Managing Virtual Records in Relational Databases

Frank Wm. Tompa
Ahmed Ataullah
ACKNOWLEDGEMENTS


Research funded in part by NSERC BIN Strategic Network, RIM, and Open Text Corporation.
Enterprises must ensure business records (typically documents) are available for:

- Operational needs
- Legal / regulatory compliance
- Fiscal management
- Other vital information
- Historical value

**Key aspects:**

- Preservation
- Accuracy
- Efficient access
WHY DATABASES?

Database Management System

Insert / Update / Delete

Queries & Results

Database

Emp

Dept

.....
WHY DATABASES?

Preservation, accuracy, & efficient access is what database systems do well.

Consistency and Integrity

Insert / Update / Delete

Queries & Results

Declarative Query Processing

Scalability

Database Management System

Database

Emp

Dept
Absent extraordinary circumstances, if an organization has implemented a clearly defined records management program specifying what information and records should be kept for legal, financial, operational or knowledge value reasons and has set appropriate retention systems or periods, then information not meeting these retention guidelines can, and should, be destroyed. [Sedona Guidelines]

**Improved efficiency (space and time)**

- More applicable to physical records

**Protection during litigation or government investigation**

- Avoid superfluous liability
- Avoid releasing collateral information; protect privacy
- Must prove it was destroyed as part of corporate *retention policy* (documented and consistently applied) and it is *irrecoverable*

Database systems are *not* good at this.
Business policy:

- Summary of a set of rules; a high-level overview
- Specification of what should (or should not) happen in the operations of a business
  - Typically written in natural language
  - Policy modeling: Hierarchical access control, Object Constraint Language, …
- Includes records management, workflow, privacy, regulatory compliance, etc.

Business policy model implemented by DB policy model
DATABASE SYSTEM PERSPECTIVE

DB instance must comply with published policy

- DBMS continuously monitors updates to ensure compliance.
  - Using check constraints and transaction termination triggers
  - Access control restrictions
- DBMS provides a single compliance layer:
  - No need for policy checking logic in every application
  - Platform for detecting policy conflicts and guaranteeing compliance

Need policy-to-constraint clarity and manageability

- How to make the task of the programmer easier?
  - Mapping business policies → DB constraints
- How to make database rules understandable to business managers?
  - Mapping business policies ← DB constraints
DIFFICULTIES

Each department deals with separate policies.

Many constraints are complex (temporal, conditional, path oriented).

• Typically correspond to complex triggers invoked by transaction termination
• e.g., A private physician in Ontario must keep patient records for 15 years after the last entry in the record, or for 10 years after the patient turns 18 years old, whichever comes later, but cannot delete a record containing information that has been requested under PHIPA nor a record subject to a litigation hold.

Business policies across enterprise produce many constraints

• Scale ⇒ manageability a major problem for DB administrators and programmers
TALK OUTLINE

➢ What is a record in a relational database?
➢ How can record lifecycles be specified?
➢ How can update constraints be enforced?
➢ How can record disposition be enforced?

Disclaimer

➢ Mechanisms described are not available today as tools
➢ Mechanisms can all be implemented with today’s tools
  • Relational database management systems
  • Symbolic model checkers
### Records From Databases

A record is data presented in a certain context such that it holds meaning for a user.
### Underlying data may be shared

- Multiple policies can apply to a single datum
- Retention policies must be enforced carefully
Formally, a record type is a logical view specified over a fixed physical schema

- The result of any (meaningful) query posed to a database
- Class of records
- *Record* is a uniquely identifiable row in such a view

**Specified by**

- Policy managers, DBA, data analysts, lawyers

**Record as row in a view ⇒**

- More expressive than rows in a base table
- Unambiguous, declarative, subject to formal reasoning
- Many already defined for operational reasons
- Represents the states of a business object
**RECORD STATES**

State = condition on records in the view

- Object $x$ is in state $S \iff$ its attributes satisfy $S(x)$
- Example

```
define record UPS_ORDERS as
  select *
  from (ORDERS natural join CUSTOMERS) left outer join
       SHIPMENTS on ORDERS.ShipRef = SHIPMENTS.ShipRef
       where ShipMethod = 'UPS'

- UPS_ORDERS is user-defined view
- “Order objects” are rows in the view

define state Paid on UPS_ORDERS
  where Paid = true

- Object $O$ (tuple $t$) is in state $P$, the “Paid state”, if the condition
  UPS_ORDERS.Paid = true for $t$

⇒ An object can be in multiple states at once.
```
In workflows, for example,

- Rectangles represent business states
- Transitions represent processes/actions
- Stick-figures represent agents
- Constraints implied by absence of transitions
  - E.g., Paid orders cannot go back to awaiting approval
States of an object are typically an interpretation of stages in business process.
States of an object are typically an interpretation of stages in business process.

- Homomorphism from business states to DB states
States of an object are typically an interpretation of stages in business process.

- Homomorphism from business states to DB states

<table>
<thead>
<tr>
<th>State</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awaiting Approval</td>
<td>Approved = false</td>
</tr>
<tr>
<td>Paid</td>
<td>Paid = true</td>
</tr>
<tr>
<td>Awaiting Payment</td>
<td>Approved = true AND Paid = false</td>
</tr>
</tbody>
</table>
States of an object are typically an interpretation of stages in business process.

- Homomorphism from business states to DB states
- Including object creation and destruction
States of an object are typically an interpretation of stages in business process.

- Homomorphism from business states to DB states
- Including object creation and destruction

All constraints should be made explicit in such a constraint diagram.
CONSTRAINTS ON TRANSITIONS

Define various types of constraint

- Syntax: diagrammatic form
- Semantics: using past temporal logic

Special interpretations for transitions to or from Φ

\[ \Diamond A(x) \Rightarrow \neg B(x) \]

\[ \neg \Box B(x) \land B(x) \Rightarrow \Box A(x) \land P \]

\[ \Box A(x) \land \neg A(x) \Rightarrow B(x) \land P \]
ALLOW COMPLEX PATH CONSTRAINTS

For example, an object should never reach state C if it has previously transitioned from A to B

Constraint 1: $B(r) \land \neg \Diamond B(r) \Rightarrow \Box A(r)$

Constraint 2: $\Diamond 1(r) \Rightarrow \neg C(r)$

Resulting constraint: $\Diamond (B(r) \land \neg \Diamond B(r)) \Rightarrow \neg C(r)$
ALLOW MULTIPLE DIAGRAMS

Model divisions’ business processes as individual constraint diagrams

- **R₁: Awaiting Approval**
  - Approved = false

- **R₁: Under Review**
  - Approved = false

- **R₁: Awaiting Payment**
  - Approved = true & Paid = false

- **R₁: Old**
  - years(now-PaidDate) > 7 | (years(now-OrderDate) > 3 & Paid=false)

- **R₁: Paid**
  - Paid = true

- **Records Retention**

- **Financial Integrity**
Convert to temporal logic

\[ \text{New}(x) \Rightarrow \text{AwaitingApp}(x) \]

\[ \neg \bullet \text{AwaitingPay}(x) \land \text{AwaitingPay}(x) \Rightarrow \bullet \text{AwaitingApp}(x) \land \text{UserDept} = \text{Sales} \]

\[ \neg \bullet \text{Paid}(x) \land \text{Paid}(x) \Rightarrow \bullet \text{AwaitingPay}(x) \land \text{UserDept} = \text{Finance} \]

\[ \text{Paid}(x) \land \text{years}(\text{now} - \text{PaidDate}(x)) \leq 7 \Rightarrow \text{Retain}(x) \]
**STATES OF AN OBJECT**

States = \{ AwaitingApproval, AwaitingPayment, Paid \}

State space = \{(0,0,0),(0,0,1),(0,1,0),\ldots,(1,1,1)\}

- Some configurations are not satisfiable
- Others may be subject to policy constraints

**Property verification**: use model checker to prove consistency, etc.
Complete, *temporally ordered* list of state configurations per object

**Object \( O_1 \) history**

<table>
<thead>
<tr>
<th>Time</th>
<th>Awaiting Approval</th>
<th>Awaiting Payment</th>
<th>Paid</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( t_{new} )</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Enforcement:** every time an object is modified, look back at history and check each constraint

- Experiments show that constraint checking overhead is insignificant.
RECORDS DISPOSITION POLICIES

Objective: to destroy (or anonymize) records or to archive them once certain conditions are met

- Condition typically triggered by advance of time
- Requires database to take action
- Cf. Other policies disallow classes of updates

Disposition states similar to other DB states

- But records in disposition states must be archived, deleted, or altered

Disposition transitions can be scheduled

- Cf. Other policy transitions must be continuously enforced
- When to invoke transition balances efficiency vs. risk
- Is each disposition transition effective?
- Do combined disposition transitions always terminate?
CONCLUSIONS

Emphasizes data, but models processes and user interactions

Policy designer only needs to list the “states of interest”

- By specifying the conditions that the object must satisfy to be in those states
- Each policy designer in the organization can list his/her own states of interest

Policy restricts paths that objects can (or should) traverse

- Constraint diagram (the model)
  - Some paths must always be taken, some must never be traversed, and others can be conditionally traversed

Database system enforces constraints automatically

- Every update transaction triggers check of objects’ state configuration history
- Disposition transitions checked and enforced as per schedule
Atalluah & Tompa: *Proc. CIKM’08, RELAW’09, POLICY’11, PLVDB’11*


**Thank you**