UNIVERSITY OF WISCONSIN–MADISON

CS 764: Topics in Database Management Systems

Lecture 6: Granularity of Locks

Xiangyao Yu

9/23/2020

Discussion Highlights

```
SELECT      JOB.title, count(*)
FROM        JOB, EMP, DEPT
WHERE       JOB.jid = EMP.jid
AND         EMP.did = DEPT.did
AND         DEPT.location="Madison"
GROUP BY    JOB.title
```

|EMP| = 10000 tuples
|DEPT| = 100 tuples
|JOB| = 10 tuples

** assuming one-on-one mapping between jid and title*

Consider only nested loop join and only the cost in terms of the **# comparisons** in the join (note that which relation is inner vs. outer in a join does not matter in this case)

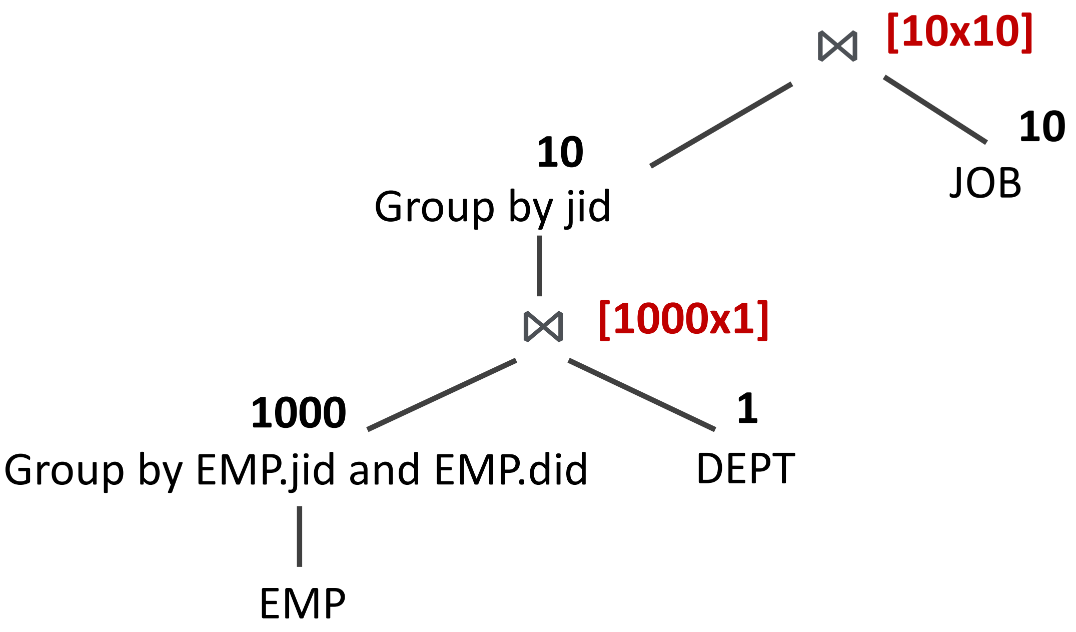
Q1: If only one department is in Madison, what's the cheapest plan?
(hint: group-by can be partially pushed down)
Q2 [optional]: If all departments are in Madison, what's the cheapest plan?

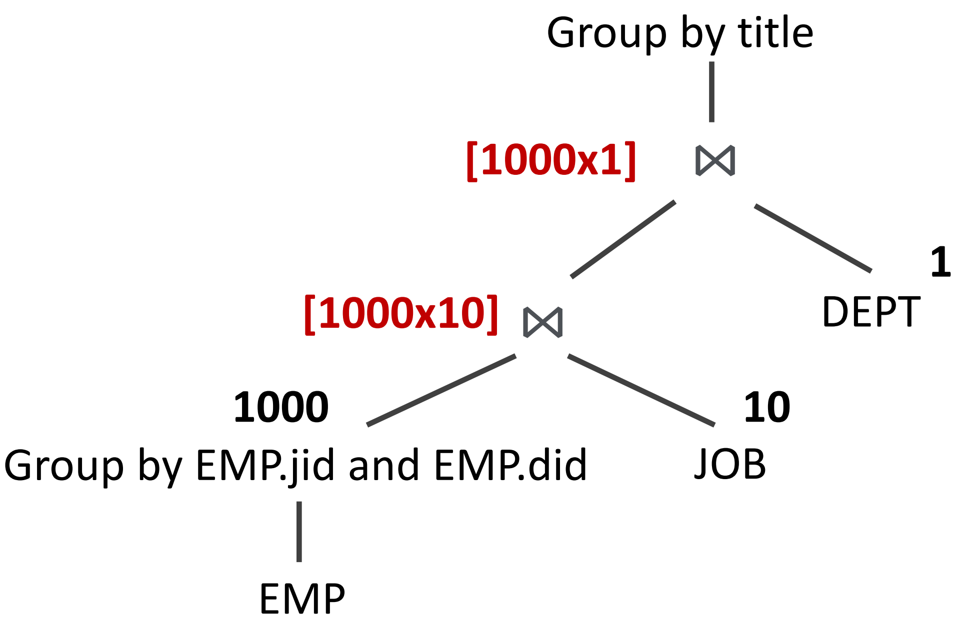
Discussion Highlights – One Dept. in Madison

```
SELECT      JOB.title, count(*)
FROM        JOB, EMP, DEPT
WHERE       JOB.jid = EMP.jid
AND         EMP.did = DEPT.did
AND         DEPT.location=“Madison”
GROUP BY    JOB.title
```

|EMP| = 10000 tuples
|DEPT| = 100 tuples
|JOB| = 10 tuples

** assuming one-on-one mapping between jid and title*



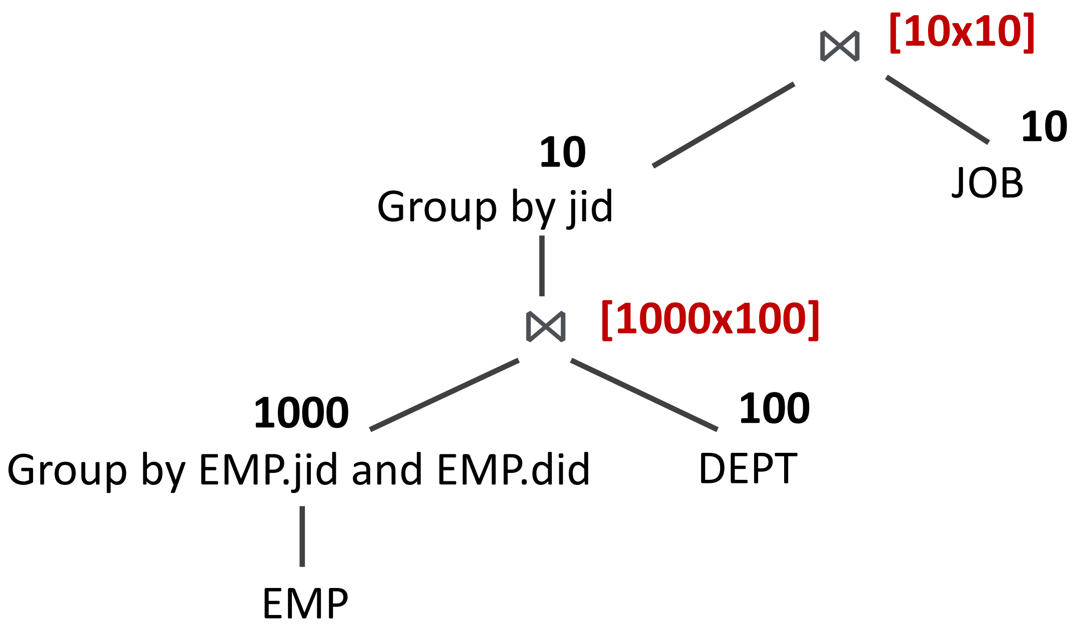


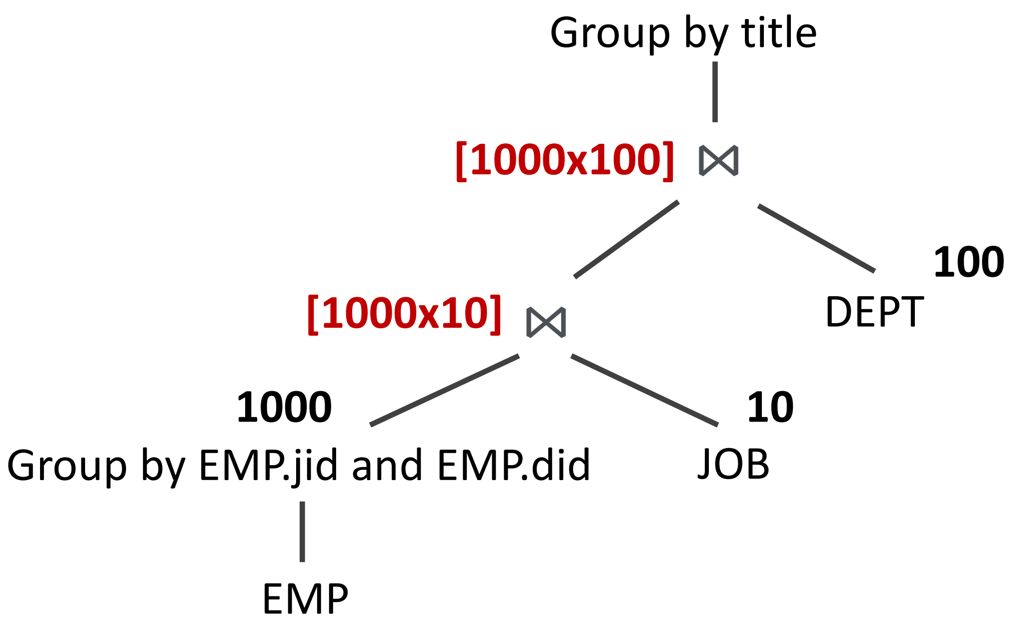
Discussion Highlights – All Dept. in Madison

```
SELECT     JOB.title, count(*)
FROM       JOB, EMP, DEPT
WHERE      JOB.jid = EMP.jid
AND        EMP.did = DEPT.did
AND        DEPT.location="Madison"
GROUP BY   JOB.title
```

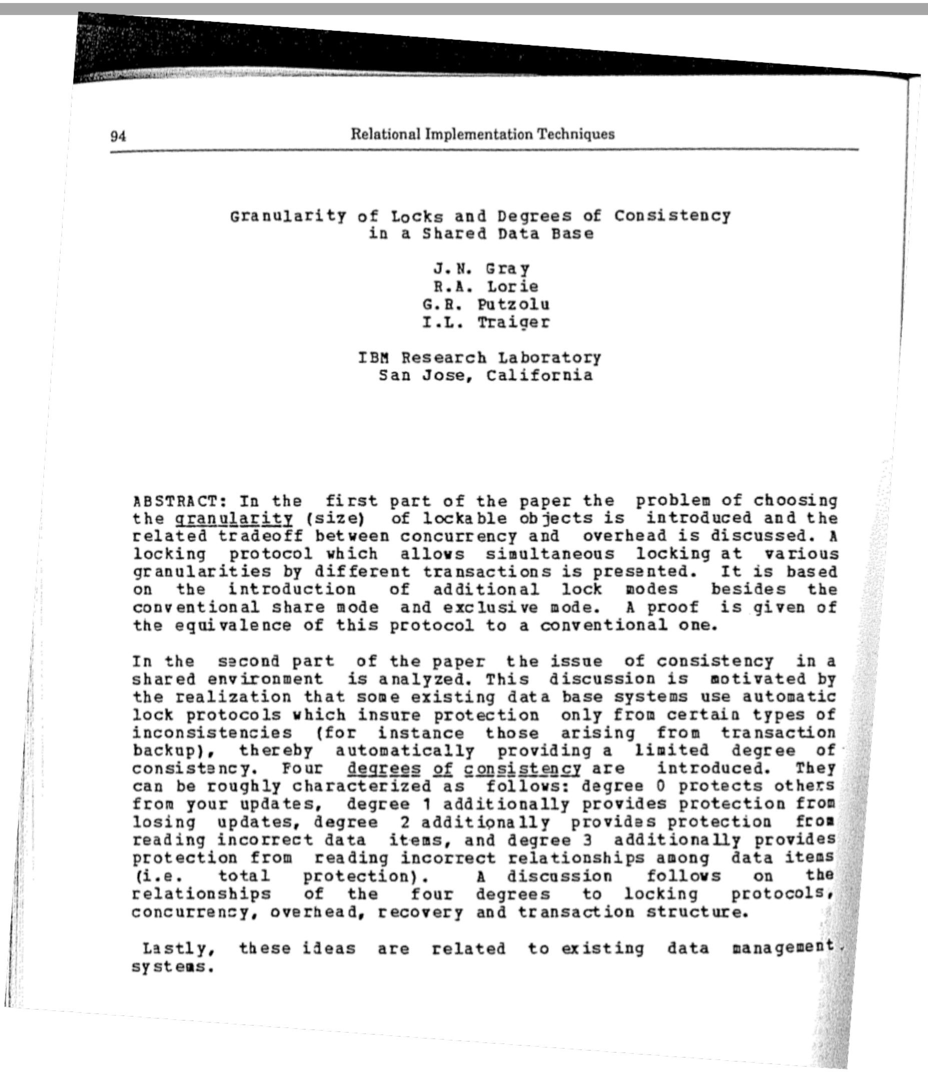
|EMP| = 10000 tuples
|DEPT| = 100 tuples
|JOB| = 10 tuples

** assuming one-on-one mapping between jid and title*





Today's Paper: Granularity of Locks



Granularity of Locks and Degrees of Consistency in a Shared Data Base

J.N. Gray
R.A. Lorie
G.R. Putzolu
I.L. Traiger

IBM Research Laboratory
San Jose, California

ABSTRACT: In the first part of the paper the problem of choosing the granularity (size) of lockable objects is introduced and the related tradeoff between concurrency and overhead is discussed. A locking protocol which allows simultaneous locking at various granularities by different transactions is presented. It is based on the introduction of additional lock modes besides the conventional share mode and exclusive mode. A proof is given of the equivalence of this protocol to a conventional one.

In the second part of the paper the issue of consistency in a shared environment is analyzed. This discussion is motivated by the realization that some existing data base systems use automatic lock protocols which insure protection only from certain types of inconsistencies (for instance those arising from transaction backup), thereby automatically providing a limited degree of consistency. Four degrees of consistency are introduced. They can be roughly characterized as follows: degree 0 protects others from your updates, degree 1 additionally provides protection from losing updates, degree 2 additionally provides protection from reading incorrect data items, and degree 3 additionally provides protection from reading incorrect relationships among data items (i.e. total protection). A discussion follows on the relationships of the four degrees to locking protocols, concurrency, overhead, recovery and transaction structure.

Lastly, these ideas are related to existing data management systems.

Modelling in Data Base Management Systems 1976

Agenda

Transaction basics

Locking

Degree of consistency

ACID Properties in Transactions

Atomicity: Either all operations occur, or nothing occurs (all or nothing)

Consistency: Integrity constraints are satisfied

Isolation: How operations of transactions interleave

Durability: A transaction’s updates persist when system fails

This lecture touches A, C, and I

Locking Granularity

Locks are a critical part of concurrency control

Choosing a locking granularity

- Entire database
- Relation
- Records …

Locking Granularity

Locks are a critical part of concurrency control

Choosing a locking granularity

- Entire database
- Relation
- Records …

↓ Increasing concurrency
Increasing overhead when many records are accessed

Goal: high concurrency and low cost

Locking Granularity

Locks are a critical part of concurrency control

Choosing a locking granularity

- Entire database
- Relation
- Records …

⇩ Increasing concurrency
Increasing overhead when many records are accessed

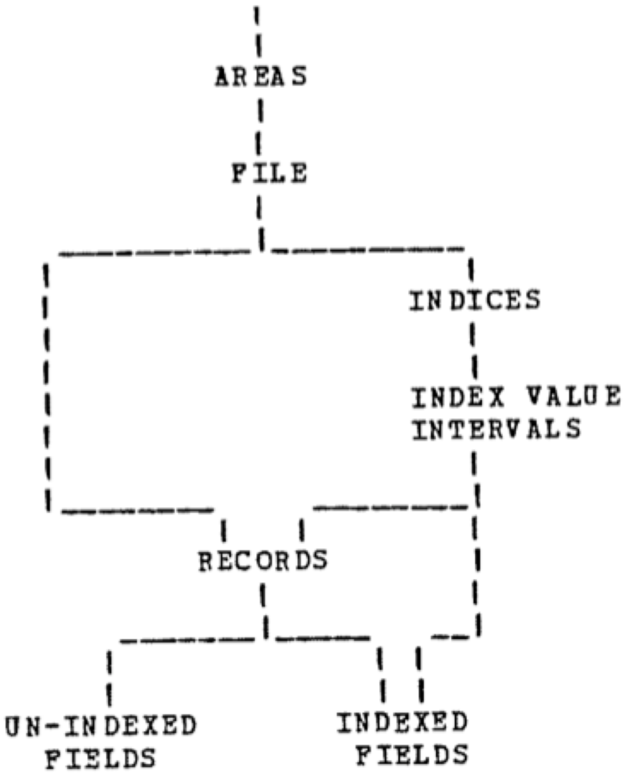
Goal: high concurrency and low cost

Solution: **Hierarchical locks**

Hierarchical Locks

```
DB
|
Areas
|
Files
|
Records
```

```
        DB
        |
      Areas
     /     \
Files     Indices
     \     /
     Records
```

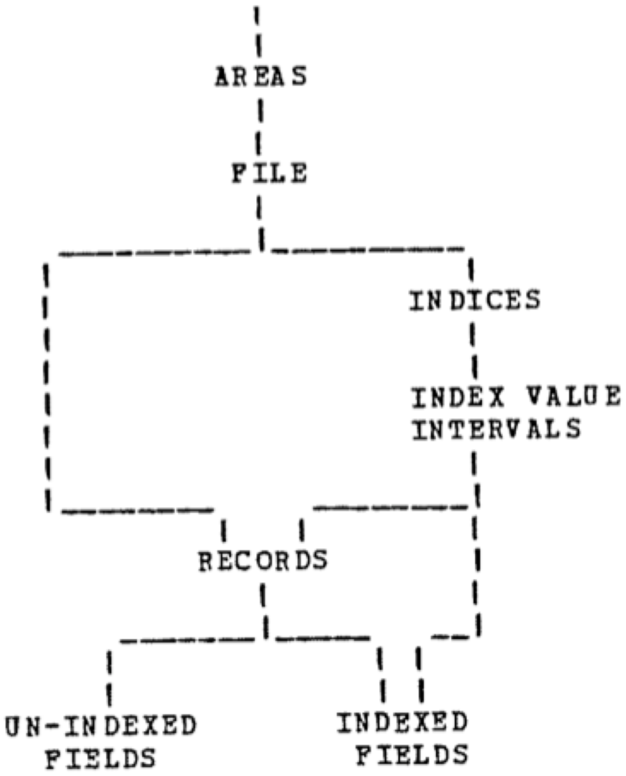


Lock a high-level node if a large number of records are accessed

- All descendants are implicitly locked in the same mode

Hierarchical Locks

```
     DB                      DB
      |                       |
    Areas                   Areas
      |                    /     \
    Files               Files   Indices
      |                    \     /
   Records                 Records
```



Lock a high-level node if a large number of records are accessed

- All descendants are implicitly locked in the same mode
- **Intention lock** to avoid conflict with implicit locks

Locking Modes

Basic locking modes

- S: Shared lock
- X: Exclusive lock

Locking Modes

Basic locking modes

- S: Shared lock
- X: Exclusive lock

Intention modes:

- IS: Intention to share
- IX: Intention to acquire X lock below the lock hierarchy
- SIX: Read large portions and update a few parts

Locking Modes

Basic locking modes

- S: Shared lock
- X: Exclusive lock

Intention modes:

- IS: Intention to share
- IX: Intention to acquire X lock below the lock hierarchy
- SIX: Read large portions and update a few parts

Example: read record (T1)

DB	**IS**
|	
Areas	**IS**
|	
Files	**IS**
|	
Records	**S**

Locking Modes

Basic locking modes

- S: Shared lock
- X: Exclusive lock

Intention modes:

- IS: Intention to share
- IX: Intention to acquire X lock below the lock hierarchy
- SIX: Read large portions and update a few parts

Example: read record (T1) update record (T2)

	read record (T1)	update record (T2)
DB	IS	IX
Areas	IS	IX
Files	IS	IX
Records	S	X

Locking Modes

Basic locking modes

- S: Shared lock
- X: Exclusive lock

Intention modes:

- IS: Intention to share
- IX: Intention to acquire X lock below the lock hierarchy
- SIX: Read large portions and update a few parts

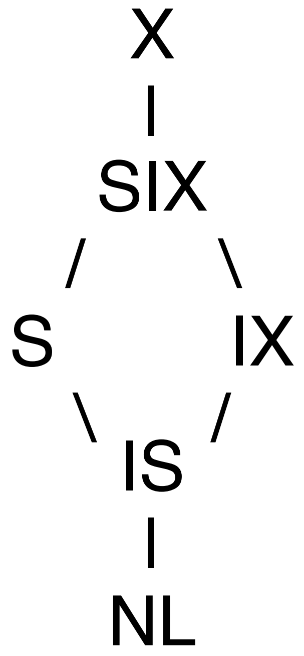
Example:

	read record (T1)	update record (T2)	scan + occasional updates (T3)
DB	**IS**	**IX**	**IX**
Areas	**IS**	**IX**	**IX**
Files	**IS**	**IX**	**SIX**
Records	**S**	**X**	**lock specific records in X mode**

Lock Compatibility

Increasing lock strength →

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



Most privileged

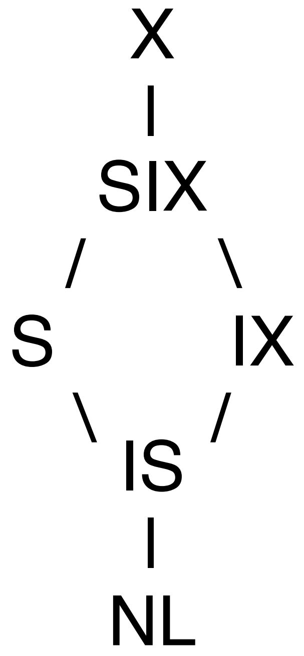
least privileged

Lock Compatibility

Increasing lock strength →

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

Most privileged

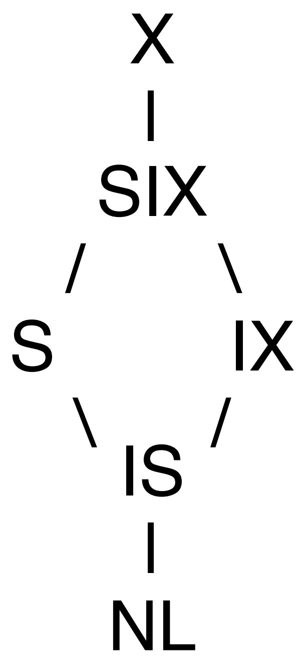


least privileged

Lock Compatibility

Increasing lock strength →

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



Most privileged

least privileged

Rules for Lock Requests

- Before requesting S or IS on a node, all ancestor nodes of the requested node must be held in IS or IX

Rules for Lock Requests

- Before requesting S or IS on a node, all ancestor nodes of the requested node must be held in IS or IX

- Before requesting X, SIX, or IX on a node, all ancestor nodes of the requesting node must be held in SIX or IX

Rules for Lock Requests

- Before requesting S or IS on a node, all ancestor nodes of the requested node must be held in IS or IX

- Before requesting X, SIX, or IX on a node, all ancestor nodes of the requesting node must be held in SIX or IX

- Locks requested **root to leaf**
- Locks released **leaf to root** or any order at the end of the transaction

Summary of Lock Granularity

	Implicit lock	Desc. lock	Anc. lock (DAG)
IS (Intention share)	None	S or IS	IX or IS, at least one parent
IX (Intention exclusive)	None	X, SIX, IX, IS	SIX or IX, all parents
S (Share)	S on all desc	-	IX or IS, at least one parent
SIX (Shared and intention exclusive)	S on all desc	X, SIX, IX	SIX or IX, all parents
X (Exclusive)	X o all desc	-	SIX or IX, all parents

Summary of Lock Granularity

	Implicit lock	Desc. lock	Anc. lock (DAG)
IS (Intention share)	None	S or IS	IX or IS, at least one parent
IX (Intention exclusive)	None	X, SIX, IX, IS	SIX or IX, all parents
S (Share)	S on all desc	-	IX or IS, at least one parent
SIX (Shared and intention exclusive)	S on all desc	X, SIX, IX	SIX or IX, all parents
X (Exclusive)	X o all desc	-	SIX or IX, all parents

Summary of Lock Granularity

	Implicit lock	Desc. lock	Anc. lock (DAG)
IS (Intention share)	None	S or IS	IX or IS, at least one parent
IX (Intention exclusive)	None	X, SIX, IX, IS	SIX or IX, all parents
S (Share)	S on all desc	-	IX or IS, at least one parent
SIX (Shared and intention exclusive)	S on all desc	X, SIX, IX	SIX or IX, all parents
X (Exclusive)	X o all desc	-	SIX or IX, all parents

Summary of Lock Granularity

	Implicit lock	Desc. lock	Anc. lock (DAG)
IS (Intention share)	None	S or IS	IX or IS, at least one parent
IX (Intention exclusive)	None	X, SIX, IX, IS	SIX or IX, all parents
S (Share)	S on all desc	-	IX or IS, at least one parent
SIX (Shared and intention exclusive)	S on all desc	X, SIX, IX	SIX or IX, all parents
X (Exclusive)	X o all desc	-	SIX or IX, all parents

Dynamic Lock Graphs

The lock graph can be dynamic (e.g., indices created, records inserted)

Must deal with **Phantoms**

Phantom Effect

Sailors

Age	Rating
80	1
75	1
90	2
85	2

T1: Find oldest sailors for ratings 1 and 2

T2: Insert (age:99, rating:1) and delete oldest sailor with rating 2

Phantom Effect

Sailors

Age	Rating
80	1
75	1
90	2
85	2

T1: Find oldest sailors for ratings 1 and 2

T2: Insert (age:99, rating:1) and delete oldest sailor with rating 2

T1 locks oldest sailor in rating 1

Phantom Effect

Sailors

Age	Rating
80	1
75	1
90	2
85	2
99	1

T1: Find oldest sailors for ratings 1 and 2

T2: Insert (age:99, rating:1) and delete oldest sailor with rating 2

T1 locks oldest sailor in rating 1

T2 inserts a tuple with (age:99, rating:1)

Phantom Effect

Sailors

Age	Rating
80	1
75	1
~~90~~	~~2~~
85	2
99	1

T1: Find oldest sailors for ratings 1 and 2

T2: Insert (age:99, rating:1) and delete oldest sailor with rating 2

T1 locks oldest sailor in rating 1

T2 inserts a tuple with (age:99, rating:1)

T2 deletes oldest sailor with rating 2

Phantom Effect

Sailors

Age	Rating
80	1
75	1
85	2
99	1

T1: Find oldest sailors for ratings 1 and 2

T2: Insert (age:99, rating:1) and delete oldest sailor with rating 2

T1 locks oldest sailor in rating 1

T2 inserts a tuple with (age:99, rating:1)

T2 deletes oldest sailor with rating 2

T2 commits

Phantom Effect

Sailors

Age	Rating
80	1
75	1
85	2
99	1

T1: Find oldest sailors for ratings 1 and 2

T2: Insert (age:99, rating:1) and delete oldest sailor with rating 2

T1 locks oldest sailor in rating 1

T2 inserts a tuple with (age:99, rating:1)

T2 deletes oldest sailor with rating 2

T2 commits

T1 locks oldest sailor in rating 2

Phantom Effect

Sailors

Age	Rating
80	1
75	1
85	2
99	1

T1: Find oldest sailors for ratings 1 and 2

T2: Insert (age:99, rating:1) and delete oldest sailor with rating 2

T1 locks oldest sailor in rating 1

T2 inserts a tuple with (age:99, rating:1)

T2 deletes oldest sailor with rating 2

T2 commits

T1 locks oldest sailor in rating 2

T1 commits. Output: (80,1), (85, 2)

Phantom Effect

Sailors

Age	Rating
80	1
75	1
85	2
99	1

Phantom

T1: Find oldest sailors for ratings 1 and 2

T2: Insert (age:99, rating:1) and delete oldest sailor with rating 2

Output: (80,1), (85, 2)

Different from all sequential execution output

- T1 -> T2. Output: (80, 1), (90, 2)
- T2 -> T1. Output: (99, 1), (85, 2)

Solution to Phantoms

Observation: Inserts and deletes are **writes** to the index; lookups are **reads** to the index

Can lock the index in **X** or **S** mode

Optimization: lock intervals and predicate locking

- E.g., lock age=80 and the interval of age > 80 (prevent age 99 from inserted)

Degree of Consistency (Isolation)

How can transactions interleave?

One extreme: concurrent execution produces the same results as some serial execution (serializability)

- Limited concurrency and performance
- Intuitive and easy to reason about

Another extreme: transaction operations can arbitrarily interleave

Degree of Consistency (Isolation)

	Locks	Non-Recoverable	Dirty Reads	Non-repeatable or fuzzy Reads	SQL Isolation level	Dependency
Degree 3	Long-X Long-R	No	No	No	Serializable	W->W W->R R->W
Degree 2	Long-X Short-R	No	No	Yes	Read committed	W->W W->R
Degree 1	Long-X	No	Yes	Yes	Read uncommitted	W->W
Degree 0	Short-X	Yes	Yes	Yes		None

Degree of Consistency (Isolation)

	Locks	Non-Recoverable	Dirty Reads	Non-repeatable or fuzzy Reads	SQL Isolation level	Dependency
Degree 3	Long-X Long-R	No	No	No	Serializable	W->W W->R R->W
Degree 2	Long-X Short-R	No	No	Yes	Read committed	W->W W->R
Degree 1	Long-X	No	Yes	Yes	Read uncommitted	W->W
Degree 0	Short-X	Yes	Yes	Yes		None

Degree of Consistency (Isolation)

	Locks	Non-Recoverable	Dirty Reads	Non-repeatable or fuzzy Reads	SQL Isolation level	Dependency
Degree 3	Long-X Long-R	No	No	No	Serializable	W->W W->R R->W
Degree 2	Long-X Short-R	No	No	Yes	Read committed	W->W W->R
Degree 1	Long-X	No	Yes	Yes	Read uncommitted	W->W
Degree 0	Short-X	Yes	Yes	Yes		None

Degree of Consistency (Isolation)

	Locks	Non-Recoverable	Dirty Reads	Non-repeatable or fuzzy Reads	SQL Isolation level	Dependency
Degree 3	Long-X Long-R	No	No	No	Serializable	W->W W->R R->W
Degree 2	Long-X Short-R	No	No	Yes	Read committed	W->W W->R
Degree 1	Long-X	No	Yes	Yes	Read uncommitted	W->W
Degree 0	Short-X	Yes	Yes	Yes		None

Degree of Consistency (Isolation)

	Locks	**Non-Recoverable**	**Dirty Reads**	**Non-repeatable or fuzzy Reads**	**SQL Isolation level**	**Dependency**
Degree 3	Long-X Long-R	No	No	No	Serializable	W->W W->R R->W
Degree 2	Long-X Short-R	No	No	Yes	Read committed	W->W W->R
Degree 1	Long-X	No	Yes	Yes	Read uncommitted	W->W
Degree 0	Short-X	Yes	Yes	Yes		None

Degree of Consistency (Isolation)

	Locks	Non-Recoverable	Dirty Reads	Non-repeatable or fuzzy Reads	SQL Isolation level	Dependency
Degree 3	Long-X Long-R	No	No	No	Serializable	W->W W->R R->W
Degree 2	Long-X Short-R	No	No	Yes	Read committed	W->W W->R
Degree 1	Long-X	No	Yes	Yes	Read uncommitted	W->W
Degree 0	Short-X	Yes	Yes	Yes		None

Degree of Consistency (Isolation)

	Locks	Non-Recoverable	Dirty Reads	Non-repeatable or fuzzy Reads	SQL Isolation level	Dependency
Degree 3	Long-X Long-R	No	No	No	Serializable	W->W W->R R->W
Degree 2	Long-X Short-R	No	No	Yes	Read committed	W->W W->R
Degree 1	Long-X	No	Yes	Yes	Read uncommitted	W->W
Degree 0	Short-X	Yes	Yes	Yes		None

Degree of Consistency (Isolation)

	Locks	Non-Recoverable	Dirty Reads	Non-repeatable or fuzzy Reads	Phantom	SQL Isolation level	Dependency
Degree 3	Long-X Long-R	No	No	No	No	Serializable	W->W W->R R->W
		No	No	No	Yes	Repeatable reads	
Degree 2	Long-X Short-R	No	No	Yes	Yes	Read committed	W->W W->R
Degree 1	Long-X	No	Yes	Yes	Yes	Read uncommitted	W->W
Degree 0	Short-X	Yes	Yes	Yes	Yes		None

Two-Phase Locking

A transaction is two phase if it does not lock an entity after unlocking some entity

- Growing phase: acquiring locks
- Shrinking phase: releasing locks

Two-phase locking (2PL) guarantees serializability

Two-Phase Locking

A transaction is two phase if it does not lock an entity after unlocking some entity

- Growing phase: acquiring locks
- Shrinking phase: releasing locks

Two-phase locking (2PL) guarantees serializability

Strict 2PL: 2PL + all exclusive locks released *after* transaction commits

- Strict 2PL guarantees ACA (Avoiding Cascading Aborts)

Q/A – Granularity of Locks

Multi-granularity locks used in modern database?

Research papers focus on tuple-level locking?

SQL vs. NoSQL regarding locking?

How is the action of placing a lock itself thread-safe?

Implementation of Internal locking? (checkout *next-key locking*)

Before Next Lecture

Submit review for

- H. T. Kung, John T. Robinson, On Optimistic Methods for Concurrency Control. ACM Trans. Database Syst. 1981.