Data Warehousing

Warehousing

- Growing industry:
  - $8 billion in 1998
  - $17 billion in 2018
- Range from desktop to huge:
  - Wal-Mart in 1990's
    - 900-CPU, 2,700 disk, 23TB, Teradata system
  - What about 2007?
    - Wal-Mart Plans for Its 4PB Data Warehouse
    - 100 billion rows in tables
    - 276 million records/day from POS systems
- Lots of buzzwords, hype
  - Data cube, rollup, drill-down, slice/dice, ...

What is a Warehouse?

- Collection of diverse data
  - subject oriented
  - aimed at decision maker
  - often a copy of operational data
  - with value-added data (e.g., summaries, history)
  - integrated
  - time-varying
  - non-volatile

more

What is a Warehouse?

- Collection of tools
  - gathering data
  - cleansing, integrating, ...
  - querying, reporting, analysis
  - data mining
  - monitoring, administering warehouse

Data Warehouse—Subject-Oriented

- Organized around major subjects, such as customer, product, sales
- Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing
- Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process

Data Warehouse—Integrated

- Constructed by integrating multiple, heterogeneous data sources
  - relational databases, flat files, on-line transaction records
- Data cleaning and data integration techniques are applied.
  - Ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources
  - When data is moved to the warehouse, it is converted.

Data Warehouse—Time Variant

- The time horizon for the data warehouse is significantly longer than that of operational systems
  - Operational database: current value data
  - Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)

Data Warehouse—Nonvolatile

- A physically separate store of data transformed from the operational environment
- Operational update of data does not occur in the data warehouse environment
  - Does not require transaction processing, recovery, and concurrency control mechanisms
  - Requires only two operations in data accessing:
    - *initial loading of data* and *access of data*

What is Data Warehouse?

- In sum, data warehouse is a semantically consistent data store that serves as a physical implementation of a decision support data model and stores the information on which an enterprise needs to make strategic decisioins

Conventional Query Tools

- Ad-hoc queries and reports using conventional database tools
  - E.g. Access queries.
- Typical database designs include fixed sets of reports and queries to support them
  - The end-user is often not given the ability to do ad-hoc queries
  - Data structure is often very complex (normalized database)

Different Goal

- Aggregation, summarization and exploration
- Of historical data
- To help management make informed decisions

Product	Branch	Time	Price
Coke (0.5 gallon)	Convoy Street	2006-03-01 09:00:01	$1.00
Pepsi (0.5 gallon)	UTC	2006-03-01 09:00:01	$1.03
Coke (1 gallon)	UTC	2006-03-01 09:00:02	$1.50
Altoids	Costa Verde	2006-03-01 09:01:33	$0.30
...			

- Find the total sales for each product and month
- Find the percentage change in the total monthly sales for each product

Different Requirements

- OLTP – On-Line Transaction Processing
- OLAP – On-Line Analytical Processing

	OLTP	OLAP
Tasks	Day to day operation	High level decision support
Size of database	Gigabytes	Terabytes
Time span	Recent, up-to-date	Spanning over months / years
Size of working set	Tens of records, accessed through primary keys	Consolidated data from multiple databases
Workload	Structured / repetitive	Ad-hoc, exploratory queries
Performance	Transaction throughput	Query latency

OLTP vs. OLAP

- On-Line Transaction Processing (OLTP):
  - technology used to perform updates on operational or transactional systems (e.g., point of sale systems)

- On-Line Analytical Processing (OLAP):
  - technology used to perform complex analysis of the data in a data warehouse

    OLAP is a category of software technology that enables analysts, managers, and executives to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information that has been transformed from raw data to reflect the dimensionality of the enterprise as understood by the user.
    [source: OLAP Council: www.olapcouncil.org]

OLTP vs. OLAP

OLTP

- Mostly updates
- Many small transactions
- Mb-Tb of data
- Raw data
- Clerical users/clients/customers
- Up-to-date data
- Consistency, recoverability-critical

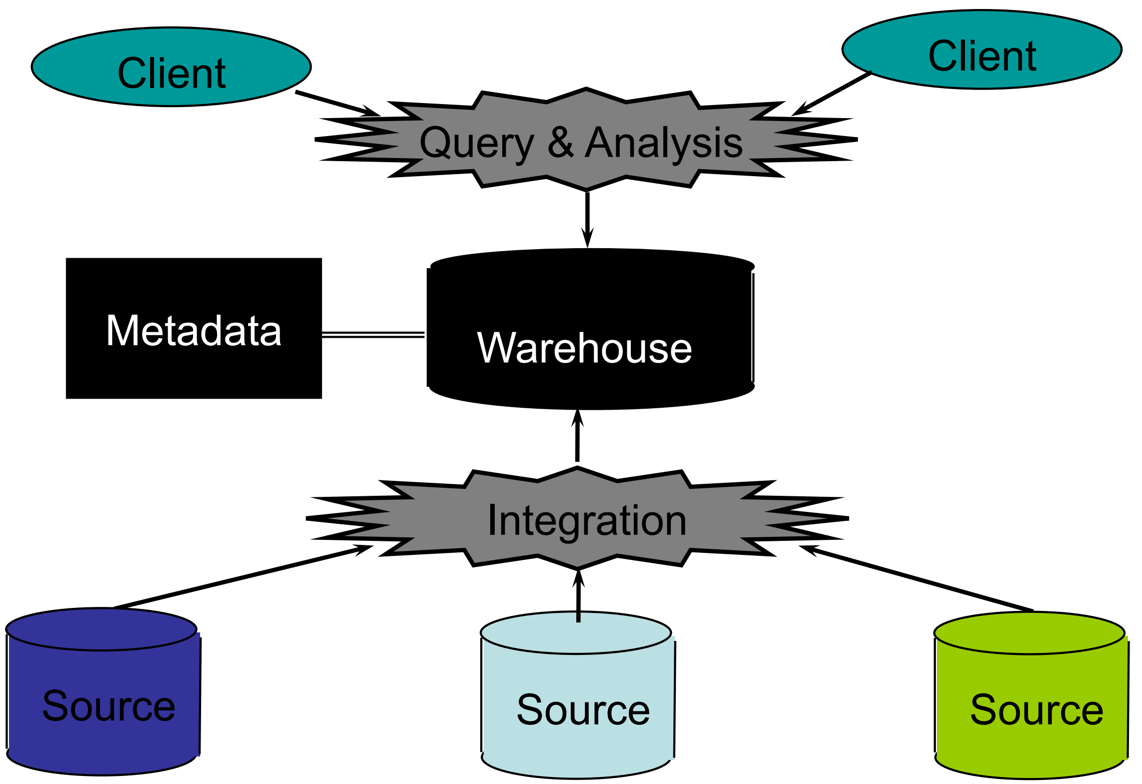
OLAP

- Mostly reads
- Queries long, complex
- Gb-Tb of data
- Summarized, consolidated data
- Decision-makers, analysts as users
- Historical data
- Query performance critical

OLAP Conceptual Data Model

- Goal of OLAP is to support ad-hoc querying for the business analyst
- Business analysts are familiar with spreadsheets
- Extend spreadsheet analysis model to work with warehouse data
- *Multidimensional* view of data is the foundation of OLAP

Warehouse Architecture



Data Marts

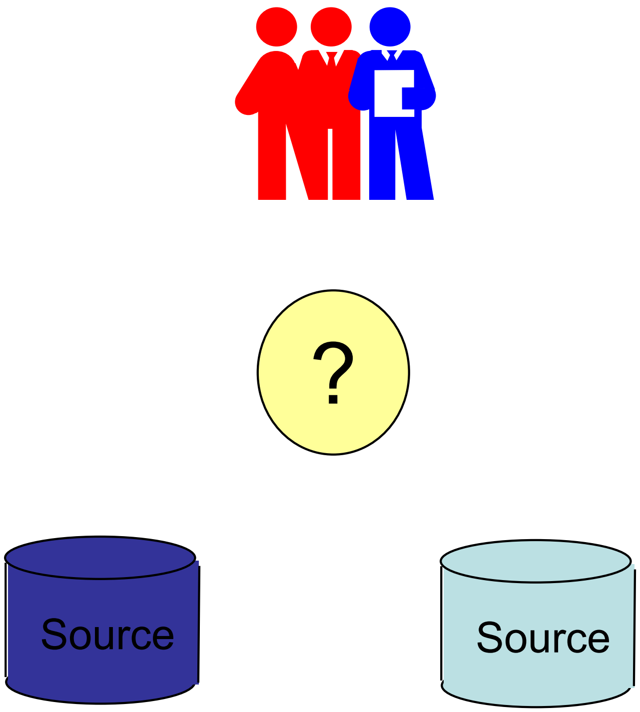
- Smaller warehouses
- Spans part of organization
  - e.g., marketing (customers, products, sales)
- Do not require enterprise-wide consensus
  - but long term integration problems?

What are some of the challenges?

- Different schema in different databases
- Data may be inconsistent
- Data is very large
- Dynamic, exploratory queries
- Connecting data together may be hard
  - Data integration

Why a Warehouse?

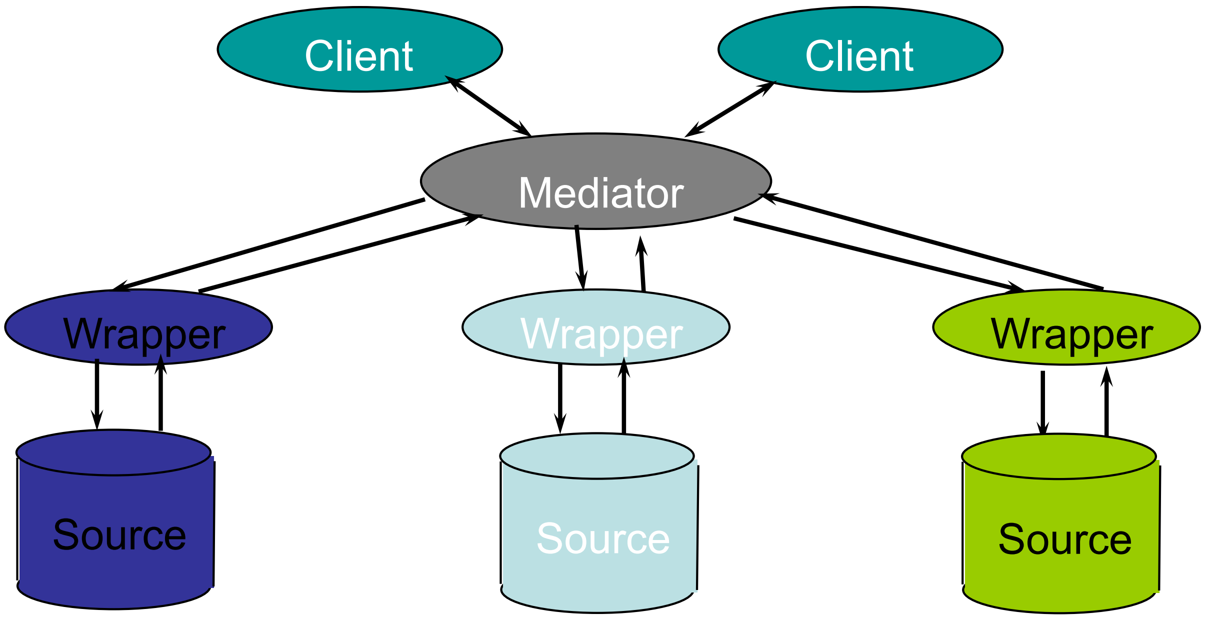
- Two Approaches:
  - Query-Driven (Lazy)
  - Warehouse (Eager)
    - Update driven



Data Integration

- Define a global model for integrated data
  - Domain model
- Identify source models
- Create mappings from source to domain models
  - Global as view (GAV) or Local as view (LAV)
- Optimize queries for distribution and data specifics

Query-Driven Approach



Warehousing

- Define domain model
- Identify source models
- Define procedure for mapping from source model to domain model
  - Extract/Translate/Load (ETL)
  - Extract/Load/Translate (ELT)
- Copy (load) data into dedicated analytic database

Advantages of Warehousing

- High query performance
- Queries not visible outside warehouse
- Local processing at sources unaffected
- Can operate when sources unavailable
- Can query data not stored in a DBMS
- Extra information at warehouse
  - Modify, summarize (store aggregates)
  - Add historical information

Advantages of Query-Driven

- No need to copy data
  - less storage
  - no need to purchase data
- More up-to-date data
- Query needs can be unknown
- Only query interface needed at sources
- May be less draining on sources

Tools: OLAP Servers

- Support multidimensional OLAP queries
- Often characterized by how the underlying data stored
- Relational OLAP (ROLAP) Servers
  - Data stored in relational tables
  - Examples: Microstrategy Intelligence Server, MetaCube (Informix/IBM)
- Multidimensional OLAP (MOLAP) Servers
  - Data stored in array-based structures
  - Examples: Hyperion Essbase, Fusion (Information Builders)
- Hybrid OLAP (HOLAP)
  - Examples: PowerPlay (Cognos), Brio, Microsoft Analysis Services, Oracle Advanced Analytic Services

MOLAP vs. OLAP

- Commercial offerings of both types are available
- In general, MOLAP is good for smaller warehouses and is optimized for canned queries
- In general, ROLAP is more flexible and leverages relational technology on the data server and uses a ROLAP server as intermediary. May pay a performance penalty to realize flexibility

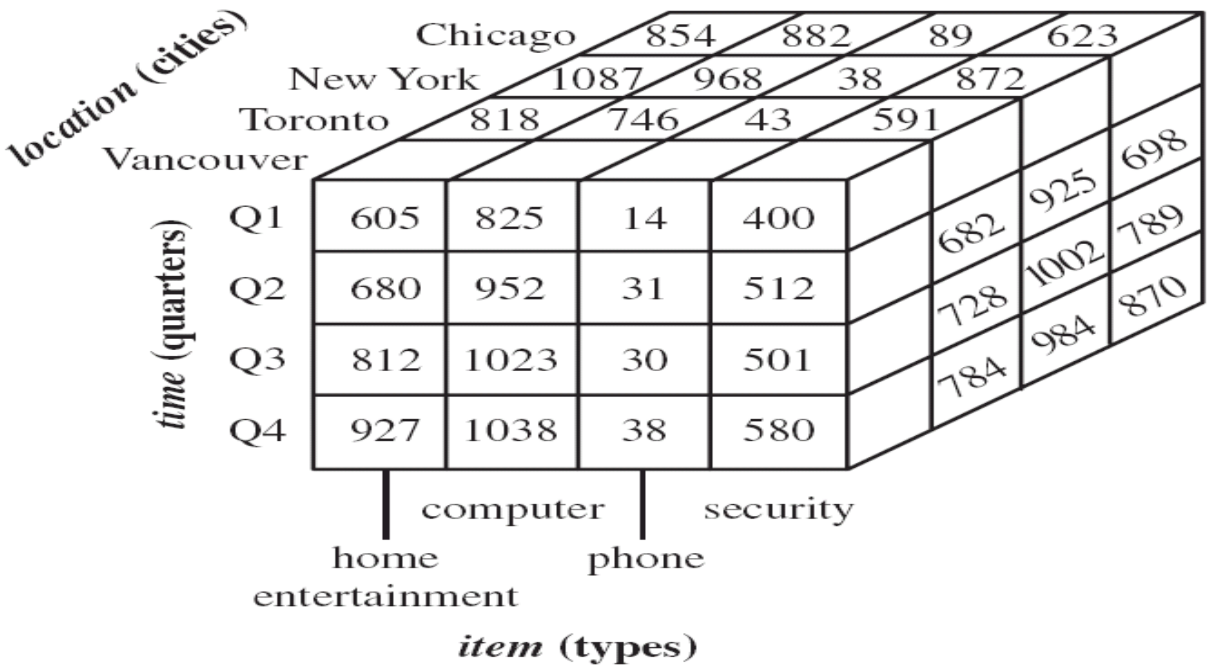
A multi-dimensional data model

- A data warehouse is based on a *multidimensional data model* which views data in the form of a data cube

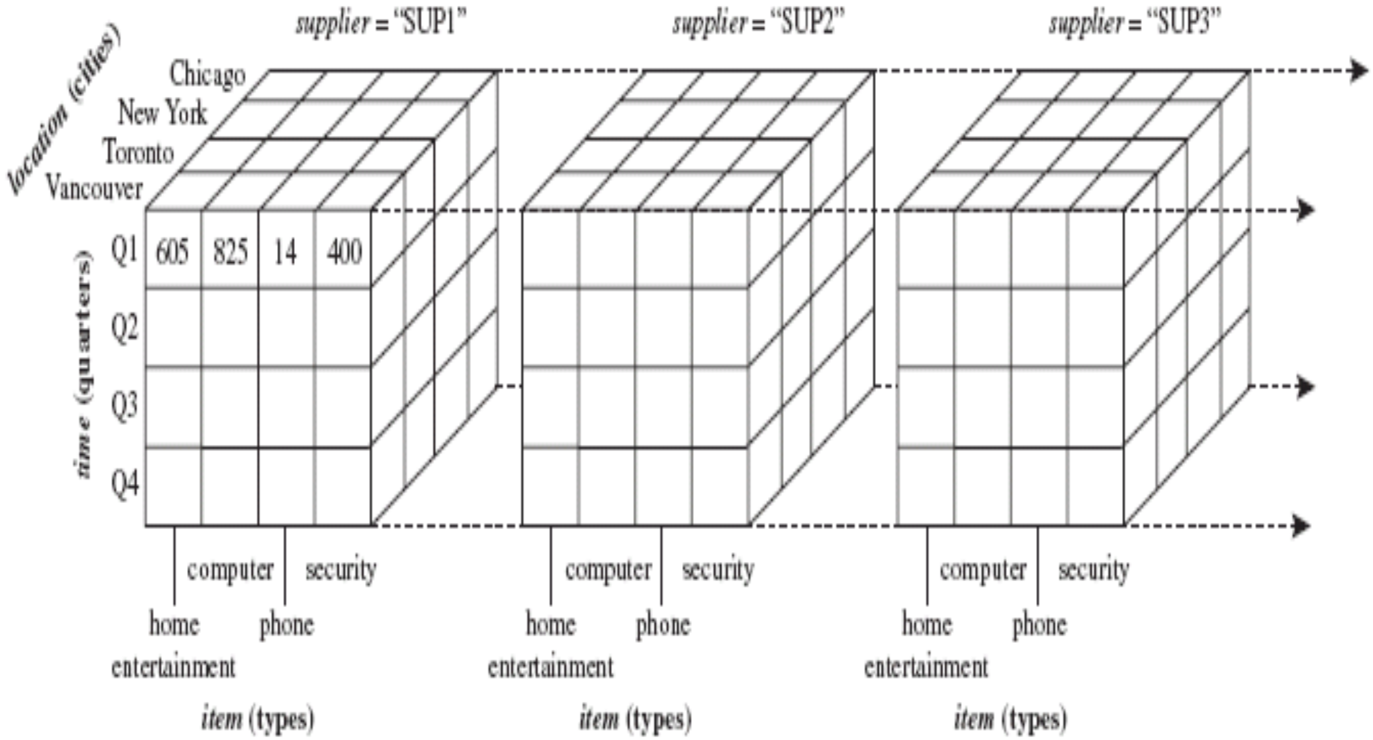
Data cube

- A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions
- Suppose ALLELETRONICS create a *sales* data warehouse with respect to dimensions
  - Time
  - Item
  - Location

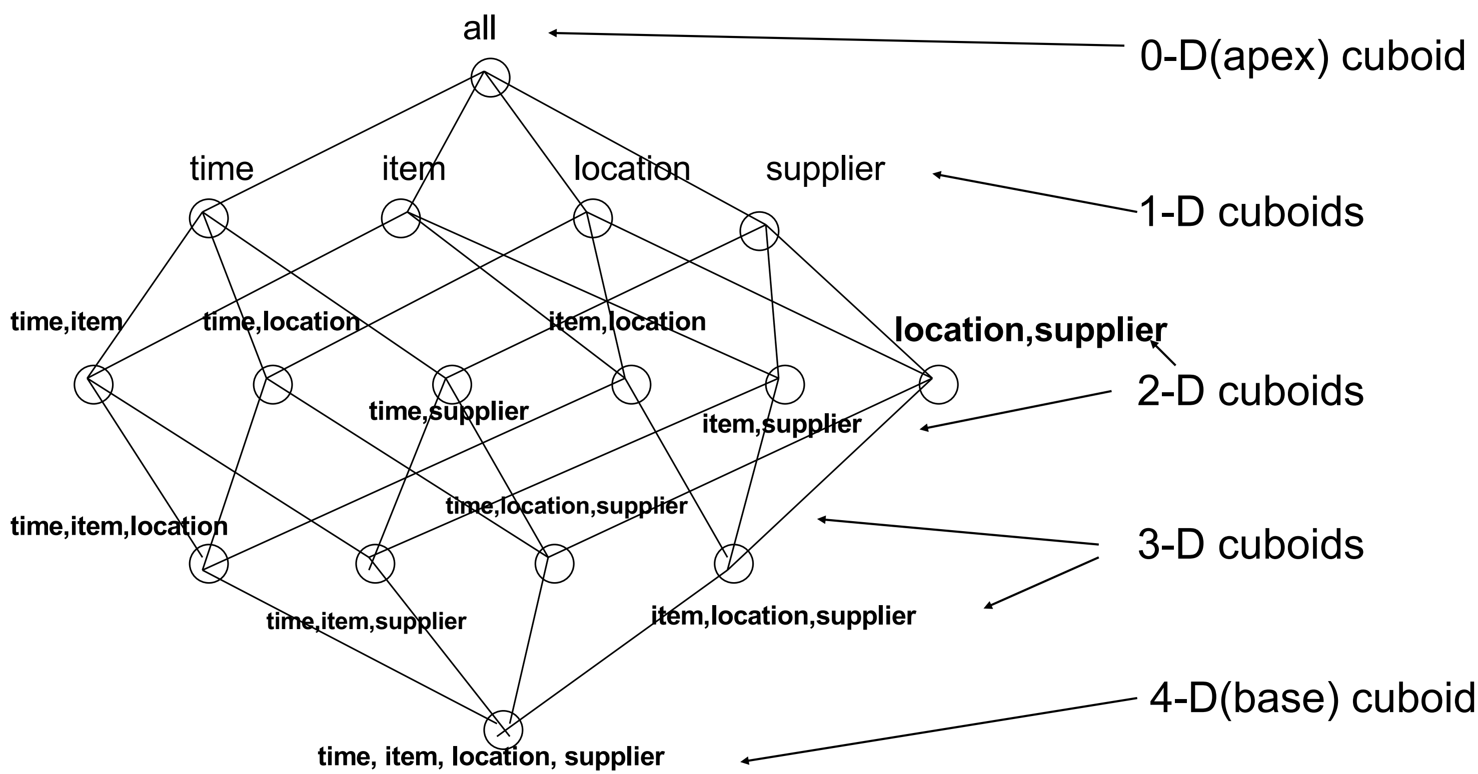
3D Data cube Example



4D Data cube Example



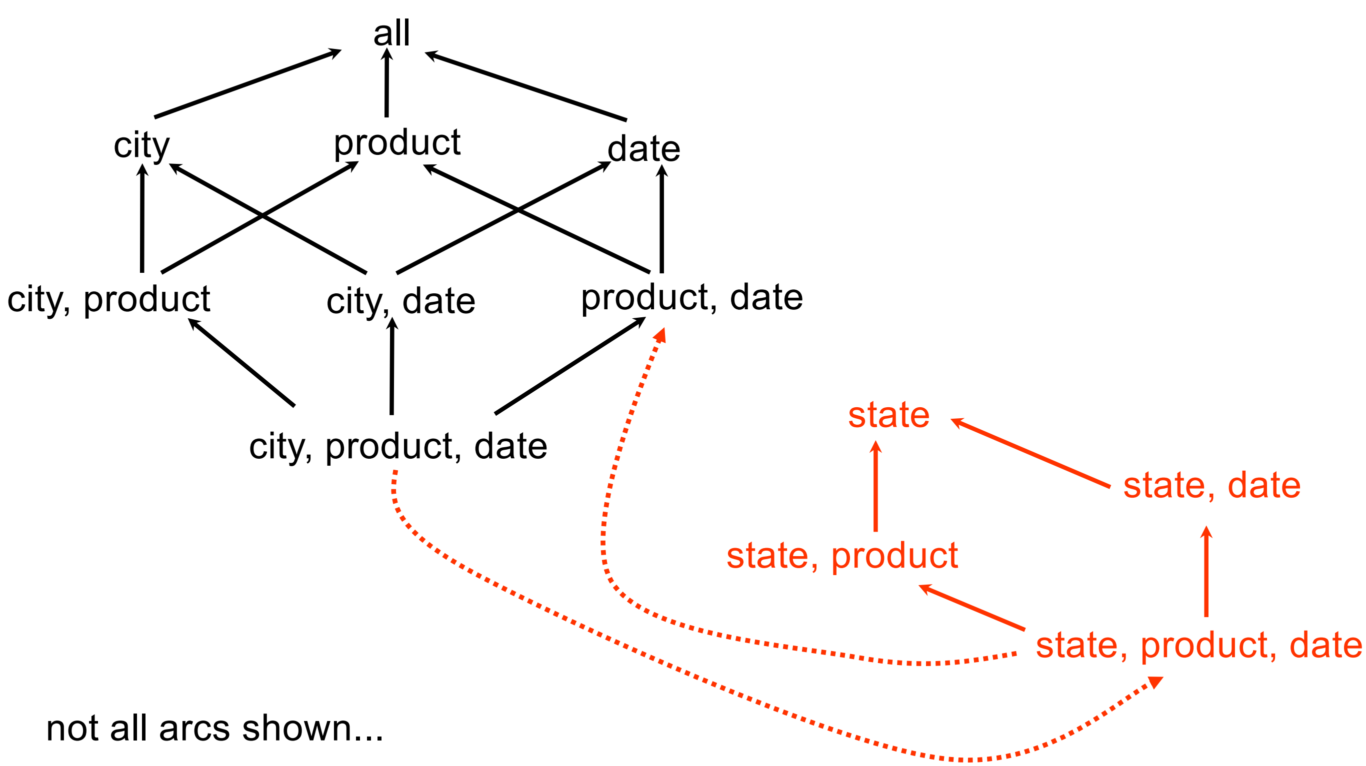
Cube: A Lattice of Cuboids



Practice Question

- What is a 5D cube looks like?

Dimension Hierarchies

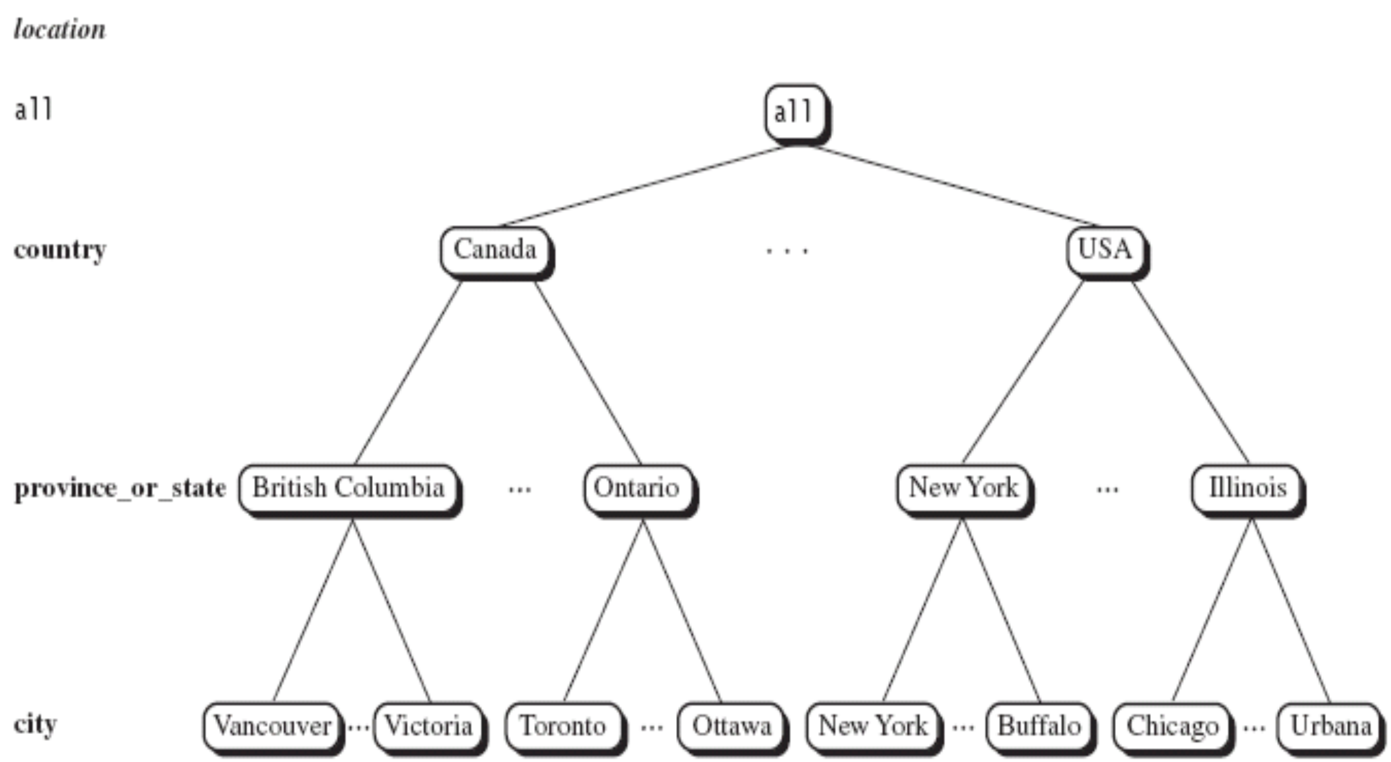


not all arcs shown...

Concept Hierarchies

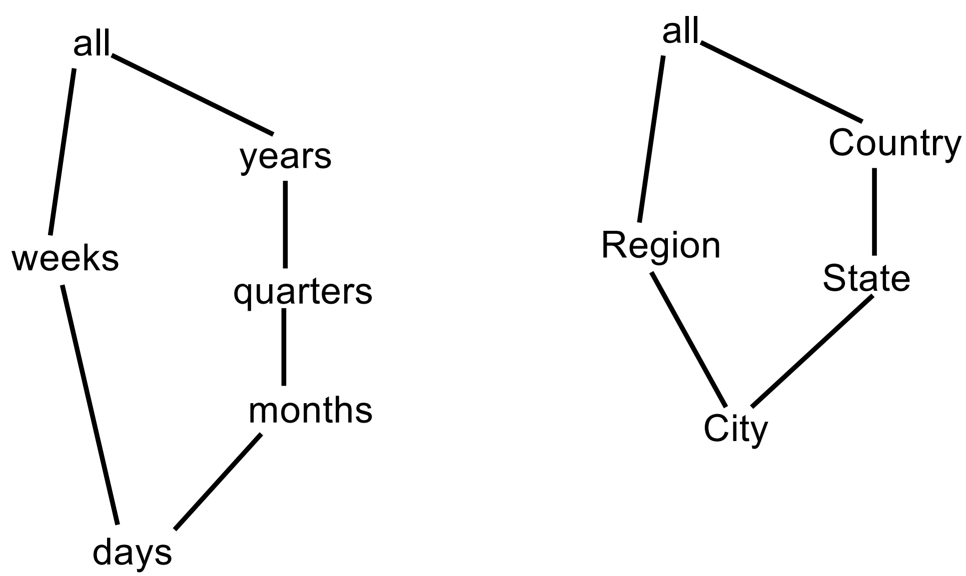
- A **Concept Hierarchy** defines a sequence of mappings from a set of low-level concepts to high-level
- Consider a concept hierarchy for the dimension **“Location”**

Concept Hierarchies



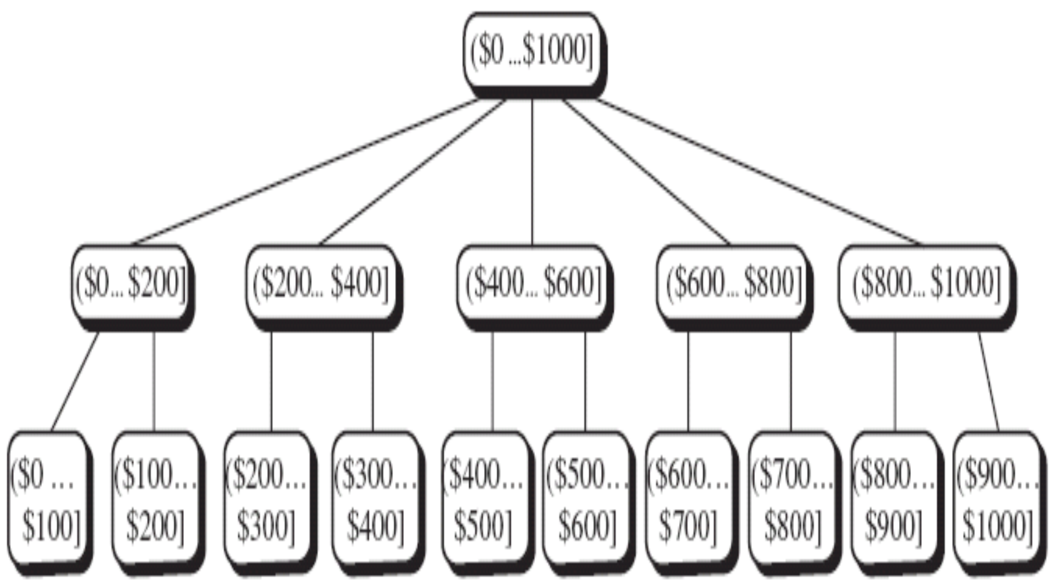
Concept Hierarchies

- Many concept hierarchies are implicit within the database system



Concept Hierarchies

- Concept hierarchies may also be defined by grouping values for a given dimension or attribute, resulting in a **set-grouping hierarchy**



Typical OLAP Operations

- Roll up (drill-up): summarize data
  - *by climbing up hierarchy or by dimension reduction*
  - dimension reduction: e.g., total sales by city
  - summarization over aggregate hierarchy: e.g., total sales by city and year -> total sales by region and by year
- Drill down (roll down): reverse of roll-up
  - *from higher level summary to lower level summary or detailed data, or introducing new dimensions*
  - e.g., (sales - expense) by city, top 3% of cities by average income

Typical OLAP Operations

- Slice and dice: *project and select*
  - e.g., sales where city = Palo Alto and date = 1/15/96
- Pivot (rotate):
  – *reorient the cube, visualization, 3D to series of 2D planes*

Data Cube

Fact table view:

sale	prodId	storeId	amt
	p1	s1	12
	p2	s1	11
	p1	s3	50
	p2	s2	8

Multi-dimensional cube:

	s1	s2	s3
p1	12		50
p2	11	8	

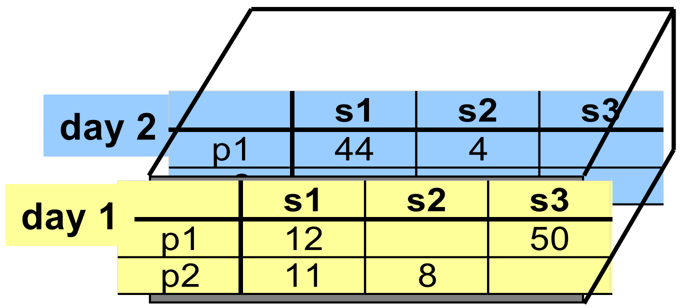
dimensions = 2

3-D Cube

Fact table view:

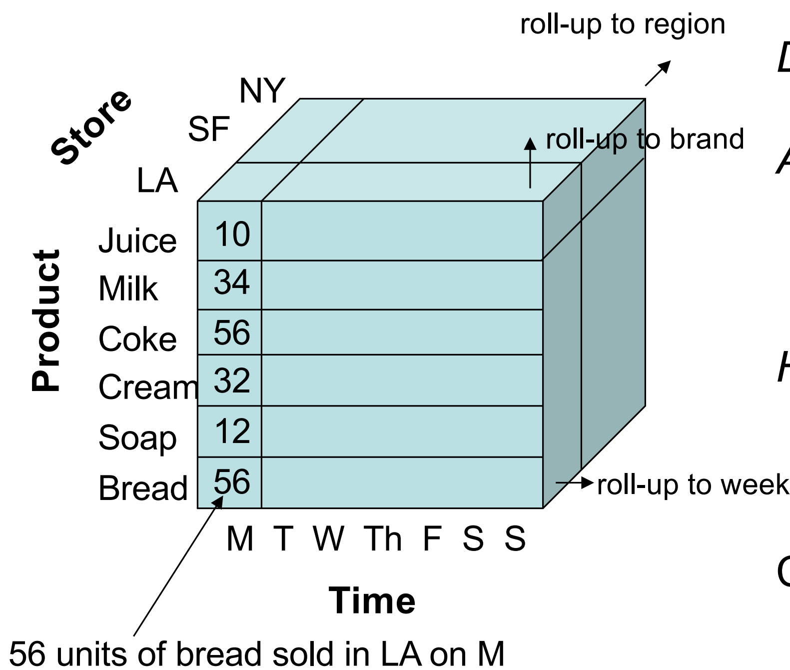
sale	prodId	storeId	date	amt
	p1	s1	1	12
	p2	s1	1	11
	p1	s3	1	50
	p2	s2	1	8
	p1	s1	2	44
	p1	s2	2	4

Multi-dimensional cube:



dimensions = 3

Example



Dimensions:

Time, Product, Store

Attributes:

Product (upc, price, …)

Store …

…

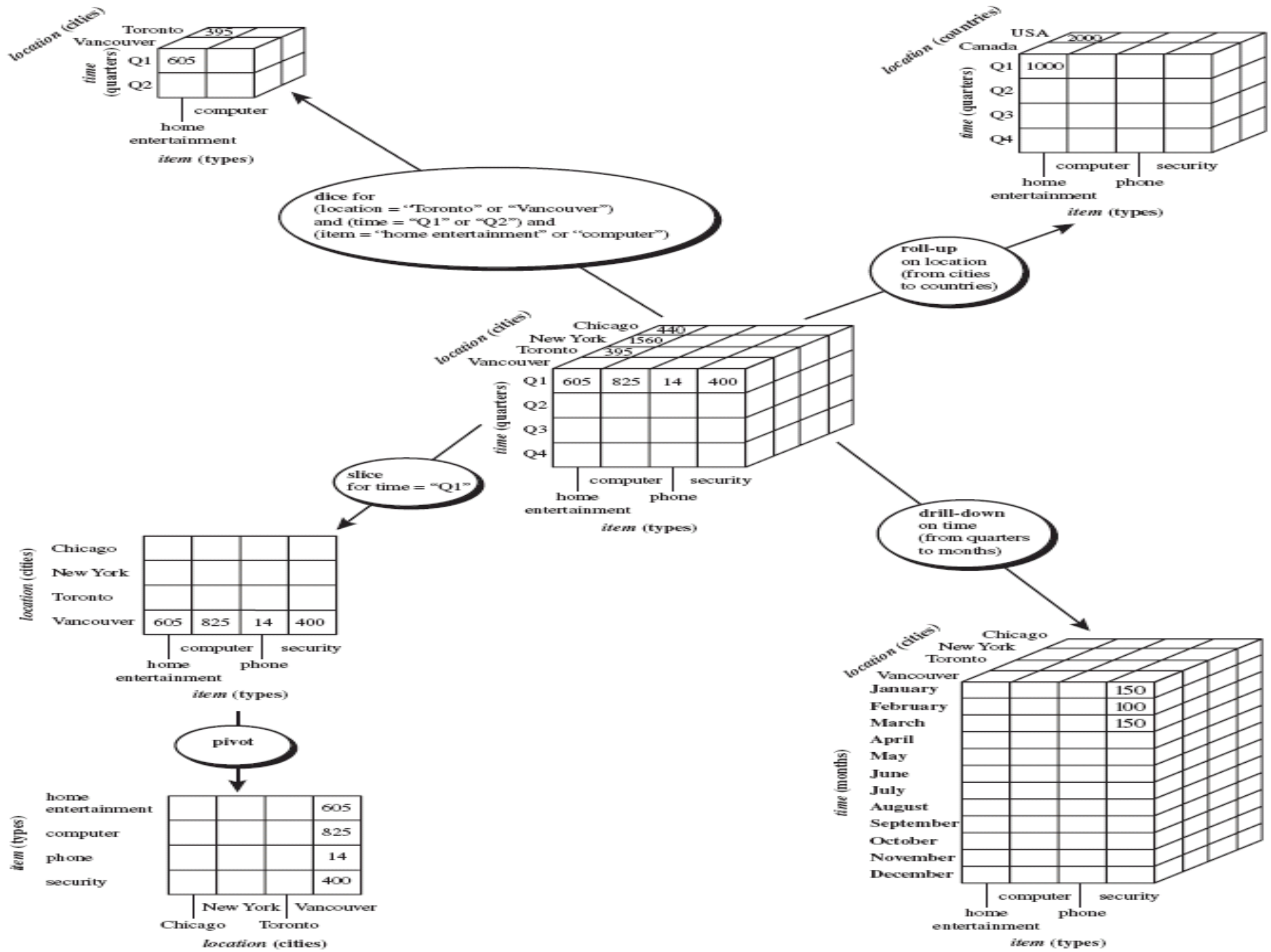
Hierarchies:

Product → Brand → …

Day → Week → Quarter

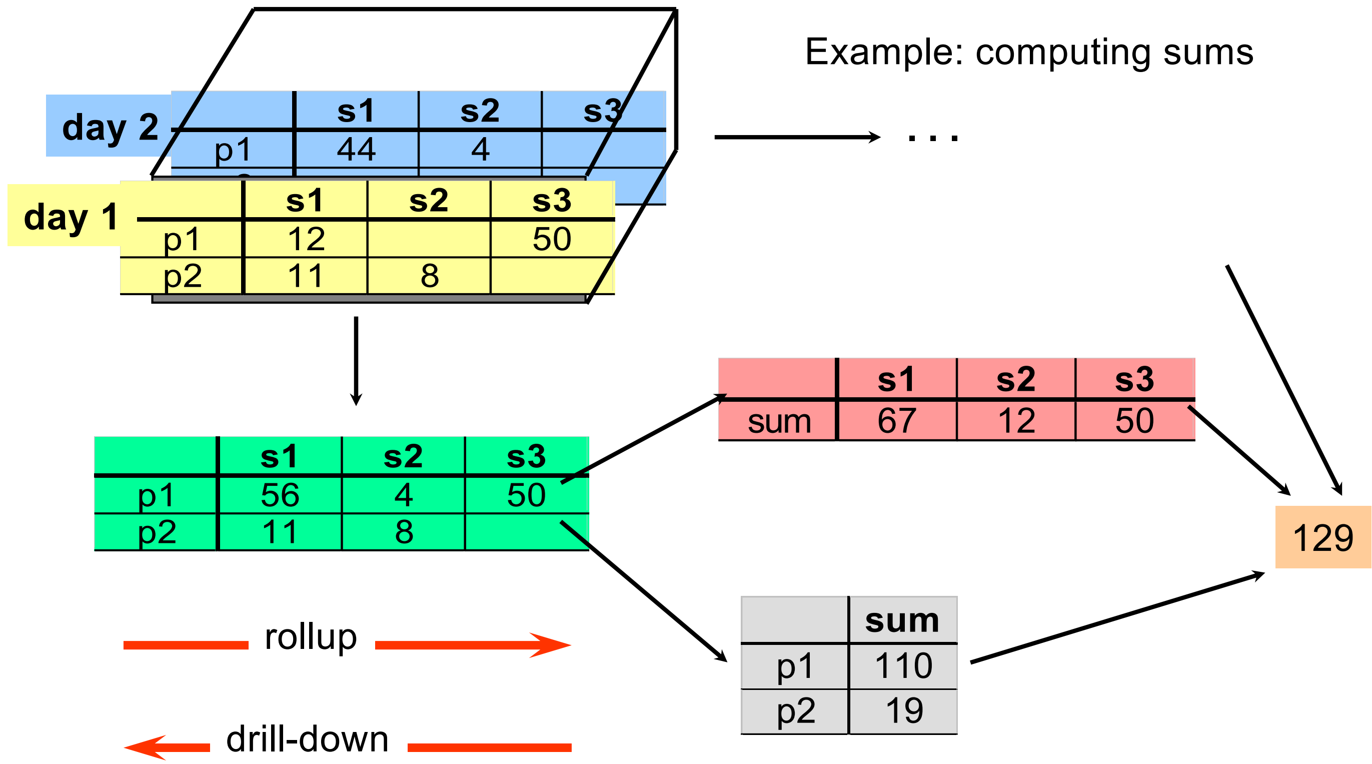
Store → Region →

Country

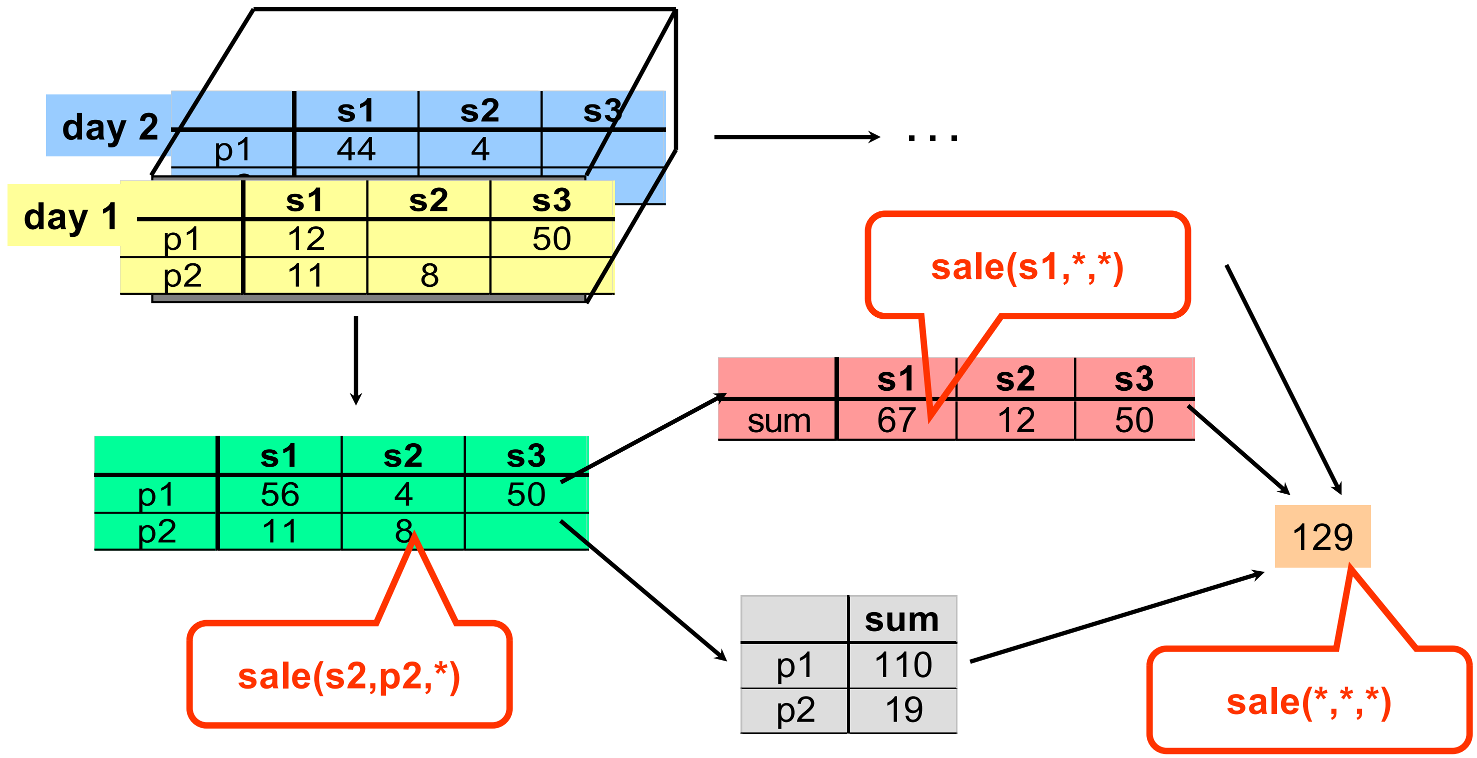


location (cities)
Toronto
395
Vancouver
time (quarters)
Q1
605
Q2
computer
home entertainment
item (types)
dice for
(location = "Toronto" or "Vancouver")
and (time = "Q1" or "Q2") and
(item = "home entertainment" or "computer")
location (countries)
USA
2000
Canada
Q1
1000
Q2
Q3
Q4
time (quarters)
computer
security
home entertainment
phone
item (types)
roll-up
on location
(from cities
to countries)
location (cities)
Chicago
440
New York
1560
Toronto
395
Vancouver
Q1
605
825
14
400
Q2
Q3
Q4
time (quarters)
computer
security
home entertainment
phone
item (types)
slice
for time = "Q1"
Chicago
New York
Toronto
Vancouver
605
825
14
400
location (cities)
computer
security
home entertainment
phone
item (types)
pivot
home entertainment
605
computer
825
phone
14
security
400
item (types)
New York
Vancouver
Chicago
Toronto
location (cities)
drill-down
on time
(from quarters
to months)
location (cities)
Chicago
New York
Toronto
Vancouver
January
150
February
100
March
150
April
May
June
July
August
September
October
November
December
time (months)
computer
security
home entertainment
phone
item (types)

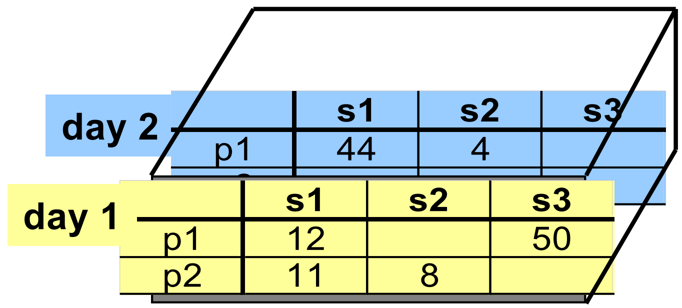
Cube Aggregation: Roll-up



Cube Operators for Roll-up



Aggregation Using Hierarchies



	region A	region B
p1	56	54
p2	11	8

store

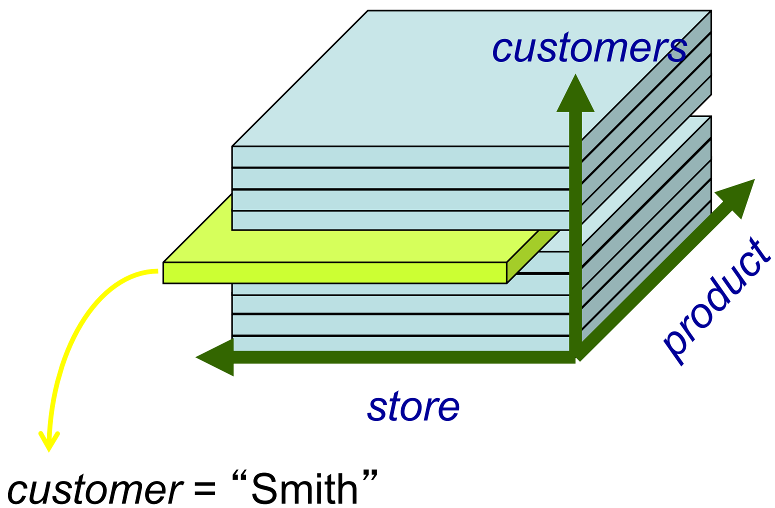
region

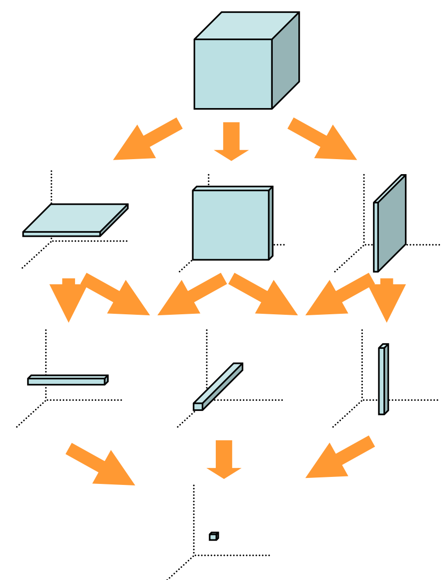
country

(store s1 in Region A;
stores s2, s3 in Region B)

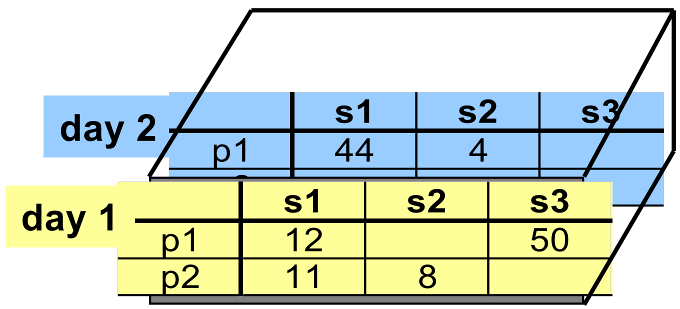
Slice and Dice Queries

- <u>Slice and Dice</u>: select and project on one or more dimensions





Slicing



TIME = day 1

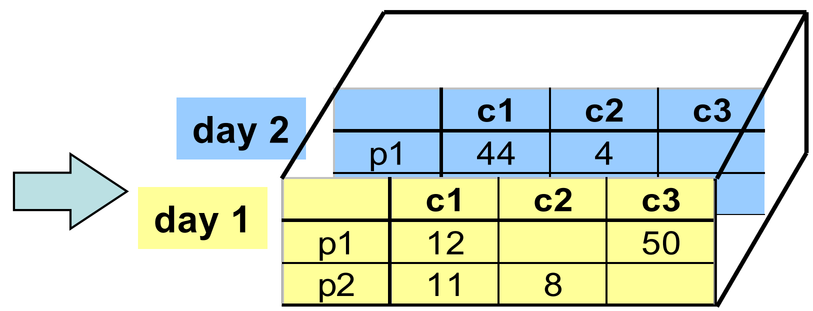
	s1	s2	s3
p1	12		50
p2	11	8	

Pivoting

Fact table view:

sale	prodId	storeId	date	amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4

Multi-dimensional cube:



	c1	c2	c3
p1	56	4	50
p2	11	8	

Slicing & Pivoting

	Sales ($ millions)			
	Products	Time		
		d1	d2	
Store s1	Electronics	$5.2		
	Toys	$1.9		
	Clothing	$2.3		
	Cosmetics	$1.1		
Store s2	Electronics	$8.9		
	Toys	$0.75		
	Clothing	$4.6		
	Cosmetics	$1.5		

	Sales ($ millions)			
	Products	d1		
		Store s1	Store s2	
Store s1	Electronics	$5.2	$8.9	
	Toys	$1.9	$0.75	
	Clothing	$2.3	$4.6	
	Cosmetics	$1.1	$1.5	
Store s2	Electronics			
	Toys			
	Clothing			

Relational DBMS as Warehouse Server

- Schema design
- Specialized scan, indexing and join techniques
- Handling of aggregate views (querying and materialization)
- Supporting query language extensions beyond SQL
- Complex query processing and optimization
- Data partitioning and parallelism

Conceptual Modeling of Data Warehouses

- The most popular data model for a data warehouse is a multi-dimensional model
- Such a model can exist in the form of:
  - Star schema
  - Snowflake schema
  - Fact constellations

Conceptual Modeling of Data Warehouses

- Star schema: A fact table in the middle connected to a set of dimension tables
- It contains:
  - A large central table (fact table)
  - A set of smaller attendant tables (dimension table), one for each dimension

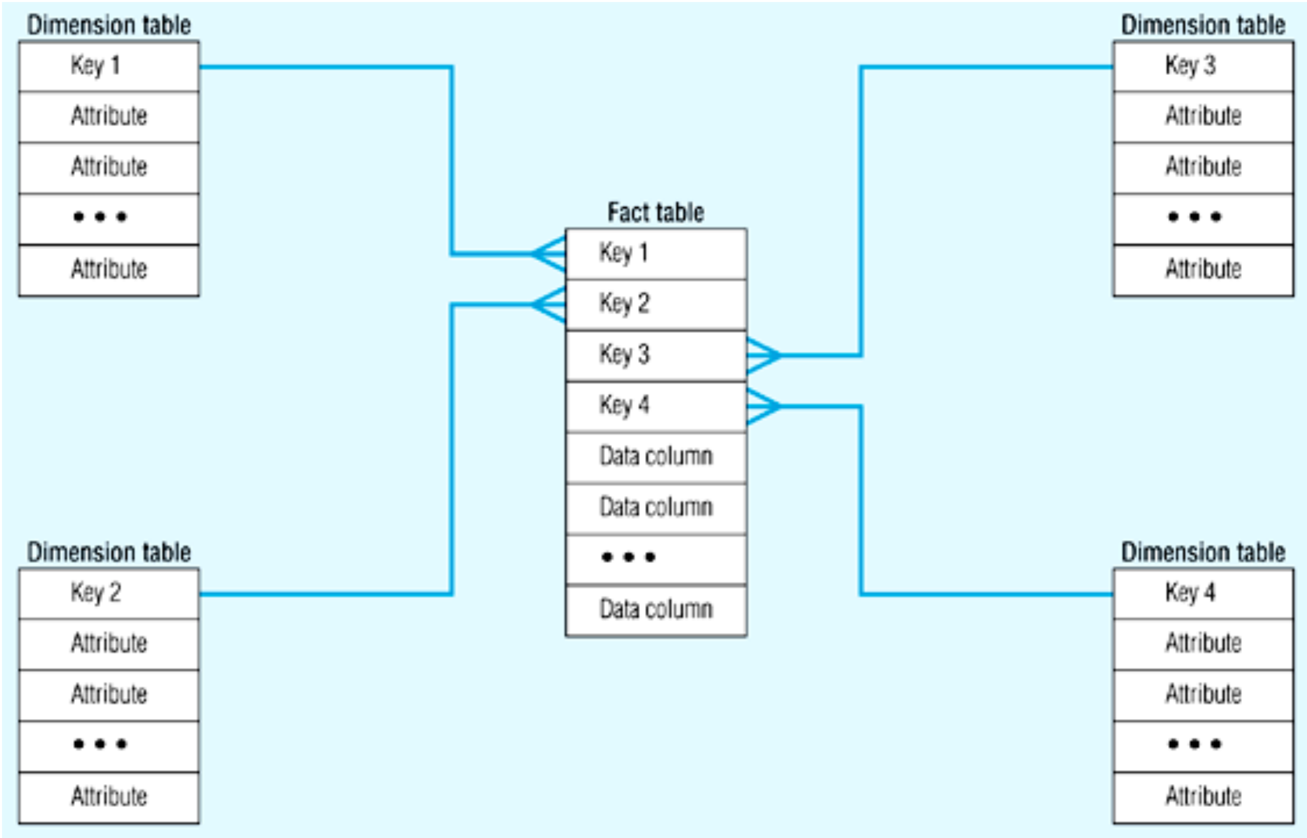
Star Schema

- Typical design for a Data Warehouse for Decision
  - Particularly suited to ad-hoc queries
  - Dimensional data separate from fact or event data
- Fact tables contain factual or quantitative data
- Dimension tables hold data about the subjects of the organization
- Typically there is one Fact table with multiple dimension tables

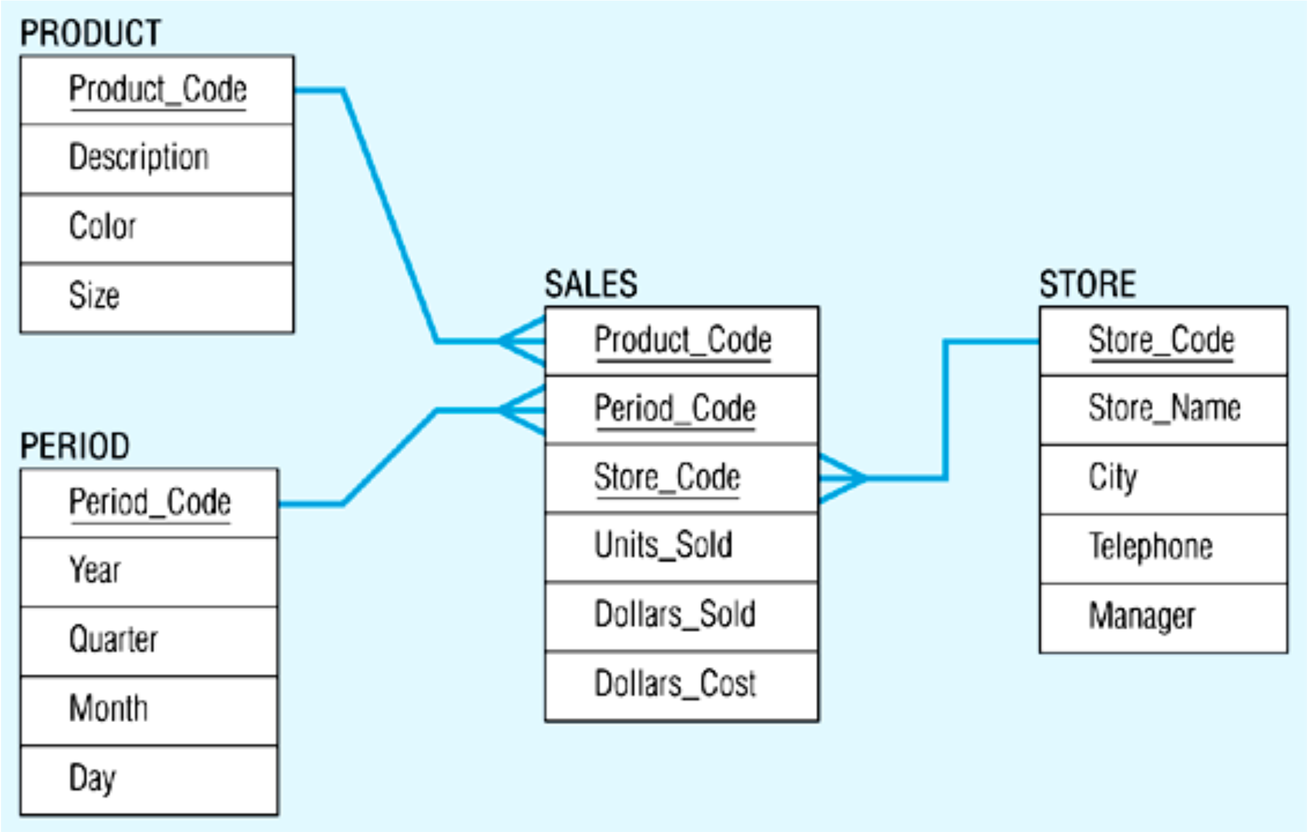
The “Classic” Star Schema

- A relational model with a one-to-many relationship between dimension table and fact table.
- A single fact table, with detail and summary data
- Fact table primary key has only one key column per dimension
- Each dimension is a single table, highly denormalized
- **Benefits**: Easy to understand, intuitive mapping between the business entities, easy to define hierarchies, reduces # of physical joins, low maintenance, very simple metadata
- **Drawbacks**: Summary data in the fact table yields poorer performance for summary levels, huge dimension tables a problem

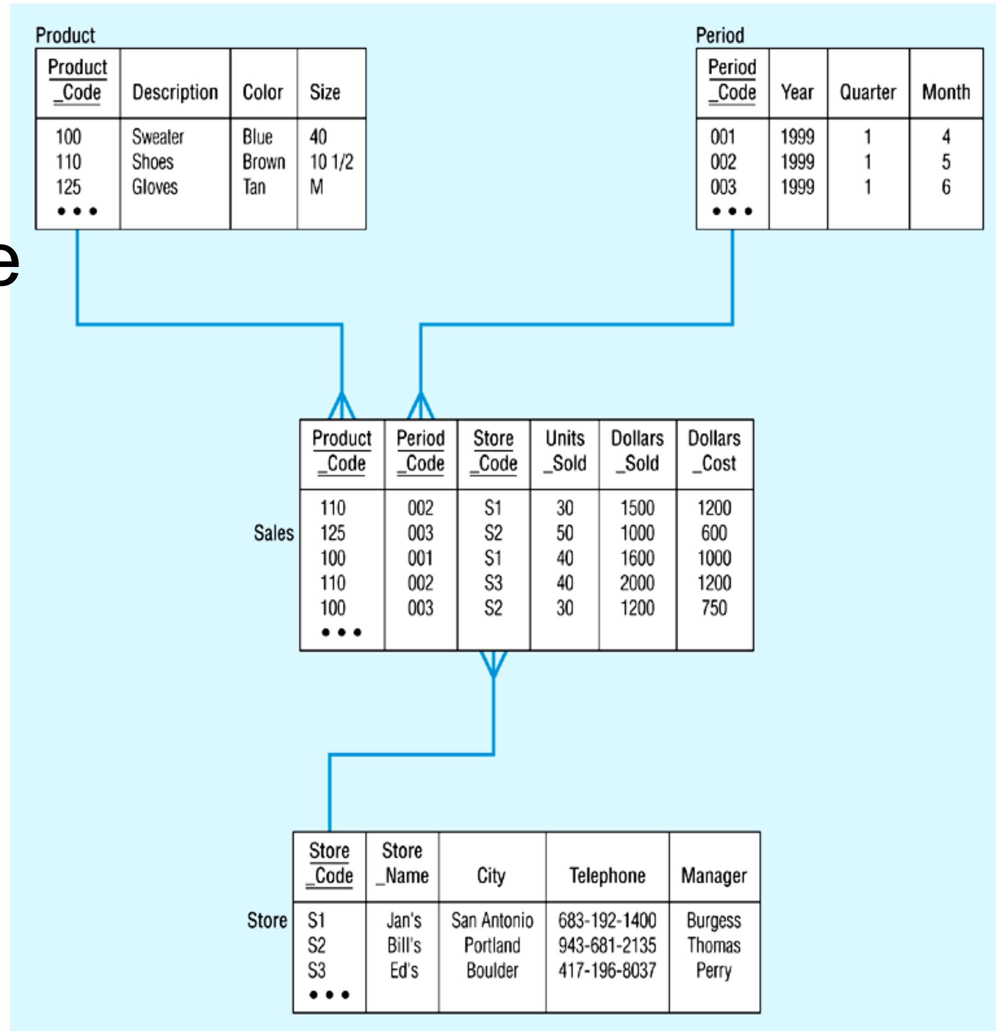
Star Schema (in RDBMS)



Star Schema Example



Star Schema with Sample Data



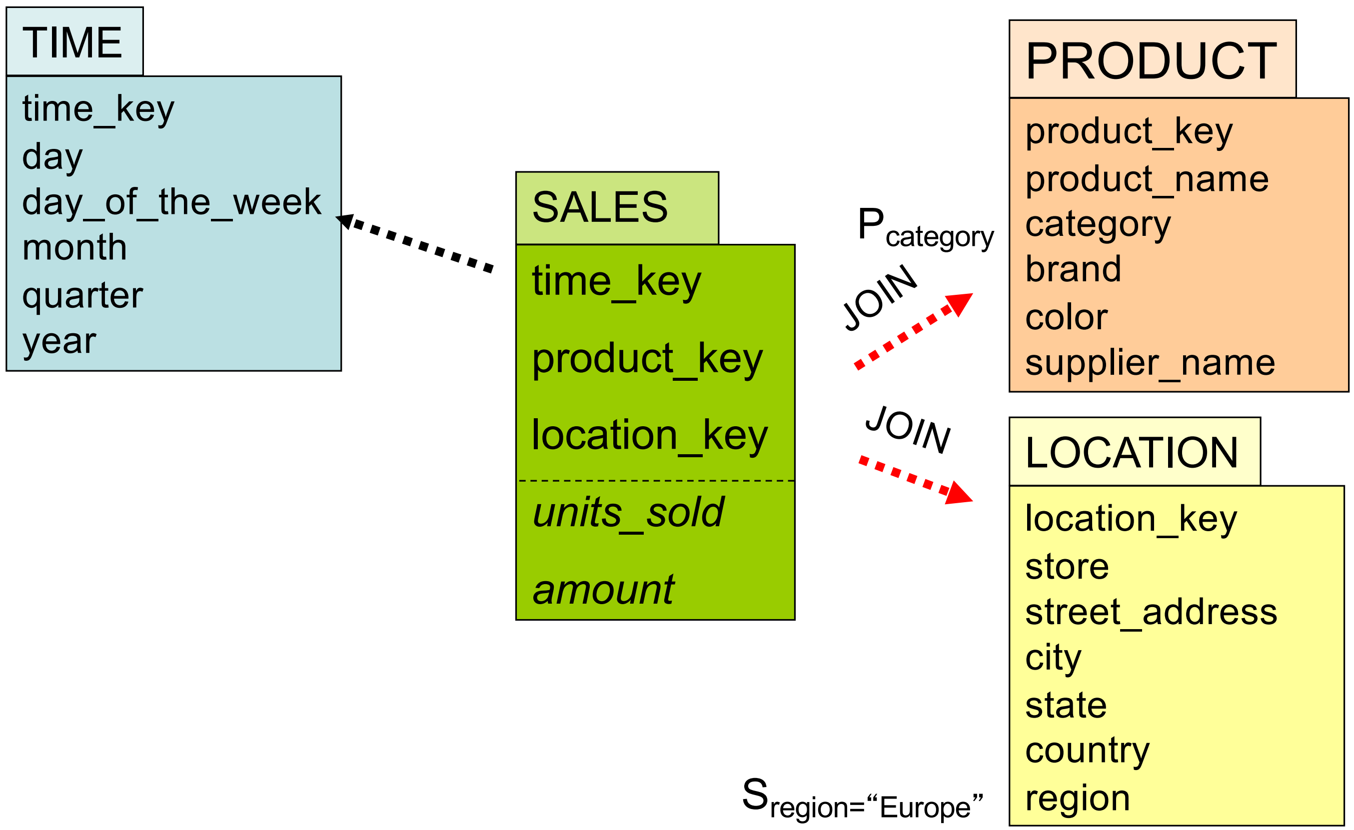
Advantages of Star Schema

- Facts and dimensions are clearly depicted
  - dimension tables are relatively static, data is loaded (append mostly) into fact table(s)
  - easy to comprehend (and write queries)

"Find total sales per product-category in our stores in Europe"

```
SELECT PRODUCT.category, SUM(SALES.amount)
FROM     SALES, PRODUCT,LOCATION
WHERE    SALES.product_key = PRODUCT.product_key
AND      SALES.location_key = LOCATION.location_key
AND      LOCATION.region="Europe"
GROUP BY PRODUCT.category
```

Star Schema Query Processing



Exercise

- Design a data warehouse to record the quantity and sales of beer purchases
  - DRINKER(Code, Name, Address, Phone, BDay, Gender)
  - STORE(Code, Name, Address, Phone)
  - BEER (Code, Name, Type, BottlePrice, CasePrice)
  - TYPE(Code, Name, Region)
  - TIME (TimeStamp, Date, Year)
  - PURCHASE(Drinker, Beer, Time, nrBottles, nrCases)

Need for Aggregates

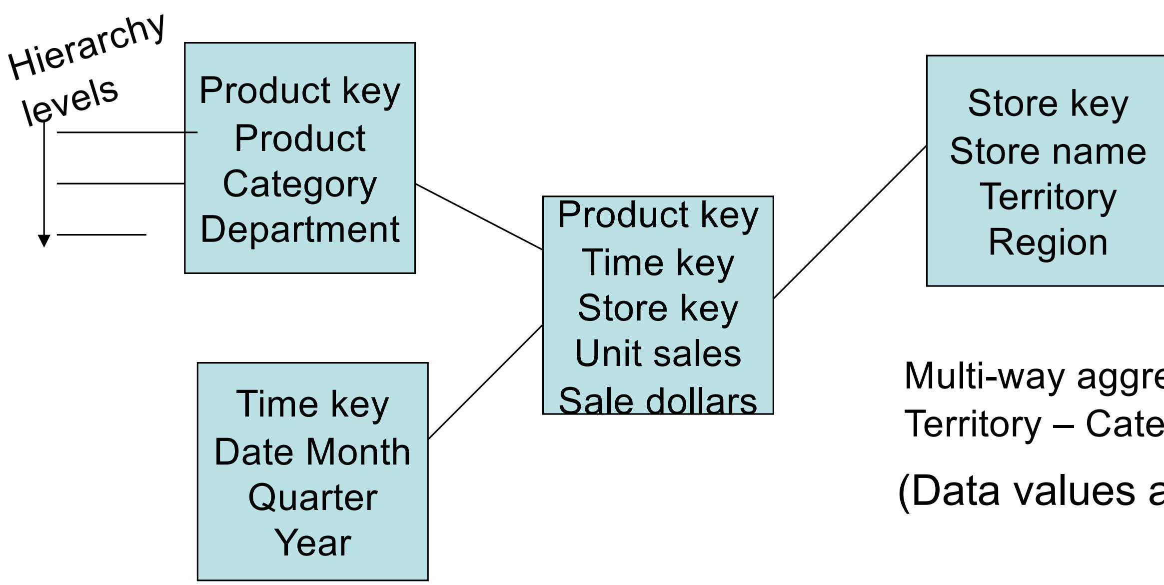
- Sizes of typical tables:
  - Time dimension: 5 years x 365 days = 1825
  - Store dimension: 300 stores reporting daily sales
  - Production dimension: 40,000 products in each store (about 4000 sell in each store daily)
  - Maximum number of base fact table records: 2 billion (lowest level of detail)
- A query involving 1 brand, all store, 1 year: retrieve/summarize over 7 million fact table rows.

OLAP Operation

- So, how are *concept hierarchies* useful in OLAP?
- In the multidimensional model, data are organized into multiple dimensions,
- And each dimension contains multiple levels of abstraction defined by concept hierarchies

Aggregating Fact Tables

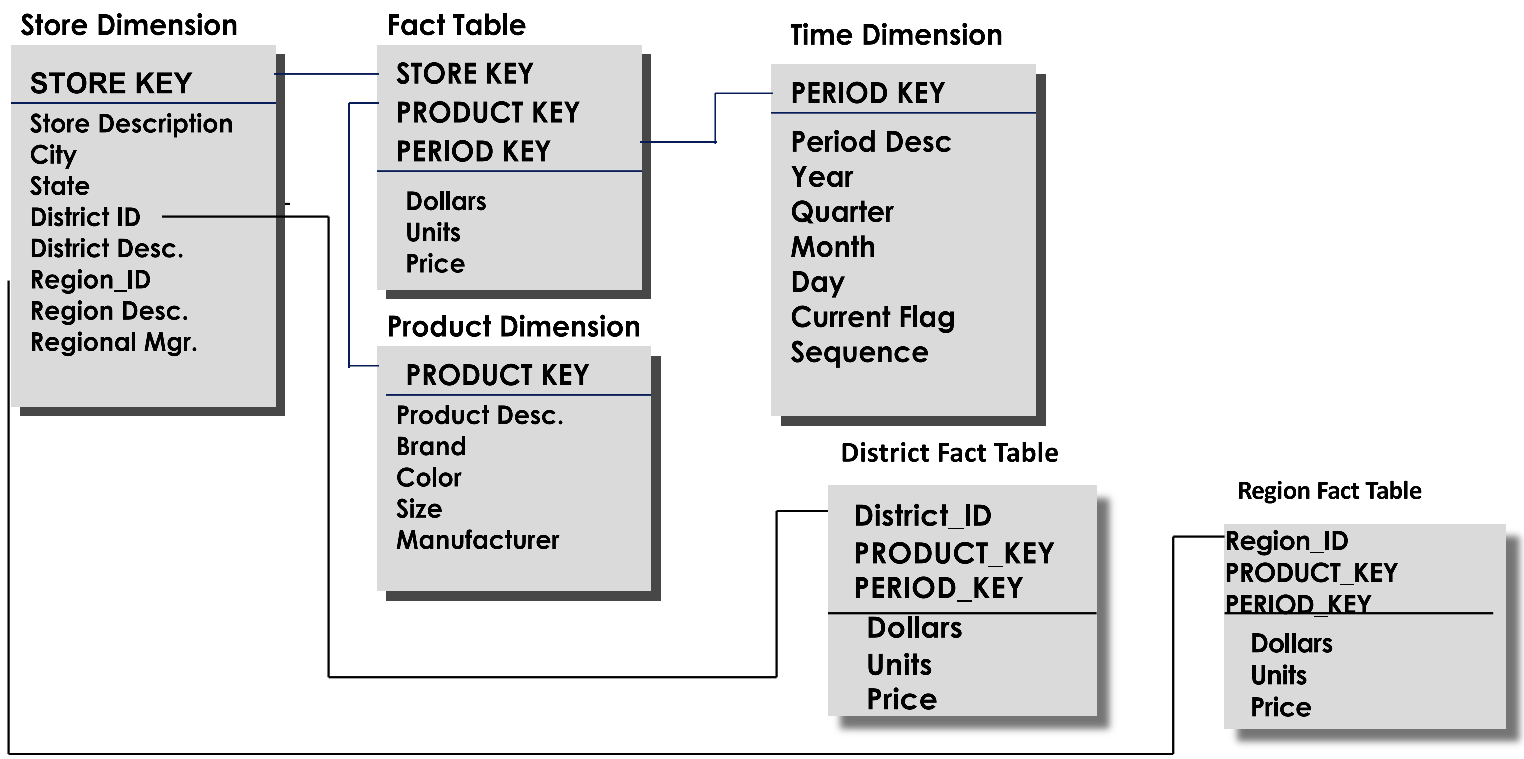
- Aggregate fact tables are summaries of the most granular data at higher levels along the dimension hierarchies.



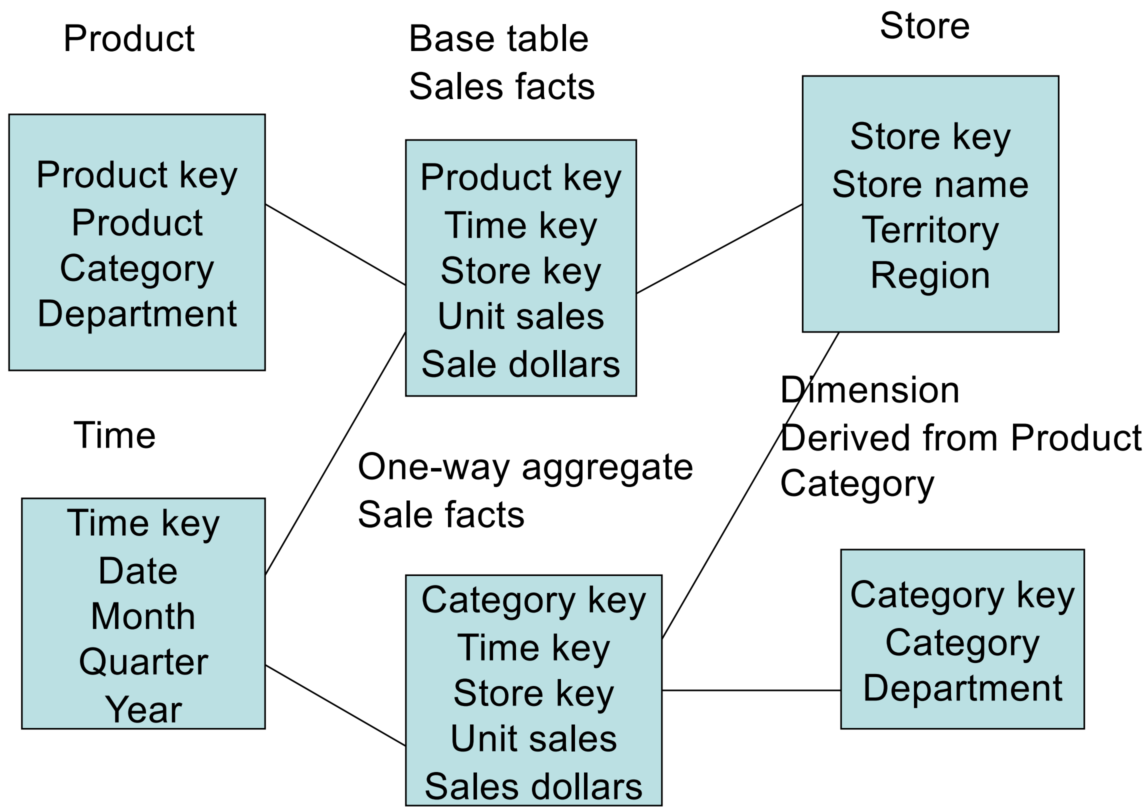
Multi-way aggregates:
Territory – Category – Month

(Data values at higher level)

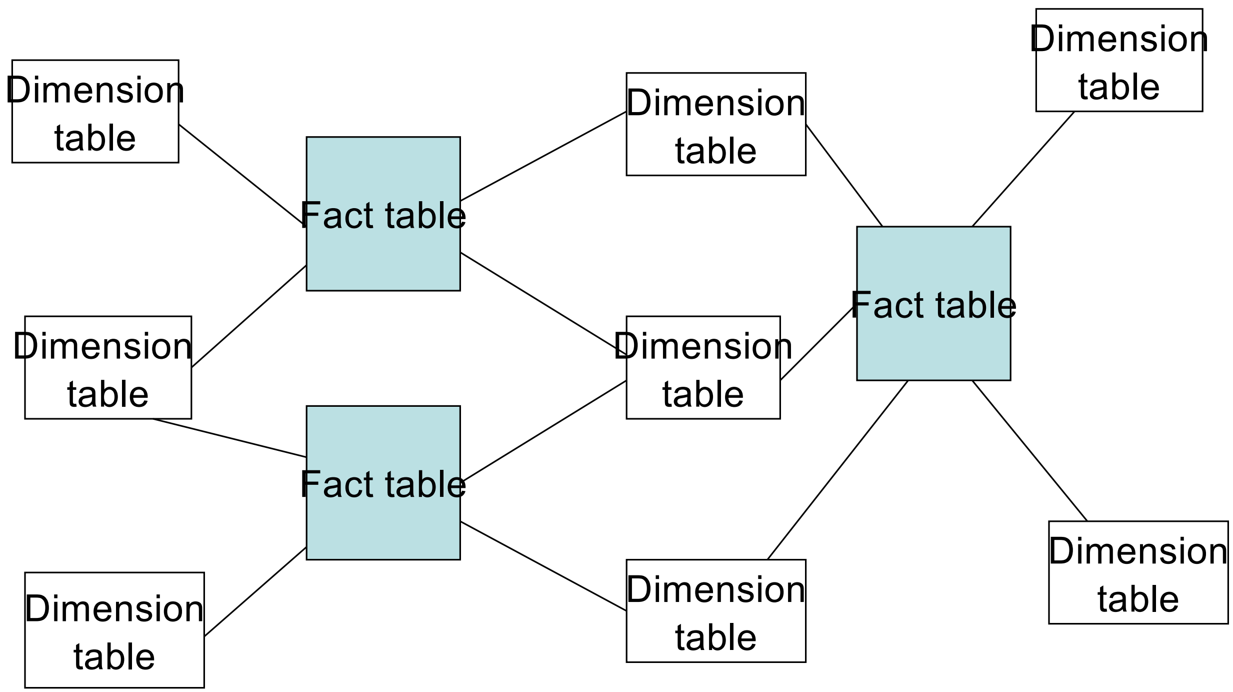
The "Fact Constellation" Schema



Aggregate Fact Tables



Families of Stars



Database Tables and Normalization

- Normalization
  - Process for evaluating and correcting table structures to minimize data redundancies
    - Reduces data anomalies
  - Works through a series of stages called normal forms:
    - First normal form (1NF)
    - Second normal form (2NF)
    - Third normal form (3NF)

Database Tables and Normalization (continued)

- Normalization (continued)
  - 2NF is better than 1NF; 3NF is better than 2NF
  - For most business database design purposes, 3NF is as high as we need to go in normalization process
  - Highest level of normalization is not always most desirable

The Normalization Process

- Each table represents a single subject
- No data item will be unnecessarily stored in more than one table
- All attributes in a table are dependent on the primary key

Conversion to First Normal Form

- Repeating group
  - Derives its name from the fact that a group of multiple entries of same type can exist for any single key attribute occurrence
- Relational table must not contain repeating groups
- Normalizing table structure will reduce data redundancies
- Normalization is three-step procedure

Denormalization

- Creation of normalized relations is important database design goal
- Processing requirements should also be a goal
- If tables decomposed to conform to normalization requirements:
  - Number of database tables expands

Denormalization (continued)

- Joining the larger number of tables takes additional input/output (I/O) operations and processing logic, thereby reducing system speed
- Conflicts between design efficiency, information requirements, and processing speed are often resolved through compromises that may include denormalization

Denormalization (continued)

- Unnormalized tables in production database tend to suffer from these defects:
  - Data updates are less efficient because programs that read and update tables must deal with larger tables
  - Indexing is more cumbersome
  - Unnormalized tables yield no simple strategies for creating virtual tables known as views

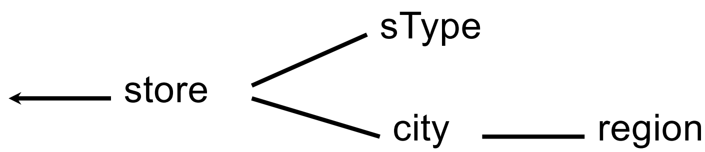
Denormalization (continued)

- Use denormalization cautiously
- Understand why—under some circumstances—unnormalized tables are better choice

Snowflake Schema

- Snowflake schema is a type of star schema but a more complex model.
- "Snowflaking" is a method of normalizing the dimension tables in a star schema.
- The normalization eliminates redundancy.
- The result is more complex queries and reduced query performance.

Dimension Hierarchies



store	<u>storeId</u>	cityId	tId	mgr
	s5	sfo	t1	joe
	s7	sfo	t2	fred
	s9	la	t1	nancy

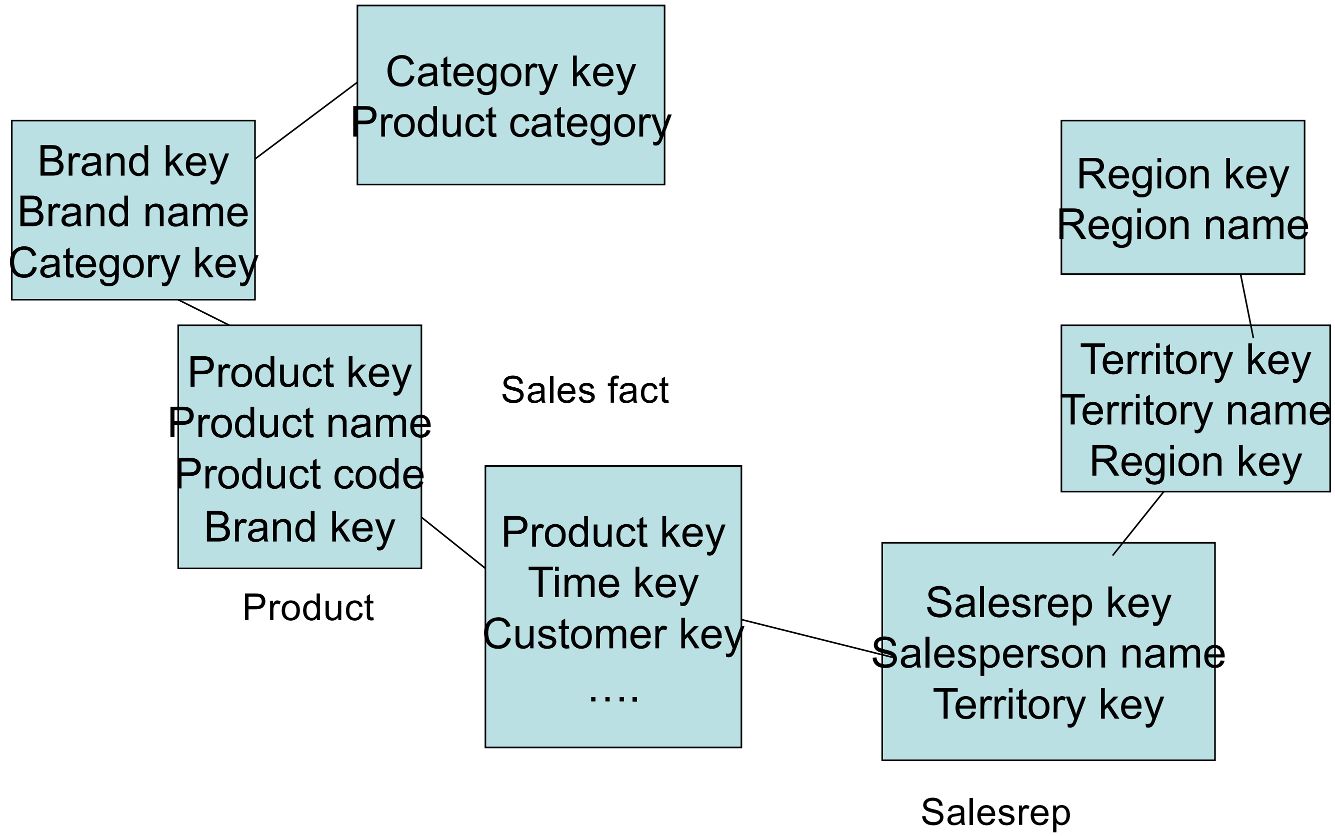
sType	<u>tId</u>	size	location
	t1	small	downtown
	t2	large	suburbs

city	<u>cityId</u>	pop	regId
	sfo	1M	north
	la	5M	south

region	<u>regId</u>	name
	north	cold region
	south	warm region

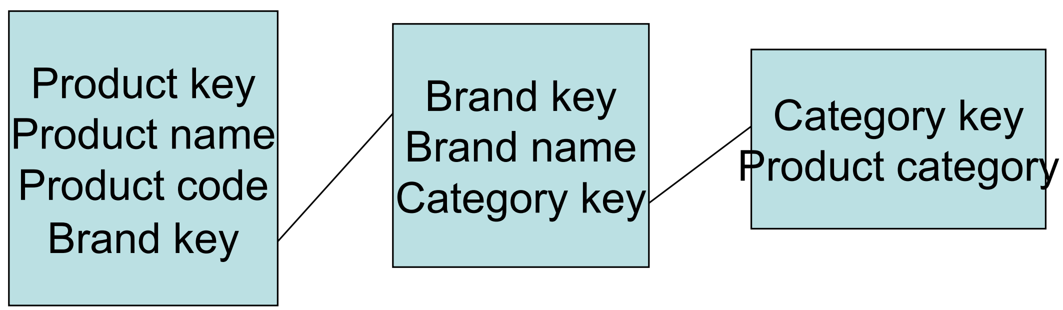
- ➔ snowflake schema
- ➔ constellations

Sales: Snowflake Schema



Snowflaking

- The attributes with low cardinality in each original dimension table are removed to form separate tables. These new tables are linked back to the original dimension table through artificial keys.



Snowflake Schema

- Advantages:
  - Small saving in storage space
  - Normalized structures are easier to update and maintain
- Disadvantages:
  - Schema less intuitive and end-users are put off by the complexity
  - Ability to browse through the contents difficult
  - Degrade query performance because of additional joins

What is the Best Design?

- Performance benchmarking can be used to determine what is the best design.
- Snowflake schema: easier to maintain dimension tables when dimension tables are very large (reduce overall space). It is not generally recommended in a data warehouse environment.
- Star schema: more effective for data cube browsing (less joins): can affect performance.

Aggregates

- Add up amounts for day 1
- In SQL: SELECT sum(amt) FROM SALE
  WHERE date = 1

sale	prodId	storeId	date	amt
	p1	s1	1	12
	p2	s1	1	11
	p1	s3	1	50
	p2	s2	1	8
	p1	s1	2	44
	p1	s2	2	4

⇨ 81

Aggregates

- Add up amounts by day
- In SQL: SELECT date, sum(amt) FROM SALE
  GROUP BY date

sale	prodId	storeId	date	amt
	p1	s1	1	12
	p2	s1	1	11
	p1	s3	1	50
	p2	s2	1	8
	p1	s1	2	44
	p1	s2	2	4

ans	date	sum
	1	81
	2	48

Another Example

- Add up amounts by day, product
- In SQL: SELECT date, sum(amt) FROM SALE
  GROUP BY date, prodId

sale	prodId	storeId	date	amt
	p1	s1	1	12
	p2	s1	1	11
	p1	s3	1	50
	p2	s2	1	8
	p1	s1	2	44
	p1	s2	2	4

sale	prodId	date	amt
	p1	1	62
	p2	1	19
	p1	2	48

rollup →

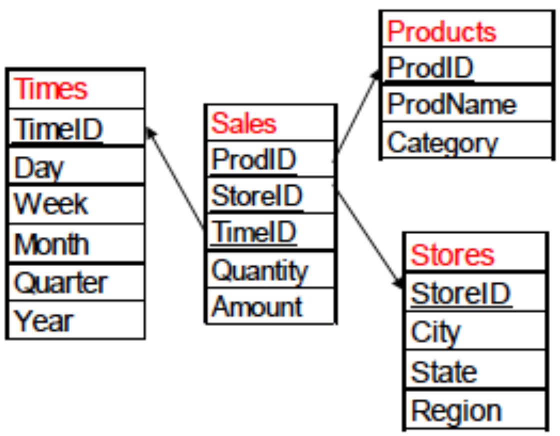
← drill-down

Aggregates

- Operators: sum, count, max, min, median, ave
- "Having" clause
- Using dimension hierarchy
  - average by region (within store)
  - maximum by month (within date)

OLAP Example (1)

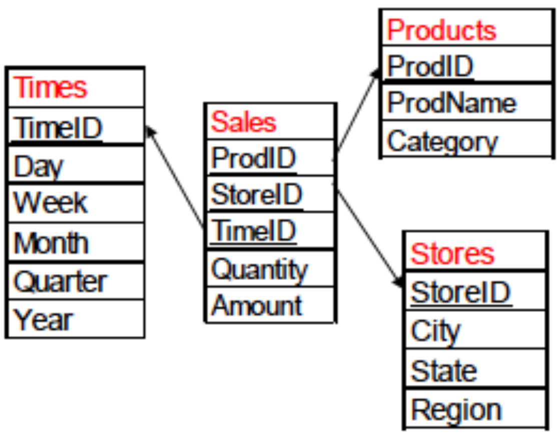
- Sales in 2009 were not good
  - Rollup by products, stores, and also by time to year (i.e., climbing up the hierarchy)
  - Slice on time dimension (year = 2009)
  - Drill down by time to month




```
select month, sum(amount)
from sales s, times t
    where s.timeid = t.timeid
            and t.year = 2009
group by month
```

OLAP Example (2)

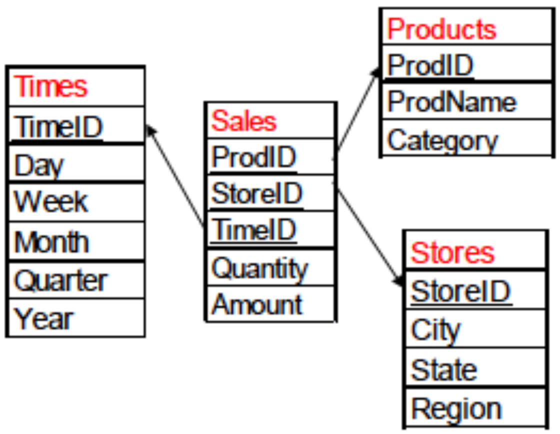
- Monthly sales amounts are roughly the same
  - Drill down by product to category to find out why




```
select month, category,
        sum(quantity) as units_sold
from sales s, times t, products p
where s.timeid = t.timeid
        and s.prodid = p.prodid
        and t.year = 2009
group by month, category
```

OLAP Example (3)

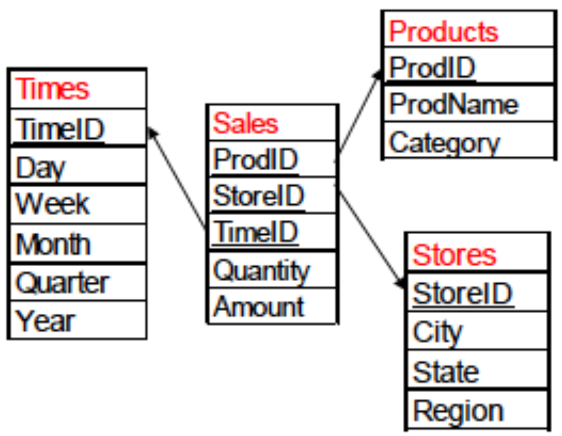
- Category ‘laptop’ not selling well
  - – Slice on Laptop
  - – Drill down by store to region to find out why




```
select month, l.region, sum(quantity)
from sales s, times t, products p, stores l
where s.timeid = t.timeid
        and s.prodid = p.prodid
        and t.year = 2009
        and p.category = ‘Laptop’
        and s.storeid = l.storeid
group by t.month, l.region
```

OLAP Example (4)

- Difficult to spot problem: month might be too fine-grained
  - rolling up by time to quarter




```
select quarter, l.region, sum(quantity)
from sales s, times t, products p, stores l
where s.timeid = t.timeid
        and s.prodid = p.prodid
        and t.year = 2009
        and p.category = ‘Laptop’
        and s.storeid = l.storeid
group by t.quarter, l.region
```

Laptops not selling well in west coast in the 4th quarter of 2009!

SQL Support for OLAP

- CUBE, ROLLUP, PIVOT Operations.

Database….

```sql
CREATE TABLE sales (
    brand VARCHAR NOT NULL,
    segment VARCHAR NOT NULL,
    quantity INT NOT NULL,
    PRIMARY KEY (brand, segment)
);

INSERT INTO sales (brand, segment, quantity)
VALUES
    ('ABC', 'Premium', 100),
    ('ABC', 'Basic', 200),
    ('XYZ', 'Premium', 100),
    ('XYZ', 'Basic', 300);
```

brand	segment	quantity
ABC	Premium	100
ABC	Basic	200
XYZ	Premium	100
XYZ	Basic	300

GROUP BY....

```sql
SELECT
    brand,
    segment,
    SUM (quantity)
FROM
    sales
GROUP BY
    brand,
    segment;
```

brand	segment	sum
ABC	Premium	100
XYZ	Premium	100
XYZ	Basic	300
ABC	Basic	200

What if you want all of em..

```
SELECT
   brand,
   segment,
   SUM (quantity)
FROM
   sales
GROUP BY
   brand,
   segment

UNION ALL

SELECT
   brand,
   NULL,
   SUM (quantity)
FROM
   sales
GROUP BY
   brand

UNION ALL
```

```
 SELECT
   NULL,
   segment,
   SUM (quantity)
FROM
   sales
GROUP BY
   segment

UNION ALL

SELECT
   NULL,
   NULL,
   SUM (quantity)
FROM
   sales;
```

Grouping set

```
SELECT
   c1,
   c2,
   aggregate_function(c3)
FROM
   table_name
GROUP BY
   GROUPING SETS (
      (c1, c2),
      (c1),
      (c2),
      ()
);
```

GROUPING Function

```sql
SELECT
   brand,
   segment,
   SUM (quantity)
FROM
   sales
GROUP BY
   GROUPING SETS (
      (brand, segment),
      (brand),
      (segment),
      ()
   );
```

brand	segment	sum
ABC	Basic	200
ABC	Premium	100
ABC	(Null)	300
XYZ	Basic	300
XYZ	Premium	100
XYZ	(Null)	400
(Null)	Basic	500
(Null)	Premium	200
(Null)	(Null)	700

CUBE

```sql
SELECT
   c1,
   c2,
   c3,
   aggregate (c4)
FROM
   table_name
GROUP BY
   CUBE (c1, c2, c3);
```

<- IS ->

```sql
CUBE(c1,c2,c3)

GROUPING SETS (
   (c1,c2,c3),
   (c1,c2),
   (c1,c3),
   (c2,c3),
   (c1),
   (c2),
   (c3),
   ()
)
```

brand	segment	quantity
ABC	Premium	100
ABC	Basic	200
XYZ	Premium	100
XYZ	Basic	300

-->

```sql
SELECT
    brand,
    segment,
    SUM (quantity)
FROM
    sales
GROUP BY
    CUBE (brand, segment)
ORDER BY
    brand,
    segment;
```

-->

brand	segment	sum
ABC	Basic	200
ABC	Premium	100
ABC	(Null)	300
XYZ	Basic	300
XYZ	Premium	100
XYZ	(Null)	400
(Null)	Basic	500
(Null)	Premium	200
(Null)	(Null)	700

ROLLUP

- What if you only need a subset of combinations?

CUBE(c1,c2,c3)

```
(c1, c2, c3)
(c1, c2)
(c2, c3)
(c1,c3)
(c1)
(c2)
(c3)
()
```

ROLLUP(c1,c2,c3)

```
(c1, c2, c3)
(c1, c2)
(c1)
()
```

```
SELECT
   c1,
   c2,
   c3,
   aggregate(c4)
FROM
   table_name
GROUP BY
   ROLLUP (c1, c2, c3);
```

```
SELECT
   brand,
   segment,
   SUM (quantity)
FROM
   sales
GROUP BY
   ROLLUP (brand, segment)
ORDER BY
   brand,
   segment;
```

brand	segment	sum
ABC	Basic	200
ABC	Premium	100
ABC	(Null)	300
XYZ	Basic	300
XYZ	Premium	100
XYZ	(Null)	400
(Null)	(Null)	700

Implementing a Warehouse

- *Monitoring*: Sending data from sources
- *Integrating*: Loading, cleansing,...
- *Processing*: Query processing, indexing, ...
- *Managing*: Metadata, Design, ...

Monitoring

- Source Types: relational, flat file, IMS, VSAM, IDMS, WWW, news-wire, …
- Incremental vs. Refresh

customer	<u>id</u>	name	address	city
	53	joe	10 main	sfo
	81	fred	12 main	sfo
	111	**sally**	**80 willow**	**la**

new

Monitoring Techniques

- Periodic snapshots
- Database triggers
- Log shipping
- Data shipping (replication service)
- Transaction shipping
- Polling (queries to source)
- Screen scraping
- Application level monitoring

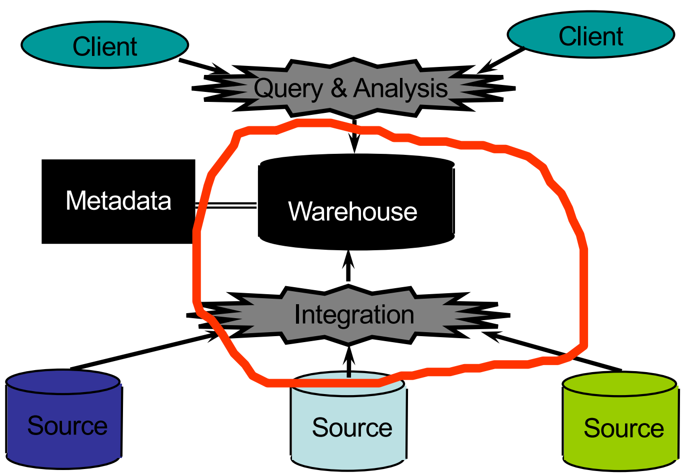
→ Advantages & Disadvantages!!

Monitoring Issues

- Frequency
  - periodic: daily, weekly, …
  - triggered: on “big” change, lots of changes, ...
- Data transformation
  - convert data to uniform format
  - remove & add fields (e.g., add date to get history)
- Standards (e.g., ODBC)
- Gateways

Integration

- Data Cleaning
- Data Loading
- Derived Data



When and where to transform?

- Extract Transform and Load (ETL)
  - Data transformation is done outside of database
- Extract, Load and Transform
  - Use SQL as the transformation language to convert data into uniform representation

Data Cleaning

- Migration (e.g., yen ⇨ dollars)
- Scrubbing: use domain-specific knowledge (e.g., social security numbers)
- Fusion (e.g., mail list, customer merging)

billing DB → customer1(Joe) → merged_customer(Joe)

service DB → customer2(Joe) → merged_customer(Joe)

- Auditing: discover rules & relationships (like data mining)

Cleaning

- Removal of duplicate records
- Removal of records with gaps
- Enforcement of check constraints
- Removal of null values
- Removal of implausible frequent values

Enrichment

- Supplementing operational data with outside data sources
  - Pharmacological research results
  - Demographic norms
  - Epidemiological findings
  - Cost factors
  - Medium range predictions

Coding and Organizing

- Un-Normalizing
- Rescaling
- Nonlinear transformations
- Categorizing
- Recoding, especially of null values

Loading Data

- Incremental vs. refresh
- Off-line vs. on-line
- Frequency of loading
  - At night, 1x a week/month, continuously
- Parallel/Partitioned load

Derived Data

- Derived Warehouse Data
  - indexes
  - aggregates
  - materialized views (next slide)
- When to update derived data?
- Incremental vs. refresh

Materialized Views

- Define new warehouse relations using SQL expressions

sale	prodId	storeId	date	amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4

product	id	name	price
	p1	bolt	10
	p2	nut	5

joinTb	prodId	name	price	storeId	date	amt
	p1	bolt	10	c1	1	12
	p2	nut	5	c1	1	11
	p1	bolt	10	c3	1	50
	p2	nut	5	c2	1	8
	p1	bolt	10	c1	2	44
	p1	bolt	10	c2	2	4

does not exist at any source

Materialization Factors

- Type/frequency of queries
- Query response time
- Storage cost
- Update cost

Design

- What data is needed?
- Where does it come from?
- How to clean data?
- How to represent in warehouse (schema)?
- What to summarize?
- What to materialize?
- What to index?

Current State of Industry

- Extraction and integration done off-line
  - Usually in large, time-consuming, batches
- Everything copied at warehouse
  - Not selective about what is stored
  - Query benefit vs storage & update cost
- Query optimization aimed at OLTP
  - High throughput instead of fast response
  - Process whole query before displaying anything