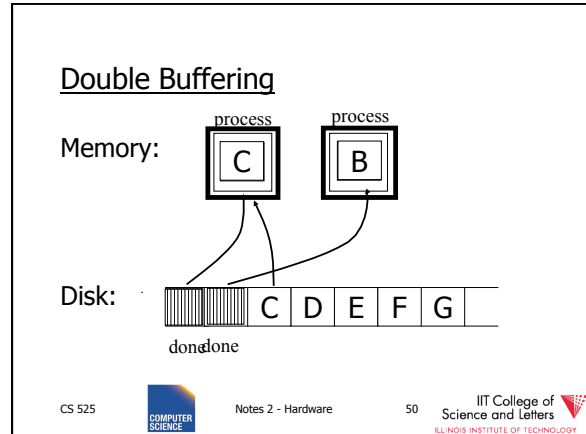
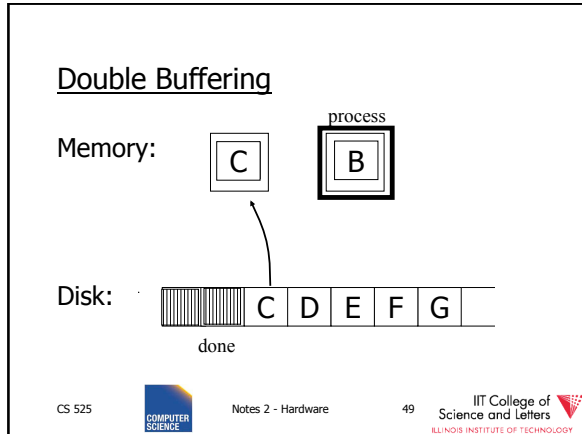
CS 525



Notes 2 - Hardware

48


IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY



Say $P \geq R$

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

What is processing time?


CS 525  Notes 2 - Hardware 51 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

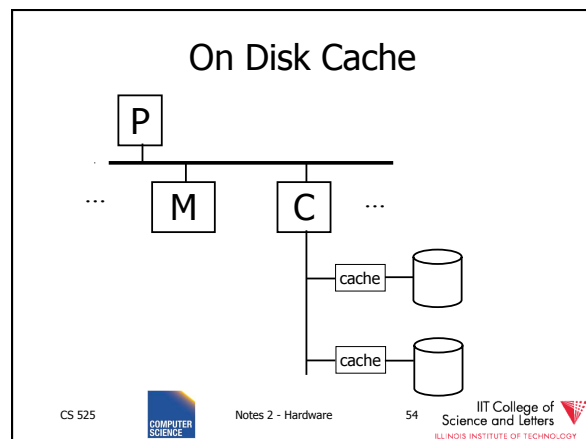
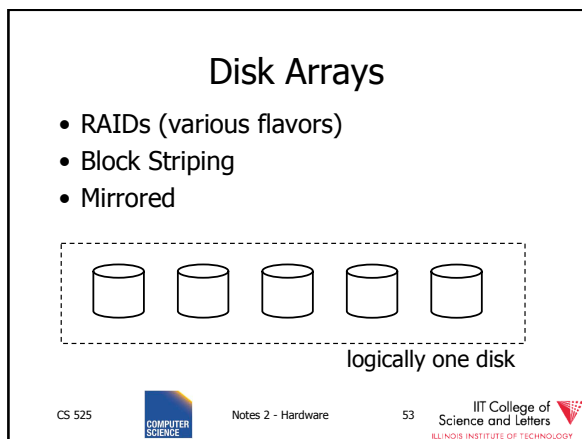
Say $P \geq R$

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

What is processing time?

- Double buffering time = $R + nP$
- Single buffering time = $n(R+P)$

CS 525  Notes 2 - Hardware 52 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

CS 525



Notes 2 - Hardware

55

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Trend

- As memory prices drop, blocks get bigger ...

CS 525

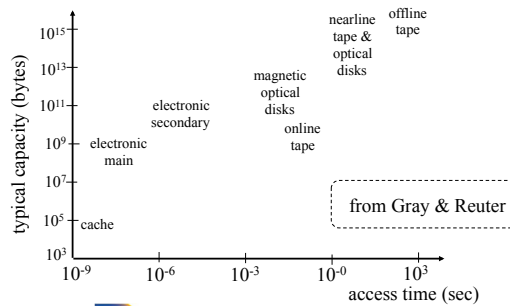


Notes 2 - Hardware

56

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Storage Cost



CS 525

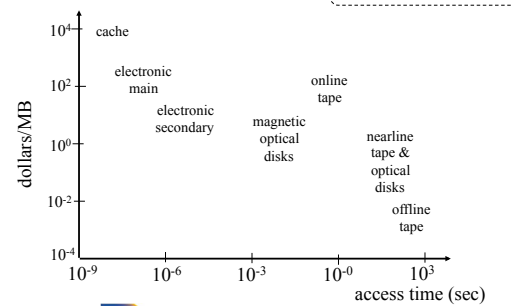


Notes 2 - Hardware

57

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Storage Cost



CS 525



Notes 2 - Hardware

58

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?

CS 525



Notes 2 - Hardware

59

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 2 - Hardware

60

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
 - In X seconds, unit can do XI IOs
 - So CD = \$D / XI

CS 525



Notes 2 - Hardware

61

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

CS 525



Notes 2 - Hardware

62

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

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

CS 525



Notes 2 - Hardware

63

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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)

CS 525



Notes 2 - Hardware

64

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Disk Failures

- Partial → Total
- Intermittent → Permanent

CS 525



Notes 2 - Hardware

65

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Coping with Disk Failures

- Detection
 - e.g. Checksum
- Correction
 - ⇒ Redundancy

CS 525



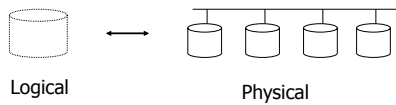
Notes 2 - Hardware

66

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



CS 525

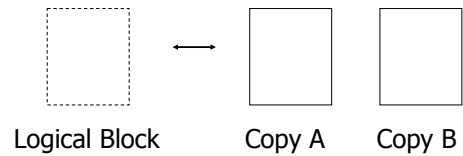


Notes 2 - Hardware

67

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

→ Operating System e.g., Stable Storage



CS 525



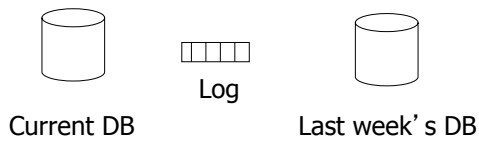
Notes 2 - Hardware

68

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

→ Database System

- e.g.,



CS 525



Notes 2 - Hardware

69

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Summary

- Secondary storage, mainly disks
- I/O times
- I/Os should be avoided,
especially random ones.....

CS 525



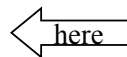
Notes 2 - Hardware

70

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



CS 525



Notes 2 - Hardware

71

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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, ...

CS 525

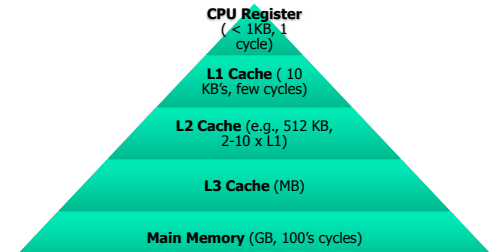


Notes 2 - Hardware

72

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Memory Hierarchy



CS 525



Notes 2 - Hardware

73

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Memory Hierarchy

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

CS 525



Notes 2 - Hardware

74

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Increasing Amount of Parallelism

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

CS 525



Notes 2 - Hardware

75

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525




Notes 2 - Hardware

76

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

CS 525

Notes 3
1




Topics for today

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

CS 525

Notes 3
2


What are the data items we want to store?


- a salary
- a name
- a date
- a picture



CS 525

Notes 3
3


What are the data items we want to store?

- a salary
- a name
- a date
- a picture

⇒ What we have available: Bytes



CS 525

Notes 3
4


To represent:

- Integer (short): 2 bytes
e.g., 35 is

00000000


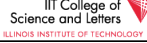
00100011

Endian! Could as well be

00100011

00000000

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



CS 525

Notes 3
5


To represent:

- Characters
→ various coding schemes suggested,
most popular is ASCII (1 byte encoding)

Example:

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

CS 525

Notes 3
6


To represent:

- Boolean
e.g., TRUE

1111	1111
------	------

FALSE

0000	0000
------	------
- Application specific
e.g., enumeration
RED → 1 GREEN → 3
BLUE → 2 YELLOW → 4 ...

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 YYYYMMDD! Why?)
- Time
e.g. - Integer, seconds since midnight
- characters, HHMMSSFF

To represent:

- String of characters
- Null terminated
e.g.,

c	a	t	⊗		
---	---	---	---	--	--
- Length given
e.g.,

3	c	a	t	⊗	
---	---	---	---	---	--
- Fixed length

To represent:

- Bag of bits


Length	Bits
--------	------

Key Point

- Fixed length items
- Variable length items
- usually length given at beginning


Also

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

CS 525  Notes 3 13 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY


Overview

Data Items
↓
Records
↓
Blocks
↓
Files
⋮
Memory

CS 525  Notes 3 14 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY


Record - Collection of related data items (called FIELDS)

E.g.: Employee record:
name field,
salary field,
date-of-hire field, ...

CS 525  Notes 3 15 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Types of records:


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

CS 525  Notes 3 16 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Fixed format

A SCHEMA (not record) contains following information

- # fields
- type of each field
- order in record
- meaning of each field

CS 525  Notes 3 17 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

CS 525  Notes 3 18 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Variable format

- Record itself contains format
“Self Describing”

CS 525

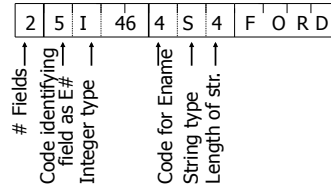


Notes 3

19

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example: variable format and length



Field name codes could also be strings, i.e. TAGS

CS 525



Notes 3

20

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Variable format useful for:

- “sparse” records
- repeating fields
- evolving formats

.....→ But may waste space...

CS 525



Notes 3

21

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- EXAMPLE:** var format record with repeating fields
Employee → one or more → children

3	E_name: Fred	Child: Sally	Child: Tom
---	--------------	--------------	------------

CS 525



Notes 3

22

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Note: Repeating fields does not imply

- variable format, nor
- variable size

John	Sailing	Chess	--
------	---------	-------	----

CS 525



Notes 3

23

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 → null)

CS 525



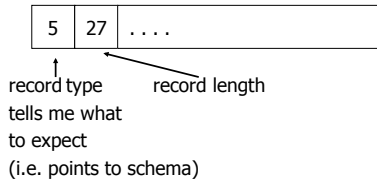
Notes 3

24

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

☆ Many variants between fixed - variable format:

Example: Include record type in record



Record header - data at beginning that describes record

May contain:

- record type
- record length
- time stamp
- null-value bitmap
- other stuff ...

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

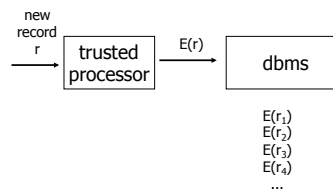
Record Header – null-map

- 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

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

Encrypting Records



Encrypting Records

$E(r_1)$
 $E(r_2)$
 $E(r_3)$
 $E(r_4)$
 ...

CS 525 COMPUTER SCIENCE Notes 3 31 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Search Key in the Clear

$[1, E(b_1)]$
 $[2, E(b_2)]$
 $[3, E(b_3)]$
 $[4, E(b_4)]$
 ...

- each record is $[k, b]$
- store $[k, E(b)]$
- can search for records with $k=x$

CS 525 COMPUTER SCIENCE Notes 3 32 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Encrypt Key

$[E(1), E(b_1)]$
 $[E(2), E(b_2)]$
 $[E(3), E(b_3)]$
 $[E(4), E(b_4)]$
 ...

- each record is $[k, b]$
- store $[E(k), E(b)]$
- can search for records with $k=E(x)$

CS 525 COMPUTER SCIENCE Notes 3 33 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Issues

- Hard to do range queries
- Encryption not good
- Better to use encryption that does not always generate same cyphertext

CS 525 COMPUTER SCIENCE Notes 3 34 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

How Do We Search Now?

$[E(1, abc), E(b_1)]$
 $[E(2, dhe), E(b_2)]$
 $[E(3, nft), E(b_3)]$
 $[E(2, lkz), E(b_4)]$
 ...

- each record is $[k, b]$
- store $[E(k, rand), E(b)]$
- can search for records with $k=E(x, ???)$

CS 525 COMPUTER SCIENCE Notes 3 35 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

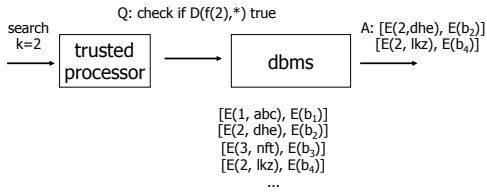
Solution?

- Develop new decryption function:
 $D(f(k_1), E(k_2, rand))$ is true if $k_1=k_2$

CS 525 COMPUTER SCIENCE Notes 3 36 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Solution?

- Develop new decryption function:
 $D(f(k_1), E(k_2, \text{rand}))$ is true if $k_1 = k_2$



CS 525



Notes 3

37

IIT College of Science and Letters
 ILLINOIS INSTITUTE OF TECHNOLOGY

Issues?

- Cannot do non-equality predicates
- Hard to build indexes

CS 525

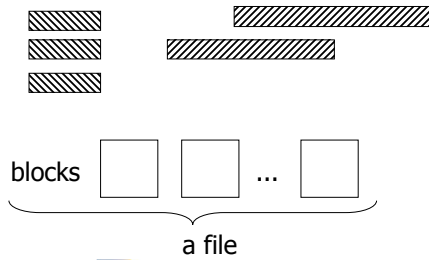


Notes 3

38

IIT College of Science and Letters
 ILLINOIS INSTITUTE OF TECHNOLOGY

Next: placing records into blocks



CS 525

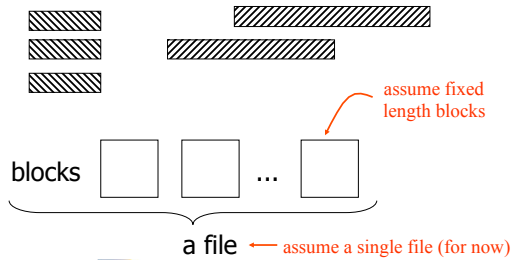


Notes 3

39

IIT College of Science and Letters
 ILLINOIS INSTITUTE OF TECHNOLOGY

Next: placing records into blocks



CS 525



Notes 3

40

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

CS 525



Notes 3

41

IIT College of Science and Letters
 ILLINOIS INSTITUTE OF TECHNOLOGY

(1) Separating records



- no need to separate - fixed size recs.
- special marker
- give record lengths (or offsets)
 - within each record
 - in block header

CS 525



Notes 3

42

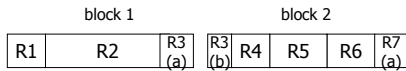
IIT College of Science and Letters
 ILLINOIS INSTITUTE OF TECHNOLOGY

(2) Spanned vs. Unspanned

- Unspanned: records must be within one block



- Spanned



CS 525

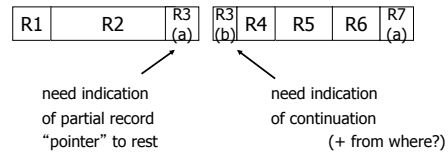


Notes 3

43

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

With spanned records:



CS 525



Notes 3

44

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Spanned vs. unspanned:

- Unspanned is much simpler, but may waste space...
- Spanned essential if record size > block size

CS 525



Notes 3

45

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(3) Sequencing

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

Sequential file (\Rightarrow sequenced)

CS 525



Notes 3

46

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Why sequencing?

Typically to make it possible to efficiently read records in order
(e.g., to do a merge-join — discussed later)

CS 525



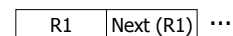
Notes 3

47

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Sequencing Options

(a) Next record physically contiguous



(b) Linked



CS 525



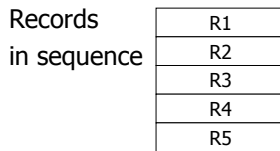
Notes 3

48

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Sequencing Options

(c) Overflow area



CS 525



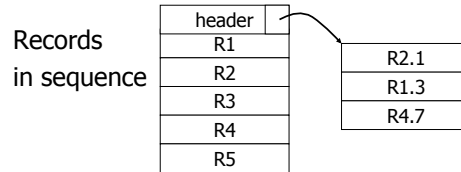
Notes 3

49

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Sequencing Options

(c) Overflow area



CS 525



Notes 3

50

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(4) Indirection

- How does one refer to records?



CS 525



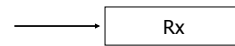
Notes 3

51

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(4) Indirection

- How does one refer to records?



Many options:

Physical ↔ Indirect

CS 525

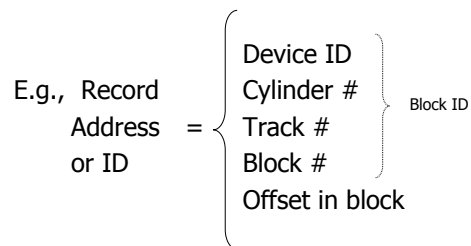


Notes 3

52

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

☆ Purely Physical



CS 525



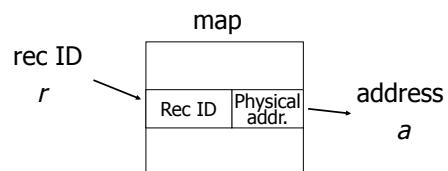
Notes 3

53

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

☆ Fully Indirect

E.g., Record ID is arbitrary bit string



CS 525



Notes 3

54

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Tradeoff

Flexibility \longleftrightarrow Cost
to move records of indirection
(for deletions, insertions)

CS 525



Notes 3

55

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Physical \longleftrightarrow Indirect
 \uparrow
Many options
in between ...

CS 525

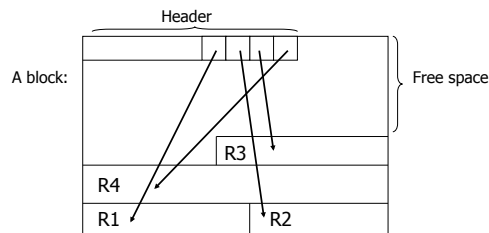


Notes 3

56

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Indirection in block



CS 525



Notes 3

57

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 3

58

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 point to record
 - Replace record with pointer (new TID)
 - Case 2: TID points to pointer (TID)
 - Replace pointer with new pointer

CS 525



Notes 3

59

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 3


60

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Block header - 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 ...

CS 525  Notes 3 61 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Options for storing records in blocks:

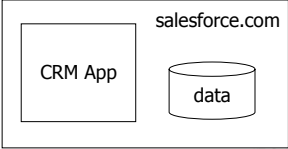
- (1) separating records
- (2) spanned vs. unspanned
- (3) sequencing
- (4) indirection


CS 525  Notes 3 62 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Case Study: salesforce.com

- salesforce.com provides CRM services
- salesforce customers are *tenants*
- Tenants run apps and DBMS as service


tenant A
tenant B
tenant C



CS 525  Notes 3 63 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Options for Hosting

- Separate DBMS per tenant
- One DBMS, separate tables per tenant
- One DBMS, shared tables

CS 525  Notes 3 64 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


Tenants have similar data

tenant 1:

customer	A	B	C	D	E	F
a1	b1	c1	d1	e1	-	
a2	b2	c2	-	e2	f2	

tenant 2:

customer	A	B	C	D	G
a3	b3	c2	-	-	
a1	b1	c1	-	g1	
a4	-	-	d1		

CS 525  Notes 3 65 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


salesforce.com solution

customer	tenant	A	B	C
1	a1	b1	c1	
1	a2	b2	c2	
2	a3	b3	c2	
2	a1	b1	c1	

← fixed schema for all tenants


cust-other	tenant	A	f1	v1	f2	v2 ...
1	a1	D	d1	E	e1	
1	a2	E	e2	F	f2	
2	a1	G	g1			
3	a4	D	d1			

← var schema for all tenants

CS 525  Notes 3 66 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

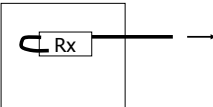
Other Topics


- (1) Insertion/Deletion
- (2) Buffer Management
- (3) Comparison of Schemes

CS 525  Notes 3 67 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion


Block



CS 525  Notes 3 68 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY


Options:

- (a) Immediately reclaim space
- (b) Mark deleted

CS 525  Notes 3 69 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY


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

CS 525  Notes 3 70 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY


☆ As usual, many tradeoffs...


- How expensive is to move valid record to free space for immediate reclaim?
- How much space is wasted?
 - e.g., deleted records, delete fields, free space chains,...

CS 525  Notes 3 71 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY


Concern with deletions

Dangling pointers




CS 525  Notes 3 72 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Solution #1: Do not worry

CS 525  Notes 3 73 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Solution #2: Tombstones

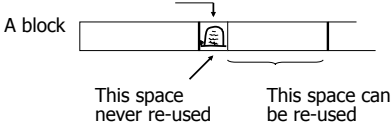
E.g., Leave “MARK” in map or old location

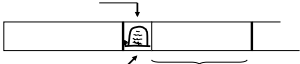
CS 525  Notes 3 74 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Solution #2: Tombstones


E.g., Leave “MARK” in map or old location

- Physical IDs



A block 

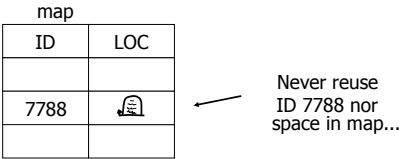
This space never re-used This space can be re-used

CS 525  Notes 3 75 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...

CS 525  Notes 3 76 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...

CS 525  Notes 3 77 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...

CS 525  Notes 3 78 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?

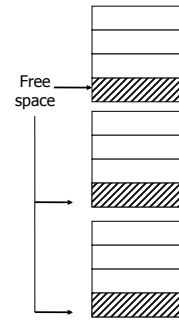
CS 525



Notes 3

79

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY



CS 525



Notes 3

80

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Buffer Management

- DB features needed
- Buffer Replacement Strategies
 - E.g., LRU, clock
- Pinned blocks
- Forced output -----> in Notes02
- Double buffering
- Swizzling

CS 525



Notes 3

81

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 3

82

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

CS 525



Notes 3

83

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

CS 525



Notes 3

84

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

FIFO

- **First In, First Out**
- Replace page that has been in the buffer for the longest time
- Implementation: E.g., pointer to oldest page (circular buffer)
 - $\text{Pointer} \rightarrow \text{next} = \text{Pointer}++ \% M$
- Simple, but not prioritizing frequently accessed pages

CS 525



Notes 3

85

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

CS 525



Notes 3

86

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 \rightarrow bit = 1
- Find page to replace (advance pointer)
 - Return first frame with bit = 0
 - On the way set all bits to 0

CS 525



Notes 3

87

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Clock Example

Reference bit

S → 0	Page 0
1	Page 1
1	Page 2
0	Page 3
1	Page 4

CS 525



Notes 3

88

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Other Replacement Strategies

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

CS 525

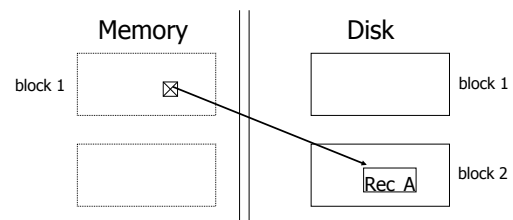


Notes 3

89

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Swizzling



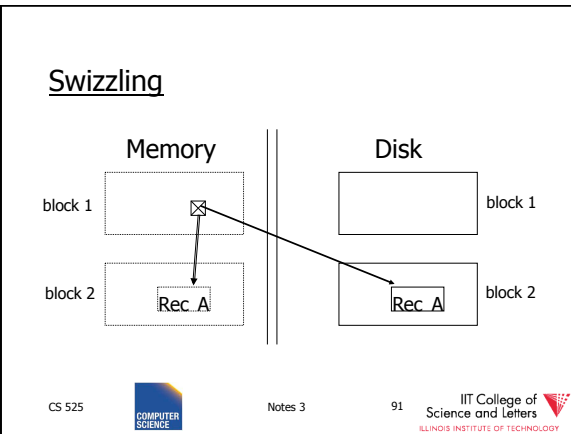
CS 525



Notes 3

90

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 (row store)...
- Another option is to store like fields together (column store)

CS 525 Notes 3 92 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

CS 525 Notes 3 93 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	id1	prod1	id1	price1	qty1
id2	cust2	id2	prod2	id2	price2	qty2
id3	cust3	id3	prod3	id3	price3	qty3
id4	cust4	id4	prod4	id4	price4	qty4
...

ids may or may not be stored explicitly

CS 525 Notes 3 94 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 reads on data mining operations
- Advantages of Row Store
 - writes (multiple fields of one record) more efficient
 - efficient reads for record access (OLTP)

CS 525 Notes 3 95 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Comparison

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


Which is right for me?

CS 525 Notes 3 96 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Issues:


Flexibility ——— Space Utilization

Complexity ——— Performance

CS 525  Notes 3 97 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 all file
 - reorganize file

CS 525  Notes 3 98 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

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?

CS 525  Notes 3 99 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

CS 525  Notes 3 100 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Next


How to find a record quickly,
given a key

CS 525  Notes 3 101 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY



CS 525: Advanced Database Organization

04: Indexing

Boris Glavic



Slides: adapted from a [course](#) taught by [Hector Garcia-Molina](#), Stanford InfoLab

CS 525

Notes 4 - Indexing
1


Part 04

Indexing & Hashing

value → (?) → record

value



→

?

→



record

value

CS 525

Notes 4 - Indexing
2




Topics

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

CS 525

Notes 4 - Indexing
3


Sequential File

10	
20	
30	
40	
50	
60	
70	
80	
90	
100	

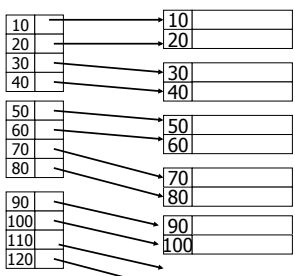
CS 525

Notes 4 - Indexing
4



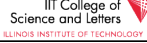
Dense Index

10
20
30
40
50
60
70
80
90
100
110
120

Sequential File

10	
20	
30	
40	
50	
60	
70	
80	
90	
100	



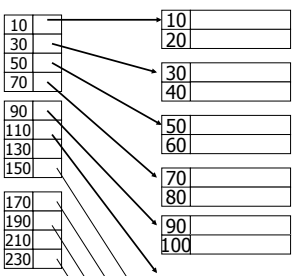
CS 525

Notes 4 - Indexing
5




Sparse Index

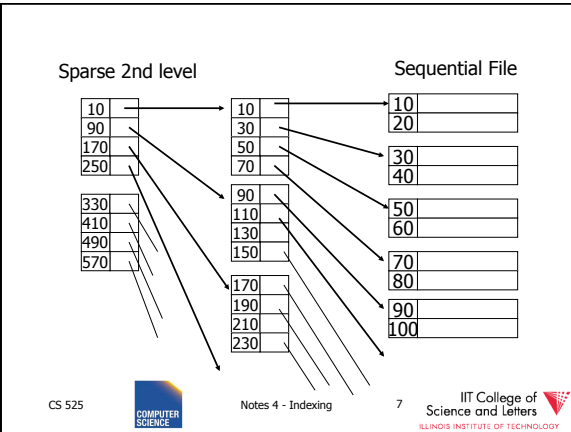
10
30
50
70
90
110
130
150
170
190
210
230

Sequential File

10	
20	
30	
40	
50	
60	
70	
80	
90	
100	



CS 525

Notes 4 - Indexing
6




• Comment:
 {FILE,INDEX} may be contiguous
 or not (blocks chained)

CS 525 Notes 4 - Indexing 8 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Question:

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

CS 525 Notes 4 - Indexing 9 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Notes on pointers:

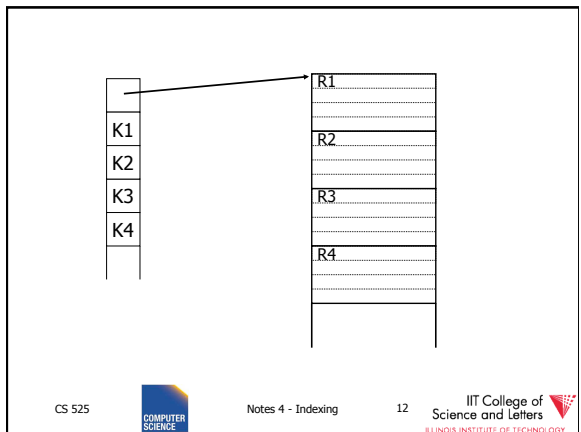
(1) Block pointer (sparse index) can be smaller than record pointer

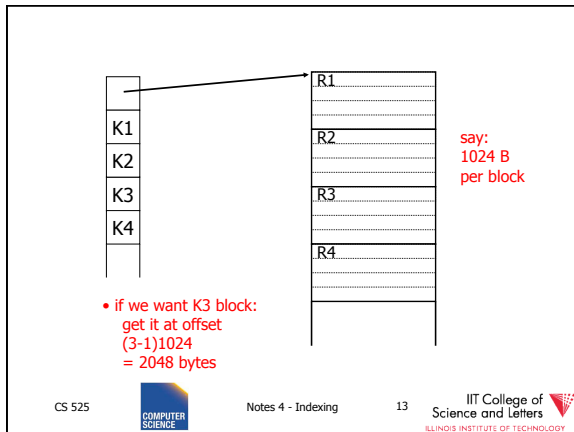
CS 525 Notes 4 - Indexing 10 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Notes on pointers:

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

CS 525 Notes 4 - Indexing 11 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY





CS 525



Notes 4 - Indexing

13

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Sparse vs. Dense Tradeoff

- Sparse:** Less index space per record can keep more of index in memory
- Dense:** Can tell if any record exists without accessing file

(Later:

- sparse better for insertions
- dense needed for secondary indexes)

CS 525



Notes 4 - Indexing

14

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

CS 525



Notes 4 - Indexing

15

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Next:

- Duplicate keys
- Deletion/Insertion
- Secondary indexes

CS 525

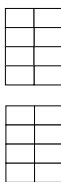


Notes 4 - Indexing

16

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate keys



10	
10	
10	
20	
20	
30	
30	
40	
45	

CS 525



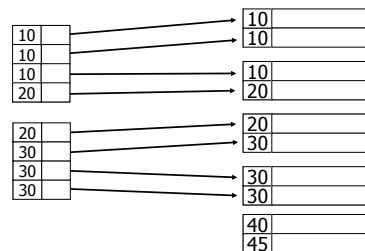
Notes 4 - Indexing

17

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate keys

Dense index, one way to implement?



CS 525



Notes 4 - Indexing

18

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate keys

Dense index, better way?

CS 525 Notes 4 - Indexing 19 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate keys

Sparse index, one way?

CS 525 Notes 4 - Indexing 20 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate keys

Sparse index, one way?

careful if looking for 20 or 30!

CS 525 Notes 4 - Indexing 21 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate keys

Sparse index, another way?

- place first new key from block

CS 525 Notes 4 - Indexing 22 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate keys

Sparse index, another way?

- place first new key from block

should this be 40?

CS 525 Notes 4 - Indexing 23 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

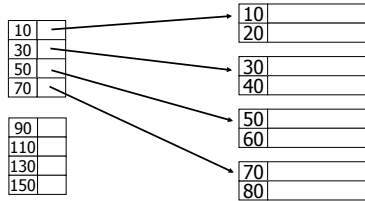
Summary

Duplicate values, primary index

- Index may point to first instance of each value only

CS 525 Notes 4 - Indexing 24 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from sparse index



CS 525



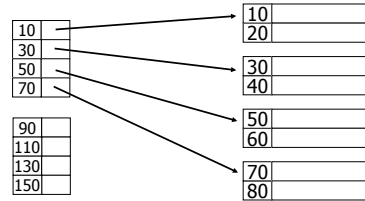
Notes 4 - Indexing

25

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from sparse index

- delete record 40



CS 525



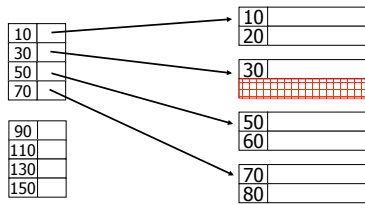
Notes 4 - Indexing

26

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from sparse index

- delete record 40



CS 525



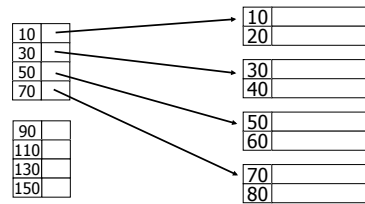
Notes 4 - Indexing

27

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from sparse index

- delete record 30



CS 525



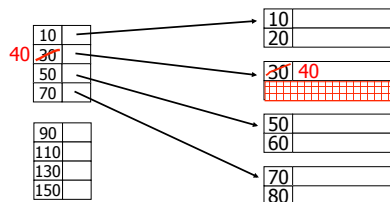
Notes 4 - Indexing

28

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from sparse index

- delete record 30



CS 525



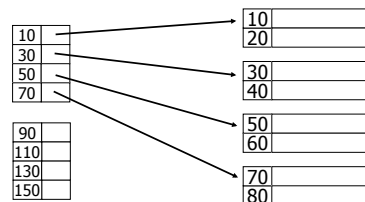
Notes 4 - Indexing

29

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from sparse index

- delete records 30 & 40



CS 525





Notes 4 - Indexing

30

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY



Deletion from sparse index

- delete records 30 & 40



CS 525  Notes 4 - Indexing 31  IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from sparse index

- delete records 30 & 40



CS 525  Notes 4 - Indexing 32  IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from dense index

CS 525  Notes 4 - Indexing 33  IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY



Deletion from dense index

- delete record 30

CS 525  Notes 4 - Indexing 34  IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY



Deletion from dense index

- delete record 30

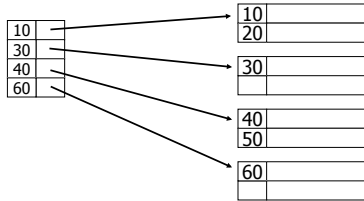
CS 525  Notes 4 - Indexing 35  IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from dense index

- delete record 30

CS 525  Notes 4 - Indexing 36  IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Insertion, sparse index case



CS 525



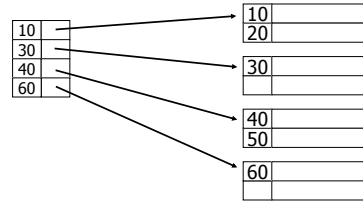
Notes 4 - Indexing

37

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Insertion, sparse index case

- insert record 34



CS 525



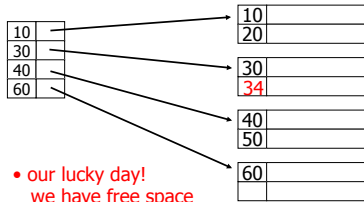
Notes 4 - Indexing

38

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Insertion, sparse index case

- insert record 34



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

CS 525



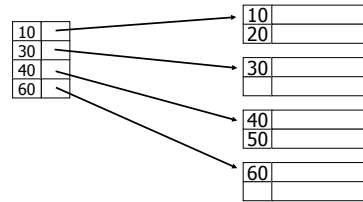
Notes 4 - Indexing

39

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Insertion, sparse index case

- insert record 15



CS 525



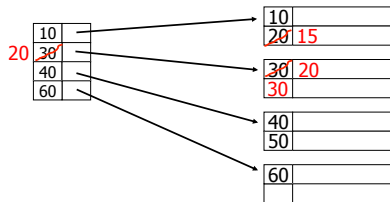
Notes 4 - Indexing

40

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Insertion, sparse index case

- insert record 15



CS 525



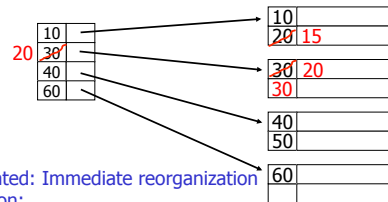
Notes 4 - Indexing

41

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

CS 525



Notes 4 - Indexing

42

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Insertion, sparse index case

- insert record 25

CS 525 Notes 4 - Indexing 43 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Insertion, sparse index case

- insert record 25

CS 525 Notes 4 - Indexing 44 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Insertion, dense index case

- Similar
- Often more expensive . . .

CS 525 Notes 4 - Indexing 45 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Secondary indexes

CS 525 Notes 4 - Indexing 46 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Secondary indexes

- Sparse index

CS 525 Notes 4 - Indexing 47 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

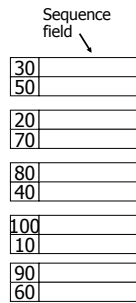
Secondary indexes

- Sparse index

CS 525 Notes 4 - Indexing 48 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Secondary indexes

- Dense index



CS 525



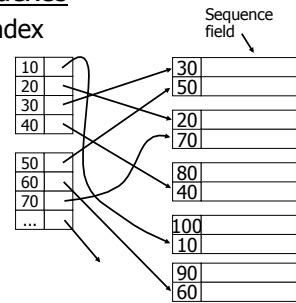
Notes 4 - Indexing

49

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Secondary indexes

- Dense index



CS 525



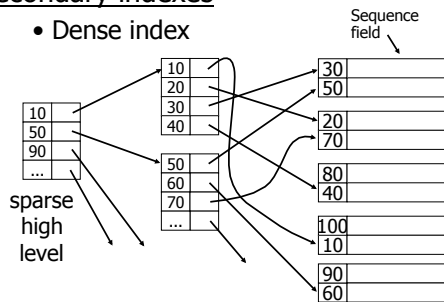
Notes 4 - Indexing

50

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Secondary indexes

- Dense index



CS 525



Notes 4 - Indexing

51

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

With secondary indexes:

- Lowest level is dense
- Other levels are sparse

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

CS 525

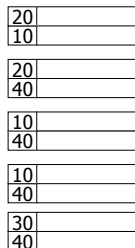


Notes 4 - Indexing

52

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate values & secondary indexes



CS 525



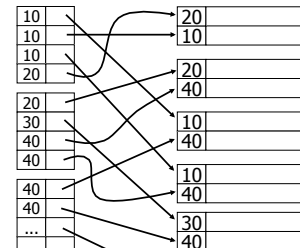
Notes 4 - Indexing

53

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate values & secondary indexes

one option...



CS 525



Notes 4 - Indexing

54

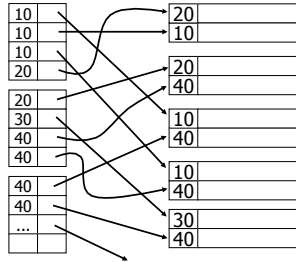
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate values & secondary indexes

one option...

Problem:
excess overhead!

- disk space
- search time



CS 525



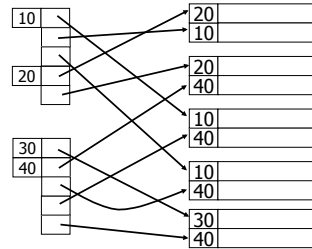
Notes 4 - Indexing

55

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate values & secondary indexes

another option...



CS 525



Notes 4 - Indexing

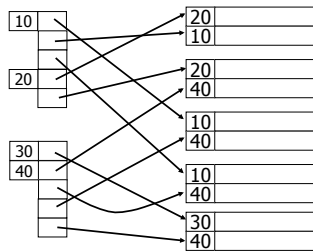
56

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate values & secondary indexes

another option...

Problem:
variable size records in index!



CS 525

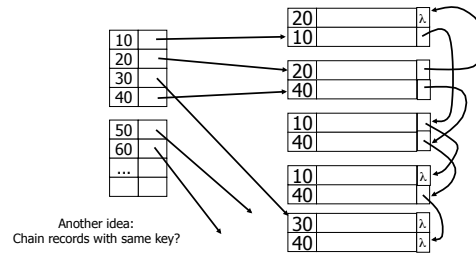


Notes 4 - Indexing

57

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate values & secondary indexes



Another idea:
Chain records with same key?

CS 525

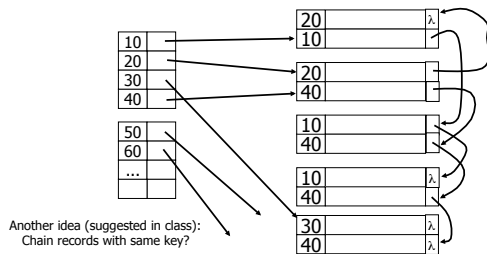


Notes 4 - Indexing

58

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

CS 525

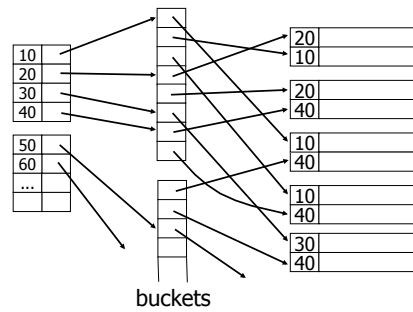


Notes 4 - Indexing

59

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate values & secondary indexes



buckets

CS 525



Notes 4 - Indexing

60

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Why “bucket” idea is useful

Indexes Records

Name: primary EMP (name,dept,floor,...)

Dept: secondary

Floor: secondary

CS 525 COMPUTER SCIENCE Notes 4 - Indexing 61 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

Dept. index EMP Floor index

CS 525 COMPUTER SCIENCE Notes 4 - Indexing 62 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

Dept. index EMP Floor index

→ Intersect toy bucket and 2nd Floor bucket to get set of matching EMP's

CS 525 COMPUTER SCIENCE Notes 4 - Indexing 63 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 ...

CS 525 COMPUTER SCIENCE Notes 4 - Indexing 64 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

This idea used in text information retrieval

cat dog

Documents

- ...the cat is fat ...
- ...was raining cats and dogs...
- ...Fido the dog ...

Inverted lists

CS 525 COMPUTER SCIENCE Notes 4 - Indexing 65 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”

CS 525 COMPUTER SCIENCE Notes 4 - Indexing 66 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- Tricks to weigh scores + normalize

e.g.: Match on common word not as useful as match on rare words...

CS 525



Notes 4 - Indexing

73

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- How to process V.S. Queries?

$$Q = \begin{matrix} & w1 & w2 & w3 & w4 & w5 & w6 & \dots \\ < & 0 & 0 & 0 & 1 & 1 & 0 & \dots > \end{matrix}$$

CS 525



Notes 4 - Indexing

74

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- Try Stanford Libraries
- Try Google, Yahoo, ...

CS 525



Notes 4 - Indexing

75

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

CS 525



Notes 4 - Indexing

76

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Conventional indexes

Advantage:

- Simple
- Index is sequential file good for scans

Disadvantage:

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

CS 525

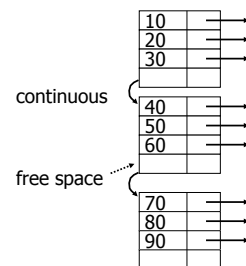


Notes 4 - Indexing

77

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example Index (sequential)



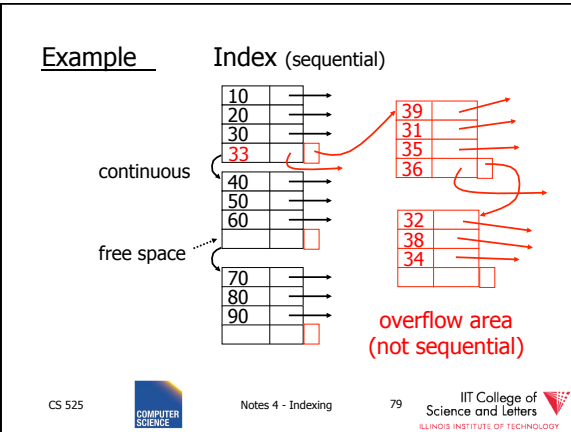
CS 525



Notes 4 - Indexing

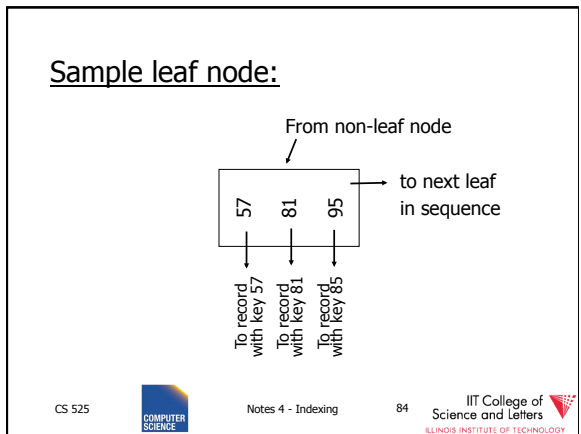
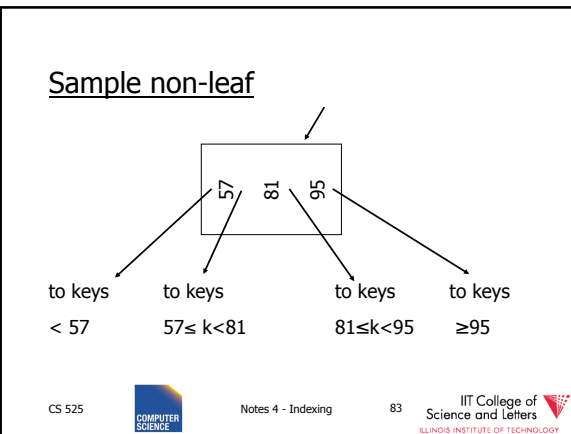
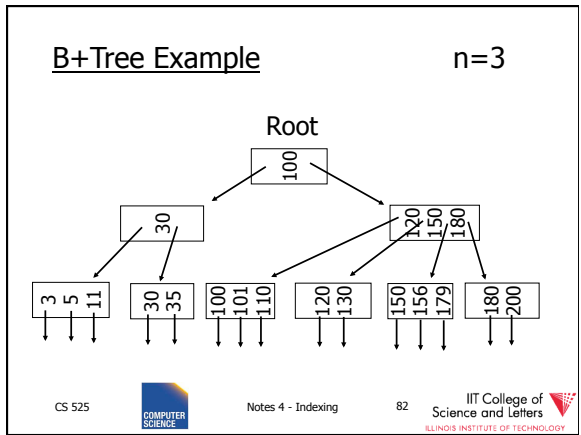
78

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

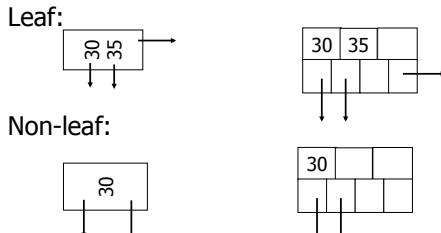



- Outline:**
- Conventional indexes
 - B-Trees ⇒ NEXT
 - Hashing schemes
 - Advanced Index Techniques
- CS 525 COMPUTER SCIENCE Notes 4 - Indexing 80 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


- **NEXT:** Another type of index
 - Give up on sequentiality of index
 - Try to get "balance"
- CS 525 COMPUTER SCIENCE Notes 4 - Indexing 81 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY




In textbook's notation n=3

Leaf: 

Non-leaf: 

CS 525  Notes 4 - Indexing 85 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Size of nodes: $\left\{ \begin{array}{l} n+1 \text{ pointers} \\ n \text{ keys} \end{array} \right.$ (fixed)


CS 525  Notes 4 - Indexing 86 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

CS 525  Notes 4 - Indexing 87 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


n=3

Full node min. node

Non-leaf $\left\{ \begin{array}{l} \text{Full node} \\ \text{min. node} \end{array} \right.$


Leaf $\left\{ \begin{array}{l} \text{Full node} \\ \text{min. node} \end{array} \right.$

counts even if null

CS 525  Notes 4 - Indexing 88 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)
- (2) Pointers in leaves point to records except for "sequence pointer"

CS 525  Notes 4 - Indexing 89 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

(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

CS 525  Notes 4 - Indexing 90 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
 - $\text{Key}[i] < k < \text{Key}[i + 1]$
 - Follow $i+1^{\text{th}}$ pointer
- If current node is leaf return pointer to record or fail (no such record in tree)

CS 525

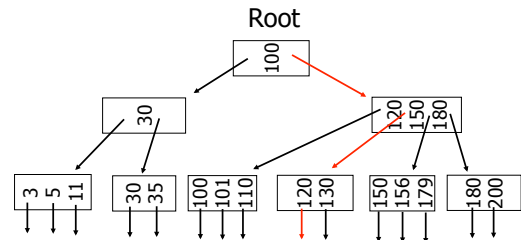


Notes 4 - Indexing

91

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Search Example **k=120** **n=3**



CS 525



Notes 4 - Indexing

92

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 $O(\log_2(n))$

CS 525



Notes 4 - Indexing

93

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Insert into B+tree

- simple case
 - space available in leaf
- leaf overflow
- non-leaf overflow
- new root

CS 525



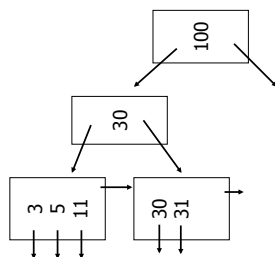
Notes 4 - Indexing

94

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(a) Insert key = 32

n=3



CS 525



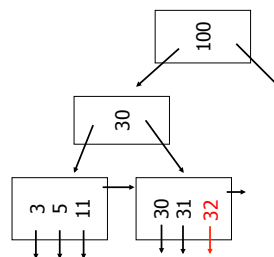
Notes 4 - Indexing

95

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(a) Insert key = 32

n=3



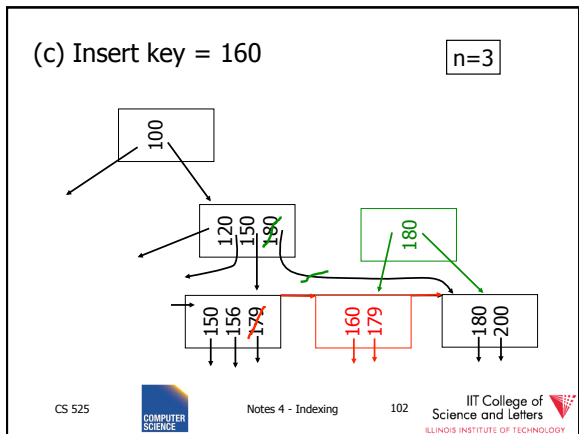
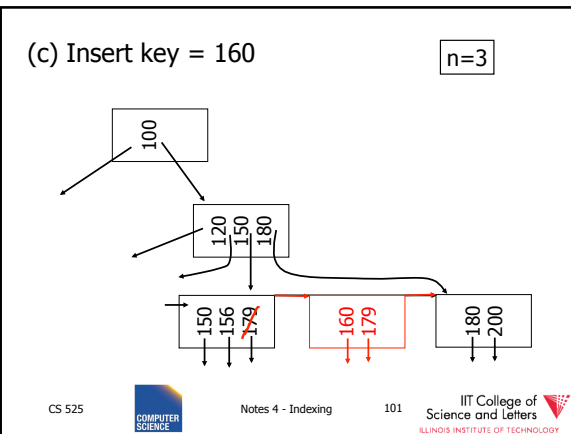
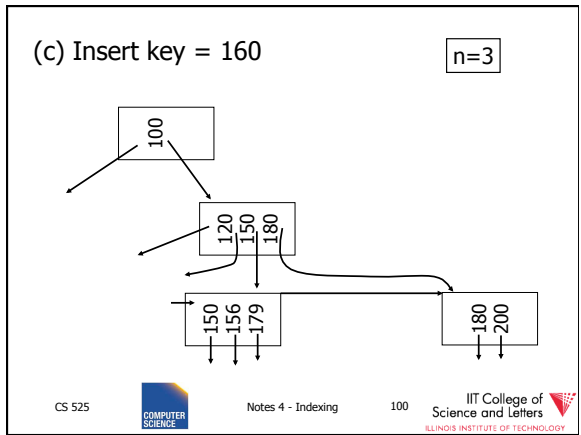
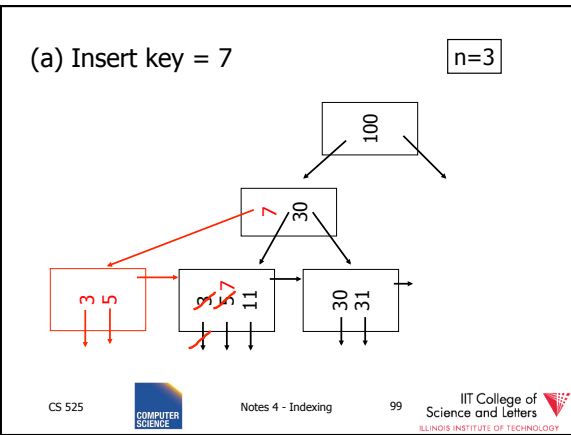
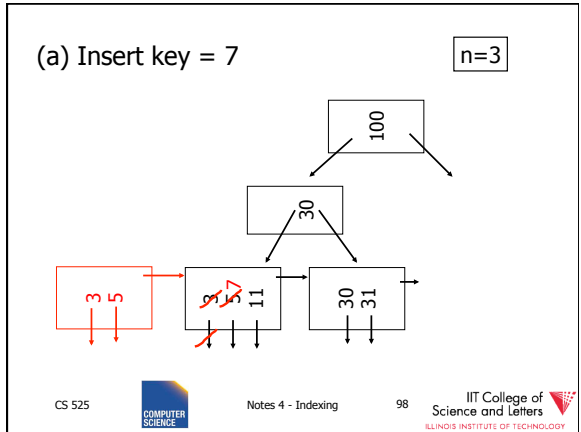
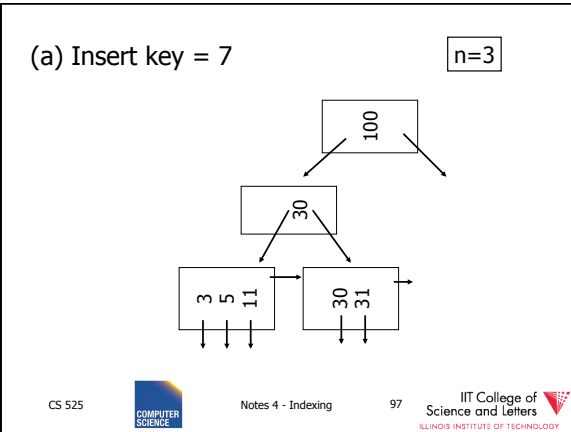
CS 525

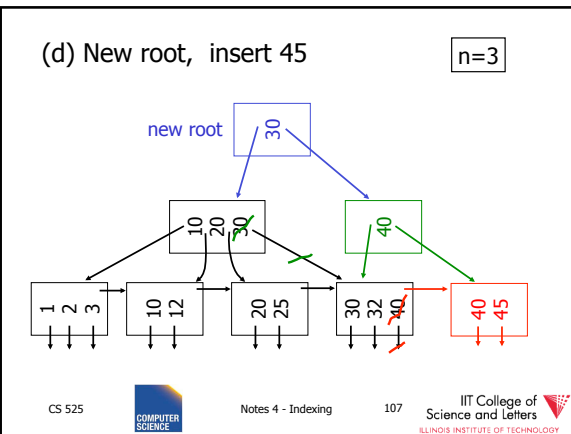
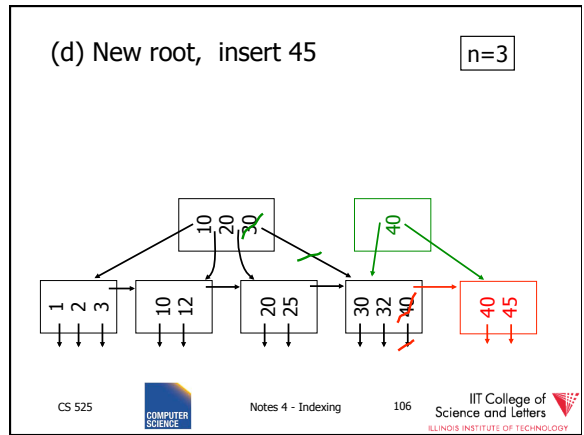
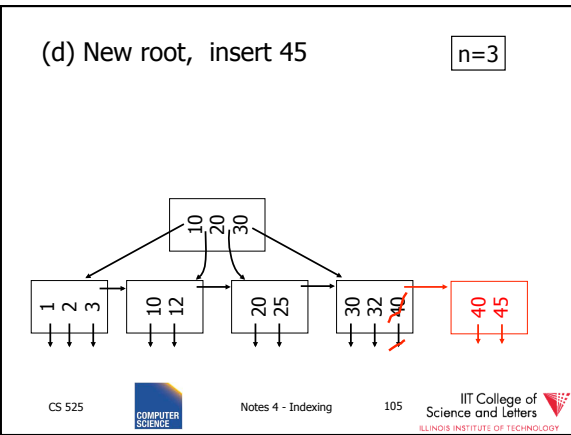
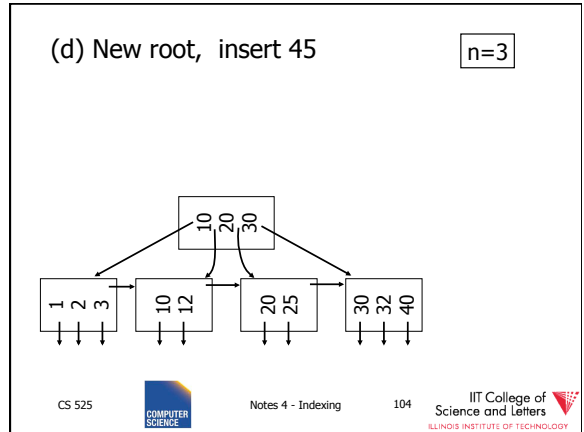
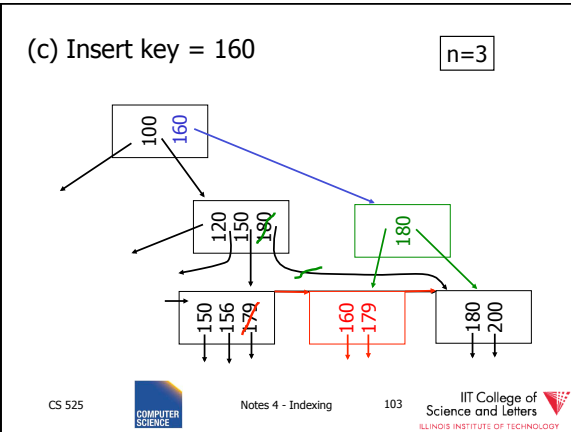


Notes 4 - Indexing

96

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

CS 525 Notes 4 - Indexing 108 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

CS 525



Notes 4 - Indexing

109

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion from B+tree

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

CS 525



Notes 4 - Indexing

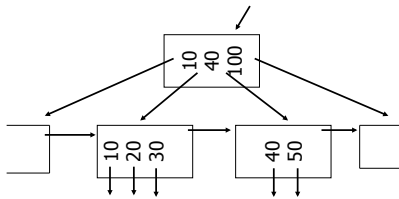
110

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(b) Coalesce with sibling

- Delete 50

n=4



CS 525



Notes 4 - Indexing

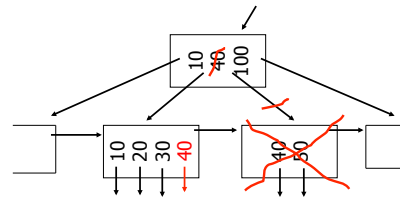
111

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(b) Coalesce with sibling

- Delete 50

n=4



CS 525



Notes 4 - Indexing

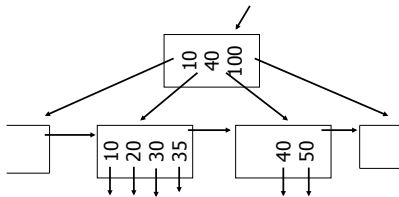
112

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(c) Redistribute keys

- Delete 50

n=4



CS 525



Notes 4 - Indexing

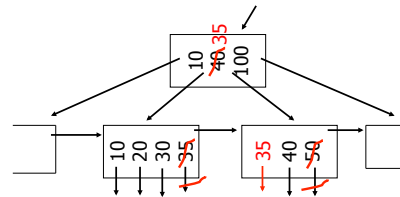
113

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

(c) Redistribute keys

- Delete 50

n=4



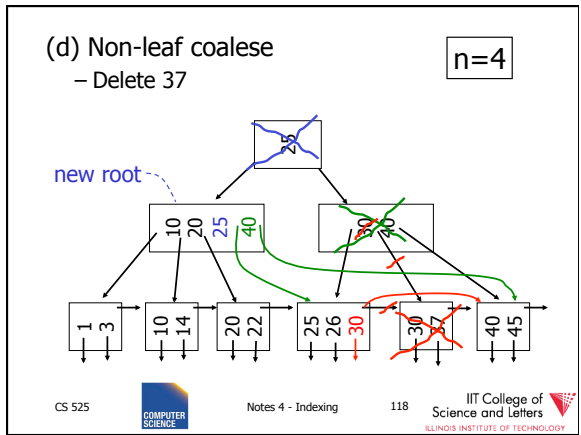
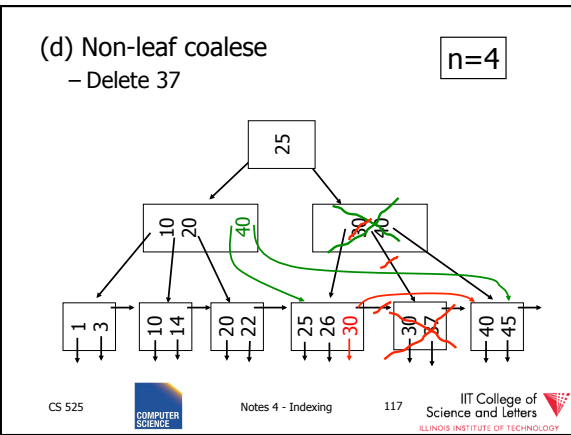
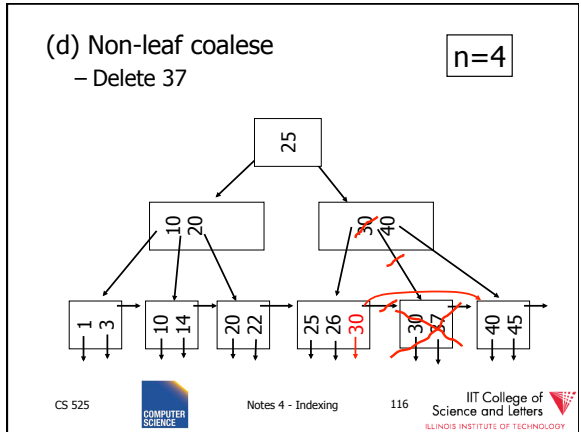
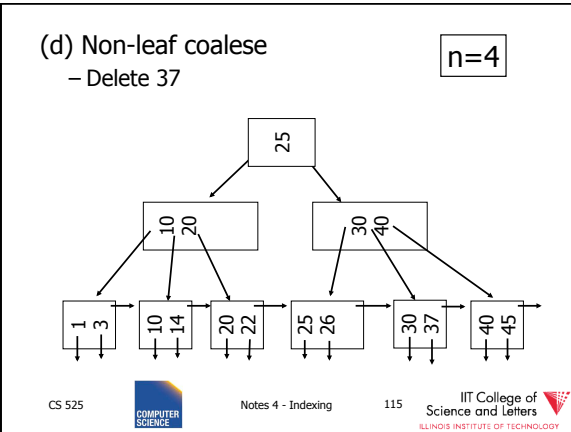
CS 525



Notes 4 - Indexing

114

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
- CS 525 Notes 4 - Indexing 119 IIT College of Science and Letters

- ### 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
- CS 525 Notes 4 - Indexing 120 IIT College of Science and Letters



B+tree deletions in practice

- Often, coalescing is not implemented
 - Too hard and not worth it!
 - Assumption: nodes will fill up in time again

CS 525  Notes 4 - Indexing 121 

Comparison: B-trees vs. static indexed sequential file



Ref #1: Held & Stonebraker
 "B-Trees Re-examined"
 CACM, Feb. 1978

CS 525  Notes 4 - Indexing 122 

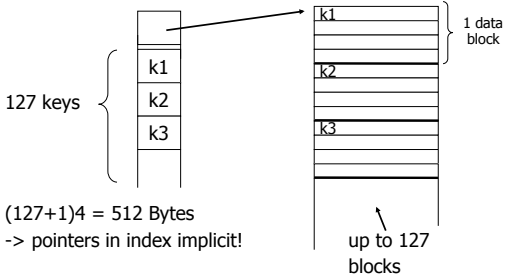
Ref # 1 claims:



- 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

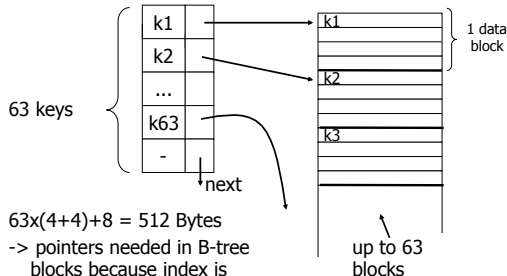
CS 525  Notes 4 - Indexing 123 



Example: 1 block static index





CS 525  Notes 4 - Indexing 124 

Example: 1 block B-tree



CS 525  Notes 4 - Indexing 125 

Static Index		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

CS 525  Notes 4 - Indexing 126 

Ref. #1 analysis claims

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

CS 525



Notes 4 - Indexing

127

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
⇒ Static index saves enough accesses to allow for reorganization

Ref. #1 conclusion → Static index better!!

CS 525



Notes 4 - Indexing

128

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!!

CS 525



Notes 4 - Indexing

129

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Ref. #2 conclusion → B-trees better!!

- DBA does not know when to reorganize
- DBA does not know how full to load pages of new index

CS 525



Notes 4 - Indexing

130

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 buffers size needed for this)

CS 525



Notes 4 - Indexing

131

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 4 - Indexing

132

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)

CS 525



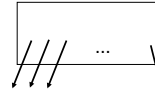
Notes 4 - Indexing

133

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

CS 525



Notes 4 - Indexing

134

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Sample assumptions:

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

CS 525



Notes 4 - Indexing

135

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 $a \ll S$

CS 525



Notes 4 - Indexing

136

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



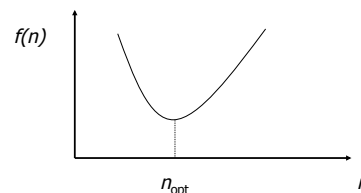
Notes 4 - Indexing

137

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

►►Can get:

$f(n) = \text{time to find a record}$



CS 525




Notes 4 - Indexing

138

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

➔ FIND n_{opt} by $f'(n) = 0$

Answer is $n_{opt} = \text{"few hundred"}$


CS 525  Notes 4 - Indexing 139 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

➔ FIND n_{opt} by $f'(n) = 0$

Answer is $n_{opt} = \text{"few hundred"}$


➔ What happens to n_{opt} as

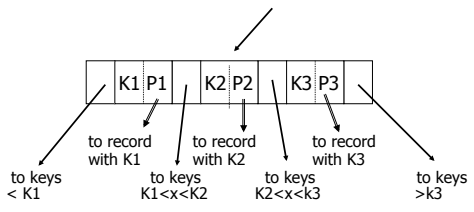
- Disk gets faster?
- CPU get faster?
- Memory hierarchy?


CS 525  Notes 4 - Indexing 140 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

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

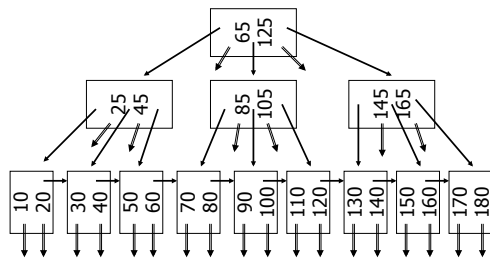
- Idea:
 - Avoid duplicate keys
 - Have record pointers in non-leaf nodes


CS 525  Notes 4 - Indexing 141 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY



CS 525  Notes 4 - Indexing 142 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

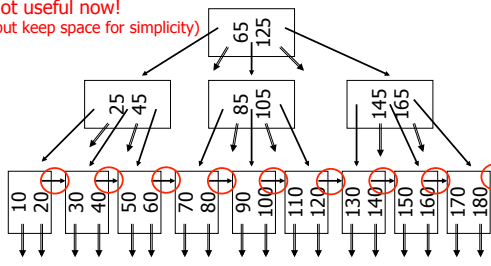
B-tree example $n=2$




CS 525  Notes 4 - Indexing 143 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

B-tree example $n=2$

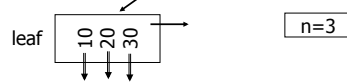
- sequence pointers
not useful now!
(but keep space for simplicity)



CS 525  Notes 4 - Indexing 144 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Note on inserts

- Say we insert record with key = 25



CS 525



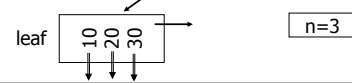
Notes 4 - Indexing

145

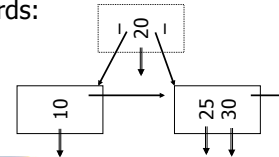
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Note on inserts

- Say we insert record with key = 25



- Afterwards:



CS 525



Notes 4 - Indexing

146

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 4 - Indexing

147

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

CS 525



Notes 4 - Indexing

148

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!

CS 525



Notes 4 - Indexing

149

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

But note:

- If blocks are fixed size
(due to disk and buffering restrictions)
Then lookup for B-tree is actually better!!

CS 525



Notes 4 - Indexing

150

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example:

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

CS 525



Notes 4 - Indexing

151

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

B-tree:

Root has 8 keys + 8 record pointers
+ 9 son pointers
 $= 8 \times 4 + 8 \times 4 + 9 \times 4 = 100$ bytes

CS 525



Notes 4 - Indexing

152

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

B-tree:

Root has 8 keys + 8 record pointers
+ 9 son pointers
 $= 8 \times 4 + 8 \times 4 + 9 \times 4 = 100$ bytes

Each of 9 sons: 12 rec. pointers (+12 keys)
 $= 12 \times (4+4) + 4 = 100$ bytes

CS 525



Notes 4 - Indexing

153

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

B-tree:

Root has 8 keys + 8 record pointers
+ 9 son pointers
 $= 8 \times 4 + 8 \times 4 + 9 \times 4 = 100$ bytes

Each of 9 sons: 12 rec. pointers (+12 keys)
 $= 12 \times (4+4) + 4 = 100$ bytes

2-level B-tree, Max # records =
 $12 \times 9 + 8 = 116$

CS 525



Notes 4 - Indexing

154

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

B+tree:

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

CS 525



Notes 4 - Indexing

155

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

B+tree:

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

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

CS 525



Notes 4 - Indexing

156

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

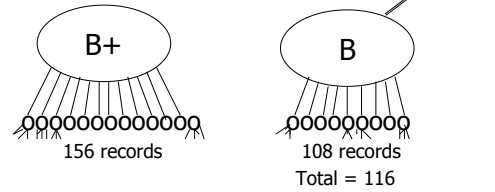
B+tree:

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

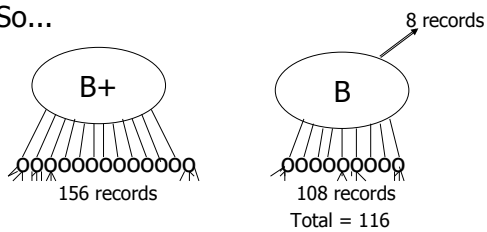
Each of 13 sons: 12 rec. ptrs (+12 keys)
 = $12 \times (4 + 4) + 4 = 100$ bytes

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

So...



So...



- Conclusion:
 - For fixed block size,
 - B+ tree is better because it is bushier

Additional B-tree Variants

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

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

One Solution: Multiple Indexes

- Example: I1, I2

day	days indexed	days indexed
	I1	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

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

CS 525



Notes 4 - Indexing

163

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 4 - Indexing

164

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Locks Nodes

- 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 if node has more than $n/2$

	Read	Write
Read	X	-
Write	-	-

CS 525



Notes 4 - Indexing

165

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

CS 525



Notes 4 - Indexing

166

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 4 - Indexing

167

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

CS 525



Notes 4 - Indexing

168

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

CS 525: Advanced Database Organization

05: Hashing and More

Boris Glavic



Slides: adapted from a [course](#) taught by [Hector Garcia-Molina](#), Stanford InfoLab

CS 525



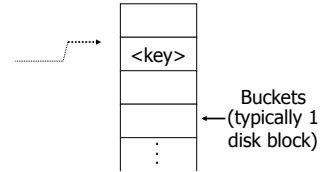
Notes 5 - Hashing

1

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Hashing

key \rightarrow h(key)



CS 525



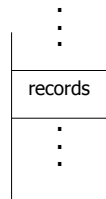
Notes 5 - Hashing

2

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Two alternatives

(1) key \rightarrow h(key)



CS 525



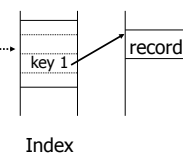
Notes 5 - Hashing

3

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Two alternatives

(2) key \rightarrow h(key)



CS 525



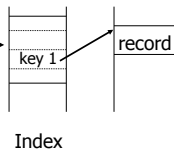
Notes 5 - Hashing

4

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Two alternatives

(2) key \rightarrow h(key)



- Alt (2) for "secondary" search key

CS 525



Notes 5 - Hashing

5

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example hash function

- Key = 'x₁ x₂ ... x_n' n byte character string
- Have b buckets
- h: add $x_1 + x_2 + \dots + x_n$
- compute sum modulo b

CS 525



Notes 5 - Hashing

6

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.

CS 525



Notes 5 - Hashing

7

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.

Good hash function: ↗ Expected number of keys/bucket is the same for all buckets

CS 525



Notes 5 - Hashing

8

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

CS 525

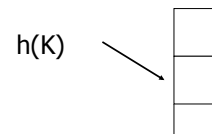


Notes 5 - Hashing

9

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Next: example to illustrate inserts, overflows, deletes



CS 525



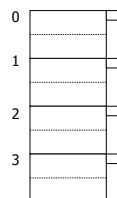
Notes 5 - Hashing

10

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

EXAMPLE 2 records/bucket

INSERT:
 $h(a) = 1$
 $h(b) = 2$
 $h(c) = 1$
 $h(d) = 0$



CS 525



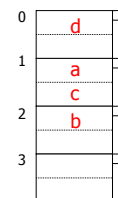
Notes 5 - Hashing

11

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

EXAMPLE 2 records/bucket

INSERT:
 $h(a) = 1$
 $h(b) = 2$
 $h(c) = 1$
 $h(d) = 0$
 $h(e) = 1$



CS 525



Notes 5 - Hashing

12

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

EXAMPLE 2 records/bucket

INSERT:

$h(a) = 1$
 $h(b) = 2$
 $h(c) = 1$
 $h(d) = 0$
 $h(e) = 1$

CS 525 Notes 5 - Hashing 13 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EXAMPLE: deletion

Delete:

e
f

CS 525 Notes 5 - Hashing 14 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EXAMPLE: deletion

Delete:

e
f
c

CS 525 Notes 5 - Hashing 15 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EXAMPLE: deletion

Delete:

e
f
c

CS 525 Notes 5 - Hashing 16 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Rule of thumb:

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

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

CS 525 Notes 5 - Hashing 17 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

CS 525 Notes 5 - Hashing 18 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY



How do we cope with growth?

- Overflows and reorganizations
- Dynamic hashing

CS 525  Notes 5 - Hashing 19 

How do we cope with growth?

- Overflows and reorganizations
- Dynamic hashing
 - Extensible
 - Linear

CS 525  Notes 5 - Hashing 20 



Extensible hashing: two ideas

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

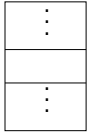
$h(k) \rightarrow$ 00110101



← b →

use $i \rightarrow$ grows over time....

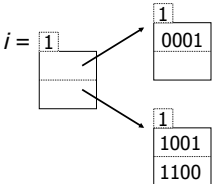
CS 525  Notes 5 - Hashing 21 

(b) Use directory



$h(k)[i]$  \rightarrow to bucket

CS 525  Notes 5 - Hashing 22 

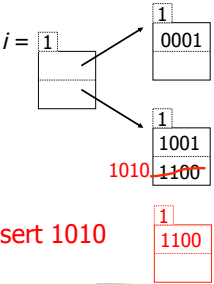
Example: $h(k)$ is 4 bits; 2 keys/bucket

$i =$ 1 



Insert 1010

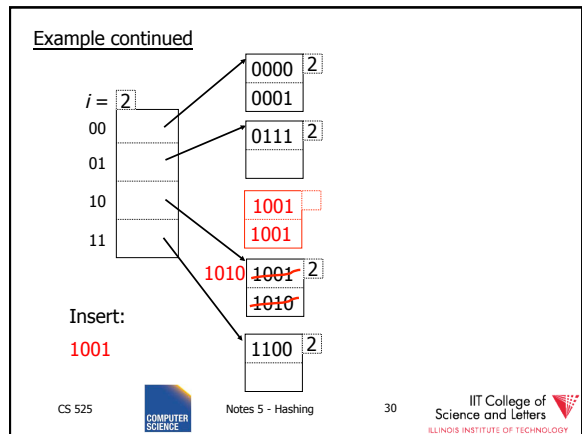
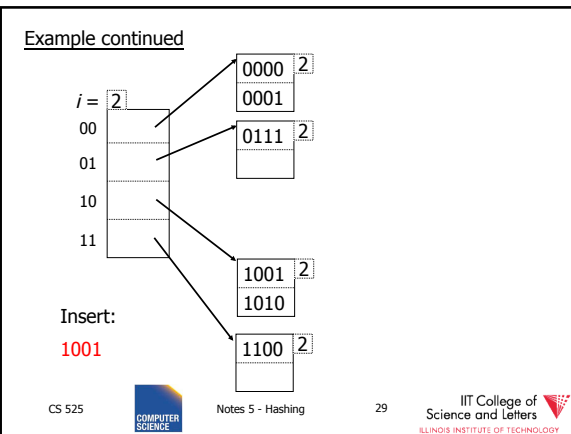
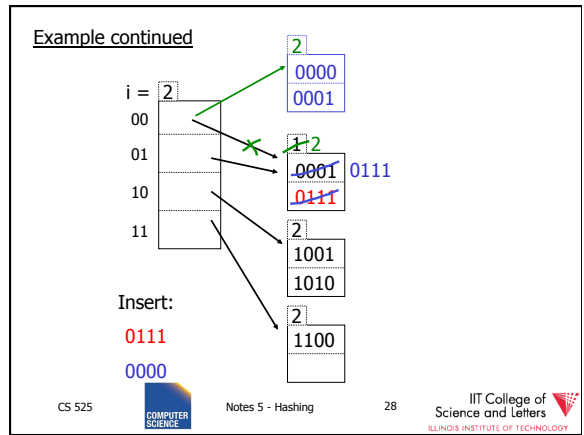
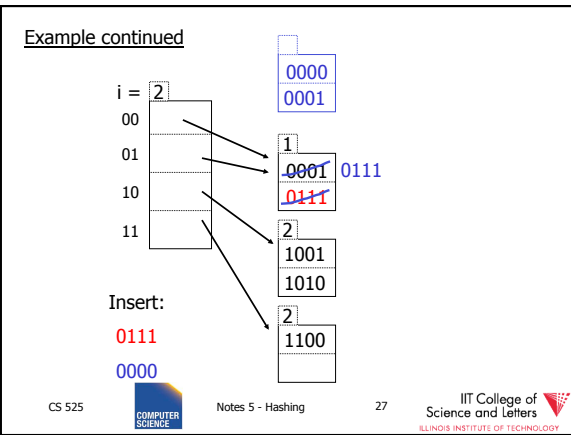
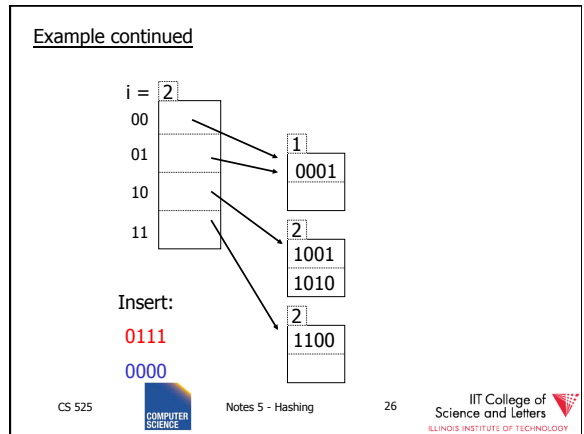
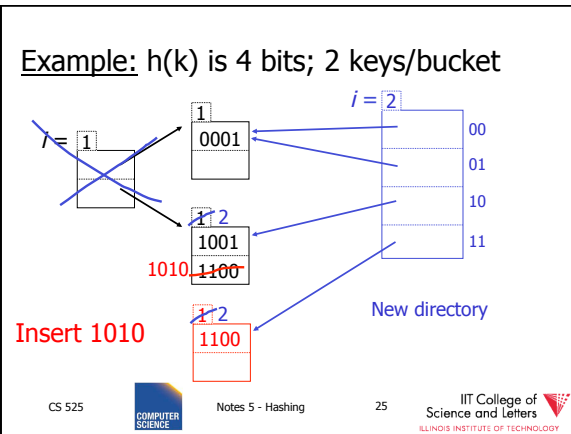
CS 525  Notes 5 - Hashing 23 

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

$i =$ 1 


Insert 1010

CS 525  Notes 5 - Hashing 24 



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)

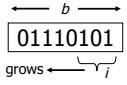
CS 525  Notes 5 - Hashing 37 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY


Linear hashing

- Another dynamic hashing scheme

Two ideas:

(a) Use i low order bits of hash



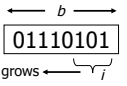
CS 525  Notes 5 - Hashing 38 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Linear hashing


- Another dynamic hashing scheme


Two ideas:

(a) Use i low order bits of hash



(b) File grows linearly



CS 525  Notes 5 - Hashing 39 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

CS 525  Notes 5 - Hashing 40 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)

Rule If $h(k)[i] \leq m$, then
look at bucket $h(k)[i]$
else, look at bucket $h(k)[i] - 2^{i-1}$

CS 525  Notes 5 - Hashing 41 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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


- insert 0101

0000	0101		
1010	1111		

00 01 10 11

$m = 01$ (max used block)

Rule If $h(k)[i] \leq m$, then
look at bucket $h(k)[i]$
else, look at bucket $h(k)[i] - 2^{i-1}$

CS 525  Notes 5 - Hashing 42 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

• insert 0101
• can have overflow chains!

0000 0101
1010 1111

00 01 10 11

$m = 01$ (max used block)

Rule If $h(k)[i] \leq m$, then
look at bucket $h(k)[i]$
else, look at bucket $h(k)[i] - 2^{i-1}$

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 43 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Note

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

0000 0101
1010 1111

00 01 10 11

$m = 01$ (max used block)

$n = 10$

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 44 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)

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 45 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

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

0000 0101 1010
1010 1111

00 01 10 11

$m = 10$ (max used block)

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 46 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

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

• insert 0101

0000 0101 1010
1010 1111

00 01 10 11

$m = 10$ (max used block)

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 47 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

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

• insert 0101

0000 0101 1010 1111
1010



00 01 10 11

$m = 11$ (max used block)

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 48 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY



☛ When do we expand file?

- Keep track of:
$$\frac{\text{\# used slots}}{\text{total \# of slots}} = U$$
- If $U > \text{threshold}$ then increase m (and maybe i)

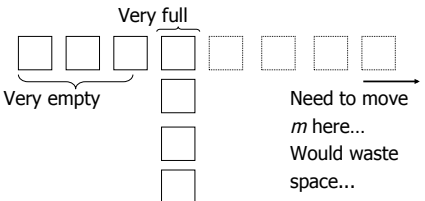
CS 525  Notes 5 - Hashing 55 



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

CS 525  Notes 5 - Hashing 56 

Example: BAD CASE





CS 525  Notes 5 - Hashing 57 

Summary



Hashing

- How it works
- Dynamic hashing
 - Extensible
 - Linear

CS 525  Notes 5 - Hashing 58 

Next:



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

CS 525  Notes 5 - Hashing 59 

Indexing vs Hashing

- Hashing good for probes given key

e.g.,
 SELECT ...
 FROM R
 WHERE R.A = 5

CS 525  Notes 5 - Hashing 60 

Indexing vs Hashing

- INDEXING (Including B Trees) good for Range Searches:
e.g.,
SELECT
FROM R
WHERE R.A > 5

CS 525



Notes 5 - Hashing

61

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Index definition in SQL

- Create index name on rel (attr)
- Create unique index name on rel (attr)
└─ defines candidate key
- Drop INDEX name

CS 525



Notes 5 - Hashing

62

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 SQL...

CS 525



Notes 5 - Hashing

63

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Note ATTRIBUTE LIST ⇒ MULTIKEY INDEX
(next)

e.g., CREATE INDEX foo ON R(A,B,C)

CS 525



Notes 5 - Hashing

64

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Multi-key Index

Motivation: Find records where
DEPT = "Toy" AND SAL > 50k

CS 525



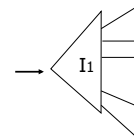
Notes 5 - Hashing

65

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



CS 525



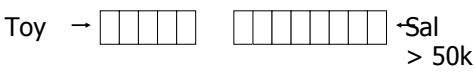
Notes 5 - Hashing


66

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Strategy II:

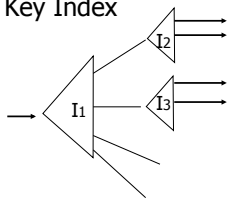
- Use 2 Indexes; Manipulate Pointers


Toy → 

CS 525  Notes 5 - Hashing 67 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Strategy III:

- Multiple Key Index

One idea: 

CS 525  Notes 5 - Hashing 68 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Art
Sales
Toy

Dept Index


10k
15k
17k
21k

Salary Index

12k
15k
15k
19k


Example Record

Name=Joe
DEPT=Sales
SAL=15k

CS 525  Notes 5 - Hashing 69 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

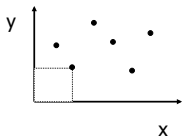
For which queries is this index good?

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

CS 525  Notes 5 - Hashing 70 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


Interesting application:

- Geographic Data




DATA:

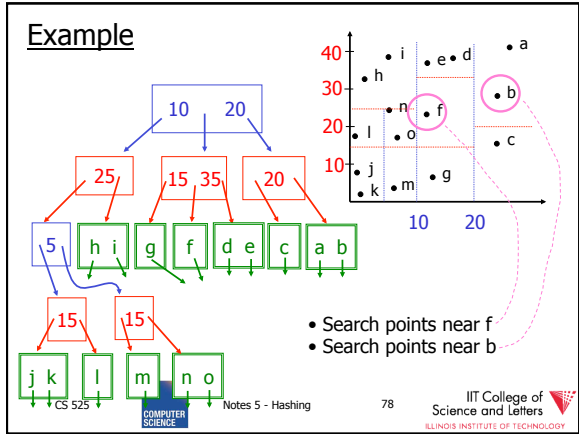
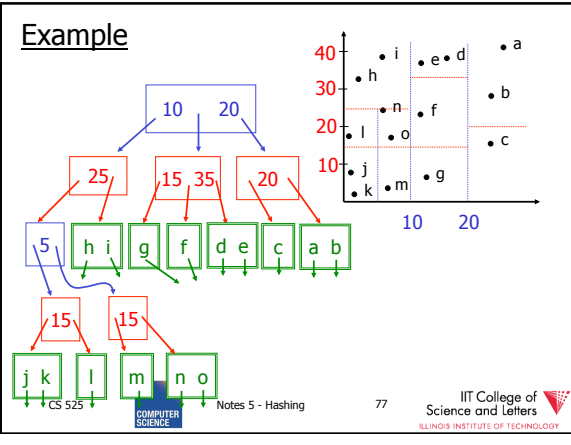
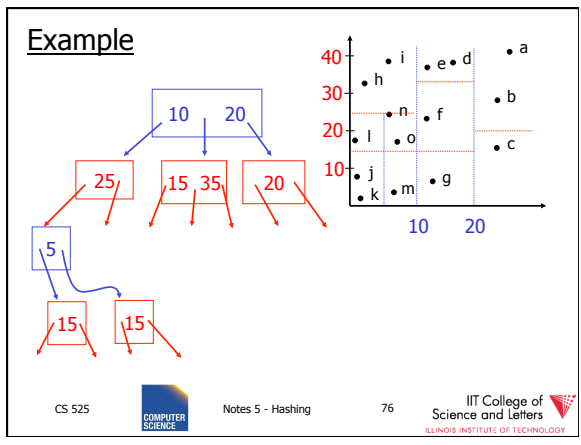
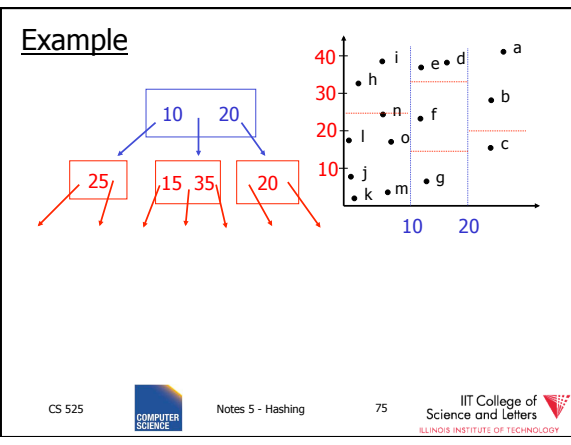
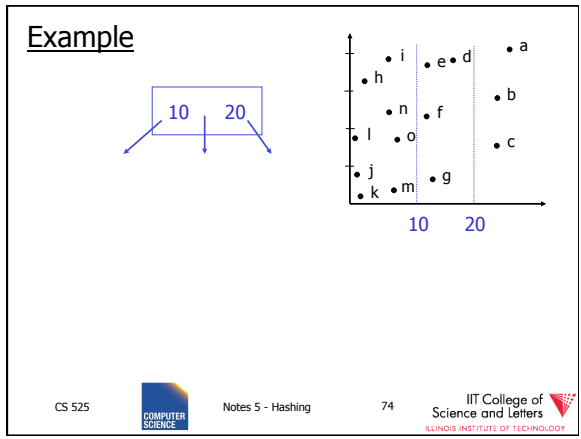
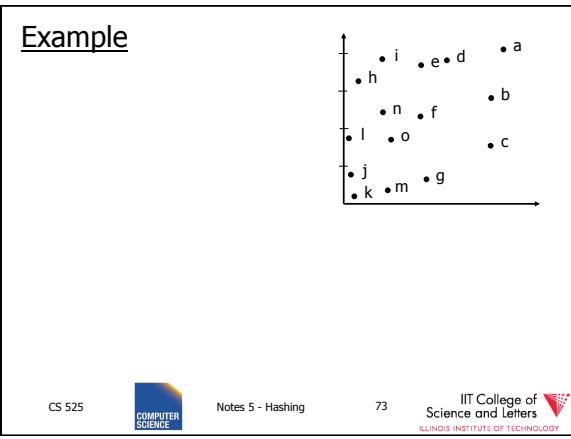
- <X1,Y1, Attributes>
- <X2,Y2, Attributes>
- ⋮

CS 525  Notes 5 - Hashing 71 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Queries:

- What city is at <Xi,Yi>?
- What is within 5 miles from <Xi,Yi>?
- Which is closest point to <Xi,Yi>?

CS 525  Notes 5 - Hashing 72 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 $i = \langle 12, 38 \rangle$
- Find points “close” to $b = \langle 7, 24 \rangle$

CS 525



Notes 5 - Hashing

79

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Next

- Even more index structures ☺

CS 525



Notes 5 - Hashing


80

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

CS 525: Advanced Database Organization

06: Even more index structures

Boris Glavic



Slides: adapted from a [course](#) taught by [Hector Garcia-Molina](#), Stanford InfoLab

CS 525 COMPUTER SCIENCE Notes 6 - More Indices 1 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Recap

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

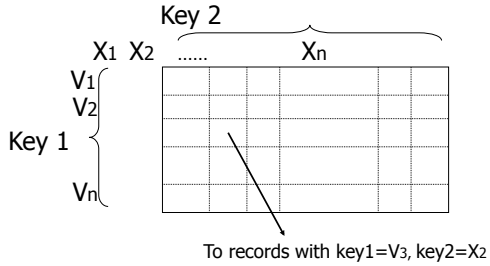
CS 525 COMPUTER SCIENCE Notes 6 - More Indices 2 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

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**

CS 525 COMPUTER SCIENCE Notes 6 - More Indices 3 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Grid Index



CS 525 COMPUTER SCIENCE Notes 5 - Hashing 4 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

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 5 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 $< X_j$

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 6 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 $N=4$

$pos(i, j) =$

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	

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 7 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...

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	

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 8 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Solution: Use Indirection

Buckets

V1 X1 X2 X3

V2

V3

V4

* only pointers to buckets

Buckets

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 9 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

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 10 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Can also index grid on value ranges

Salary Grid

0-20K	1
20K-50K	2
50K-∞	3

Linear Scale

1	2	3
Toy	Sales	Personnel

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 11 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

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 12 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Partitioned hash function

Idea:

010110 1110010

Key1 → h1 ← Key2 → h2 ←

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 13 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy) = 0	000	
h1(sales) = 1	001	
h1(art) = 1	010	
.	011	
.	100	
h2(10k) = 01	101	
h2(20k) = 11	110	
h2(30k) = 01	111	
h2(40k) = 00		

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

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 14 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy) = 0	000	
h1(sales) = 1	001	<Fred>
h1(art) = 1	010	
.	011	
.	100	
h2(10k) = 01	101	
h2(20k) = 11	110	<Joe><Sally>
h2(30k) = 01	111	
h2(40k) = 00		

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

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 15 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy) = 0	000	<Fred>
h1(sales) = 1	001	<Joe><Jan>
h1(art) = 1	010	<Mary>
.	011	
.	100	<Sally>
h2(10k) = 01	101	
h2(20k) = 11	110	<Tom><Bill>
h2(30k) = 01	111	<Andy>
h2(40k) = 00		

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

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 16 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy) = 0	000	<Fred>
h1(sales) = 1	001	<Joe><Jan>
h1(art) = 1	010	<Mary>
.	011	
.	100	<Sally>
h2(10k) = 01	101	
h2(20k) = 11	110	<Tom><Bill>
h2(30k) = 01	111	<Andy>
h2(40k) = 00		

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

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 17 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy) = 0	000	<Fred>
h1(sales) = 1	001	<Joe><Jan>
h1(art) = 1	010	<Mary>
.	011	
.	100	<Sally>
h2(10k) = 01	101	
h2(20k) = 11	110	<Tom><Bill>
h2(30k) = 01	111	<Andy>
h2(40k) = 00		



Find Emp. with Sal=30k

CS 525 COMPUTER SCIENCE Notes 5 - Hashing 18 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

EX:

h1(toy) =0	000	<Fred>
h1(sales) =1	001	<Joe><Jan>
h1(art) =1	010	<Mary>
.	011	
.	100	<Sally>
h2(10k) =01	101	<Tom><Bill>
h2(20k) =11	110	<Tom><Bill>
h2(30k) =01	111	<Andy>
h2(40k) =00		



Find Emp. with Sal=30k

CS 525  Notes 5 - Hashing 19 

EX:

h1(toy) =0	000	<Fred>
h1(sales) =1	001	<Joe><Jan>
h1(art) =1	010	<Mary>
.	011	
.	100	<Sally>
h2(10k) =01	101	
h2(20k) =11	110	<Tom><Bill>
h2(30k) =01	111	<Andy>
h2(40k) =00		



Find Emp. with Dept. = Sales

CS 525  Notes 5 - Hashing 20 

EX:



h1(toy) =0	000	<Fred>
h1(sales) =1	001	<Joe><Jan>
h1(art) =1	010	<Mary>
.	011	
.	100	<Sally>
h2(10k) =01	101	<Tom><Bill>
h2(20k) =11	110	<Tom><Bill>
h2(30k) =01	111	<Andy>
h2(40k) =00		

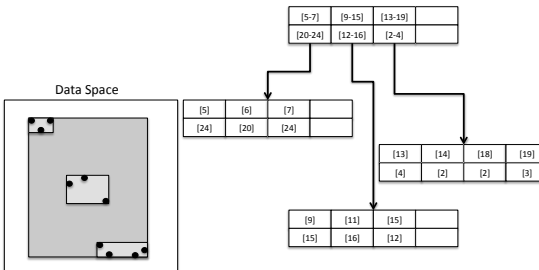
Find Emp. with Dept. = Sales



CS 525  Notes 5 - Hashing 21 

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

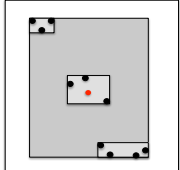
CS 525  Notes 6 - More Indices 22 





CS 525  Notes 6 - More Indices 23 

R-tree - Search

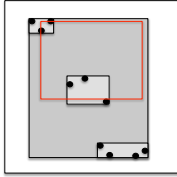
- Point Search
 - Search for $p = \langle x_i, y_i \rangle$
 - Keep list of potential nodes
 - Needed because of overlap
 - Traverse to child if MBR of child contains p





CS 525  Notes 6 - More Indices 24 

R-tree - Search



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



CS 525

Notes 6 - More Indices 25




R-tree - Search

Search $\langle 5, 24 \rangle$

CS 525

Notes 6 - More Indices 26




R-tree - Insert

- Similar to B-tree, but more complex
 - Overlap -> multiple choice 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

CS 525

Notes 6 - More Indices 27




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

CS 525

Notes 6 - More Indices 28




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

CS 525

Notes 6 - More Indices 29




Bitmap Index

- Domain of values $D = \{d_1, \dots, d_n\}$
 - 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

CS 525

Notes 6 - More Indices 30


Bitmap Index Example

Age			Todlers			Gender	
1	2	3	Name	Age	Gender	male	female
1	0	0	Peter	1	male	1	0
0	1	0	Gertrud	2	female	0	1
1	0	0	Joe	1	male	1	0
0	0	1	Marry	3	female	0	1



CS 525  Notes 6 - More Indices 31 

Bitmap Index Example

Age			Todlers			Gender	
1	2	3	Name	Age	Gender	male	female
1	0	0	Peter	1	male	1	0
0	1	0	Gertrud	2	female	0	1
1	0	0	Joe	1	male	1	0
0	0	1	Marry	3	female	0	1

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

0
1
0
0



CS 525  Notes 6 - More Indices 32 

Bitmap Index Example

Age			Todlers			Gender	
1	2	3	Name	Age	Gender	male	female
1	0	0	Peter	1	male	1	0
0	1	0	Gertrud	2	female	0	1
1	0	0	Joe	1	male	1	0
0	0	1	Marry	3	female	0	1



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

0
1
0
1

CS 525  Notes 6 - More Indices 33 



Compression

- Observation:
 - Each record has one value in index 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

CS 525  Notes 6 - More Indices 34 



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

CS 525  Notes 6 - More Indices 35 

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)

CS 525  Notes 6 - More Indices 36 

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)**

CS 525
Notes 6 - More Indices 37

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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)

CS 525
Notes 6 - More Indices 38

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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	

CS 525
Notes 6 - More Indices 39

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Bitmap Indices

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

CS 525
Notes 6 - More Indices 40

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

CS 525
Notes 6 - More Indices 41

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Trie

- Each node has pointers to $|D|$ child nodes
 - One for each value of D
- Searching for a key $k = [d_1, \dots, d_n]$
 - Start at the root
 - Follow child for value d_i

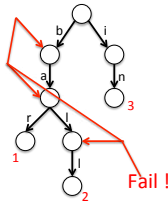
CS 525
Notes 6 - More Indices 42

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Trie Example

Words: bar, ball, in

Search for **bald**



CS 525



Notes 6 - More Indices

43

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Tries Implementation

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

CS 525



Notes 6 - More Indices

44

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



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

CS 525
Notes 7 - Query Processing
1 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Query Processing

Q → Query Plan

CS 525
Notes 7 - Query Processing
2 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Query Processing

Q → Query Plan

Focus: Relational System

- Others?

CS 525
Notes 7 - Query Processing
3 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Select B,D
From R,S
Where R.A = "c" ∧ S.E = 2 ∧
R.C=S.C

CS 525
Notes 7 - Query Processing
4 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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		

CS 525
Notes 7 - Query Processing
5 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525
Notes 7 - Query Processing
6 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

• How do we execute query?

One idea

- Do Cartesian product
- Select tuples
- Do projection

CS 525



Notes 7 - Query Processing

7

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	
⋮						

CS 525



Notes 7 - Query Processing

8

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...

CS 525

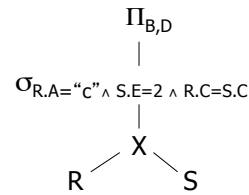


Notes 7 - Query Processing

9

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

Ex: Plan I



CS 525

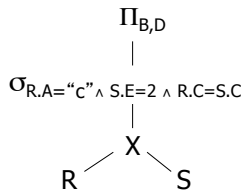


Notes 7 - Query Processing

10

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

Ex: Plan I



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

CS 525

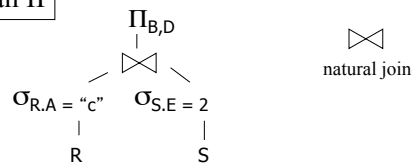


Notes 7 - Query Processing

11

Another idea:

Plan II

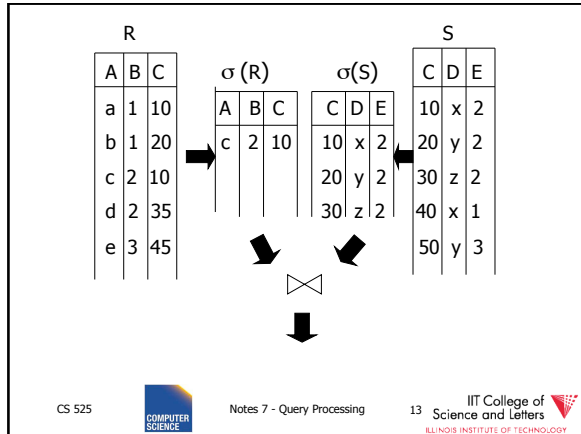


CS 525



Notes 7 - Query Processing

12



Plan III

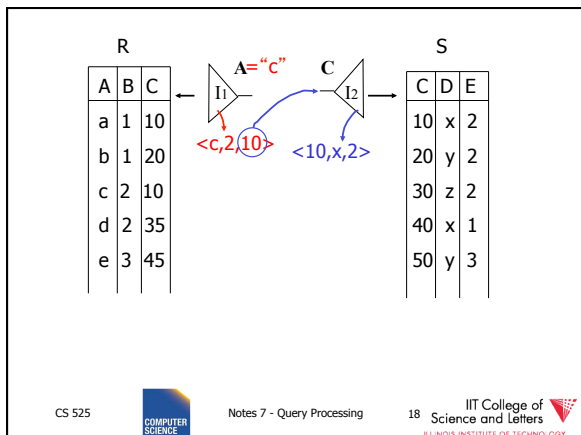
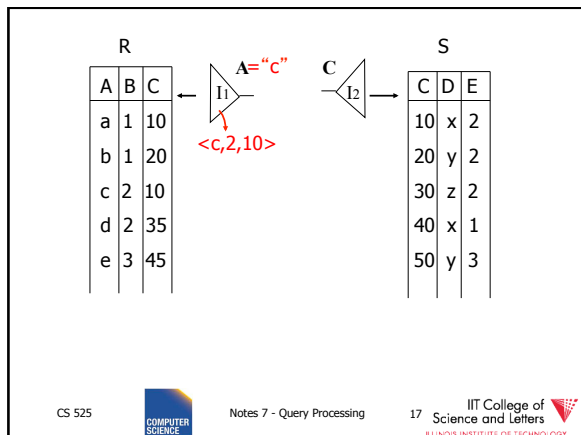
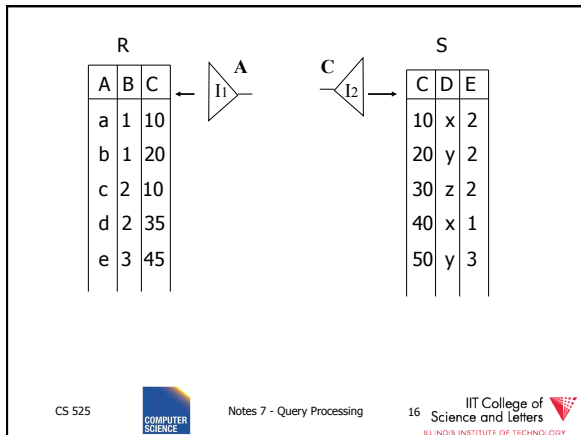
Use R.A and S.C Indexes

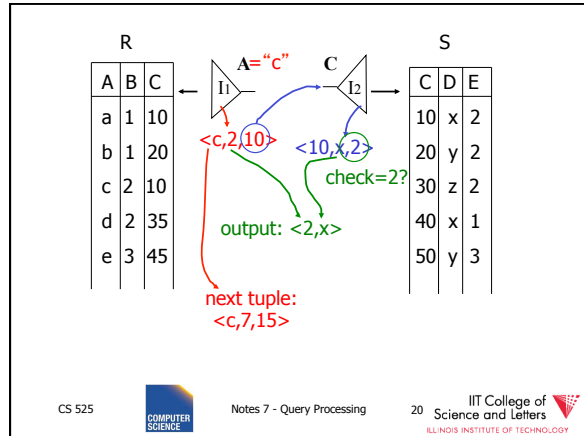
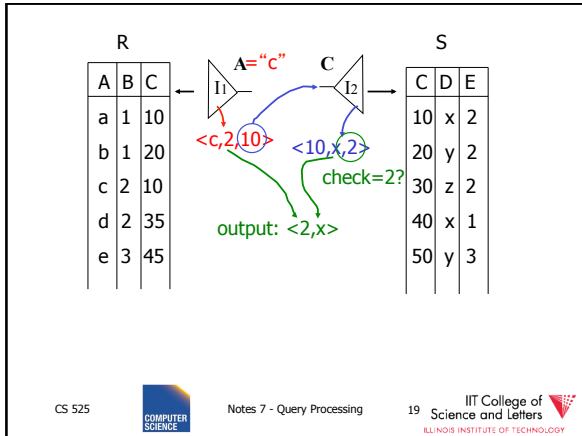
- (1) Use R.A index to select R tuples with R.A = "c"
- (2) For each R.C value found, use S.C index to find matching tuples

Plan III

Use R.A and S.C Indexes

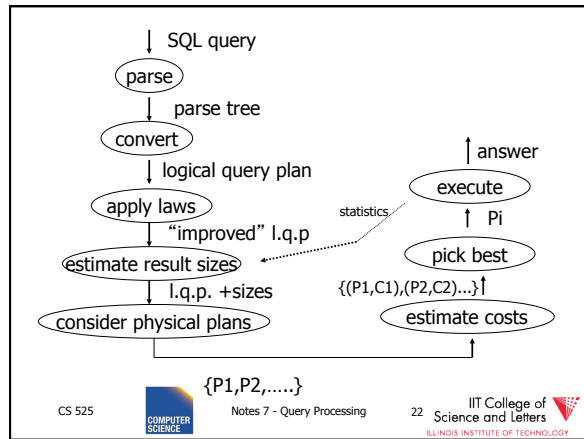
- (1) Use R.A index to select R tuples with R.A = "c"
- (2) For each R.C value found, use S.C index to find matching tuples
- (3) Eliminate S tuples S.E \neq 2
- (4) Join matching R,S tuples, project B,D attributes and place in result





Overview of Query Optimization

CS 525 COMPUTER SCIENCE Notes 7 - Query Processing 21 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY



Example: SQL query

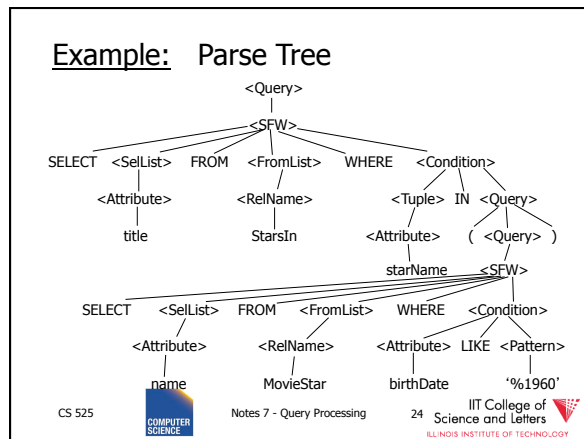
```

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

```

(Find the movies with stars born in 1960)

CS 525 COMPUTER SCIENCE Notes 7 - Query Processing 23 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY



Example: Generating Relational Algebra

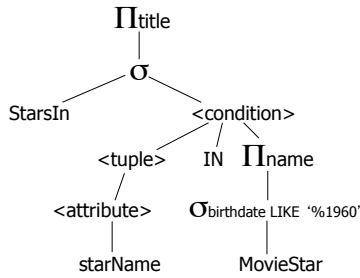


Fig. 7.15: An expression using a two-argument σ , midway between a parse tree and relational algebra

Example: Logical Query Plan

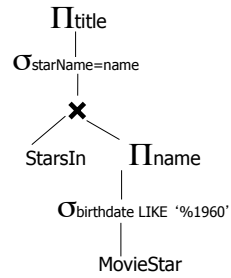
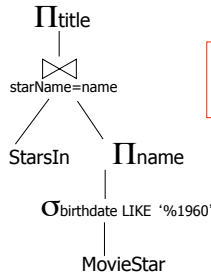


Fig. 7.18: Applying the rule for IN conditions

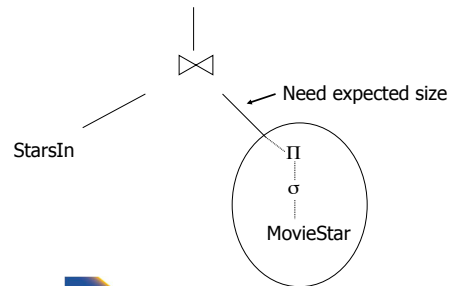
Example: Improved Logical Query Plan



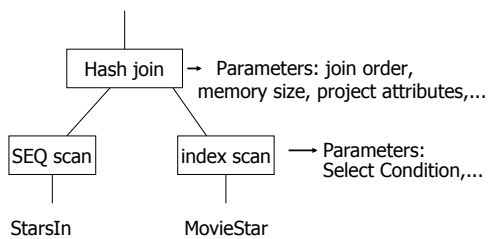
Question:
Push project to StarsIn?

Fig. 7.20: An improvement on fig. 7.18.

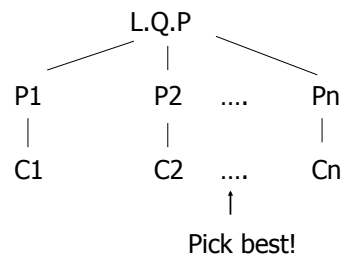
Example: Estimate Result Sizes



Example: One Physical Plan



Example: Estimate costs





CS 525: Advanced Database Organisation

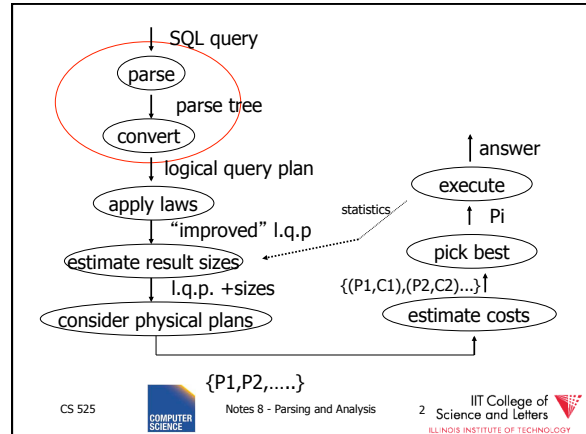
08: Query Processing Parsing and Analysis

Boris Glavic

Slides: adapted from a [course](#) taught by [Hector Garcia-Molina](#), Stanford InfoLab




CS 525  Notes 8 - Parsing and Analysis 1 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY




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

CS 525  Notes 8 - Parsing and Analysis 3 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

CS 525  Notes 8 - Parsing and Analysis 4 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

Parsing

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

CS 525  Notes 8 - Parsing and Analysis 6 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

CS 525



Notes 8 - Parsing and Analysis

7

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)

CS 525

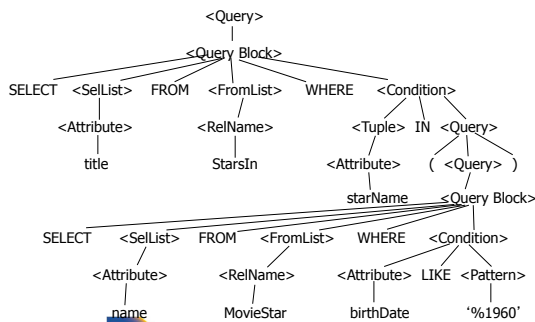


Notes 8 - Parsing and Analysis

8

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Parse Tree



CS 525



Notes 8 - Parsing and Analysis

9

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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>
```

CS 525



Notes 8 - Parsing and Analysis

10

IIT College of Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## Query Blocks

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

**SELECT** (1 + 2) AS result

```
result
3
```

CS 525



Notes 8 - Parsing and Analysis

11

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

CS 525



Notes 8 - Parsing and Analysis

12

IIT College of Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## SELECT clause - example

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

| person |        | result  |      |
|--------|--------|---------|------|
| name   | gender | initial | name |
| Joe    | male   | J       | Joe  |
| Jim    | male   | J       | Jim  |

CS 525



Notes 8 - Parsing and Analysis

13

IIT College of Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## SELECT clause – set functions

- Function `extrChar(string)`

```
SELECT extrChar(p.name) AS n
FROM person p
```

| person |        |
|--------|--------|
| name   | gender |
| Joe    | male   |
| Jim    | male   |

result

| n |
|---|
| J |
| o |
| e |
| J |
| i |
| m |

CS 525



Notes 8 - Parsing and Analysis

14

IIT College of Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## SELECT clause – DISTINCT

```
SELECT DISTINCT gender
FROM person p
```

| person |        | result |
|--------|--------|--------|
| name   | gender | gender |
| Joe    | male   | male   |
| Jim    | male   |        |

CS 525



Notes 8 - Parsing and Analysis

15

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

CS 525



Notes 8 - Parsing and Analysis

16

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'
```

CS 525



Notes 8 - Parsing and Analysis

17

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 attributes
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
```

CS 525



Notes 8 - Parsing and Analysis

18

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

CS 525



Notes 8 - Parsing and Analysis

19

IIT College of  
Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## FROM clause examples

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

**result**

| a |
|---|
| 1 |
| 2 |
| 3 |

CS 525



Notes 8 - Parsing and Analysis

20

IIT College of  
Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## 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    |
| ...     | ...     |

CS 525



Notes 8 - Parsing and Analysis

21

IIT College of  
Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## FROM clause - correlation

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

CS 525



Notes 8 - Parsing and Analysis

22

IIT College of  
Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## Correlation - Example

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

**employee**

| name | dep       |
|------|-----------|
| Joe  | IT        |
| Jim  | Marketing |
| ...  | ...       |

**result**

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

CS 525



Notes 8 - Parsing and Analysis

23

IIT College of  
Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## Correlation - Example

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

**employee**

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

**result**

| name |
|------|
| Jim  |

CS 525



Notes 8 - Parsing and Analysis

24

IIT College of  
Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## WHERE clause

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

CS 525



Notes 8 - Parsing and Analysis

25



## 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

CS 525



Notes 8 - Parsing and Analysis

26



## Nested Subqueries

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

CS 525



Notes 8 - Parsing and Analysis

27



## Nested Subqueries Semantics

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

CS 525



Notes 8 - Parsing and Analysis

28



## Scalar subquery

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

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

CS 525



Notes 8 - Parsing and Analysis

29



## Existential Quantification

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

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

CS 525



Notes 8 - Parsing and Analysis

30



## Existential Quantification

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

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

CS 525



Notes 8 - Parsing and Analysis

31

IIT College of  
Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## 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)
```

CS 525



Notes 8 - Parsing and Analysis

32

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)
```

CS 525



Notes 8 - Parsing and Analysis

33

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 |

CS 525



Notes 8 - Parsing and Analysis

34

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

CS 525



Notes 8 - Parsing and Analysis

35

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

CS 525



Notes 8 - Parsing and Analysis

36

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

CS 525



Notes 8 - Parsing and Analysis

37

IIT College of  
Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

```
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)
```

### result

| numP | numF |
|------|------|
| 1    | 1    |
| 2    | 2    |

### friends

| name  | with  |
|-------|-------|
| Joe   | Jim   |
| Joe   | Peter |
| Jim   | Joe   |
| Jim   | Peter |
| Peter | Joe   |

CS 525



Notes 8 - Parsing and Analysis

38

IIT College of  
Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## HAVING clause

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

CS 525



Notes 8 - Parsing and Analysis

39

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
```

CS 525



Notes 8 - Parsing and Analysis

40

IIT College of  
Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## ORDER BY clause

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

CS 525



Notes 8 - Parsing and Analysis

41

IIT College of  
Science and Letters  
ILLINOIS INSTITUTE OF TECHNOLOGY

## 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
```

CS 525



Notes 8 - Parsing and Analysis

42

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

CS 525



Notes 8 - Parsing and Analysis

43

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

44

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

CS 525



Notes 8 - Parsing and Analysis

45

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?

CS 525



Notes 8 - Parsing and Analysis

46

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Database Catalog

- Stores information about database objects
- Aliases:
 - Information Schema
 - System tables
 - Data Dictionary

CS 525



Notes 8 - Parsing and Analysis

47

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

CS 525



Notes 8 - Parsing and Analysis

48

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
- ...

CS 525



Notes 8 - Parsing and Analysis

49

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

CS 525



Notes 8 - Parsing and Analysis

50

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 than SELECT clause expression can only reference group by expressions or aggregated values

CS 525



Notes 8 - Parsing and Analysis

51

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 8 - Parsing and Analysis

52

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
```

CS 525



Notes 8 - Parsing and Analysis

53

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

CS 525



Notes 8 - Parsing and Analysis

54

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

CS 525



Notes 8 - Parsing and Analysis

55

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

56

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 8 - Parsing and Analysis

57

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Relational Algebra

- 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

CS 525



Notes 8 - Parsing and Analysis

58

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Other Internal Representations

- Practical implementations
 - Mostly following structure of SQL query blocks
 - Store data type and meta-data (where necessary)

CS 525



Notes 8 - Parsing and Analysis

59

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

CS 525



Notes 8 - Parsing and Analysis

60

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

CS 525



Notes 8 - Parsing and Analysis

61

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- 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)

CS 525



Notes 8 - Parsing and Analysis

62

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

CS 525



Notes 8 - Parsing and Analysis

63

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Bag semantics notation

- We use \mathbf{t}^m to denote tuple t appears with multiplicity m

CS 525



Notes 8 - Parsing and Analysis

64

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Set- vs. Bag semantics

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

CS 525



Notes 8 - Parsing and Analysis

65

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Operators

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

CS 525



Notes 8 - Parsing and Analysis

66

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Selection

- Syntax: $\sigma_c(R)$
 - R is input
 - C is a condition
- Semantics:
 - Return all tuples that match condition C
 - Set: $\{ t \mid t \in R \text{ AND } t \text{ fulfills } C \}$
 - Bag: $\{ t^i \mid t^i \in R \text{ AND } t \text{ fulfills } C \}$

CS 525



Notes 8 - Parsing and Analysis

67

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

CS 525



Notes 8 - Parsing and Analysis

68

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Renaming

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

CS 525



Notes 8 - Parsing and Analysis

69

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

CS 525



Notes 8 - Parsing and Analysis

70

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Projection

- Syntax: $\Pi_A(R)$
 - R is input
 - A is a list of projection expressions
 - Standard: only attribute in A
- Semantics:
 - Project all inputs on projection expressions
 - Set: $\{ t.A \mid t \in R \}$
 - Bag: $\{ (t.A)^i \mid t^i \in R \}$

CS 525



Notes 8 - Parsing and Analysis

71

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Projection Example

- $\Pi_b(R)$

R	
a	b
1	13
3	12
6	14

Result
b
13
12
14

CS 525



Notes 8 - Parsing and Analysis

72

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Cross Product

- Syntax: $R \times S$
 - R and S are inputs
- Semantics:
 - All combinations of tuples from R and S
 - = mathematical definition of cross product
 - Set: $\{ (t,s) \mid t \in R \text{ AND } s \in S \}$
 - Bag: $\{ (t,s)^{n^*m} \mid t^n \in R \text{ AND } s^m \in S \}$

CS 525



Notes 8 - Parsing and Analysis

73

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Cross Product Example

- $R \times S$

R		S		Result			
a	b	c	d	a	b	c	d
1	13	a	5	1	13	a	5
		b	3	1	13	b	3
		c	4	1	13	c	4
3	12			3	12	a	5
				3	12	b	3
				3	12	c	4

CS 525



Notes 8 - Parsing and Analysis

74

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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: $\{ (t,s)^{n^*m} \mid t^n \in R \text{ AND } s^m \in S \text{ AND } (t,s) \text{ matches } C \}$

CS 525



Notes 8 - Parsing and Analysis

75

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Join Example

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

R		S		Result			
a	b	c	d	a	b	c	d
1	13	a	5				
		b	3				
		c	4				
3	12			3	12	b	3

CS 525



Notes 8 - Parsing and Analysis

76

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: $\{ (t,s,C) \mid t \in R \text{ AND } s \in S \text{ AND } t.A=s.A \}$
 - Bag: $\{ (t,s,C)^{n^*m} \mid t^n \in R \text{ AND } s^m \in S \text{ AND } t.A=s.A \}$

CS 525



Notes 8 - Parsing and Analysis

77

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Natural Join Example

- $R \bowtie S$

R		S		Result		
a	b	c	a	a	b	c
1	13	a	5			
		b	3			
		c	4			
3	12			3	12	b

CS 525



Notes 8 - Parsing and Analysis

78

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
 - $t \in R$ without matches fill S attributes with NULL
 - $\{ (t,s) \mid t \in R \text{ AND } s \in S \text{ AND } (t,s) \text{ matches } C \}$
 - union
 - $\{ (t, \text{NULL}(S)) \mid t \in R \text{ AND NOT exists } s \in S: (t,s) \text{ matches } C \}$

CS 525



Notes 8 - Parsing and Analysis

79

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Left-outer Join Example

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

R		S		Result			
a	b	c	d	a	b	c	d
1	13	a	5	1	13	NULL	NULL
3	12	b	3	3	12	b	3
		c	4				

CS 525



Notes 8 - Parsing and Analysis

80

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
 - $s \in S$ without matches fill R attributes with NULL
 - $\{ (t,s) \mid t \in R \text{ AND } s \in S \text{ AND } (t,s) \text{ matches } C \}$
 - union
 - $\{ (\text{NULL}(R),s) \mid s \in S \text{ AND NOT exists } t \in R: (t,s) \text{ matches } C \}$

CS 525



Notes 8 - Parsing and Analysis

81

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Right-outer Join Example

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

R		S		Result			
a	b	c	d	a	b	c	d
1	13	a	5	NULL	NULL	a	5
3	12	b	3	3	12	b	3
		c	4	NULL	NULL	c	4

CS 525



Notes 8 - Parsing and Analysis

82

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Full-outer Join

- Syntax: $R \bowtie_C S$
 - R and S are inputs and C is condition
- Semantics:
 - $\{ (t,s) \mid t \in R \text{ AND } s \in S \text{ AND } (t,s) \text{ matches } C \}$
 - union
 - $\{ (\text{NULL}(R),s) \mid s \in S \text{ AND NOT exists } t \in R: (t,s) \text{ matches } C \}$
 - union
 - $\{ (t, \text{NULL}(S)) \mid t \in R \text{ AND NOT exists } s \in S: (t,s) \text{ matches } C \}$

CS 525



Notes 8 - Parsing and Analysis

83

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Full-outer Join Example

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

R		S		Result			
a	b	c	d	a	b	c	d
1	13	a	5	1	13	NULL	NULL
3	12	b	3	NULL	NULL	a	5
		c	4	3	12	b	3
				NULL	NULL	c	4

CS 525



Notes 8 - Parsing and Analysis

84

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Semijoin

- Syntax: $R \bowtie S$ and $R \ltimes S$
 - R and S are inputs
- Semantics:
 - All tuples from R that have a matching tuple from relation S on the common attributes A

$$\{ t \mid t \in R \text{ AND exists } s \in S: t.A = s.A \}$$

CS 525



Notes 8 - Parsing and Analysis

85

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Semijoin Example

- $R \ltimes S$

R		S		Result	
a	b	c	a	a	b
1	13	a	5	3	12
3	12	b	3		
		c	4		

CS 525



Notes 8 - Parsing and Analysis

86

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Antijoin

- Syntax: $R \not\bowtie S$
 - R and S are inputs
- Semantics:
 - All tuples from R that have no matching tuple from relation S on the common attributes A

$$\{ t \mid t \in R \text{ AND NOT exists } s \in S: t.A = s.A \}$$

CS 525



Notes 8 - Parsing and Analysis

87

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Antijoin Example

- $R \not\bowtie S$

R		S		Result	
a	b	c	a	a	b
1	13	a	5	1	13
3	12	b	3		
		c	4		

CS 525



Notes 8 - Parsing and Analysis

88

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Aggregation

- Syntax: $\alpha_A \alpha_G (R)$
 - 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
 - $\{ (t.G, \text{agg}(G(t)) \mid t \in R \}$
 - $G(t) = \{ t' \mid t' \in R \text{ AND } t'.G = t.G \}$

CS 525



Notes 8 - Parsing and Analysis

89

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Aggregation Example

- $\alpha_{\text{sum}(a)} \alpha_G (R)$

R		Result	
a	b	sum(a)	b
1	1	4	1
3	1	9	2
6	2		
3	2		

CS 525



Notes 8 - Parsing and Analysis

90

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate Removal

- Syntax: $\delta(R)$
 - R is input
- Semantics:
 - Remove duplicates from input
 - Set: N/A
 - Bag: $\{t^i \mid t^i \in R\}$

CS 525



Notes 8 - Parsing and Analysis

91

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Duplicate Removal Example

- $\delta(R)$

R	
a	b
1	13
1	13
6	14

Result	
a	b
1	13
6	14

CS 525



Notes 8 - Parsing and Analysis

92

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

CS 525



Notes 8 - Parsing and Analysis

93

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Union

- Syntax: $R \cup S$
 - R and S are union-compatible inputs
- Semantics:
 - Set: $\{t \mid t \in R \text{ OR } t \in S\}$
 - Bag: $\{(t,s)^{n+m} \mid t^i \in R \text{ AND } s^m \in S\}$
 - Assumption t^i with $n < 1$ for tuple not in relation

CS 525



Notes 8 - Parsing and Analysis

94

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Union Example

- $R \cup S$

R
a
1
3

S
b
1
2
3

Result
a
1
2
3
1
3

CS 525



Notes 8 - Parsing and Analysis

95

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Intersection

- Syntax: $R \cap S$
 - R and S are union-compatible inputs
- Semantics:
 - Set: $\{t \mid t \in R \text{ AND } t \in S\}$
 - Bag: $\{(t,s)^{\min(n,m)} \mid t^i \in R \text{ AND } s^m \in S\}$

CS 525



Notes 8 - Parsing and Analysis

96

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Intersection Example

- $R \cap S$

R
a
1
3

S
b
1
2
3

Result
a
1
3

CS 525



Notes 8 - Parsing and Analysis

97

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

CS 525



Notes 8 - Parsing and Analysis

98

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

CS 525



Notes 8 - Parsing and Analysis

99

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

CS 525



Notes 8 - Parsing and Analysis

100

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 aggregation
 - **DISTINCT** into duplicate removal

CS 525



Notes 8 - Parsing and Analysis

101

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

CS 525



Notes 8 - Parsing and Analysis

102

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



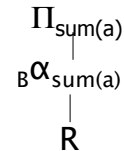
Notes 8 - Parsing and Analysis

103

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Relational Algebra Translation

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



CS 525



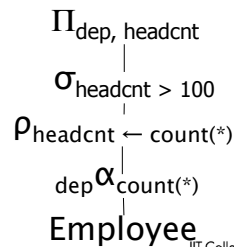
Notes 8 - Parsing and Analysis

104

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Relational Algebra Translation

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



CS 525



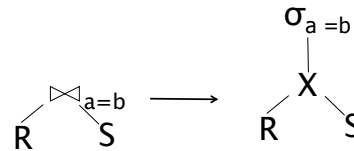
Notes 8 - Parsing and Analysis

105

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Relational Algebra Translation

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



CS 525



Notes 8 - Parsing and Analysis

106

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Parsing and Analysis Summary

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

CS 525





Notes 8 - Parsing and Analysis

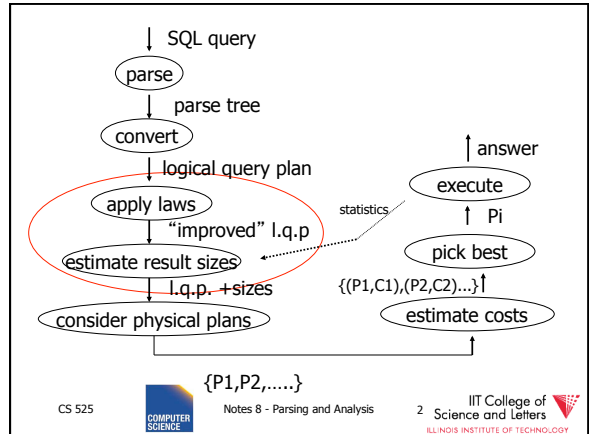
107

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




CS 525

Notes 9 - Logical Optimization
1 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY




Query Optimization

- Relational algebra level
- Detailed query plan level

CS 525

Notes 9 - Logical Optimization
3 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY


Query Optimization

- Relational algebra level
- Detailed query plan level
 - Estimate Costs
 - without indexes
 - with indexes
 - Generate and compare plans

CS 525

Notes 9 - Logical Optimization
4 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

CS 525

Notes 9 - Logical Optimization
5 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

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

CS 525

Notes 9 - Logical Optimization
6 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: Natural joins & cross products & union

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

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

CS 525



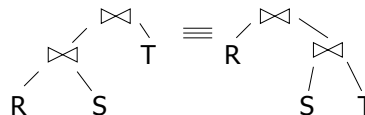
Notes 9 - Logical Optimization

7

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Note:

- Carry attribute names in results, so order is not important
- Can also write as trees, e.g.:



CS 525



Notes 9 - Logical Optimization

8

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: Natural joins & cross products & union

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

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

$$R \times S = S \times R$$

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

$$R \cup S = S \cup R$$

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

CS 525



Notes 9 - Logical Optimization

9

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: Selects

$$\sigma_{p_1 \wedge p_2}(R) =$$

$$\sigma_{p_1 \vee p_2}(R) =$$

CS 525



Notes 9 - Logical Optimization

10

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: Selects

$$\sigma_{p_1 \wedge p_2}(R) = \sigma_{p_1} [\sigma_{p_2}(R)]$$

$$\sigma_{p_1 \vee p_2}(R) = [\sigma_{p_1}(R)] \cup [\sigma_{p_2}(R)]$$

CS 525



Notes 9 - Logical Optimization

11

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Bags vs. Sets

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

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

$$R \cup S = ?$$

CS 525



Notes 9 - Logical Optimization

12

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Bags vs. Sets

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

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

$R \cup S = ?$

- Option 1 SUM
 $R \cup S = \{a,a,b,b,b,b,c,c,c,d\}$
- Option 2 MAX
 $R \cup S = \{a,a,b,b,b,c,c,d\}$

CS 525



Notes 9 - Logical Optimization

13

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Option 2 (MAX) makes this rule work:

$$\sigma_{p_1 \vee p_2}(R) = \sigma_{p_1}(R) \cup \sigma_{p_2}(R)$$

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

P_1 satisfied by a,b ; P_2 satisfied by b,c

CS 525



Notes 9 - Logical Optimization

14

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Option 2 (MAX) makes this rule work:

$$\sigma_{p_1 \vee p_2}(R) = \sigma_{p_1}(R) \cup \sigma_{p_2}(R)$$

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

P_1 satisfied by a,b ; P_2 satisfied by b,c

$$\sigma_{p_1 \vee p_2}(R) = \{a,a,b,b,b,c\}$$

$$\sigma_{p_1}(R) = \{a,a,b,b,b\}$$

$$\sigma_{p_2}(R) = \{b,b,b,c\}$$

$$\sigma_{p_1}(R) \cup \sigma_{p_2}(R) = \{a,a,b,b,b,c\}$$

CS 525



Notes 9 - Logical Optimization

15

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

“Sum” option makes more sense:

Senators (.....)

Rep (.....)

$T_1 = \pi_{yr,state} \text{Senators}; T_2 = \pi_{yr,state} \text{Reps}$

T1	Yr	State	T2	Yr	State
	97	CA		99	CA
	99	CA		99	CA
	98	AZ		98	CA

Union?

CS 525



Notes 9 - Logical Optimization

16

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Executive Decision

- > Use “SUM” option for bag unions
- > Some rules cannot be used for bags

CS 525



Notes 9 - Logical Optimization

17

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: Project

Let: $X =$ set of attributes

$Y =$ set of attributes

$$XY = X \cup Y$$

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

CS 525



Notes 9 - Logical Optimization

18

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

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

CS 525



Notes 9 - Logical Optimization

19



Rules: Project

Let: X = set of attributes
Y = set of attributes
XY = X U Y

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

CS 525



Notes 9 - Logical Optimization

20



Rules: $\sigma + \bowtie$ combined

Let p = predicate with only R attribs
q = predicate with only S attribs
m = predicate with only R,S attribs

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

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

CS 525



Notes 9 - Logical Optimization

21



Rules: $\sigma + \bowtie$ combined

Let p = predicate with only R attribs
q = predicate with only S attribs
m = predicate with only R,S attribs

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

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

CS 525



Notes 9 - Logical Optimization

22



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

Some Rules can be Derived:

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

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

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

CS 525



Notes 9 - Logical Optimization

23



Do one:

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

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

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

CS 525



Notes 9 - Logical Optimization

24



--> Derivation for first one:

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

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

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

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

CS 525



Notes 9 - Logical Optimization

25

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)

$$\pi_x [\sigma_p (R)] =$$

CS 525



Notes 9 - Logical Optimization

26

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)

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

CS 525



Notes 9 - Logical Optimization

27

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)

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

CS 525



Notes 9 - Logical Optimization

28

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: π, \bowtie combined

Let x = subset of R attributes

y = subset of S attributes

z = intersection of R,S attributes

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

CS 525



Notes 9 - Logical Optimization

29

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rules: π, \bowtie combined

Let x = subset of R attributes

y = subset of S attributes

z = intersection of R,S attributes

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

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

CS 525



Notes 9 - Logical Optimization

30

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 9 - Logical Optimization

31

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

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

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

CS 525



Notes 9 - Logical Optimization

32

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rules for σ, π combined with X

similar...

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

CS 525



Notes 9 - Logical Optimization

33

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rules σ, \cup combined:

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

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

CS 525



Notes 9 - Logical Optimization

34

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Which are “good” transformations?

- $\sigma_{p1 \wedge p2} (R) \rightarrow \sigma_{p1} [\sigma_{p2} (R)]$
- $\sigma_p (R \bowtie S) \rightarrow [\sigma_p (R)] \bowtie S$
- $R \bowtie S \rightarrow S \bowtie R$
- $\pi_x [\sigma_p (R)] \rightarrow \pi_x \{ \sigma_p [\pi_{xz} (R)] \}$

CS 525



Notes 9 - Logical Optimization

35

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Conventional wisdom:
do projects early

Example: $R(A,B,C,D,E)$ $x=\{E\}$
 $P: (A=3) \wedge (B=\text{“cat”})$

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

CS 525

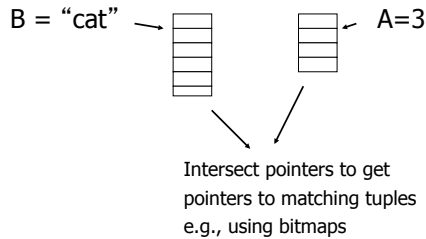


Notes 9 - Logical Optimization

36

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

But What if we have A, B indexes?



CS 525



Notes 9 - Logical Optimization

37

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Bottom line:

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

CS 525



Notes 9 - Logical Optimization

38

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

More transformations

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

CS 525



Notes 9 - Logical Optimization

39

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 be the same
 - Called Equivalence classes
- Example: $R(a,b), S(c,d)$

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

CS 525



Notes 9 - Logical Optimization

40

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

CS 525



Notes 9 - Logical Optimization

41

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Summary Equivalences

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

CS 525



Notes 9 - Logical Optimization

42

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

CS 525



Notes 9 - Logical Optimization

43

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- Estimating cost of query plan

- (1) Estimating size of results
- (2) Estimating # of IOs

CS 525



Notes 9 - Logical Optimization

44

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

CS 525



Notes 9 - Logical Optimization

45

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

CS 525



Notes 9 - Logical Optimization

46

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

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

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

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

CS 525



Notes 9 - Logical Optimization

47

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Size estimates for $W = R1 \times R2$

$$T(W) =$$

$$S(W) =$$

CS 525



Notes 9 - Logical Optimization

48

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Size estimates for $W = R_1 \times R_2$

$$T(W) = T(R_1) \times T(R_2)$$

$$S(W) = S(R_1) + S(R_2)$$

CS 525



Notes 9 - Logical Optimization

49

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

$$S(W) = S(R)$$

$$T(W) = ?$$

CS 525



Notes 9 - Logical Optimization

50

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	

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

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

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

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

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

CS 525



Notes 9 - Logical Optimization

51

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	

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

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

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

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

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

CS 525



Notes 9 - Logical Optimization

52

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Assumption:

Values in select expression $Z = val$ are uniformly distributed over possible $V(R,Z)$ values.

CS 525



Notes 9 - Logical Optimization

53

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Alternate Assumption:

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

CS 525



Notes 9 - Logical Optimization

54

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	

Alternate assumption
 $V(R,A)=3 \quad \text{DOM}(R,A)=10$
 $V(R,B)=1 \quad \text{DOM}(R,B)=10$
 $V(R,C)=5 \quad \text{DOM}(R,C)=10$
 $V(R,D)=4 \quad \text{DOM}(R,D)=10$

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

$C=\text{val} \Rightarrow T(W) = (1/10)1 + (1/10)1 + \dots$
 $= (5/10) = 0.5$

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

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

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
 $V(R,A)=3 \quad \text{DOM}(R,A)=10$
 $V(R,B)=1 \quad \text{DOM}(R,B)=10$
 $V(R,C)=5 \quad \text{DOM}(R,C)=10$
 $V(R,D)=4 \quad \text{DOM}(R,D)=10$

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

Selection cardinality

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

$$SC(R,A) = \begin{cases} \frac{T(R)}{V(R,A)} \\ \frac{T(R)}{\text{DOM}(R,A)} \end{cases}$$

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

$T(W) = ?$

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

$T(W) = ?$

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

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

$T(W) = ?$

- Solution # 1:

$$T(W) = T(R)/2$$

- Solution # 2:

$$T(W) = T(R)/3$$

CS 525



Notes 9 - Logical Optimization

61

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- Solution # 3: Estimate values in range

Example R

	Z

Min=1
↓
Max=20

$V(R,Z)=10$

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

CS 525



Notes 9 - Logical Optimization

62

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- Solution # 3: Estimate values in range

Example R

	Z

Min=1
↓
Max=20

$V(R,Z)=10$

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

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

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

CS 525



Notes 9 - Logical Optimization

63

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Equivalently:

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

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

CS 525



Notes 9 - Logical Optimization

64

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Size estimate for $W = R1 \bowtie R2$

Let $x = \text{attributes of } R1$

$y = \text{attributes of } R2$

CS 525



Notes 9 - Logical Optimization

65

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Size estimate for $W = R1 \bowtie R2$

Let $x = \text{attributes of } R1$

$y = \text{attributes of } R2$

Case 1

$$X \cap Y = \emptyset$$

Same as $R1 \times R2$

CS 525




Notes 9 - Logical Optimization

66

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
----	---	---	---	----	---	---

CS 525  Notes 9 - Logical Optimization 67 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

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

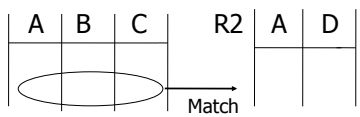
R1	A	B	C	R2	A	D
----	---	---	---	----	---	---


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

CS 525  Notes 9 - Logical Optimization 68 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Computing T(W) when $V(R1,A) \leq V(R2,A)$

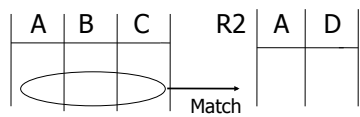
R1	A	B	C	R2	A	D
----	---	---	---	----	---	---

Take 1 tuple  Match

CS 525  Notes 9 - Logical Optimization 69 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


Computing T(W) when $V(R1,A) \leq V(R2,A)$

R1	A	B	C	R2	A	D
----	---	---	---	----	---	---

Take 1 tuple  Match


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

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

CS 525  Notes 9 - Logical Optimization 70 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


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

[A is common attribute]

CS 525  Notes 9 - Logical Optimization 71 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

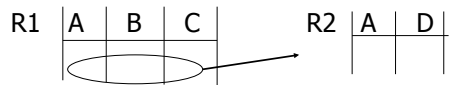
In general $W = R1 \bowtie R2$

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

CS 525  Notes 9 - Logical Optimization 72 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Case 2 with alternate assumption

Values uniformly distributed over domain



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

Assume the same

In all cases:

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

size of attribute A

Using similar ideas,
we can estimate sizes of:

- $\Pi_{AB} (R)$
- $\sigma_{A=a \wedge B=b} (R)$
- $R \bowtie S$ with common attribs. A,B,C
- Union, intersection, diff,

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

E.g. $W = [\sigma_{A=a} (R1)] \bowtie R2$

Treat as relation U

$$T(U) = T(R1)/V(R1,A) \quad S(U) = S(R1)$$

Also need V (U, *) !!

To estimate Vs

E.g., $U = \sigma_{A=a} (R1)$
Say R1 has attribs A,B,C,D

$$V(U, A) =$$

$$V(U, B) =$$

$$V(U, C) =$$

$$V(U, D) =$$

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	

$$V(R1,A)=3$$

$$V(R1,B)=1$$

$$V(R1,C)=5$$

$$V(R1,D)=3$$

$$U = \sigma_{A=a}(R1)$$

$$V(U,A) = 1 \quad V(U,B) = 1 \quad V(U,C) = \frac{T(R1)}{V(R1,A)}$$

$V(D,U)$... somewhere in between

CS 525



Notes 9 - Logical Optimization

79

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Possible Guess $U = \sigma_{A=a}(R)$

$$V(U,A) = 1$$

$$V(U,B) = V(R,B)$$

CS 525



Notes 9 - Logical Optimization

80

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

For Joins $U = R1(A,B) \bowtie R2(A,C)$

$$V(U,A) = \min \{ V(R1, A), V(R2, A) \}$$

$$V(U,B) = V(R1, B)$$

$$V(U,C) = V(R2, C)$$

CS 525



Notes 9 - Logical Optimization

81

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
----	--------------	------------	-------------

CS 525



Notes 9 - Logical Optimization

82

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Partial Result: $U = R1 \bowtie R2$

$$T(U) = \frac{1000 \times 2000}{200} \quad V(U,A) = 50$$

$$V(U,B) = 100$$

$$V(U,C) = 300$$

CS 525



Notes 9 - Logical Optimization

83

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

$Z = U \bowtie R3$

$$T(Z) = \frac{1000 \times 2000 \times 3000}{200 \times 300} \quad V(Z,A) = 50$$

$$V(Z,B) = 100$$

$$V(Z,C) = 90$$

$$V(Z,D) = 500$$

CS 525



Notes 9 - Logical Optimization

84

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 9 - Logical Optimization

85



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

CS 525

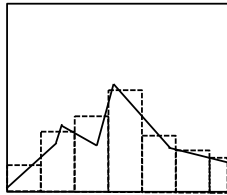


Notes 9 - Logical Optimization

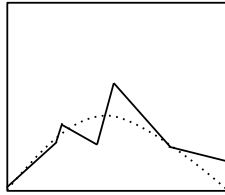
86



Histograms



Parameterized Distribution



CS 525



Notes 9 - Logical Optimization

87



Maintaining Statistics

- Use separate command that triggers statistics collection
 - Postgres: ANALYZE
- During query processing
 - Overhead for queries
- Use Sampling?

CS 525

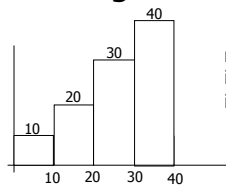


Notes 9 - Logical Optimization

88



Estimating Result Size using Histograms



number of tuples
in R with A value
in given range

$$\sigma_{A=val}(R) = ?$$

CS 525



Notes 9 - Logical Optimization

89



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



Notes 9 - Logical Optimization

90



Join Size using Histograms

- $R \bowtie S$
- Use

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

- Apply for each bucket

CS 525



Notes 9 - Logical Optimization

91

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Join Size using Histograms

- $V(R1,A) = V(R2,A) = \text{bucket size } |B|$

$$T(W) = \sum_{\text{buckets}} \frac{\#B(R2) \#B(R1)}{|B|}$$

CS 525



Notes 9 - Logical Optimization

92

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525

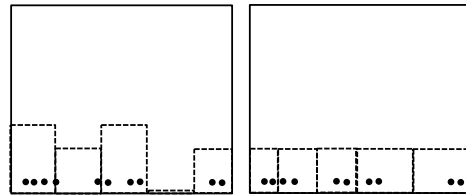


Notes 9 - Logical Optimization

93

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Equi-width vs. Equi-depth



CS 525



Notes 9 - Logical Optimization

94

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Construct Equi-depth Histograms

- Sort input
- Determine size of buckets
 - $\# \text{bucket} / \# \text{tuples}$
- 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]

CS 525



Notes 9 - Logical Optimization

95

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Advanced Techniques

- Wavelets
- Approximate Histograms
- Sampling Techniques
- Compressed Histograms

CS 525



Notes 9 - Logical Optimization

96

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?)

CS 525



Notes 9 - Logical Optimization

97

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

- Estimating cost of query plan
 - Estimating size of results ← done!
 - Estimating # of IOs ← next...
 - Operator Implementations
- Generate and compare plans

CS 525



Notes 9 - Logical Optimization


98

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY



CS 525: Advanced Database Organization

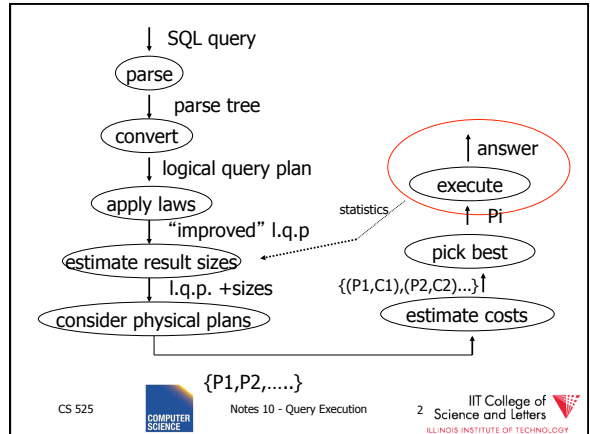
10: Query Execution

Boris Glavic





Slides: adapted from a [course](#) taught by [Hector Garcia-Molina](#), Stanford InfoLab

CS 525  Notes 10 - Query Execution 1  IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY





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

CS 525  Notes 10 - Query Execution 3  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

CS 525  Notes 10 - Query Execution 4  IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY



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

CS 525  Notes 10 - Query Execution 5  IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Estimating IOs:

- Count # of disk blocks that must be read (or written) to execute query plan

CS 525  Notes 10 - Query Execution 6  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

CS 525



Notes 10 - Query Execution

7

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

CS 525

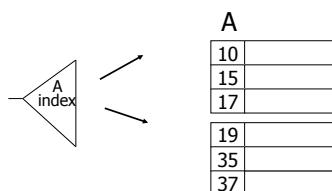


Notes 10 - Query Execution

8

Clustered index

Index that allows tuples to be read in an order that corresponds to physical order



CS 525



Notes 10 - Query Execution

9

Operators Overview

- (External) Sorting
- Joins (Nested Loop, Merge, Hash, ...)
- Aggregation (Sorting, Hash)
- Selection, Projection (Index, Scan)
- Union, Set Difference
- Intersection
- Duplicate Elimination

CS 525



Notes 10 - Query Execution

10

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

CS 525



Notes 10 - Query Execution

11

Execution Strategies

- Compiled
 - Translate into C/C++/Assembler code
 - Compile, link, and execute code
- Interpreted
 - Generic operator implementations
 - Generic executor
 - Interprets query plan

CS 525



Notes 10 - Query Execution

12

Virtual Machine Approach

- Implement virtual machine of low-level DBMS operations
- Compile query into machine-code for that machine

CS 525



Notes 10 - Query Execution

13

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 10 - Query Execution

14

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Iterator Model - Interface

- **Open**
 - Prepare operator to read outputs
- **Close**
 - Close operator and clean up
- **Next**
 - Return next result tuple

CS 525

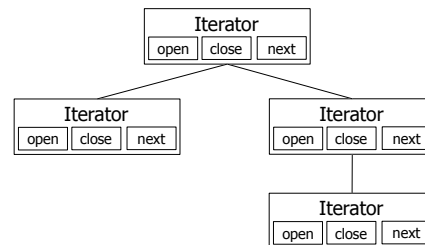


Notes 10 - Query Execution

15

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Query Execution – Iterator Model



CS 525

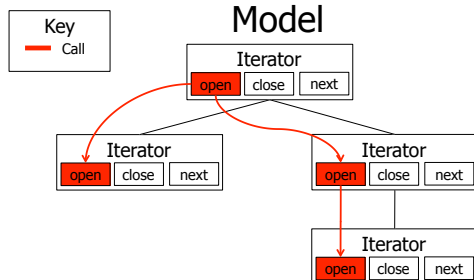


Notes 10 - Query Execution

16

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Query Execution – Iterator Model



CS 525

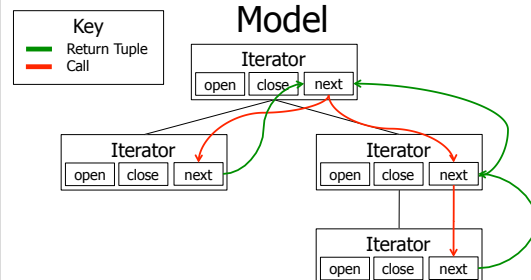


Notes 10 - Query Execution

17

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Query Execution – Iterator Model



CS 525



Notes 10 - Query Execution

18

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Parallelism

- Iterator Model
 - **Pull-based** query execution
- Potential types of parallelism
 - Inter-query (every multiuser system)
 - Intra-operator
 - Inter-operator

CS 525



Notes 10 - Query Execution

19

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

CS 525



Notes 10 - Query Execution

20

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 directly produce results
 - Pipeline-breakers
 - Operators that need to consume the whole input (or large parts) before producing outputs

CS 525



Notes 10 - Query Execution

21

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

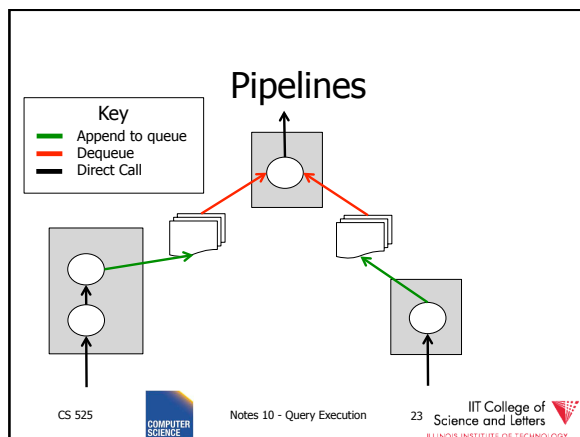
CS 525



Notes 10 - Query Execution

22

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

CS 525



Notes 10 - Query Execution

24

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

CS 525



Notes 10 - Query Execution

25

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-sort
 - Aggregation by sorting
 - Duplicate removal using sorting

CS 525



Notes 10 - Query Execution

26

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

In-memory sorting

- Algorithms from data structures 101
 - Quick sort
 - Merge sort
 - Heap sort
 - Intro sort
 - ...

CS 525



Notes 10 - Query Execution

27

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

External sorting

- Problem:
 - Sort N pages of data with M pages of memory
- Solutions?

CS 525



Notes 10 - Query Execution

28

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

First Idea

- Split data into runs of size M
- Sort each run in memory and write back to disk
 - $\lceil N/M \rceil$ sorted runs of size M
- Now what?



CS 525



Notes 10 - Query Execution

29

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 ?

CS 525



Notes 10 - Query Execution

30

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merging Runs using 3 pages

- Merging 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

CS 525

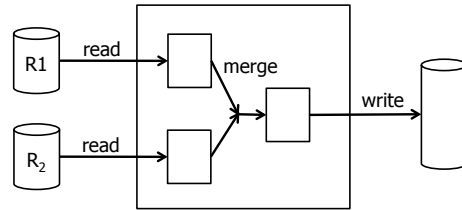


Notes 10 - Query Execution

31

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merging Runs



CS 525



Notes 10 - Query Execution

32

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

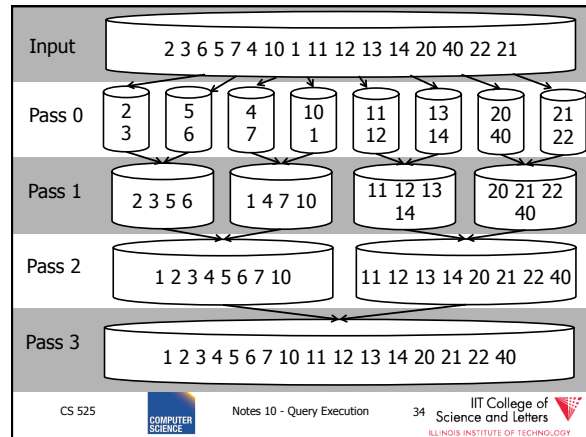
CS 525



Notes 10 - Query Execution

33

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY



CS 525



Notes 10 - Query Execution

34

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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
 - $1 + \lceil \log_{M-1} (B(R) / M) \rceil$ runs
 - $2 B(R) * (1 + \lceil \log_{M-1} (B(R) / M) \rceil)$ I/Os

CS 525

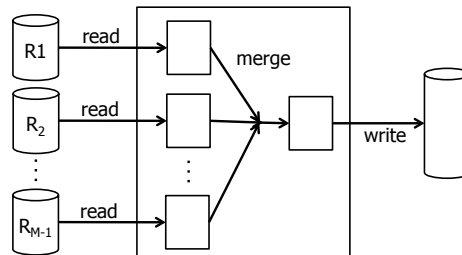


Notes 10 - Query Execution

35

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merging Runs



CS 525



Notes 10 - Query Execution

36

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 10 - Query Execution

37

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

CS 525



Notes 10 - Query Execution

38

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

CS 525



Notes 10 - Query Execution

39

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(log₂(M))**

CS 525



Notes 10 - Query Execution

40

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

CS 525



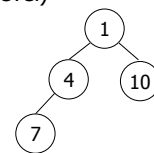
Notes 10 - Query Execution

41

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 }



CS 525



Notes 10 - Query Execution

42

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

CS 525



Notes 10 - Query Execution

43

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

CS 525



Notes 10 - Query Execution

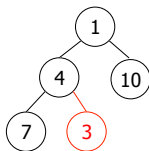
44

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

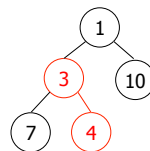
Insertion

- **Insert 3**

Insert at first free position



Restore heap property



CS 525

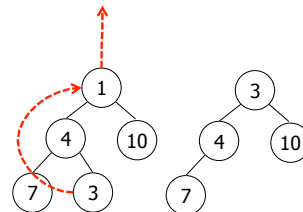


Notes 10 - Query Execution

45

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Dequeue



CS 525

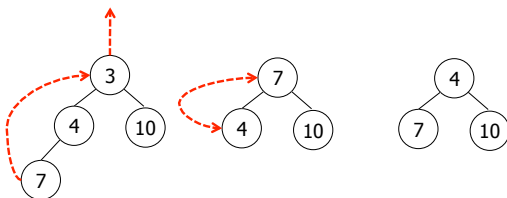


Notes 10 - Query Execution

46

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Dequeue



CS 525



Notes 10 - Query Execution

47

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Min/Max-Heap Complexity

- Head is a complete tree
 - Height is $O(\log_2(n))$
- Insertion
 - Maximal height of the tree switches
 - $\rightarrow O(\log_2(n))$
- Dequeue
 - Maximal height of the tree switches
 - $\rightarrow O(\log_2(n))$

CS 525



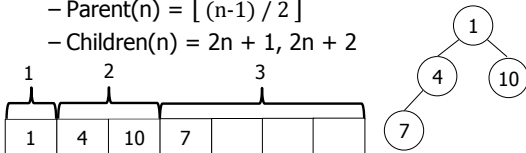
Notes 10 - Query Execution

48

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Min-Heap Implementation

- Full tree
 - Use array to implement tree
- Compute positions
 - $\text{Parent}(n) = \lfloor (n-1) / 2 \rfloor$
 - $\text{Children}(n) = 2n + 1, 2n + 2$



CS 525

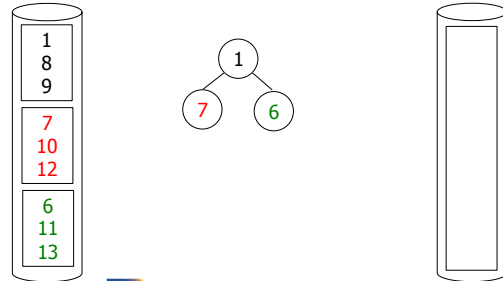


Notes 10 - Query Execution

49

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merging with Priority Queue



CS 525

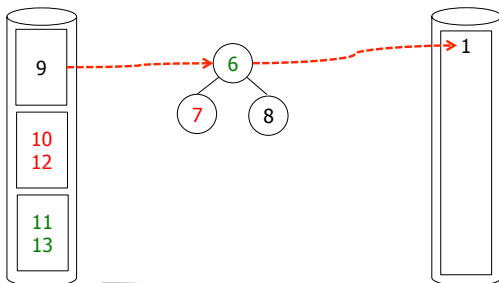


Notes 10 - Query Execution

50

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merging with Priority Queue



CS 525

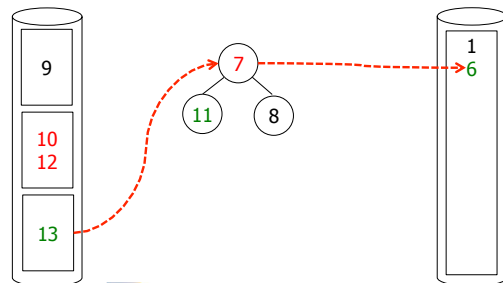


Notes 10 - Query Execution

51

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merging with Priority Queue



CS 525



Notes 10 - Query Execution

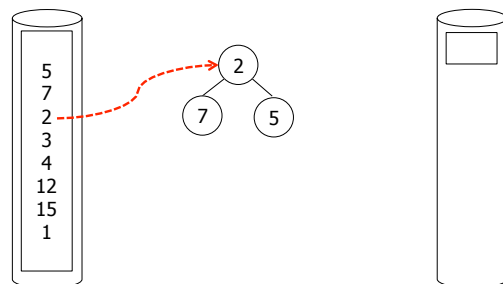
52

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 to current run then write to current run
 - Else create a new run
 - Add new element from input to heap

Using a heap to generate runs



CS 525



Notes 10 - Query Execution

53

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

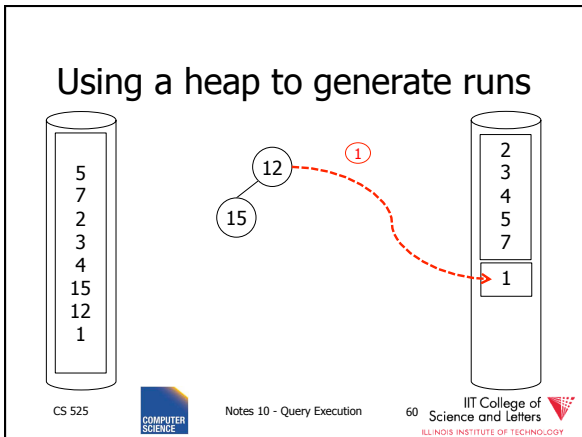
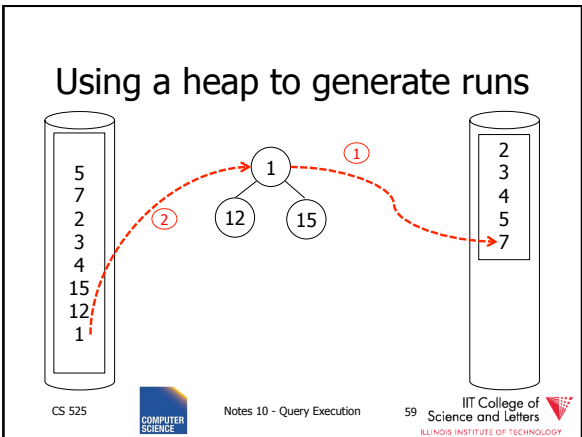
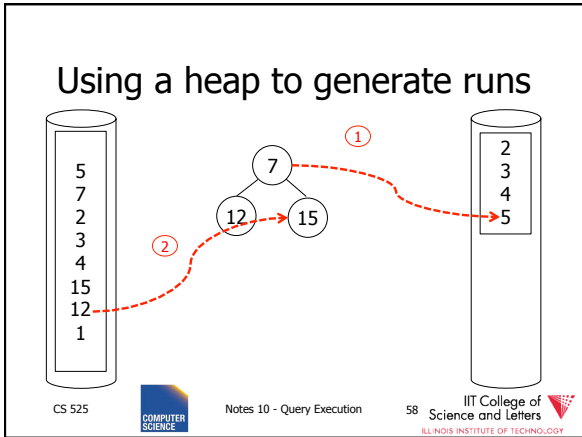
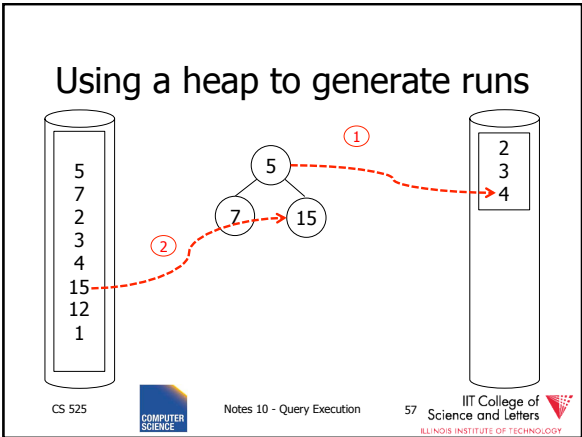
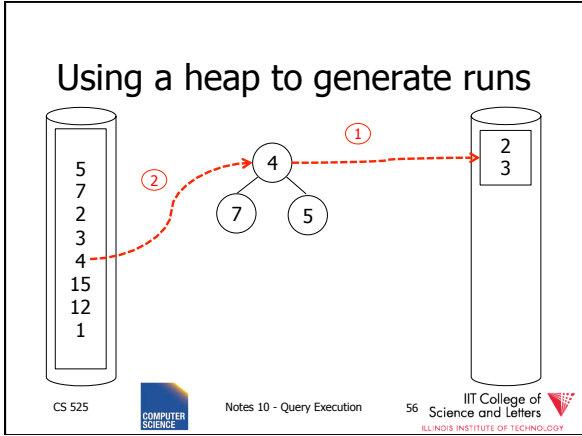
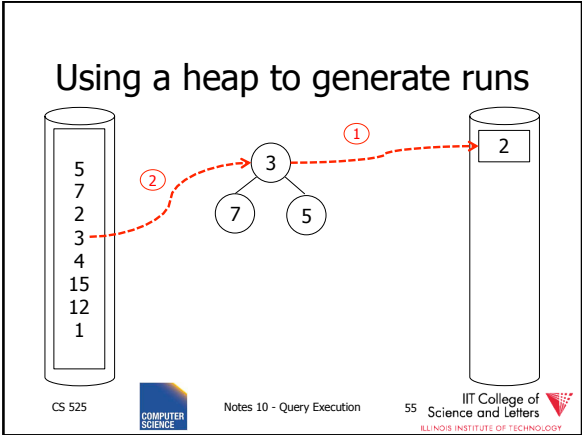
CS 525



Notes 10 - Query Execution

54

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)

CS 525



Notes 10 - Query Execution

61

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)**

CS 525



Notes 10 - Query Execution

62

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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!

CS 525



Notes 10 - Query Execution

63

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

CS 525



Notes 10 - Query Execution

64

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		

CS 525



Notes 10 - Query Execution

65

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

CS 525



Notes 10 - Query Execution

66

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

CS 525



Notes 10 - Query Execution

67

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

CS 525



Notes 10 - Query Execution

68

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

CS 525



Notes 10 - Query Execution

69

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Nested Loop Join (conceptually)

```
for each r ∈ R1 do
  for each s ∈ R2 do
    if (r,s) ⊢ C then output (r,s)
```

Applicable to:

- Any join condition C
- Cross-product

CS 525



Notes 10 - Query Execution

70

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Merge Join (conceptually)

- (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 output Tuples
 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 $i \leftarrow i+1$

Applicable to:

- C is conjunction of equalities or $</>$:
 $A_1 = B_1 \text{ AND } \dots \text{ AND } A_n = B_n$

CS 525



Notes 10 - Query Execution

71

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Procedure Output-Tuples

```
While  $(R_1\{i\}.C = R_2\{j\}.C) \wedge (i \leq T(R_1))$  do
  [jj ← j;
   while  $(R_1\{i\}.C = R_2\{jj\}.C) \wedge (jj \leq T(R_2))$  do
     [output pair  $R_1\{i\}, R_2\{jj\}$ ;
      jj ← jj+1 ]
   i ← i+1 ]
```

CS 525



Notes 10 - Query Execution

72

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

CS 525



Notes 10 - Query Execution

73

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Index nested loop (Conceptually)

```

For each r ∈ R1 do
    [ X ← index (R2, C, r.C)
    for each s ∈ X do
        output (r,s) pair]
    
```

Note: $X \leftarrow \text{index}(\text{rel}, \text{attr}, \text{value})$
then $X = \text{set of rel tuples with attr} = \text{value}$

CS 525



Notes 10 - Query Execution

74

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, \dots, G_k
Buckets for R_2 : H_0, H_1, \dots, H_k

Applicable to:

- C is conjunction of equalities
 $A_1 = B_1 \text{ AND } \dots \text{ AND } A_n = B_n$

CS 525



Notes 10 - Query Execution

75

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, \dots, G_k
Buckets for R_2 : H_0, H_1, \dots, 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

CS 525



Notes 10 - Query Execution

76

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Simple example hash: even/odd

R_1	R_2	Buckets	
2	5	Even: 2 4 8	4 12 8 14
4	4		R_1 R_2
3	12	Odd: 3 5 9	5 3 13 11
5	3		
8	13		
9	8		
	11		
	14		

CS 525



Notes 10 - Query Execution

77

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Factors that affect performance

- (1) Tuples of relation stored physically together?
- (2) Relations sorted by join attribute?
- (3) Indexes exist?

CS 525



Notes 10 - Query Execution

78

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example 1(a) Iteration Join $R_1 \bowtie R_2$

- Relations not contiguous
- Recall $\begin{cases} T(R_1) = 10,000 & T(R_2) = 5,000 \\ S(R_1) = S(R_2) = 1/10 \text{ block} \\ \text{MEM} = 101 \text{ blocks} \end{cases}$

CS 525



Notes 10 - Query Execution

79



Example 1(a)

Nested Loop Join $R_1 \bowtie R_2$

- Relations not contiguous
- Recall $\begin{cases} T(R_1) = 10,000 & T(R_2) = 5,000 \\ S(R_1) = S(R_2) = 1/10 \text{ block} \\ \text{MEM} = 101 \text{ blocks} \end{cases}$

Cost: for each R_1 tuple:

[Read tuple + Read R_2]

Total = $10,000 [1 + 500] = 5,010,000$ IOs

CS 525



Notes 10 - Query Execution

80



- Can we do better?

CS 525



Notes 10 - Query Execution

81



- 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

CS 525



Notes 10 - Query Execution

82



Cost: for each R_1 chunk:

Read chunk: 100 IOs
Read R_2 : $\frac{500 \text{ IOs}}{600}$

CS 525



Notes 10 - Query Execution

83



Cost: for each R_1 chunk:

Read chunk: 100 IOs
Read R_2 : $\frac{500 \text{ IOs}}{600}$

Total = $\frac{1,000}{100} \times 600 = 6,000$ IOs

CS 525



Notes 10 - Query Execution

84



- Can we do better?

CS 525



Notes 10 - Query Execution

85

- Can we do better?

• Reverse join order: $R_2 \bowtie R_1$

$$\text{Total} = \frac{500}{100} \times (100 + 1,000) =$$

$$5 \times 1,100 = 5,500 \text{ IOs}$$

CS 525



Notes 10 - Query Execution

86

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 $(r,s) \models C$ then output (r,s)

CS 525



Notes 10 - Query Execution

87

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

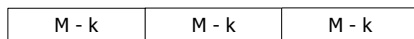
CS 525



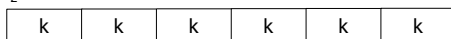
Notes 10 - Query Execution

88

R_1



R_2



CS 525



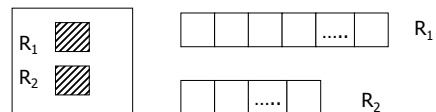
Notes 10 - Query Execution

89

Example 1(b) Merge Join

- Both R_1, R_2 ordered by C ; relations contiguous

Memory



CS 525



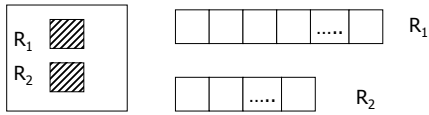
Notes 10 - Query Execution

90

Example 1(b) Merge Join

- Both R_1, R_2 ordered by C; relations contiguous

Memory



Total cost: Read R_1 cost + read R_2 cost
 = 1000 + 500 = 1,500 IOs

CS 525



Notes 10 - Query Execution

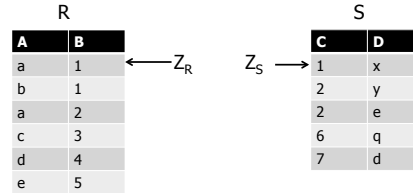
91

IIT College of Science and Letters
 ILLINOIS INSTITUTE OF TECHNOLOGY

Merge Join Example

$R \bowtie_{B=C} S$

Output: (a,1,1,X)



CS 525



Notes 10 - Query Execution

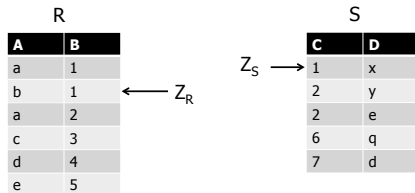
92

IIT College of Science and Letters
 ILLINOIS INSTITUTE OF TECHNOLOGY

Merge Join Example

$R \bowtie_{B=C} S$

Output: (b,1,1,X)



CS 525



Notes 10 - Query Execution

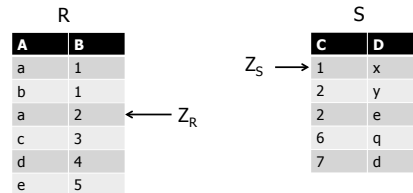
93

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



CS 525



Notes 10 - Query Execution

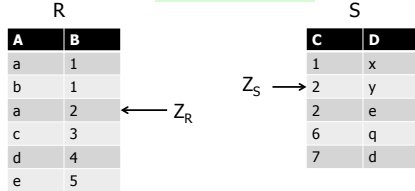
94

IIT College of Science and Letters
 ILLINOIS INSTITUTE OF TECHNOLOGY

Merge Join Example

$R \bowtie_{B=C} S$

Output: (a,2,2,y)



CS 525



Notes 10 - Query Execution

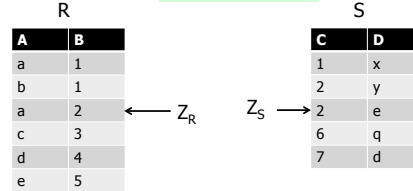
95

IIT College of Science and Letters
 ILLINOIS INSTITUTE OF TECHNOLOGY

Merge Join Example

$R \bowtie_{B=C} S$

Output: (a,2,2,e)



CS 525



Notes 10 - Query Execution

96

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		S	
A	B	C	D
a	1	1	x
b	1	2	y
a	2	2	e
c	3	6	q
d	4	7	d
e	5		

$Z_S \rightarrow$ (from S row 2 to S row 3)
 $Z_R \leftarrow$ (to R row 3)

CS 525  Notes 10 - Query Execution 97 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		S	
A	B	C	D
a	1	1	x
b	1	2	y
a	2	2	e
c	3	6	q
d	4	7	d
e	5		

$Z_S \rightarrow$ (from S row 2 to S row 3)
 $Z_R \leftarrow$ (to R row 3)

CS 525  Notes 10 - Query Execution 98 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		S	
A	B	C	D
a	1	1	x
b	1	2	y
a	2	2	e
c	3	6	q
d	4	7	d
e	5		

$Z_S \rightarrow$ (from S row 2 to S row 3)
 $Z_R \leftarrow$ (to R row 3)

CS 525  Notes 10 - Query Execution 99 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


Merge Join Example

$R \bowtie_{B=C} S$

R.B < S.C: **DONE**

R		S	
A	B	C	D
a	1	1	x
b	1	2	y
a	2	2	e
c	3	6	q
d	4	7	d
e	5		


$Z_S \rightarrow$ (from S row 2 to S row 3)
 $Z_R \leftarrow$ (to R row 3)

CS 525  Notes 10 - Query Execution 100 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Example 1(c) Merge Join

- R_1, R_2 not ordered, but contiguous

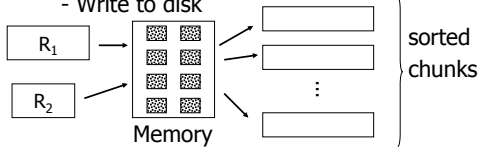
--> Need to sort R_1, R_2 first

CS 525  Notes 10 - Query Execution 101 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


One way to sort: Merge Sort

(i) For each 100 blk chunk of R:

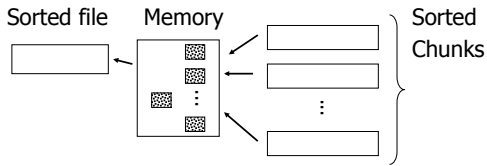
- Read chunk
- Sort in memory
- Write to disk



sorted chunks

CS 525  Notes 10 - Query Execution 102 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

(ii) Read all chunks + merge + write out



CS 525



Notes 10 - Query Execution

103

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Cost: Sort

Each tuple is read, written,
read, written

SO...

Sort cost R_1 : $4 \times 1,000 = 4,000$

Sort cost R_2 : $4 \times 500 = 2,000$

CS 525



Notes 10 - Query Execution

104

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
= $6,000 + 1,500 = 7,500$ IOs

CS 525



Notes 10 - Query Execution

105

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
= $6,000 + 1,500 = 7,500$ IOs

But: Iteration cost = 5,500
so merge joint does not pay off!

CS 525



Notes 10 - Query Execution

106

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

But say $R_1 = 10,000$ blocks contiguous
 $R_2 = 5,000$ blocks not ordered

Iterate: $\frac{5000}{100} \times (100 + 10,000) = 50 \times 10,100$
= 505,000 IOs

Merge join: $5(10,000 + 5,000) = 75,000$ IOs

Merge Join (with sort) WINS!

CS 525



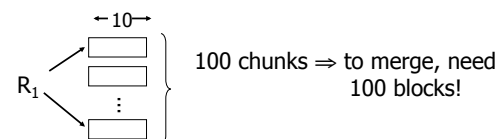
Notes 10 - Query Execution

107

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How much memory do we need for merge sort?

E.g: Say I have 10 memory blocks



CS 525



Notes 10 - Query Execution

108

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

CS 525



Notes 10 - Query Execution

109



In general:

Say k blocks in memory
 x blocks for relation sort
chunks = (x/k) size of chunk = k
chunks < buffers available for merge

CS 525



Notes 10 - Query Execution

110



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... $(x/k) \leq k$
or $k^2 \geq x$ or $k \geq \sqrt{x}$

CS 525



Notes 10 - Query Execution

111



In our example

R_1 is 1000 blocks, $k \geq 31.62$
 R_2 is 500 blocks, $k \geq 22.36$

Need at least 32 buffers

Again: in practice we would not want to use only one buffer per run!

CS 525



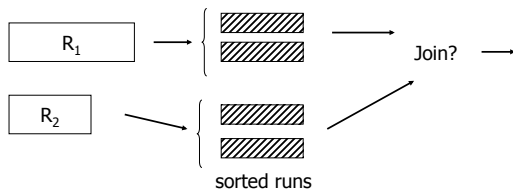
Notes 10 - Query Execution

112



Can we improve on merge join?

Hint: do we really need the fully sorted files?



CS 525



Notes 10 - Query Execution

113



Cost of improved merge join:

$C = \text{Read } R_1 + \text{write } R_1 \text{ into runs}$
+ read R_2 + write R_2 into runs
+ join
= 2,000 + 1,000 + 1,500 = 4,500

--> Memory requirement?

CS 525



Notes 10 - Query Execution

114



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

CS 525



Notes 10 - Query Execution

115

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Cost: Reads: 500 IOs

for each R_2 tuple:

- probe index - free
- if match, read R_1 tuple: 1 IO

CS 525



Notes 10 - Query Execution

116

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 = $10,000/5,000 = 2$

CS 525



Notes 10 - Query Execution

117

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What is expected # of matching tuples?

(c) Say $DOM(R_1, C) = 1,000,000$

$$T(R_1) = 10,000$$

with alternate assumption

$$\text{Expect} = \frac{10,000}{1,000,000} = \frac{1}{100}$$

CS 525



Notes 10 - Query Execution

118

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 = $500 + 5000(1/100)1 = 550$

CS 525



Notes 10 - Query Execution

119

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

$$E = (0)\frac{99}{200} + (1)\frac{101}{200} \approx 0.5$$

CS 525



Notes 10 - Query Execution

120

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Total cost (including probes)

$$\begin{aligned}
 &= 500+5000 \text{ [Probe + get records]} \\
 &= 500+5000 [0.5+2] \quad \text{uniform assumption} \\
 &= 500+12,500 = 13,000 \quad \text{(case b)}
 \end{aligned}$$

CS 525



Notes 10 - Query Execution

121



Total cost (including probes)

$$\begin{aligned}
 &= 500+5000 \text{ [Probe + get records]} \\
 &= 500+5000 [0.5+2] \quad \text{uniform assumption} \\
 &= 500+12,500 = 13,000 \quad \text{(case b)}
 \end{aligned}$$

For case (c):

$$\begin{aligned}
 &= 500+5000[0.5 \times 1 + (1/100) \times 1] \\
 &= 500+2500+50 = 3050 \text{ IOs}
 \end{aligned}$$

CS 525



Notes 10 - Query Execution

122



So far

{	Nested Loop	5500
	Merge join	1500
	Sort+Merge Join	7500 → 4500
	R ₁ .C Index	5500 → 3050 → 550
	R ₂ .C Index	_____

CS 525



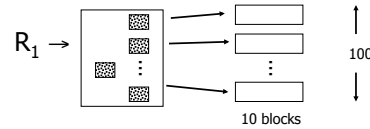
Notes 10 - Query Execution

123



Example 1(e) Partition Hash Join

- R₁, R₂ contiguous (un-ordered)
- Use 100 buckets
- Read R₁, hash, + write buckets



CS 525

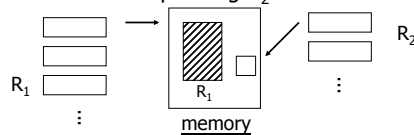


Notes 10 - Query Execution

124



- > Same for R₂
- > Read one R₁ bucket; build memory hash table -using different hash function h'
- > Read corresponding R₂ bucket + hash probe



Then repeat for all buckets

CS 525



Notes 10 - Query Execution

125



Cost:

“Bucketize:” Read R₁ + write
Read R₂ + write
Join: Read R₁, R₂

$$\text{Total cost} = 3 \times [1000+500] = 4500$$

CS 525



Notes 10 - Query Execution

126



Cost:

“Bucketize:” Read R_1 + write
Read R_2 + write
Join: Read R_1, R_2

Total cost = $3 \times [1000+500] = 4500$

Note: this is an approximation since buckets will vary in size and we have to round up to blocks

CS 525

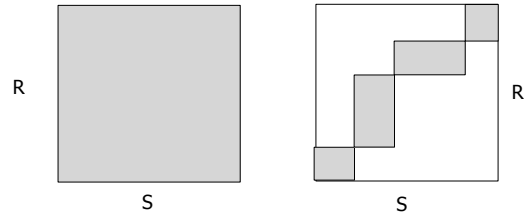


Notes 10 - Query Execution

127

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Why is Hash Join good?



CS 525



Notes 10 - Query Execution

128

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

CS 525



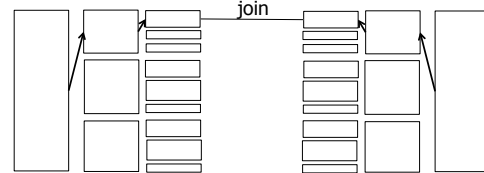
Notes 10 - Query Execution

129

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



CS 525



Notes 10 - Query Execution

130

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

CS 525



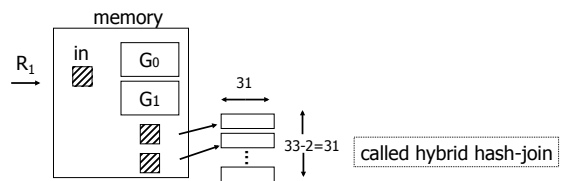
Notes 10 - Query Execution

131

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



CS 525



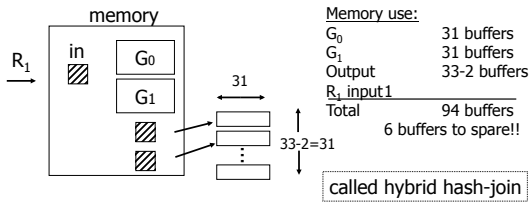
Notes 10 - Query Execution

132

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

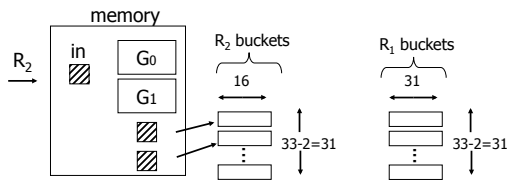
Trick: keep some buckets in memory

E.g., $k' = 33$ R_1 buckets = 31 blocks
keep 2 in memory



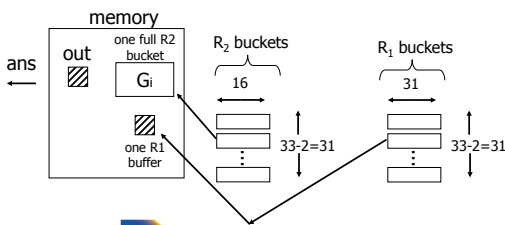
Next: Bucketize R_2

- 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

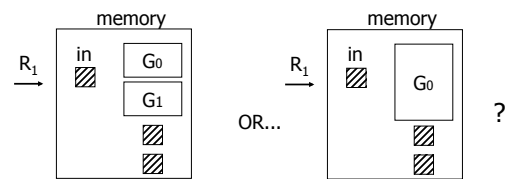


Cost

- Bucketize $R_1 = 1000 + 31 \times 31 = 1961$
- To bucketize R_2 , only write 31 buckets:
so, cost = $500 + 31 \times 16 = 996$
- To compare join (2 buckets already done)
read $31 \times 31 + 31 \times 16 = 1457$

Total cost = $1961 + 996 + 1457 = 4414$

• How many buckets in memory?



• See textbook for answer...

Another hash join trick:

- Only write into buckets $\langle val, ptr \rangle$ pairs
- When we get a match in join phase, must fetch tuples

- To illustrate cost computation, assume:
 - 100 <val,ptr> pairs/block
 - expected number of result tuples is 100

CS 525



Notes 10 - Query Execution

139

- 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 $\sim 100 R_2$ tuples

CS 525



Notes 10 - Query Execution

140

- 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 $\sim 100 R_2$ tuples

<u>Total cost</u> =	Read R_2 :	500
	Read R_1 :	1000
	Get tuples:	<u>100</u>
		1600

CS 525



Notes 10 - Query Execution

141

So far:

Iterate	5500
Merge join	1500
Sort+merge join	7500
$R_1.C$ index	5500 \rightarrow 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

CS 525



Notes 10 - Query Execution

142

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

CS 525



Notes 10 - Query Execution

143

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

CS 525



Notes 10 - Query Execution

144

- 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 could be useful (depends on expected result size)

CS 525



Notes 10 - Query Execution

145

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)	$B(R_1) * B(R_2) / M$	3	0
Index Nested Loop	$B(R_1) + N_1 * P_{12}$	$B(\text{Index}) + 2$	0
Merge (sorted)	$B(R_1) + B(R_2)$	Max tuples =	0
Merge (unsorted)	$B(R_1) + B(R_2) + (\text{sort} - 1 \text{ pass})$	sort	$B(R_1) + B(R_2)$
Hash	$(2\#P + 1) (B(R_1) + B(R_2))$	$\text{root}(\max(B(R_1), B(R_2)), \#P + 1)$	$\sim B(R_1) + B(R_2)$

CS 525



Notes 10 - Query Execution

146

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

CS 525



Notes 10 - Query Execution

147

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

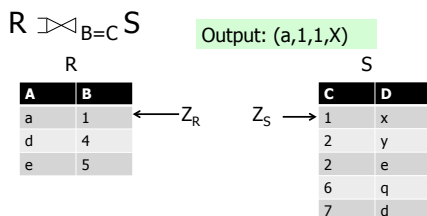
CS 525



Notes 10 - Query Execution

148

Merge Left Outer Join



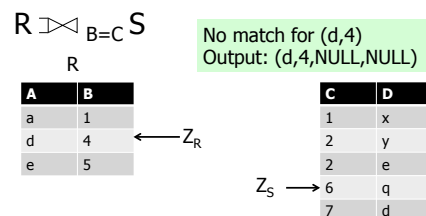
CS 525



Notes 10 - Query Execution

149

Merge Left Outer Join



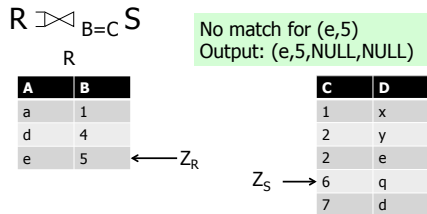
CS 525



Notes 10 - Query Execution

150

Merge Left Outer Join



CS 525



Notes 10 - Query Execution

151

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

CS 525



Notes 10 - Query Execution

152

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Aggregation

- Have to compute aggregation functions
 - for each group of tuples from input
- Groups
 - Determined by equality of group-by attributes

CS 525



Notes 10 - Query Execution

153

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

sum(a)	b
6	1
6	2

CS 525



Notes 10 - Query Execution

154

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

CS 525



Notes 10 - Query Execution

155

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Implementation SUM(A)

- `init()`
 - `sum := 0`
- `update(tuple)`
 - `sum += tuple.A`
- `close()`
 - **return sum**

CS 525



Notes 10 - Query Execution

156

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

CS 525



Notes 10 - Query Execution

157

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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()

CS 525



Notes 10 - Query Execution

158

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Aggregation Example

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

sort

a	b	a	b
3	1	3	1
4	2	3	1
3	1	4	2
1	2	1	2
1	2	1	2

init()

0

CS 525



Notes 10 - Query Execution

159

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

CS 525



Notes 10 - Query Execution

160

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

CS 525



Notes 10 - Query Execution

161

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

Group by changed!
close(), init(), update(4,2)

① 6

② 0

③ 4

output

CS 525



Notes 10 - Query Execution

162

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 output calling `close()`

CS 525



Notes 10 - Query Execution

163

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



CS 525



Notes 10 - Query Execution

164

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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



CS 525



Notes 10 - Query Execution

165

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



CS 525



Notes 10 - Query Execution

166

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



CS 525



Notes 10 - Query Execution

167

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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



CS 525



Notes 10 - Query Execution

168

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

CS 525



Notes 10 - Query Execution

169

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

CS 525



Notes 10 - Query Execution

170

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

CS 525



Notes 10 - Query Execution

171

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

CS 525



Notes 10 - Query Execution

172

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

CS 525



Notes 10 - Query Execution

173

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Union

- Bag union
 - Append the two inputs
 - E.g., using three buffers
- Set union
 - Apply duplicate removal to result

CS 525



Notes 10 - Query Execution

174

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Intersection

- Set version
 - Equivalent to join + project + duplicate removal
 - 3-state aggregate function (found left, found right, found both)
- Bag version
 - Join + project + $\min(i,j)$
 - Aggregate $\min(\text{count}(i), \text{count}(j))$

CS 525



Notes 10 - Query Execution

175

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
 - $\text{count}(i) - \text{count}(j)$ (**bag**)
 - $\text{true}(i) \text{ AND } \text{false}(j)$ (**set**)

CS 525



Notes 10 - Query Execution

176

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Summary

- Operator implementations
 - Joins!
 - Other operators
- Cost estimations
 - I/O
 - memory
- Query processing architectures

CS 525



Notes 10 - Query Execution

177

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Next

- Query Optimization Physical
- -> How to **efficiently** choose an **efficient** plan

CS 525



Notes 10 - Query Execution

178

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY


CS 525: Advanced Database Organization


11: Query Optimization

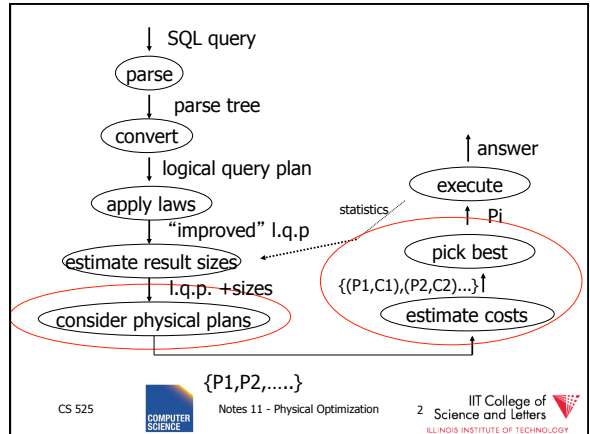
Physical

Boris Glavic

Slides: adapted from a [course](#) taught by [Hector Garcia-Molina](#), Stanford InfoLab




CS 525

Notes 11 - Physical Optimization
1 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY




Physical Optimization

- Apply after applying heuristics in logical optimization
- 1) Enumerate potential execution plans
 - All?
 - Subset
- 2) Cost plans
 - What cost function?

CS 525

Notes 11 - Physical Optimization
3 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525

Notes 11 - Physical Optimization
4 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'
    
```

```

    graph TD
      J1((⋈  
name=emp)) --- Employee
      J1 --- EmpDep
      J2((⋈  
dep=dep)) --- J1
      J2 --- Department
      J3((⋈  
dep=CS)) --- J2
      J3 --- Department
  
```

CS 525

Notes 11 - Physical Optimization
5 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'
    
```

```

    graph TD
      J1((⋈  
name=emp)) --- Employee
      J1 --- EmpDep
      J2((⋈  
dep=dep)) --- J1
      J2 --- Department
      J3((⋈  
dep=CS)) --- J2
      J3 --- Department
      
      subgraph Annotations
        direction TB
        NL((NL)) --- J2
        MJ((MJ)) --- J1
        IS((IS)) --- J3
        SS_E[SSEmployee] --- Employee
        SS_ED[SSEmpDep] --- EmpDep
      end
  
```

CS 525

Notes 11 - Physical Optimization
6 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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)

CS 525



Notes 11 - Physical Optimization

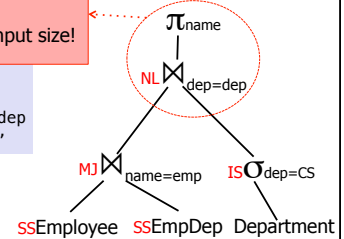
7

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example Query – Possible Plan

```
SELECT e.name
FROM Employee
EmpDep ed
Department d
WHERE e.name = ed.emp
AND ed.dep = d.dep
AND d.dep = 'CS'
```

Cost?
Need input size!



CS 525



Notes 11 - Physical Optimization

8

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 11 - Physical Optimization

9

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525

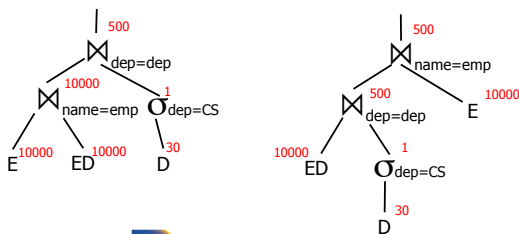


Notes 11 - Physical Optimization

10

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example Join Ordering Result Sizes



CS 525



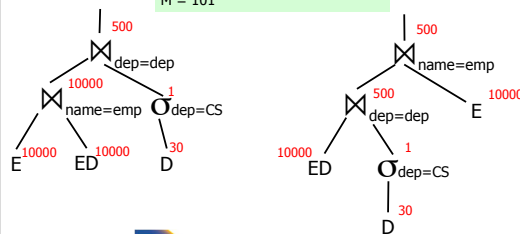
Notes 11 - Physical Optimization

11

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example Join Ordering Cost (only NL)

$S(E) = S(ED) = S(D) = 1/10$ block
 $M = 101$



CS 525



Notes 11 - Physical Optimization

12

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

$S(E) = S(ED) = S(D) = 1/10$ block
 $M = 101$
 I/O costs only

$1100 \times 10 + 3 + 1000 = 12003$ I/Os $1000 + 1000 + 3 = 2003$ I/Os

CS 525 IIT College of Science and Letters

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

CS 525 IIT College of Science and Letters

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

CS 525 IIT College of Science and Letters

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. $\sigma_{a=b} (R \times S) \equiv R \bowtie_{a=b} S$

CS 525 IIT College of Science and Letters

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 if a is from R and S from b (vice-versa)

CS 525 IIT College of Science and Letters

Why Cross-Products are bad

- We discussed efficient join algorithms
 - Merge-join $O(n)$ resp. $O(n \log(n))$
 - Vs. Nested-loop $O(n^2)$
- $R \times S$
 - Result size is $O(n^2)$
 - Cannot be better than $O(n^2)$
 - Surprise, surprise: merge-join doesn't work

CS 525 IIT College of Science and Letters

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**

CS 525



Notes 11 - Physical Optimization

19

Join Graph

- Assumptions
 - Only equi-joins ($a = b$)
 - a and b are either constants or attributes
 - Only conjunctive join conditions (AND)

CS 525



Notes 11 - Physical Optimization

20

Join Graph

- Nodes: Relations R_1, \dots, 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

CS 525

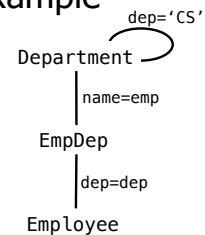


Notes 11 - Physical Optimization

21

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'
```



CS 525

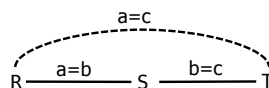


Notes 11 - Physical Optimization

22

Notes on Join Graph

- Join Graph tells us in which ways we can join without using cross products
- However, ...
 - Only if transitivity is considered



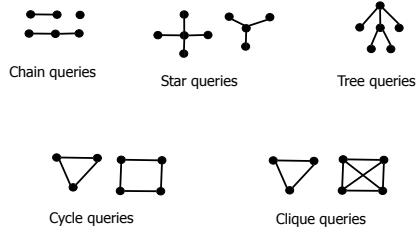
CS 525



Notes 11 - Physical Optimization

23

Join Graph Shapes



CS 525



Notes 11 - Physical Optimization

24

Join Graph Shapes

Chain queries



```
SELECT *  
FROM R, S, T  
WHERE R.a = S.b  
AND S.c = T.d
```

CS 525



Notes 11 - Physical Optimization

25

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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
```

CS 525



Notes 11 - Physical Optimization

26

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Join Graph Shapes

```
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
```



Tree queries

CS 525

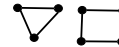


Notes 11 - Physical Optimization

27

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Join Graph Shapes



Cycle queries

```
SELECT *  
FROM R, S, T  
WHERE R.a = S.a  
AND S.b = T.b  
AND T.c = R.c
```

CS 525



Notes 11 - Physical Optimization

28

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

CS 525



Notes 11 - Physical Optimization

29

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



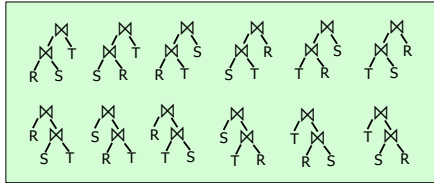
Notes 11 - Physical Optimization

30

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How many join orders?

- Example 3 relations R,S,T
– 12 orders



CS 525



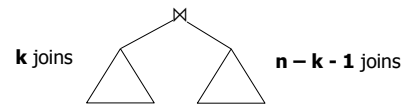
Notes 11 - Physical Optimization

31

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 ($0 \leq k \leq n-1$)



CS 525



Notes 11 - Physical Optimization

32

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How many join orders?

- This are the **Catalan numbers**

$$C_n = \sum_{k=0}^{n-1} C_k \times C_{n-k-1} = (2n)! / (n+1)!n!$$

CS 525



Notes 11 - Physical Optimization

33

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

$$(2n)! / (n+1)!n! * (n+1)! = (2n)!/n!$$

CS 525



Notes 11 - Physical Optimization

34

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

CS 525



Notes 11 - Physical Optimization

35

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How many join orders?

- If for each join we consider k equal algorithms then for n relations we have
 - Multiply with a factor k^{n-1}
- Example consider
 - Nested loop
 - Merge
 - Hash

CS 525



Notes 11 - Physical Optimization

36

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

CS 525



Notes 11 - Physical Optimization

37

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

CS 525



Notes 11 - Physical Optimization

38

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

CS 525



Notes 11 - Physical Optimization

39

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 11 - Physical Optimization

40

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

CS 525



Notes 11 - Physical Optimization

41

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)
}
```

CS 525

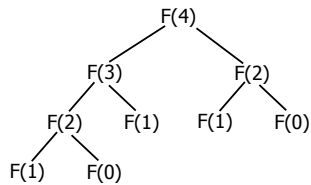


Notes 11 - Physical Optimization

42

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example Fibonacci Numbers



CS 525



Notes 11 - Physical Optimization

43

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Complexity

- Number of calls
 - $C(n) = C(n-1) + C(n-2) + 1 = \text{Fib}(n+2)$
 - $O(2^n)$

CS 525



Notes 11 - Physical Optimization

44

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];
}
```

CS 525

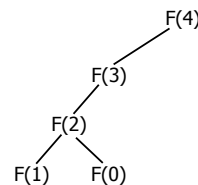


Notes 11 - Physical Optimization

45

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example Fibonacci Numbers



CS 525



Notes 11 - Physical Optimization

46

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What do we gain?

- $O(n)$ instead of $O(2^n)$

CS 525



Notes 11 - Physical Optimization

47

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Dynamic Programming for Join Enumeration

- Find cheapest plan for n -relation join in n passes
- For each i in $1 \dots n$
 - Construct solutions of size i from best solutions of size $< i$

CS 525



Notes 11 - Physical Optimization

48

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

DP Join Enumeration

```

optPlan ← Map({R}, {plan})
find_join_dp(q(R1, ..., Rn))
{
  for i=1 to n
    optPlan[{Ri}] ← access_paths(Ri)
  for i=2 to n
    foreach S ⊆ {R1, ..., Rn} 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[{R1, ..., Rn}]
}

```

CS 525



Notes 11 - Physical Optimization

49

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 for between the input plans
- **prune_plans({plan})**
 - Only keep cheapest plan from input set

CS 525



Notes 11 - Physical Optimization

50

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

DP-JE Complexity

- Time: $O(3^n)$
- Space: $O(2^n)$
- Still too much for large number of joins (10-20)

CS 525



Notes 11 - Physical Optimization

51

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Types of join trees

Left-deep



zig-zag



bushy



Right-deep



CS 525



Notes 11 - Physical Optimization

52

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Number of Join-Trees

- Number of join trees for **n** relations
- Left-deep: **n!**
- Right-deep: **n!**
- Zig-zag: **$2^n \cdot 2n!$**

CS 525



Notes 11 - Physical Optimization

53

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How many join orders?

#relations	#bushy join trees	#left-deep join trees
2		2
3		6
4		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

CS 525



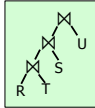
Notes 11 - Physical Optimization

54

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



CS 525



Notes 11 - Physical Optimization

55

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 the reduces cost of joining above
- Keep track of **interesting orders**

CS 525



Notes 11 - Physical Optimization

56

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

CS 525



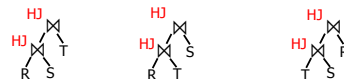
Notes 11 - Physical Optimization

57

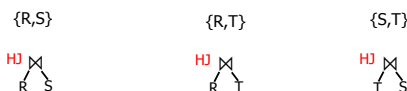
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example Interesting Orders

Left-deep best plans: 3-way {R,S,T}



Left-deep best plans: 2-way



CS 525



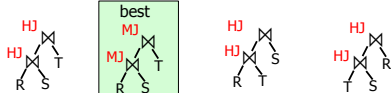
Notes 11 - Physical Optimization

58

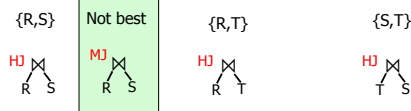
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example Interesting Orders

Left-deep best plans: 3-way {R,S,T}



Left-deep best plans: 2-way



CS 525



Notes 11 - Physical Optimization

59

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 11 - Physical Optimization

60

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 = argminj,k∈{1,...,n} (cost(Pj ⋈ Pk))
    plans ← plans \ {Pj, Pk} ∪ {Pj ⋈ Pk}
  return plans // single plan left
}
```

CS 525



Notes 11 - Physical Optimization

61

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

CS 525



Notes 11 - Physical Optimization

62

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Randomized Join-Algorithms

- Iterative improvement
- Simulated annealing
- Tabu-search
- Genetic algorithms

CS 525



Notes 11 - Physical Optimization

63

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Transformative Approach

- Start from (random) complete solutions
- Apply transformations to generate new solutions
 - Direct application of equivalences
 - Commutativity
 - Associativity
 - Combined equivalences
 - E.g., $(R \bowtie S) \bowtie T \equiv T \bowtie (S \bowtie R)$

CS 525



Notes 11 - Physical Optimization

64

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

CS 525



Notes 11 - Physical Optimization

65

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Iterative Improvement

```
improve(q(R1, ..., Rn))
{
  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(improved) < cost(best))
          best ← improved
    return best
}
```

CS 525



Notes 11 - Physical Optimization

66

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

CS 525



Notes 11 - Physical Optimization

67

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Simulated Annealing

```
SA(q(R1, ..., Rn))
{
  best ← random_plan(q)
  curplan ← best
  t ← tinit // "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(improved) < cost(best))
      best ← improved
  reduce(t)
  return best
}
```



IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 11 - Physical Optimization

69

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"



CS 525



Notes 11 - Physical Optimization

70

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Mutation

- Switch random two random position
- Is applied with a certain fixed probability
- E.g., "1342" -> "4312"

CS 525



Notes 11 - Physical Optimization

71

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 $J' = "5643278"$

CS 525



Notes 11 - Physical Optimization

72

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

CS 525



Notes 11 - Physical Optimization

73

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

CS 525



Notes 11 - Physical Optimization

74

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 change to keep solution based on cost

CS 525



Notes 11 - Physical Optimization

75

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.

CS 525



Notes 11 - Physical Optimization

76

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
 - ...

CS 525



Notes 11 - Physical Optimization

77

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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?

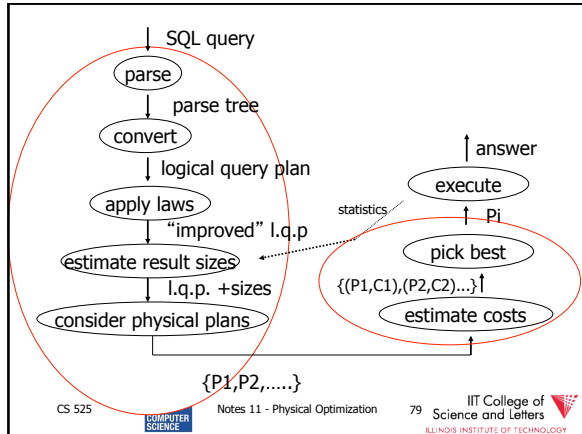
CS 525



Notes 11 - Physical Optimization

78

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

CS 525 IIT College of Science and Letters
Notes 11 - Physical Optimization 80 ILLINOIS INSTITUTE OF TECHNOLOGY

Query Graph Model

- 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

CS 525 IIT College of Science and Letters
Notes 11 - Physical Optimization 81 ILLINOIS INSTITUTE OF TECHNOLOGY

QGM example

```

SELECT name, city
FROM
  (SELECT *
   FROM person) AS p,
  (SELECT *
   FROM address) AS a
WHERE p.addrId = a.id
  
```

CS 525 IIT College of Science and Letters
Notes 11 - Physical Optimization 82 ILLINOIS INSTITUTE OF TECHNOLOGY

Postgres Example

```

QUERY
commandType 1
querySource 0
scanGetTag true
subquery 0
resultRelation 0
joinClassid 0
joinAggr false
hashCollines false
rttable 1
ORTE
alias
joinname p
joinname <->
}
nest
alias
joinname p
joinname (name"addrId")
}
subbind 1
subquery
QUERY
commandType 1
querySource 0
scanGetTag true
  
```

CS 525 IIT College of Science and Letters
Notes 11 - Physical Optimization 83 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

CS 525 IIT College of Science and Letters
Notes 11 - Physical Optimization 84 ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 11 - Physical Optimization

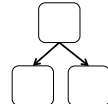
85

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
```



CS 525



Notes 11 - Physical Optimization

86

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 11 - Physical Optimization

87

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

CS 525



Notes 11 - Physical Optimization

88

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

CS 525



Notes 11 - Physical Optimization

89

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 11 - Physical Optimization

90

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

PREPARE statement

- In SQL
 - **PREPARE** name (parameters) AS query
 - **EXECUTE** name (parameters)

CS 525



Notes 11 - Physical Optimization

91

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Nested Subqueries

```
SELECT name
FROM person p
WHERE EXISTS (SELECT newspaper
              FROM hasRead h
              WHERE h.name = p.name
              AND h.newspaper = 'Tribune')
```

CS 525



Notes 11 - Physical Optimization

92

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

CS 525



Notes 11 - Physical Optimization

93

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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')
```

CS 525



Notes 11 - Physical Optimization

94

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Nested Iteration - Uncorrelated

```
q ← outer query
q' ← inner query
result ← execute(q)
result' ← execute (qt)
foreach tuple t in result
  evaluate_nested_condition (t,result')
```

CS 525



Notes 11 - Physical Optimization

95

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		hasRead	
name	gender	name	newspaper
Alice	female	Alice	Tribune
Bob	male	Alice	Courier
Joe	male	Joe	Courier

CS 525



Notes 11 - Physical Optimization

96

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		hasRead	
name	gender	name	newspaper
Alice	female	Alice	Tribune
Bob	male	Alice	Courier
Joe	male	Joe	Courier

CS 525



Notes 11 - Physical Optimization

97

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 = 'Alice'
      AND h.newspaper
      = 'Tribune'
    
```

person		hasRead	
name	gender	name	newspaper
Alice	female	Alice	Tribune
Bob	male	Alice	Courier
Joe	male	Joe	Courier

CS 525



Notes 11 - Physical Optimization

98

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		hasRead		result'
name	gender	name	newspaper	newspaper
Alice	female	Alice	Tribune	Tribune
Bob	male	Alice	Courier	
Joe	male	Joe	Courier	

CS 525



Notes 11 - Physical Optimization

99

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')
    
```

EXISTS evaluates to true!

Output(Alice)

person		hasRead		result'
name	gender	name	newspaper	newspaper
Alice	female	Alice	Tribune	Tribune
Bob	male	Alice	Courier	
Joe	male	Joe	Courier	

CS 525



Notes 11 - Physical Optimization

100

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')
    
```

Empty result set -> EXISTS evaluates to false

person		hasRead		result'
name	gender	name	newspaper	newspaper
Alice	female	Alice	Tribune	
Bob	male	Alice	Courier	
Joe	male	Joe	Courier	

CS 525



Notes 11 - Physical Optimization

101

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')
    
```

Empty result set -> EXISTS evaluates to false

person		hasRead		result'
name	gender	name	newspaper	newspaper
Alice	female	Alice	Tribune	
Bob	male	Alice	Courier	
Joe	male	Joe	Courier	

CS 525



Notes 11 - Physical Optimization

102

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 11 - Physical Optimization

103

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

CS 525



Notes 11 - Physical Optimization

104

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Equivalences

- Classify types of nesting
- Equivalence rules will have preconditions
- Can be applied heuristically before plan enumeration or using a transformative approach

CS 525



Notes 11 - Physical Optimization

105

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')
```

CS 525



Notes 11 - Physical Optimization

106

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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)
```

CS 525



Notes 11 - Physical Optimization

107

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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')
```

CS 525



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)
```

CS 525



IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Unnesting A-type

- Move nested query to FROM clause
- Turn nested condition (op ANY, IN) into op with result attribute of nested query

CS 525



Notes 11 - Physical Optimization

110

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

CS 525



Notes 11 - Physical Optimization

111

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
```

CS 525



Notes 11 - Physical Optimization

112

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
```

CS 525



Notes 11 - Physical Optimization

113

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
```

CS 525



Notes 11 - Physical Optimization

114

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
  
```

CS 525  Notes 11 - Physical Optimization 115 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

CS 525  Notes 11 - Physical Optimization 116 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
  
```

CS 525  Notes 11 - Physical Optimization 117 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 i.cust = sub.cust
  
```

CS 525  Notes 11 - Physical Optimization 118 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
  
```

CS 525  Notes 11 - Physical Optimization 119 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


Unnesting Benefits Example

- $N(\text{orders}) = 1.000.000$
- $V(\text{cust,orders}) = 10.000$
- $S(\text{orders}) = 1/10$ 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
  
```

CS 525  Notes 11 - Physical Optimization 120 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

- $N(\text{orders}) = 1.000.000$
- $V(\text{cust,orders}) = 10.000$
- $S(\text{orders}) = 1/10 \text{ block}$
- $M = 10.000$

```
SELECT name
FROM orders o
WHERE o.amount = (SELECT max(amount)
                  FROM orders i
                  WHERE i.cust = o.cust)
```

- Inner query:
 - One scan $B(\text{orders}) = 100.000$ I/Os
- Outer query:
 - One scan $B(\text{orders}) = 100.000$ I/Os
 - 1.000.000 tuples
- Total cost: $1.000.001 \times 100.000 \approx 10^{11}$ I/Os

CS 525



Notes 11 - Physical Optimization

121

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- $N(\text{orders}) = 1.000.000$
- $V(\text{cust,orders}) = 10.000$
- $S(\text{orders}) = 1/10 \text{ block}$
- $M = 10.000$

```
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 $B(\text{orders}) = 100.000$ I/Os
 - 1.000.000 result tuples
 - Sort (assume 1 pass) = $3 \times 100.000 = 300.000$ I/Os
 - 10.000 result tuples
- The join: use merge
 - $3 \times (1.000 + 100.000)$ I/Os = 303.000 I/Os
- Total cost: 603.000 I/Os

CS 525




Notes 11 - Physical Optimization

122

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY


CS 525: Advanced Database Organization

12: Transaction Management




Boris Glavic

Slides: adapted from a [course](#) taught by [Hector Garcia-Molina](#), Stanford InfoLab

CS 525  Notes 12 - Transaction Management 1 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


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**)

CS 525  Notes 12 - Transaction Management 2 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**)


CS 525  Notes 12 - Transaction Management 3 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

CS 525  Notes 12 - Transaction Management 4 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


Integrity or consistency constraints

- Predicates data must satisfy
- Examples:
 - x is key of relation R
 - $x \rightarrow y$ holds in R
 - $\text{Domain}(x) = \{\text{Red, Blue, Green}\}$
 - α is valid index for attribute x of R
 - no employee should make more than twice the average salary

CS 525  Notes 12 - Transaction Management 5 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Definition:


- Consistent state: satisfies all constraints
- Consistent DB: DB in consistent state

CS 525  Notes 12 - Transaction Management 6 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

CS 525  Notes 12 - Transaction Management 7 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Note: could be “emulated” by simple constraints, e.g.,

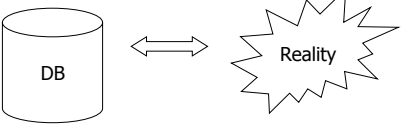
account


Acct #	...	balance	deleted?
--------	-----	---------	----------

CS 525  Notes 12 - Transaction Management 8 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




CS 525  Notes 12 - Transaction Management 9 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

☞ in any case, continue with constraints...

Observation: DB cannot be consistent always!

Example: $a_1 + a_2 + \dots + a_n = \text{TOT}$ (constraint)


Deposit \$100 in a_2 : $\begin{cases} a_2 \leftarrow a_2 + 100 \\ \text{TOT} \leftarrow \text{TOT} + 100 \end{cases}$

CS 525  Notes 12 - Transaction Management 10 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Example: $a_1 + a_2 + \dots + a_n = \text{TOT}$ (constraint)


Deposit \$100 in a_2 : $\begin{cases} a_2 \leftarrow a_2 + 100 \\ \text{TOT} \leftarrow \text{TOT} + 100 \end{cases}$

a ₂	:	:	:
	50	150	150
TOT	:	:	:
	1000	1000	1100

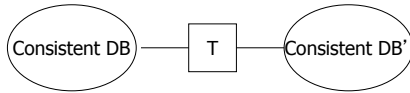
CS 525  Notes 12 - Transaction Management 11 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

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

Transaction: collection of actions that preserve consistency



CS 525



Notes 12 - Transaction Management

13

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Big assumption:

If T starts with consistent state +
T executes in isolation
⇒ T leaves consistent state

CS 525



Notes 12 - Transaction Management

14

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Correctness (informally)

- If we stop running transactions, DB left consistent
- Each transaction sees a consistent DB

CS 525



Notes 12 - Transaction Management

15

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Transactions - ACID

- **A**tomicity
 - Either all or no commands of transaction are executed
- **C**onsistency
 - After transaction DB is consistent
- **I**solation
 - Transactions are running isolated from each other
- **D**urability
 - Modification executed by transaction are never lost

CS 525



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 ⇒ systems analysts

CS 525



Notes 12 - Transaction Management

17

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

CS 525



Notes 12 - Transaction Management

18

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 12 - Transaction Management

19

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Operations:

- Input (x): block containing x → memory
- Output (x): block containing x → disk

CS 525



Notes 12 - Transaction Management

20

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Operations:

- Input (x): block containing x → memory
- Output (x): block containing x → disk
- Read (x,t): do input(x) if necessary
t ← value of x in block
- Write (x,t): do input(x) if necessary
value of x in block ← t

CS 525



Notes 12 - Transaction Management

21

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Key problem Unfinished transaction **(Atomicity)**

Example

Constraint: A=B

T1: A ← A × 2

B ← B × 2

CS 525

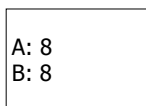


Notes 12 - Transaction Management

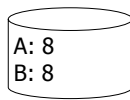
22

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);



memory



disk

CS 525

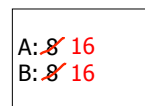


Notes 12 - Transaction Management

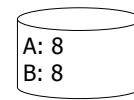
23

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);



memory



disk

CS 525



Notes 12 - Transaction Management

24

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

T1: Read (A,t); t ← tx2
 Write (A,t);
 Read (B,t); t ← tx2
 Write (B,t);
 Output (A);
 Output (B); failure!

A: ~~8~~ 16
 B: ~~8~~ 16

A: ~~8~~ 16
 B: 8

memory
disk

CS 525 Notes 12 - Transaction Management 25 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Transactions in SQL

- BEGIN TRANSACTION
 - Start new transaction
- COMMIT
 - Finish and make all modifications of transactions persistent
- ABORT/ROLLBACK
 - Finish and undo all changes of transaction

CS 525 Notes 12 - Transaction Management 26 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Example

time ↓

BEGIN TRANSACTION;
 UPDATE accounts
 SET bal = bal + 40
 WHERE acc = 10;

 UPDATE accounts
 SET bal = bal - 40
 WHERE acc = 9;
 COMMIT;

BEGIN TRANSACTION;
 UPDATE accounts
 SET bal = bal * 1.05;
 COMMIT;

CS 525 Notes 12 - Transaction Management 27 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Example

time ↓

BEGIN TRANSACTION;
 UPDATE accounts
 SET bal = bal + 40
 WHERE acc = 10;

 UPDATE accounts
 SET bal = bal - 40
 WHERE acc = 9;
 COMMIT;

BEGIN TRANSACTION;
 UPDATE accounts
 SET bal = bal * 1.05;
 COMMIT;

Bank customer transfers money from account 9 to account 10

CS 525 Notes 12 - Transaction Management 28 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Example

time ↓

BEGIN TRANSACTION;
 UPDATE accounts
 SET bal = bal + 40
 WHERE acc = 10;

 UPDATE accounts
 SET bal = bal - 40
 WHERE acc = 9;
 COMMIT;

BEGIN TRANSACTION;
 UPDATE accounts
 SET bal = bal * 1.05;
 COMMIT;

Bank adds interest to all accounts

CS 525 Notes 12 - Transaction Management 29 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Example

time ↓

BEGIN TRANSACTION;
 UPDATE accounts
 SET bal = bal
 WHERE acc = 10

 UPDATE accounts
 SET bal = bal - 40
 WHERE acc = 9;
 COMMIT;

SET bal = bal * 1.05;
 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)

CS 525 Notes 12 - Transaction Management 30 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

CS 525



Notes 12 - Transaction Management

31

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

$$T_1 = r_1(a_{10}), w_1(a_{10}), r_1(a_9), w_1(a_9), c_1$$

time

```
BEGIN TRANSACTION;
UPDATE accounts
  SET bal = bal + 40
  WHERE acc = 10;
```

```
UPDATE accounts
  SET bal = bal - 40
  WHERE acc = 9;
```

```
COMMIT;
```

CS 525



Notes 12 - Transaction Management

32

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

$$T_1 = r_1(a_{10}), w_1(a_{10}), r_1(a_9), w_1(a_9), c_1$$

$$T_2 = r_2(a_1), w_2(a_1), r_2(a_2), w_2(a_2), r_2(a_9), w_2(a_9), r_2(a_{10}), w_2(a_{10}), c_1$$

```
BEGIN TRANSACTION;
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 TRANSACTION;
UPDATE accounts
  SET bal = bal * 1.05;
COMMIT;
```

CS 525



Notes 12 - Transaction Management

33

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Schedules

- A **schedule S** for a set of transactions $T = \{T_1, \dots, T_n\}$ is a partial order over operations of T so that
 - **S** contains a prefix of the operations of each T_i
 - Operations of T_i appear in the same order in **S** as in T_i
 - For any two conflicting operations they are ordered

CS 525



Notes 12 - Transaction Management

34

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How to model execution order?

- Schedules model the order of the execution for operations of a set of transactions

CS 525



Notes 12 - Transaction Management

35

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!

CS 525



Notes 12 - Transaction Management

36

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

CS 525



Notes 12 - Transaction Management

37

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**

CS 525



Notes 12 - Transaction Management

38

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

$$S = r_2(a_1), r_1(a_{10}), c_1$$

CS 525



Notes 12 - Transaction Management

39

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

$$S_1 = \dots w_2(a_1) \dots w_1(a_{10})$$

$$S_2 = \dots w_1(a_1) \dots w_2(a_{10})$$

CS 525



Notes 12 - Transaction Management

40

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 12 - Transaction Management

41



IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

CS 525: Advanced Database Organization

13: Failure and Recovery

Boris Glavic


Slides: adapted from a [course](#) taught by [Hector Garcia-Molina](#), Stanford InfoLab

CS 525 Notes 13 - Failure and Recovery 1 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Now


- Crash recovery



CS 525 Notes 13 - Failure and Recovery 2 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Correctness (informally)


- If we stop running transactions, DB left consistent
- Each transaction sees a consistent DB



CS 525 Notes 13 - Failure and Recovery 3 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



CS 525 Notes 13 - Failure and Recovery 4 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Recovery

- First order of business: Failure Model




CS 525 Notes 13 - Failure and Recovery 5 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Events — Desired

 Undesired — Expected

 Unexpected



CS 525 Notes 13 - Failure and Recovery 6 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Our failure model

The diagram shows a central horizontal line representing a system bus. Above the bus is a box labeled 'CPU' with the text '---processor' to its right. Below the bus are two components: a box labeled 'M' with the text 'memory ---->' to its left, and a cylinder labeled 'D' with the text '---disk' to its right.

CS 525 Notes 13 - Failure and Recovery 7 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

CS 525 Notes 13 - Failure and Recovery 8 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!

CS 525 Notes 13 - Failure and Recovery 9 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Undesired Unexpected: Everything else!

Examples:

- Disk data is lost
- Memory lost without CPU halt
- CPU implodes wiping out universe....

CS 525 Notes 13 - Failure and Recovery 10 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525 Notes 13 - Failure and Recovery 11 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Second order of business:

Storage hierarchy

The diagram shows two boxes connected by a horizontal line. The left box is labeled 'Memory' and 'DB Buffer' below it, and contains a small square with an 'X' inside. The right box is labeled 'Disk' below it and contains a small square with an 'X' inside.

CS 525 Notes 13 - Failure and Recovery 12 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Operations:

- Input (x): block containing x → memory
- Output (x): block containing x → disk

CS 525



Notes 13 - Failure and Recovery

13

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Operations:

- Input (x): block containing x → memory
- Output (x): block containing x → disk
- Read (x,t): do input(x) if necessary
t ← value of x in block
- Write (x,t): do input(x) if necessary
value of x in block ← t

CS 525



Notes 13 - Failure and Recovery

14

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Key problem Unfinished transaction

Example

Constraint: A=B

T1: A ← A × 2

B ← B × 2

CS 525

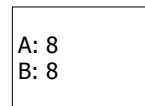


Notes 13 - Failure and Recovery

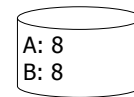
15

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

T1: Read (A,t); t ← tx2
Write (A,t);
Read (B,t); t ← tx2
Write (B,t);
Output (A);
Output (B);



memory



disk

CS 525

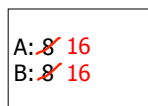


Notes 13 - Failure and Recovery

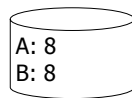
16

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

T1: Read (A,t); t ← tx2
Write (A,t);
Read (B,t); t ← tx2
Write (B,t);
Output (A);
Output (B);



memory



disk

CS 525

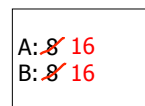


Notes 13 - Failure and Recovery

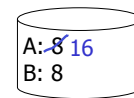
17

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

T1: Read (A,t); t ← tx2
Write (A,t);
Read (B,t); t ← tx2
Write (B,t);
~~Output (A);~~
Output (B); failure!



memory



disk

CS 525




Notes 13 - Failure and Recovery

18


IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- Need atomicity:
 - execute all actions of a transaction or none at all

CS 525  Notes 13 - Failure and Recovery 19 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

CS 525  Notes 13 - Failure and Recovery 20 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


CS 525  Notes 13 - Failure and Recovery 21 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

T₁: Read (A,t); t ← tx2
 Write (A,t);
 Read (B,t); t ← tx2
 Write (B,t);
Output (A);
 Output (B); failure!

T₁ is unfinished
 -> need to undo the write to A to recover to consistent state


A: ~~8~~ 16
B: 8

memory
disk

CS 525  Notes 13 - Failure and Recovery 22 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


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

CS 525  Notes 13 - Failure and Recovery 23 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Buffer Replacement Revisited

- Now we are interested in knowing how buffer replacement influences recovery!

CS 525  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

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

Effects of Buffer Replacement

	force	No force
No steal	<ul style="list-style-type: none"> • No Undo • No Redo 	<ul style="list-style-type: none"> • No Undo • Redo
steal	<ul style="list-style-type: none"> • Undo • No Redo 	<ul style="list-style-type: none"> • Redo • Undo

Schedules and Recovery

- Are there certain schedules that are easy/hard/impossible to recover from?

Recoverable Schedules

- We should never have to rollback an already committed transaction (D in ACID)
- **Recoverable** 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

$$T_1 = w_1(X), c_1$$

$$T_2 = r_2(X), w_2(X), c_2$$

Recoverable 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

$$S = \dots w_1(X) \dots r_2(X) \dots a_1$$

CS 525



Notes 13 - Failure and Recovery

31

Cascadeless Schedules

- Cascadeless** schedules guarantee that there are no cascading aborts
 - Transactions only read values written by already committed transactions

CS 525



Notes 13 - Failure and Recovery

32

$$T_1 = w_1(X), c_1$$

$$T_2 = r_2(X), w_2(X), c_2$$

Cascadeless Schedule

$$S_1 = w_1(X), c_1, r_2(X), w_2(X), c_2$$

Recoverable Schedule

$$S_2 = w_1(X), r_2(X), w_2(X), c_1, c_2$$

Nonrecoverable Schedule

$$S_3 = w_1(X), r_2(X), w_2(X), c_2, c_1$$

CS 525



Notes 12 - Transaction Management

33

$$T_1 = w_1(X), a_1$$

$$T_2 = r_2(X), w_2(X), c_2$$

Cascadeless Schedule

$$S_1 = w_1(X), a_1, r_2(X), w_2(X), c_2$$

Recoverable 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!

CS 525



Notes 12 - Transaction Management

34

Strict Schedules

- Strict** schedules guarantee that to Undo the effect of a transaction we simply have to undo each of its writes
 - Transactions do not read nor write items written by uncommitted transactions

CS 525



Notes 13 - Failure and Recovery

35

$$T_1 = w_1(X), c_1$$

$$T_2 = r_2(X), w_2(X), c_2$$

Cascadeless Schedule

$$S_1 = w_1(X), c_1, r_2(X), w_2(X), c_2$$

Recoverable Schedule

$$S_2 = w_1(X), r_2(X), w_2(X), c_1, c_2$$

Nonrecoverable Schedule

$$S_3 = w_1(X), r_2(X), w_2(X), c_2, c_1$$

CS 525



Notes 12 - Transaction Management

36

Compare Classes

ST \subset CL \subset RC \subset ALL

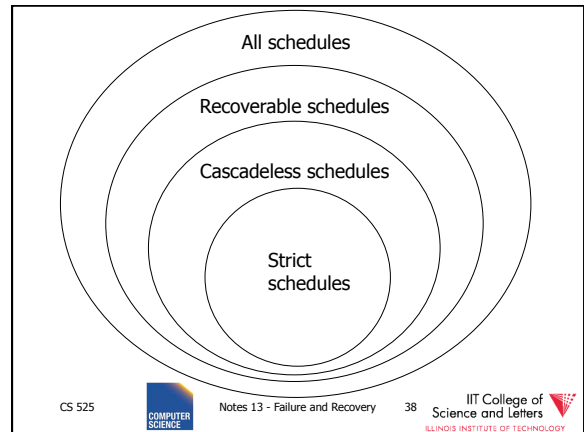
CS 525



Notes 13 - Failure and Recovery

37

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY



CS 525



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

CS 525



Notes 13 - Failure and Recovery

39

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

One solution: undo logging (immediate modification)

due to: Hansel and Gretel, 782 AD

CS 525



Notes 13 - Failure and Recovery

40

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

CS 525



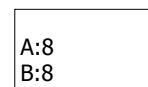
Notes 13 - Failure and Recovery

41

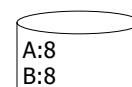
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Undo logging (Immediate modification)

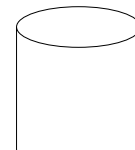
T1: Read (A,t); $t \leftarrow t \times 2$ A=B
Write (A,t);
Read (B,t); $t \leftarrow t \times 2$
Write (B,t);
Output (A);
Output (B);



memory



disk



log

CS 525



Notes 13 - Failure and Recovery

42

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Undo logging (Immediate modification)

T1: Read (A,t); t ← tx2 A=B
 Write (A,t);
 Read (B,t); t ← tx2
 Write (B,t);
 Output (A);
 Output (B);

memory: A: 8, B: 16
 disk: A: 8, B: 8
 log: <T1, start>, <T1, A, 8>

CS 525 Notes 13 - Failure and Recovery 43 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Undo logging (Immediate modification)

T1: Read (A,t); t ← tx2 A=B
 Write (A,t);
 Read (B,t); t ← tx2
 Write (B,t);
 Output (A);
 Output (B);

memory: A: 8, B: 16
 disk: A: 8, B: 16
 log: <T1, start>, <T1, A, 8>, <T1, B, 8>

CS 525 Notes 13 - Failure and Recovery 44 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Undo logging (Immediate modification)

T1: Read (A,t); t ← tx2 A=B
 Write (A,t);
 Read (B,t); t ← tx2
 Write (B,t);
 Output (A);
 Output (B);

memory: A: 8, B: 16
 disk: A: 8, B: 16
 log: <T1, start>, <T1, A, 8>, <T1, B, 8>

CS 525 Notes 13 - Failure and Recovery 45 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Undo logging (Immediate modification)

T1: Read (A,t); t ← tx2 A=B
 Write (A,t);
 Read (B,t); t ← tx2
 Write (B,t);
 Output (A);
 Output (B);

memory: A: 8, B: 16
 disk: A: 8, B: 16
 log: <T1, start>, <T1, A, 8>, <T1, B, 8>, <T1, commit>

CS 525 Notes 13 - Failure and Recovery 46 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, B: 16, Log: <T1, start>, <T1, A, 8>, <T1, B, 8>
 DB: A: 8, B: 8
 Log: (empty)

CS 525 Notes 13 - Failure and Recovery 47 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

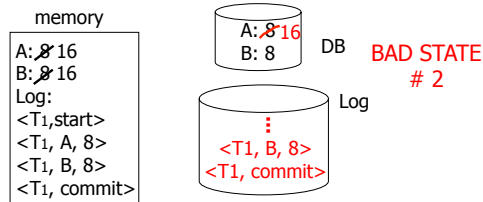
memory: A: 8, B: 16, Log: <T1, start>, <T1, A, 8>, <T1, B, 8>
 DB: A: 8, B: 8
 Log: (empty)

BAD STATE # 1

CS 525 Notes 13 - Failure and Recovery 48 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



CS 525



Notes 13 - Failure and Recovery

49

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 13 - Failure and Recovery

50

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Recovery rules: Undo logging

- For every T_i with $\langle T_i, \text{start} \rangle$ in log:
 - If $\langle T_i, \text{commit} \rangle$ or $\langle T_i, \text{abort} \rangle$ in log, do nothing
 - Else { For all $\langle T_i, X, v \rangle$ in log:
 - write (X, v)
 - output (X)
 Write $\langle T_i, \text{abort} \rangle$ to log

CS 525



Notes 13 - Failure and Recovery

51

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Recovery rules: Undo logging

- For every T_i with $\langle T_i, \text{start} \rangle$ in log:
 - If $\langle T_i, \text{commit} \rangle$ or $\langle T_i, \text{abort} \rangle$ in log, do nothing
 - Else { For all $\langle T_i, X, v \rangle$ in log:
 - write (X, v)
 - output (X)
 Write $\langle T_i, \text{abort} \rangle$ to log

► IS THIS CORRECT??

CS 525



Notes 13 - Failure and Recovery

52

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Recovery rules: Undo logging

- (1) Let S = set of transactions with $\langle T_i, \text{start} \rangle$ in log, but no $\langle T_i, \text{commit} \rangle$ (or $\langle T_i, \text{abort} \rangle$) record in log
- (2) For each $\langle T_i, X, v \rangle$ in log, in reverse order (latest \rightarrow earliest) do:
 - if $T_i \in S$ then {
 - write (X, v)
 - output (X)
- (3) For each $T_i \in S$ do
 - write $\langle T_i, \text{abort} \rangle$ to log

CS 525



Notes 13 - Failure and Recovery

53

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Question

- Can writes of $\langle T_i, \text{abort} \rangle$ records be done in any order (in Step 3)?
 - Example: T_1 and T_2 both write A
 - T_1 executed before T_2
 - T_1 and T_2 both rolled-back
 - $\langle T_1, \text{abort} \rangle$ written but NOT $\langle T_2, \text{abort} \rangle$?
 - $\langle T_2, \text{abort} \rangle$ written but NOT $\langle T_1, \text{abort} \rangle$?



CS 525



Notes 13 - Failure and Recovery

54

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What if failure during recovery?

No problem! ⇒ Undo idempotent

- An operation is called **idempotent** if the number of times it is applied do not effect the result
- For Undo:
 - $\text{Undo}(\log) = \text{Undo}(\text{Undo}(\dots(\text{Undo}(\log)) \dots))$

CS 525



Notes 13 - Failure and Recovery

55

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

CS 525



Notes 13 - Failure and Recovery

56

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 13 - Failure and Recovery

57

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

To discuss:

- Redo logging
- Undo/redo logging, why both?
- Real world actions
- Checkpoints
- Media failures

CS 525



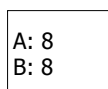
Notes 13 - Failure and Recovery

58

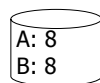
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Redo logging (deferred modification)

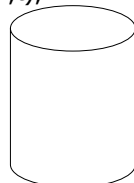
T1: Read(A,t); t ← tx2; write (A,t);
Read(B,t); t ← tx2; write (B,t);
Output(A); Output(B)



memory



DB



LOG

CS 525



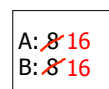
Notes 13 - Failure and Recovery

59

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Redo logging (deferred modification)

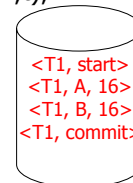
T1: Read(A,t); t ← tx2; write (A,t);
Read(B,t); t ← tx2; write (B,t);
Output(A); Output(B)



memory



DB



LOG

CS 525



Notes 13 - Failure and Recovery

60

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Redo logging (deferred modification)

T1: Read(A,t); t ← tx2; write (A,t);
 Read(B,t); t ← tx2; write (B,t);
 Output(A); Output(B)

memory: A: 8, B: 16
 DB: A: 8, B: 16
 LOG: <T1, start>, <T1, A, 16>, <T1, B, 16>, <T1, commit>

CS 525 Notes 13 - Failure and Recovery 61 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Redo logging (deferred modification)

T1: Read(A,t); t ← tx2; write (A,t);
 Read(B,t); t ← tx2; write (B,t);
 Output(A); Output(B)

memory: A: 8, B: 16
 DB: A: 8, B: 16
 LOG: <T1, start>, <T1, A, 16>, <T1, B, 16>, <T1, end>

CS 525 Notes 13 - Failure and Recovery 62 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

CS 525 Notes 13 - Failure and Recovery 63 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Recovery rules: Redo logging

- For every T_i with $\langle T_i, \text{commit} \rangle$ in log:
 - For all $\langle T_i, X, v \rangle$ in log:
 - Write(X, v)
 - Output(X)

CS 525 Notes 13 - Failure and Recovery 64 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Recovery rules: Redo logging

- For every T_i with $\langle T_i, \text{commit} \rangle$ in log:
 - For all $\langle T_i, X, v \rangle$ in log:
 - Write(X, v)
 - Output(X)

➡ IS THIS CORRECT??

CS 525 Notes 13 - Failure and Recovery 65 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Recovery rules: Redo logging

- (1) Let S = set of transactions with $\langle T_i, \text{commit} \rangle$ (and no $\langle T_i, \text{end} \rangle$) in log
- (2) For each $\langle T_i, X, v \rangle$ in log, in forward order (earliest → latest) do:
 - if $T_i \in S$ then
 - Write(X, v)
 - Output(X)
- (3) For each $T_i \in S$, write $\langle T_i, \text{end} \rangle$

CS 525 Notes 13 - Failure and Recovery 66 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

CS 525



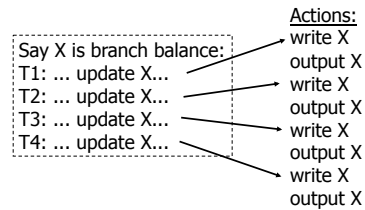
Notes 13 - Failure and Recovery

67

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Combining <Ti, end> Records

- Want to delay DB flushes for hot objects



CS 525



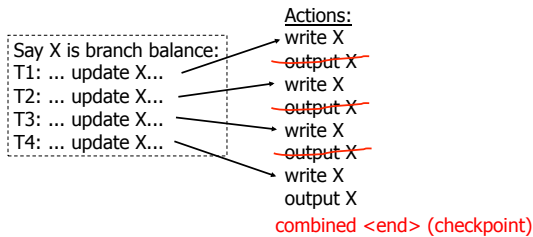
Notes 13 - Failure and Recovery

68

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Combining <Ti, end> Records

- Want to delay DB flushes for hot objects



CS 525



Notes 13 - Failure and Recovery

69

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Solution: Checkpoint

- no <ti, end> actions
- simple 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

CS 525



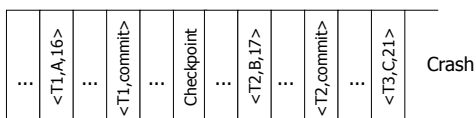
Notes 13 - Failure and Recovery

70

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example: what to do at recovery?

Redo log (disk):



CS 525



Notes 13 - Failure and Recovery

71

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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!

CS 525



Notes 13 - Failure and Recovery

72

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 13 - Failure and Recovery

73

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

CS 525



Notes 13 - Failure and Recovery

74

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Solution: undo/redo logging!

Update \Rightarrow $\langle T_i, Xid, \text{New } X \text{ val}, \text{Old } X \text{ val} \rangle$
page X

CS 525



Notes 13 - Failure and Recovery

75

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rules

- Page X can be flushed before or after T_i commit
- Log record flushed before corresponding updated page (WAL)
- Flush at commit (log only)

CS 525



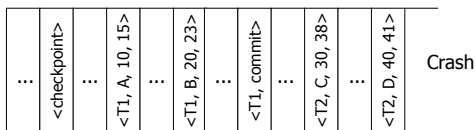
Notes 13 - Failure and Recovery

76

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Undo/Redo logging what to do at recovery?

log (disk):



CS 525



Notes 13 - Failure and Recovery

77

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

CS 525

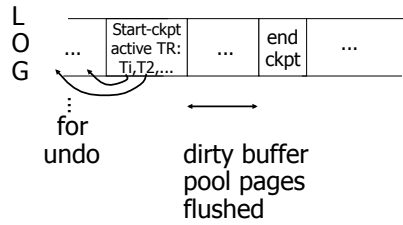


Notes 13 - Failure and Recovery

78

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Non-quietse checkpoint



CS 525

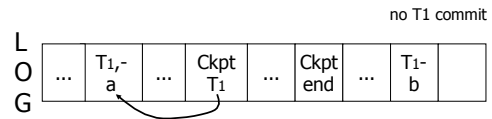


Notes 13 - Failure and Recovery

79

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Examples what to do at recovery time?



CS 525

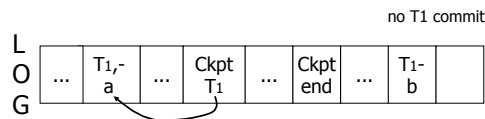


Notes 13 - Failure and Recovery

80

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Examples what to do at recovery time?



➡ Undo T1 (undo a,b)

CS 525

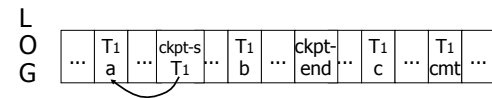


Notes 13 - Failure and Recovery

81

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example



CS 525

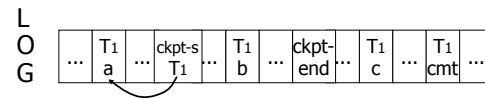


Notes 13 - Failure and Recovery

82

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example



➡ Redo T1: (redo b,c)

CS 525

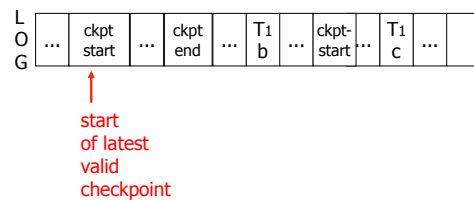


Notes 13 - Failure and Recovery

83

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Recover From Valid Checkpoint:



CS 525



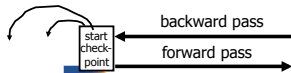
Notes 13 - Failure and Recovery

84

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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



Real world actions

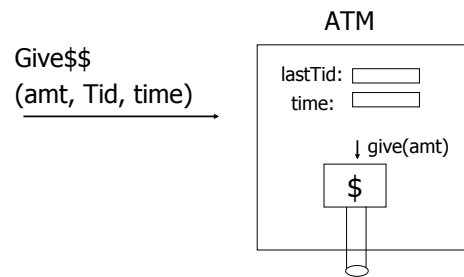
E.g., dispense cash at ATM

$$T_i = a_1 a_2 \dots a_j \dots a_n$$

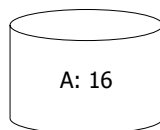
↓
\$

Solution

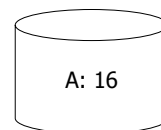
- (1) execute real-world actions after commit
- (2) try to make idempotent



Media failure (loss of non-volatile storage)



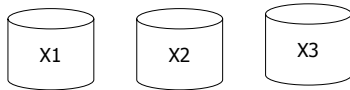
Media failure (loss of non-volatile storage)



Solution: Make copies of data!

Example 1 Triple modular redundancy

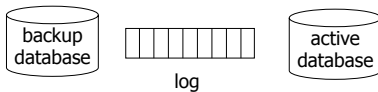
- Keep 3 copies on separate disks
- Output(X) --> three outputs
- Input(X) --> three inputs + vote



Example #2 Redundant writes, Single reads

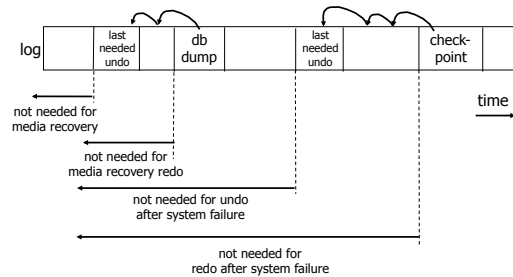
- Keep N copies on separate disks
 - Output(X) --> N outputs
 - Input(X) --> Input one copy
- done { - if ok,
- else try another one
- ⇒ Assumes bad data can be detected

Example #3: DB Dump + Log



- If active database is lost,
 - restore active database from backup
 - bring up-to-date using redo entries in log

When can log be discarded?



Practical Recovery with ARIES

- **ARIES**
 - Algorithms for Recovery and Isolation Exploiting Semantics
- Implemented in, e.g.,
 - DB2
 - MSSQL

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

Log Entry Structure

- **LSN**
 - Log sequence number
 - Order of entries in the log
 - Usually **log file id** and **offset** for direct access

CS 525



Notes 13 - Failure and Recovery

97

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- **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

CS 525



Notes 13 - Failure and Recovery

98

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

CS 525



Notes 13 - Failure and Recovery

99

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

CS 525



Notes 13 - Failure and Recovery

100

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 intention to modify
 - Insert **<PageId,ReclSN>** with **ReclSN** being the current end of the log
 - Flushing a page removes it from the Dirty page table

CS 525



Notes 13 - Failure and Recovery

101

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Dirty Page Table

- Used for checkpointing
- Used for recovery to figure out what to redo

CS 525



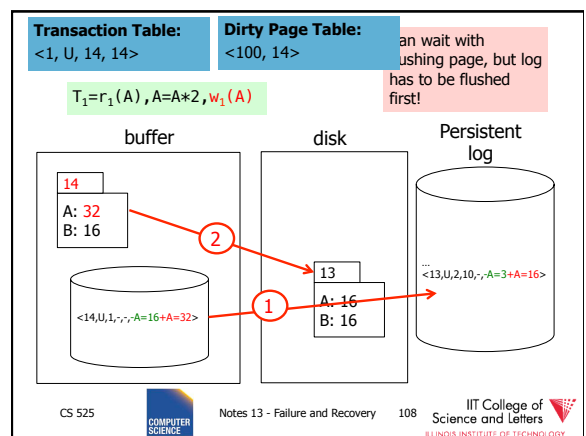
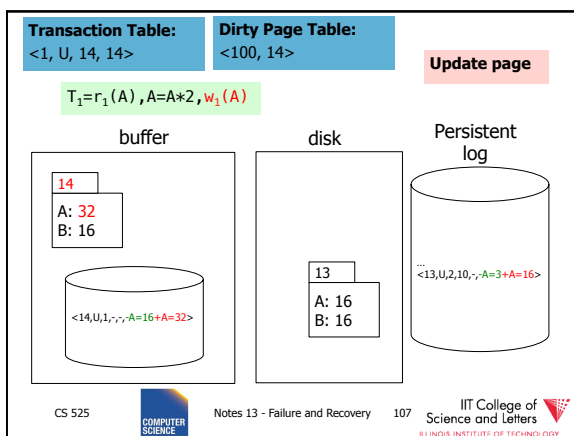
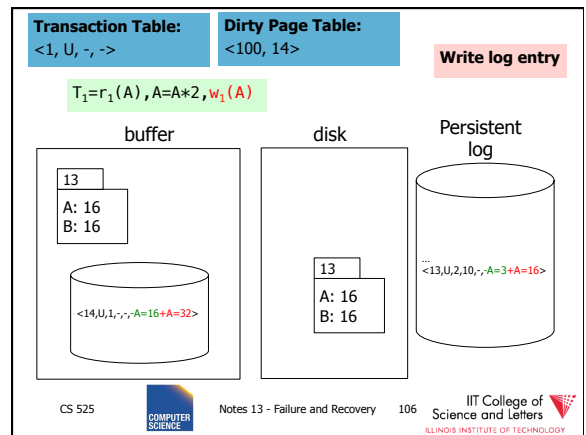
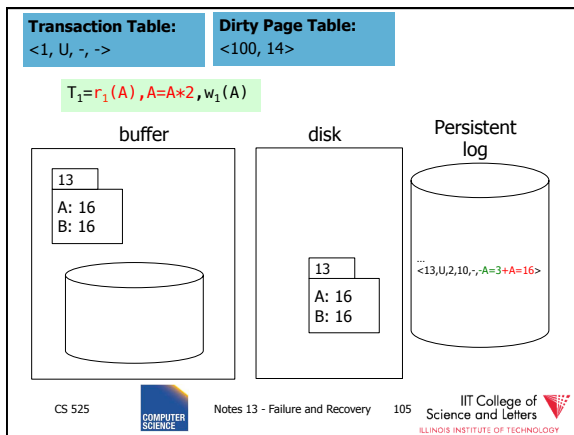
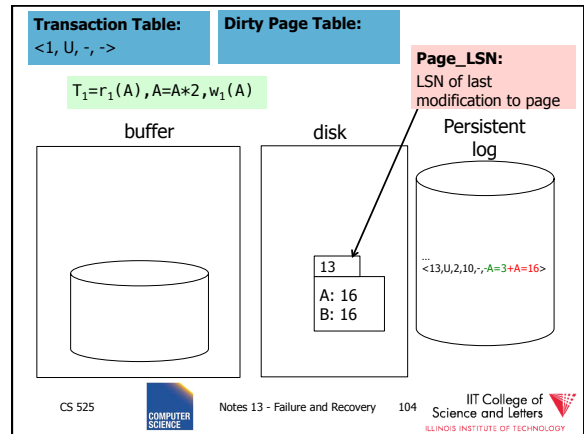
Notes 13 - Failure and Recovery

102

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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



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

CS 525



Notes 13 - Failure and Recovery

109

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

CS 525



Notes 13 - Failure and Recovery

110

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

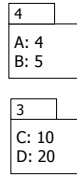
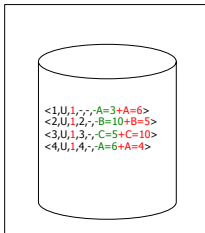
Transaction Table:

<1, U, 4, 4>

Undo T₁

T₁ = w₁(A), w₁(B), w₁(C), w₁(A), a₁

buffer



CS 525



Notes 13 - Failure and Recovery

111

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

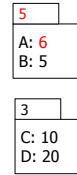
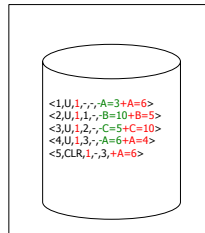
Transaction Table:

<1, U, 5, 3>

Undo T₁

T₁ = w₁(A), w₁(B), w₁(C), w₁(A), a₁

buffer



CS 525



Notes 13 - Failure and Recovery

112

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

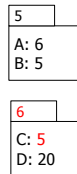
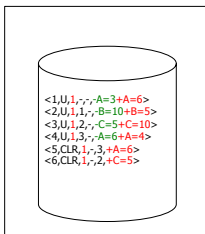
Transaction Table:

<1, U, 6, 2>

Undo T₁

T₁ = w₁(A), w₁(B), w₁(C), w₁(A), a₁

buffer



CS 525



Notes 13 - Failure and Recovery

113

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

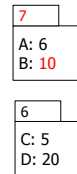
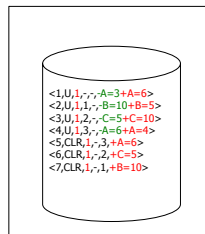
Transaction Table:

<1, U, 7, 1>

Undo T₁

T₁ = w₁(A), w₁(B), w₁(C), w₁(A), a₁

buffer



CS 525



Notes 13 - Failure and Recovery

114

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Transaction Table:
 <1, U, 8, ->

Undo T₁

T₁ = w₁(A), w₁(B), w₁(C), w₁(A), a₁

buffer

CS 525 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

CS 525 Notes 13 - Failure and Recovery 116 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Restart Recovery

1. Analysis Phase
2. Redo Phase
3. Undo Phase

CS 525 Notes 13 - Failure and Recovery 117 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Analysis Phase

- 1) Determine LSN of last checkpoint using Master Record
- 2) Log Dirty Page Table and Transaction Table from checkpoint record
- 3) **RedoLSN** = min(ReclSN) from Dirty Page Table or checkpoint LSN if no dirty page

CS 525 Notes 13 - Failure and Recovery 118 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 to Page to Dirty Page Table
 - Add Transaction to Transaction Table or update LastLSN
 - Transaction end entry
 - Remove transaction from Transaction Table

CS 525 Notes 13 - Failure and Recovery 119 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525 Notes 13 - Failure and Recovery 120 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
- One redo once
 - Only redo operations that have not been reflected on disk (PageLSN)

CS 525



Notes 13 - Failure and Recovery

121

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 $PageLSN \geq LSN$ then operation already reflected on disk
 - Skip log entry
 - Otherwise apply update

CS 525



Notes 13 - Failure and Recovery

122

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Redo Phase

- Result
 - State of DB before Failure

CS 525



Notes 13 - Failure and Recovery

123

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Undo Phase

- Scan log backwards from end using Transaction Table
 - Repeatedly take log entry with max LSN from all the current action to be undone for each transaction
 - Write CLR
 - Update Transaction Table

CS 525



Notes 13 - Failure and Recovery

124

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Undo Phase

- All unfinished transactions have been rolled back

CS 525



Notes 13 - Failure and Recovery

125

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 CLR we do not undo this action again

CS 525



Notes 13 - Failure and Recovery

126

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

$T_1 = w_1(A), w_1(B), w_1(C), w_1(A), c_1$
 $T_2 = w_1(X), r(A), w(A)$

Log

```

<1,begin(T1),->
<2,begin(T2),->
<3,write(A,T1),1>
<4,write(X,T2),2>
<5,write(B,T1),3>
<6,write(C,T1),5>
<7,write(A,T1),6>
<8,commit(T1),7>
<9,write(A,T2),4>

```

CS 525 Notes 13 - Failure and Recovery 127 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

$T_1 = w_1(A), w_1(B), w_1(C), w_1(A), c_1$
 $T_2 = w_1(X), r(A), w(A)$

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 page table (RecLSN 3)
- add X to page table (RecLSN 4)
- add B to page table (RecLSN 5)
- add C to page table (RecLSN 6)
- remove T_1 from Transaction Table (rec. 8)

Log

```

<1,begin(T1),->
<2,begin(T2),->
<3,write(A,T1),1>
<4,write(X,T2),2>
<5,write(B,T1),3>
<6,write(C,T1),5>
<7,write(A,T1),6>
<8,commit(T1),7>
<9,write(A,T2),4>

```

CS 525 Notes 13 - Failure and Recovery 128 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

$T_1 = w_1(A), w_1(B), w_1(C), w_1(A), c_1$
 $T_2 = w_1(X), r(A), w(A)$

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(T1),->
<2,begin(T2),->
<3,write(A,T1),1>
<4,write(X,T2),2>
<5,write(B,T1),3>
<6,write(C,T1),5>
<7,write(A,T1),6>
<8,commit(T1),7>
<9,write(A,T2),4>

```

CS 525 Notes 13 - Failure and Recovery 129 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

$T_1 = w_1(A), w_1(B), w_1(C), w_1(A), c_1$
 $T_2 = w_1(X), r(A), w(A)$

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,begin(T1),->
<2,begin(T2),->
<3,write(A,T1),1>
<4,write(X,T2),2>
<5,write(B,T1),3>
<6,write(C,T1),5>
<7,write(A,T1),6>
<8,commit(T1),7>
<9,write(A,T2),4>

```

CS 525 Notes 13 - Failure and Recovery 130 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

$T_1 = w_1(A), w_1(B), w_1(C), w_1(A), c_1$
 $T_2 = w_1(X), r(A), w(A)$

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(T1),->
<2,begin(T2),->
<3,write(A,T1),1>
<4,write(X,T2),2>
<5,write(B,T1),3>
<6,write(C,T1),5>
<7,write(A,T1),6>
<8,commit(T1),7>
<9,write(A,T2),4>
<10,CLR(A,T2),4>
<11,CLR(X,T2),->

```

CS 525 Notes 13 - Failure and Recovery 131 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

CS 525 Notes 13 - Failure and Recovery 132 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Media Recovery

- What if disks where log or DB is stored fails
 - ->keep backups of log + DB state

CS 525



Notes 13 - Failure and Recovery

133

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

CS 525



Notes 13 - Failure and Recovery

134

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 13 - Failure and Recovery

135

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Summary

- Consistency of data
- One source of problems: failures
 - Logging
 - Redundancy
- Another source of problems:
Data Sharing..... next

CS 525




Notes 13 - Failure and Recovery

136

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

CS 525: Advanced Database Organization

14: Concurrency Control

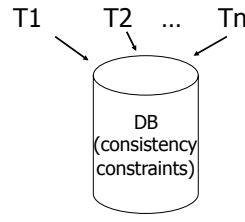


Boris Glavic

Slides: adapted from a [course](#) taught by [Hector Garcia-Molina](#), Stanford InfoLab

CS 525
COMPUTER SCIENCE
Notes 14 - Concurrency Control
1
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Chapter 18 [18] Concurrency Control



CS 525
COMPUTER SCIENCE
Notes 14 - Concurrency Control
2
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example:

<p>T1: Read(A) A ← A+100 Write(A) Read(B) B ← B+100 Write(B)</p>	<p>T2: Read(A) A ← A×2 Write(A) Read(B) B ← B×2 Write(B)</p>
--	--

Constraint: A=B

CS 525
COMPUTER SCIENCE
Notes 14 - Concurrency Control
3
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule A

<p>T1 Read(A); A ← A+100 Write(A); Read(B); B ← B+100; Write(B);</p>	<p>T2 Read(A); A ← A×2; Write(A); Read(B); B ← B×2; Write(B);</p>
--	---

CS 525
COMPUTER SCIENCE
Notes 14 - Concurrency Control
4
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule A

<p>T1 Read(A); A ← A+100 Write(A); Read(B); B ← B+100; Write(B);</p>	<p>T2 Read(A); A ← A×2; Write(A); Read(B); B ← B×2; Write(B);</p>
--	---

A	B
25	25
125	
250	125
250	250
250	250

CS 525
COMPUTER SCIENCE
Notes 14 - Concurrency Control
5
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY


Schedule B

<p>T1 Read(A); A ← A+100 Write(A); Read(B); B ← B+100; Write(B);</p>	<p>T2 Read(A); A ← A×2; Write(A); Read(B); B ← B×2; Write(B);</p>
--	---

CS 525
COMPUTER SCIENCE
Notes 14 - Concurrency Control
6
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY


Schedule B

T1	T2	A	B
		25	25
	Read(A); A ← A×2;		
	Write(A);	50	
	Read(B); B ← B×2;		
	Write(B);		50
Read(A); A ← A+100			
Write(A);		150	
Read(B); B ← B+100;			
Write(B);			150
		150	150

CS 525  Notes 14 - Concurrency Control 7 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


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);

CS 525  Notes 14 - Concurrency Control 8 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


Schedule C

T1	T2	A	B
		25	25
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

CS 525  Notes 14 - Concurrency Control 9 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+100;	
Write(B);	
	Read(B); B ← B×2;
	Write(B);

CS 525  Notes 14 - Concurrency Control 10 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule D


T1	T2	A	B
		25	25
Read(A); A ← A+100			
Write(A);		125	
	Read(A); A ← A×2;		
	Write(A);	250	
	Read(B); B ← B×2;		
	Write(B);		50
Read(B); B ← B+100;			
Write(B);			150
		250	150

CS 525  Notes 14 - Concurrency Control 11 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule E

Same as Schedule D but with new T2'

T1	T2'
Read(A); A ← A+100	
Write(A);	
	Read(A); A ← A×1;
	Write(A);
Read(B); B ← B+100;	
Write(B);	
	Read(B); B ← B×1;
	Write(B);

CS 525  Notes 14 - Concurrency Control 12 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule E Same as Schedule D but with new T2'

		A	B
T1	T2'	25	25
Read(A); A ← A+100 Write(A);	Read(A); A ← A×1; Write(A);	125	125
	Read(B); B ← B×1; Write(B);	25	25
Read(B); B ← B+100; Write(B);		125	125
		125	125

CS 525 Notes 14 - Concurrency Control
13 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

CS 525 Notes 14 - Concurrency Control
14 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Definition: Serial Schedule

- No transactions are interleaved
 - There exists no two operations from transactions T_i and T_j so that both operations are executed before either transaction commits

CS 525 Notes 14 - Concurrency Control
15 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

$S_2 = r_2(A), w_2(A), r_1(A), w_1(A), r_2(B), w_2(B), c_2, r_1(B), w_1(B), c_1$

CS 525 Notes 12 - Transaction Management
16 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Compare Classes

$S \subset ST \subset CL \subset RC \subset ALL$

CS 525 Notes 13 - Failure and Recovery
17 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

CS 525 Notes 13 - Failure and Recovery
18 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Why not serial schedules?

- No concurrency! ☹️

CS 525



Notes 14 - Concurrency Control

19

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:

$S_c = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$

CS 525



Notes 14 - Concurrency Control

20

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

CS 525



Notes 14 - Concurrency Control

21

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example:

$S_c = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$

$S_c' = r_1(A)w_1(A) \underbrace{r_1(B)w_1(B)r_2(A)w_2(A)}_{T_1} \underbrace{r_2(B)w_2(B)}_{T_2}$

CS 525



Notes 14 - Concurrency Control

22

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

However, for S_d :

$S_d = r_1(A)w_1(A)r_2(A)w_2(A) \underbrace{r_2(B)w_2(B)r_1(B)w_1(B)}_{\text{crossed}}$

- as a matter of fact, T_2 must precede T_1 in any equivalent schedule, i.e., $T_2 \rightarrow T_1$

CS 525



Notes 14 - Concurrency Control

23

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- $T_2 \rightarrow T_1$
- Also, $T_1 \rightarrow T_2$



- ⇒ S_d cannot be rearranged into a serial schedule
- ⇒ S_d is not “equivalent” to any serial schedule
- ⇒ S_d is “bad”

CS 525



Notes 14 - Concurrency Control

24

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Returning to Sc

$$Sc = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$$

CS 525 Notes 14 - Concurrency Control 25 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Returning to Sc

$$Sc = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$$

no cycles \Rightarrow Sc is "equivalent" to a serial schedule (in this case T_1, T_2)

CS 525 Notes 14 - Concurrency Control 26 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Concepts

Transaction: sequence of $r_i(x)$, $w_i(x)$ actions

Conflicting actions:

$$\begin{matrix} r_1(A) & w_2(A) & w_1(A) \\ & \swarrow \quad \searrow & \\ & w_2(A) & r_1(A) & w_2(A) \end{matrix}$$

Schedule: represents chronological order in which actions are executed

Serial schedule: no interleaving of actions or transactions

CS 525 Notes 14 - Concurrency Control 27 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

What about concurrent actions?

CS 525 Notes 14 - Concurrency Control 28 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

What about concurrent actions?

CS 525 Notes 14 - Concurrency Control 29 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

So net effect is either

- $S = \dots r_1(x) \dots w_2(b) \dots$ or
- $S = \dots w_2(B) \dots r_1(x) \dots$

CS 525 Notes 14 - Concurrency Control 30 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

CS 525 Notes 14 - Concurrency Control 31 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”

CS 525 Notes 14 - Concurrency Control 32 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

CS 525 Notes 14 - Concurrency Control 33 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Conflict Equivalence

- Define equivalence based on the order of conflicting actions

CS 525 Notes 14 - Concurrency Control 34 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Definition

S_1, S_2 are conflict equivalent schedules if S_1 can be transformed into S_2 by a series of swaps on non-conflicting actions.

Alternatively:
 If the order of conflicting actions in S_1 and S_2 is the same

CS 525 Notes 14 - Concurrency Control 35 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

CS 525 Notes 14 - Concurrency Control 36 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Definition

A schedule is conflict serializable (CSR) if it is conflict equivalent to some serial schedule.

CS 525



Notes 14 - Concurrency Control

37

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

CS 525



Notes 14 - Concurrency Control

38

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?

CS 525



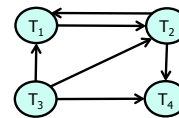
Notes 14 - Concurrency Control

39

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?

CS 525



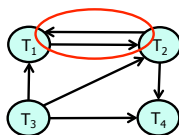
Notes 14 - Concurrency Control

40

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?

CS 525



Notes 14 - Concurrency Control

41

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Another Exercise:

- What is P(S) for
 $S = w_1(A) r_2(A) r_3(A) w_4(A) ?$

CS 525



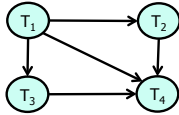
Notes 14 - Concurrency Control

42

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Another Exercise:

- What is P(S) for
 $S = w_1(A) r_2(A) r_3(A) w_4(A)$?



CS 525



Notes 14 - Concurrency Control

43

Lemma

S_1, S_2 conflict equivalent $\Rightarrow P(S_1)=P(S_2)$

CS 525



Notes 14 - Concurrency Control

44

Lemma

S_1, S_2 conflict equivalent $\Rightarrow P(S_1)=P(S_2)$

Proof: $(a \rightarrow b \text{ same as } \neg b \rightarrow \neg a)$

Assume $P(S_1) \neq P(S_2)$

$\Rightarrow \exists T_i: T_i \rightarrow T_j$ in S_1 and not in S_2

$\Rightarrow S_1 = \dots p_i(A) \dots q_j(A) \dots$

$S_2 = \dots q_j(A) \dots p_i(A) \dots$

$\left. \begin{array}{l} p_i, q_j \\ \text{conflict} \end{array} \right\}$

$\Rightarrow S_1, S_2$ not conflict equivalent

CS 525



Notes 14 - Concurrency Control

45

Note: $P(S_1)=P(S_2) \not\Rightarrow S_1, S_2$ conflict equivalent

CS 525



Notes 14 - Concurrency Control

46

Note: $P(S_1)=P(S_2) \not\Rightarrow S_1, S_2$ conflict equivalent

Counter example:

$S_1 = w_1(A) r_2(A) \quad w_2(B) r_1(B)$

$S_2 = r_2(A) w_1(A) \quad r_1(B) w_2(B)$

CS 525



Notes 14 - Concurrency Control

47

Theorem

$P(S_1)$ acyclic $\iff S_1$ conflict serializable

(\Leftarrow) 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

CS 525



Notes 14 - Concurrency Control

48

Theorem

$P(S_1)$ acyclic $\iff S_1$ conflict serializable

(\implies) Assume $P(S_1)$ is acyclic

Transform S_1 as follows:

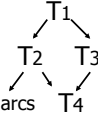
(1) Take T_1 to be transaction with no incident arcs

(2) Move all T_1 actions to the front

$S_1 = \dots\dots q_j(A)\dots\dots p_1(A)\dots\dots$

(3) we now have $S_1 = \langle T_1 \text{ actions} \rangle \langle \dots \text{rest} \dots \rangle$

(4) repeat above steps to serialize rest!



CS 525



Notes 14 - Concurrency Control

49

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's the damage?

- Classification of "bad" things that can happen schedules
 - Lost updates
 - Dirty reads
 - Nonrepeatable reads
 - Phantom reads (later)

CS 525



Notes 14 - Concurrency Control

50

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Lost Updates

- The value written by a transaction is overwritten by another transaction
- The update of the first transaction is "lost"

CS 525

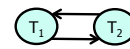


Notes 14 - Concurrency Control

51

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Lost Update



T_1	T_2	
Read(A), A += 100		A=50
	Read(A), A +=200	T_1 : A = 150
Write(A);		T_2 : A = 250
	Write(A);	A = 150
Commit		A = 250
	Commit	

$S_1 = r_1(A), r_2(A), w_1(A), w_2(A), c_1, c_2$

CS 525



Notes 14 - Concurrency Control

52

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Inconsistent 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

CS 525

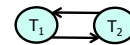


Notes 14 - Concurrency Control

53

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Inconsistent Read



T_1	T_2	
Read(A), A += 100		A=B=150
Write(A);		A = 250
	Read(A), sum = A	sum = 250
	Read(B); sum+=B	sum = 400
Read(B), B -= 100		B=50
Write(B)		
Commit		
	Commit	

$S_1 = r_1(A), w_1(A), r_2(A), r_2(B), r_1(B), w_1(B), c_1, c_2$

CS 525



Notes 14 - Concurrency Control

54

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

CS 525

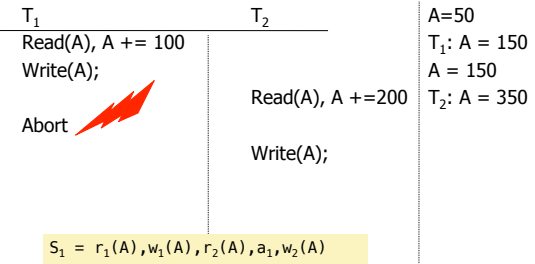


Notes 14 - Concurrency Control

55

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Dirty Read



CS 525



Notes 14 - Concurrency Control

56

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

CS 525



Notes 14 - Concurrency Control

57

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**

CS 525



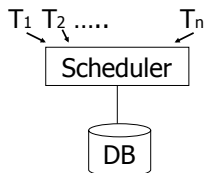
Notes 14 - Concurrency Control

58

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

How to enforce serializable schedules?

Option 2: prevent $P(S)$ cycles from occurring



CS 525



Notes 14 - Concurrency Control

59

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**

CS 525



Notes 14 - Concurrency Control

60

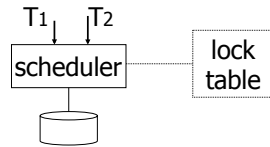
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

A locking protocol

Two new actions:

lock (exclusive): $li(A)$

unlock: $ui(A)$



CS 525



Notes 14 - Concurrency Control

61

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rule #1: Well-formed transactions

$T_i: \dots li(A) \dots pi(A) \dots ui(A) \dots$

- 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

CS 525



Notes 14 - Concurrency Control

62

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Rule #2 Legal scheduler

$S = \dots li(A) \dots ui(A) \dots$

←
no $lj(A)$

- 4) Only one transaction can hold a lock on A at the same time

CS 525



Notes 14 - Concurrency Control

63

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Exercise:

- What schedules are legal?
What transactions are well-formed?

$S_1 = li(A)li(B)r_1(A)w_1(B)l_2(B)u_1(A)u_1(B)$

$r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)$

$S_2 = li(A)r_1(A)w_1(B)u_1(A)u_1(B)$

$l_2(B)r_2(B)w_2(B)l_3(B)r_3(B)u_3(B)$

$S_3 = li(A)r_1(A)u_1(A)l_1(B)w_1(B)u_1(B)$

$l_2(B)r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)$

CS 525



Notes 14 - Concurrency Control

64

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Exercise:

- What schedules are legal?
What transactions are well-formed?

$S_1 = li(A)li(B)r_1(A)w_1(B)l_2(B)u_1(A)u_1(B)$

$r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)$

$S_2 = li(A)r_1(A)w_1(B)u_1(A)u_1(B)$

$l_2(B)r_2(B)w_2(B)l_3(B)r_3(B)u_3(B)$

$S_3 = li(A)r_1(A)u_1(A)l_1(B)w_1(B)u_1(B)$

$l_2(B)r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)$

CS 525



Notes 14 - Concurrency Control

65

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule F

T1	T2
$li(A); Read(A)$	
$A \leftarrow A+100; Write(A); u_1(A)$	
	$l_2(A); Read(A)$
	$A \leftarrow Ax2; Write(A); u_2(A)$
	$l_2(B); Read(B)$
	$B \leftarrow Bx2; Write(B); u_2(B)$
$l_1(B); Read(B)$	
$B \leftarrow B+100; Write(B); u_1(B)$	

CS 525



Notes 14 - Concurrency Control


66

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule F

	A	B
T1	25	25
$l_1(A); \text{Read}(A)$		
$A \leftarrow A+100; \text{Write}(A); u_1(A)$	125	
	250	
T2		50
$l_2(A); \text{Read}(A)$		
$A \leftarrow Ax2; \text{Write}(A); u_2(A)$		
$l_2(B); \text{Read}(B)$		
$B \leftarrow Bx2; \text{Write}(B); u_2(B)$		150
	250	150

$l_1(B); \text{Read}(B)$
 $B \leftarrow B+100; \text{Write}(B); u_1(B)$


CS 525  Notes 14 - Concurrency Control 67 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

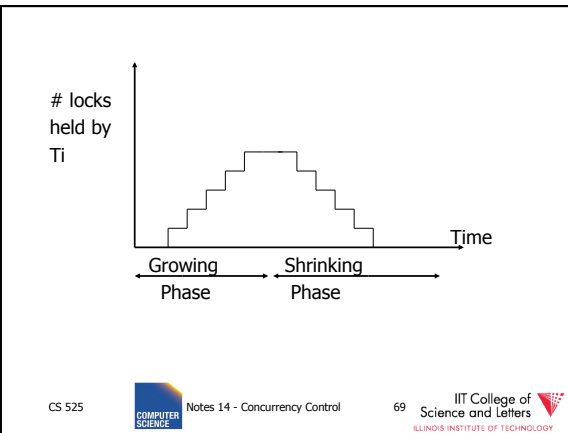
Rule #3 Two phase locking (2PL) for transactions

$T_i = \dots l_i(A) \dots u_i(A) \dots$

← no unlocks no locks →


5) A transaction does not require new locks after its first unlock operation

CS 525  Notes 14 - Concurrency Control 68 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY




Schedule G

T1	T2
$l_1(A); \text{Read}(A)$	
$A \leftarrow A+100; \text{Write}(A)$	
$l_1(B); u_1(A)$	
	$l_2(A); \text{Read}(A)$ delayed
	$A \leftarrow Ax2; \text{Write}(A); l_2(B)$

CS 525  Notes 14 - Concurrency Control 70 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


Schedule G

T1	T2
$l_1(A); \text{Read}(A)$	
$A \leftarrow A+100; \text{Write}(A)$	
$l_1(B); u_1(A)$	
	$l_2(A); \text{Read}(A)$ delayed
	$A \leftarrow Ax2; \text{Write}(A); l_2(B)$
$\text{Read}(B); B \leftarrow B+100$	
$\text{Write}(B); u_1(B)$	

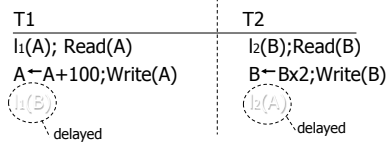
CS 525  Notes 14 - Concurrency Control 71 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule G

T1	T2
$l_1(A); \text{Read}(A)$	
$A \leftarrow A+100; \text{Write}(A)$	
$l_1(B); u_1(A)$	
	$l_2(A); \text{Read}(A)$ delayed
	$A \leftarrow Ax2; \text{Write}(A); l_2(B)$
$\text{Read}(B); B \leftarrow B+100$	
$\text{Write}(B); u_1(B)$	
	$l_2(B); u_2(A); \text{Read}(B)$
	$B \leftarrow Bx2; \text{Write}(B); u_2(B);$

CS 525  Notes 14 - Concurrency Control 72 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule H (T₂ reversed)



CS 525



Notes 14 - Concurrency Control

73

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
 - T₁ is waiting for T₂ and is making no progress
 - T₂ is waiting for T₁ and is making no progress
 - -> if we do not do anything they would wait forever

CS 525



Notes 14 - Concurrency Control

74

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!

CS 525



Notes 14 - Concurrency Control

75

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Next step:

Show that rules #1,2,3 \Rightarrow conflict-serializable schedules

CS 525



Notes 14 - Concurrency Control

76

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 $\langle u_i(A), u_j(A) \rangle$, $\langle l_i(A), r_j(A) \rangle$, ...

CS 525



Notes 14 - Concurrency Control

77

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Theorem Rules #1,2,3 \Rightarrow conflict-serializable schedule (2PL)

CS 525



Notes 14 - Concurrency Control

78

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Theorem Rules #1,2,3 \Rightarrow conflict serializable schedule (2PL)

To help in proof:

Definition Shrink(T_i) = SH(T_i) = first unlock action of T_i

CS 525



Notes 14 - Concurrency Control

79

Lemma

$T_i \rightarrow T_j$ in $S \Rightarrow SH(T_i) <_S SH(T_j)$

CS 525



Notes 14 - Concurrency Control

80

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

$S = \dots p_i(A) \dots q_j(A) \dots$; p, q conflict

By rules 1,2:

$S = \dots p_i(A) \dots u_i(A) \dots l_j(A) \dots q_j(A) \dots$

CS 525



Notes 14 - Concurrency Control

81

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

$S = \dots p_i(A) \dots q_j(A) \dots$; p, q conflict

By rules 1,2:

$S = \dots p_i(A) \dots u_i(A) \dots l_j(A) \dots q_j(A) \dots$

By rule 3: $SH(T_i) \quad SH(T_j)$

So, $SH(T_i) <_S SH(T_j)$

CS 525



Notes 14 - Concurrency Control

82

Theorem Rules #1,2,3 \Rightarrow conflict serializable schedule (2PL)

Proof:

(1) Assume $P(S)$ has cycle

$T_1 \rightarrow T_2 \rightarrow \dots \rightarrow T_n \rightarrow T_1$

(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

CS 525



Notes 14 - Concurrency Control

83

2PL subset of Serializable

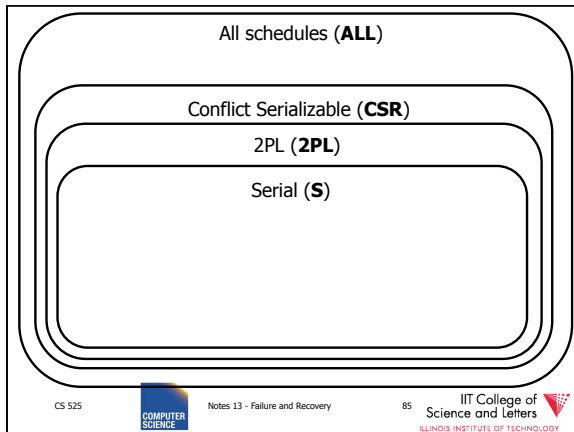
S \subset 2PL \subset CSRC \subset ALL

CS 525



Notes 14 - Concurrency Control

84



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).

CS 525 Notes 14 - Concurrency Control 86 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)

CS 525 Notes 14 - Concurrency Control 87 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

CS 525 Notes 14 - Concurrency Control 88 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Shared locks

So far:

$S = \dots l_1(A) r_1(A) u_1(A) \dots l_2(A) r_2(A) u_2(A) \dots$

↙ ↘
Do not conflict

CS 525 Notes 14 - Concurrency Control 89 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Shared locks

So far:

$S = \dots l_1(A) r_1(A) u_1(A) \dots l_2(A) r_2(A) u_2(A) \dots$

↙ ↘
Do not conflict



Instead:

$S = \dots l_{s1}(A) r_1(A) l_{s2}(A) r_2(A) \dots u_{s1}(A) u_{s2}(A)$

CS 525 Notes 14 - Concurrency Control 90 IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY



Lock actions
 l-t(A): lock A in t mode (t is S or X)
 u-t(A): unlock t mode (t is S or X)

Shorthand:
 u_i(A): unlock whatever modes
 T_i has locked A

CS 525  Notes 14 - Concurrency Control 91  IIT College of Science and Letters
 ILLINOIS INSTITUTE OF TECHNOLOGY



Rule #1 Well formed transactions

T_i = ... l-S₁(A) ... r₁(A) ... u₁(A) ...
 T_i = ... l-X₁(A) ... w₁(A) ... u₁(A) ...

CS 525  Notes 14 - Concurrency Control 92  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-X₁(A) ... r₁(A) ... w₁(A) ... u(A) ...



CS 525  Notes 14 - Concurrency Control 93  IIT College of Science and Letters
 ILLINOIS INSTITUTE OF TECHNOLOGY

- 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 = ... l-S₁(A) ... r₁(A) ... l-X₁(A) ... w₁(A) ... u(A) ...



Think of
 - Get 2nd lock on A, or
 - Drop S, get X lock

CS 525  Notes 14 - Concurrency Control 94  IIT College of Science and Letters
 ILLINOIS INSTITUTE OF TECHNOLOGY

Rule #2 Legal scheduler

S = ... l-S₁(A) ... u₁(A) ...
 ←→
 no l-X_j(A)

S = ... l-X_i(A) ... u_i(A) ...
 ←→
 no l-X_j(A)
 no l-S_j(A)



CS 525  Notes 14 - Concurrency Control 95  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

CS 525  Notes 14 - Concurrency Control 96  IIT College of Science and Letters
 ILLINOIS INSTITUTE OF TECHNOLOGY

Rule # 3 2PL transactions

No change except for upgrades:

- (I) If upgrade gets more locks (e.g., $S \rightarrow \{S, X\}$) then no change!
- (II) If upgrade releases read (shared) lock (e.g., $S \rightarrow X$) - can be allowed in growing phase

CS 525



Notes 14 - Concurrency Control

97

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 $\text{comp}(t_i, r_j)$

$l-t_i(A), u-r_j(A)$ do not conflict if $\text{comp}(t_i, r_j)$

CS 525



Notes 14 - Concurrency Control

98

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Lock types beyond S/X

Examples:

- (1) increment lock
- (2) update lock

CS 525



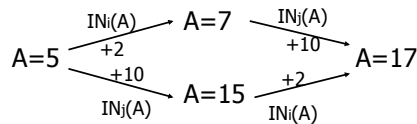
Notes 14 - Concurrency Control

99

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Example (1): increment lock

- Atomic increment action: $IN_i(A)$
 $\{ \text{Read}(A); A \leftarrow A+k; \text{Write}(A) \}$
- $IN_i(A), IN_j(A)$ do not conflict!



CS 525



Notes 14 - Concurrency Control

100

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Comp

	S	X	I
S			
X			
I			

CS 525



Notes 14 - Concurrency Control

101

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

CS 525



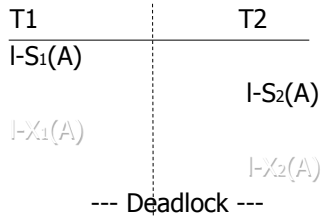
Notes 14 - Concurrency Control

102

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Update locks

A common deadlock problem with upgrades:



CS 525



Notes 14 - Concurrency Control

103

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Solution

If T_i wants to read A and knows it may later want to write A, it requests update lock (not shared)

CS 525



Notes 14 - Concurrency Control

104

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

New request

Comp

Lock
already
held in

	S	X	U
S			
X			
U			

CS 525



Notes 14 - Concurrency Control

105

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?

CS 525



Notes 14 - Concurrency Control

106

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Note: object A may be locked in different modes at the same time...

$$S_1 = \dots I-S_1(A) \dots I-S_2(A) \dots I-U_3(A) \dots \begin{cases} I-S_4(A) \dots? \\ I-U_4(A) \dots? \end{cases}$$

CS 525



Notes 14 - Concurrency Control

107

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Note: object A may be locked in different modes at the same time...

$$S_1 = \dots I-S_1(A) \dots I-S_2(A) \dots I-U_3(A) \dots \begin{cases} I-S_4(A) \dots? \\ I-U_4(A) \dots? \end{cases}$$

- To grant a lock in mode t, mode t must be compatible with all currently held locks on object

CS 525



Notes 14 - Concurrency Control

108

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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 ...

CS 525



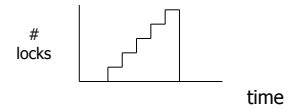
Notes 14 - Concurrency Control

109

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Sample Locking System:

- (1) Don't trust transactions to request/release locks
- (2) Hold all locks until transaction commits



CS 525



Notes 14 - Concurrency Control

110

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)**

CS 525



Notes 14 - Concurrency Control

111

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

All schedules (**ALL**)

Conflict Serializable (**CSR**)

2PL (**2PL**)

SS2PL (**SS2PL**)

Serial (**S**)

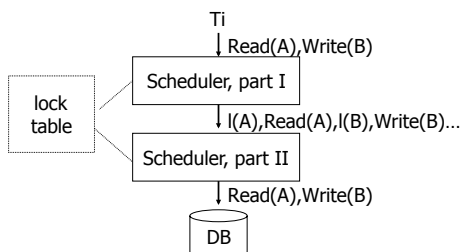
CS 525



Notes 13 - Failure and Recovery

112

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY



CS 525

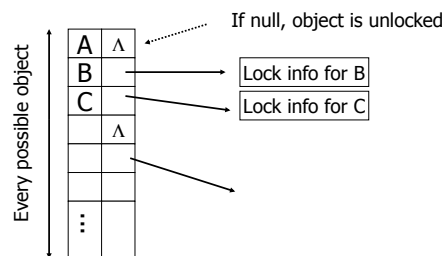


Notes 14 - Concurrency Control

113

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Lock table Conceptually



CS 525

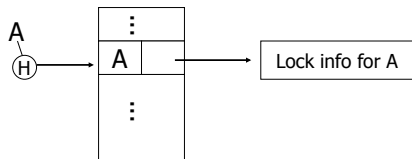


Notes 14 - Concurrency Control

114

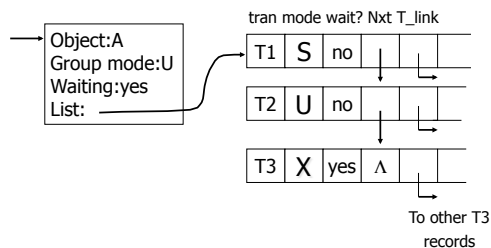
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

But use hash table:

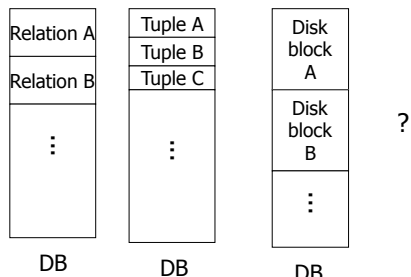


If object not found in hash table, it is unlocked

Lock info for A - example



What are the objects we lock?

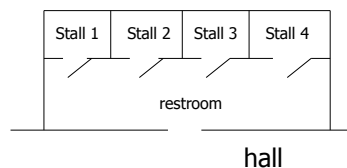


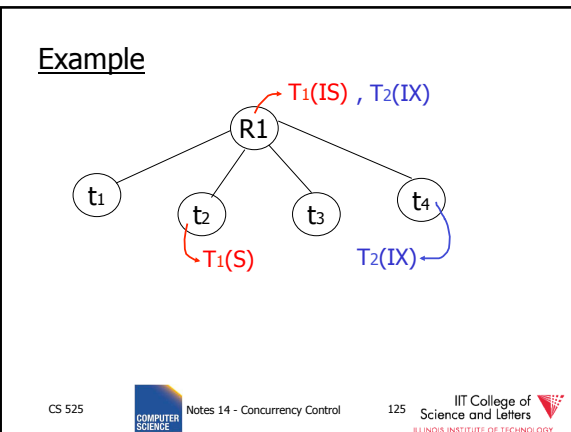
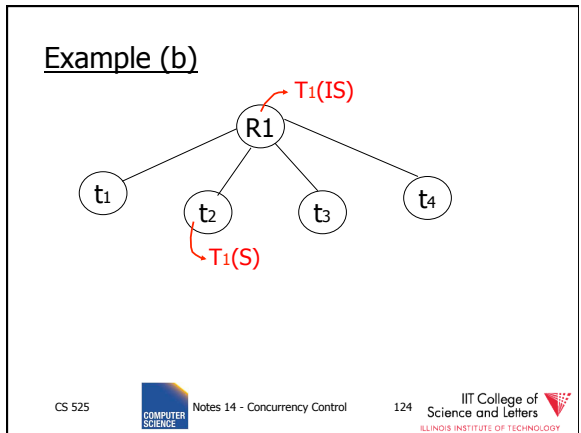
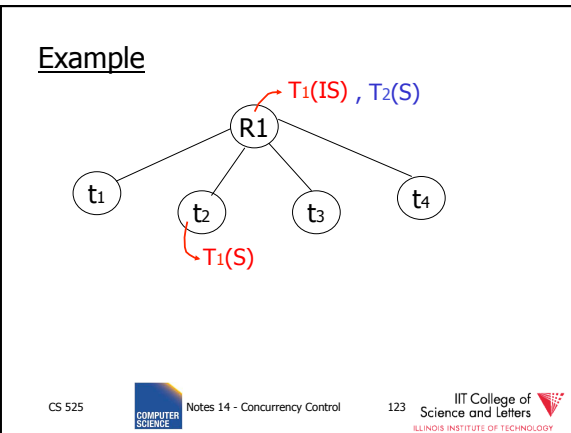
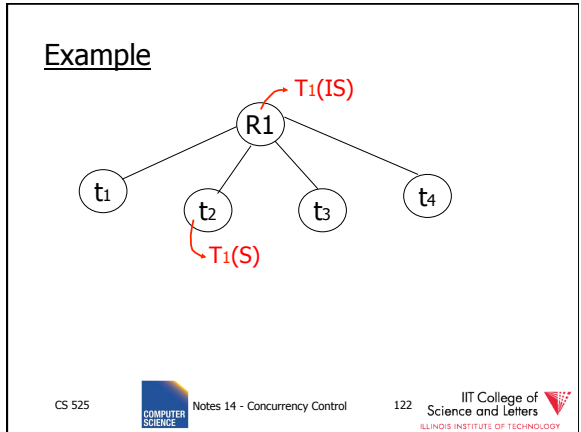
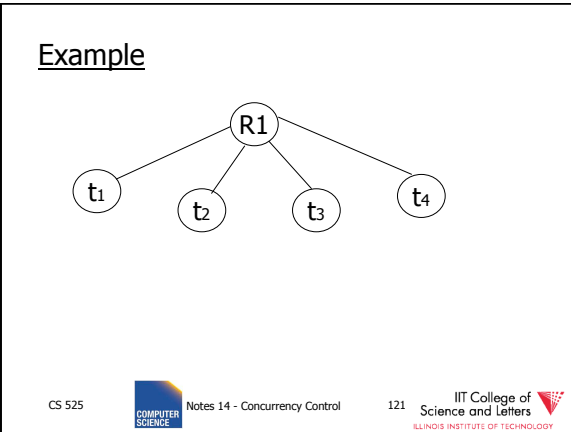
- Locking works in any case, but should we choose small or large objects?

- Locking works in any case, but should we choose small or large objects?
- If we lock large objects (e.g., Relations)
 - Need few locks
 - Low concurrency
- If we lock small objects (e.g., tuples, fields)
 - Need more locks
 - More concurrency

We can have it both ways!!

Ask any janitor to give you the solution...





Multiple granularity

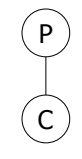
Comp	Requestor				
	IS	IX	S	SIX	X
Holder	IS				
	IX				
	S				
	SIX				
	X				

CS 525 Notes 14 - Concurrency Control 126 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

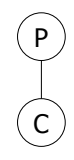
Multiple granularity

Comp		Requestor				
		IS	IX	S	SIX	X
Holder	IS	T	T	T	T	F
	IX	T	T	F	F	F
	S	T	F	T	F	F
	SIX	T	F	F	F	F
	X	F	F	F	F	F

Parent locked in	Child can be locked in
IS	
IX	
S	
SIX	
X	



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

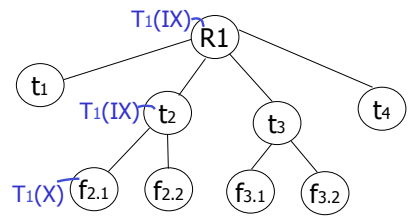


Rules

- (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

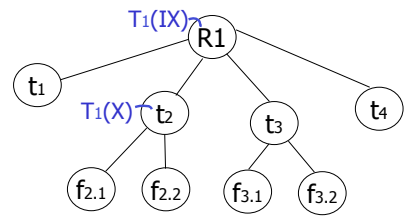
Exercise:

- Can T2 access object f2.2 in X mode? What locks will T2 get?



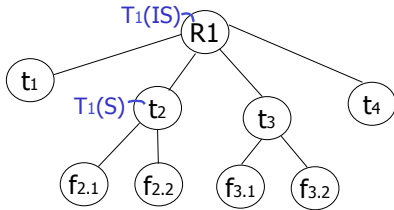
Exercise:

- Can T2 access object f2.2 in X mode? What locks will T2 get?



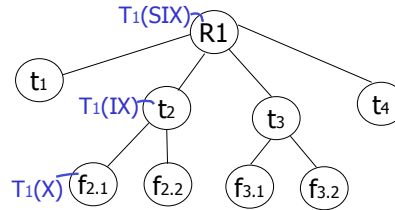
Exercise:

- Can T2 access object f3.1 in X mode?
What locks will T2 get?



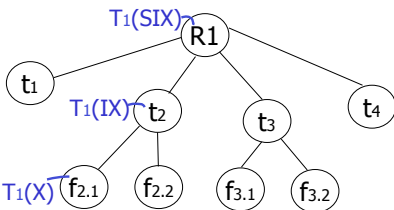
Exercise:

- Can T2 access object f2.2 in S mode?
What locks will T2 get?

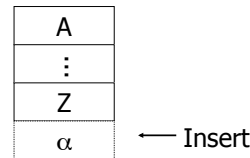


Exercise:

- Can T2 access object f2.2 in X mode?
What locks will T2 get?



Insert + delete operations



Modifications to locking rules:

- (1) Get exclusive lock on A before deleting A
- (2) At insert A operation by T_i, T_i is given exclusive lock on A


Still have a problem: **Phantoms**

Example: relation R (E#,name,...)
constraint: E# is key
use tuple locking

R	E#	Name	...
o1	55	Smith	
o2	75	Jones	

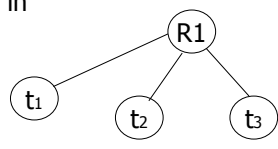
T1: Insert <08,Obama,...> into R
 T2: Insert <08,McCain,...> into R


T1	T2
S1(o1)	S2(o1)
S1(o2)	S2(o2)
Check Constraint	Check Constraint
⋮	⋮
Insert o3[08,Obama,..]	Insert o4[08,McCain,..]

CS 525  Notes 14 - Concurrency Control 139 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




CS 525  Notes 14 - Concurrency Control 140 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Back to example

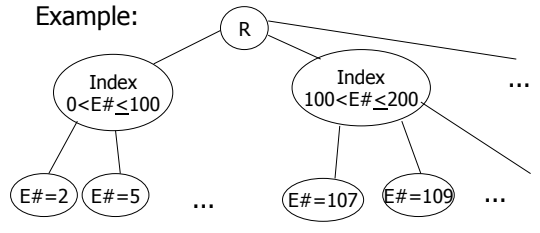
T1: Insert<04,Kerry> T2: Insert<04,Bush>


T1	T2
X1(R)	X2(R) <i>delayed</i>
Check constraint Insert<04,Kerry> U(R)	X2(R) Check constraint Oops! e# = 04 already in R!

CS 525  Notes 14 - Concurrency Control 141 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY


Instead of using R, can use index on R:

Example:




CS 525  Notes 14 - Concurrency Control 142 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

- This approach can be generalized to multiple indexes...

CS 525  Notes 14 - Concurrency Control 143 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Next:

- Tree-based concurrency control
- Validation concurrency control

CS 525  Notes 14 - Concurrency Control 144 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Example

- all objects accessed through root, following pointers

```

graph TD
    A((A)) --- B((B))
    A --- C((C))
    B --- D((D))
    D --- E((E))
    D --- F((F))
  
```

CS 525 Notes 14 - Concurrency Control 145 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Example

- all objects accessed through root, following pointers

CS 525 Notes 14 - Concurrency Control 146 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Example

- all objects accessed through root, following pointers

CS 525 Notes 14 - Concurrency Control 147 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Idea: traverse like "Monkey Bars"

CS 525 Notes 14 - Concurrency Control 148 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Idea: traverse like "Monkey Bars"

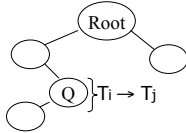
CS 525 Notes 14 - Concurrency Control 149 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

Idea: traverse like "Monkey Bars"

CS 525 Notes 14 - Concurrency Control 150 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525



Notes 14 - Concurrency Control

151

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 $\text{parent}(Q)$ locked by T_i
- (3) Items may be unlocked at any time
- (4) After T_i unlocks Q , it cannot relock Q

CS 525

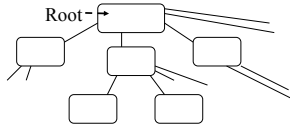


Notes 14 - Concurrency Control

152

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

- Tree-like protocols are used typically for B-tree concurrency control



E.g., during insert, do not release parent lock, until you are certain child does not have to split

CS 525



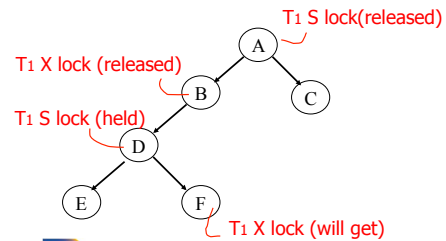
Notes 14 - Concurrency Control

153

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Tree Protocol with Shared Locks

- Rules for shared & exclusive locks?



CS 525



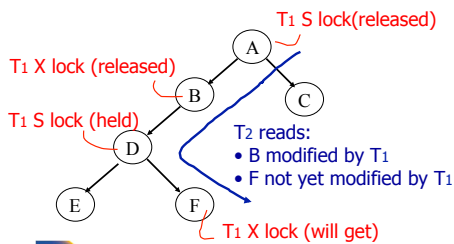
Notes 14 - Concurrency Control

154

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Tree Protocol with Shared Locks

- Rules for shared & exclusive locks?



CS 525



Notes 14 - Concurrency Control

155

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

CS 525



Notes 14 - Concurrency Control

156

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

CS 525



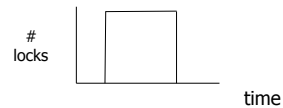
Notes 14 - Concurrency Control

157

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deadlock Prevention

- Option 1:
 - 2PL + transaction has to acquire all locks at transaction start following a global order



CS 525



Notes 14 - Concurrency Control

158

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deadlock Prevention

- Option 1:
 - Long log 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!

CS 525



Notes 14 - Concurrency Control

159

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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)

CS 525



Notes 14 - Concurrency Control

160

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 ☹

CS 525



Notes 14 - Concurrency Control

161

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

CS 525



Notes 14 - Concurrency Control

162

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 $T_i < T_j \rightarrow$ roll-back T_j
 - Timestamp $T_i > T_j \rightarrow$ wait

CS 525
Notes 14 - Concurrency Control
163
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Deadlock Prevention

- Option 3:
 - Additional transaction roll-backs ☹️

CS 525
Notes 14 - Concurrency Control
164
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Timeout-based Scheme

- Option 4:
 - After waiting for a lock longer than X, a transaction is rolled back

CS 525
Notes 14 - Concurrency Control
165
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

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

CS 525
Notes 14 - Concurrency Control
166
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 \rightarrow Deadlock
 - Abort one of the transaction in cycle to resolve deadlock

CS 525
Notes 14 - Concurrency Control
167
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?

```

    graph TD
      T1((T1)) --> T2((T2))
      T2 --> T3((T3))
      T3 --> T4((T4))
      T4 --> T5((T5))
      T5 --> T1
    
```

CS 525
Notes 14 - Concurrency Control
168
IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Optimistic Concurrency Control:

Validation

Transactions have 3 phases:

(1) Read

- all DB values read
- writes to temporary storage
- no locking

(2) Validate

- check if schedule so far is serializable

(3) Write

- if validate ok, write to DB

CS 525



Notes 14 - Concurrency Control

169

Key idea

- Make validation atomic
- If T_1, T_2, T_3, \dots is validation order, then resulting schedule will be conflict equivalent to $S_s = T_1 T_2 T_3 \dots$

CS 525



Notes 14 - Concurrency Control

170

To implement validation, system keeps two sets:

- FIN = transactions that have finished phase 3 (and are all done)
- VAL = transactions that have successfully finished phase 2 (validation)

CS 525



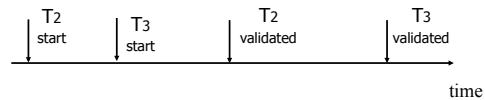
Notes 14 - Concurrency Control

171

Example of what validation must prevent:

$$RS(T_2) = \{B\} \quad \cap \quad RS(T_3) = \{A, B\} \neq \phi$$

$$WS(T_2) = \{B, D\} \quad \cap \quad WS(T_3) = \{C\}$$



CS 525



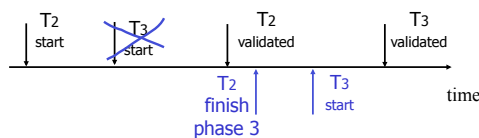
Notes 14 - Concurrency Control

172

Example of what validation must allow prevent:

$$RS(T_2) = \{B\} \quad \cap \quad RS(T_3) = \{A, B\} \neq \phi$$

$$WS(T_2) = \{B, D\} \quad \cap \quad WS(T_3) = \{C\}$$



CS 525



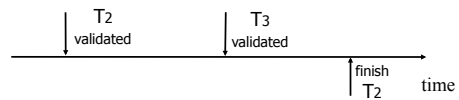
Notes 14 - Concurrency Control

173

Another thing validation must prevent:

$$RS(T_2) = \{A\} \quad \cap \quad RS(T_3) = \{A, B\}$$

$$WS(T_2) = \{D, E\} \quad \cap \quad WS(T_3) = \{C, D\}$$



CS 525

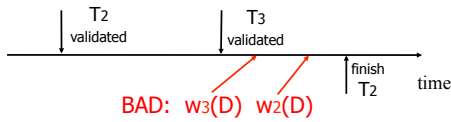


Notes 14 - Concurrency Control

174

Another thing validation must prevent:

$RS(T_2) = \{A\}$ $RS(T_3) = \{A, B\}$
 $WS(T_2) = \{D, E\}$ $WS(T_3) = \{C, D\}$



CS 525



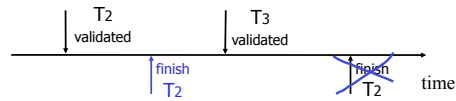
Notes 14 - Concurrency Control

175

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Another thing validation must prevent:

$RS(T_2) = \{A\}$ $RS(T_3) = \{A, B\}$
 $WS(T_2) = \{D, E\}$ $WS(T_3) = \{C, D\}$



CS 525



Notes 14 - Concurrency Control

176

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Validation rules for T_j :

- (1) When T_j starts phase 1:
 $ignore(T_j) \leftarrow FIN$
- (2) at T_j Validation:
 if check (T_j) then
 [$VAL \leftarrow VAL \cup \{T_j\}$;
 do write phase;
 $FIN \leftarrow FIN \cup \{T_j\}$]

CS 525



Notes 14 - Concurrency Control

177

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Check (T_j):

For $T_i \in VAL - IGNORE(T_j)$ DO
 IF [$WS(T_i) \cap RS(T_j) \neq \emptyset$ OR
 $T_i \notin FIN$] THEN RETURN false;
 RETURN true;

CS 525



Notes 14 - Concurrency Control

178

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Check (T_j):

For $T_i \in VAL - IGNORE(T_j)$ DO
 IF [$WS(T_i) \cap RS(T_j) \neq \emptyset$ OR
 $T_i \notin FIN$] THEN RETURN false;
 RETURN true;

Is this check too restrictive ?

CS 525



Notes 14 - Concurrency Control

179

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

Improving Check(T_j)

For $T_i \in VAL - IGNORE(T_j)$ DO
 IF [$WS(T_i) \cap RS(T_j) \neq \emptyset$ OR
 ($T_i \notin FIN$ AND $WS(T_i) \cap WS(T_j) \neq \emptyset$)]
 THEN RETURN false;
 RETURN true;

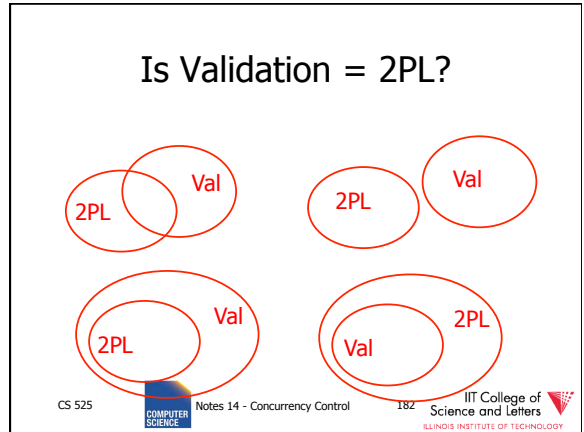
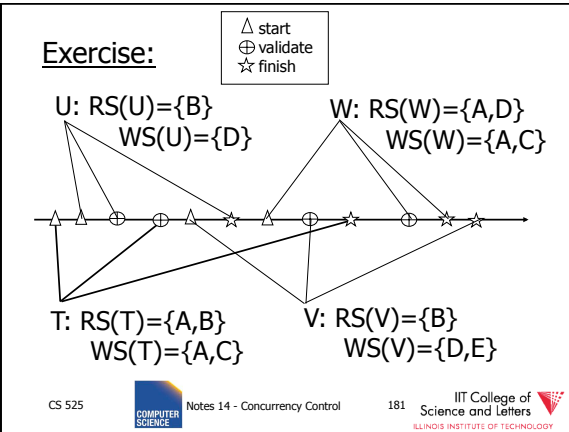
CS 525



Notes 14 - Concurrency Control

180

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY



S2: w2(y) w1(x) w2(x)

- S2 can be achieved with 2PL:
 $l_2(y) w_2(y) l_1(x) w_1(x) u_1(x) l_2(x) w_2(x) u_2(y) u_2(x)$
- 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.

CS 525 Notes 14 - Concurrency Control 183 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

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)

CS 525 Notes 14 - Concurrency Control 184 IIT College of Science and Letters ILLINOIS INSTITUTE OF TECHNOLOGY

- Say S' not legal:
 $S' : \dots l_1(x) w_2(x) r_1(x) val_1 u_2(x) \dots$
 - At val1: T2 not in Ignore(T1); T2 in VAL
 - T1 does not validate: $WS(T_2) \cap RS(T_1) \neq \emptyset$
 - contradiction!
- Say S' not legal:
 $S' : \dots val_1 l_1(x) w_2(x) w_1(x) u_2(x) \dots$
 - Say T2 validates first (proof similar in other case)
 - At val1: T2 not in Ignore(T1); T2 in VAL
 - T1 does not validate:
 $T_2 \notin FIN \text{ AND } WS(T_1) \cap WS(T_2) \neq \emptyset$
 - contradiction!

CS 525 Notes 14 - Concurrency Control 185 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

CS 525 Notes 14 - Concurrency Control 186 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)

CS 525



Notes 14 - Concurrency Control

187

IIT College of
Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY