

CMU SCS

Carnegie Mellon Univ.
Dept. of Computer Science
15-415 - Database Applications

Lecture #22PLUS: Concurrency Control
Part 2 (R&G ch. 17)

Faloutsos SCS 15-415 #1

CMU SCS

Outline

- Q1: Which order to release locks in multiple-granularity locking?
 - A1: bottom up
- Q2: Which order to release locks in tree-locking?
 - A2: top down (to max concurrency)
- Details on tree locking, for hw10

Faloutsos SCS 15-415 #2

CMU SCS

Multiple granularity

- Eg:

```
graph TD; DB((DB)) --> Table1((Table1)); DB --> Table2((Table2)); Table1 --> record1((record1)); Table1 --> record2((record2)); Table1 --> record_n((record-n)); record1 --> attr1_1((attr1)); record1 --> attr2((attr2)); record2 --> attr1_2((attr1));
```

Faloutsos SCS 15-415 #3

CMU SCS

Multiple Granularity Lock Protocol

- Each Xact: lock root.
- To get S or IS lock on a node, must hold **at least** IS on parent node.
 - What if Xact holds SIX on parent? S on parent?
- To get X or IX or SIX on a node, must hold **at least** IX on parent node.
- Must release locks in bottom-up order.

Faloutsos SCS 15-415 **Originally, page 18**

CMU SCS

Outline

- Q1: Which order to release locks in multiple-granularity locking?
 - A1: bottom up
- Q2: Which order to release locks in tree-locking?
 - A2: as early as possible (to max concurrency)
- Details on tree locking

Faloutsos SCS 15-415 #5

CMU SCS

A Simple Tree Locking Algorithm: “crabbing”

- **Search:** Start at root and go down; repeatedly,
 - S lock child
 - then unlock parent
- **Insert/Delete:** Start at root and go down, obtaining X locks as needed. Once child is locked, check if it is **safe**:
 - If child is safe, release all locks on ancestors.

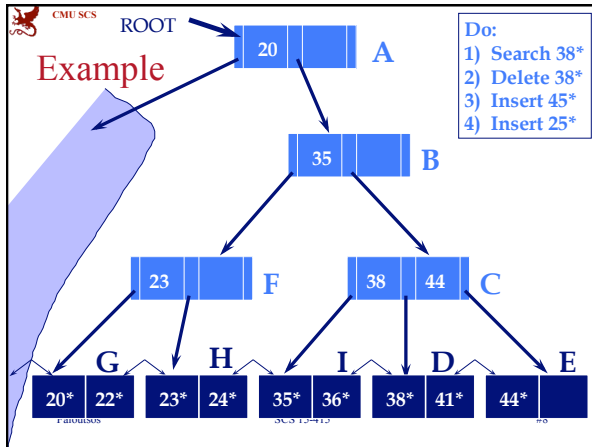
Faloutsos SCS 15-415 **Original page 36**

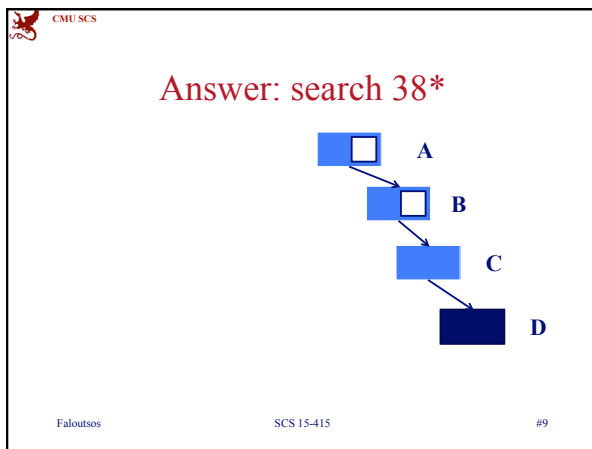
CMU SCS

Use for hw10

- The algo before (original page #36)
 - No ‘gambling’ of [Bayer, Schkolnick]
 - No lock upgrades and no deadlocks
 - Ignore the update of sibling pointers
- Next examples: repeats of examples in original lecture, with more details

Faloutsos SCS 15-415 #7





CMU SCS

Answer: search 38*

SA
SB
UA
SC
UB
SD
UC
<read 38*>
UD

Faloutsos SCS 15-415 #10

CMU SCS

Answer: delete 38*

← max concurrency

Faloutsos SCS 15-415 #11

CMU SCS

Answer: delete 38*

XA
XB
XC
UA ← max concurrency
UB
XD
UC
<delete 38*>
UD

Faloutsos SCS 15-415 #12

CMU SCS

Answer: insert 45*

Faloutsos SCS 15-415 #13

CMU SCS

Answer: insert 45*

X A
X B
 U A
X C
X E
 U B
 U C
<insert 45* >
 U E

Faloutsos SCS 15-415 #14

CMU SCS

Answer: insert 25*

Faloutsos SCS 15-415 #15

CMU SCS

Answer: insert 25*

X A
 X B U A
 X F U B
 X H
 <insert 25*>
 <split H>
 <update F>
 U F
 U H

Faloutsos SCS 15-415 #16

CMU SCS

Answer: insert 25*

X A
 X B U A
 X F U B
 X H
 <insert 25*>
 <split H>
 <update F>
 U H
 U F

Q: Why not swap?

Faloutsos SCS 15-415 #17

CMU SCS

Answer: insert 25*

X A
 X B U A
 X F U B
 X H
 <insert 25*>
 <split H>
 <update F>
 U H
 U F

Q: Why not swap?
A: swapping does not help concurrency!

Faloutsos
