DATA CUBES

E0 261

Jayant Haritsa

Computer Science and Automation

Indian Institute of Science

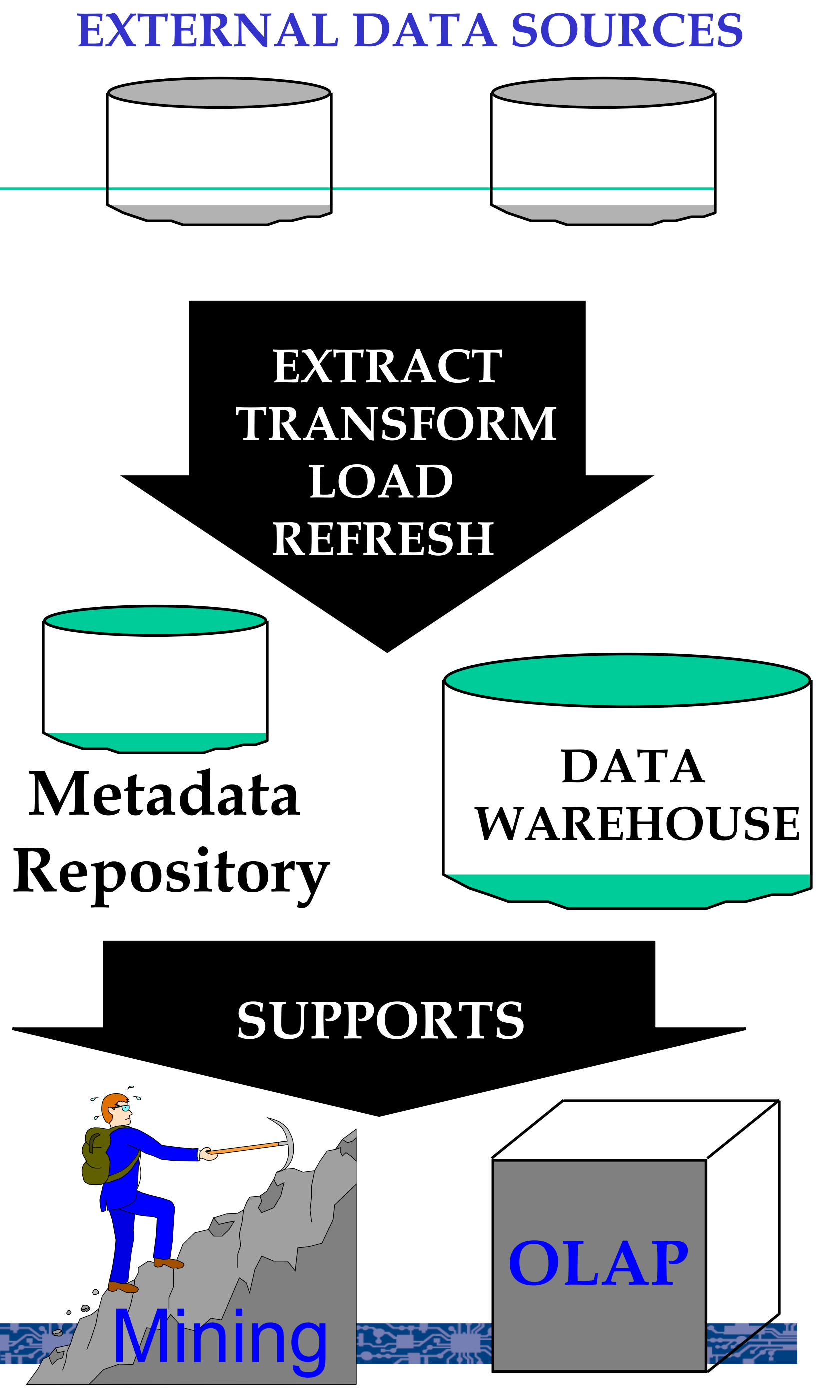
Introduction

- Increasingly, organizations are analyzing historical data to identify useful patterns and support business strategies.
- Emphasis is on complex, interactive, exploratory analysis of very large datasets created by integrating data from across all parts of an enterprise; data is fairly static.
  - Contrast such **On-Line Analytic Processing (OLAP)** with traditional **On-line Transaction Processing (OLTP):** mostly long queries, instead of short update Xacts.

Overview

- Data Warehousing: Consolidate data from many sources in one large repository
  - Loading, periodic synchronization of replicas.
  - Semantic integration.
- OLAP:
  - Complex SQL queries and views.
  - Queries based on spreadsheet-style operations and "multidimensional" view of data.
  - Interactive and "online" queries.
- Note that Data Warehouses form the substrate on which Data Mining can be carried out.

Data Warehousing



- Integrated data spanning long time periods, often augmented with summary information.
- Several terabytes to petabytes common.
- Interactive response times expected for complex queries; ad-hoc updates uncommon.

Warehousing Issues

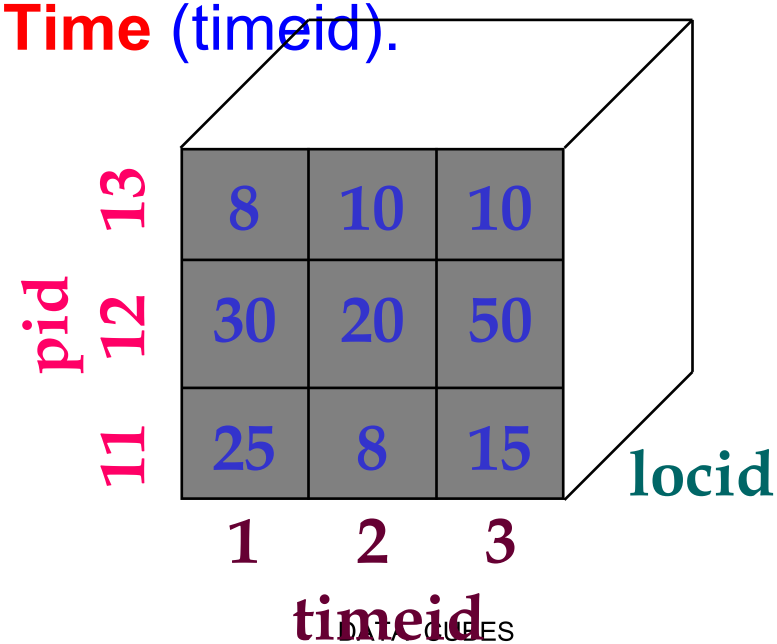
- Semantic Integration: When getting data from multiple sources, must eliminate mismatches, e.g., different currencies, schemas.
- Heterogeneous Sources: Must access data from a variety of source formats and repositories.
- Load, Refresh, Purge: Must load data, periodically refresh it, and purge too-old data.
- Metadata Management: Must keep track of source, loading time, and other information for all data in the warehouse.

Multidimensional Data Model

- Collection of numeric measures, which depend on a set of dimensions.
  - E.g., measure **Sales**, dimensions **Product** (pid), **Location** (locid), and **Time** (timeid).

pid	timeid	locid	sales
11	1	1	25
11	2	1	8
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
13	1	1	8
13	2	1	10
13	3	1	10
11	1	2	35

Slice locid=1 is shown:



MOLAP vs ROLAP

- Multidimensional data can be stored physically in a (disk-resident, persistent) array; called MOLAP systems (covered in TIDS course)
  - Essbase
- Alternatively, can store as a relation; called ROLAP systems (today's paper)
  - Redbrick, Oracle

ROLAP Tables

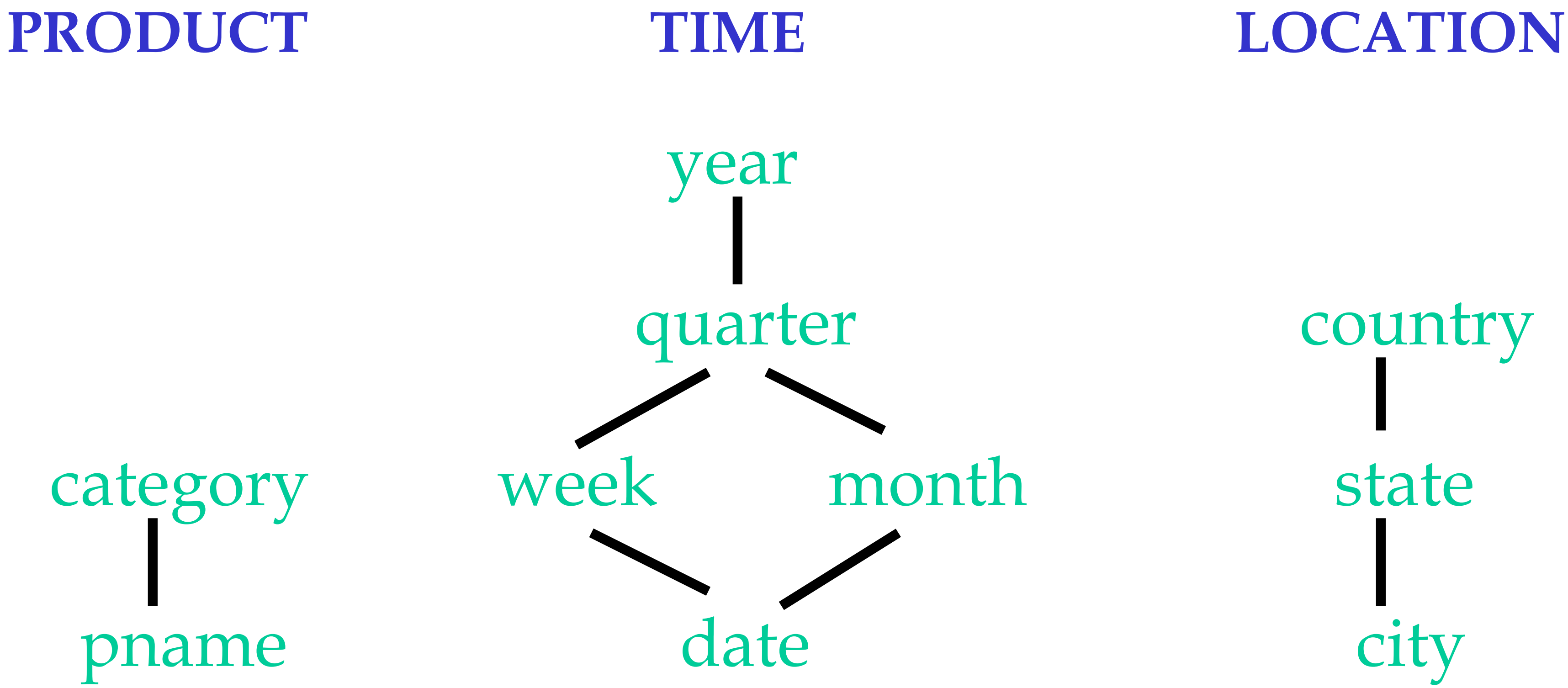
- The main relation, which relates dimensions to a measure, is called the fact table.
  - Example: SALES**(pid, timeid, locid, sales)**

- Each dimension can have additional attributes and an associated dimension table.
  - Example: Products**(pid, pname, category, price)**

- Fact tables are much larger than dimensional tables. Their schema is {foreign keys of all dimension tables + all measure attributes}

Dimension Hierarchies

- For each dimension, the set of values can be organized in a hierarchy:



OLAP Queries

- Influenced by SQL and by spreadsheets.
- A common operation is to aggregate a measure over one or more dimensions.
  - Find total sales.
  - Find total sales for each city, or for each state.
  - Find top five products ranked by total sales.
- Roll-up: Aggregating at different levels of a dimension hierarchy.
  - E.g., Given total sales by city, we can roll-up to get sales by state.

OLAP Queries (contd)

- Drill-down: The inverse of roll-up.
  - E.g., Given total sales by state, can drill-down to get total sales by city.
  - E.g., Can also drill-down on different dimension to get total sales by product for each state.
- Pivoting: Aggregation on selected dimensions.
  - E.g., Pivoting on Location and Time yields this **cross-tabulation**:
  - equivalent to "rotation"

	WB	UP	Total
1995	63	81	144
1996	38	107	145
1997	75	35	110
Total	176	223	339

OLAP Queries (contd)

- Slicing : Equality selections on one or more dimensions.
- Dicing: Range selections on one or more dimensions.

Comparison with SQL Queries

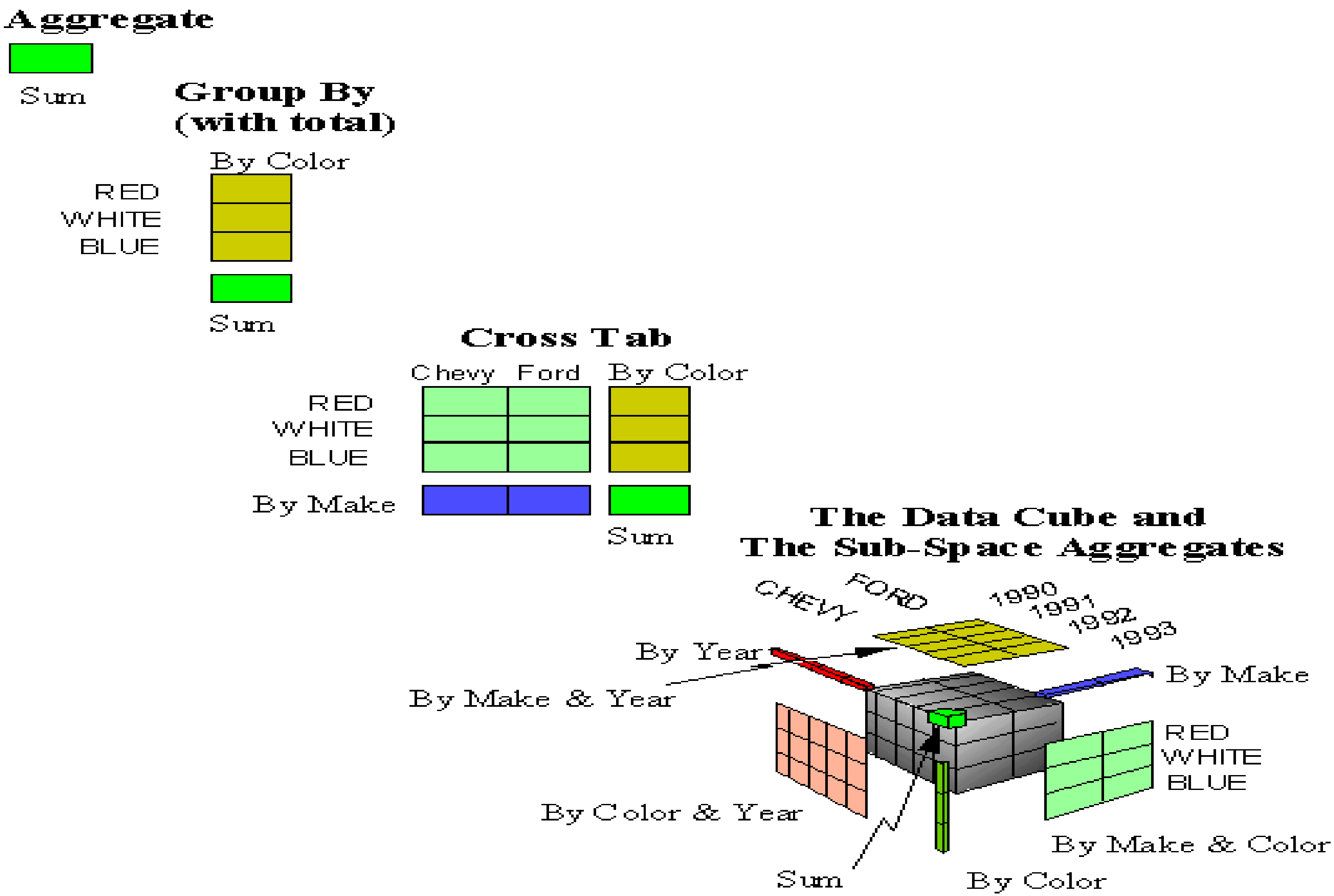
- The cross-tabulation obtained by pivoting can also be computed using a collection of SQL queries:

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

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

```
SELECT SUM(S.sales)
FROM   Sales S, Location L
WHERE  S.locid=L.locid
GROUP BY L.state
```

3-D CUBE Model



The CUBE Operator

- Generalizing the previous example, if there are k dimensions, we have 2^k possible SQL GROUP BY queries that can be generated through pivoting on a subset of dimensions.
- CUBE pid, locid, timeid BY SUM Sales
  - Equivalent to rolling up Sales on all eight subsets of the set {pid, locid, timeid}; each roll-up corresponds to an SQL query of the form:

Lots of work on optimizing the CUBE operator!

```
SELECT SUM(S.sales)
FROM    Sales S
GROUP BY grouping-list
```

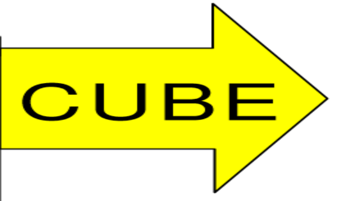
CUBE TABLES

- Use “ALL” to represent the set over which aggregation is computed (Fig 4)

```
SELECT Model, Year, Color, SUM(sales) AS Sales
FROM Sales
WHERE Model in {'Ford', 'Chevy'}
  AND Year BETWEEN 1990 AND 1992
GROUP BY CUBE Model, Year, Color;
```

SALES

Model	Year	Color	Sales
Chevy	1990	red	5
Chevy	1990	white	87
Chevy	1990	blue	62
Chevy	1991	red	54
Chevy	1991	white	95
Chevy	1991	blue	49
Chevy	1992	red	31
Chevy	1992	white	54
Chevy	1992	blue	71
Ford	1990	red	64
Ford	1990	white	62
Ford	1990	blue	63
Ford	1991	red	52
Ford	1991	white	9
Ford	1991	blue	55
Ford	1992	red	27
Ford	1992	white	62
Ford	1992	blue	39



DATA CUBE

Model	Year	Color	Sales
Chevy	1990	blue	62
Chevy	1990	red	5
Chevy	1990	white	95
Chevy	1990	ALL	154
Chevy	1991	blue	49
Chevy	1991	red	54
Chevy	1991	white	95
Chevy	1991	ALL	198
Chevy	1992	blue	71
Chevy	1992	red	31
Chevy	1992	white	54
Chevy	1992	ALL	156
Chevy	ALL	blue	182
Chevy	ALL	red	90
Chevy	ALL	white	236
Chevy	ALL	ALL	508
Ford	1990	blue	63
Ford	1990	red	64
Ford	1990	white	62
Ford	1990	ALL	189
Ford	1991	blue	55
Ford	1991	red	52
Ford	1991	white	9
Ford	1991	ALL	116
Ford	1992	blue	39
Ford	1992	red	27
Ford	1992	white	62
Ford	1992	ALL	128
Ford	ALL	blue	157
Ford	ALL	red	143
Ford	ALL	white	133
Ford	ALL	ALL	433
ALL	1990	blue	125
ALL	1990	red	69
ALL	1990	white	149
ALL	1990	ALL	343
ALL	1991	blue	106
ALL	1991	red	104
ALL	1991	white	110
ALL	1991	ALL	314
ALL	1992	blue	110
ALL	1992	red	58
ALL	1992	white	116
ALL	1992	ALL	284
ALL	ALL	blue	339
ALL	ALL	red	233
ALL	ALL	white	369
ALL	ALL	ALL	941

Cube Operator (contd)

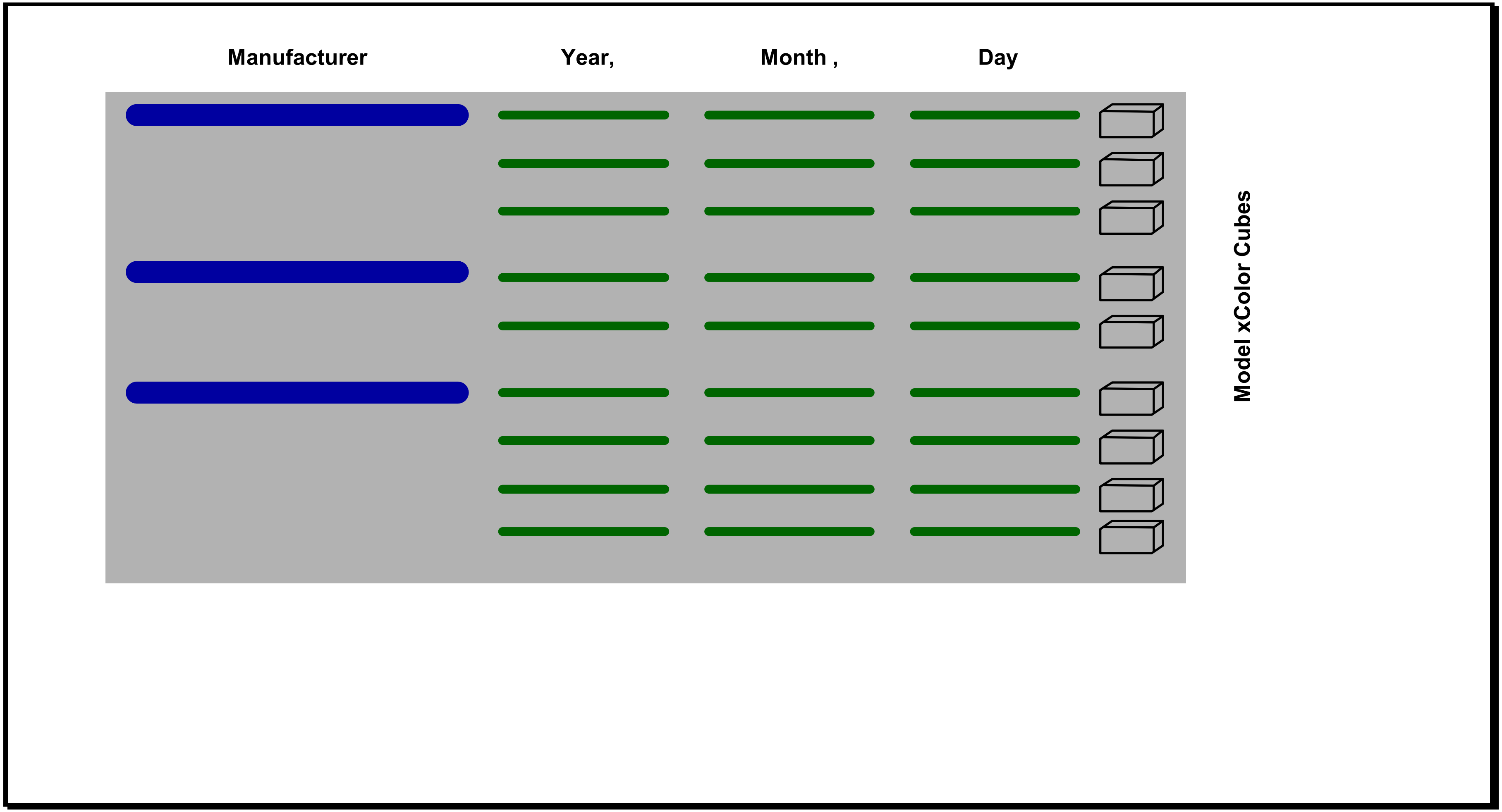
- Devised by Jim Gray et al
  - Jim Gray, a key architect of System R
  - IBM Almaden for several years, then with Tandem, then Digital, then Microsoft
  - Very practical orientation
  - Received Turing award in 1999 !
  - Went missing on a boat trip off California coast in January 2007, no trace found ☹

Aggregation Hierarchy

- GROUP BY <select list 3>
  ROLLUP <select list 2>
  CUBE <select list 1>

```
SELECT manufacturer, year, month, day, color, model, SUM (price) as revenue
FROM Sales
GROUP BY    Manufacturer
    ROLLUP      Year(Time) as Year, Month(Time) as Month, Day(Time) as Day,
        CUBE        Color, Model
```

Figure 5

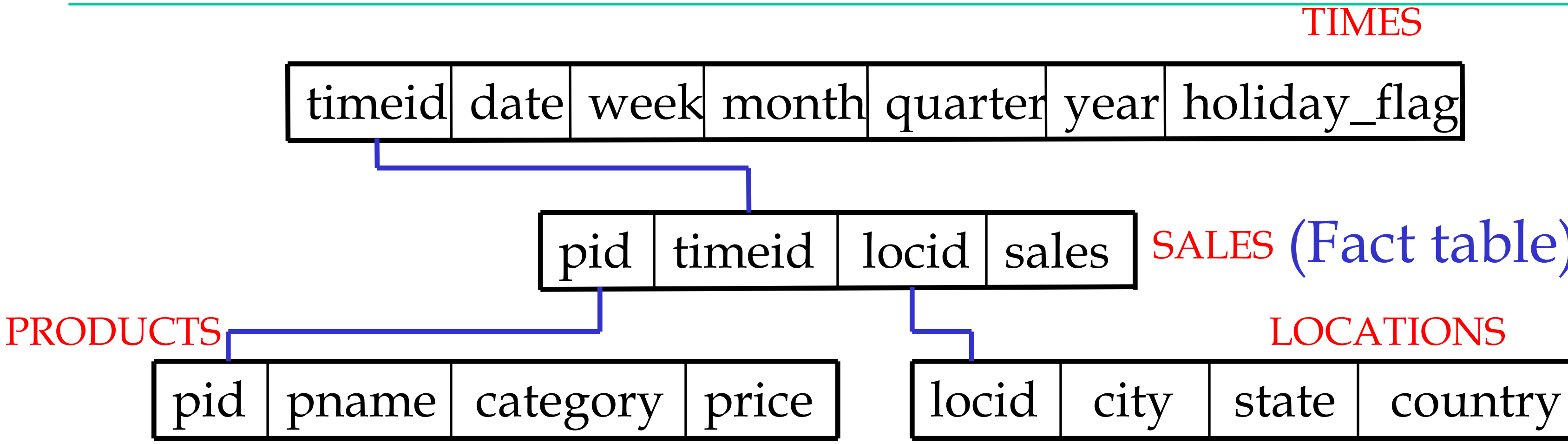


CUBE Research Issues

- Storage Structures for Cubes (ROLAP, MOLAP)
- Schemas for ROLAP Cube (Star, Snowflake)
- Index Structures for Cubes
- Level of Materialization of Cube (next paper)
- ....

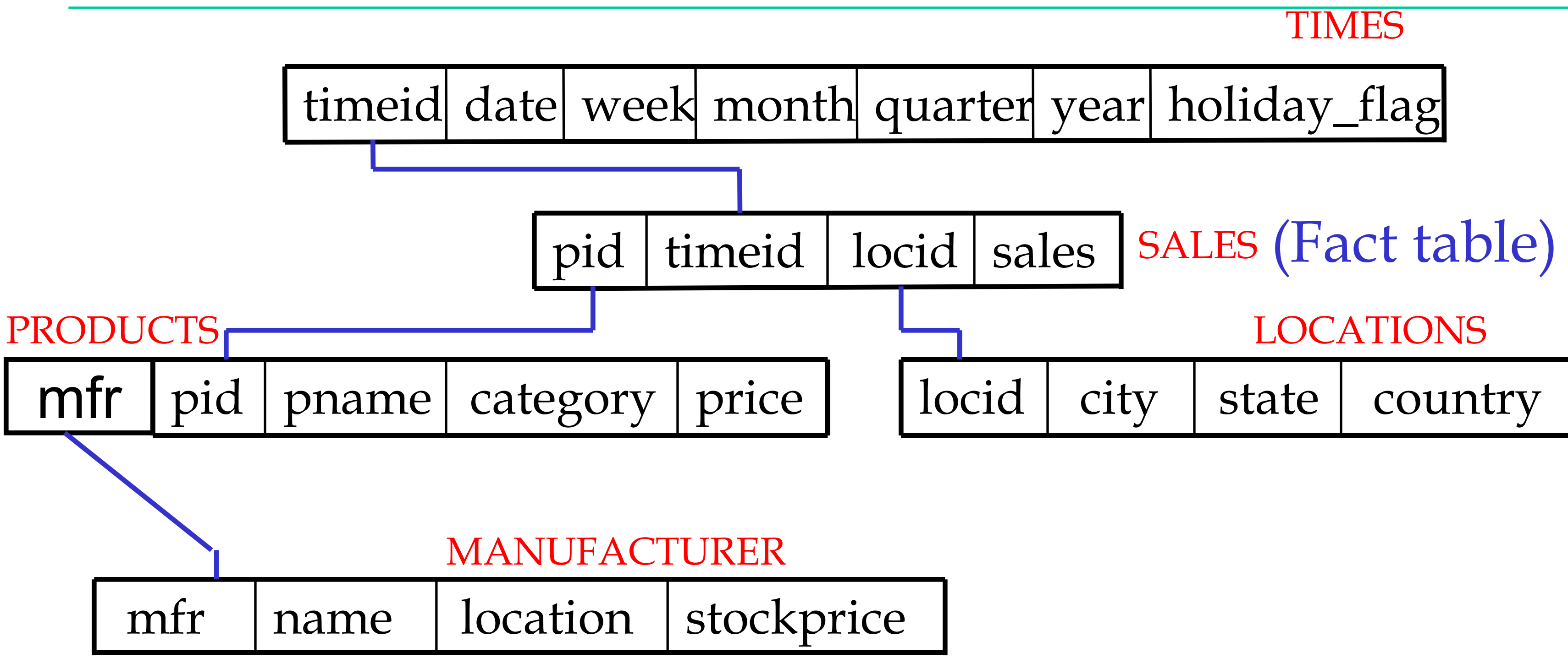
CUBE SCHEMAS

Star Schema



- Fact table in BCNF; dimension tables not normalized.
  - Dimension tables are small; updates/inserts/deletes are rare. So, anomalies less important than good query performance.
- Computing the join of all these relations is called a star join.

Snowflake Schema



- Dimension tables are also normalized to reduce redundancy.

CUBE INDEXES

Indexes

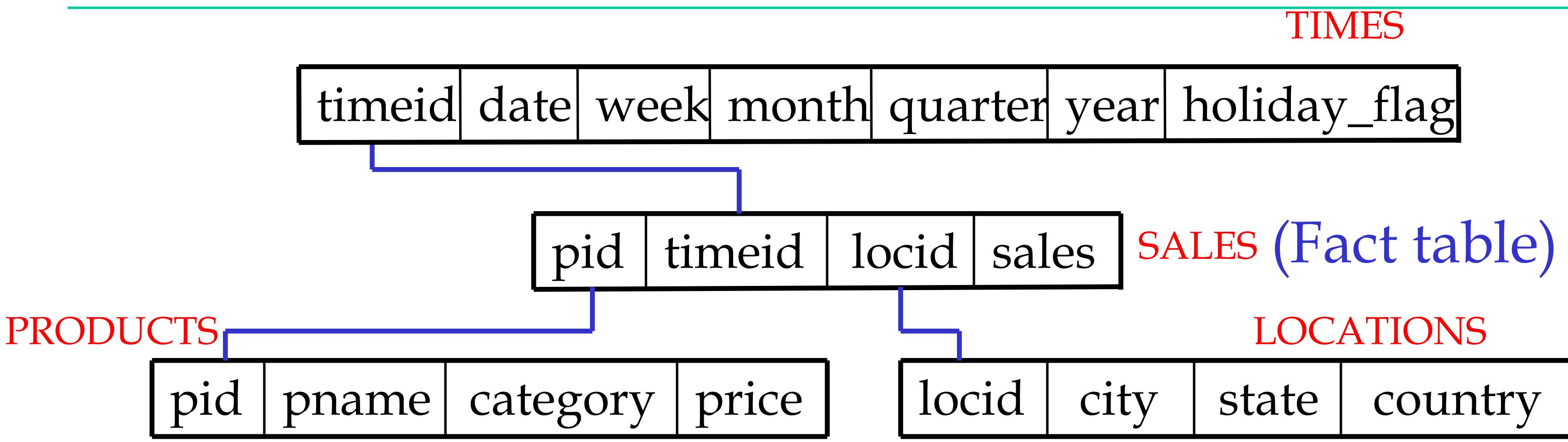
- Joins are common between fact table and dimension tables
- Interactive response is expected, so indexes are required.
- Problems:
  - Space
  - maintenance is not a problem because read-only data

Join Indexes

- Consider the join of Sales, Products, Times, and Locations, possibly with additional selection conditions (e.g., country="India").
  - A join index can be constructed to speed up such joins. The index contains [p,t,l,s] if there are tuples p in Products, t in Times and l in Locations that satisfy the join (with sid) s in Sales .
    E.g. [Beer,March 9, Malleswaram, {2567,3089,156323}]

- Problem: Number of join indexes can grow rapidly.
  - A variant of the idea addresses this problem: For each column with an additional selection (e.g., country), build an index with [c,s] in it if a dimension table tuple with value c in the selection column joins with a Sales tuple with sid s; if indexes are bitmaps, called bitmapped join index.

Bitmapped Join Index



- Consider a query with conditions price=10 and country=“India”. Suppose tuple (with sid) s in Sales joins with a tuple p with price=10 and a tuple l with country =“India”. There are two join indexes; one containing [10,s] and the other [India,s].
- Intersecting these indexes tells us which tuples in Sales are in the join and satisfy the given selection.

Summary

- Decision support is an emerging, rapidly growing subarea of databases.
- Involves the creation of large, consolidated data repositories called data warehouses.
- Warehouses exploited using sophisticated analysis techniques: complex SQL queries and OLAP "multidimensional" queries (influenced by both SQL and spreadsheets).
- New techniques for database design, indexing, view maintenance, and interactive querying need to be supported.

END DATA CUBES

E0 261