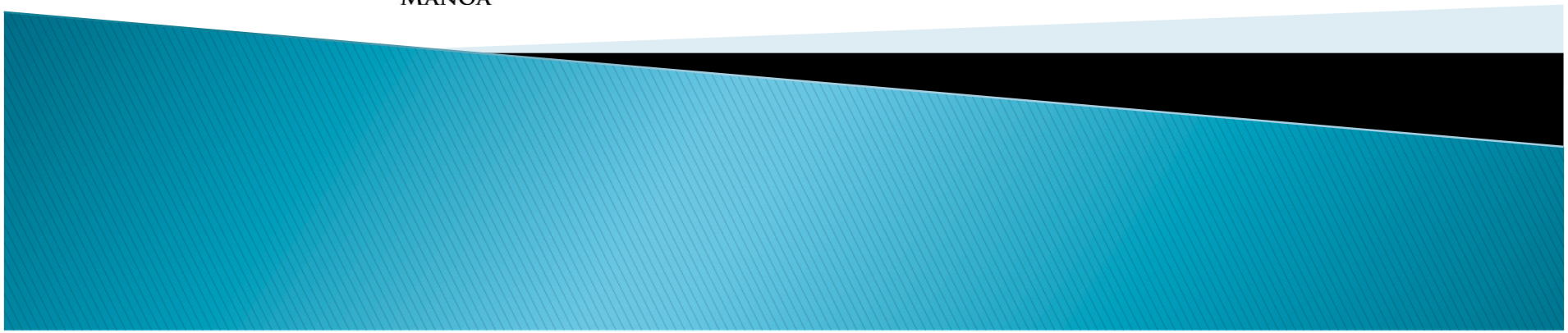
Optimizing Access Across Multiple Hierarchies in Data Warehouses

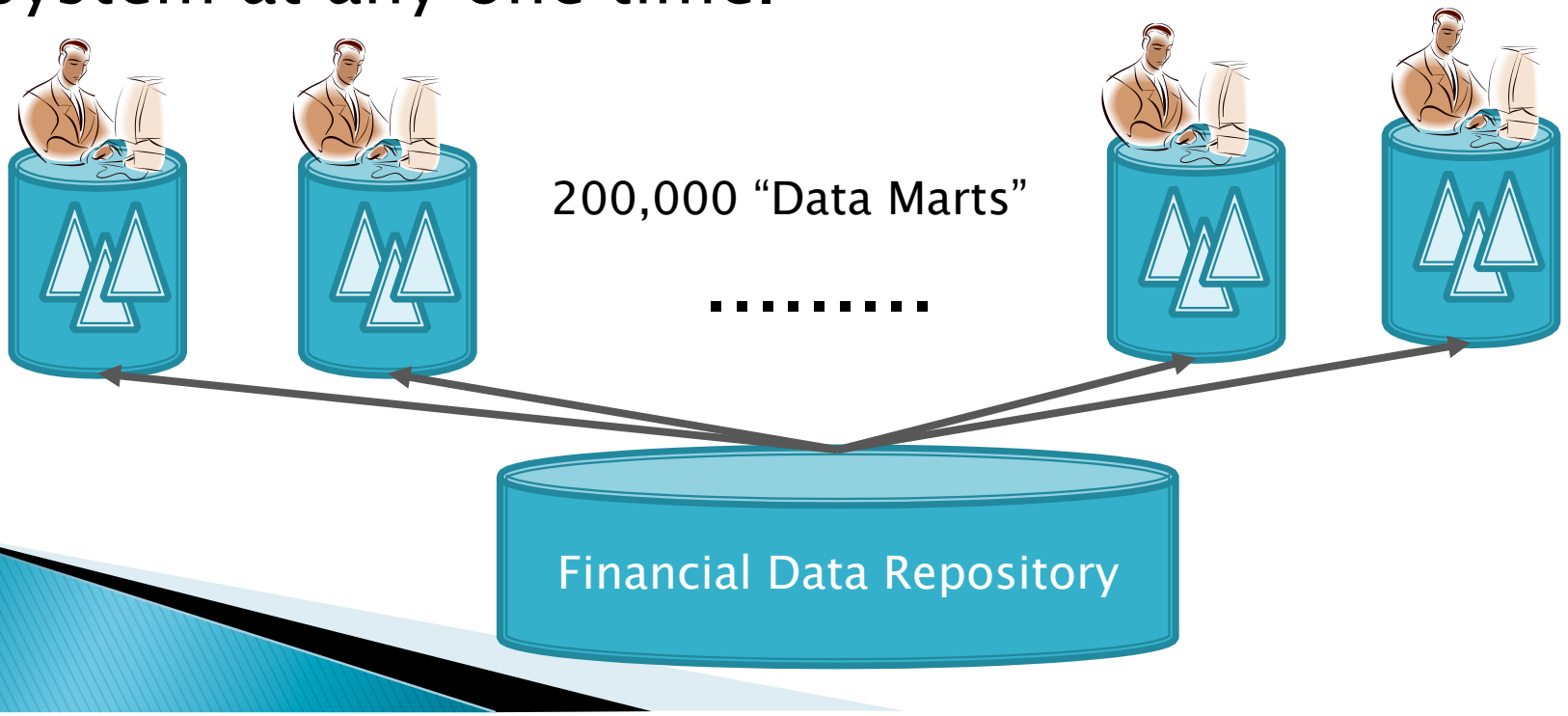
Lipyeow Lim
University of Hawai`i at Mānoa

Bishwaranhan Bhattacharjee
IBM Thomas J Watson Research Center

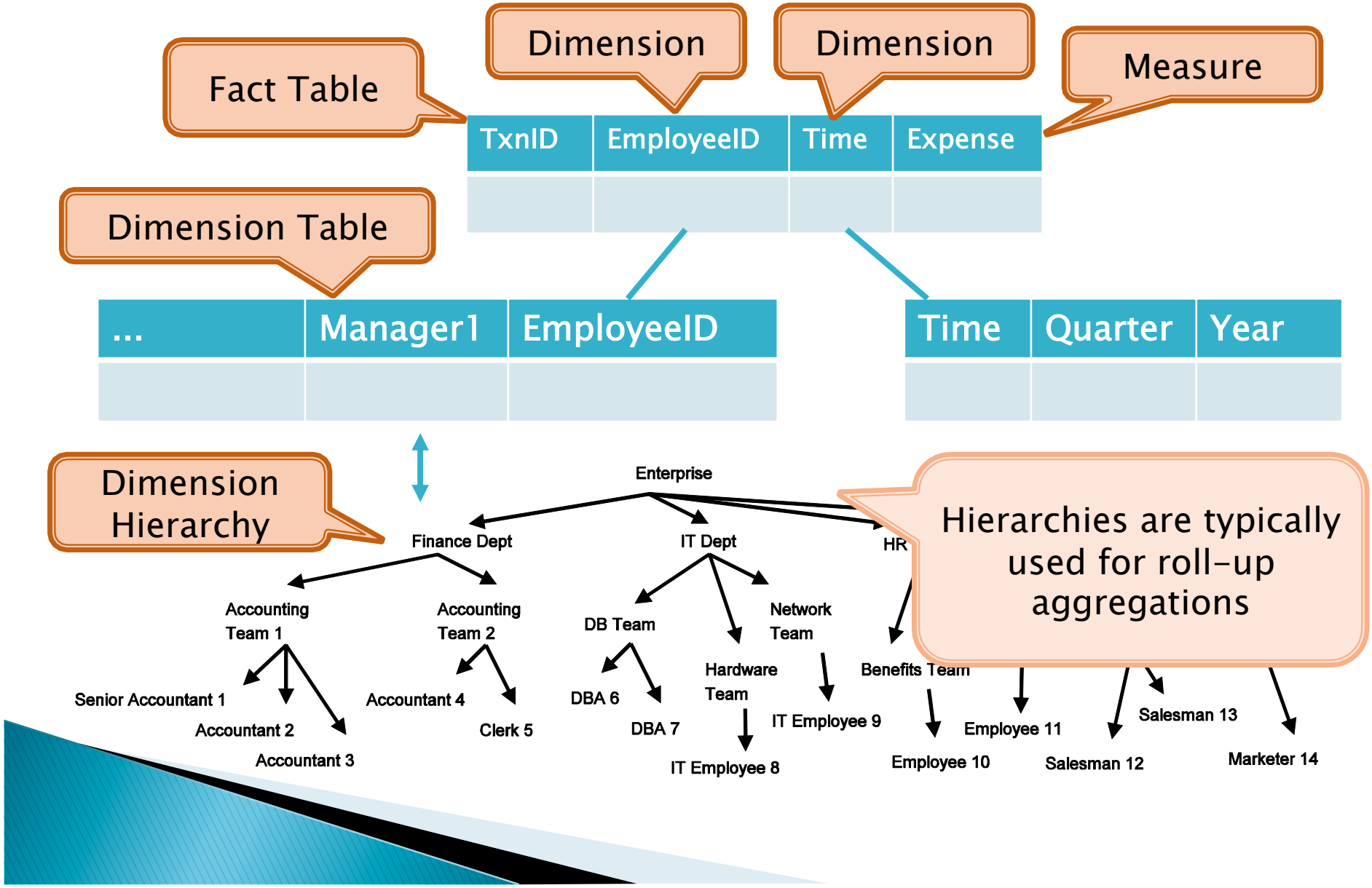


Background

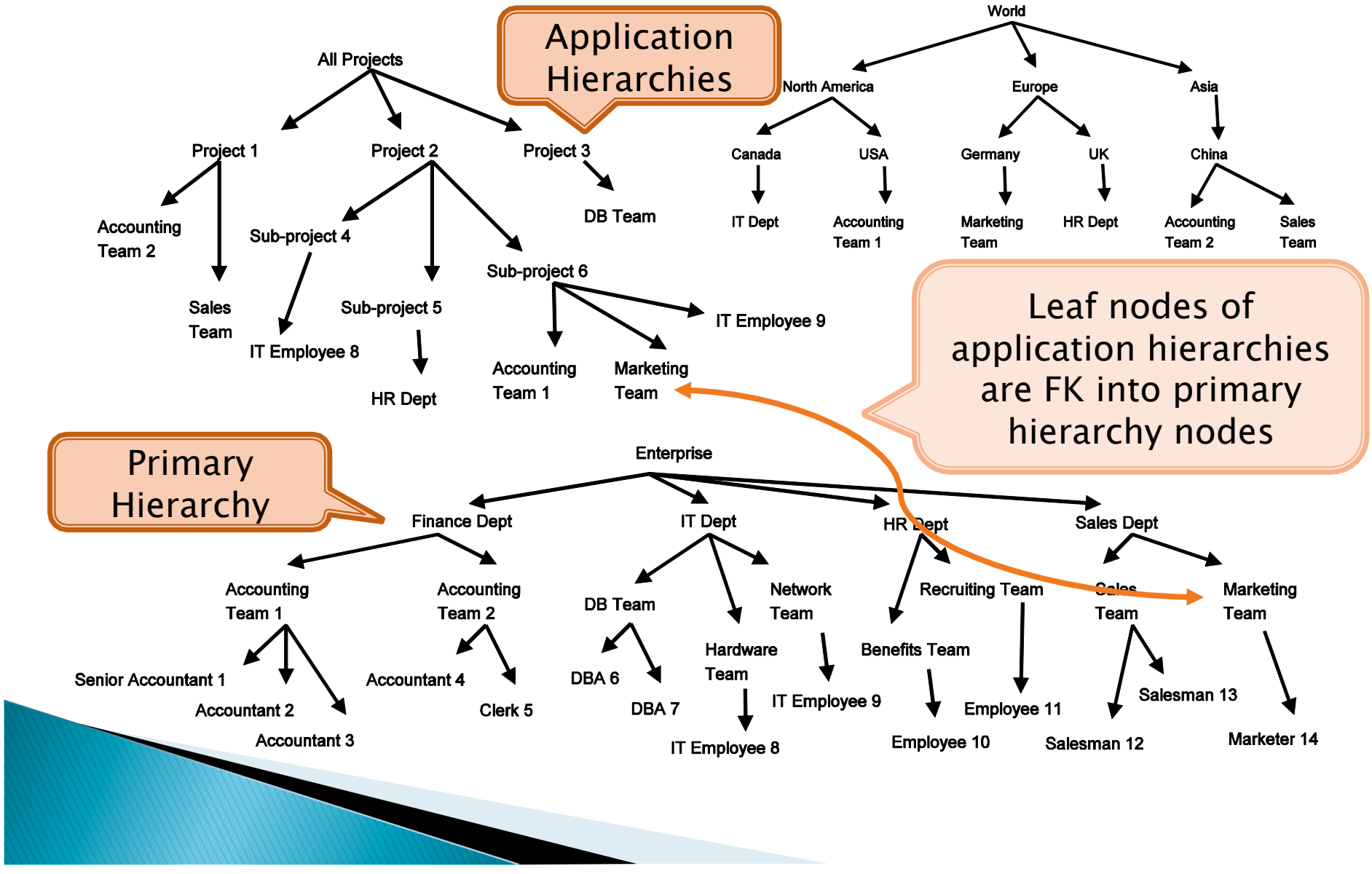
- A large US Bank with a financial data warehouse
- 200,000 business units defining hierarchies
- Dimension tables grew to 100 million rows
- At most 20 users (out of 1500) able to use the system at any one time.



Data Warehousing



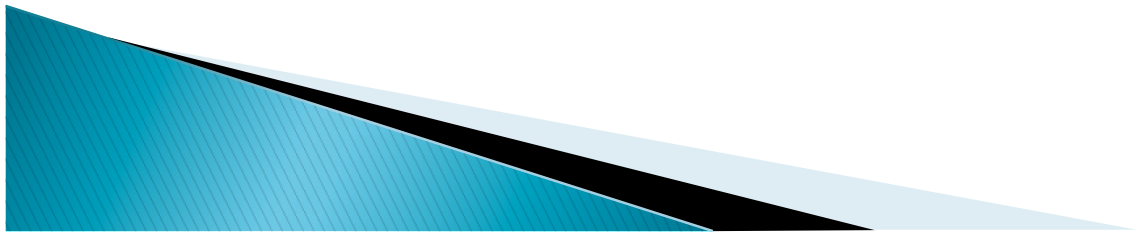
Application Hierarchies



Master Data Management ?

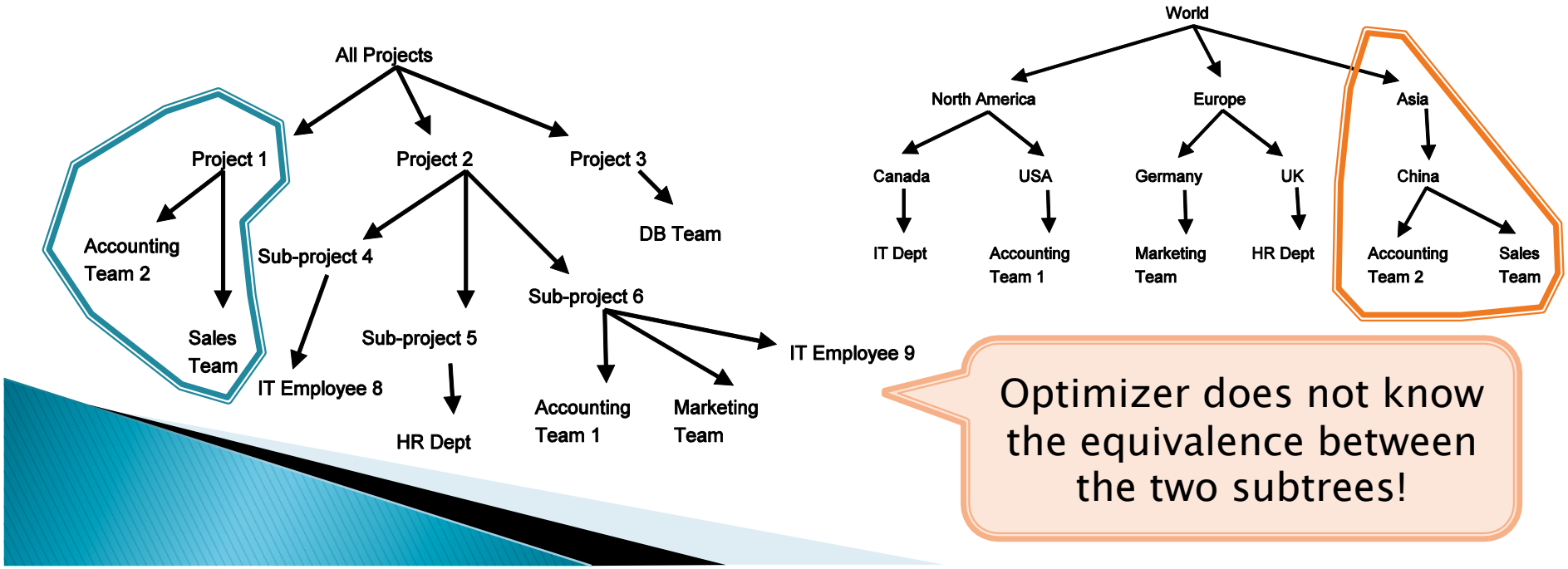
- If only Bank X had used MDM, there would not be an uncontrolled proliferation of application hierarchies ...but...

- What can be done to deal with the slow down caused by the large number of application hierarchies ?
  - Pre-compute aggregations on hierarchies
  - Cache and reuse previous aggregations

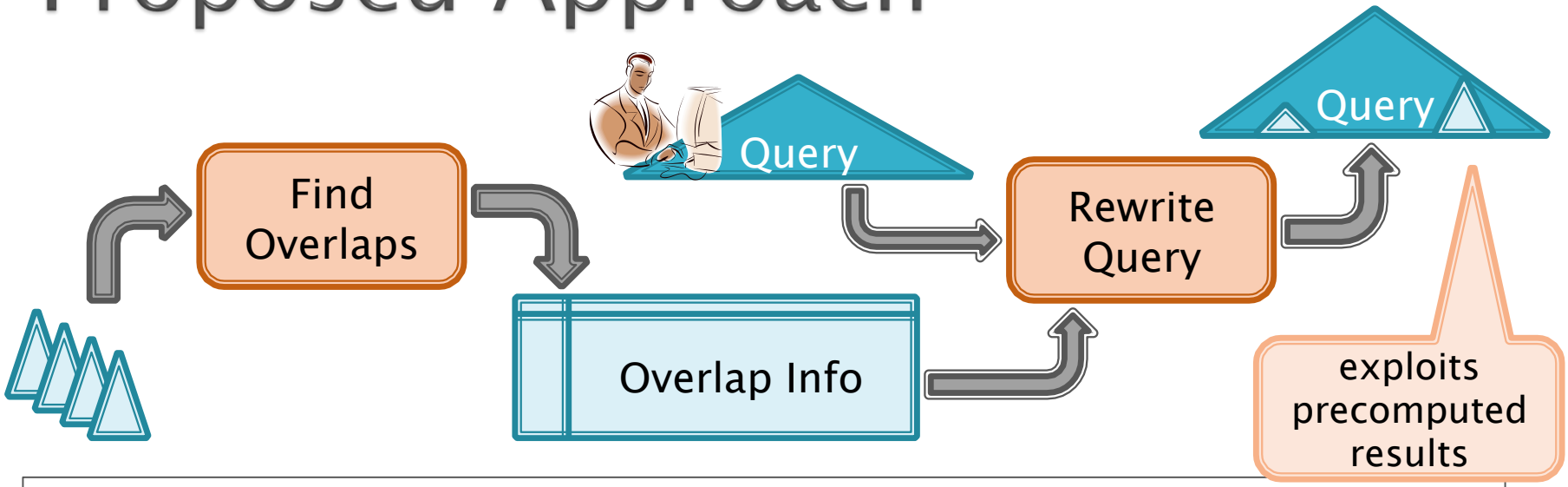


Exploiting Precomputed Aggregates

- Consider a query for an aggregation of "Asia"
- Suppose aggregation of "Project 1" precomputed
- Can the aggregate for "Project 1" be used to answer query for "Asia" ?



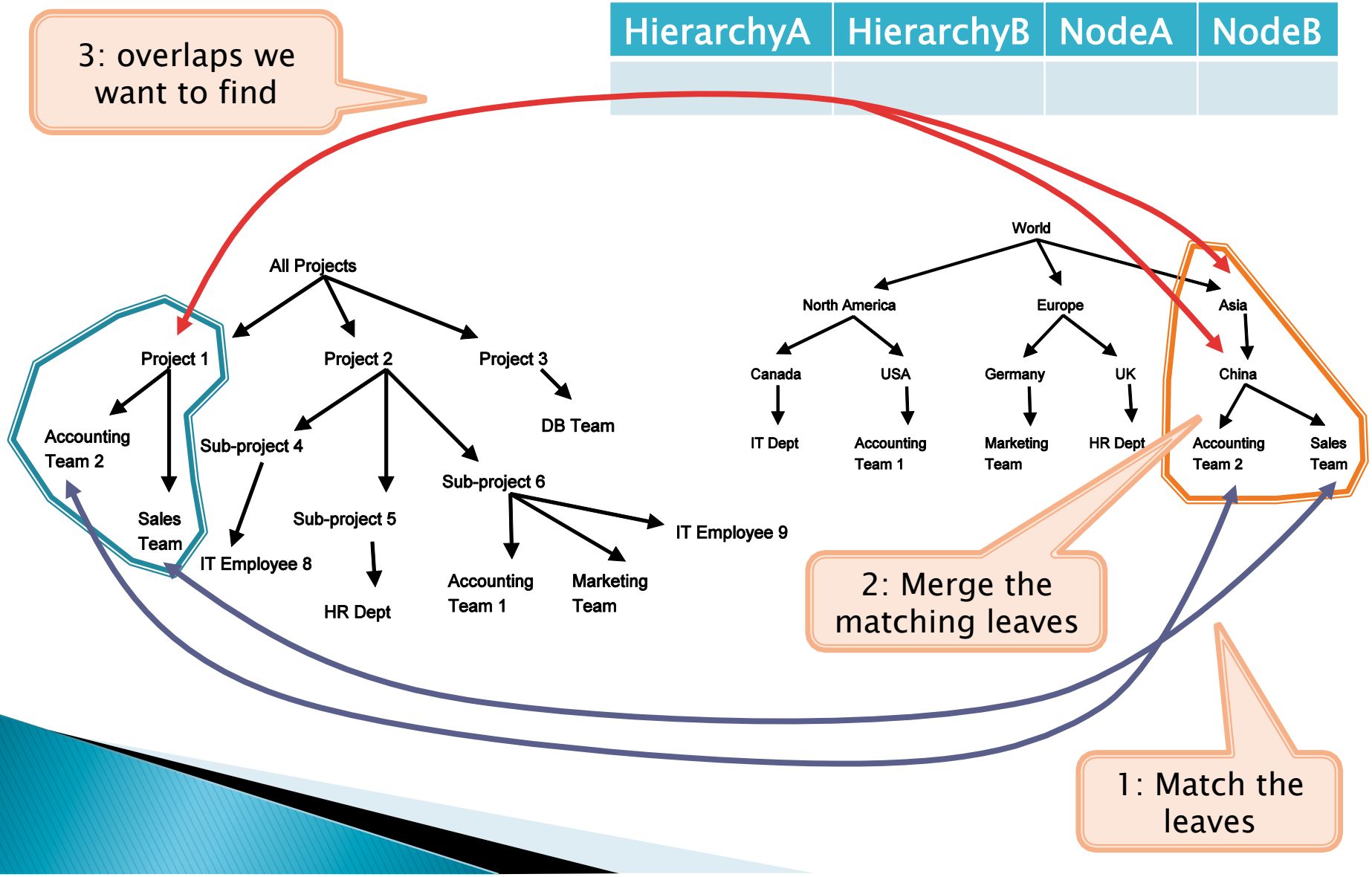
Proposed Approach



- Off-line Phase finds and stores overlaps
  - *Sub-tree isomorphism problem*
- On-line Phase rewrites queries using overlap information to exploit pre-computed results
  - *View containment problem*

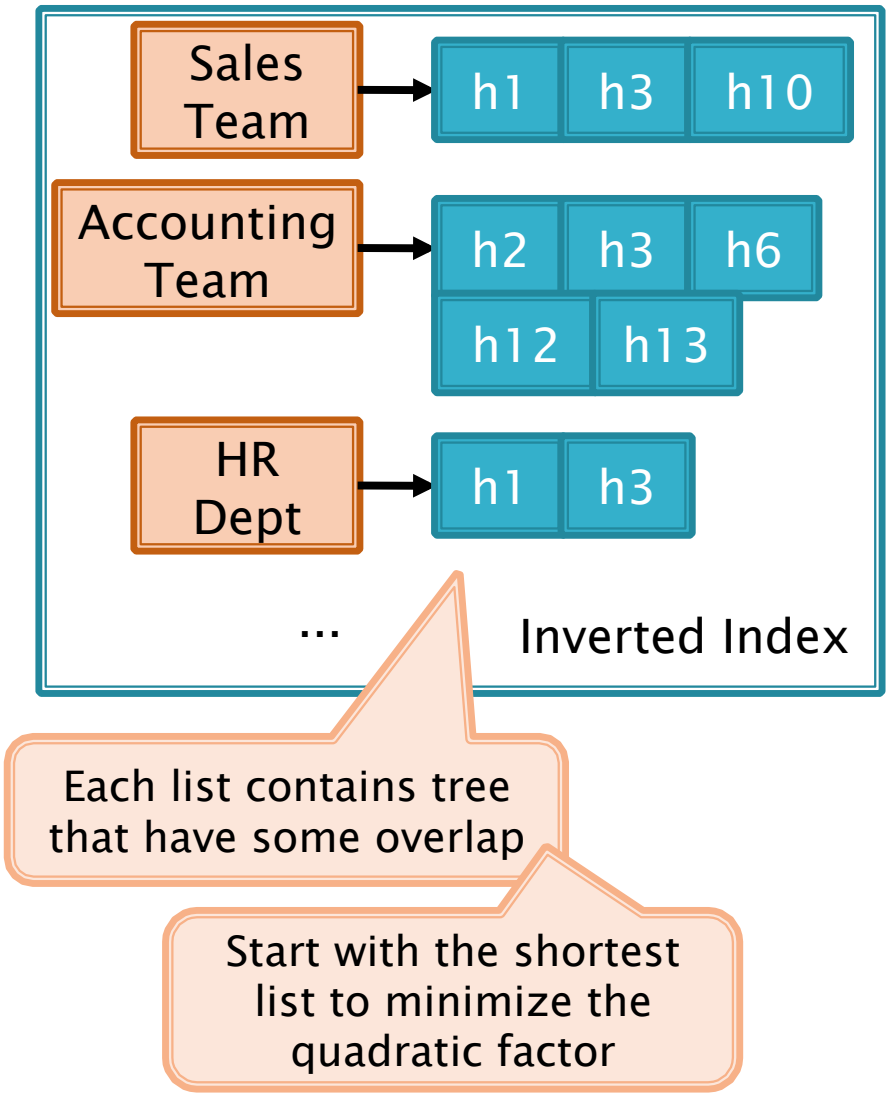
Intuition for Finding Overlaps

HierarchyA	HierarchyB	NodeA	NodeB

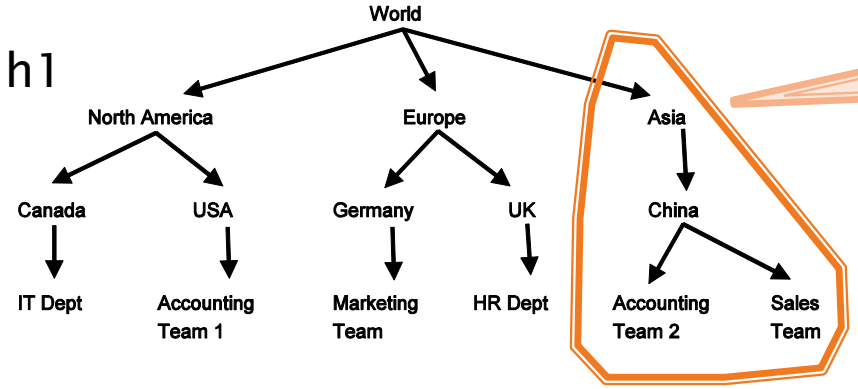


Finding Overlaps for Many Trees

- Given Trees { h1, h2, h3, … hn }
- Consider all pairs of trees
  - $O(n^2)$ – too expensive
- Use an inverted index
  - Construct an inverted index of leaf labels to tree IDs.
  - Eliminate all singleton inverted lists.
  - Starting from the smallest inverted list, consider all pairs.
  - Keep track of which pairs have been "done"



Rewriting Queries



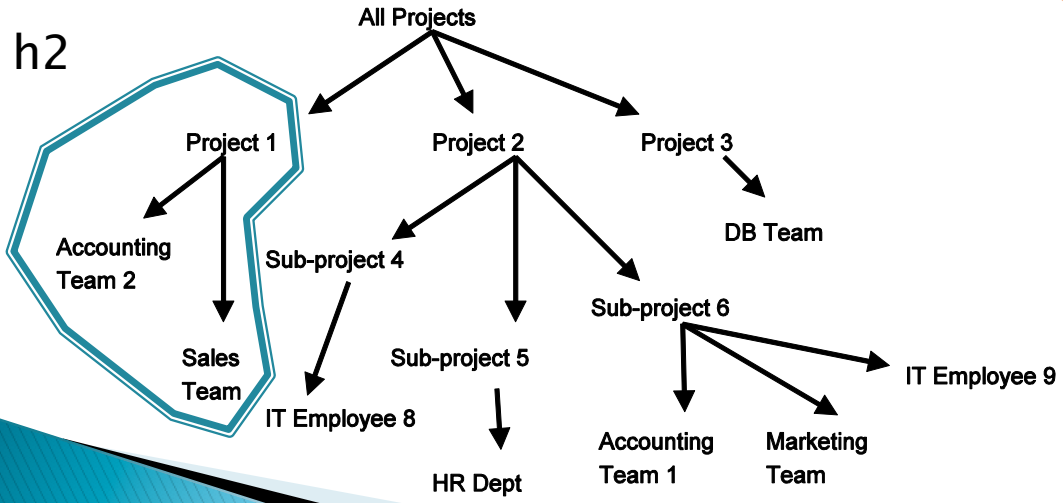
1: Find QN, the set of covering tree nodes

Query on h1: Accounting Team 2 + Sales Team

HierarchyA	HierarchyB	NodeA	NodeB

2: Find hierarchies that overlap with h1

3: Find set of alternate nodes that are equivalent to each covering tree node

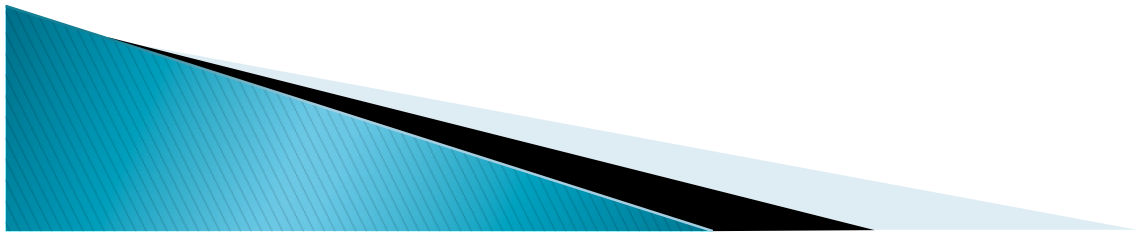


Experiments

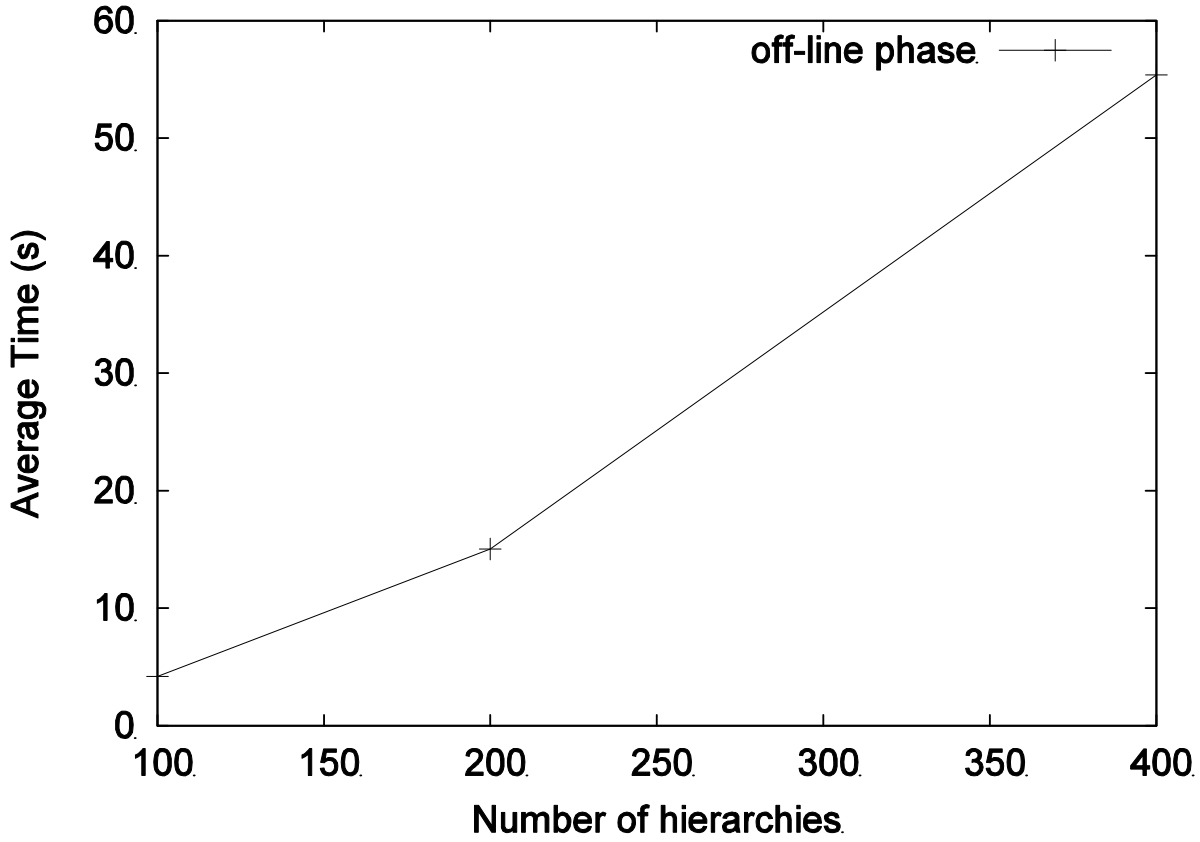
- We evaluated the off-line phase using synthetically generated trees with controlled overlaps
- Perl prototype
- Data generation
  - Generate 100 random trees to be used as overlaps
  - Generate application hierarchies that include an "overlap tree" with some probability "sharedprob"
  - Otherwise expand tree using "expandprob" and a maximum fanout.
  - Recursion stops when maximum depth is reached.
- Results show that the off-line phase is feasible.

Conclusion

- We found problems with uncontrolled proliferation of application hierarchies in a real data warehouse deployment at a bank
- One key performance problem is the inability to exploit pre-computed aggregates.
- We propose to find hierarchy overlap information and exploit them for optimizing queries using pre-computed aggregations.
- Our preliminary experiments show that finding overlap information is feasible.
- Future work: an end-to-end experimental evaluation



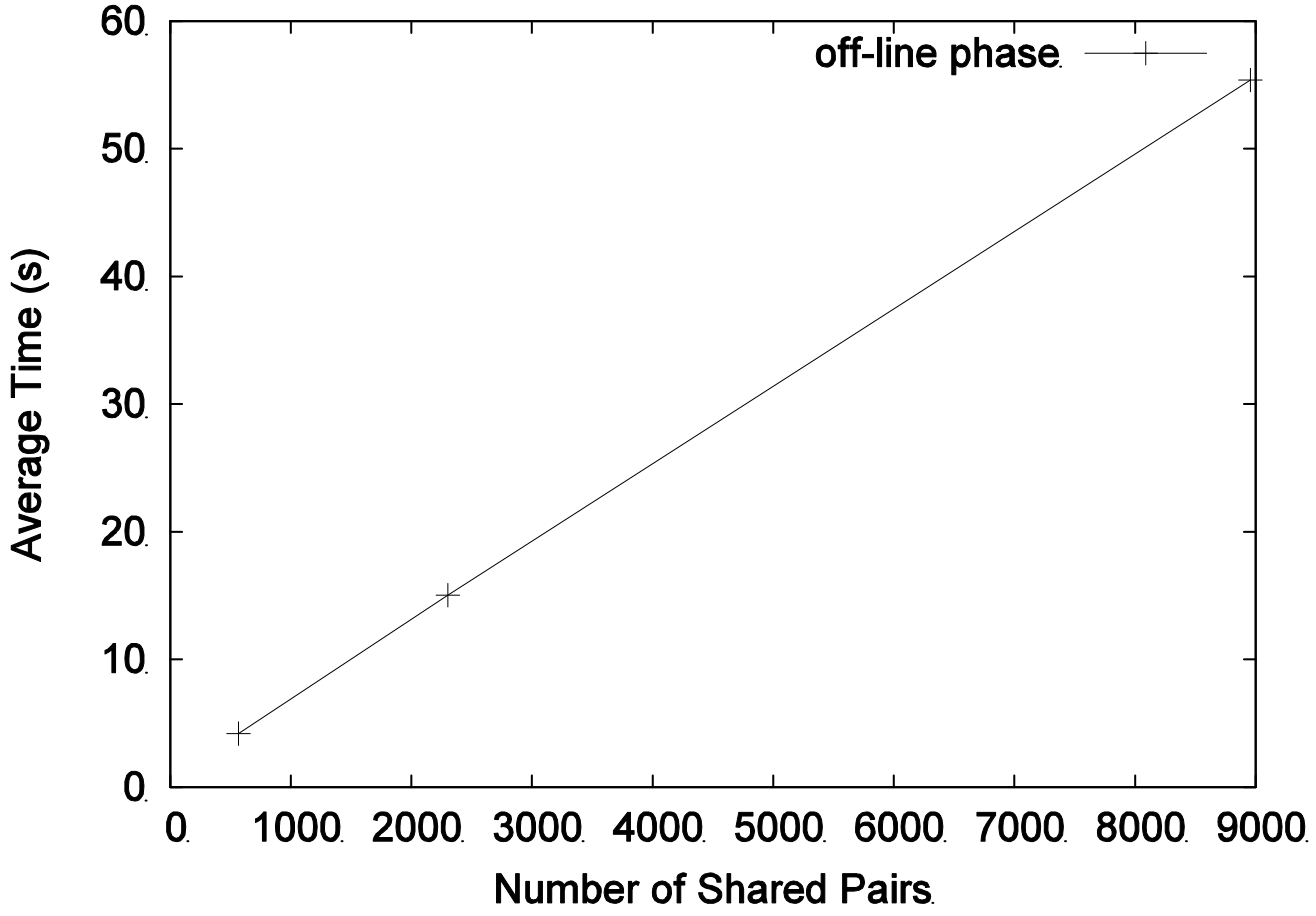
Avg. Time vs No. of Trees



Maxfanout = 5
Maxdepth = 16
Expandprob = 0.8
Sharedprob = 0.8

Averaged over 10 random data sets

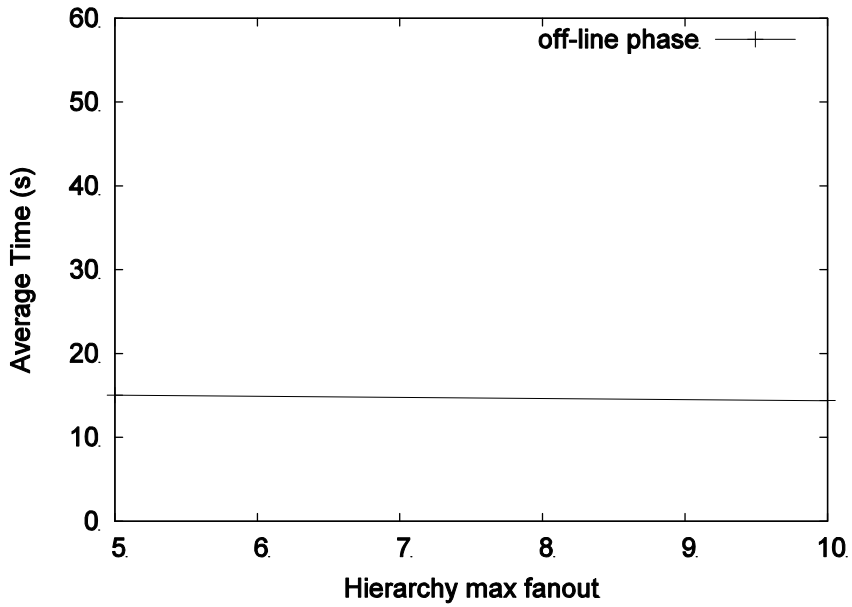
Avg. Time vs No. Shared Pairs



Count the number of shared pairs output for the x-axis

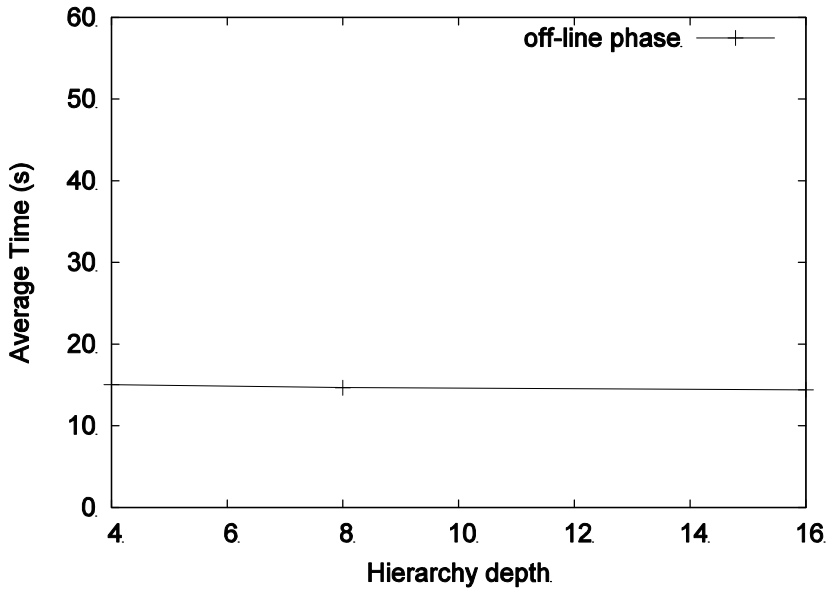
Sensitivity to Tree Sizes

Vs Max Fanout



No. Hierarchies = 200
Maxdepth = 16
Expandprob = 0.8
Sharedprob = 0.8

Vs Max Depth



No. Hierarchies = 200
Maxfanout = 10
Expandprob = 0.8
Sharedprob = 0.8

Related Work

- Treescape
- View Selection Problem
- Subtree mining
- Partial sums