Data Warehousing and OLAP Technology

1. Objectives

- What is a data warehouse?

- Data warehouse design issues.

- General architecture of a data warehouse

- Introduction to Online Analytical Processing (OLAP) technology.

- Data warehousing and data mining relationship.

2. What is Data Warehouse?

2.1. Definitions

- Defined in many different ways, but not rigorously.
- A decision support database that is maintained separately from the organization's operational database
- Support information processing by providing a solid platform of consolidated, historical data for analysis.
- "A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision-making process."—W. H. Inmon
- Operational Data: Data used in day-to-day needs of company.
- Informational Data: Supports other functions such as planning and forecasting.
- Data mining tools often access data warehouses rather than operational data.
- Data warehousing: The process of constructing and using data warehouses.

2.2. 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.

2.3. 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
        - E.g., Hotel price: currency, tax, breakfast covered, etc.

    - When data is moved to the warehouse, it is converted.

2.4. 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)

- Every key structure in the data warehouse
  - Contains an element of time, explicitly or implicitly
  - But the key of operational data may or may not contain "time element".

2.5. Data Warehouse—Non-Volatile

- 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*.

2.6. Data Warehouse vs. Heterogeneous DBMS

- Traditional heterogeneous DB integration:
  - Build wrappers/mediators on top of heterogeneous databases
  - Query driven approach
    - When a query is posed to a client site, a meta-dictionary is used to translate the query into queries appropriate for individual heterogeneous sites involved, and the results are integrated into a global answer set
    - Complex information filtering, compete for resources
- Data warehouse: update-driven, high performance
  - Information from heterogeneous sources is integrated in advance and stored in warehouses for direct query and analysis

2.7. Data Warehouse vs. Operational DBMS

- OLTP (on-line transaction processing)
  - Major task of traditional relational DBMS
  - Day-to-day operations: purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.
- OLAP (on-line analytical processing)
  - Major task of data warehouse system
  - Data analysis and decision making
- Distinct features (OLTP vs. OLAP):
  - User and system orientation: customer vs. market
  - Data contents: current, detailed vs. historical, consolidated
  - Database design: ER + application vs. star + subject
  - View: current, local vs. evolutionary, integrated

- Access patterns: update vs. read-only but complex queries

2.8. OLTP vs. OLAP

	OLTP	**OLAP**
Users	Clerk, IT professional	Knowledge worker
Function	Day to day operations	Decision support
DB design	Application-oriented	Subject-oriented
Data	Current, up-to-date Detailed, flat relational Isolated	Historical, Summarized, multidimensional Integrated, consolidated
Usage	Repetitive	Ad-hoc
Access	Read/write, Index/hash on prim. Key	Lots of scans
Unit of work	Short, simple transaction	Complex query
# records accessed	Tens	Millions
#users	Thousands	Hundreds
DB size	100MB-GB	100GB-TB
Metric	Transaction throughput	Query throughput, response

2.9. Why Separate Data Warehouse?

- High performance for both systems
  - DBMS— tuned for OLTP: access methods, indexing, concurrency control, recovery
  - Warehouse—tuned for OLAP: complex OLAP queries, multidimensional view, and consolidation.
- Different functions and different data:
  - Missing data: Decision support requires historical data which operational DBs do not typically maintain
  - Data consolidation: DS requires consolidation (aggregation, summarization) of data from heterogeneous sources
  - Data quality: different sources typically use inconsistent data representations, codes and formats which have to be reconciled.

3. Multidimensional Data Model

3.1. Definitions

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

- This is not a 3-dimensional cube: it is n-dimensional cube.

- Dimensions of the cube are the equivalent of entities in a database, e.g., how the organization wants to keep records.

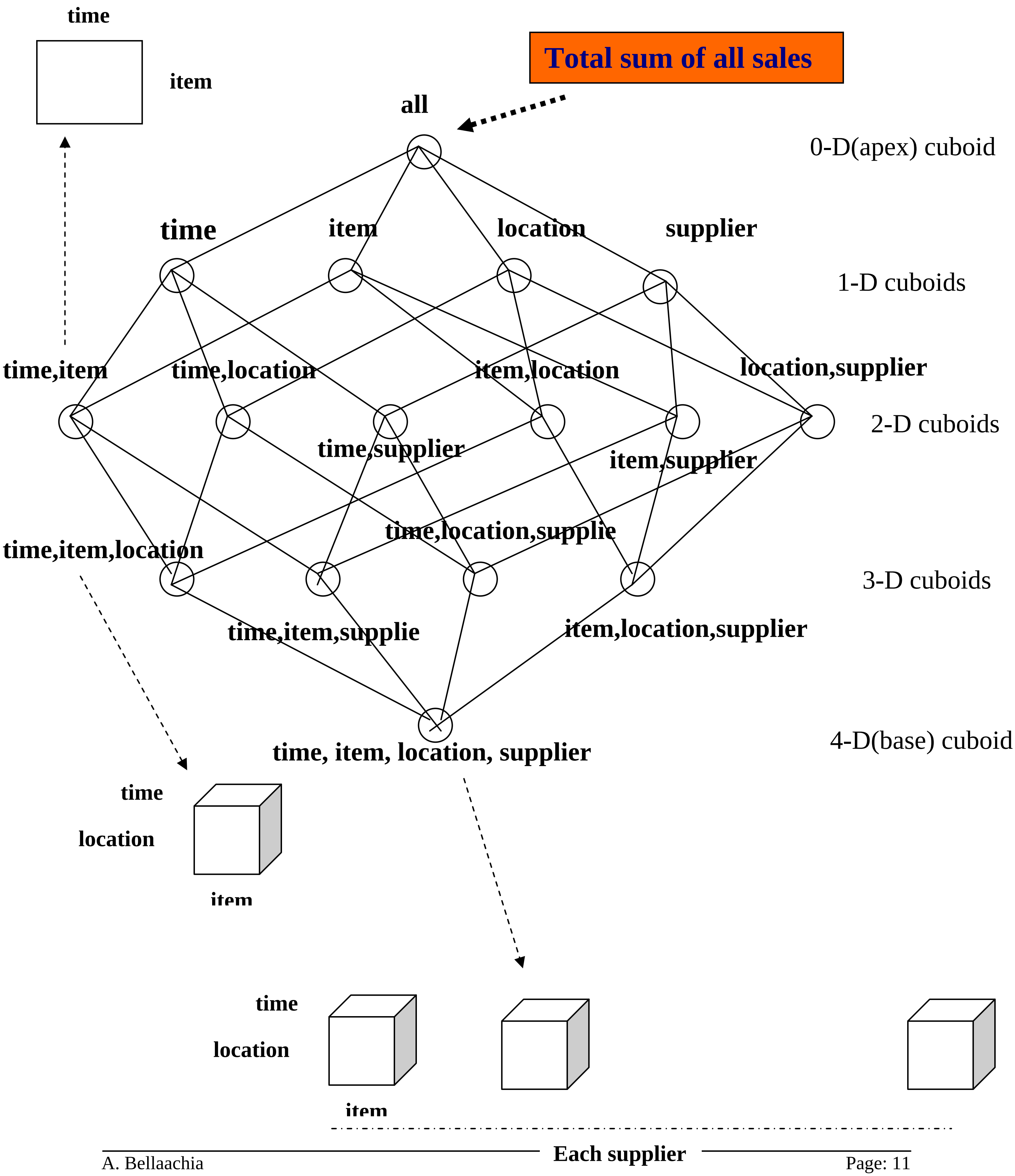
- Examples:
    - Product
    - Dates
    - Locations

- A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions

    - **Dimension tables**, such as item (item_name, brand, type), or time(day, week, month, quarter, year)
    - **Fact table** contains measures (such as dollars_sold) and keys to each of the related dimension tables

- In data warehousing literature, an n-D base cube is called a base cuboid. The top most 0-D cuboid, which holds the highest-level of summarization, is called the apex cuboid. The lattice of cuboids forms a data cube.

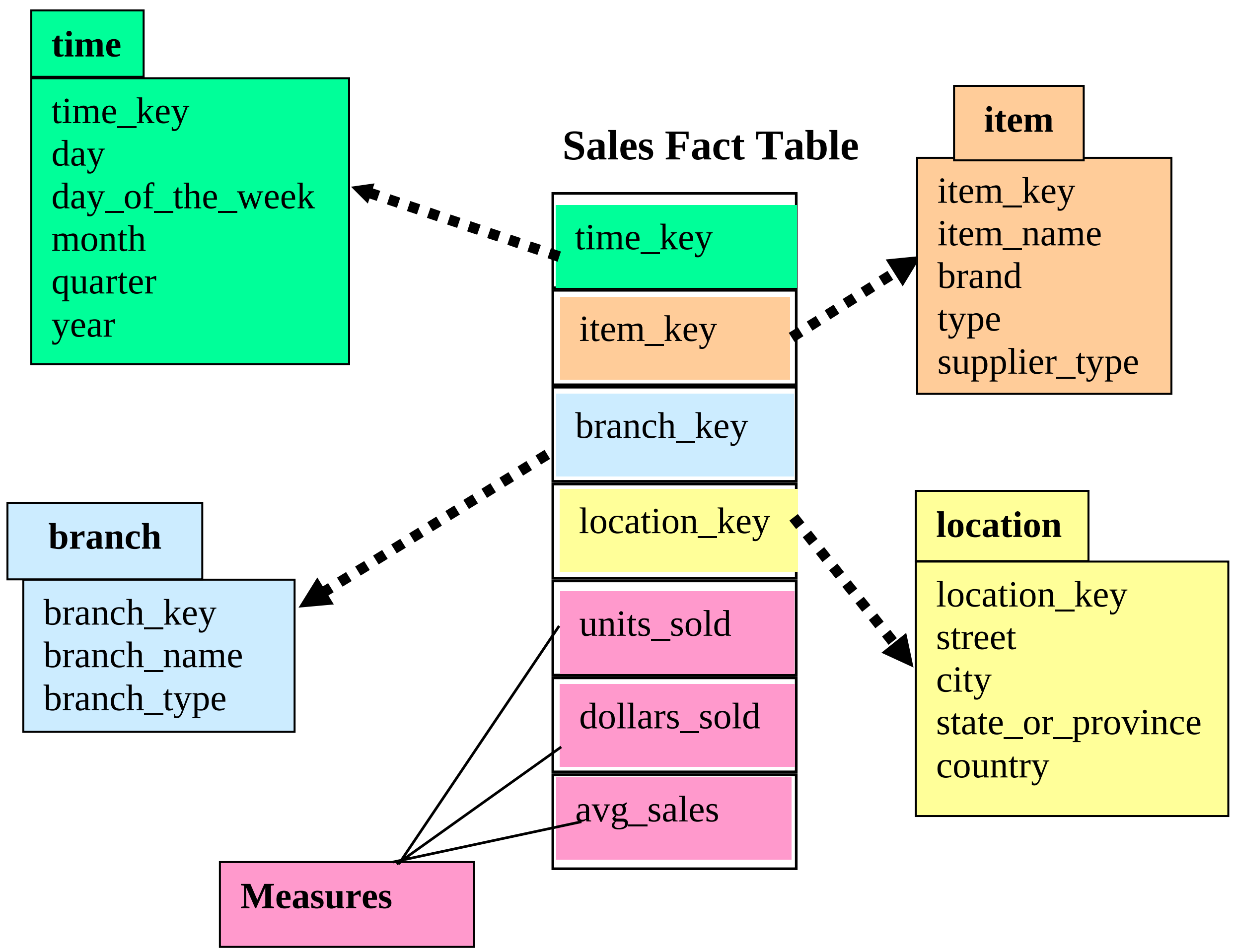
- Cube: A lattice of cuboids



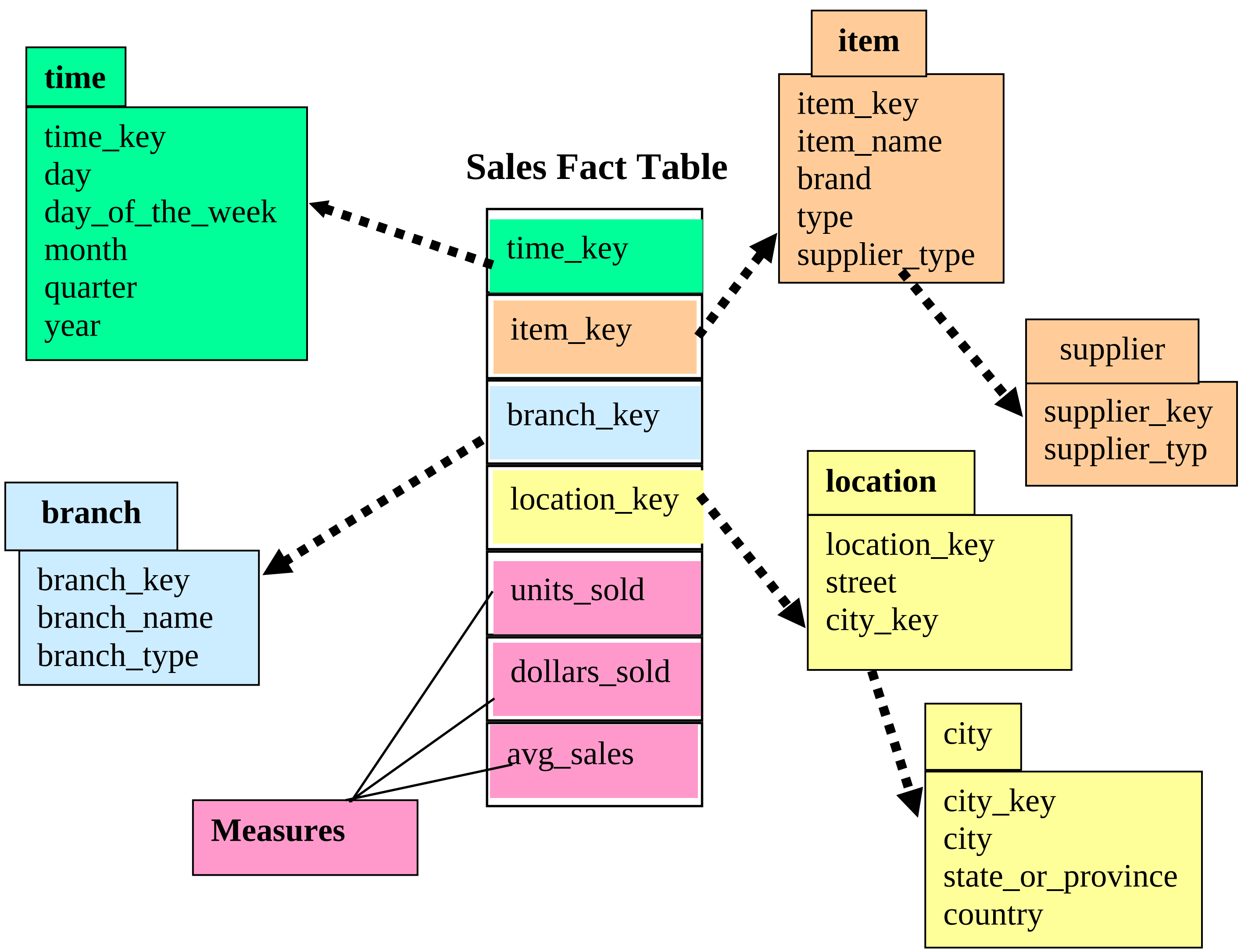
4. Conceptual Modeling of Data Warehousing

- Modeling data warehouses: dimensions & measures
  - Star schema: A fact table in the middle connected to a set of dimension tables
  - Snowflake schema: A refinement of star schema where some dimensional hierarchy **is normalized** into a set of smaller dimension tables, forming a shape similar to snowflake
  - Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation

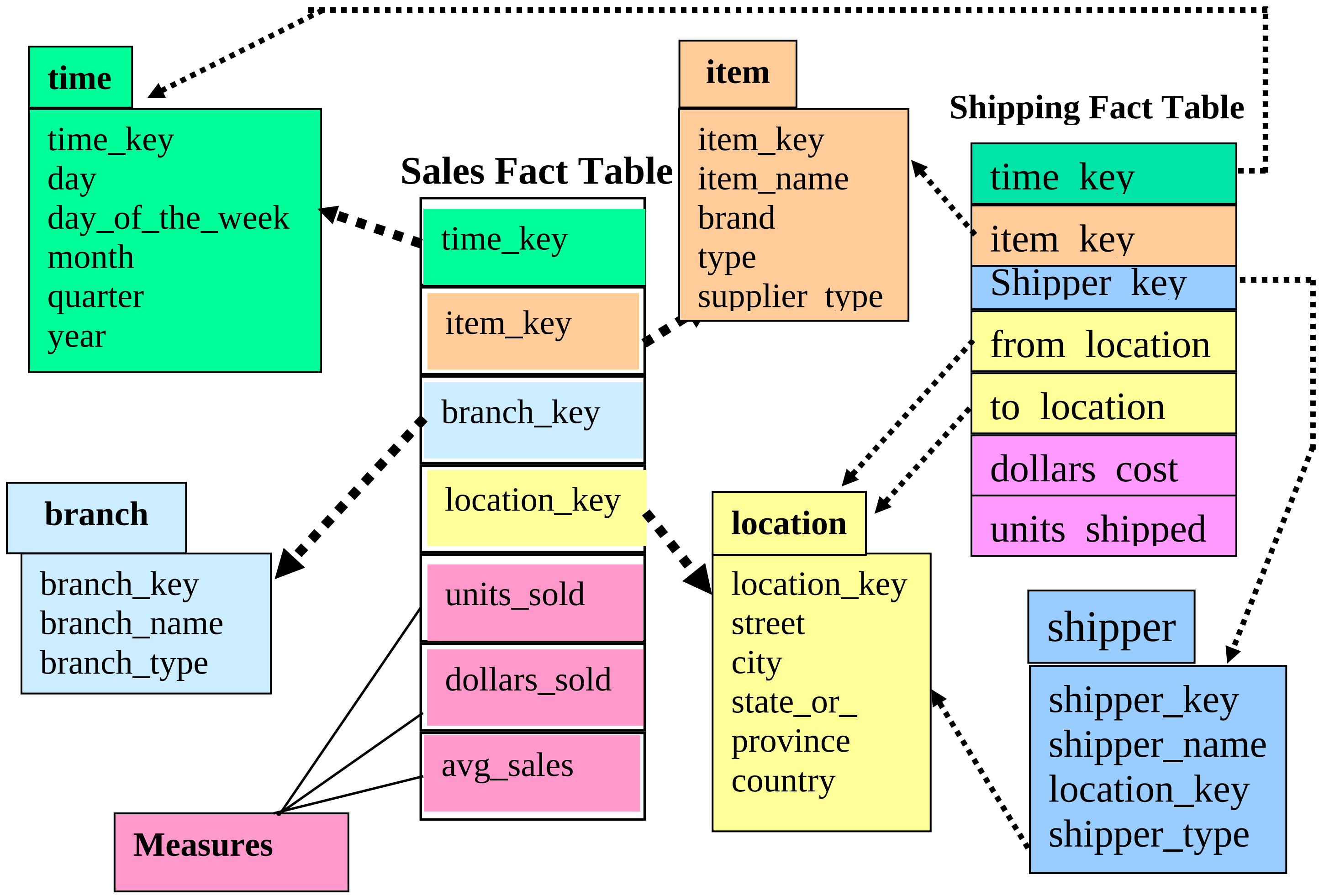
4.1. Star Schema



4.2. Snowflake Schema



4.3. Fact Constellation



5. A Data Mining Query Language: DMQL

5.1. Definitions and syntax

- Similar to RDBMS, we need a DDL (data definition language) to define the tables in the conceptual model.

- Cube Definition (Fact Table)

  - **Syntax:**
    define cube <cube_name> [<dimension_list>]: <measure_list>

  - **Example**
    define cube sales_star [time, item, branch, location]:
        dollars_sold = sum(sales_in_dollars)**,**
        avg_sales = avg(sales_in_dollars)**,**
        units_sold = count(*)

- Dimension Definition (Dimension Table)

  - Syntax:
        define dimension <dimension_name>
        as (<attribute_or_subdimension_list>)

  - Example:
        define dimension item
        as (item_key, item_name, brand, type,
            supplier_type)

- Special Case (Shared Dimension Tables)

    - First time as “cube definition”

    - Syntax:
      define dimension <dimension_name>
      as <dimension_name_first_time>
      in cube <cube_name_first_time>

    - Example:

      define dimension item **as** item **in cube** sales

5.2. Defining a Star Schema in DMQL

define cube sales_star [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars),
avg_sales = avg(sales_in_dollars),
units_sold = count(*)

define dimension time **as** (time_key, day, day_of_week, month, quarter, year)

define dimension item **as** (item_key, item_name, brand, type, supplier_type)

define dimension branch **as** (branch_key, branch_name, branch_type)

define dimension location **as** (location_key, street, city, province_or_state, country)

5.3. Defining a Snowflake Schema in DMQL

define cube sales_snowflake [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars),
avg_sales = avg(sales_in_dollars),
units_sold = count(*)

define dimension time **as** (
time_key,
day,
day_of_week,
month,
quarter,
year
)

define dimension item **as** (
item_key,
item_name,
brand, type,
supplier(supplier_key, supplier_type)
)

define dimension branch **as** (branch_key, branch_name, branch_type)

define dimension location **as** (
location_key,
street,
city(city_key, province_or_state, country)
)

5.4. Defining a Fact Constellation in DMQL

define cube sales [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars),
avg_sales = avg(sales_in_dollars),
units_sold = count(*)

define dimension time
as (time_key, day, day_of_week, month, quarter, year)

define dimension item
as (item_key, item_name, brand, type, supplier_type)

define dimension branch
as (branch_key, branch_name, branch_type)

define dimension location
as (location_key, street, city, province_or_state, country)

define cube shipping [time, item, shipper, from_location, to_location]:
dollar_cost = sum(cost_in_dollars),
unit_shipped = count(*)

define dimension time
as time
in cube sales

define dimension item
as item
in cube sales

```
define dimension shipper
as (    shipper_key,
        shipper_name,
        location as location in cube sales,
        shipper_type)

define dimension from_location
as location
in cube sales

define dimension to_location
as location
in cube sales
```

5.5. Measures: Three Categories

- A data cube function is a numerical function that can be evaluated at each point in the data cube space.

- Given a data point in the data cube space:

  Entry(v1, v2, …, vn)

  where vi is the value corresponding to dimension di.

  We need to apply the aggregate measures to the dimonsion values v1, v2, …, vn

- **<u>Distributive</u>**:

  - If the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning.

  - Example: count(), sum(), min(), max().

- **<u>Algebraic</u>**:

  - Use distributive aggregate functions.

  - If it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function.

- Example: avg(), min_N(), standard_deviation().

- **Holistic**:
  - If there is no constant bound on the storage size needed to describe a subaggregate.
  - E.g., median(), mode(), rank().

5.6. How to compute data cube measures?

- How do evaluate the dollars_sold and unit_sold in the star schema of the previous example?

- Assume that the relation database schema corresponding to our example is the following:

time (time_key, day, day_of_week, month, quarter, year)
item (item_key, item_name, brand, type, supplier(supplier_key, supplier_type))
branch (branch_key, branch_name, branch_type)
location (location_key, street, city, province_or_state, country)
sales (time_key, item_key, branch_key, location_key, number_of_unit_sold, price)

- Let us then compute the two measures we have in our data cube: dollars_sold and units_sold

```
select s.time_key, s.item_key, s.branch_key,
          s.location_key, sum(s.number_of_units_sold*s.price),
         sum(s.number_of_units_sold)
from time t, item i, branch b, location l, sales s
where s.time_key = t.time_key
          and s.item_key = i.item_key
          and s.branch_key = b.branch_key
          and s.location_key = l.location_key
group by   s.time_key, s.item_key, s.branch_key,
                  s.location_key
```

- Relationship between “data cube” and “group by”?
    - The above query corresponds to the base cuboid.
    - By changing the group by clause in our query, we may generate other cuboids.
    - What is query for the 0-D cuboid or apex?

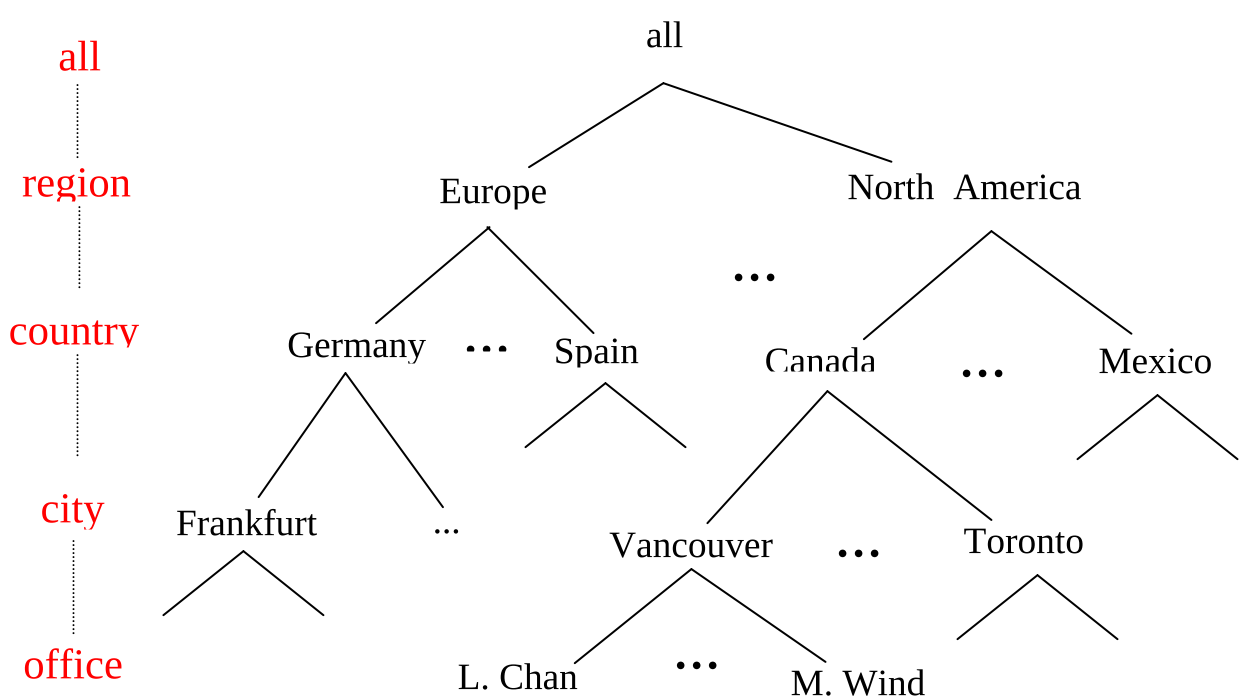
6. A Concept Hierarchy

- A concept hierarchy is an order relation between a set of attributes of a concept or dimension.

- It can be manually (users or experts) or automatically generated (statistical analysis).

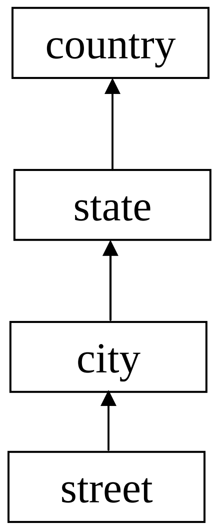
- Multidimensional data is usually organized into dimension and each dimension is further defined into a lower level of abstractions defined by concept hierarchies.

- Example: Dimension (location)

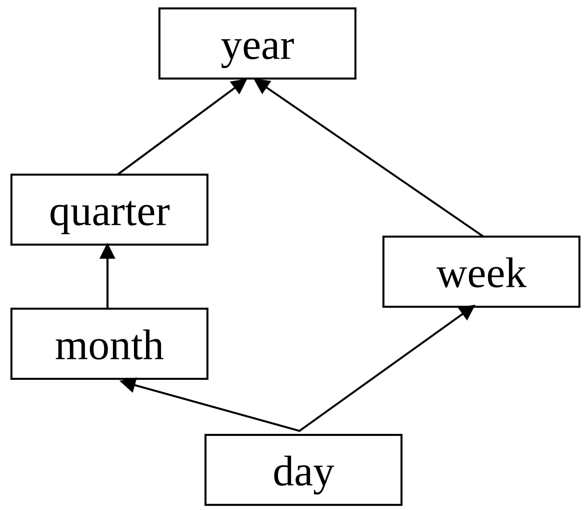


- The order can be either partial or total:

Location dimension: Street <city<state<country
Time dimension: Day < {month<quarter ; week} < year



Total order hierarchy



Partial order hierarchy

- Set-grouping hierarchy:
    - It is a concept hierarchy among groups of values.
    - Example: {1..10} < inexpensive

7. OLAP Operations in a Multidimensional Data

- Sales volume as a function of **product**, **time**, and **region.**

- **Dimensions hierarchical concepts: Product, Location, Time**

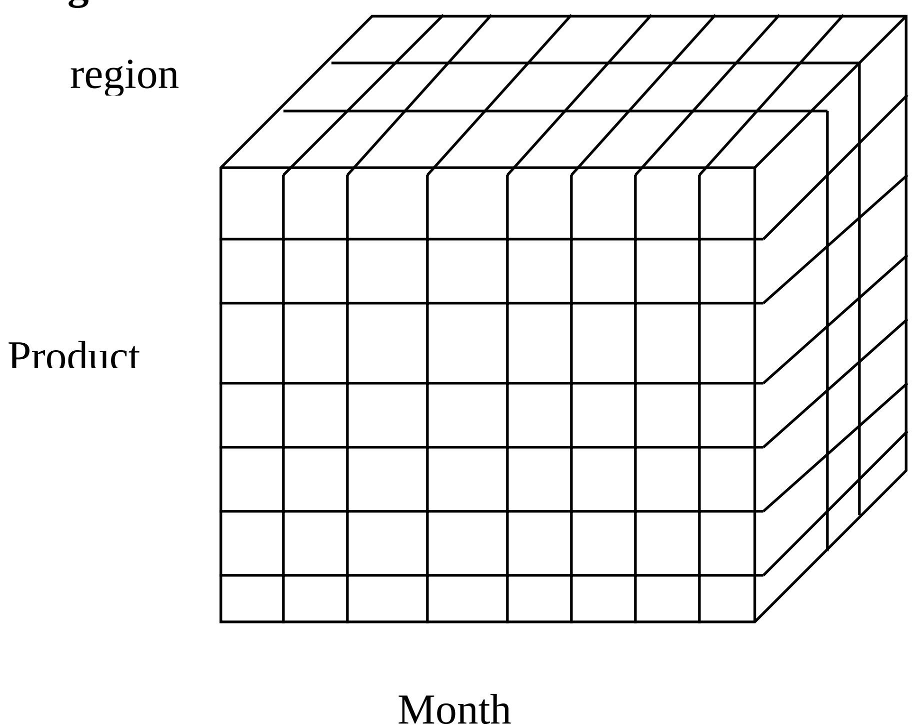
  Industry → Category → Product

  Region → Country → City → Office

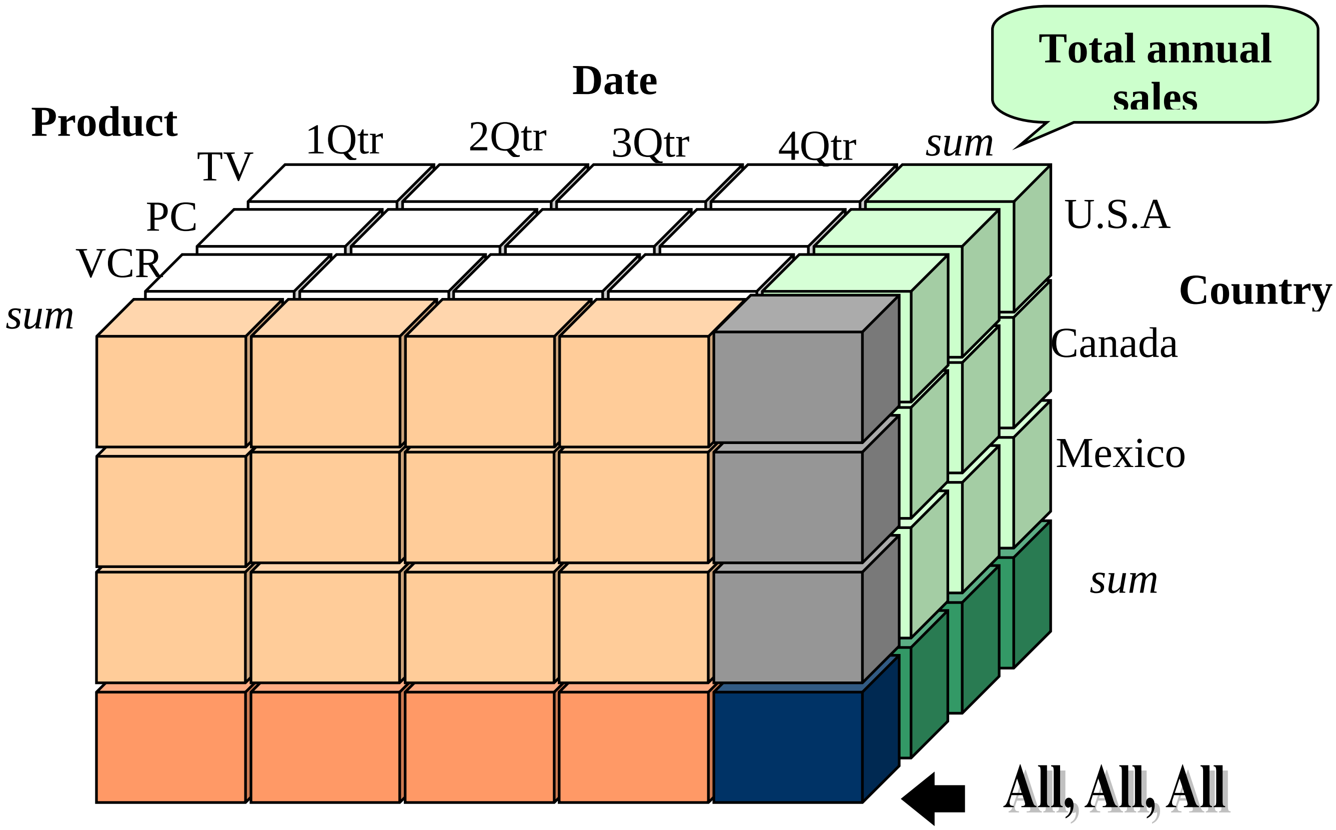
  Year → Quarter → Month → Day

  Week

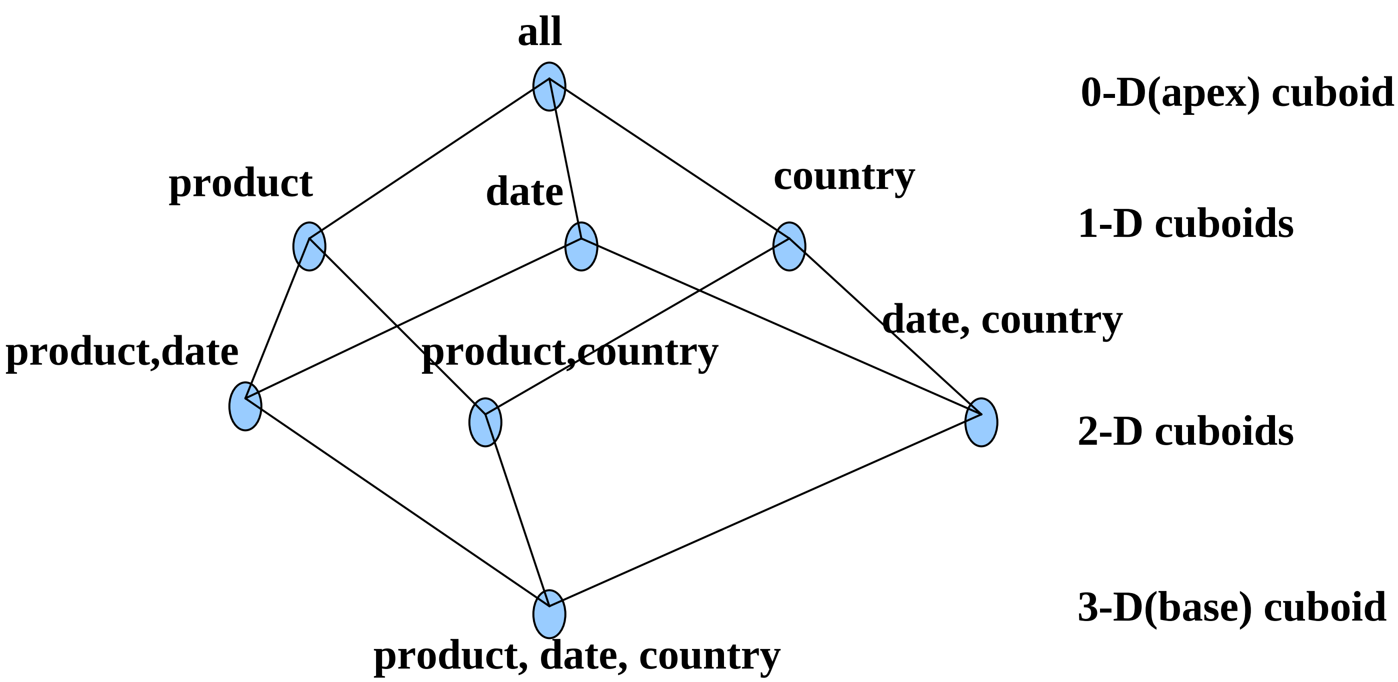
- Sales volume as a function of **product**, **month**, and **region.**



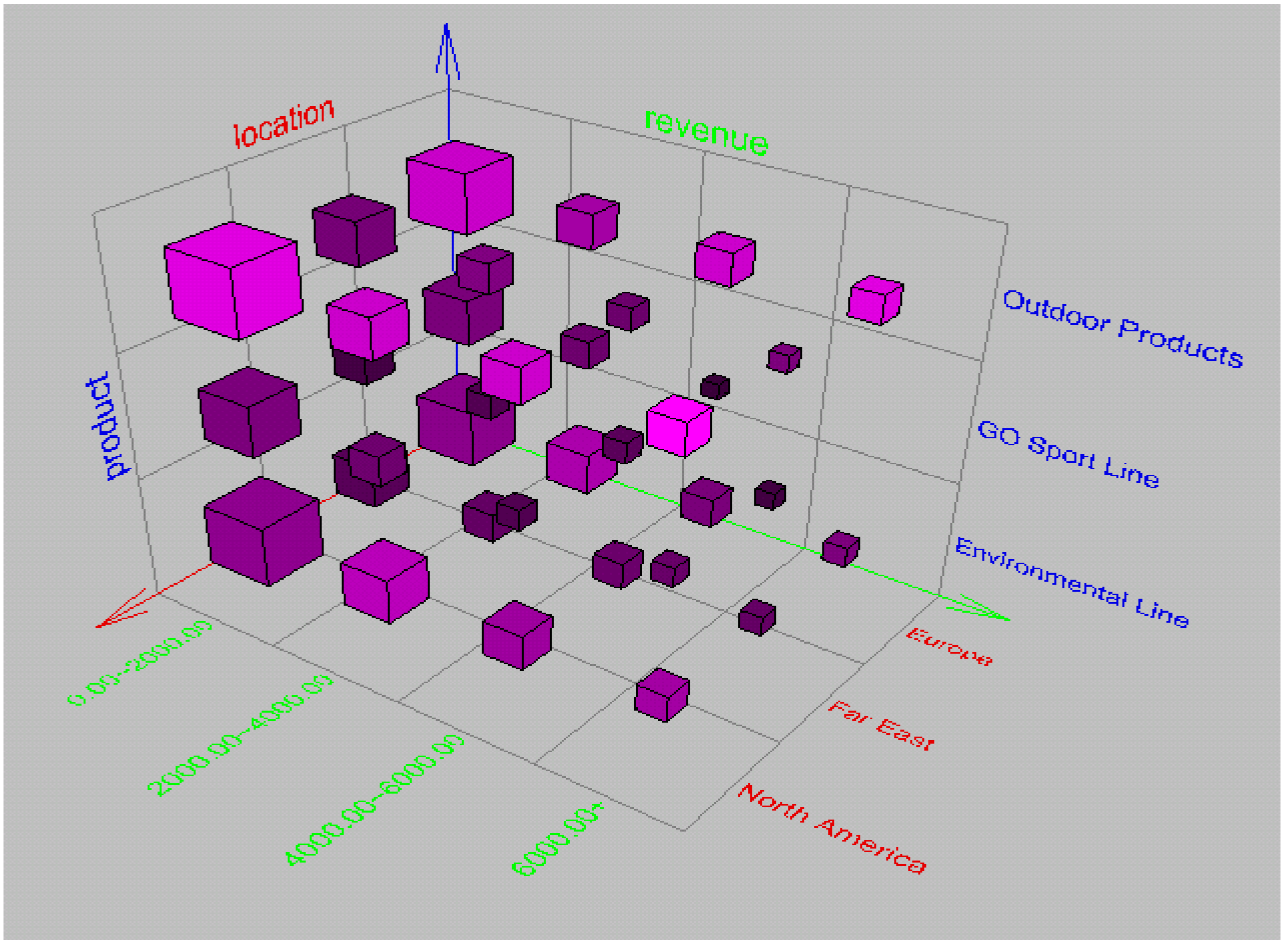
- A Sample data cube:



- Cuboids of the sample cube:



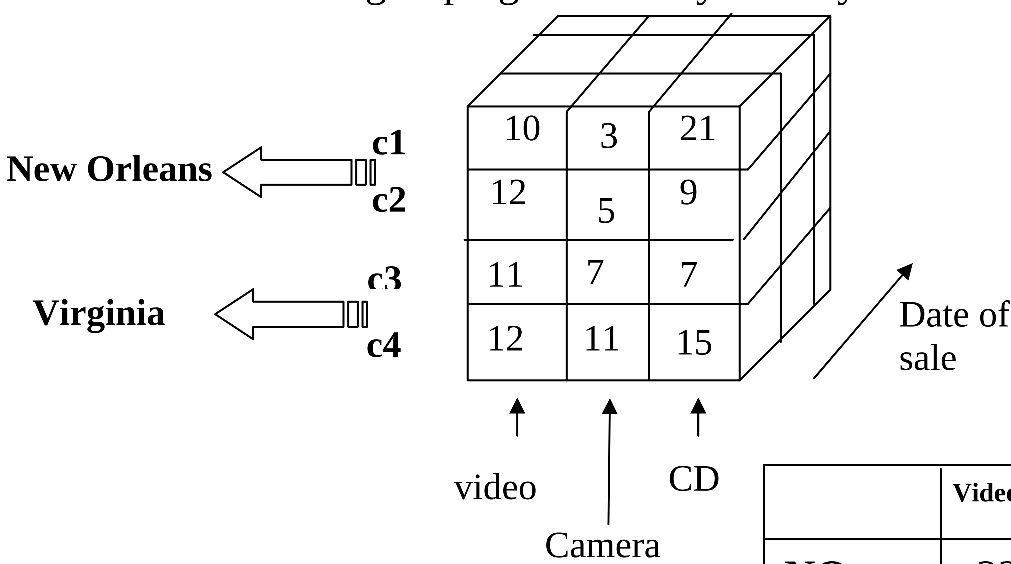
- Querying a data cube



8. OLAP Operations

- Objectives:
  - OLAP is a powerful analysis tool:
    - Forecasting
    - Statistical computations,
    - aggregations,
    - etc.

- Roll up (drill-up): summarize data
  - It is performed by climbing up hierarchy of a dimension or by dimension reduction (reduce the cube by one or more dimensions).
  - The roll up operation in the example is based location (roll up on location) is equivalent to grouping the data by country.



roll up

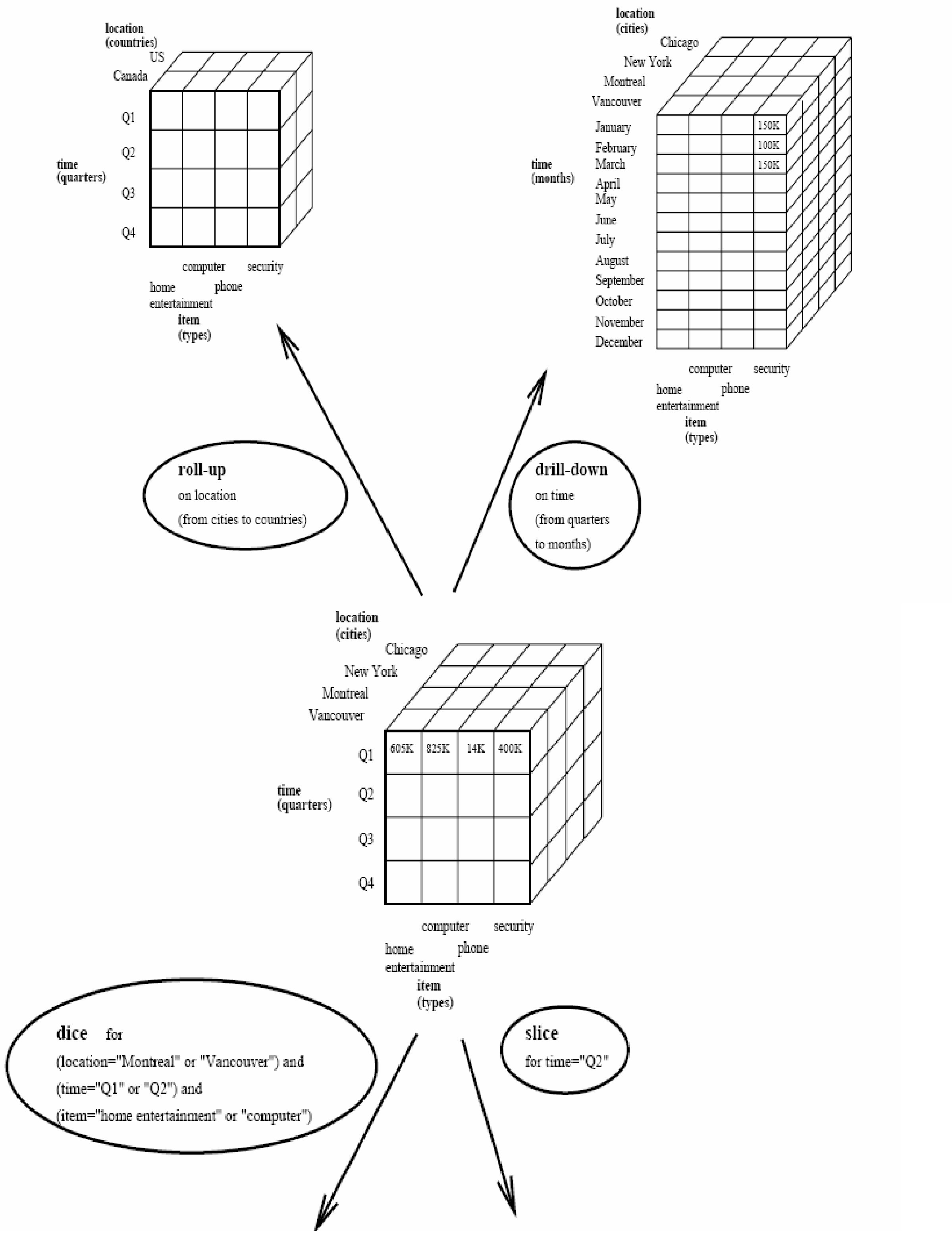
	Video	Camera	CD
NO	22	8	30
VA	23	18	22

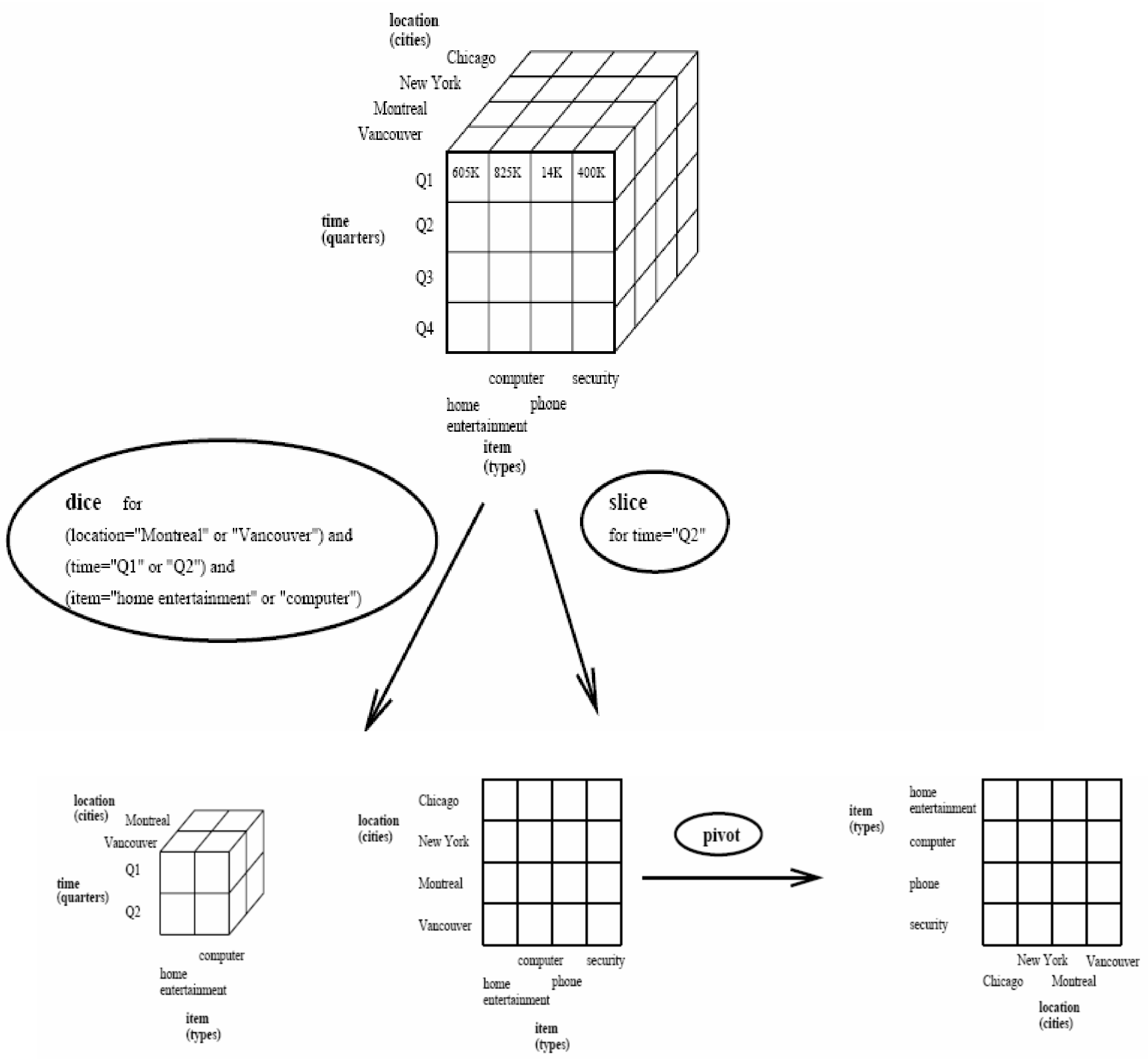
- Drill down (roll down):
    - It is the reverse of roll-up
    - It is performed by stepping down a concept hierarchy for a dimension or introducing new dimensions.

- Slice and Dice:
    - Project and Select operations
    - Check the example.

- Pivot (rotate):
    - Re-orient the cube for an alternative presentation of the data
    - Transform 3D view to series of 2D planes.

- Other operations
    - Drill across: involving (across) more than one fact table.
    - Drill through: through the bottom level of the cube to its back-end relational tables (using SQL)

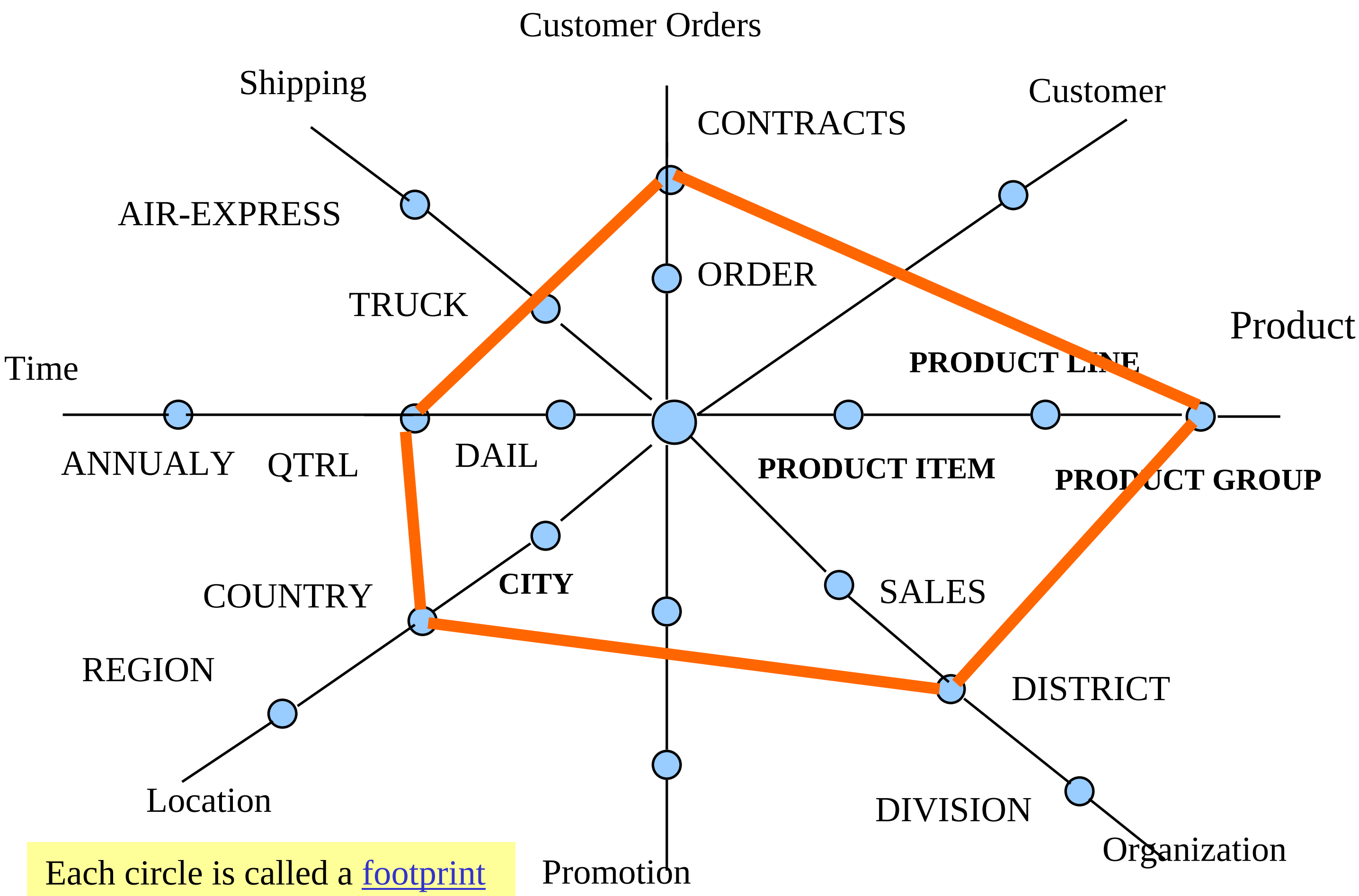

location
(countries)
US
Canada
Q1
Q2
Q3
Q4
time
(quarters)
computer
security
home
entertainment
phone
item
(types)
location
(cities)
Chicago
New York
Montreal
Vancouver
January
February
March
April
May
June
July
August
September
October
November
December
150K
100K
150K
time
(months)
computer
security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



location
(cities)
Chicago
New York
Montreal
Vancouver
Q1
605K
825K
14K
400K
time
(quarters)
Q2
Q3
Q4
computer
security
home
entertainment
phone
item
(types)
dice for
(location="Montreal" or "Vancouver") and
(time="Q1" or "Q2") and
(item="home entertainment" or "computer")
slice
for time="Q2"
pivot

9. Starnet Query Model for Multidimensional Databases

- Each radial line represents a dimension
- Each abstraction level in a hierarchy concept is called a **footprint**
- Apply OLAP operations.



Each circle is called a footprint

10. Data warehouse architecture

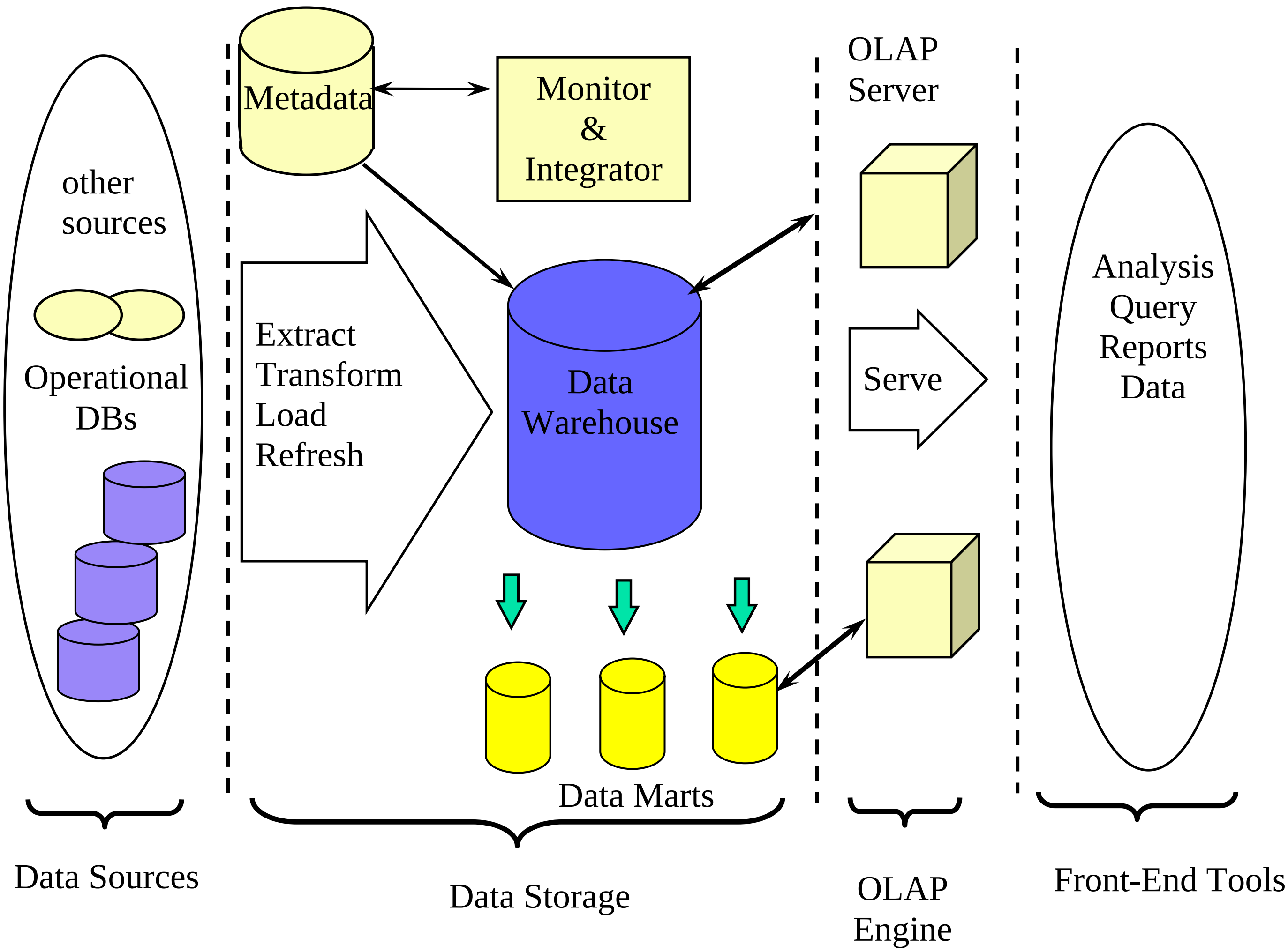
- The design of a successful DW requires the understanding and the analysis of business requirements:
    - Competitive advantage
    - Enhance business productivity
    - Cost reduction

- Four views regarding the design of a data warehouse:
    - Top-down view:
        - allows selection of the relevant information necessary for the data warehouse. It covers the current and future business needs.
    - Data source view:
        - This view exposes the information being captured, stored, and managed by operational systems.
        - Usually modeled by traditional data modeling techniques, e.g., ER model.
    - Data warehouse view:
        - This view consists of fact tables and dimension tables.
    - Business query view:
        - This view sees the perspectives of data in the warehouse from the view of end-user

10.1. DW Design Process

- Top-down, bottom-up approaches or a combination of both
- Top-down: Starts with overall design and planning (mature)
- Bottom-up: Starts with experiments and prototypes (rapid)
    - From software engineering point of view
    - Waterfall: structured and systematic analysis at each step before proceeding to the next
    - Spiral: rapid generation of increasingly functional systems, short turn around time, quick turn around
- Typical data warehouse design process
    - Choose a business process to model, e.g., orders, invoices, etc.
    - Choose the grain (atomic level of data) of the business process
    - Choose the dimensions that will apply to each fact table record
    - Choose the measure that will populate each fact table record

- Multi-Tiered Architecture



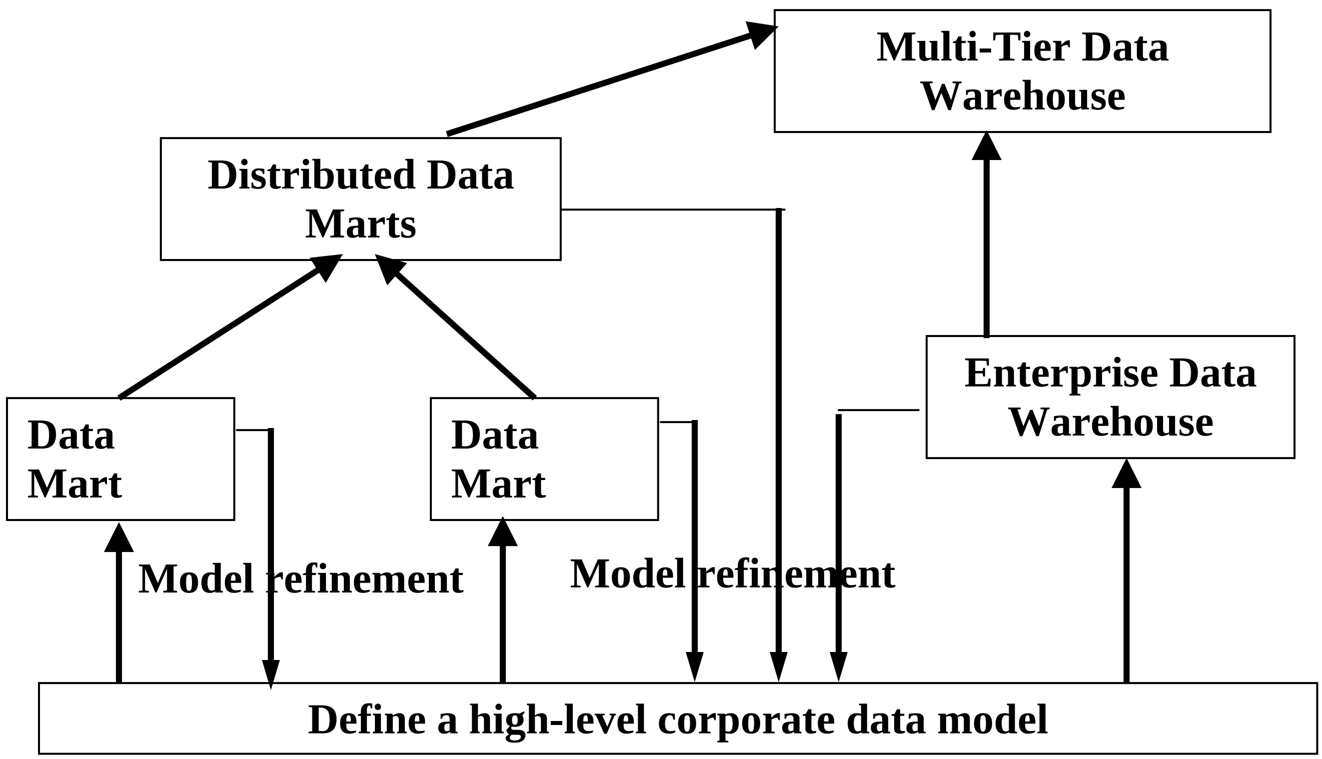
10.2. Three Data Warehouse models

- Enterprise warehouse
    - Collect all of the information about subjects spanning the entire organization.

- Data Mart
    - a subset of corporate-wide data that is of value to a specific groups of users. Its scope is confined to specific, selected groups, such as marketing data mart
        - Independent vs. dependent (directly from warehouse) data mart.

- Virtual warehouse
    - A set of views over operational databases
    - Only some of the possible summary views may be materialized

- A Recommended Approach



- Build the data warehouse incrementally, data marts → data warehouse:
    - Start with a data model
    - Build each data mart in the organization in parallel
    - Integrate the data marts

10.3. OLAP Server Architectures

- Relational OLAP (ROLAP)
    - Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware to support missing pieces
    - Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
    - greater scalability

- Multidimensional OLAP (MOLAP)
    - Array-based multidimensional storage engine (sparse matrix techniques)
    - fast indexing to pre-computed summarized data

- Hybrid OLAP (HOLAP)
    - User flexibility, e.g., low level: relational, high-level: array
    - Specialized SQL servers
    - specialized support for SQL queries over star/snowflake schemas

- How data is actually stored in ROLAP and MOLAB?

    - Two methods:
        - Base cuboid data is stored in a `base fact table
        - Aggregate data:
            - Data can be stored in the base fact table (Summary Fact table), or
            - Data can be stored in a separate summary fact tables to store each level of abstraction.

11. Data Warehouse Implementation

- Objectives:

    - **Monitoring:** Sending data from sources
    - **Integrating:** Loading, cleansing,...
    - **Processing:** Efficient cube computation, and query processing in general, indexing, ...

- Cube Computation

    - One approach extends SQL using compute cube operator
    - A cube operator is the n-dimensional generalization of the group-by SQL clause.
    - OLAP needs to compute the cuboid corresponding each input query.
    - Pre-computation: for fast response time, it seems a good idea to pre-compute data for all cuboids or at least a subset of cuboids since the number of cuboids is:

$$\text{number of cuboids} = \begin{cases} 2^n & \textit{If no hierarchy} \\ \prod_{i=1}^{n} (L_i + 1) & \begin{array}{l} \textit{if hierarchy and} \\ L_i \textit{ is number of levels} \\ \textit{associated with dimension } i \end{array} \end{cases}$$

11.1. Materialization of data cube

- Store in warehouse results useful for common queries
- Pre-compute some cuboids

- This is equivalent to the define new warehouse relations using SQL expressions
- Materialize every (cuboid) (full materialization), none (no materialization), or some (partial materialization)
- Selection of which cuboids to materialize
    - Based on size, sharing, access frequency, etc.
    - Define new warehouse relations using SQL expressions

11.2. Cube Operation

- Cube definition and computation in DMQL

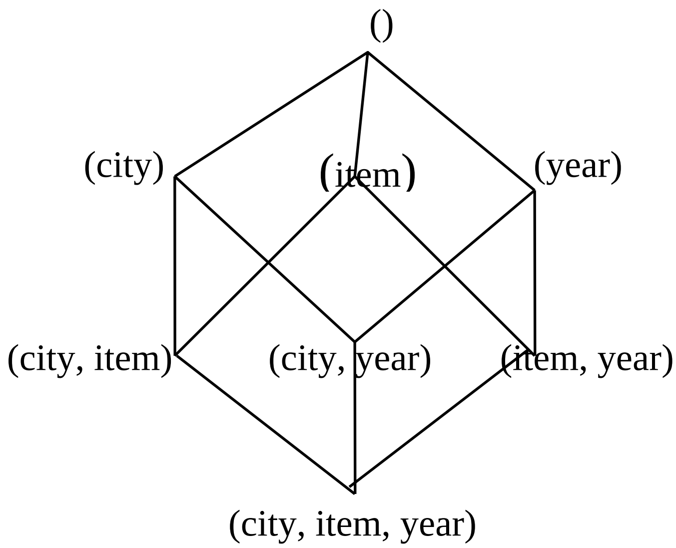
> **define** cube sales[item, city, year]: sum(sales_in_dollars)
> **compute cube** sales

- Transform it into a SQL-like language (with a new operator cube by, introduced by Gray et al.'96)

> SELECT item, city, year, SUM (amount)
> FROM SALES
> **CUBE BY** item, city, year

- Need compute the following **Group-Bys**

> *(date, product, customer),*
> *(date,product),(date, customer), (product, customer),*
> *(date), (product), (customer)*
> *()*


()
(city)
(item)
(year)
(city, item)
(city, year)
(item, year)
(city, item, year)

11.3. Cube Computation Methods

- ROLAP-based cubing
    - Sorting, hashing, and grouping operations are applied to the dimension attributes in order to reorder and cluster related tuples
    - Grouping is performed on some subaggregates as a "partial grouping step"
    - Aggregates may be computed from previously computed aggregates, rather than from the base fact table
- MOLAP Approach
    - Uses Array-based algorithm
    - The base cuboid is stored as multidimensional array.
    - Read in a number of cells to compute partial cuboids

11.4. Indexing OLAP Data: Bitmap Index

- Approach:
  - Index on a particular column
  - Each value in the column has a bit vector: bit-op is fast
  - The length of the bit vector: # of records in the base table
  - The *i*-th bit is set if the *i*-th row of the base table has the value for the indexed column
  - Not suitable for high cardinality domains
- Example:

Base Table:

Cust	Region	Type
C1	Asia	Retail
C2	Europe	Dealer
C3	Asia	Dealer
C4	America	Retail
C5	Europe	Dealer

Index on Region:

RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

Index on Type:

RecID	Retail	Dealer
1	1	0
2	0	1
3	0	1
4	1	0
5	0	1

11.5. Indexing OLAP Data: Join Indices

- Join index:

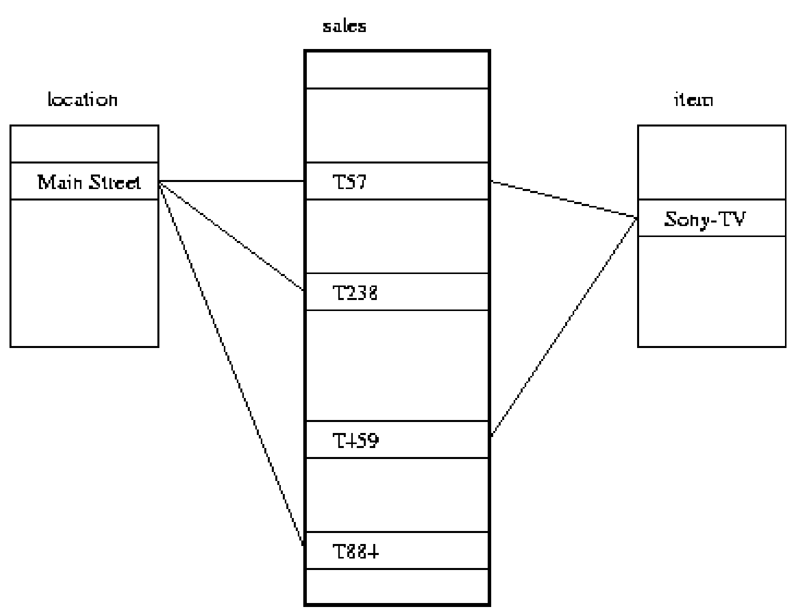
JI(R-id, S-id)

where R (R-id, …) >< S (S-id, …)

- Traditional indices map the values to a list of record ids

- It materializes relational join in JI file and speeds up relational join — a rather costly operation

- In data warehouses, join index relates the values of the dimensions of a star schema to rows in the fact table.
    - E.g. fact table: *Sales* and two dimensions *city* and *product*
        - A join index on *city* maintains for each distinct city a list of R-IDs of the tuples recording the Sales in the city
        - Join indices can span multiple dimensions



11.6. Efficient Processing OLAP Queries

- Determine which operations should be performed on the available cuboids:
    - transform drill, roll, etc. into corresponding SQL and/or OLAP operations, e.g, dice = selection + projection

- Determine to which materialized cuboid(s) the relevant operations should be applied.

- Exploring indexing structures and compressed vs. dense array structures in MOLAP

11.7. Data Warehouse Usage

- Three kinds of data warehouse applications
    - Information processing
        - supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs
    - Analytical processing
        - multidimensional analysis of data warehouse data
        - supports basic OLAP operations, slice-dice, drilling, pivoting
    - Data mining
        - knowledge discovery from hidden patterns
        - supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools.
- Differences among the three tasks

11.8. Why online analytical mining?

- High quality of data in data warehouses
  - DW contains integrated, consistent, cleaned data

- Available information processing structure surrounding data warehouses
  - ODBC, OLEDB, Web accessing, service facilities, reporting and OLAP tools

- OLAP-based exploratory data analysis
  - mining with drilling, dicing, pivoting, etc.

- On-line selection of data mining functions
  - Integration and swapping of multiple mining functions, algorithms, and tasks.

- Architecture of OLAM

12. An OLAM Architecture