Invariant Control in Eventually Consistent Databases.

Paulo Arion Flores and Frank Siqueira
Paulo Arion Flores

System Analyst (CIASC)  
MSc Candidate in Computer Science (UFSC)
BSc in Computer Science (UFSC)

pauloarion@gmail.com
https://www.linkedin.com/in/paulo-arion-flores-9b692313
Summary

- Context
- Objectives
- Related Work
- Solution
- Example
- Evaluation
- Conclusion
Context

- Distributed Systems.
- Replicated databases.
- **Consistency.**
- ACID
- CAP
NoSQL

- Relational
- NoSQL
- NewSQL
- Data model
- Developers
Invariants

- Many approaches
- RDT
- Intermediate languages
- FOL
- Consistency levels
- Integrity constraints
- Multi-variable
Objectives

▪ Replicated DDBS consistency.

▪ Integrity Constraints using RDT and FOL contracts.

▪ Multi-variable RDT control mechanism.
## Related work

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Invariants</th>
<th>Consistency</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>RedBlue (Li et al)</td>
<td>States</td>
<td>Red/Blue</td>
<td>Any</td>
</tr>
<tr>
<td>Indigo (Balegas et al)</td>
<td>Hoare logic</td>
<td>Reservations</td>
<td>Any</td>
</tr>
<tr>
<td>I-Confluence (Bailis et al)</td>
<td>States</td>
<td>Coordination-free</td>
<td>Any</td>
</tr>
<tr>
<td>CISE (Gotsman et al)</td>
<td>RDT</td>
<td>Hybrid</td>
<td>Any</td>
</tr>
<tr>
<td>SIEVE (Li et al)</td>
<td>States/CRDT</td>
<td>Red/Blue</td>
<td>SQL</td>
</tr>
<tr>
<td>QUELEA (Sivaramakrishnan et al)</td>
<td>RDT/FOL</td>
<td>Contracts</td>
<td>Any</td>
</tr>
<tr>
<td>Homeostasis protocol (Roy et al)</td>
<td>Symbolic tables</td>
<td>LR-slices and treaties</td>
<td>Any</td>
</tr>
</tbody>
</table>
Solution
Invariant Description

Model
+ attribute: type
+ attribute: type
+ attribute: type

Relation

Model
+ attribute: type
+ attribute: type
+ attribute: type

Constraint
0..*

Constraint = CHECKON | NOTNULL | UNIQUE | REFERENCES

Predicate expression = Predicate
| Predicate ∨ Predicate
| Predicate ∧ Predicate
| Predicate → Predicate

Predicate = IN | BETWEEN | FUNCTION
Transition states consistency

\[(A + B > 10) \rightarrow (C < 50)\]

\[
\begin{align*}
\uparrow & \quad \text{A and B} \\
& \quad \text{BECOMING TRUE} \\
& \quad \text{MORE TRUE} \\
& \quad \text{LESS FALSE} \\
\downarrow & \quad \text{C}
\end{align*}
\]

\[
\begin{align*}
\downarrow & \quad \text{A and B} \\
& \quad \text{LESS TRUE} \\
& \quad \text{BECOMING FALSE} \\
& \quad \text{MORE FALSE} \\
\uparrow & \quad \text{C}
\end{align*}
\]

\[
\begin{align*}
\text{=} & \quad \text{A and B} \\
& \quad \text{KEEPING TRUE} \\
& \quad \text{KEEPING FALSE} \\
\text{=} & \quad \text{C}
\end{align*}
\]
RDT Generation

- **NOT NULL** $\rightarrow$ No RDT
- **UNIQUE** $\rightarrow$ Simple RDT

- **REFERENCES** $\rightarrow$ INSERT_PRIMARY_KEY $\rightarrow$ RDT
  | REMOVE_PRIMARY_KEY
  | INSERT_FOREIGN_KEY
  | REMOVE_FOREIGN_KEY

- **CHECK ON** $\rightarrow$ BECOMING TRUE | LESS TRUE $\rightarrow$ RDT
  | MORE TRUE | KEEPING TRUE
  | BECOMING FALSE | LESS FALSE
  | MORE FALSE | KEEPING FALSE
# RDT Consistency

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>P → P</th>
<th>P ∧ P</th>
<th>P ∨ P</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Becoming true</strong></td>
<td>Rejected</td>
<td>Both</td>
<td>Rejected</td>
<td>Eventual</td>
</tr>
<tr>
<td><strong>More true</strong></td>
<td>Eventual</td>
<td>Eventual</td>
<td>Eventual</td>
<td>Eventual</td>
</tr>
<tr>
<td><strong>Less true</strong></td>
<td>Strong</td>
<td>Both</td>
<td>Strong</td>
<td>Strong</td>
</tr>
<tr>
<td><strong>Keeping true</strong></td>
<td>Eventual</td>
<td>Eventual</td>
<td>Eventual</td>
<td>Eventual</td>
</tr>
<tr>
<td><strong>Becoming false</strong></td>
<td>Rejected</td>
<td>Both</td>
<td>Rejected</td>
<td>Strong</td>
</tr>
<tr>
<td><strong>More false</strong></td>
<td>Rejected</td>
<td>Eventual</td>
<td>Rejected</td>
<td>Eventual</td>
</tr>
<tr>
<td><strong>Less false</strong></td>
<td>Rejected</td>
<td>Both</td>
<td>Rejected</td>
<td>Eventual</td>
</tr>
<tr>
<td><strong>Keeping false</strong></td>
<td>Rejected</td>
<td>Eventual</td>
<td>Rejected</td>
<td>Eventual</td>
</tr>
</tbody>
</table>
QUELEA

- Language
- FOL
- RDT
- Haskell
- Cassandra layer

Authors: KC Sivaramakrishnan, Gowtham Kaki, and Suresh Jagannathan
Example

Taxpayer
+ id: long
+ createdDate: Date
+ startDate: Date

Member
+ idTaxPayer: long
+ idPerson: long
+ type: RelationType

Person
+ id: long
+ name: String
+ type: PersonType

Constraint: Started date > Created date

Member references: Member model contains 1 taxpayer and 1 person.

RDT
+ startDateBecomingTrue(value):Bool
+ startDateMoreTrue(value):Bool
+ startDateLessTrue(value):Bool
+ startDateKeepingTrue(value):Bool
+ createdDateBecomingTrue(value):Bool
+ createdDateMoreTrue(value):Bool
+ createdDateLessTrue(value):Bool
+ createdDateKeepingTrue(value):Bool

RDT
+ idTaxPayerRemoveForeignKey():Bool
+ idTaxPayerInsertForeignKey():Bool
+ idTaxPayerRemovePrimaryKey():Bool
+ idTaxPayerInsertPrimaryKey():Bool
+ idPersonRemoveForeignKey():Bool
+ idPersonInsertForeignKey():Bool
+ idPersonRemovePrimaryKey():Bool
+ idPersonInsertPrimaryKey():Bool
Anomalies

created = 05/14/2017,
started = 05/16/2017

S1
started =
05/15/2017

created =
05/15/2017

S2
started =
05/15/2017

created =
05/15/2017

Less true updates conflict.

created = 05/14/2017,
started = 05/16/2017

S1
created =
05/10/2017

started =
05/13/2017

S2
started =
05/13/2017

created =
05/10/2017

Less true update after more true update.
Example using QUELEA

startedDateLessTrue :: [ConstraintRDT] -> (UTCTime, UTCTime) -> Resab Bool
startedDateLessTrue ctx (value, t) =
  if (isValid value)
    then (True, Just $ StartedDateLessTrue_value t)
    else (False, Nothing)

startedDateLessTrueContract :: Contract Operation
startedDateLessTrueContract x = forall_ $ a ->
  liftProp $ (vis a x ∨ vis x a ∨ appRel SameEff x a)

select :: Contract Operation
select x = forallQ_ [startedDateMoreTrue, startedDateKeepingTrue] $ a ->
  forallQ_ [createdDateMoreTrue] $ adep ->
  forallQ_ [createdDateMoreTrue, createdDateKeepingTrue] $ b ->
  forallQ_ [startedDateMoreTrue] $ bdep ->
  liftProp $ ((vis adep a ∧ vis a x) ⇒ (vis adep x)) ∧
  ((vis bdep b ∧ vis b x) ⇒ (vis bdep x))
Evaluation - Inequation

[Graph showing the relationship between the number of database servers and the duration (seconds) for different performance levels: Strong, TSC, and Eventual.]
Evaluation - Inequation

The diagram illustrates the throughput (ops/s) across different database servers for three different systems: Strong, TSC, and Eventual. The throughput decreases as the number of database servers increases, indicating a potential bottleneck or inefficiency in the system's performance as more servers are added.
Evaluation - Inequation

The graph illustrates the relationship between database servers and integrity violations.

- **X-axis**: Database servers (1 to 5)
- **Y-axis**: Integrity violations
- **Line**: Eventual
Evaluation - Foreign key

![Graph showing duration (seconds) vs. database servers for Strong, TSC, and Eventual consistency levels.]
Evaluation - Foreign key

![Graph showing throughput vs database servers for different lock types: Strong, TSC, Eventual.]
Evaluation - Foreign key

Integrity violations

Database servers
Conclusion

- RDT for expressions
- Semantic and syntax
- Variable share in RDT
- Cache
- Framework
Questions