Overview

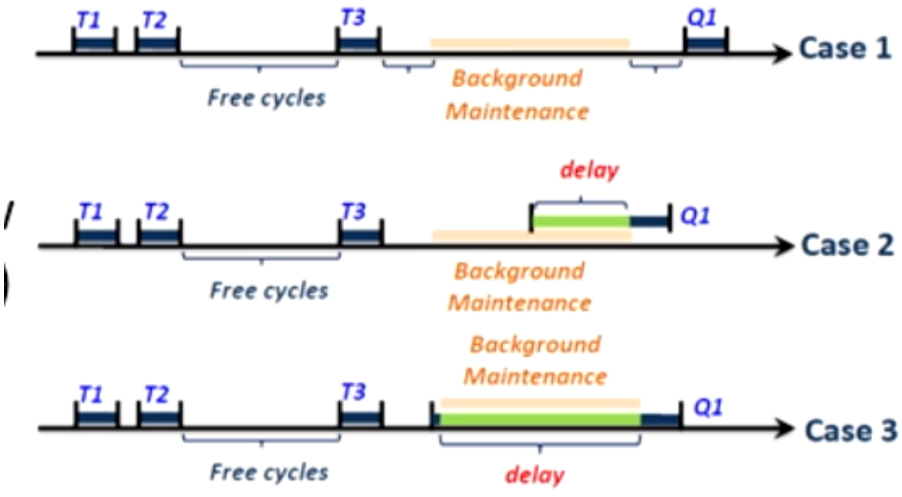
- precompute often used queries, store and reuse them multiple times
- frequently re-occuring queries (views) need to be identified

Lifecycle

- select view
  - NP-hard
  - heuristic, greedy and approximate algorithms exist
- use view within other queries
  - use some parts of the view and use more operators on top of it
- maintain view whenever data is added/removed

View Maintenance

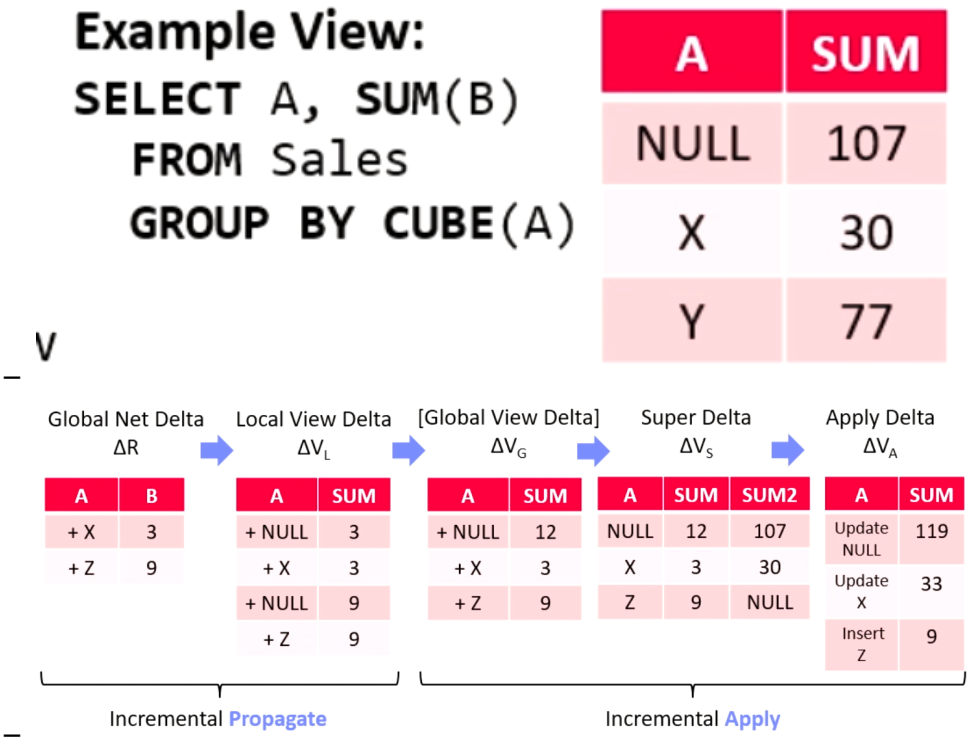
- eager maintenance
  - writer pays
  - update view whenever underlying data is changed
    - Immediate refresh: updates are directly handled (consistent view)
    - On Commit refresh: updates are forwarded at end of successful TXs
  -
- deferred maintenance
  - reader pays
  - update view on explicit user request
  - potentially inconsistent base tables and views
- lazy maintenance
  - asynd/reader pays
  - same guarantess as eager maintenance
  - defer maintenance until free cycles or view required



  -

How View Maintenance is done

- incremental maintenance
  - track changes in separate table - propagate
  - apply collected changes to view - apply



Materialized Views in PostgreSQL

- **View Selection**
  - **Manual definition** of materialized view only
  - With or without data
- **View Usage**
  - **Manual use** of view
  - No automatic query rewriting
- **View Maintenance**
  - **Manual (deferred) refresh**
  - Complete, no incremental maintenance
  - Note: Community work on IVM

[Yugo Nagata: Implementing Incremental View Maintenance on PostgreSQL, **PGConf 2018**], patch in 2019

```
CREATE MATERIALIZED VIEW TopScorer AS
  SELECT P.Name, Count(*)
    FROM Players P, Goals G
    WHERE P.Pid=G.Pid AND G.GOwn=FALSE
    GROUP BY P.Name
    ORDER BY Count(*) DESC
  WITH DATA;

REFRESH MATERIALIZED VIEW TopScorer;
```

Name	Count
James Rodríguez	6
Thomas Müller	5
Robin van Persie	4
Neymar	4
Lionel Messi	4
Arjen Robben	3

[[Database Performance Tuning]]