Near Real-time Data Warehousing with Multi-stage Trickle & Flip

Jānis Zuters,
University of Latvia,
BIR 2011,
October 8, 2011
Data Flow In a Data Warehouse

- Data Source
- ETL
- Data Warehouse
- OLAP

- Near Real-time
- CDC
- Data Loading
Near Real-time Refreshment

Data Source -> Microbatch ETL -> Data Warehouse

Frequent data loading (e.g., hourly)

Data loading conflicts

OLAP
The ‘Trickle & Flip’ Methodology

Data Source → ETL → Staging Tables → Copy of Staging Tables (flip)

Copy of Staging Tables → OLAP

1h copy
‘Trickle & Flip’: The Algorithm

ALGORITHM trickle_and_flip_refresh (R)

\[ DW \] — data warehouse
\[ D1, D2 \] — staging partitions
with the same data format as \[ DW \]
\[ R \] — refreshment rate (e.g., 1 hour)
\[ D1 \] is being fed from the source

BEGIN

\textbf{Do Every} \[ R \] \% e.g., every hour

\textbf{Copy} \[ D1 \] to \[ D2 \]
\% \[ DW \] should not be locked by querying
\textbf{Flip} \[ D2 \] and \[ DW \]
‘Trickle & Flip’: Demonstration

• 3 complete copies of data!
'Trickle & Flip’ With Real-time Partition

Data Source ➔ ETL ➔ Staging Tables ➔ Copy of Staging Tables ➔ Flip ➔ Data Warehouse

- 3 complete copies of real-time data!
Why ‘Trickle & Flip’ is Better Than “Simple” Near Real-time?

- Data warehouse suffers from data loading to a significantly less extent (flipping is very fast)
What Are The Issues of the “Pure” ‘Trickle & Flip’ Approach?

- Frequent copying of large amount of data
- If refreshment cycle times become very frequent: querying over real-time data becomes inconvenient
Assumptions to Add More Stages to ‘Trickle & Flip’

• Adding data to a **smaller table** (i.e., with less data) is faster;

• **Updating** last changes to a table is faster than making full copy of the last version
Multi-stage ‘Trickle & Flip’

Data Source

Staging
Tables 0

add

ETL

5 min.

add?

Staging
Tables 1

Real-time
Partition 1

add?

Staging
Tables 2

Real-time
Partition 2

add?

1 hour

OLAP

Static
Data

Data Warehouse
Multi-stage ‘Trickle & Flip’: The Algorithm (For a Single Stage)

**ALGORITHM** `multiple_trickle_and_flip_refresh`

\[ (R1, M, H`, H) \]

- \( H \) - real-time partition for the current hour
- \( M, H` \) - staging partitions
- \( R1 \) - refreshment rate (e.g., 5 minutes)
- \( M \) is being continuously fed from the source

**BEGIN**

**Do Every** \( R1 \) \% e.g., every 5 minutes

- Add \( M \) to \( H` \)
- Empty \( M \)
- **If** \( H \) is available \% not locked by querying
  - Add \( H` \) to \( H \)
  - Empty \( H` \)
Multi-stage ‘Trickle & Flip’: Demonstration

Data Source

ETL

Staging Tables 0

Staging Tables 1

Real-time Partition 1

Real-time Partition 2

Staging Tables 2

Static Data

OLAP
The Summary of Multi-stage ‘Trickle & Flip’

- Total amount of data copying is reduced;
- Collisions between data loading and querying activities have been reduced;
- More advanced querying system is required to fully benefit from the approach.
Thank You!

Jānis Zuters,
University of Latvia