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

Multiple granularity

• Eg:
Multiple Granularity Lock Protocol

- Each Xact: lock root.
- To get S or IS lock on a node, must hold \textbf{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 \textbf{at least} IX on parent node.
- Must release locks in bottom-up order.

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

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 \textbf{safe}:
  - If child is safe, release all locks on ancestors.
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

Example

```
ROOT

Example

Do:
1) Search 38*
2) Delete 38*
3) Insert 45*
4) Insert 25*
```

Answer: search 38*
Answer: search 38*

Answer: delete 38*

Answer: delete 38*
Answer: insert 45*

Answer: insert 45*

Answer: insert 25*
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

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!