An Authorization Model for Temporal Data

Avigdor Gal
Rutgers University
(in collaboration with Vijay Atluri)

Introduction: temporal data

- Temporal data:
  - Transaction time.
  - Valid time.
- Applications:
  - Stream data, e.g., stock market information.
  - Data warehouses.
  - Spatiotemporal databases.
Introduction: authorizations

- Authorizations:
  - Subject
  - Objects
  - Mappings

- Applications:
  - Database access
  - Web-based data access

Introduction: temporal authorizations

- Temporal data authorizations:
  - Temporal authorizations [BERTINO95]
  - Temporal objects

- Applications:
  - Temporal based access to financial data
  - Limited access to books in digital libraries
  - Limited access to temporally aggregated data
Avigdor Gal is a faculty member at the Department of MSIS at Rutgers University. He received his D.Sc. degree from the Technion-Israel Institute of Technology in 1995 in the area of temporal active databases. He has published more than 30 papers in journals (e.g. IEEE Transactions on Knowledge and Data Engineering), books (Temporal Databases: Research and Practice) and conferences (e.g. ER'95, CoopIS'98) on the topics of information systems architectures, active databases and temporal databases. Together with Dr. John Mylopoulos, Avigdor has chaired the “Distributed Heterogeneous Information Services” workshop at HICSS'98 and he was the guest editor of a special issue by the same name in the International Journal of Cooperative Information Systems. Also, he was the General co-Chair of CoopIS’2000. Avigdor has consulted in the area of eCommerce and is a member of the ACM and the IEEE computer society.

Thanks

- Vijay Atluri
- Su Kong
- George Bertele
- Digant Modha
A temporal data model

- A temporal domain \((T, \leq)\)
- Discrete model of time, \(T \subseteq \mathbb{N}\):
  - Time point:
    - 5minutes
  - Time interval \([t_a, t_c)\)
- Symbolic time interval vs. actual time interval:
  - UC
  - Now
A temporal data model

- Temporal dimension:
  - Transaction time (tx)
  - Valid time (tv)
  - Request time (treq)

- Temporal logic, based on [SHOHAM88]:
  - Wff: \((tx+5\text{minutes}\leq treq \land treq \leq tc+5\text{minutes})\)

- Temporal objects:
  - Property domain
  - Temporal domains
  - Class instances, states, and state-elements.

Stock market example

MostActive(\textbf{Market, Symbol, LastSale, LastTradeSize, 2ndActiveMarket})

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>se2</td>
<td>100, [July30:1999:14:03, UC), July30:1999:14:04</td>
</tr>
</tbody>
</table>

Table 1: The LastTradeSize of the Dell share at NYSE
Discriminating temporal dimensions

- A set of temporal dimensions is termed *discriminating temporal dimensions* if it generates a partial order among the state-elements of a state.
- Determining actual time intervals from symbolic time intervals using discriminating temporal dimensions:
  - \( s_1,t_1=[July30:1999:13:57, July30:1999:14:03) \)
  - \( s_2,t_2=[July30:1999:14:03, July30:1999:15:55) \)

TDAM: Temporal Data Authorization Model

- Privileged groups
- Security objects
- Privilege modes

**Definition 8 (Authorization)** An authorization \( a \) is a quintuple \( (pg, o, m, sign, \tau) \), where \( pg \) is a privilege group, \( o \) is an object, \( m \) is a privilege mode, \( sign \in \{+, -\} \) indicating access or denial, and \( \tau \) is a well-formed formula.
Authorizations in the stock exchange example

Table 2: The LastTradeSize of the Dell share at NYSE — a simplified version

\[
\begin{array}{l|ccc|c}
  & s_{e1} & 600, & 57, UC), & 58 \\
  & s_{e2} & 100, & 63, UC), & 64 \\
  & s_{e3} & 500, & 175, UC), & 176 \\
\end{array}
\]

\[t^0\]
\[t^x\]

\[T_{req} = 63\]

Authorization example in a digital library setting

- Self-destructing books after 21 days.
- For example: [http://www.authentica.com](http://www.authentica.com)
- \((\text{borrower,book,read, } t_r + 21 \text{ days} \leq t_{req})\)
Access Control

- A point access request is a triple \((s, o, m)\).
- An interval access request is a quadruple \((d, s, o, m)\).
- Each access request is timestamped \((t_{req})\).
- A subject can be granted access to a state-element \(se\) under \(a=(pg, o, m, sign, \tau)\) only if

\[
S \models VA(se.ts, se.te, se.tx, t_{req}, se.val)\tau
\]

Input: \((s, o, m)\) or \((d, s, o, m)\), \(t_{req}\), and \(\{se\} \in o\)

Output: \(\{(s, se, m, \delta)\}\)

Procedure:

"presence of authorization */
- If there exists an authorization \(a=(pg, o, m, +, \tau)\) \(\in \text{AE}\)
  such that \(s \in pg\), then
  \(\tau' = null\)

"formula assessment */
- If there exists an authorization \(a=(pg, o, m, -, \tau)\) \(\in \text{AB}\)
  such that \(s \in pg\), then
  \(\tau' = \tau' \land \neg \tau\)

"state-elements selection */
- If \(d = \text{null}\) then \(d = 1\)
  repeat
  \(SE = \{se \in o | \tau_2 \land \tau_4 \land \tau_2 \land \tau_4\}\)
  /* version switching */
  foreach \(se_i \in SE\)
  \(t_i = \text{bind}(se_i, t_s, se_i, t_s, se_i, t_s, se_i, t_s) \cap [t_{req}, t_{req} + d]\)
  If \(t_i \neq \emptyset\) then output \((s, se_i, m, t_i)\)
  end For
  until \(now \geq t_{req} + d\)
  end If
Unification

- Authorization specifications as an incremental process.
- Let \( \{a_1, a_2, \ldots, a_n\} \) be \( n \) authorizations, such that for any \( i < j \):
  - \( p_{a_i} = p_{a_j} \)
  - \( o_{a_i} = o_{a_j} \)
  - \( m_{a_i} = m_{a_j} \)
  - \( \text{Sign}_{a_i} = \text{Sign}_{a_j} \)
- The unification of \( \{a_1, a_2, \ldots, a_n\} \) results in a single authorization \( a \):
  \[
  a = \begin{cases} 
  (p_{a_{1:}}, o_{a_{1:}}, m_{a_{1:}}, \text{sign}_{a_{1:}}, \bigvee_{1 \leq i \leq n} t_i) & \text{sign}_{a_i} = + \\
  (p_{a_{1:}}, o_{a_{1:}}, m_{a_{1:}}, \text{sign}_{a_{1:}}, \bigwedge_{1 \leq i \leq n} t_i) & \text{sign}_{a_i} = -
  \end{cases}
  \]

Why not unify positive and negative authorizations?

Hierarchies

- Privilege modes hierarchy.
- Privileged groups hierarchy.
- Walking up and down the hierarchy:
  \[
  a_1 = (p, o, \text{read}, +, t_x + 10\text{minutes} \leq t_{req})
  \]
  \[
  a_2 = (p, o, \text{write}, +, t_x + 5\text{minutes} \leq t_{req})
  \]

The implication of \( a_2 \) on \( a_1 \) is that

\[
\tau_{a_1} = (\tau_{a_1} \vee \tau_{a_2}) \quad (1)
\]

\[
= (t_x + 10\text{minutes} \leq t_{req} \vee t_x + 5\text{minutes} \leq t_{req})
\]

\[
= t_x + 5\text{minutes} \leq t_{req} \quad (2)
\]
Segmentation

- \( \tau \) is separated into three conditions, on \( \tau_s, \tau_e, \) and \( \tau_x \).
- For example,
  - \( \tau = t_x + 5 \text{ minutes} \leq t_{req} \)
  - \( \tau = t_x \leq t_{req} - 5 \text{ minutes} \)
- Improves efficiency by relying on the temporal database natural indexing scheme.
TDAM: The demo

- http://business.rutgers.edu:4610/sign_in.html

Derived data

- New applications:
  - Information portals
  - On-line portfolio management
  - Digital libraries
  - Virtual retailing

- Characteristics:
  - Temporal data
  - Derived data:
    - Replications (mirror sites)
    - Materialized views (data warehouses)
Derivation rules

- Consider an international organization with three data warehouses, at the US, Italy and Germany:

\[ \text{InventoryLevel} = \text{US.InventoryLevel} + \text{GR.InventoryLevel} + \text{IT.InventoryLevel} \]

\[ \langle \text{IT.PriceBase}, \text{date}(t+1), \text{date}(t+2) \rangle = \langle \text{IT.US2ITConversion}, \text{date}(t), \text{date}(t+1) \rangle \times \langle \text{US.PriceBase}, \text{date}(t), \text{date}(t+1) \rangle \]

What’s wrong with derived data?

\( (1stLevelManagers, \text{US.PriceBase}, \text{update}, \langle t, t_{req} + 1 \text{year}, t_{req} \leq t_e \rangle) \)

\( (2ndLevelManagers, \text{IT.PriceBase}, \text{read}, \langle t, t_{req}, t_{req} \leq t_e \rangle) \)

- John updates the database with the daily $US to ITL conversion rate:
  \( (\text{John, IT.US2ITConversion}, \text{write}, \langle t, t_{req}, t_{req} \leq t_e \rangle) \)

- Automatic derivation of \text{IT.PriceBase} in response to the update of \text{IT.US2ITConversion}:
  \( (\text{John, f(US.PriceBase,US2ITConversion, execute}, \langle t, t_{req}, t_{req} \leq t_e \rangle) \)

\( (\text{John, IT.PriceBase}, \text{write}, \langle t, t_{req}, t_{req} \leq t_e \rangle) \)
What’s wrong with derived data?

- Access mode lattice include:
  - execute → write
  - write → read
  
  \( (\text{John, IT.PriceBase, read, (t_s \leq t_{req} \land t_{req} \leq t_e)}) \)

- From transitivity of the lattice →:
  \( (\text{John, f(US.PriceBase, US2ITConversion), read, (t_s \leq t_{req} \land t_{req} \leq t_e)}) \)

- \( \Rightarrow \text{John can compute US.PriceBase using the inverse mapping:} \)

  \( \langle \text{US.PriceBase}, [\text{date}(t), \text{date}(t+1)] \rangle = \frac{\langle \text{IT.US2ITConversion}, [\text{date}(t), \text{date}(t+1)] \rangle}{\langle \text{IT.PriceBase}, [\text{date}(t+1), \text{date}(t+2)] \rangle} \)

  \( \Rightarrow (\text{John, US.PriceBase, read, (t_s - 1440 \leq t_{req} \land t_{req} \leq t_e - 1440)}) \)