CSE 544
Principles of Database Management Systems

Magdalena Balazinska

Winter 2009

Lecture 15 - Data Warehousing: Cubes

Final Exam Overview

- Open books and open notes
- No laptops and no other mobile devices
  - But I may ask you to bring a calculator
- Content: all lectures except lectures 14 and 17
  - Review book chapters and papers
  - Review lecture notes (ppt slides)
- Sample final is posted on course website

Important Paper Sections

- L2 - Data models: Sec. 1 – 4 carefully, rest quickly
- L3 – Codd's paper: Sec 1.1-1.4 and all of Sec. 2
  - But I will not ask you about the details of the operators
- L4 – Book: Sec. 19.1 - 19.7
- L5 – Anatomy paper: Sec. 1 – 4 but skip 2.3
- L6 – Lecture notes only
- L7 – Join algos paper: All but ignore equations and skim the end
  - Make sure you understand how the algorithms work!
- L8 – Selinger's paper: All
- L9 and L10: Transactions paper: All

Important Paper Sections

- L11 – Three papers: sections as indicated on website
- L12 – Bigtable: ALL
- L13 – DBMS as a service: Only lecture notes
- L14 – Guest lecture not on the final
- L15 - Cube paper: everything except 1.2, 3.3-3.6, and 4
- L16 - C-store paper: next lecture
- L17 – Not on the final

- And remember that book chapters give fundamental material

Where We Are

- We covered all the fundamental topics
- We are now discussing advanced topics
- **Theme: big data management**
  - Parallel data processing (MapReduce, parallel DBMSs)
  - High-performance distributed storage: Bigtable
  - Databases as a service
    - Amazon Web Services, Microsoft Azure, Google App Engine
    - SimpleDB, SQL Data Services, Google App Engine Datastore
  - Today: OLAP, decision support, and data cubes

References

- Data Cube: A Relational Aggregation Operator Generalizing Group By, Cross-Tab, and Sub-Totals.

  Jim Gray et. al. Data Mining and Knowledge Discovery 1, 29-53. 1997

- Database management systems.

  Ramakrishnan and Gehrke.

  Third Ed. **Chapter 25**

Why Data Warehouses?

- DBMSs designed to manage operational data
  - Goal: support every day activities
  - Online transaction processing (OLTP)
  - Ex: Tracking sales and inventory of each Wal-mart store

- Enterprises also need to analyze and explore their data
  - Goal: summarize and discover trends to support decision making
  - Online analytical processing (OLAP)
  - Ex: Analyzing sales of all Wal-mart stores by quarter and region

- To support OLAP consolidate all data into a **warehouse**
  - (note: this is an example of lazy master replication)

Data Warehouse Overview

- Consolidated data from many sources
  - Must create a single unified schema
  - The warehouse is like a materialized view
- Very large size: terabytes of data are common
- Complex read-only queries (no updates)
- Fast response time is important

Creating a Data Warehouse

- **Extract** data from distributed operational databases
- **Clean** to minimize errors and fill in missing information
- **Transform** to reconcile semantic mismatches
  - Performed by defining views over the data sources
- **Load** to materialize the above defined views
  - Build indexes and additional materialized views
- **Refresh** to propagate updates to warehouse periodically

Alternative: Distributed DBMS

- User submits a query at one site
- Query is defined over data located at different sites
  - Different physical locations
  - Different types of DBMSs
- Query optimizer finds the best distributed query plan
  - Query executes across all the locations
  - Results shipped to query site and returned to user

Distributed DBMS Limitations

- Top-down
  - Global, a priori data placement
  - Global query optimization
    - One query at a time; no notion of load balance
  - Distributed transactions, tight coupling
- Assumes full cooperation of all sites
- Assumes uniform sites
- Assumes short-duration operations

- Limited scalability

Back to Warehouses: Outline

- Multidimensional data model and operations
- Data cube & rollup operators
- Data warehouse implementation issues
- Other extensions for data analysis

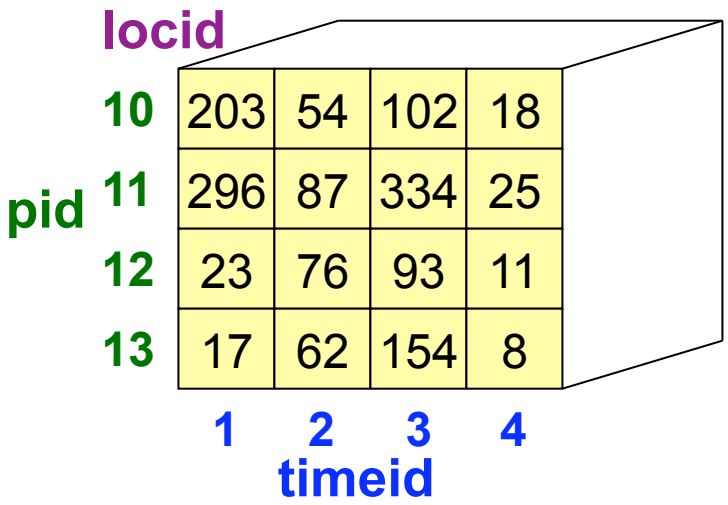
Data Analysis Cycle

- Formulate query that extracts data from the database
  - Typically ad-hoc complex query with group by and aggregate
- Visualize the data (e.g., spreadsheet)
  - Dataset is an N-dimensional space
- Analyze the data
  - Identify “interesting” subspace by aggregating other dimensions
  - Categorize the data an compare categories with each other
  - Roll-up and drill-down on the data

Multidimensional Data Model

- Focus of the analysis is a collection of measures
  - Example: Wal-mart sales

- Each measure depends on a set of dimensions
  - Example: product (pid), location (lid), and time of the sale (timeid)

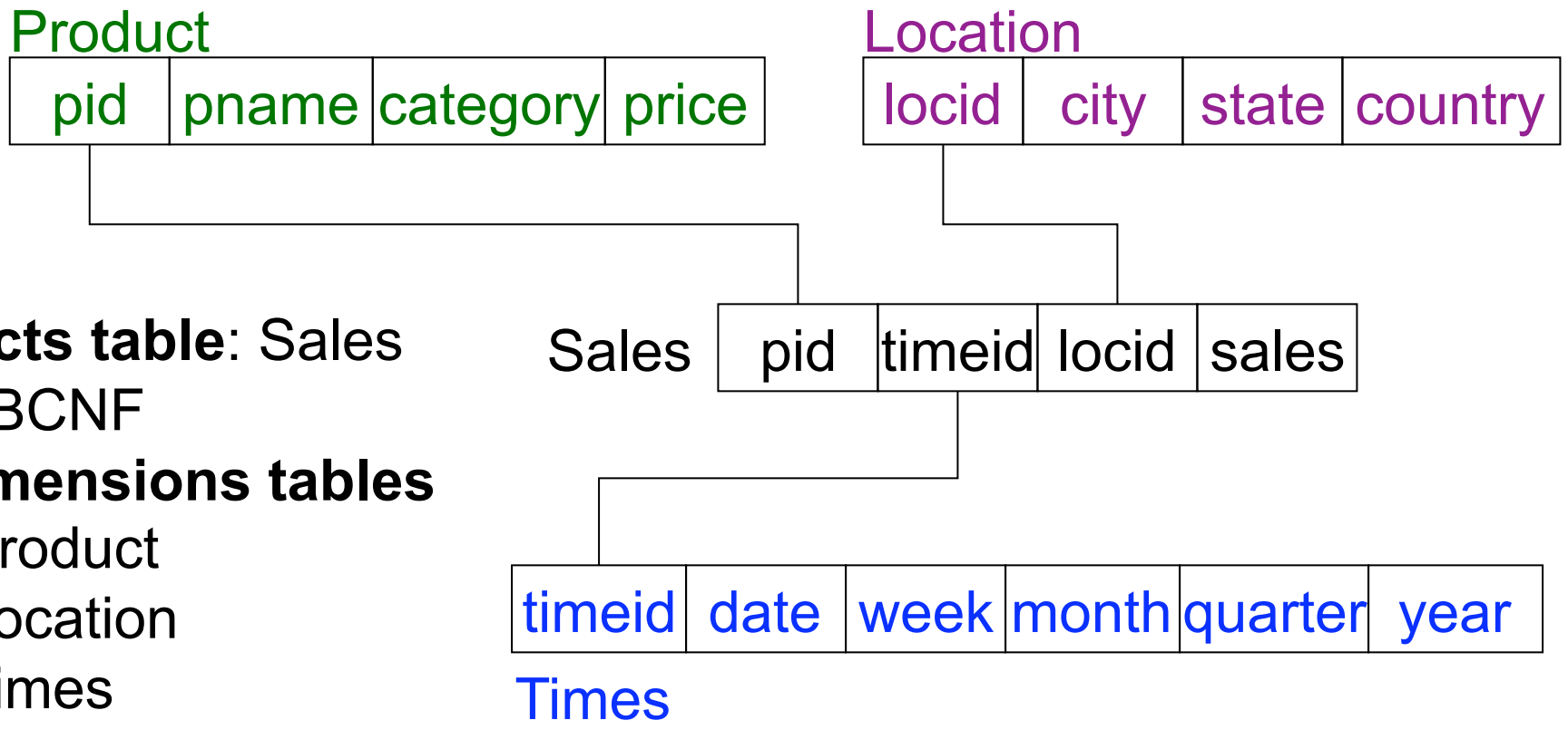


Slicing: equality selection on one or more dimensions

Dicing: range selection

Star Schema

Representing multidimensional data as relations (ROLAP)



Facts table: Sales
In BCNF

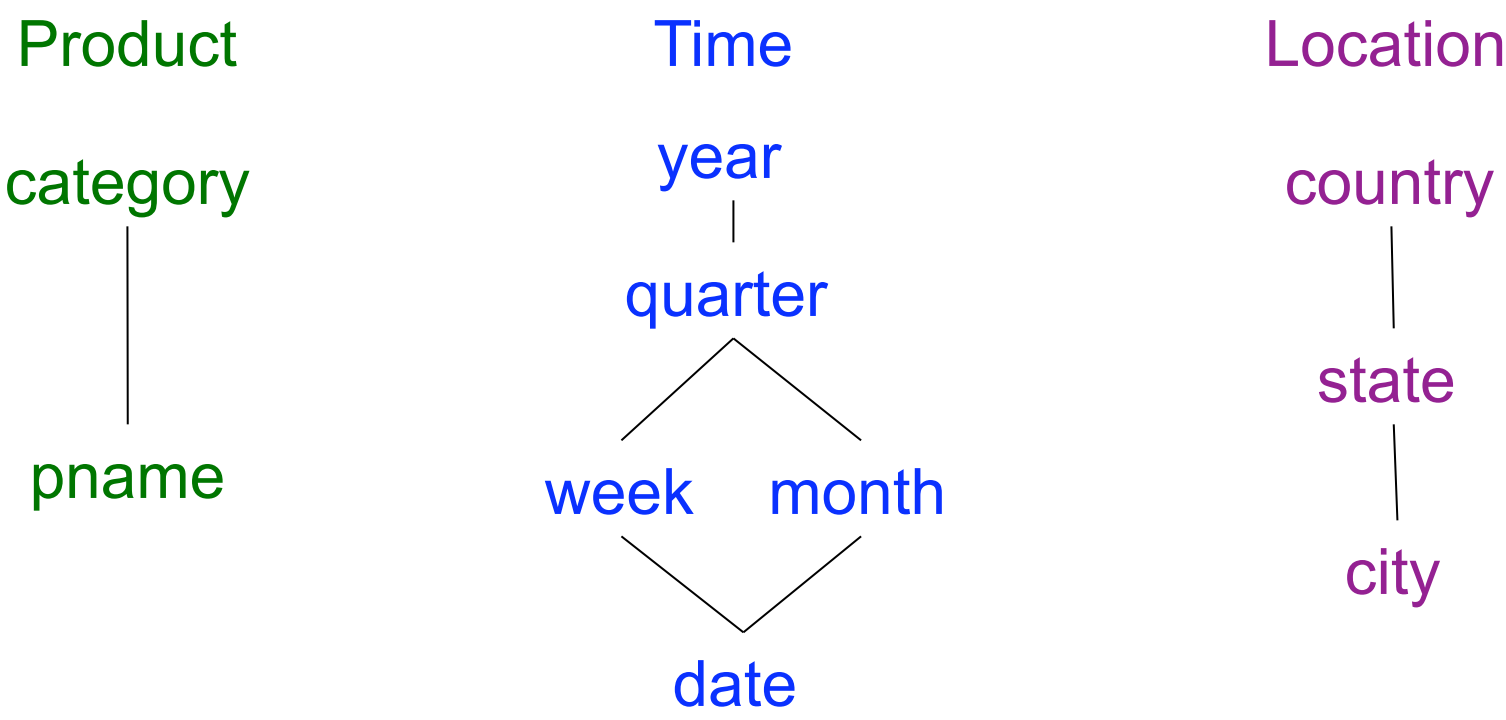
Dimensions tables

- Product
- Location
- Times

Not normalized

Dimension Hierarchies

Dimension values can form a hierarchy described by attributes



Desired Operations

- Histograms (agg. over computed categories)
  - Problem: awkward to express in SQL (paper p.34)
- Summarize at different levels: roll-up and drill-down
  - Ex: total sales by day, week, quarter, and year
- Pivoting
  - Ex: pivot on location and time
  - Result of pivot is a cross-tabulation
  - Column values become labels

	WI	CA	Total
2005	500	200	**700**
2006	150	850	**1000**
2007	250	400	**650**
Total	**900**	**1450**	**2350**

Challenge 1: Representation

- Problem: How to represent multi-level aggregation?
  - Ex: Table 3 in the paper need 2^N columns for N dimensions!
  - Ex: Table 4 has even more columns!
  - And that's without considering any hierarchy on the dimensions!
- Solution: special "all" value

T.year	L.state	SUM(S.sales)
2005	WI	**500**
2005	CA	**200**
2005	ALL	**700**
...	...	**...**
ALL	ALL	**2350**

Note: SQL-1999 standard uses NULL values instead of ALL

Challenge 2: Computing Agg.

- Need 2^N different SQL queries to compute all aggregates
  - Expressing roll-up and cross-tab queries is thus daunting
  - Cannot optimize all these independent queries

- Solution: CUBE and ROLLUP operators

Outline

- Multidimensional data model and operations
- Data cube & rollup operators
- Data warehouse implementation issues
- Other extensions for data analysis

Data Cube

- CUBE is the N-dimensional generalization of aggregate
- Cube in SQL-1999

```
SELECT T.year, L.state, SUM(S.sales)
FROM Sales S, Times T, Locations L
WHERE S.timeid=T.timeid and S.locid=L.locid
GROUP BY CUBE (T.year,L.state)
```

- Creating a data cube requires generating the power set of the aggregation columns

Rollup

- Rollup produces a subset of a cube
- Rollup in SQL-1999

```
SELECT T.year,T.quarter, SUM(S.sales)
FROM Sales S, Times T
WHERE S.timeid=T.timeid
GROUP BY ROLLUP (T.year,T.quarter)
```

- Will aggregate over each pair of (year,quarter), each year, and total, but will **not** aggregate over each quarter

Computing Cubes and Rollups

- Naive algorithm
  - For each new tuple, update each of 2^N matching cells
- More efficient algorithm
  - Use intermediate aggregates to compute others
  - Relatively easy for distributive and algebraic functions
- Updating a cube in response to updates is more challenging

Outline

- Multidimensional data model and operations
- Data cube & rollup operators
- Data warehouse implementation issues
- Other extensions for data analysis

Indexes

- Bitmap indexes: good for sparse attributes (few values)

M	F
0	1
1	0
1	0

custid	name	gender	rating
10	Alice	F	3
11	Bob	M	4
12	Chuck	M	1

1	2	3	4
0	0	1	0
0	0	0	1
1	0	0	0

- Join indexes: to speed-up specific join queries
  - Example: Join fact table F with dimension tables D1 and D2
  - Index contain triples of rids $<r_1,r_2,r>$ from D_1, D_2, and F that join
  - Alternatively, two indexes, each one with pairs $<v_1,r>$ or $<v_2,r>$ where v_1, v_2 are values of tuples from D_1, D_2 that join with r

Materialized Views

- How to choose views to materialize?
  - Physical database tuning
- How to keep view up-to-date?
  - Could recompute entire view for each update: expensive
  - Better approach: incremental view maintenance
  - Example: recompute only affected partition
  - How often to synchronize? Periodic updates (at night) are typical

Outline

- Multidimensional data model and operations
- Data cube & rollup operators
- Data warehouse implementation issues
- Other extensions for data analysis

Additional Extensions for Decision Support

- Window queries

```
SELECT L.state, T.month, AVG(S.sales) over W AS movavg
FROM Sales S, Times T, Locations L
WHERE S.timeid = T.timeid AND S.locid=L.locid
WINDOW W AS (PARTITION BY L.State
        ORDER BY T.month
        RANGE BETWEEN INTERVAL '1' MONTH PRECEDING
        AND INTERVAL '1' MONTH FOLLOWING)
```

- Top-k queries: optimize queries to return top k results
- Online aggregation: produce results incrementally