Isolation Levels
(Lecture 14, cs262a)

Ion Stoica,
UC Berkeley
October 12, 2016

Isolation Levels

A feature provided by database management systems in order to increase performance when

- Full correctness is not necessary, or
- Correctness could be assured at the application level

Explicit isolation levels

A transaction can be declared to have isolation properties that are less stringent than serializability

- However SQL standard says that default should be serializable (Gray'75 called this "level 3 isolation")
- In practice, most systems have weaker default level, and most transactions run at weaker levels!

Isolation levels are defined with respect to data access conflicts (phenomena) they preclude

Phenomena

P0: T2 writes value modified by T1 before T1 commits

- Transactions cannot be serialized by their writes

P1 – Dirty Read: T2 reads value modified by T1 before T1 commits

- If T1 aborts it will be as if transaction T2 read values that have never existed

P2 – Non-Repeatable Read: T2 reads value, after which T1 modifies it

- If T1 attempts to re-read value it can read another value

P3 – Phantom: (see next)

Phantom

1. A transaction T1 reads a set of rows that satisfy some condition

2. Another transaction T2 executes a statement that causes new rows to be added or removed from the search condition

3. If T1 repeats the read it will obtain a different set of rows.

Phantom Example

T1

```
Select count(*)
where dept = “Acct”
// find and S-lock (“Sue”, “Acct”,
3500) and (“Tim”, “Acct, 2400)
```

```
Select sum(salary)
where dept = “Acct”
// find and S-lock (“Sue”, “Acct”,
3500) and (“Tim”, “Acct, 2400) and
(“Joe”, “Acct”, 2000)
```

T2

```
Insert (“Joe”,”Acct”, 2000)
// X-lock the new record
Commit
// release locks
```

Isolation Levels

Isolation levels	Degree	Proscribed Phenomena	Read locks on data items and phantoms (same unless noted)	Write locks on data items and phantoms (always the same)
	0	none	none	Short write locks
READ UNCOMMITTED	1	P0	none	Long write locks
READ COMITTED	2	P0, P1	Short read locks	Long write locks
REAPEATABLE READ		P0, P1, P2	Long data-item read locks, short phantom locks	Long write locks
SERIALIZABLE	3	P0, P1, P2, P3	Long read locks	Long write locks

ANSI

Gray’s isolation degrees

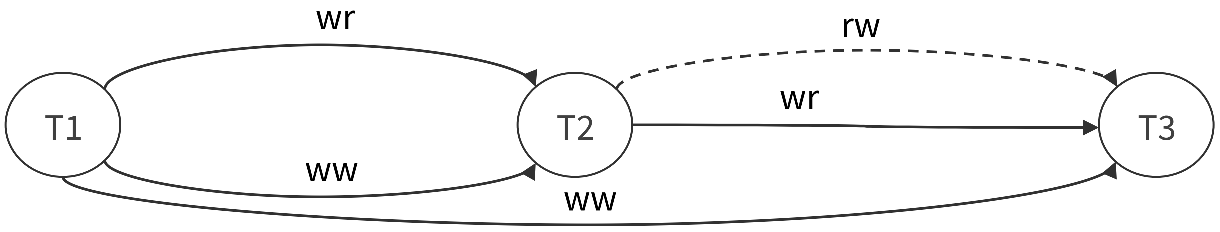
Generalized Isolation Levels

Direct Serialization Graph (DSG)

Conflict Name	Description	DSG
Directly write-depends	T1 writes value, then T2 overwrites it	$T1 \xrightarrow{ww} T2$
Directly read-depends	T1 writes value, then T2 reads it	$T1 \xrightarrow{wr} T2$
Directly anti-depends	T1 reads value, then T2 writes it	$T1 \overset{rw}{\dashrightarrow} T2$

Example:

```
T1:W(A), W(B), W(C)
T2:                       R(B), W(C)
T3:                 W(B)             R(C), W(B)
```



Disallowing P0

Writes by T1 are not overwritten by T2 while T1 is uncommitted

- Simplifies recovery from aborts, e.g.,
  - T1 updates x, T2 overwrites x , and then T1 aborts
  - The system must not restore x to T1's pre-state
  - However, if T2 aborts later, x must be restored to T1's pre-state!
- Serializes transactions based on their writes alone
  - all writes of T2 must be ordered before or after all writes of T1

G0 just disallows this one

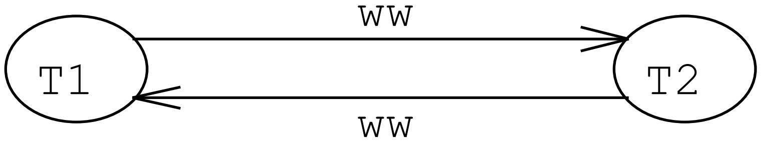
G0

G0: DSG contains a directed cycle consisting entirely of write-dependency edges

- Just ensure serialization on writes alone
- More permissive than Degree 1 as allows concurrent transactions to modify same object

Example:

```
T1:W(A)                    W(B), …
T2:      W(A), W(B), …
```



Disallowing P1

Writes of T1 could not be read by T2 while T1 is still uncommitted

- It prevents a transaction T2 from committing if T2 has read the updates of a transaction that might later abort
- It prevents transactions from reading intermediate modifications of other transactions
- It serializes committed transactions based on their read/write-dependencies (but not their antidependencies), i.e.,
  - If transaction T2 depends on T1, T1 cannot depend on T2

G1

G1a – Aborted reads: T2 has read a value written by an aborted transaction T1

G1b – Intermediate Reads: Committed transaction T2 has read an intermediate value written by transaction T1

G1c – Circular Information Flow: DSG contains a directed cycle consisting entirely of dependency edges

Disallowing P2

T1 cannot modify value read by T2

- Precludes a transaction reading inconsistent data and making inconsistent updates

G2

Just prevent transactions that perform inconsistent reads or writes from committing

G2 – Anti-dependency Cycles: DSG contains a directed cycle with one or more anti-dependency edges

G2-item – Item Anti-dependency Cycles: DSG contains a directed cycle having one or more item-antidependency edges

Generalized Isolation Levels

Isolation levels	G0	G1	G2-Item	G2
READ UNCOMMITTED	NA	NA	NA	NA
READ COMITTED	Not possible	Possible	Possible	Possible
REAPEATABLE READ	Not possible	Not possible	Not possible	Possible
SERIALIZABLE	Not possible	Not possible	Not possible	Not possible

Summary

Transactions, key abstractions on databases

- Application defined sequence of operations on one or more databases that is atomic

Key challenge: trade performance to correctness

- On one hand we want to interleave transactions to increase throughput
- On the other hand we want to isolate transactions from each other

Solution: increase interleaving by providing

- Multi-granularity locks
- Relax the isolation semantics

Announcements

Next Monday: Project Progress Review

Create a Google doc (in addition to your proposal) specifying:

- Problem you are solving (this can be the same as in your project proposal if it didn't change)
- Design, e.g.,
  - Briefly described the alternatives you considered if more than one
  - Architecture diagram of your system
  - Short description of functionality provided by each component and its API
- Preliminary results, any
- No more than 3 pages