Transactions & Update Correctness

April 11, 2018
Correctness
Correctness

- Data Correctness (Constraints)
- Query Correctness (Plan Rewrites)
Correctness

- Data Correctness (Constraints)
- Query Correctness (Plan Rewrites)
- Update Correctness (Transactions)
What could go wrong?

- **Parallelism**: What happens if two updates modify the same data?

- Maximize use of IO / Minimize Latencies.

- **Persistence**: What happens if something breaks during an update?

- When is my data safe?
What does it mean for a database operation to be correct?
What does it mean for a database operation to be correct?
What is an Update?

- INSERT INTO ...
- UPDATE ... SET ... WHERE ...
- Non-SQL?
What is an Update?

• INSERT INTO …?

• UPDATE … SET … WHERE …?

• Non-SQL?

Can we abstract?
Abstract Update Operations
Abstract Update Operations

Time
Abstract Update Operations

Time

Read

Read

Read
Abstract Update Operations

Read

Read

Read

Write

Write

Time
Abstract Update Operations

Time

- Read
- Read
- Read
- Write
- Write
- Write
- Abort
- Commit
Abstract Update Operations

[Transaction]

Read → Read → Read → Write → Write → Write → Commit → Abort

Time
What does it mean for a database operation to be correct?
Transaction Correctness

- From the user’s perspective, transactions...
  - ... execute fully or not at all. (atomicity)
  - ... preserve integrity constraints (correctness)
  - ... execute as if on their own (isolation)
  - ... have their outputs persisted (durability)
Atomicity

• A transaction completes by committing, or terminates by aborting.

• **Logging** is used to undo aborted transactions.

• **Atomicity**: A transaction is (or appears as if it were) applied in one ‘step’, independent of other transactions.

• All ops in a transaction commit or abort together.
Isolation

T1: BEGIN A=A+100, B=B-100 END
T2: BEGIN A=1.06*A, B=1.06*B END

• Intuitively, T1 transfers $100 from A to B and T2 credits both accounts with interest.

• What are possible interleaving errors?
Example: Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A=A+100</td>
<td>A=1.06*A</td>
</tr>
<tr>
<td></td>
<td>B=B-100</td>
<td>B=1.06*B</td>
</tr>
</tbody>
</table>
Example: Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A=A+100</td>
<td>A=1.06*A</td>
</tr>
<tr>
<td></td>
<td>B=B-100</td>
<td>B=1.06*B</td>
</tr>
</tbody>
</table>

OK!
Example: Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A = A + 100</td>
<td>A = 1.06 * A</td>
</tr>
<tr>
<td></td>
<td>B = B - 100</td>
<td>B = 1.06 * B</td>
</tr>
</tbody>
</table>
Example: Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A = A + 100</td>
<td>A = 1.06 * A</td>
</tr>
<tr>
<td></td>
<td>B = B - 100</td>
<td>B = 1.06 * B</td>
</tr>
</tbody>
</table>

Not OK!
Example: The DBMS’s View

<table>
<thead>
<tr>
<th>Time</th>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R(A)</td>
<td>R(A)</td>
</tr>
<tr>
<td></td>
<td>W(A)</td>
<td>W(A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R(B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W(B)</td>
</tr>
<tr>
<td></td>
<td>R(B)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W(B)</td>
<td></td>
</tr>
</tbody>
</table>
Example: The DBMS’s View

Time

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R(A)</td>
<td>R(A)</td>
</tr>
<tr>
<td></td>
<td>W(A)</td>
<td>W(A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>R(B)</td>
<td>R(B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W(B)</td>
</tr>
<tr>
<td></td>
<td>R(B)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W(B)</td>
<td></td>
</tr>
</tbody>
</table>

Not OK!
What went wrong?
What could go wrong?

Reading uncommitted data
(write-read/WR conflicts; aka “Dirty Reads”)

T1: R(A), W(A), R(B), W(B), ABRT
T2: R(A), W(A), CMT,

Unrepeatable Reads
(read-write/RW conflicts)

T1: R(A), R(A), W(A), CMT
T2: R(A), W(A), CMT,
What could go wrong?

Overwriting Uncommitted Data
(write-write/WW conflicts)

T1: \( W(A), \quad W(B), CMT \)
T2: \( W(A), W(B), CMT, \)
Schedule
An ordering of read and write operations.

Serial Schedule
No interleaving between transactions at all

Serializable Schedule
Guaranteed to produce equivalent output to a serial schedule
Conflict Equivalence

Possible Solution: Look at read/write, etc… conflicts!

Allow operations to be reordered as long as conflicts are ordered the same way

Conflict Equivalence: Can reorder one schedule into another without reordering conflicts.

Conflict Serializability: Conflict Equivalent to a serial schedule.
Conflict Serializability

- **Step 1:** Serial Schedules are Always Correct
- **Step 2:** Schedules with the same operations and the same conflict ordering are conflict-equivalent.
- **Step 3:** Schedules conflict-equivalent to an always correct schedule are also correct.
  - … or conflict serializable
Example

Time

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R(B)</td>
<td>W(B)</td>
</tr>
<tr>
<td>W(A)</td>
<td>R(A)</td>
</tr>
</tbody>
</table>

vs.

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R(B)</td>
<td>W(B)</td>
</tr>
<tr>
<td>W(A)</td>
<td>R(A)</td>
</tr>
</tbody>
</table>
Example

Time

T1

W(B)

R(B)

W(A)

R(A)

T2

Conflict

R(B)

W(B)

R(A)

W(A)

R(A)

W(B)

R(A)

W(A)
Example

\[ \begin{align*}
&\text{Time} \\
&T1 \quad T2 \\
&W(B) \quad R(B) \\
&W(A) \quad R(A) \\
&1: T2 \rightarrow T1 \\
&2: T1 \rightarrow T2 \\
\end{align*} \]

\[\neq\]

\[\begin{align*}
&\text{Time} \\
&T1 \quad T2 \\
&W(B) \quad R(B) \\
&W(A) \quad R(A) \\
&1: T2 \rightarrow T1 \\
&2: T2 \rightarrow T1 \\
\end{align*} \]
Example
Equivalence

• Look at the actual effects
  • Can’t determine effects without running
• Look at the conflicts
  • Too strict
• Look at the possible effects
Information Flow

T1
Information Flow

Old State

T1
Information Flow

Old State → T1 → New State
Information Flow

Old State — T1 — New State

R(...)
Information Flow

T1 -> T2 -> T3

R(…)

[Diagram showing the flow of information through T1, T2, and T3 with an annotation R(…).]
Information Flow

T1 → T2 → T3

R(…) → R(…) → R(…)
Information Flow

T1 → T2 → T3

R(…)

Important
Information Flow

T1 → T2 → T3

Important

Not Important

R(…)

Three women icons are present, indicating the flow of information between the entities.
Information Flow

Multiple Transactions

R(…)
R(…)
R(…)

[Diagram showing data flow between databases and users]
View Serializability

Possible Solution: Look at data flow!

View Equivalence: All reads read from the same writer
Final write in a batch comes from the same writer

View Serializability: Conflict Equivalent to a serial schedule.
View Equivalence

• For all Reads R
  • If R reads old state in S1, R reads old state in S2
  • If R reads Ti’s write in S1, R reads the the same write in S2
• For all values V being written.
  • If W is the last write to V in S1, W is the last write to V in S2
• If these conditions are satisfied, S1 and S2 are view-equivalent
View Serializability

• **Step 1**: Serial Schedules are Always Correct

• **Step 2**: Schedules with the same information flow are view-equivalent.

• **Step 3**: Schedules view-equivalent to an always correct schedule are also correct.

• … or view serializable
Example

Time

T1  T2  T3

R(A)
W(A)
W(A)

W(A)
Example

- Time

  - T1
    - R(A)
    - W(A)
  - T2
    - W(A)
  - T3
    - W(A)
Example

Write order irrelevant (T3 overwrites either way)
Enforcing Serializability
Enforcing Serializability

• Conflict Serializability:
  • Does locking enforce conflict serializability?
Enforcing Serializability

- Conflict Serializability:
  - Does locking enforce conflict serializability?
- View Serializability
  - Is view serializability stronger, weaker, or incomparable to conflict serializability?
Enforcing Serializability

- Conflict Serializability:
  - Does locking enforce conflict serializability?

- View Serializability
  - Is view serializability stronger, weaker, or incomparable to conflict serializability?

- What do we need to enforce either fully?