

# Caching Multi-dimensional Queries Using Chunks

---

Prasad Deshpande

Joint work with

Jeffrey F. Naughton

Karthikeyan Ramasamy

Amit Shukla



# OLAP Schema

- Star Schema of Fact and Dimension tables
- Example:
  - Product(pname, pid, pcategory)
  - Store(sname, sid, scity, sstate, country)
  - Date(dname, did, dday, dyear)
  - Sales(pid, sid, did, dollar\_sales)



# OLAP Queries

- View data for some dimension members at different levels of aggregation

- Example:

Select pname, dmonth, sum(dollar\_sales)

From Sales, Date, Product

Where pcategory = "clothes"

AND dmonth in {Jan, Feb, Mar}

AND Sales.did = Date.did

AND Sales.pid = Product.pid

Group by pid, dmonth

# Outline of Talk

- OLAP data model and queries
- Caching for OLAP queries
- Chunk based caching
- Chunked file organization
- Implementation and performance results
- Summary and future work

# Motivation for Caching

- Require interactive response time
- Queries computationally expensive due to aggregation
- Possible to exploit special properties of the OLAP data model and queries



# Motivation

```
Select pname, dmonth,  
       sum(dollar_sales)  
From Sales, Date, Product  
Where pname =  
       "blaire_cotton_shirts"  
AND dmonth in {Jan, Feb, Mar,  
              Apr, May, Jun}  
AND Sales.did = Date.did  
AND Sales.pid = Product.pid  
Group by pname, pmonth
```

```
Select pname, dmonth,  
       sum(dollar_sales)  
From Sales, Date, Product  
Where pname =  
       "blaire_cotton_shirts"  
AND dmonth in {Apr, May, Jun,  
              Jul, Aug, Sep}  
AND Sales.did = Date.did  
AND Sales.pid = Product.pid  
Group by pname, pmonth
```

## Overlapping Queries

# Locality in OLAP Queries

- Temporal
- Hierarchical
  - ◆ Due to access patterns along the hierarchies in the dimensions
  - ◆ Data members related by the parent-child or sibling relationships are accessed together
  - ◆ Example : Cities in Wisconsin

# Classification

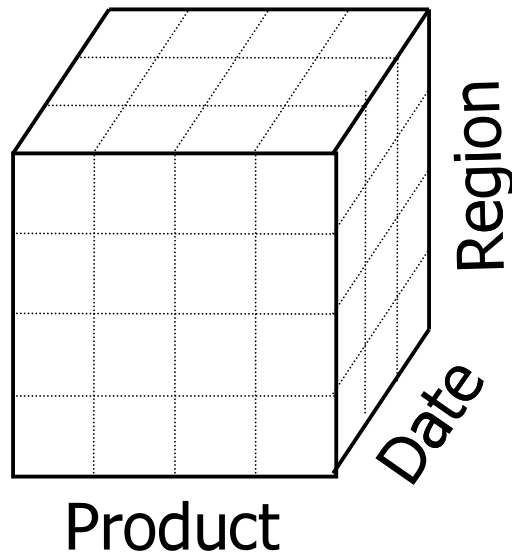
- Unit of Caching
  - ◆ Query Level, Table level, Semantic Regions etc.
- Nature of caching
  - ◆ Static, Dynamic
- Desirable properties
  - ◆ Cache only relevant parts
  - ◆ Dynamic caching



# Multi-dimensional Chunking

- Multi-dimensional arrays are typically stored in chunked format
- Distinct values of each dimension are divided into ranges
- Chunks represent semantic regions

# Chunking



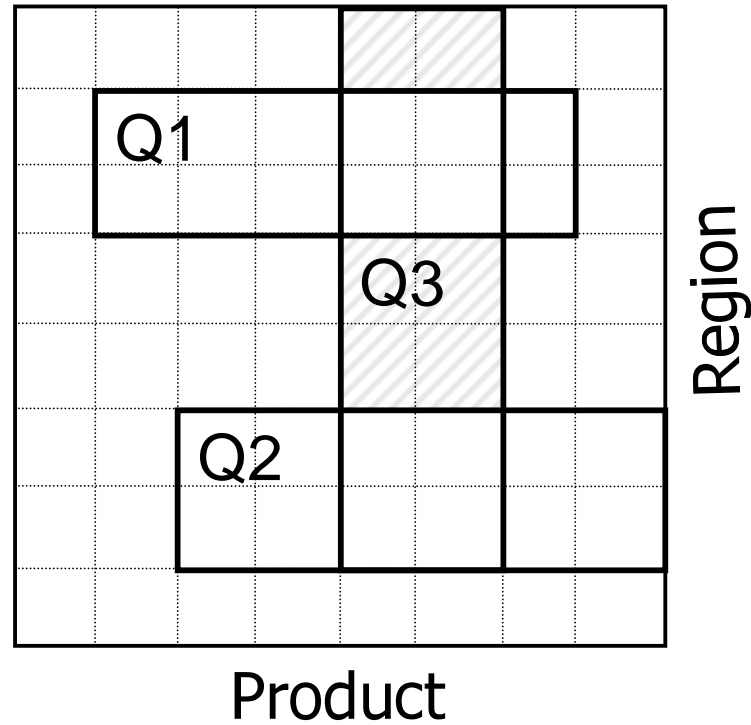
Schema : (Product, Region, Date, Dollar Sales)

# Chunk Caching

- Motivation - partial/full reuse of results
- Chunks -- Unit for caching
- Query results
  - ◆ Split into a set of chunks
  - ◆ chunks are cached
- Cache contains chunks at different levels of aggregation
- Dynamic scheme
  - ◆ different replacement policies

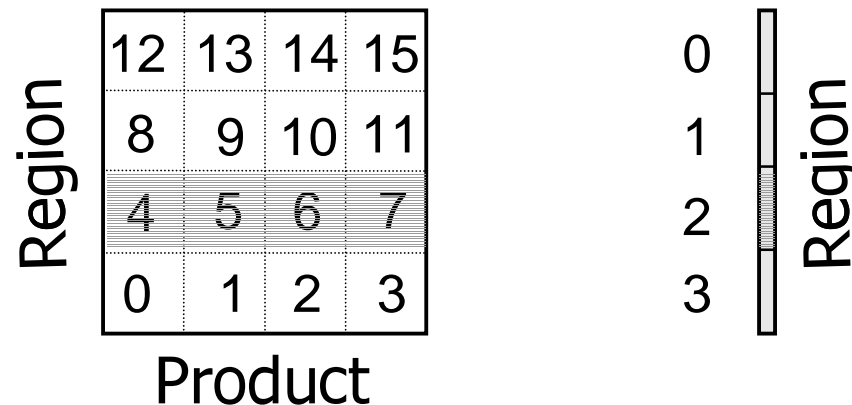


# Reusing Cached Chunks



# Closure Property of Chunks

- Mapping between chunks at different levels of aggregation leads to efficient computation of missing chunks
- Example:



# Issues

- Imposing order on the domain of dimensions
  - ◆ Use hierarchy on the dimension
  - ◆ Better use of hierarchical locality
  - ◆ Example:  
    Cities in Wisconsin are grouped together for Region
- Size of the chunk
  - ◆ Granularity vs. Overhead

# Chunked File Organization

- Motivation
  - ◆ Reduce cost of a chunk miss
- Chunk-based multi-dimensional arrays
  - ◆ Loss of relational access to the data
- ➔ Apply chunking to relational tables
  - ◆ Data still stored as tuples
  - ◆ Tuples clustered on a chunk basis
  - ◆ A chunk index is built to get access based on chunk numbers

# Advantages

- Cost of accessing a chunk proportional to the size of chunk rather than entire table
- Maintains relational interface
- Achieves multi-dimensional clustering
  - ◆ Improves performance of Bitmap Index

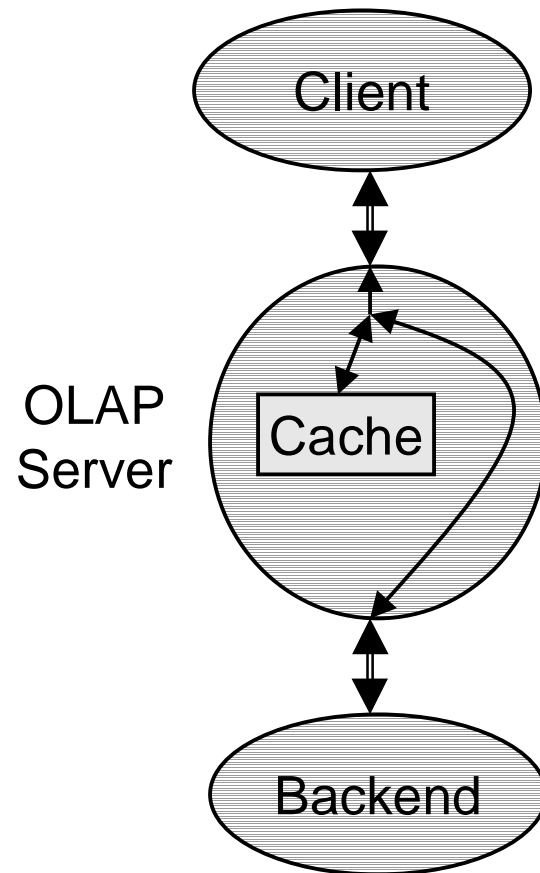


# Putting it Together

- Analyze query selection predicates to get a list of chunk numbers
- Group by clause specifies the level of aggregation (denoted by group by identifier)
- (Group by id, Chunk number) is a key for looking up in the cache



# Processing a Query



# Implementation

- Chunked file implemented in SHORE storage manager of the Paradise Database System
- Tuples are stored in a fixed length record file called Fact file
- Tuples are clustered on chunk number
- A B+-Tree is used to implement the chunk index

# Replacement Policies

- Simple LRU
  - ◆ Not very suitable for OLAP queries
  - ◆ Chunks at different levels of aggregation have different costs of computation
- Benefit Based Policies
  - ◆ Associate a “profit metric (benefit)” with each chunk
  - ◆ Benefit of a chunk is measured by the fraction of base table it represents
  - ◆ Combined CLOCK scheme with benefit

# Experimental Setup

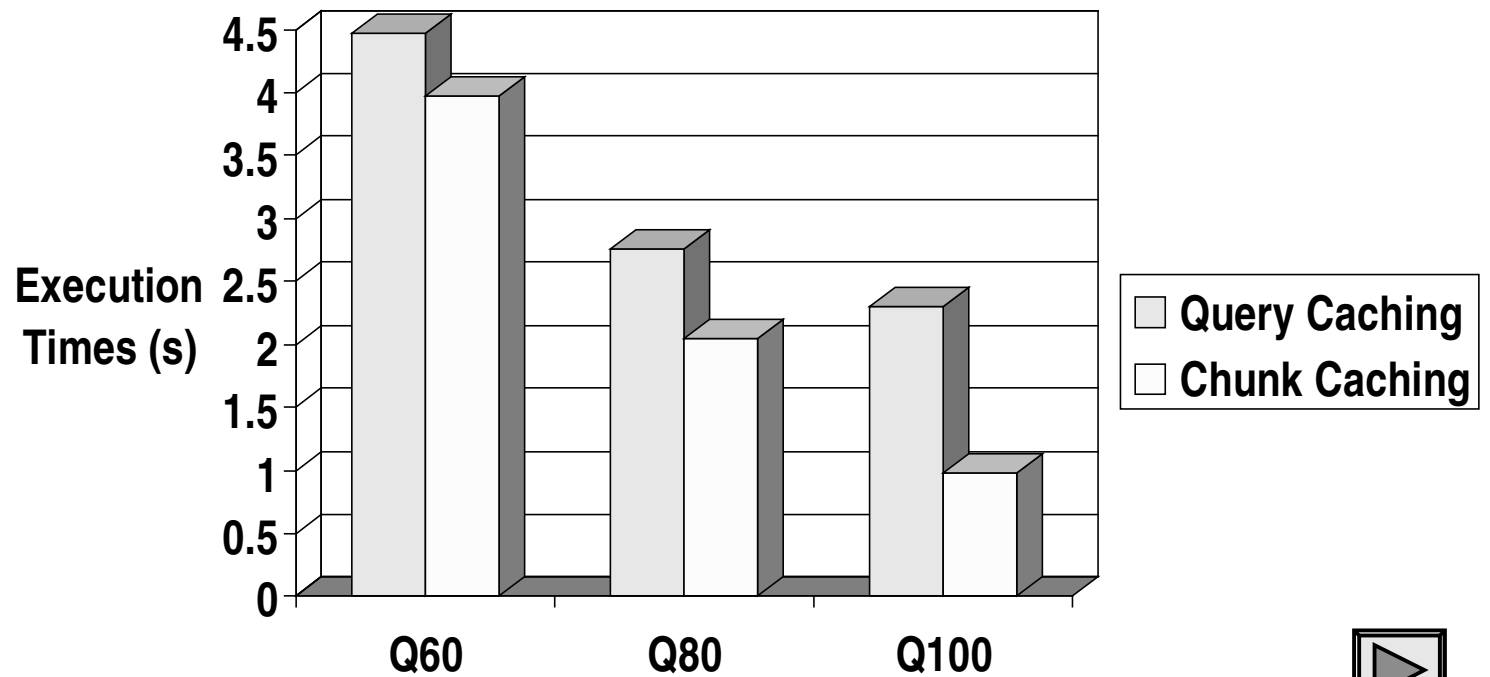
- Four dimensions with hierarchy sizes 3, 2, 3, 2
- Base data size : 500,000 tuples of 20 bytes each
- Cube size : 300 MB
- Cache size : 30 MB
- Buffer pool size at backend : 8MB
- Platform : Dual processor Pentium 133 MHz with 128 MB memory
- Query stream consisting of 1500 queries



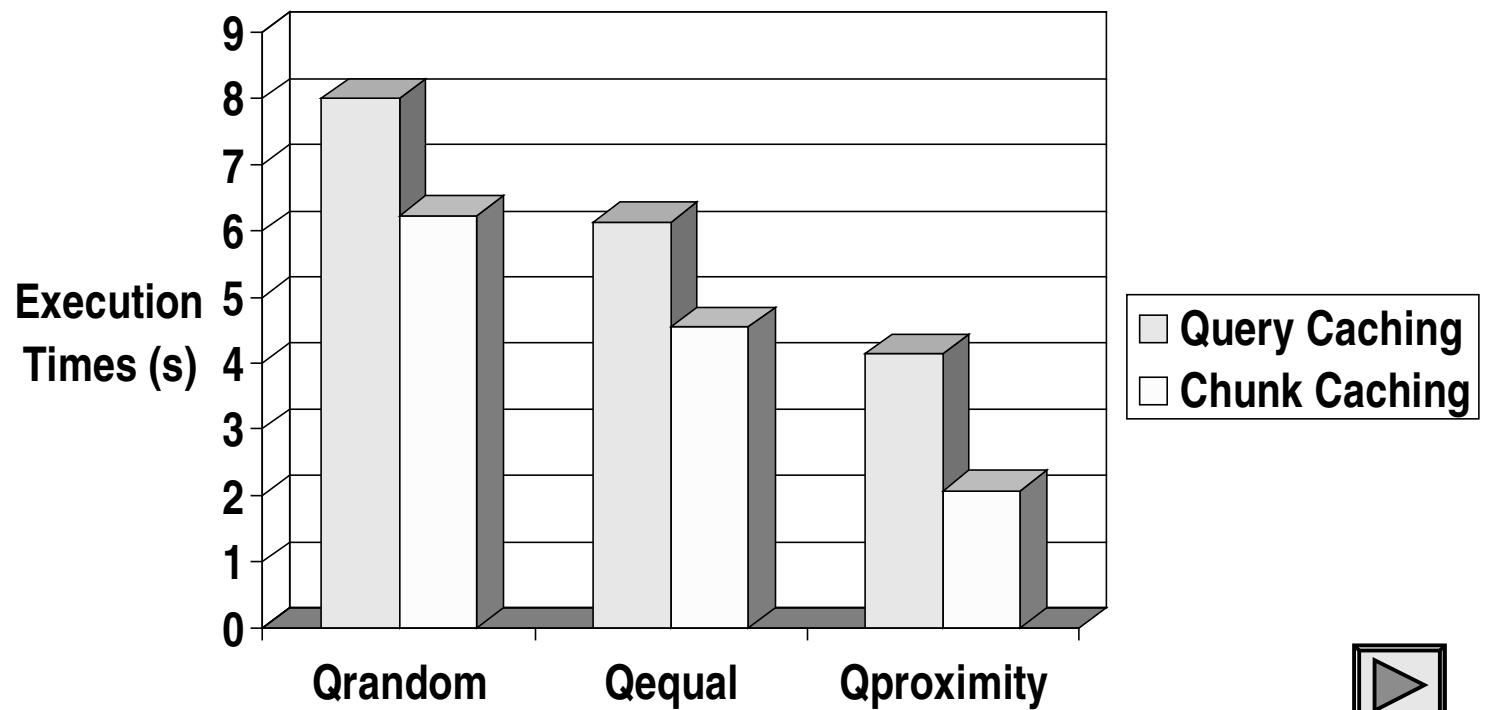
# Query Profile

- Designated hot region - a large percentage of the queries access data in the hot region
  - ◆ Q60, Q80 and Q100 - 60, 80 and 100% of the queries access 20% of the cube
- Proximity queries - model hierarchical locality
  - ◆ QRandom - 100% randomly generated
  - ◆ QEqual - 50% random, 50% proximity
  - ◆ Qproximity - 20% random, 80% proximity

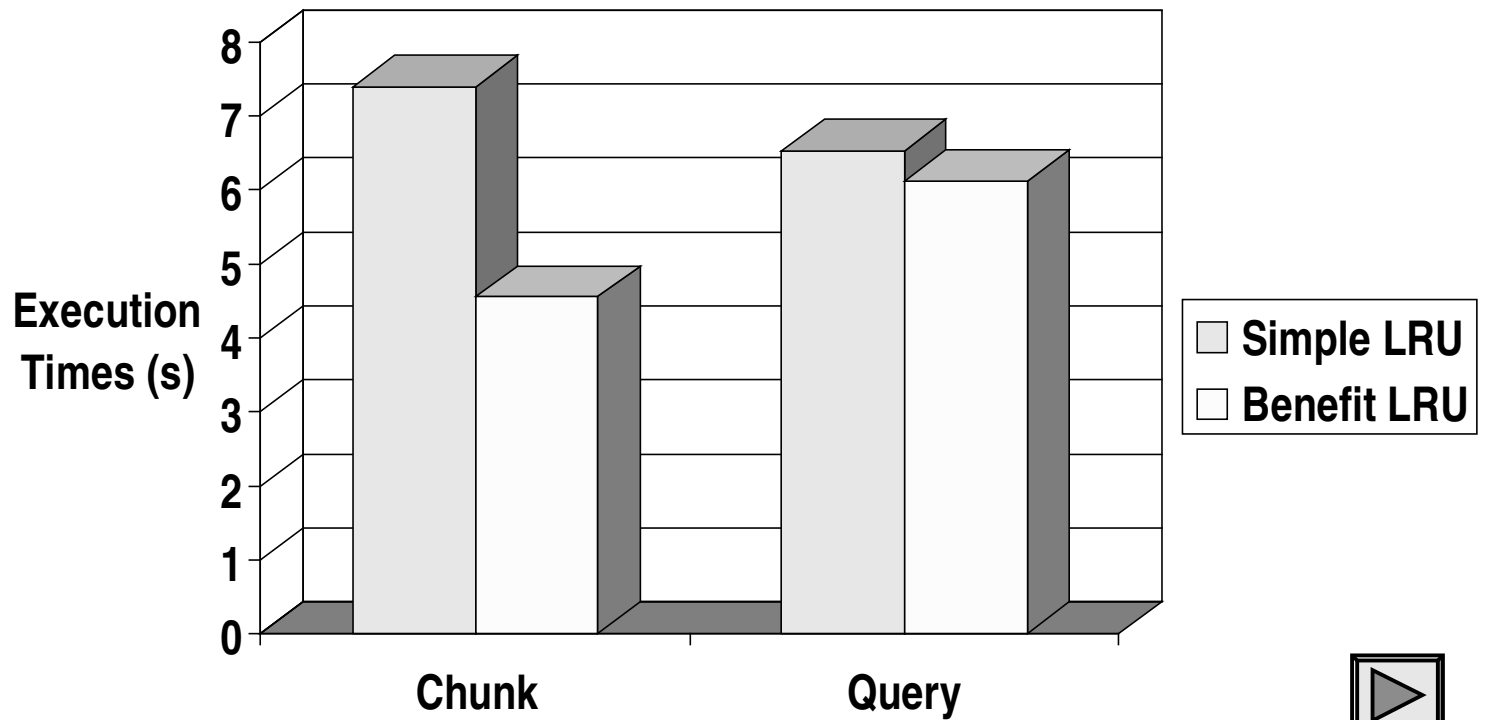
# Comparison With Query Caching



# Comparison With Query Caching



# Replacement Policies



# Summary and Future Work

- Chunk based caching performs better than traditional query caching
- Benefit based LRU performs better than Simple LRU
- Future work
  - ◆ Implement aggregation in the cache
  - ◆ Other cache policies such as pre-fetching

