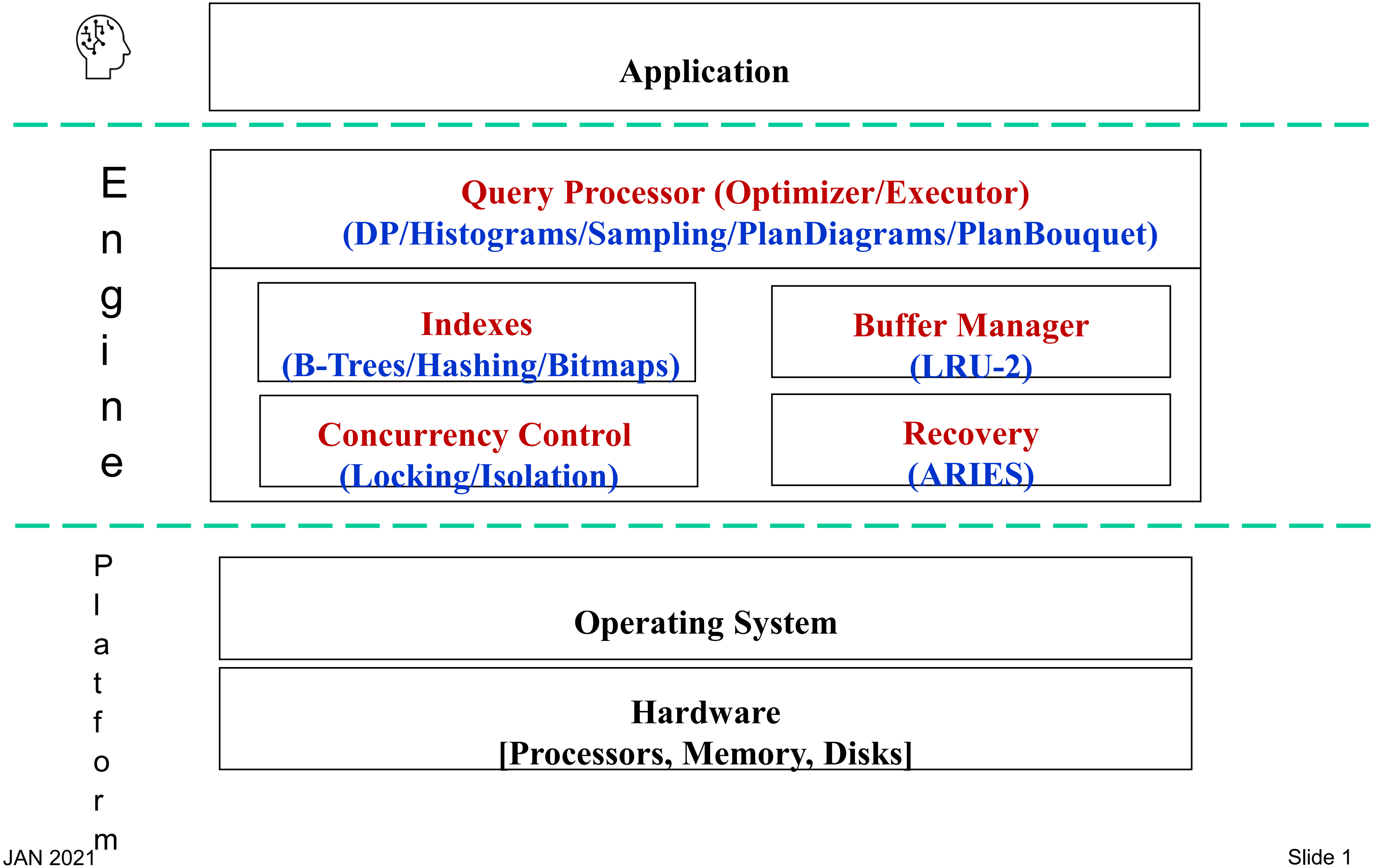
RDBMS Architecture



Application

Engine

Query Processor (Optimizer/Executor)
(DP/Histograms/Sampling/PlanDiagrams/PlanBouquet)

Indexes
(B-Trees/Hashing/Bitmaps)

Buffer Manager
(LRU-2)

Concurrency Control
(Locking/Isolation)

Recovery
(ARIES)

Platform

Operating System

Hardware
[Processors, Memory, Disks]

Concurrency Control

E0 261

Prasad Deshpande, Jayant Haritsa

Computer Science and Automation

Indian Institute of Science

†Slides adapted from ChengXiang Zhai and Hideaki Kimura

ACID Properties of Transactions

- Atomicity:
  - all or nothing at all
- Consistency
  - from one consistent state to another
- Isolation
  - as if executed alone
- Durability
  - results permanent after commit

Today's Paper

- Granularity of Locks and Degrees of Consistency in a Shared Data Base
  - J. N. Gray, R. A. Lorie, G. R. Putzolu and I. L. Traiger
- Modeling in DBMS, 1976
- Turing award in 1998 for Jim Gray

Motivation: A “Simple” Protocol

- Lock modes:
  - S for shared and X for exclusive access
  - compatibility: (S, S) = T, otherwise F
- Unit:
  - a relation
- Behavior:
  - Two-phase locking for read locks
  - Strict two-phase locking for write locks

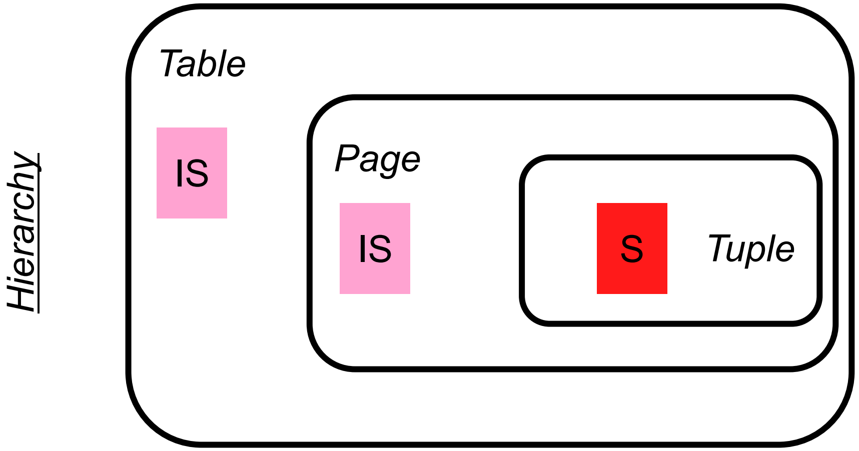
Locking Granularity

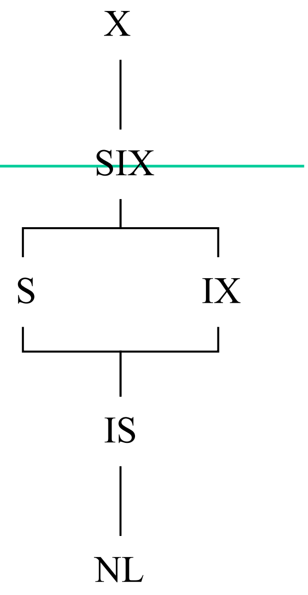
- Fine-grained units for increasing concurrency
- Coarse-grained units for decreasing overheads
- Solution:
  - hierarchical lockable units:
    - DB, areas, files, pages, tuples, attributes
- Correctness problem:
  - T1 S.locks a tuple, T2 X.locks the file?
  - solution?
    - create a collision path

Locking Protocol

- Database: as hierarchy of lockable units
- Locking: to lock a unit
  - first lock all *containing* units with “intention”
  - intention locks: IS, IX (intention to upgrade)
  - no implicit locking of sub-tree unlike S and X
  - SIX: implicit locking in S and explicit in X mode
- Unlocking:
  - release all relevant locks simultaneously at EOT, or leaf to root

Intention Locks (IS/IX/SIX)

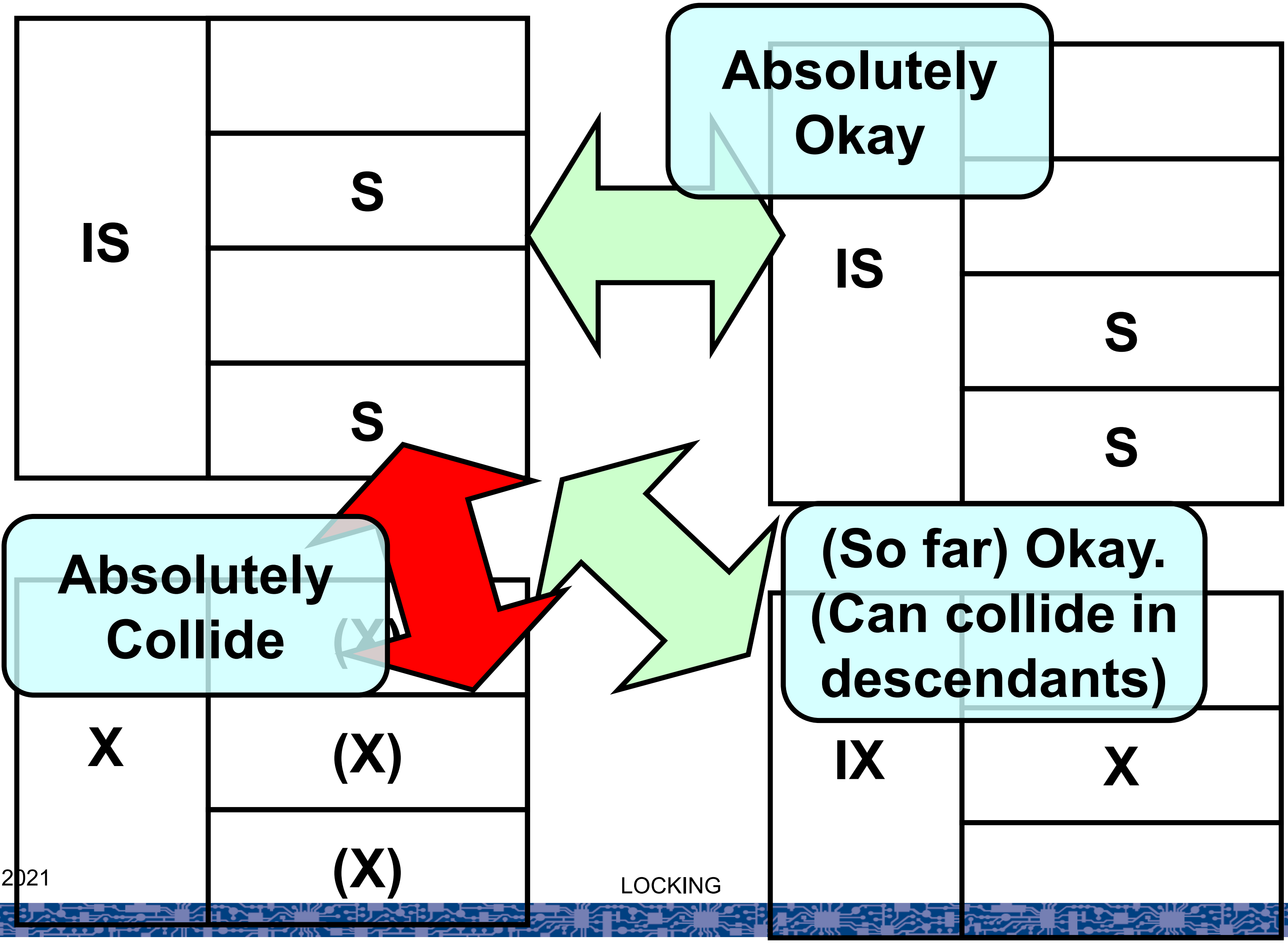




	S	X
S	Y	N
X	N	N

	IS	S	IX	SIX	X
IS	Y	Y	Y	Y	N
S	Y	Y	N	N	N
IX	Y	N	Y	N	N
SIX	Y	N	N	N	N
X	N	N	N	N	N

Intention Lock Compatibility (IS)



IS

S

S

Absolutely Okay

IS

S

S

Absolutely Collide

X

(X)

(X)

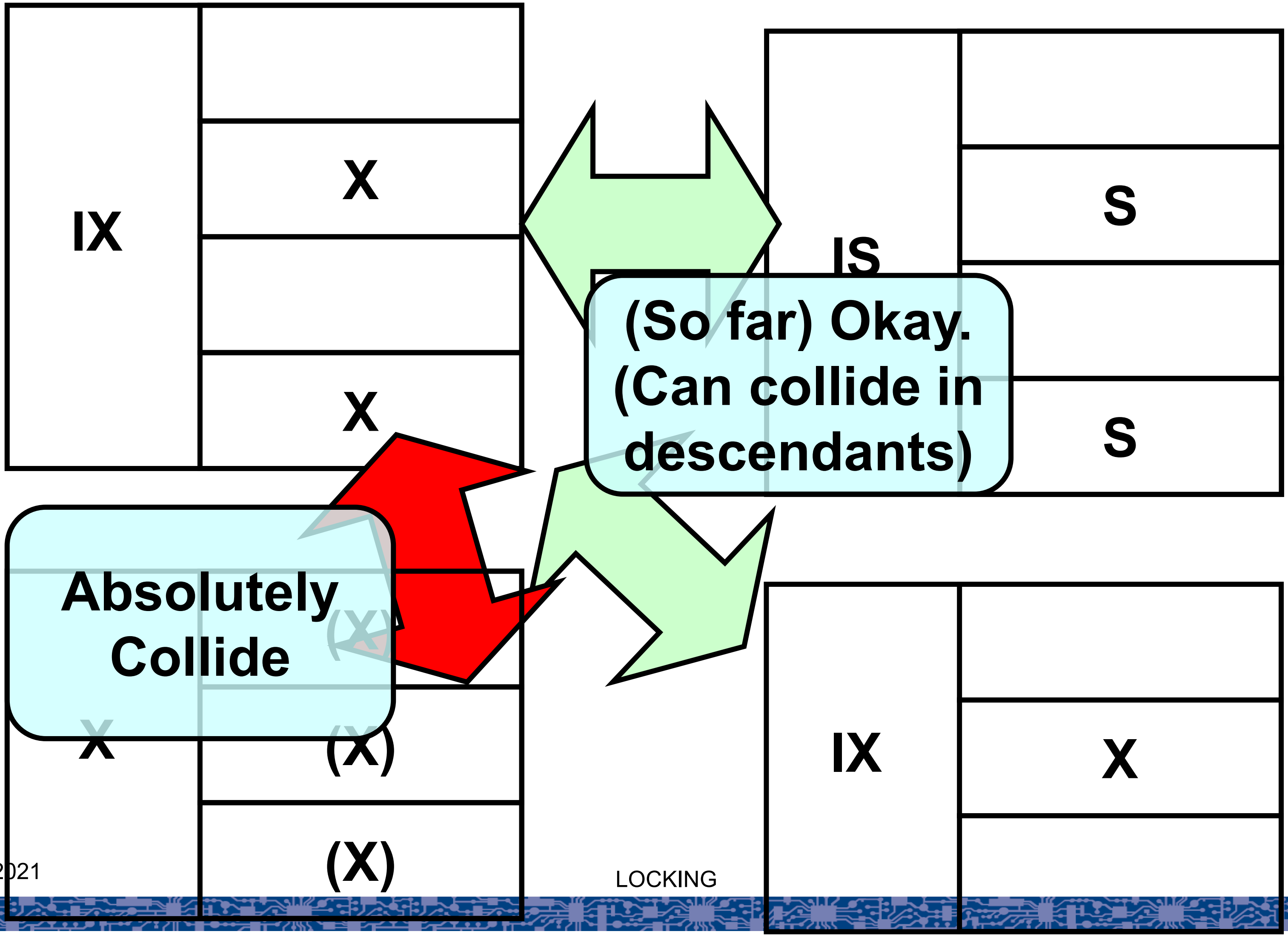
(X)

(So far) Okay. (Can collide in descendants)

IX

X

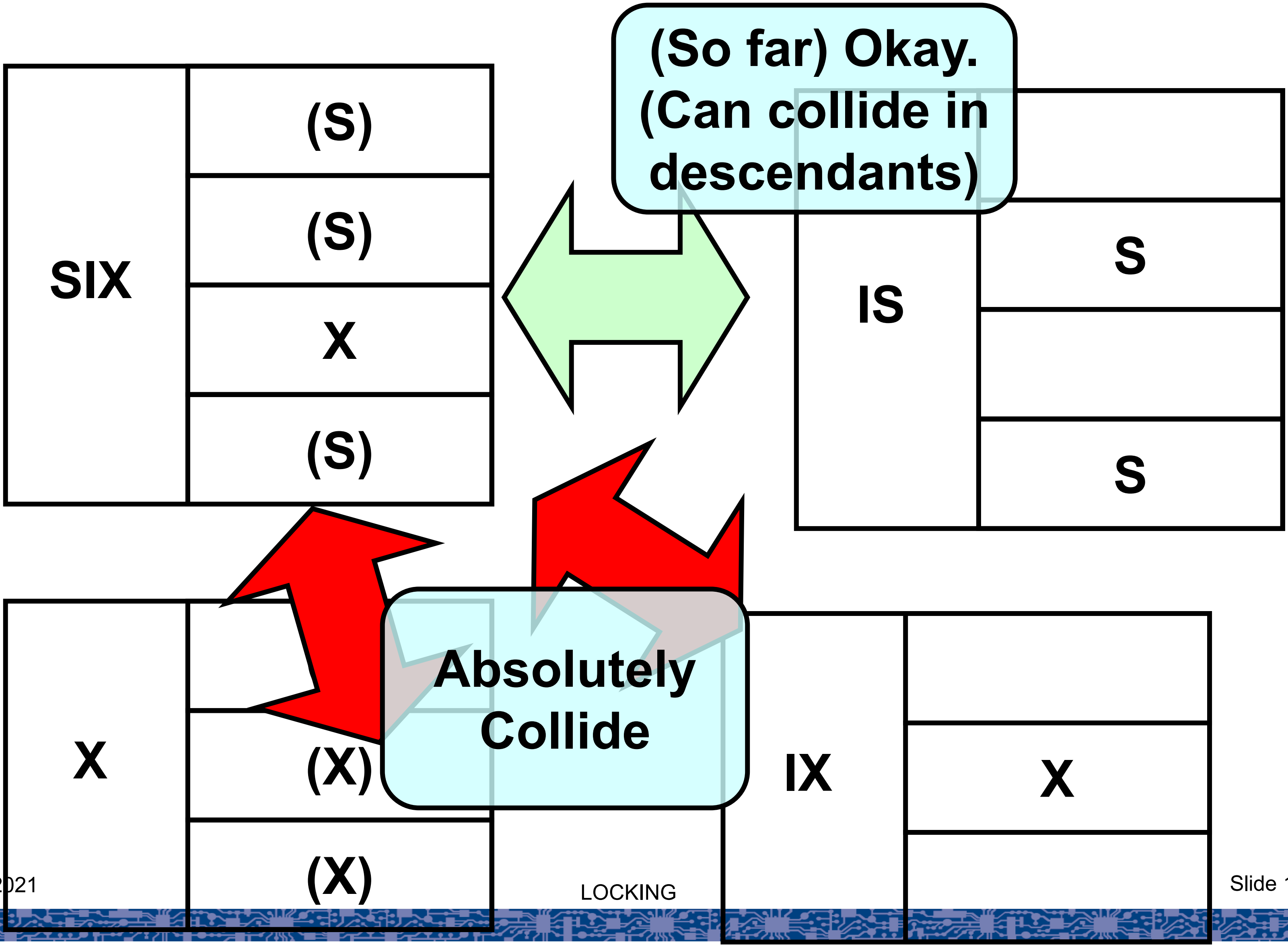
Intention Lock Compatibility (IX)



(So far) Okay. (Can collide in descendants)

Absolutely Collide

Intention Lock Compatibility (SIX)



SIX
(S)
(S)
X
(S)

(So far) Okay.
(Can collide in descendants)

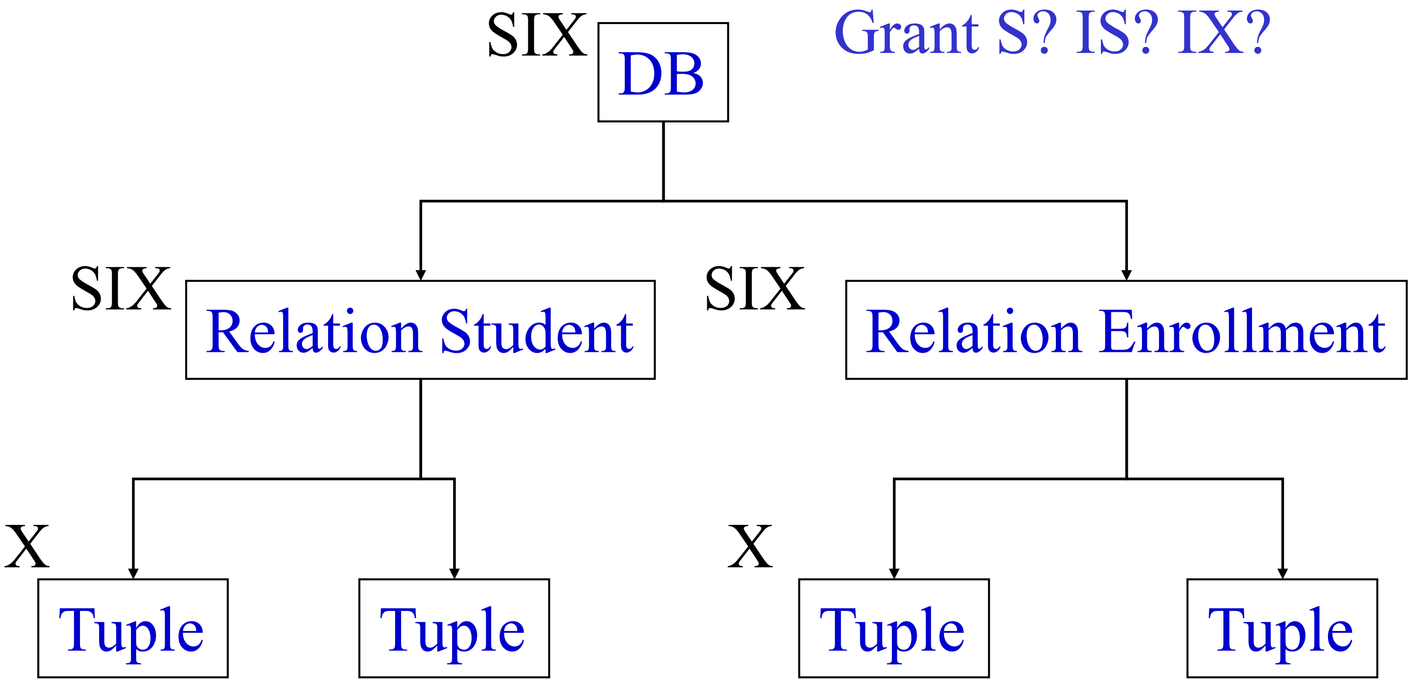
IS
S
S

X
(X)
(X)

Absolutely Collide

IX
X

Compatibility Examples



Grant S? IS? IX?

- (SIX, S) = No (SIX, IS) = Yes (SIX, IX) = No

Locking Protocol: DAG

- DAG of units:
  - – S lock at least one path to the node
  - – X lock all paths to the node

- Implicit S if one parent is S, SIX, X
- Implicit X if all parents are X

Additional Lock Modes

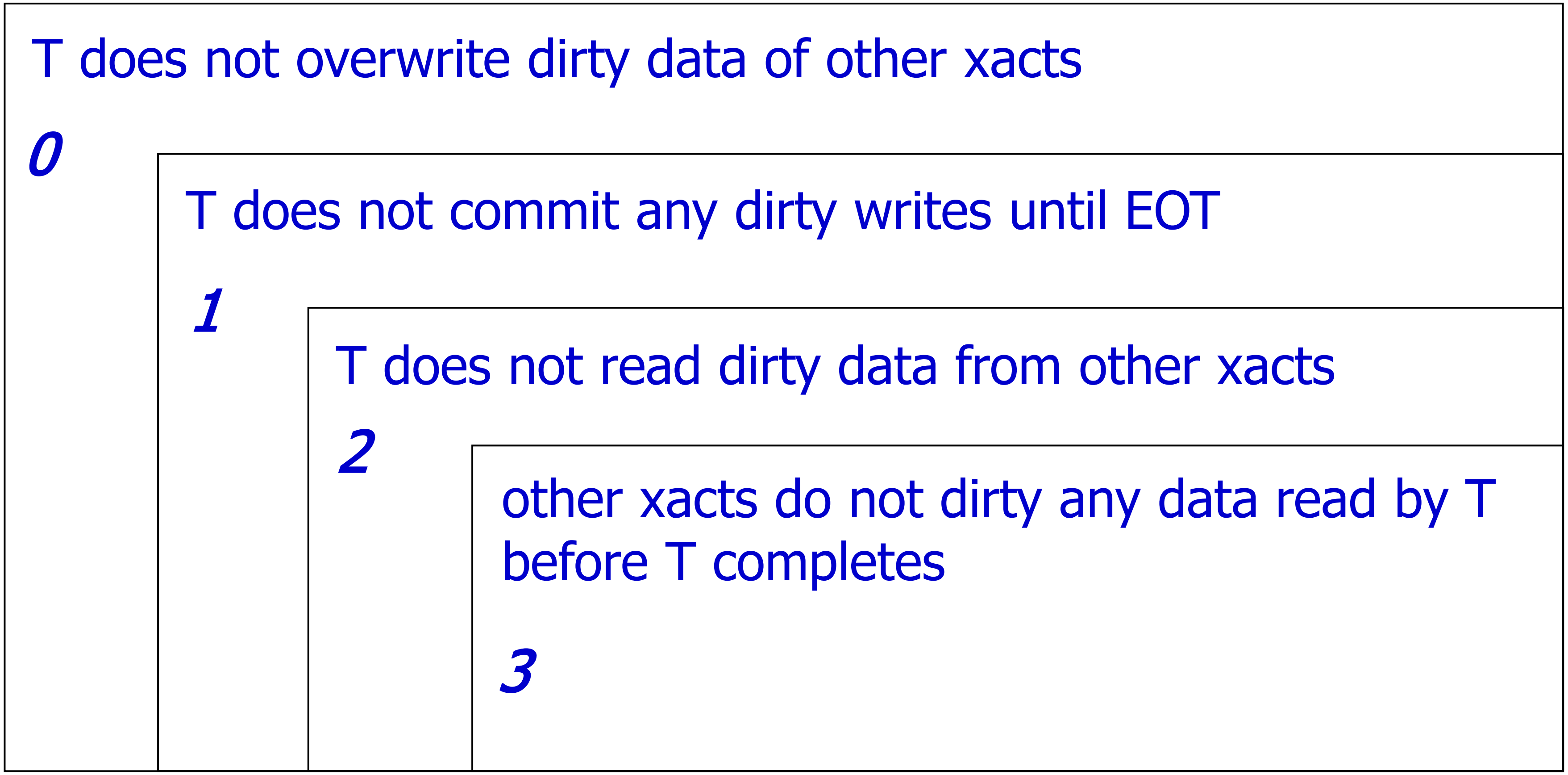
- Based on semantics of updates
  - increment / decrement locks (as in assignment!)

Transaction-level Consistency

Motivation for Consistency Degrees

- Garbage values
  - T1: A = 10; w(A)
  - T2: A = 25; w(A)
  - **Problem:** Final value may be neither 10 nor 25
  - **Solution: Short write locks (latches), Degree 0 consistency**
- Lost updates
  - T1: r(A) w(A) commit
  - T2: w(A) commit
  - **Problem:** T2's update is lost
  - **Solution: Long write locks, Degree 1 consistency**
- Dirty Reads:
  - T1: w(A) abort
  - T2: r(A)
  - **Problem:** T2 has read a non-existent value
  - **Solution: Long write locks, Short Read locks, Degree 2 consistency**
- Inconsistent Reads
  - T1: w(A) commit
  - T2: r(A) r(A)
  - Problem: Different committed values read by T2 for same data object
  - **Solution: Long write locks, Long read locks, Degree 3 consistency**

Consistency: Dirty-Data Based



T does not overwrite dirty data of other xacts

0

T does not commit any dirty writes until EOT

1

T does not read dirty data from other xacts

2

other xacts do not dirty any data read by T before T completes

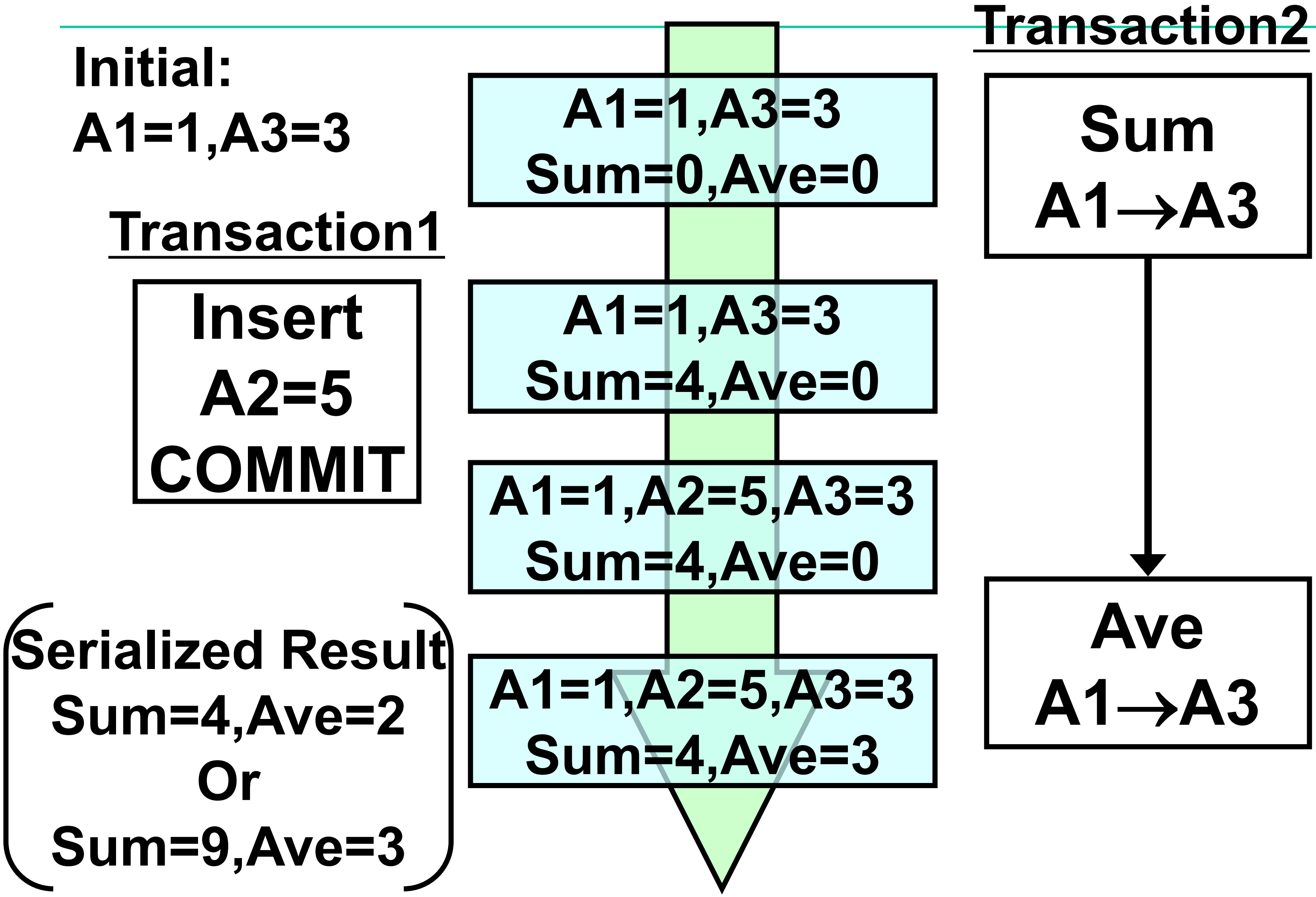
3

- How to lock for each degree?

Consistency: Locking-Based

- Corresponding to each condition:
  - T does not overwrite dirty data of other xacts
    - set write locks on dirty data (well-formed on w)
  - T does not commit any dirty writes until EOT
    - set "long" write locks (2P/EOT on w)
  - T does not read dirty data from other xacts
    - set read locks (well-formed on r)
  - other xacts do not dirty any data read by T before T completes
    - set "long" read locks (2P/EOT on r)

Phantom Tuples



Initial:
A1=1,A3=3

Transaction1

Insert
A2=5
COMMIT

Serialized Result
Sum=4,Ave=2
Or
Sum=9,Ave=3

A1=1,A3=3
Sum=0,Ave=0

A1=1,A3=3
Sum=4,Ave=0

A1=1,A2=5,A3=3
Sum=4,Ave=0

A1=1,A2=5,A3=3
Sum=4,Ave=3

Transaction2

Sum
A1→A3

Ave
A1→A3

Locks in Fully Serializable Mode

- Same Locks as Repeatable Read
- .. and Special Locks to Prevent Phantoms
  - Predicate Lock (Semantic Lock)
    - Efficiently checkable for simple predicates
      - Boolean combination of Attr op const where op is <, =, ≠, >
      - Bank_branch = “SBI IISc Campus”
    - NP-hard for arbitrary predicates
  - Key Range Lock
    - Select avg(gpa) from Student where Age between 20 and 30
    - Lock index keys between 20 and 30
  - Worst case, lock whole table

Additional Issues

- Deadlock Handling
  - Simplest solution is timeout!
  - Alternatively, cycle checking

- Deadlock Prevention
  - Use other CC methods such as Timestamping and Validation

CC Support in Practice

IBM DB2:

- Optional explicit locking (by user) at table level
  - LOCK TABLE students in SHARE/EXCLUSIVE MODE
  - locks held until and automatically released at EOT
- Automatic implicit locking (by system) with “isolation level” setting

  SQL transaction isolation levels:
  - Serializable
    - set long write / long read / phantom locks
  - repeated read
    - set long write /long read locks, thus phantom insert possible
  - read committed
    - set long write/short read locks
  - read uncommitted
    - set long write/no read locks

  (somewhat different names in DB2)
- For more information, see Jim Gray’s well-known book “Transaction Processing”

END Concurrency Control