OLAP : Concepts and Products

Alberto Mendelzon

University of Toronto

Outline

1. What is OLAP?
2. History
3. Technical Overview
4. Products
5. Research Issues

What is OLAP?

- Multi-gigabyte spreadsheets
- It's what we always thought databases were about
- Wrong: they were about OLTP until ~ 1990

> *"As enabling as RDBMs have been for users, they were never intended to provide powerful functions for data synthesis, analysis and consolidation."*
>
> E.F. Codd, 1993

- Buzzwords:

datacube	dimension	drill across	drill-down
rollup	DSS	MDDB	OLTP
pivot	ROLAP	MOLAP	slice-and-dice
snowflake	star join		

OLAP vs. OLTP

	OLTP	OLAP
Typical user	clerical	professional
Usage	run business	analyze
User interaction	pre-determined	ad-hoc
Unit of work	transaction	query
Records accessed	tens	millions
Number of users	thousands	hundreds
Focus	data in	info out

History

Pre-history

- Decision Support Systems, from the 1960's
- A.C. Nielsen, IRI, Walsh: suppliers of syndicated data

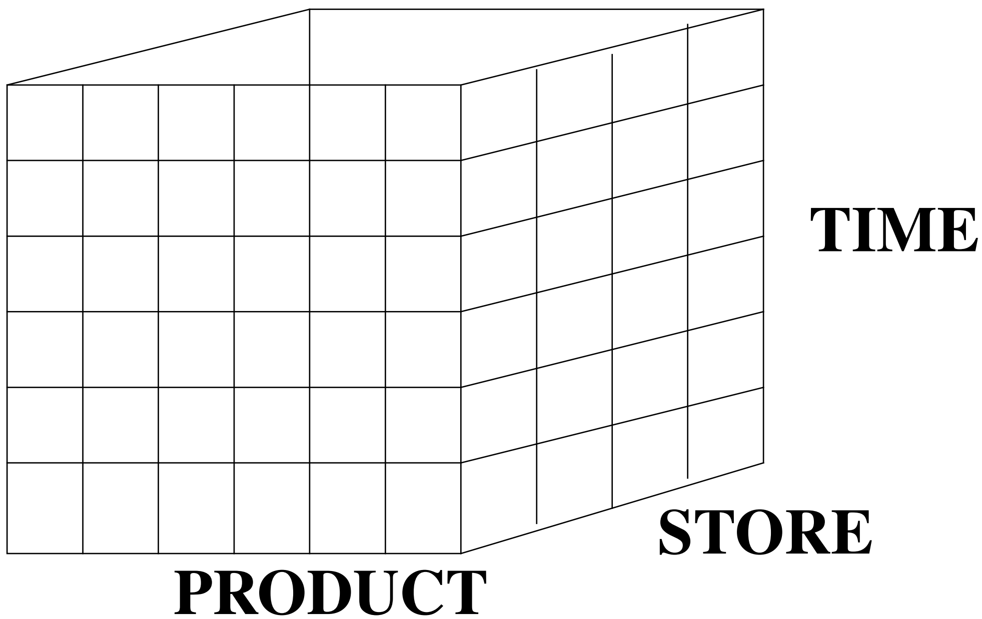
Take-off

- Codd paper: *Providing OLAP to user-analysts: an IT mandate* (commissioned by Arborsoft, 1993)
- OLAP Council, established 1995
- Gray et al *Data Cube* paper, ICDE '96
- OLAP APB-1 Benchmark, April 1996

The Multidimensional Data Model

Describe the process of interest as:

- a (huge) set of **facts** or multidimensional points
- a (small) set of descriptions for each **dimension**



Example: supermarket chain

A fact: n units of a certain **product** p, costing c dollars, were sold on **date** d for x dollars in **store** s.

Dimensions:

- *Product:* product-code, description, brand, category, unit-cost,...
- *Store:* store-code, store-name, address, floor-plan-type,...
- *Time:* date-key, day-of-week, month, quarter, year, holiday-flag,...

Measures:

- Number of units n
- Sale price x
- Cost y

Example: weather data

A fact: Temperature t and pressure p were recorded at latitude x, longitude y and altitude a on date d

Dimensions:

- *Latitude*
- *Longitude*
- *Altitude*
- *Date*

Measures:

- Temperature t
- Pressure p

Star Schema

big fact table, small dimension tables

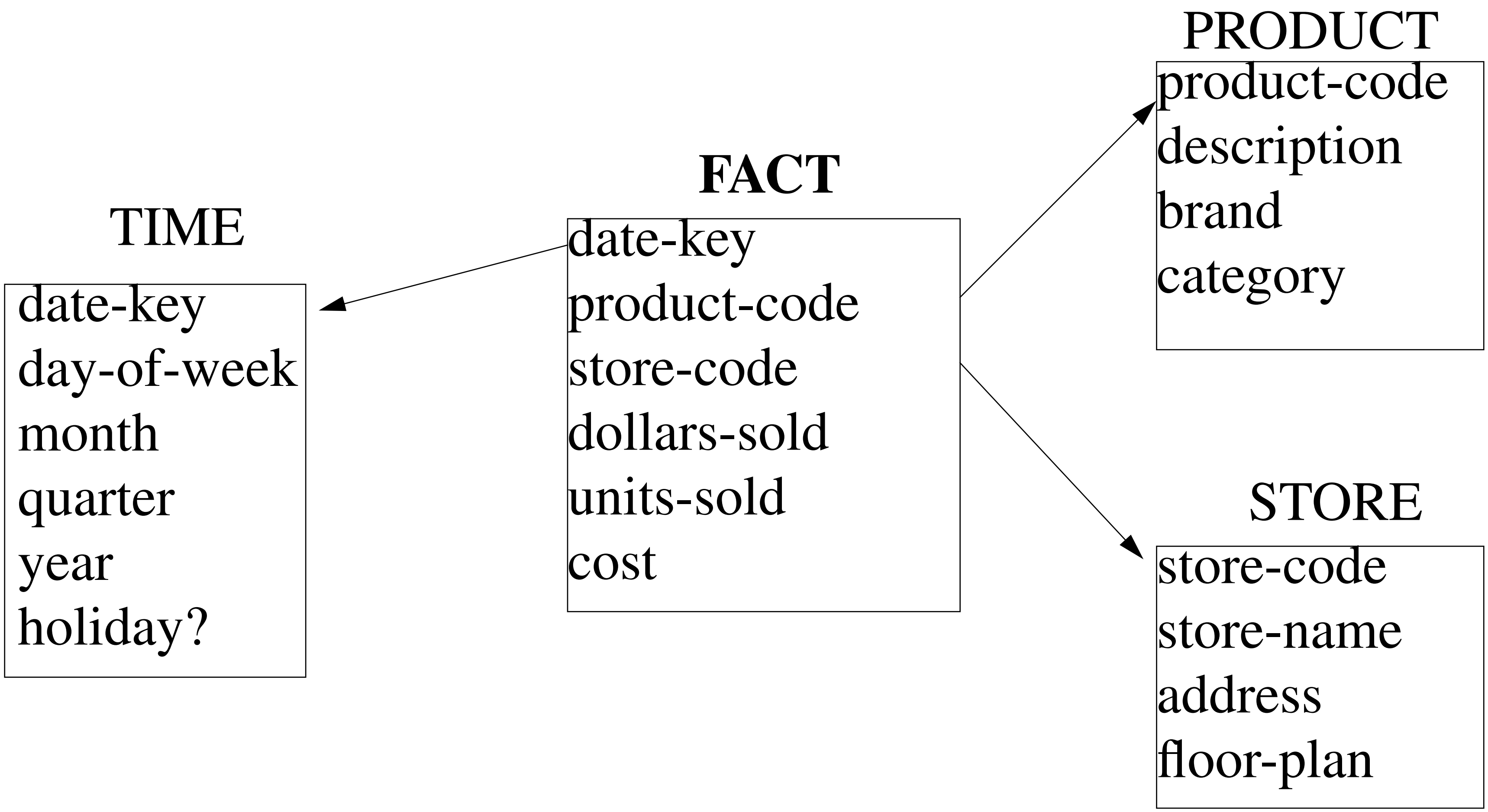


Table sizes

Time dimension: 2 years x 365 days = 730 days

Store dimension: 100 stores

Product dimension: 30,000 products, 10% sell each day per store

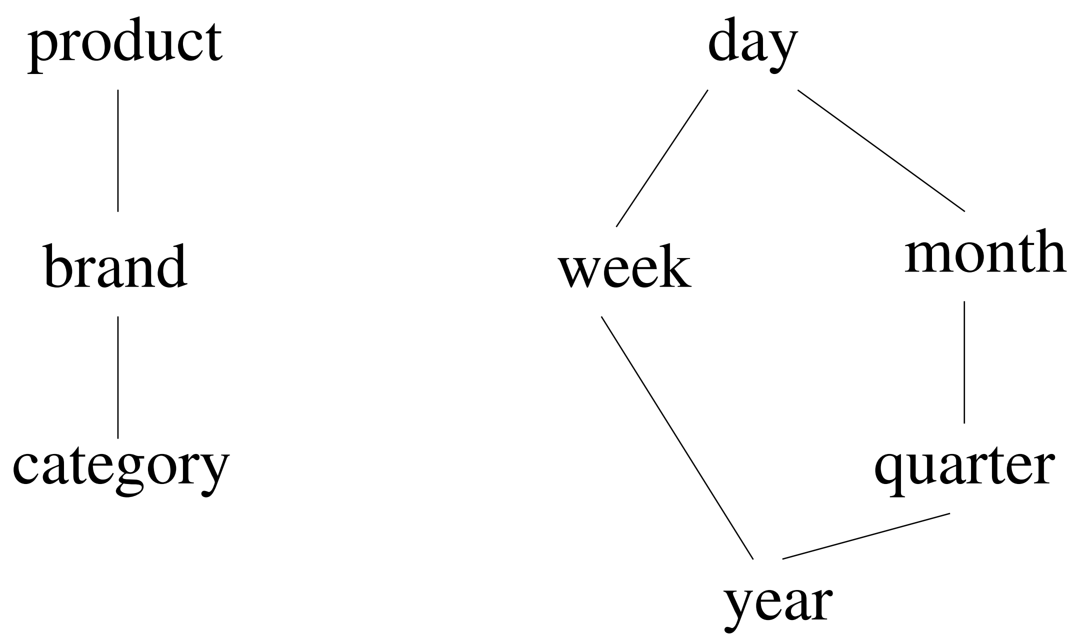
Facts: 730 x 500 x 3000 = 219 million

Fact table size = 219 million x 4 bytes x 6 = 5.2 Gigabytes

(Note implications on query optimization)

Snowflake Schema

Dimensions usually organized in hierarchies:



Normalizing dimension tables leads to snowflake schemas

Queries

Find total amount and number of units sold for each brand in the first quarter of 1997.

SQL:

```
SELECT    p.brand, sum(f.dollars),sum(f.units)
FROM      fact f, product p, time t
WHERE     f.product-code = p.product-code and
          f.date-key = t.date-key and
          t.quarter = '1Q1997'
GROUP BY  p.brand
ORDER BY  p.brand
```

Query Formulation

- Browse dimension tables
- Choose dimension constraints (e.g. 1Q97)
- Choose row headers (e.g. brands)
- Choose additive facts to summarize (e.g. units sold, dollars)
- Look at answer
- Ask "why"

OLAP Operators

pivot:	choose a set of row headers
rollup:	drop a GROUP-BY
drill-down:	add a GROUP-BY
drill-across:	ask query on more than one fact table
slice-and-dice:	combination of dimension constraints with aggregation

Is it all syntactic sugar for SQL?

No: many things are very hard in SQL

- Sales ratio of Toronto's stores vs. Montreal's
- Top 10 products
- Bottom 25% brands
- Sales during promotions compared with non-promotions
- Moving averages, quartiles, percentiles
- Break rows
- Cross-tabulations

See Kimball and Strehlo, ACM SIGMOD Record 24:3, 1995

The Datacube Operator

Gray, Bosworth, Layman, Pirahesh, Data Engineering '96

Compute all possible aggregations over all 2^N subsets of the N dimensions

SELECT product-code, store-code, date-key, sum(dollars-sold)
FROM fact
GROUP BY CUBE (product-code, store-code, date-key)

Will sum the sales over all 8 subsets of product, store, date.

The result is the Data Cube relation

Example

Fact			
Product	Store	Date	Dollars
soap	s1	1-5-97	230
shampoo	s1	2-5-97	320
soap	s2	1-5-97	200
shampoo	s2	1-5-97	350

Product	Store	Date	Total
soap	s1	ALL	230
shampoo	s1	ALL	320
soap	s2	ALL	200
shampoo	s2	ALL	350
soap	ALL	1-5-97	430
soap	ALL	2-5-97	0
shampoo	ALL	1-5-97	580
shampoo	ALL	2-5-97	320
ALL	s1	1-5-97	230
ALL	s1	2-5-97	320
ALL	s2	1-5-97	550
ALL	s2	2-5-97	0
ALL	ALL	1-5-97	780
ALL	ALL	2-5-97	320
soap	ALL	ALL	430
shampoo	ALL	ALL	670
…			
ALL	ALL	ALL	1100

Product	Store	Date	Total
soap	s1	ALL	230
shampoo	s1	ALL	320
soap	s2	ALL	200
shampoo	s2	ALL	350
soap	ALL	1-5-97	430
soap	ALL	2-5-97	0
shampoo	ALL	1-5-97	580
shampoo	ALL	2-5-97	320
ALL	s1	1-5-97	230
ALL	s1	2-5-97	320
ALL	s2	1-5-97	550
ALL	s2	2-5-97	0
ALL	ALL	1-5-97	780
ALL	ALL	2-5-97	320
soap	ALL	ALL	430
shampoo	ALL	ALL	670
…			
ALL	ALL	ALL	1100

Computing the Cube

Cube size:

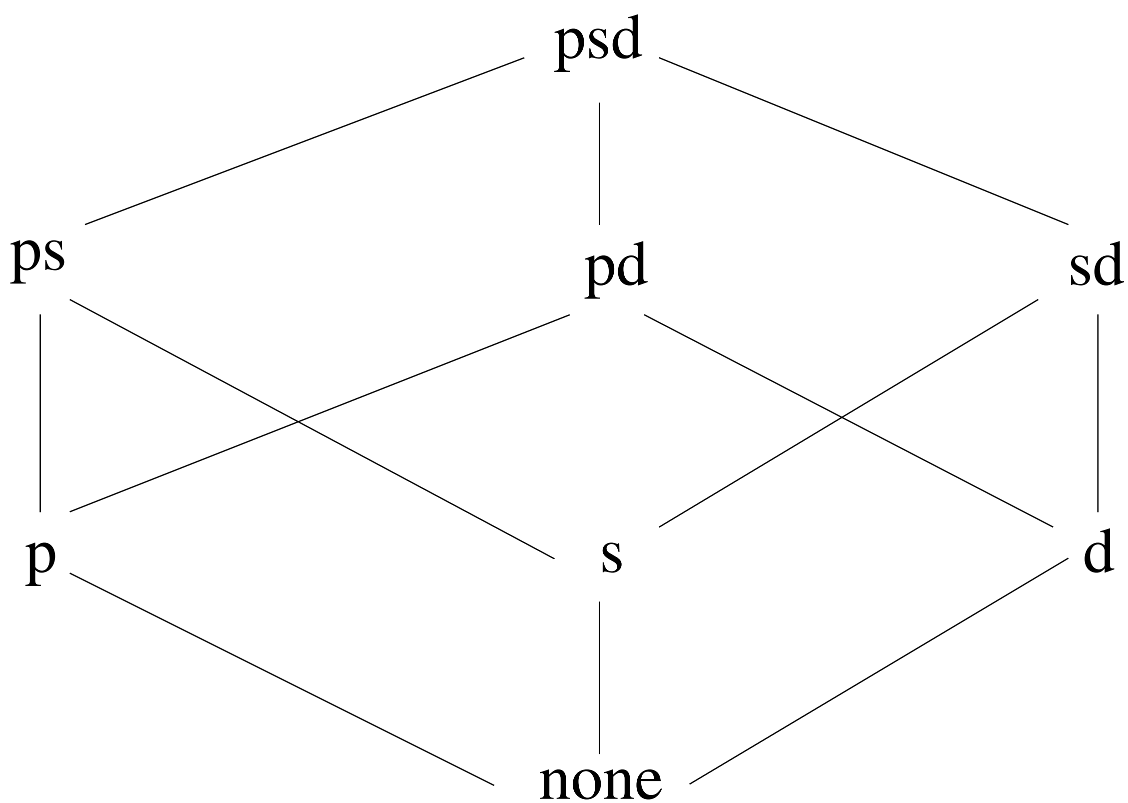
$$(C_1 + 1) \times (C_2 + 1) \ldots \times (C_n + 1)$$

where dimension i has cardinality C_i

- Close to fact table size if data are dense
- Much larger if data are sparse.

  Example: TPC/D `part, customer, supplier` benchmark, fact relation has 6M tuples, the cube has more than 19M

What to materialize?



View Lattice

View Selection Approaches

- Only store raw data (top view); compute aggregates on demand

  standard relational system

- Materialize full cube (union of all the views)

  MDDB

- Materialize only parts of the cube: which parts?

  MetaCube

  Harinarayan, Rajaraman, and Ullman, SIGMOD '96:

  Given lattice, space cost for each view, and frequency of access to each view, minimize average access time for the space available

  Greedy algorithms approximate optimal within $1 - (1/e)$

Using Views to Answer Queries

View V(product-code, quarter, units)

Query Q: How much soap was sold each year?

- Select from V where product-code = soap
- Rollup from quarters to years

In general, if:

- V and Q have same set of dimensions
- Q's condition implies V's condition
- Group-by columns in V included in Group-by columns in Q

then, can use V to answer Q (*HP Intelligent Warehouse*)

Product Categories

(source: InfoIT Inc., 1997)

Data acquisition	3	Info Directory	5
Data cleaning	6	MOLAP client	6
Data Delivery	3	MDBMS	8
Data Mart Builder	7	Query tool	25
Data Mart Suite	2	ROLAP client	7
Data Pump	2	ROLAP server	5
Data Transformer	3	Repository	4
DW Management	6	Web-based tool	9
DW Middleware	9	Data mining	28
Developer Tool	5		

- Back-end: data extraction, integration, scrubbing, loading, refreshing

- Data server: query processing

- Front-end: query tools, report generation, data mining,Web interfaces, ...

- Other tools: metadata management, monitoring, administration, security

Server Architectures

- Specialized SQL servers (*Red Brick*)

  Support extended SQL over star and snowflake schemas

- ROLAP servers (*Microstrategy's DSS Server*)

  Middleware to map OLAP to back-end relational DBMS

- MOLAP servers (Arbor's *ESSBASE*)

  Multidimensional engines

ROLAP vs. MOLAP

	Pro	Con
ROLAP	Familiar Technology	Ill-fitting data model
	Scalable	Slow
	Open	
MOLAP	Dimensional data model	Unproven technology
	Specialized query processing	Proprietary
	Specialized indexing	Less scalable

A ROLAP Server

Red Brick Systems

Specialized relational server for OLAP

- Extended SQL “RISQL”: running totals, moving sums and averages, rank, aggregate comparisons
- Optimized load performance: parallel index creation/ integrity checking/ loading
- Space conservation: compressed indexes, compact numeric data types
- Query performance: optimized star join processing, specialized indexing

A MOLAP Server

Arbor Software's ESSBASE

- Extract data subsets from the data warehouse into MD "data marts"

- MDD data model storing full data cube

  hashing on cell coordinates to avoid storing sparse data

- Use Microsoft EXCEL as front-end query tool

- "Drill-through" to SQL back-end for extra detail

Indexing for OLAP

1. Use as many indexes as possible. Don't worry about cost of index maintenance.

Informix-OnLine Extended Parallel Server

Bitmap Indexing

Instead of storing pointers to all records that have value *v* in field *F*,

store *bitmap*: B[i] = 1 if i-th record has value *v* in field *F*

Savings:

- space, for dense fields (|domain of F| x |relation|) bits
- time: very fast AND/OR/NOT of bitmaps

Indexing for OLAP (cont.)

2. Speed up joins

- Foreign column join index (Informix)
- Star query optimization (Oracle 7.3)
- Star index (Red Brick)

FCJ index:

For each dimension *D*, build an index that, given value *v* for *D*, identifies all **fact** tuples that join with the tuple with value *v*.

The others are specializations of this idea to star schemas.

Research Issues

- Data loading, cleaning, integration, maintenance
- Physical Design
  - Compression
  - Indexing
  - View caching
- Query Optimization
  - multi-way joins
  - optimizing aggregates
- Virtual Warehouses
  - Web queries