SWECCA for Data Warehouse Maintenance

Aparna Varde and Elke Rundensteiner
Database Systems Research Group,
Department of Computer Science,
Worcester Polytechnic Institute.
Data Warehousing System

Data Warehouse

Mediator

Student DB

Course DB

Professor DB

♦ DW: User-Customized Repository of Integrated Information
♦ Trend: Larger systems, many IS, multiple relations per IS
♦ Send up-to-date information to user, efficiently
Incremental View Maintenance

When DU occurs, rewrite only affected tuples
Done as and when IS data changes
Problem: Concurrency Conflict

- $\Delta R_1$ occurs, its effect is being computed
  - $\Delta R_2$ Happened Concurrently.
  - Conflict: $\Delta R_1 \bowtie \Delta R_2 \bowtie R_3$

\[
\begin{align*}
V &= R_1 \bowtie R_2 \bowtie R_3 \\
MQR_1 &= \Delta R_1 \bowtie (R_2 + \Delta R_2) \bowtie R_3 \\
MQ_1 &= \Delta R_1 \bowtie R_2 \bowtie R_3
\end{align*}
\]
Goal: To solve concurrency conflicts during Incremental VM in multi-IS multi-R DW

---

SCI-2002

Research Issue

Single Source VM Approach: ECA, CCA

Multi Source VM Approach: Strobe, SWEEP
CCA: Complete Compensating Algorithm
(Proposed at Stanford University, 1995)

All Rs in same IS
- After $MQR_1$ arrives….

For $\Delta R_1$

$$CQ_{1,2} = -(\Delta R_1 \cdot \Delta R_2 \cdot \Delta R_3)$$
$$CQ_{1,3} = - (\Delta R_1 \cdot \Delta R_2 \cdot \Delta R_3)$$
$$+ (\Delta R_1 \cdot \Delta R_2 \cdot \Delta R_3)$$

............

$V = V + MQR_1 + \text{all CQRs}$

- Similarly for all $\Delta R_s$

✓ Complete Consistency

✗ All Rs in same IS
SWEEP: Local Compensation Approach
(Proposed at University of California, Santa Barbara, 1997)

- Multiple Sources
- No remote CQs
- No infinite wait

△R occurs, SWEEP each IS to get affected tuples
- Using values in UMQ, locally calculate ∆R₁ ⊙ ∆R₂

* Only one R per IS… What if multiple Rs per IS…?
Initial Proposal: Simplistic Approach

- Treat each relation as a separate source

✘ Too much overhead, highly inefficient !!
Issue: Where SWEEP has a problem

♦ Local Compensation Not Working

\[ MQ_1 = \triangle R_1 \times (R_2 \times R_3) \]

\[ MQR_1 = \triangle R_1 \times (R_2 + \triangle R_2) \times R_3 \]

Can’t Calculate from Queue:

\[ \triangle R_1 \times \triangle R_2 \times R_3 \]

Can Calculate from Queue:

\[ \triangle R_1 \times \triangle R_2 \]

\[ V = R_1 \times R_2 \times R_3 \]
Potential Solution???

SWEEP Cannot Calculate

$\Delta R_1 \rightarrow \Delta R_2 \rightarrow R_3$

from UMQ

$\Delta R_1$

Wrapper ($\Delta R_2 \dashv \dashv R_3$)

$\Delta R_2$

$\Delta R_3$

$\Delta R_2$

MEDIATOR cannot correct this

Relational Concurrency Conflict

IS$_1$

R$_1$

IS$_2$

R$_2$

R$_3$
Motivation for SWECCA

♦ SWEEP
  – Works only for single R per IS, multiple IS

♦ CCA
  – Works only for single IS, multiple Rs

♦ Motivation
  – If multiple IS, multiple Rs per IS
    • use SWEEP at mediator
    • and CCA at wrapper
  – Redesign as needed
DW with SWEEP and CCA

- Mediator (SWEEP)
  - Wrapper (CCA) LQueue
    - IS1
    - R11, R12
  - Wrapper (CCA) LQueue
    - IS2
    - R21
  - Wrapper (CCA) LQueue
    - IS3
    - R31, R32, R33

DSRG, WPI
CCA at the Wrapper Level

Mediator

GQueue

\(\triangle IS_2\)

\(\triangle IS_1\)

\(\triangle IS_3\)

CCA Wrapper

LQueue

\(\triangle R_1\)

\(\triangle R_2\)

\(\triangle R_3\)

\(LMQ_1\)

\(LMQR_1\)

\(CQ_{1,2}\)

\(CQ_{1,3}\)

IS_2

R_1

R_2

R_3

DSRG, WPI
Global Query Processing???

- CCA: VM algorithm
- Handle DUs only
- For this, it *sends* Query to IS

- What if it *receives* a Query???
- *Query processing functionality needed*
SWECCA: At Mediator

Mediator

GQueue

△IS₁

GMQ

△IS₂ (conflict)

Conc

GMQR

△IS₃ (no conflict)

IS₁ Wrapper
(R₁₁)

IS₂ Wrapper
(R₂₁, R₂₂)

IS₃ Wrapper
(R₃₁, R₃₂, R₃₃)
**SWECCA Example: At Wrapper**

Mediator: \( V = \Pi_w (IS_1 \bowtie IS_2 \bowtie IS_3) \)

Initial \( V = \text{null} \)

\[ \text{GMQ} = \Pi_w (\triangle IS_1 \text{ join } IS_2) \]

\[ \text{GMQR} = \text{GMQR} + \text{All CQRs} = \text{null} \]

\[ \text{Wrapper2} \]

\[ CQ_{22} = - \Pi_w (R_{21} \text{ join } 2,5 \text{ join } R_{23}) \]

\[ IS_2 = \Pi_w (R_{21} \bowtie R_{22} \bowtie R_{23}) \]

\[ \begin{array}{c|c|c|c}
  W,X & X,Y & Y,Z \\
  \hline
  \text{Ini} & [1,2] & & \\
  \triangle R_{21} & +[4,2] & & \\
  \triangle R_{22} & & +[2,5] & \\
  \triangle R_{23} & & & +[5,3] \\
\end{array} \]
V = \Pi_w (IS_1 \bowtie IS_2 \bowtie IS_3)

\Delta IS_1 = - [4]

\Delta IS_2 = \text{null}

\Delta IS_3

\text{GMQ} = \Pi_w (\Delta IS_1 \text{ join IS}_2)

\text{GMQR} = \text{null}

Wrapper1

Wrapper2

Wrapper3

IS_1

IS_2

IS_3

R_{11}

R_{21} \quad R_{22} \quad R_{23}

R_{31} \quad R_{32}
Related Work

♦ RV: Recomputation of View
  – Rewrite all tuples, not only affected ones, highly inefficient if done for every update

♦ SM: Self-Maintenance
  – Warehouse maintains copies of source relations for maintenance, huge storage at warehouse

♦ MRE: Multi-Relation Encapsulation
  – Store ISV at each IS wrapper $\Leftrightarrow$ V at Mediator, huge storage at wrappers
Conclusions

♦ SWECCA has
  – Flexibility of semi-autonomous sources
  – Advantage of software re-use
  – Simplicity of architecture
  – Storage efficiency

♦ SWECCA enhancement
  – More generic system MEDWRAP
  – Works for any 2 VM algorithms
  – MEDWRAP being implemented at DSRG, WPI