CS 331
- Kangmei Yang
Transaction

ACID Property: atomicity, consistency, isolation, durability

BEGIN TRANSACTION
....active, partially committed – ABORTED
COMMIT/ROLLBACK

Consistent state?

<table>
<thead>
<tr>
<th>Time</th>
<th>Trans1</th>
<th>Trans2</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>begin_tras</td>
<td>begin_tras</td>
</tr>
<tr>
<td>t2</td>
<td>read(bal)</td>
<td>read(bal)</td>
</tr>
<tr>
<td>t3</td>
<td>read(bal)</td>
<td>bal = bal + 5</td>
</tr>
<tr>
<td>t4</td>
<td>bal = bal - 10</td>
<td>write(bal)</td>
</tr>
<tr>
<td>t5</td>
<td>write(bal)</td>
<td>commit</td>
</tr>
<tr>
<td>t6</td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>
## Transaction

### Consistent state?

<table>
<thead>
<tr>
<th>Time</th>
<th>Trans1</th>
<th>Trans2</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t5</td>
<td>begin_tras</td>
<td>begin_tras</td>
</tr>
<tr>
<td>t6</td>
<td>read(bal)</td>
<td>bal = bal + 5</td>
</tr>
<tr>
<td>t7</td>
<td>bal = bal - 10</td>
<td>write(bal)</td>
</tr>
<tr>
<td>t8</td>
<td>write(bal)</td>
<td>..</td>
</tr>
<tr>
<td>t9</td>
<td>commit</td>
<td>rollback</td>
</tr>
</tbody>
</table>

### Problem?
Concurrent transaction in nonserial schedule, when one transaction write a data item, another one read or write the same data item, the order of execution is important.

Check: Serializability
## Locking Method

**Shared lock** – read only
**Exclusive lock** – read and write

### Two-phase locking
- **Growing phase** – acquire locks
- **Shrinking phase** – release locks

<table>
<thead>
<tr>
<th>Time</th>
<th>Trans1</th>
<th>Trans2</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td></td>
<td>begin_tras</td>
</tr>
<tr>
<td>t2</td>
<td>begin_tras</td>
<td>wrt_lock(bal)</td>
</tr>
<tr>
<td>t3</td>
<td>wrt_lock(bal)</td>
<td>read(bal)</td>
</tr>
<tr>
<td>t4</td>
<td>wait</td>
<td>bal = bal + 5</td>
</tr>
<tr>
<td>t5</td>
<td>wait</td>
<td>write(bal)</td>
</tr>
<tr>
<td>t6</td>
<td>wait</td>
<td>commit/unlock</td>
</tr>
<tr>
<td>t7</td>
<td>read(bal)</td>
<td></td>
</tr>
<tr>
<td>t8</td>
<td>bal = bal − 10</td>
<td></td>
</tr>
<tr>
<td>t9</td>
<td>write(bal)</td>
<td></td>
</tr>
<tr>
<td>t10</td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>
DeadLock

When two or more transaction are each waiting for locks to be released that are held by other

Solve: Wait-die, wound-wait
Recovery

Restoring the database to a correct state in the event of failure

Backup
- Log File (either entire database or incremental backup) to track database transactions
Query Processing

Parsing, validating, optimizing, and executing a query

Query decomposition (Relational algebra expression)
- transform a high level query into a relational algebra query (parsing)
- check whether the query is syntactically and semantically correct (validating)

Query Optimization (Execution plan)
consider: 1000 Students, 10 Departments

\[ \Pi_{f\_name, \ dept} \left( \sigma_{cuny\_id='12345678'} \left( \text{Student} \bowtie_{s.\ dept = d.\ dept\_code} \text{Department} \right) \right) \]

\[ \Pi_{f\_name, \ dept} \left( \sigma_{cuny\_id='12345678'} \left( \text{Student} \right) \right) \bowtie_{s.\ dept = d.\ dept\_code} \text{Department} \]

Code generation (generated code)

Runtime query execution (output)
Query Processing

Query decomposition (Relational algebra expression)
- analysis, normalization, semantic analysis, simplification, and query restructuring

Analysis
- if the query is lexically and syntactically correct, then transform to a relational algebra tree

ex: \((\Pi_{f\_name, \ dept} (\sigma_{cuny\_id= \text{'12345678'}} (\text{Student}))))\)

\(\bowtie_{s.\ dept = d.\ dept\_code} \text{Department}\)

\(\sigma_{cuny\_id= \text{'12345678'}}\)
Query Processing

Query decomposition (Relational algebra expression)
- analysis, normalization, semantic analysis, simplification, and query restructuring

Normalization
- converts the complex predicate to either conjunctive normal form or disjunctive normal form

conjunctive
( ... \lor ... ) \land ( ... \lor ... \lor ... ) \land ...  
disjunctive
( ... \land ... ) \lor ( ... \land ... \land ... ) \lor ...
Query Processing

Query decomposition (Relational algebra expression)
- analysis, normalization, semantic analysis, simplification, and query restructuring

Semantic analysis
- check if the query contains components that do not contribute to the result
  if queries do not contain disjunction and negation, may check either by relation connection graph or normalized attribute connection graph
Semantic Analysis

SELECT s.f_name, s.l_name
FROM STUDENT s, MAJOR m
WHERE m.mjr_name = 'Computer Science' AND s.type = 'Undergrad'

relation connection graph

- missing s.major = m.mjr_code

normalized attribute connection graph

\[
\begin{align*}
s & \rightarrow \text{result} \\
m & \rightarrow \text{result} \\
s.major & \leftrightarrow m.mjr_code \\
m.mjr_name & \leftrightarrow \text{U}' \\
- \text{CS}' & \leftrightarrow \text{U}' \\
- \text{CS}' & \leftrightarrow \text{U}' \\
\end{align*}
\]
normalized attribute connection graph

SELECT sku
FROM ITEM
WHERE price > 100 AND price <= 200

for inequality, if attribute >= value, put directed edge from 0 to attribute with negative value, if attribute <= value, then put directed edge from attribute to 0 with positive value. It’s contradictory if there’s a negative cycle sum.

Ex: price > 300 AND price <= 200
Query Processing

Query decomposition (Relational algebra expression)
- analysis, normalization, semantic analysis, simplification, and query restructuring

Simplification
- detect redundant qualifications, eliminate common subexpressions, transform the query to a semantically equivalent but more easily and efficiently computed form.

Idempotency rules of boolean algebra:
\[
\begin{align*}
    p \land p &= p \\
    p \land \text{false} &= \text{false} \\
    p \land \text{true} &= p \\
    p \land \neg p &= \text{false} \\
    p \land (q \lor p) &= p \\
    p \lor p &= p \\
    p \lor \text{false} &= p \\
    p \lor \text{true} &= \text{true} \\
    p \lor \neg p &= \text{true} \\
    p \lor (q \land p) &= p
\end{align*}
\]

Sample:
```
SELECT * FROM HONORSTUDENT
WHERE GPA < 3.4
```
If there’s integrity constraint that gpa is over 3.5 for HONORSTUDENT THEN CONTRADICTORY