Data Warehousing

and

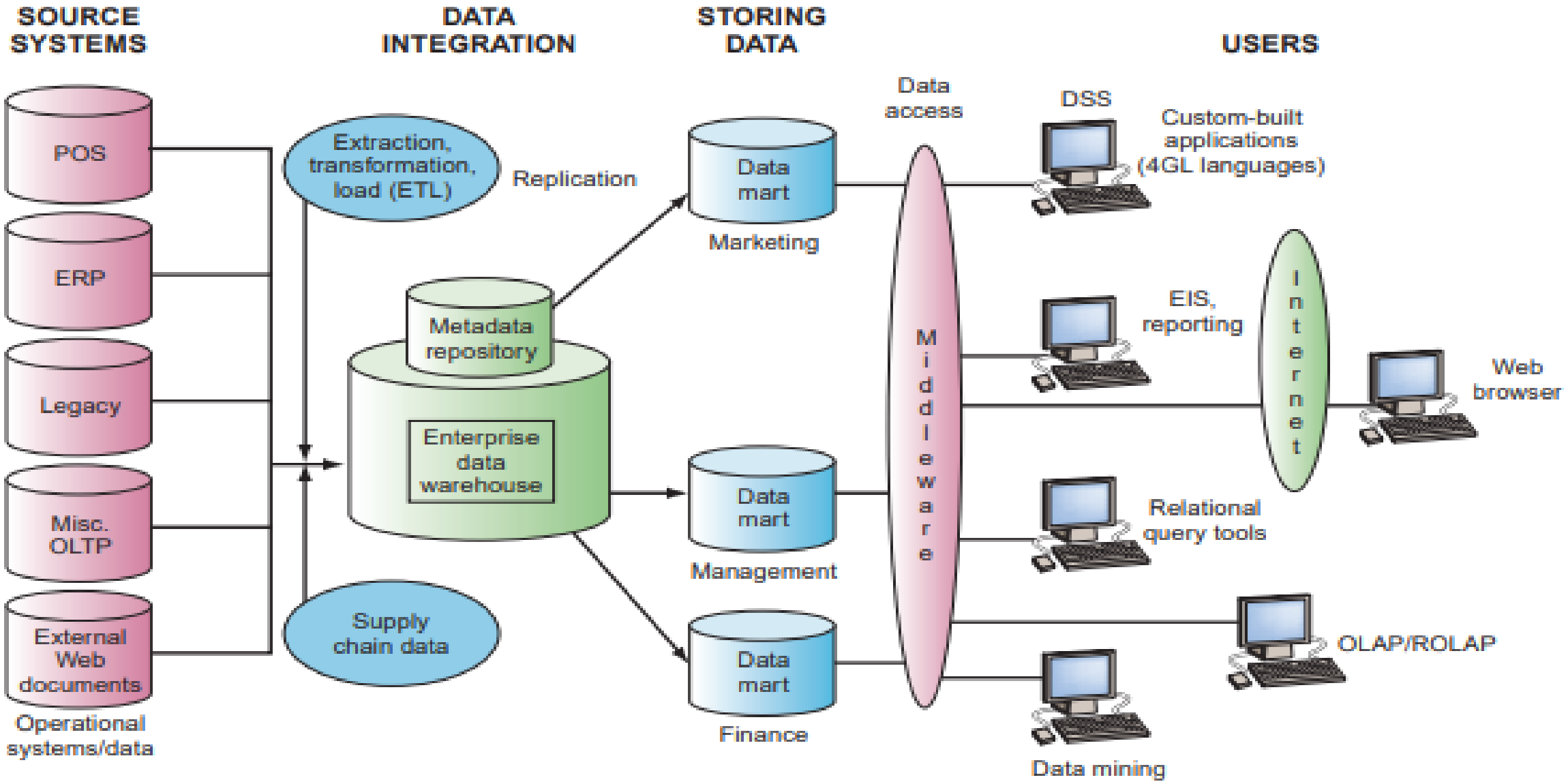
Data Mining

UNIT I

TOPICS TO BE COVERED

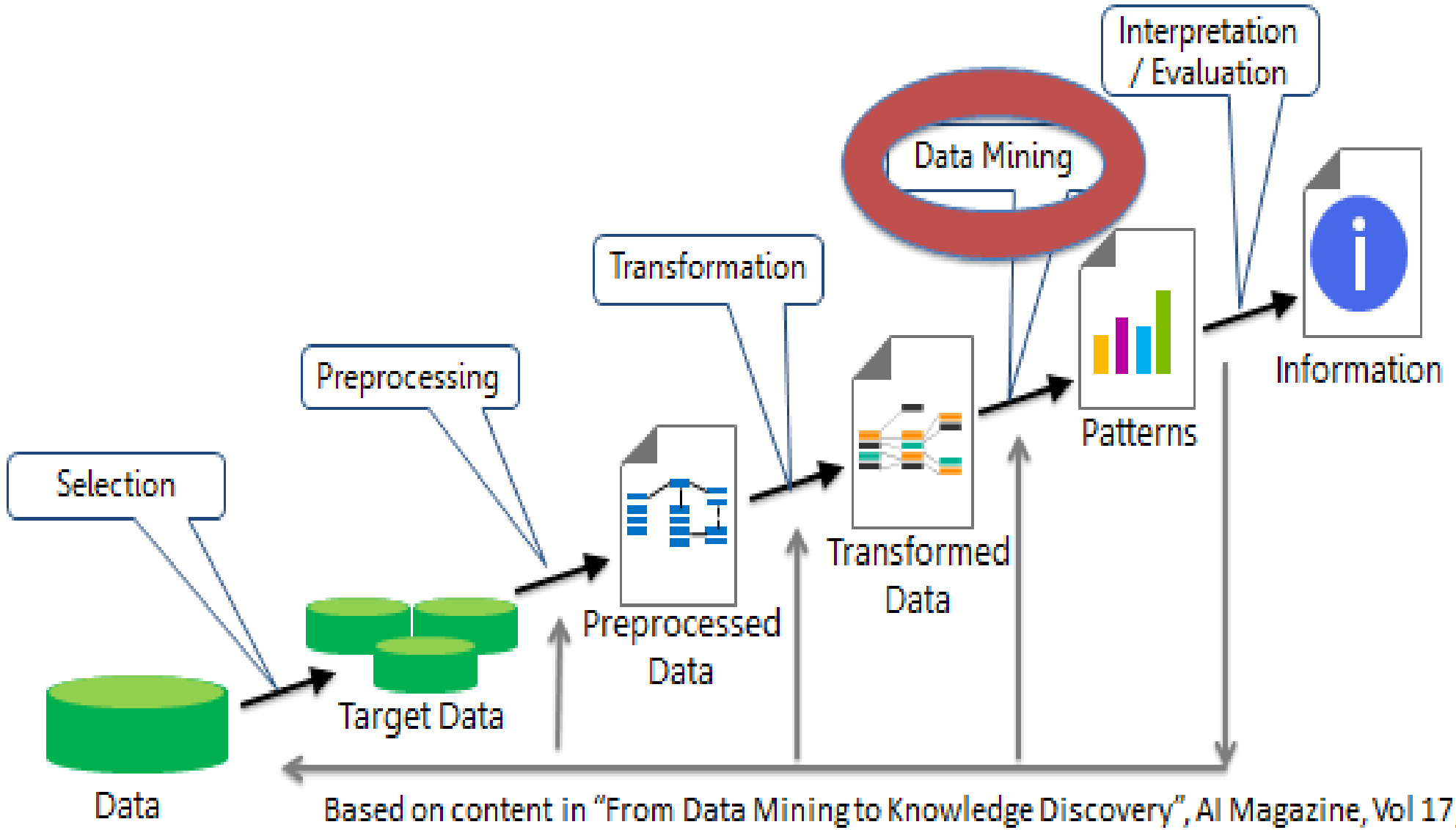
1. Basic Concepts
2. Data Ware House Modeling: Cube and OLAP
3. Data Ware House Design and Usage
4. Data Ware House Implementation

BASIC CONCEPTS



Datawarehouse and Datamart framework

KDD (Knowledge Discovery in Databases) Process



Based on content in "From Data Mining to Knowledge Discovery", AI Magazine, Vol 17, No. 3 (1996)
http://www.aaai.org/ojs/index.php/aimagazine/article/view/1230

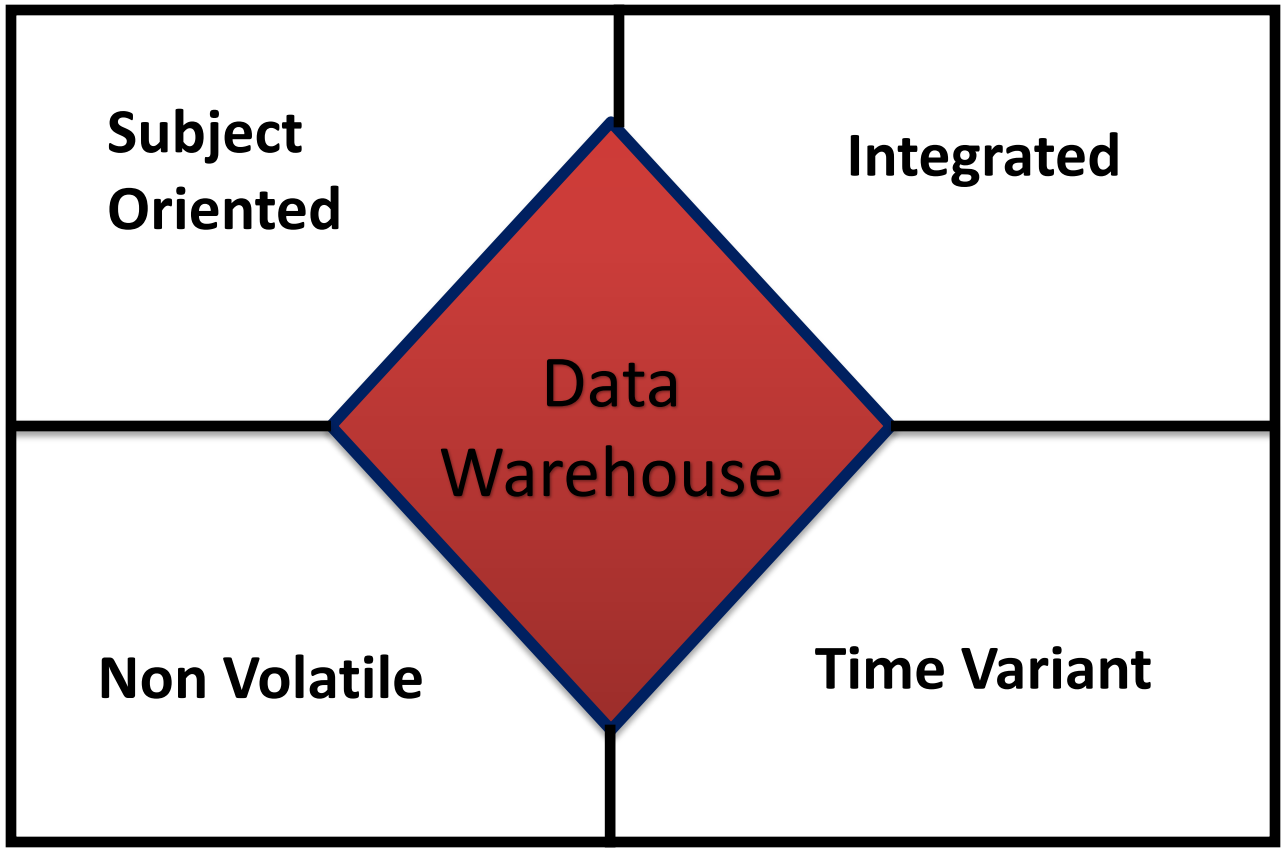
What is Data Warehouse??????

- It is an emerging data repository architecture.
- This is a repository of multiple heterogeneous data sources organized under a unified schema at a single site to facilitate management decision making.

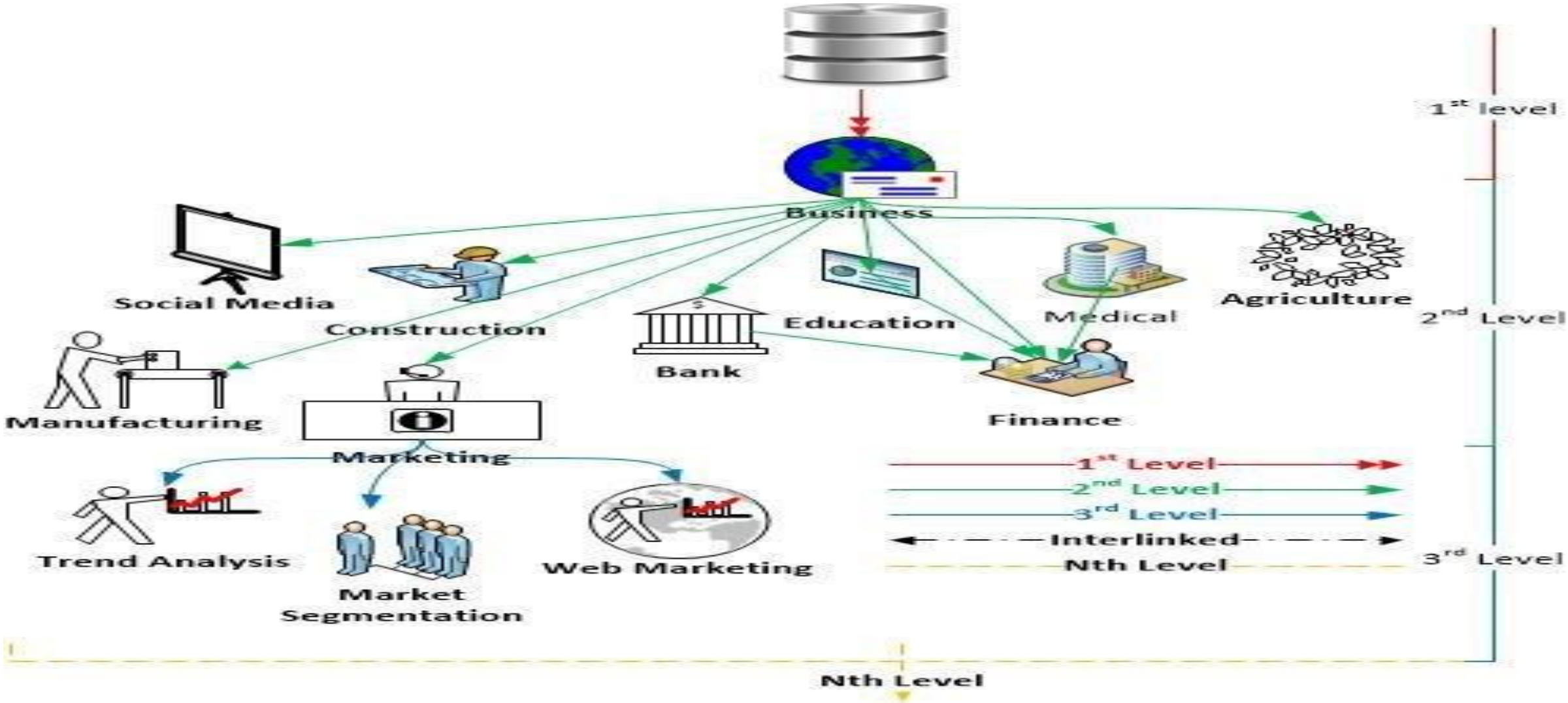
"***A data warehouse is a subject - oriented, integrated, time variant and non-volatile collection of data in support of management decision making process.***"

These four keywords—subject-oriented, integrated, time-variant, and nonvolatile distinguish data warehouses from other data repository systems, such as relational database systems, transaction processing systems, and file systems.

Data Warehouse Properties



Levels of Data Warehouse



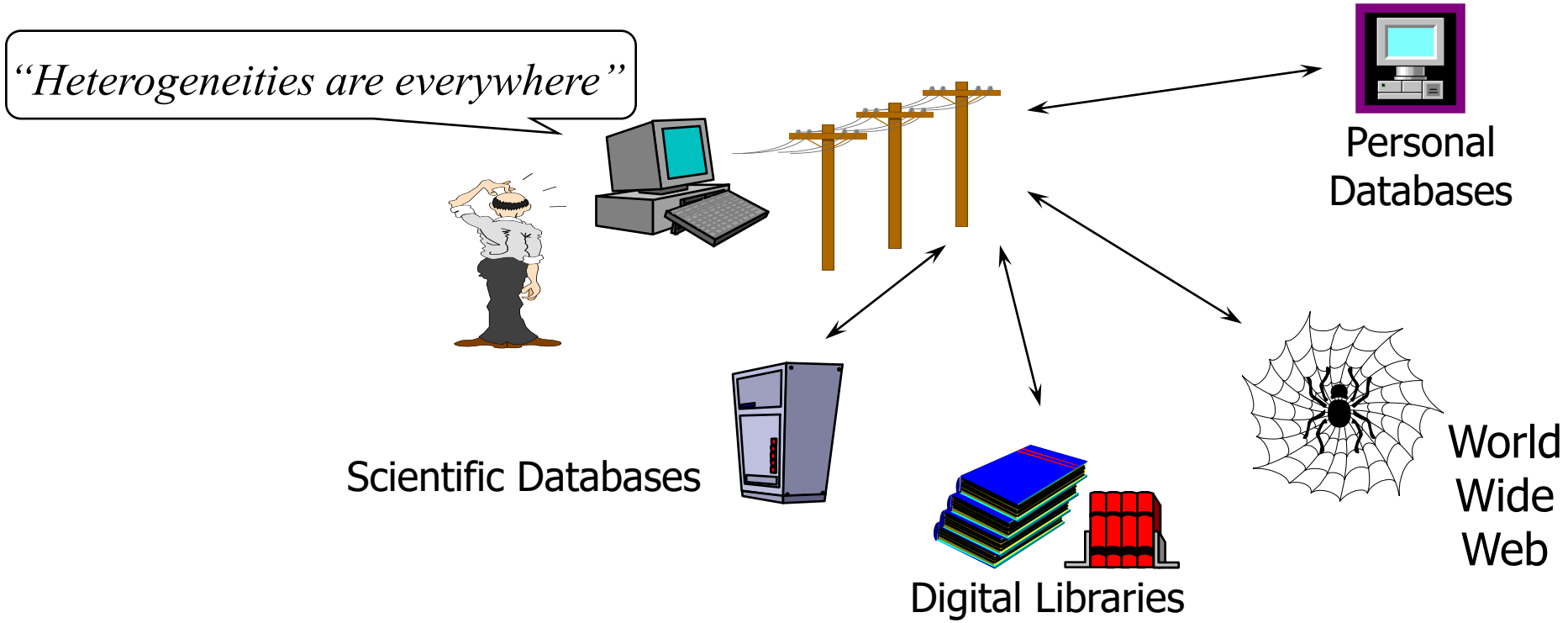
Subject-Oriented : Stored data according to target specific subjects.
Example : *It may store data regarding total Sales, Number of Customers, etc. and not general data on everyday operations*.

Integrated : Data may be distributed across heterogeneous sources which have to be integrated.
Example: *Sales data may be on RDB, Customer information on Flat files, etc.*

Time Variant : Data are stored to provide information from an historic perspective.
Example: *Data of sales in last 5 years, etc.*

Non-Volatile : It is separate from the Enterprise Operational Database and hence is not subject to frequent modification. It generally has only 2 operations performed on it: Loading of data and Access of data.

Heterogeneous Information Sources



- Different interfaces
- Different data representations
- Duplicate and inconsistent information

Features of a Warehouse…….

- It is separate from Operational Database.
- Integrates data from heterogeneous systems.
- Stores HUGE amount of data, more historical than current data.
- Does not require data to be highly accurate.
- Queries are generally complex.
- Goal is to execute statistical queries and provide results which can influence decision making in favor of the Enterprise.
- These systems are thus called Online Analytical Processing Systems (OLAP).

Need of a Separate Data Warehouse

- OLTP systems require high Concurrency, Reliability, Locking which provide good performance for short and simple OLTP queries. An OLAP query is very complex and does not require these properties. Use of OLAP query on OLTP system degrades its performance.

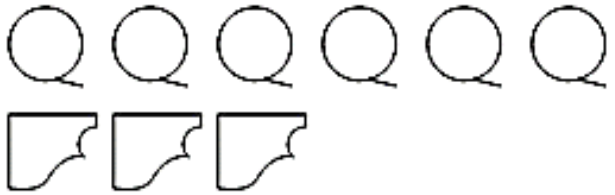
- OLAP systems access historical data and not current volatile data while OLTP systems access current up-to-date data and do not need historical data.

- An ***operational database*** is designed for known tasks like indexing and hashing using primary keys, searching for particular records, and many more.

- On the other hand, ***data warehouse*** queries are often complex. They involve the computation of large data groups at summarized levels, and may require the use of special data organization, access, and implementation methods based on multidimensional views.

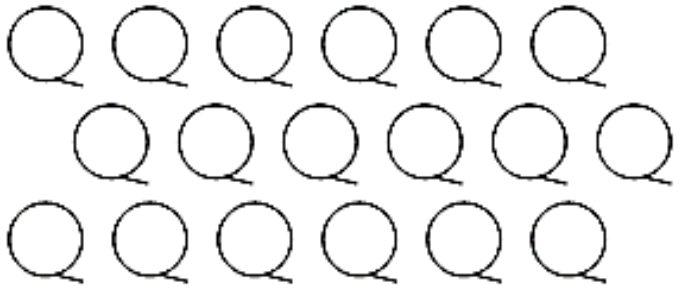
Origin/Evolution of Data Warehouse

1960



Master files, reports

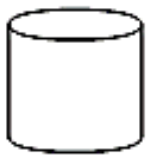
1965



Lots of master files!

- Complexity of–
  - Maintenance
  - Development
- Synchronization of data
- Hardware

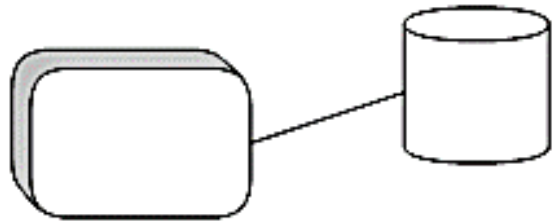
1970



DASD
DBMS

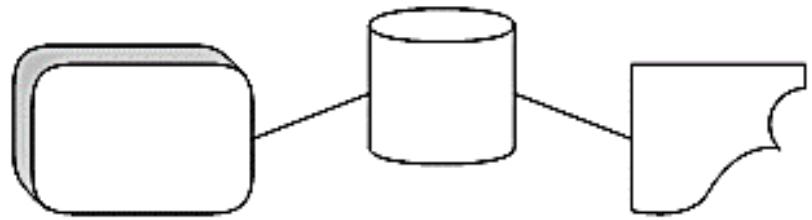
Database–"a single source of data for all processing"

1975



Online, high-performance transaction processing

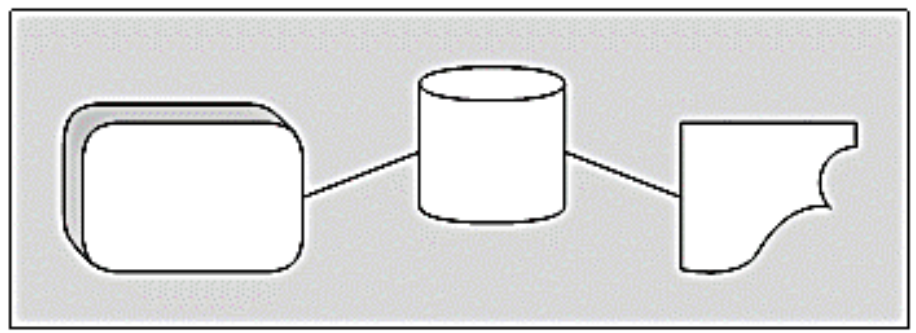
1980



tx processing

MIS/DSS

PCs, 4GL technology



The single-database-serving-all-purposes paradigm

In the early ***1960s***, the world of computation consisted of creating individual applications that were run using master files, usually built in an early language such as Fortran or COBOL. The master files were housed on **magnetic tape**. The magnetic tapes were good for storing a large volume of data cheaply, but the drawback was that they had to be accessed sequentially.

Around the ***mid-1960s***, the growth of master files and magnetic tape exploded. And with that growth came huge amounts of redundant data.

- By ***1970***, the day of a new technology for the storage and access of data had dawned. The 1970s saw the advent of disk storage, or the direct access storage device (DASD). **Disk storage** was fundamentally different from magnetic tape storage in that data could be accessed directly on a DASD. There was no need to go through records 1, 2, 3, . . . N to get to record N + 1.

- By the ***mid-1970s***, online transaction processing (OLTP) made even faster access to data possible. The computer could now be used for tasks not previously possible, including driving reservations systems, bank teller systems, manufacturing control systems, and the like.

- By the 1980s, more new technologies, such as PCs and fourth-generation languages (4GLs), began to surface.

- Key developments in early years of data warehousing:
- 1960s – General Mills and Dartmouth College, in a joint research project, develop the terms *dimensions* and *facts*.[11]
- 1970s – ACNielsen and IRI provide dimensional data marts for retail sales.[11]
- 1970s – Bill Inmon begins to define and discuss the term Data Warehouse.[*citation needed*]
- 1975 – Sperry Univac introduces MAPPER (MAintain, Prepare, and Produce Executive Reports), a database management and reporting system that includes the world's first 4GL. It is the first platform designed for building Information Centers (a forerunner of contemporary data warehouse technology).
- 1983 – Teradata introduces the DBC/1012 database computer specifically designed for decision support.[12]
- 1984 – Metaphor Computer Systems, founded by David Liddle and Don Massaro, releases a hardware/software package and GUI for business users to create a database management and analytic system.

- 1985 - Sperry Corporation publishes an article (Martyn Jones and Philip Newman) on information centers, where they introduce the term MAPPER data warehouse in the context of information centers.
- 1988 – Barry Devlin and Paul Murphy publish the article "An architecture for a business and information system" where they introduce the term "business data warehouse".[13]
- 1990 – Red Brick Systems, founded by Ralph Kimball, introduces Red Brick Warehouse, a database management system specifically for data warehousing.
- 1991 - James M. Kerr authors The IRM Imperative, which suggests data resources could be reported as an asset on a balance sheet, furthering commercial interest in the establishment of data warehouses.
- 1991 – Prism Solutions, founded by Bill Inmon, introduces Prism Warehouse Manager, software for developing a data warehouse.
- 1992 – Bill Inmon publishes the book *Building the Data Warehouse*.[14]
- 1995 – The Data Warehousing Institute, a for-profit organization that promotes data warehousing, is founded.

- 1996 – Ralph Kimball publishes the book *The Data Warehouse Toolkit*.[15]
- 2000 – Dan Linstedt releases in the public domain the Data vault modeling, conceived in 1990 as an alternative to Inmon and Kimball to provide long-term historical storage of data coming in from multiple operational systems, with emphasis on tracing, auditing and resilience to change of the source data model.
- 2008 – Bill Inmon, along with Derek Strauss and Genia Neushloss, publishes "DW 2.0: The Architecture for the Next Generation of Data Warehousing", explaining his top-down approach to data warehousing and coining the term, data-warehousing 2.0.
- 2012 – Bill Inmon develops and makes public technology known as "textual disambiguation". Textual disambiguation applies context to raw text and reformats the raw text and context into a standard data base format. Once raw text is passed through textual disambiguation, it can easily and efficiently be accessed and analyzed by standard business intelligence technology. Textual disambiguation is accomplished through the execution of textual ETL. Textual disambiguation is useful wherever raw text is found, such as in documents, Hadoop, email, and so forth.

Differences between Operational Database Systems and Data Warehouses

The major task of ***operational database systems*** is to perform online transaction and query processing. These systems are called ***Online Transaction Processing*** (OLTP) systems. It covers most of the day-to-day operations of an organization such as purchasing, inventory, manufacturing, banking, payroll, registration, and accounting.

Data warehouse systems, on the other hand, serve users or knowledge workers in the role of data analysis and decision making. These systems are known as online analytical processing (OLAP) systems.

	Operational DB Systems	Data Warehouse Systems
Users	An OLTP system is customer-oriented and used by clerks, clients, and IT professionals.	An OLAP system is market-oriented and is used for data analysis by knowledge workers, including managers, executives, and analysts.
Data Contents	Current and detailed data and is subject to modifications.	Historical data generally not modified.
Database Design	Usually E-R model. (*Application oriented Database Design*)	Usually Multidimensional model (Star, Snowflake…). (*Subject Oriented Database Design*)
Access Pattern (Nature of Query)	OLTP systems mainly consist for Short and atomic transaction desiring high performance (less latency) and accuracy. It requires Concurrence Control and Recovery mechanisms.	Mostly read only queries, operate on HUGE volumes of data, queries are quite complex.
Database Size	In Gigabytes to higher order Gigabytes.	> = Terabytes or Petabytes.
View	Detailed	Summarized.

Parameter	Database	Data Warehouse
Purpose	Is designed to record	Is designed to analyze
Processing Method	The database uses the Online Transactional Processing (OLTP)	Data warehouse uses Online Analytical Processing (OLAP).
Usage	The database helps to perform fundamental operations for your business	Data warehouse allows you to analyze your business.
Tables and Joins	Tables and joins of a database are complex as they are normalized.	Table and joins are simple in a data warehouse because they are denormalized.
Orientation	Is an application-oriented collection of data	It is a subject-oriented collection of data
Storage limit	Generally limited to a single application	Stores data from any number of applications
Availability	Data is available real-time	Data is refreshed from source systems as and when needed
Usage	ER modeling techniques are used for designing.	Data modeling techniques are used for designing.
Technique	Capture data	Analyze data
Data Type	Data stored in the Database is up to date.	Current and Historical Data is stored in Data Warehouse. May not be up to date.
Storage of data	Flat Relational Approach method is used for data storage.	Data Ware House uses dimensional and normalized approach for the data structure. Example: Star and snowflake schema.
Query Type	Simple transaction queries are used.	Complex queries are used for analysis purpose.
Data Summary	Detailed Data is stored in a database.	It stores highly summarized data.

Applications of Database	
Sector	**Usage**
Banking	Use in the banking sector for customer information, account-related activities, payments, deposits, loans, credit cards, etc.
Airlines	Use for reservations and schedule information.
Universities	To store student information, course registrations, colleges, and results.
Telecommunication	It helps to store call records, monthly bills, balance maintenance, etc.
Finance	Helps you to store information related stock, sales, and purchases of stocks and bonds.
Sales & Production	Use for storing customer, product and sales details.
Manufacturing	It is used for the data management of the supply chain and for tracking production of items, inventories status.
HR Management	Detail about employee’s salaries, deduction, generation of paychecks, etc.

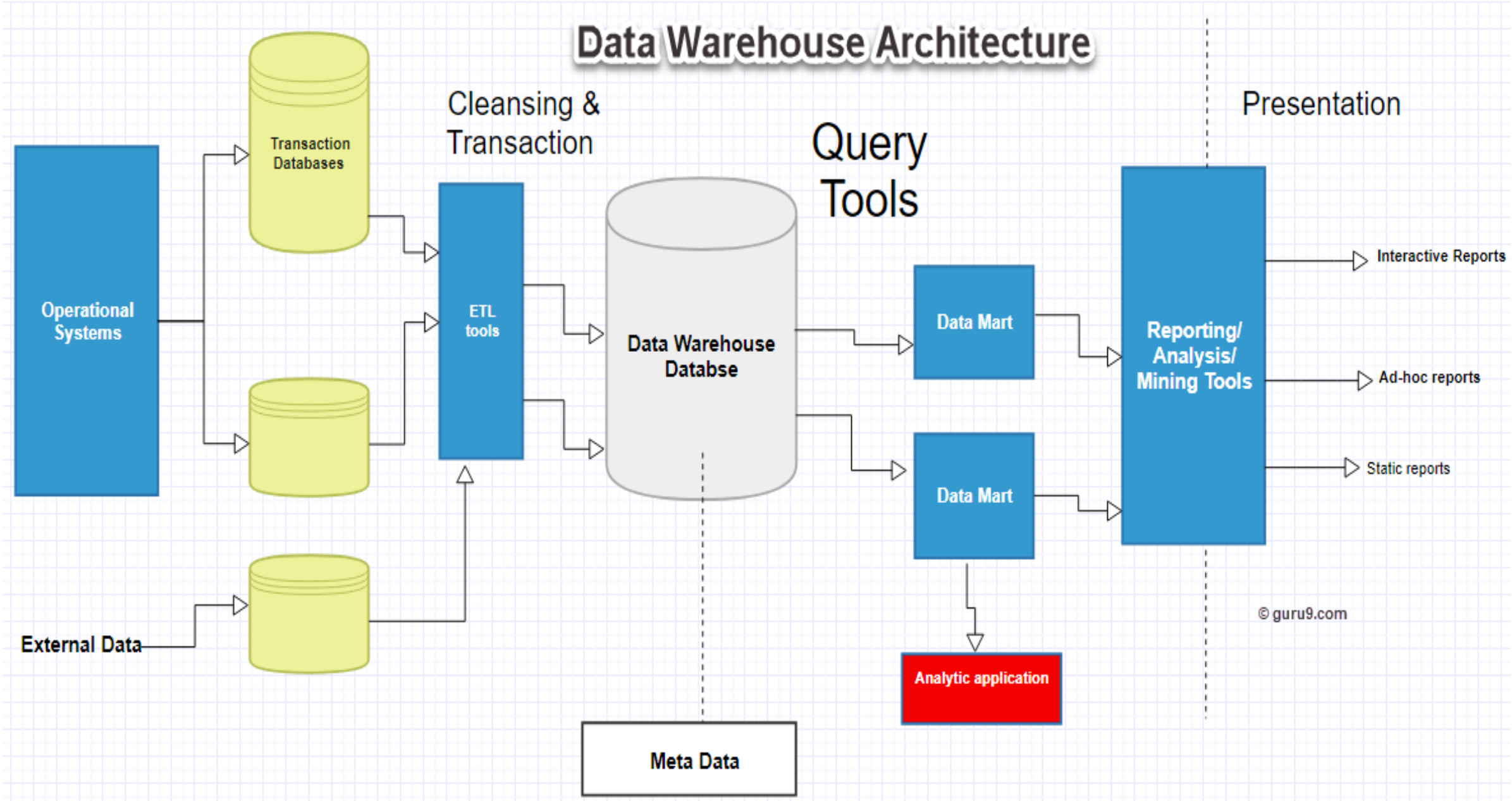
Applications of Data Warehouses	
Sector	**Usage**
Airline	It is used for airline system management operations like crew assignment, analyzes of route, frequent flyer program discount schemes for passenger, etc.
Banking	It is used in the banking sector to manage the resources available on the desk effectively.
Healthcare sector	Data warehouse used to strategize and predict outcomes, create patient's treatment reports, etc. Advanced machine learning, big data enable datawarehouse systems can predict ailments.
Insurance sector	Data warehouses are widely used to analyze data patterns, customer trends, and to track market movements quickly.
Retain chain	It helps you to track items, identify the buying pattern of the customer, promotions and also used for determining pricing policy.
Telecommunication	In this sector, data warehouse used for product promotions, sales decisions and to make distribution decisions.

Advantages of Data Warehouse (DWH)

- Data warehouse allows business users to quickly access critical data from some sources all in one place.
- Data warehouse provides consistent information on various cross-functional activities. It is also supporting ad-hoc reporting and query.
- Data Warehouse helps to integrate many sources of data to reduce stress on the production system.
- Data warehouse helps to reduce total turnaround time for analysis and reporting.
- Restructuring and Integration make it easier for the user to use for reporting and analysis.
- Data warehouse allows users to access critical data from the number of sources in a single place. Therefore, it saves user's time of retrieving data from multiple sources.
- Data warehouse stores a large amount of historical data. This helps users to analyze different time periods and trends to make future predictions.

Disadvantages of Data Warehouse:

- Not an ideal option for unstructured data.
- Creation and Implementation of Data Warehouse is surely time confusing affair.
- Data Warehouse can be outdated relatively quickly
- Difficult to make changes in data types and ranges, data source schema, indexes, and queries.
- The data warehouse may seem easy, but actually, it is too complex for the average users.
- Despite best efforts at project management, data warehousing project scope will always increase.
- Sometime warehouse users will develop different business rules.
- Organizations need to spend lots of their resources for training and Implementation purpose.


Data Warehouse Architecture
Cleansing & Transaction
Query Tools
Presentation
Operational Systems
Transaction Databases
ETL tools
Data Warehouse Databse
Data Mart
Data Mart
Reporting/ Analysis/ Mining Tools
Interactive Reports
Ad-hoc reports
Static reports
External Data
Analytic application
Meta Data
© guru9.com

Data Warehouse Tools

There are many Data Warehousing tools are available in the market

1. MarkLogic:

MarkLogic is useful data warehousing solution that makes data integration easier and faster using an array of enterprise features. This tool helps to perform very complex search operations. It can query different types of data like documents, relationships, and metadata.

https://www.marklogic.com/product/getting-started/

2. Oracle:

Oracle is the industry-leading database. It offers a wide range of choice of data warehouse solutions for both on-premises and in the cloud. It helps to optimize customer experiences by increasing operational efficiency.

https://www.oracle.com/index.html

3. Amazon RedShift:

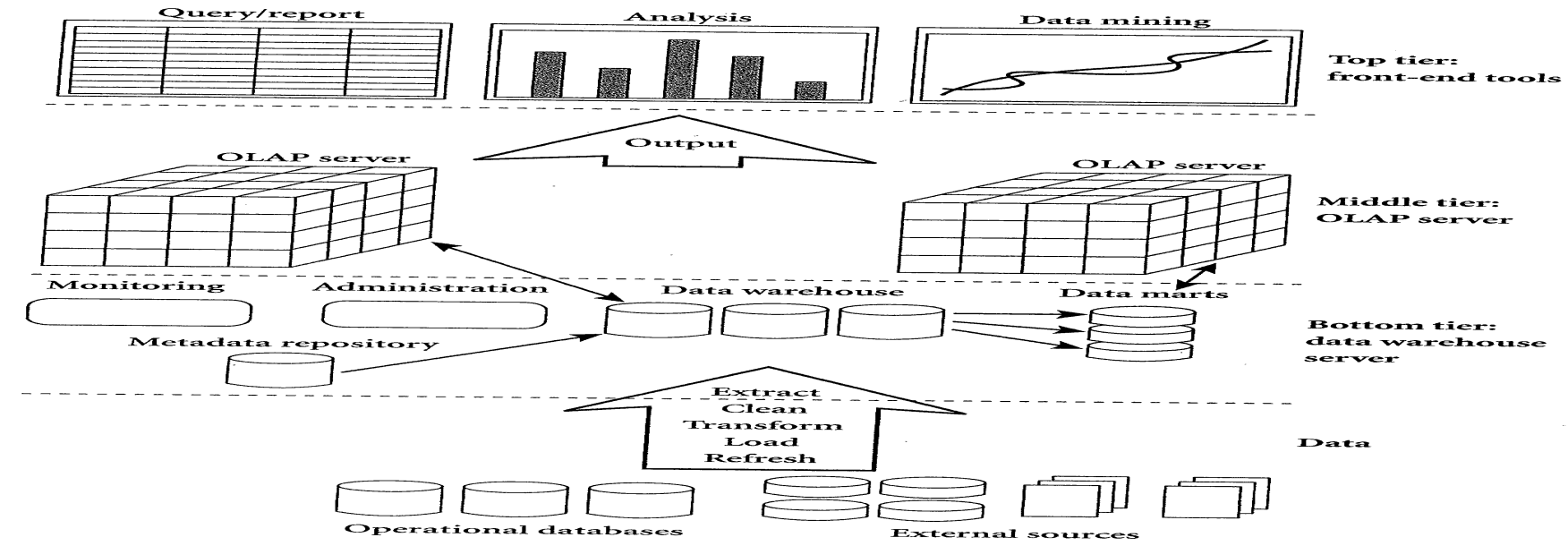
Amazon Redshift is Data warehouse tool. It is a simple and cost-effective tool to analyze all types of data using standard SQL and existing BI tools. It also allows running complex queries against petabytes of structured data, using the technique of query optimization.

https://aws.amazon.com/redshift/?nc2=h_m1

Many other tools are also available at https://www.guru99.com/top-20-etl-database-warehousing-tools.html

Data Warehouse Modeling: Cube and OLAP

OLAP (Online Analytical Processing)



Online Analytical Processing

An OLAP system manages large amount of historical data, provides facilities for summarization and aggregation, and stores and manages information at different levels of granularity.

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, 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

From Tables and Spreadsheets to Data Cubes

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

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

- 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.

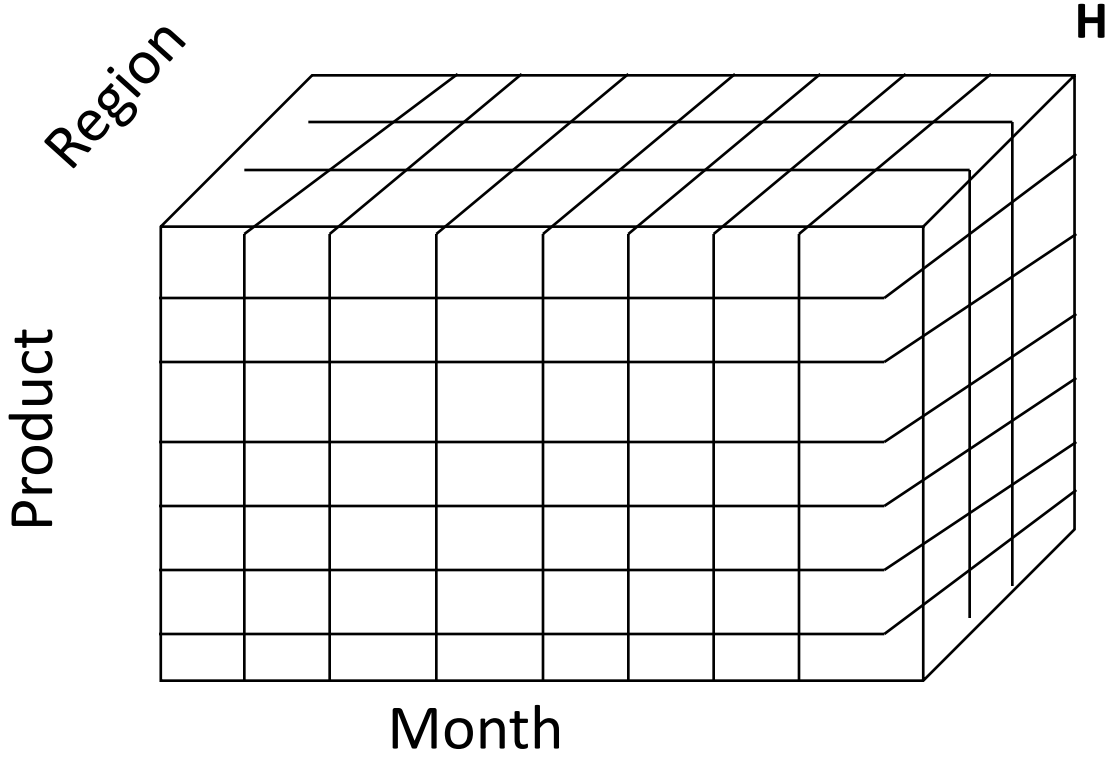
OLTP vs. OLAP

OLAP Server Architectures

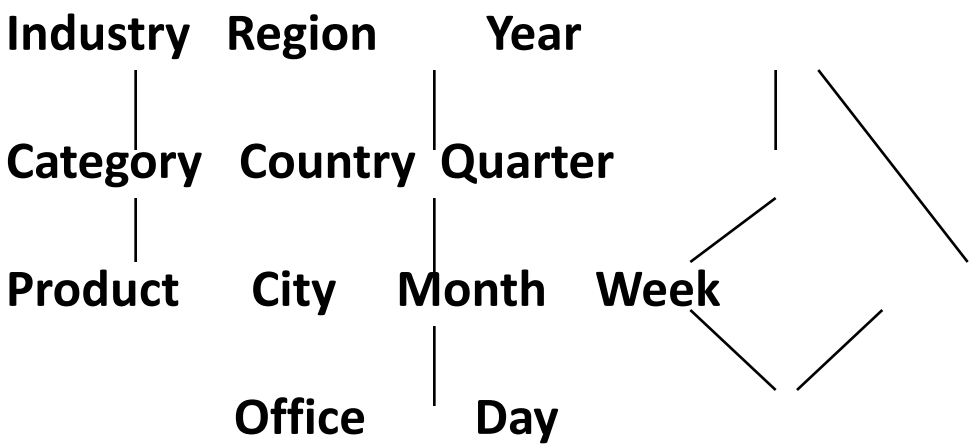
- Relational OLAP (ROLAP):
  - Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware to support missing pieces.
  - Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
  - greater scalability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

Multidimensional Data

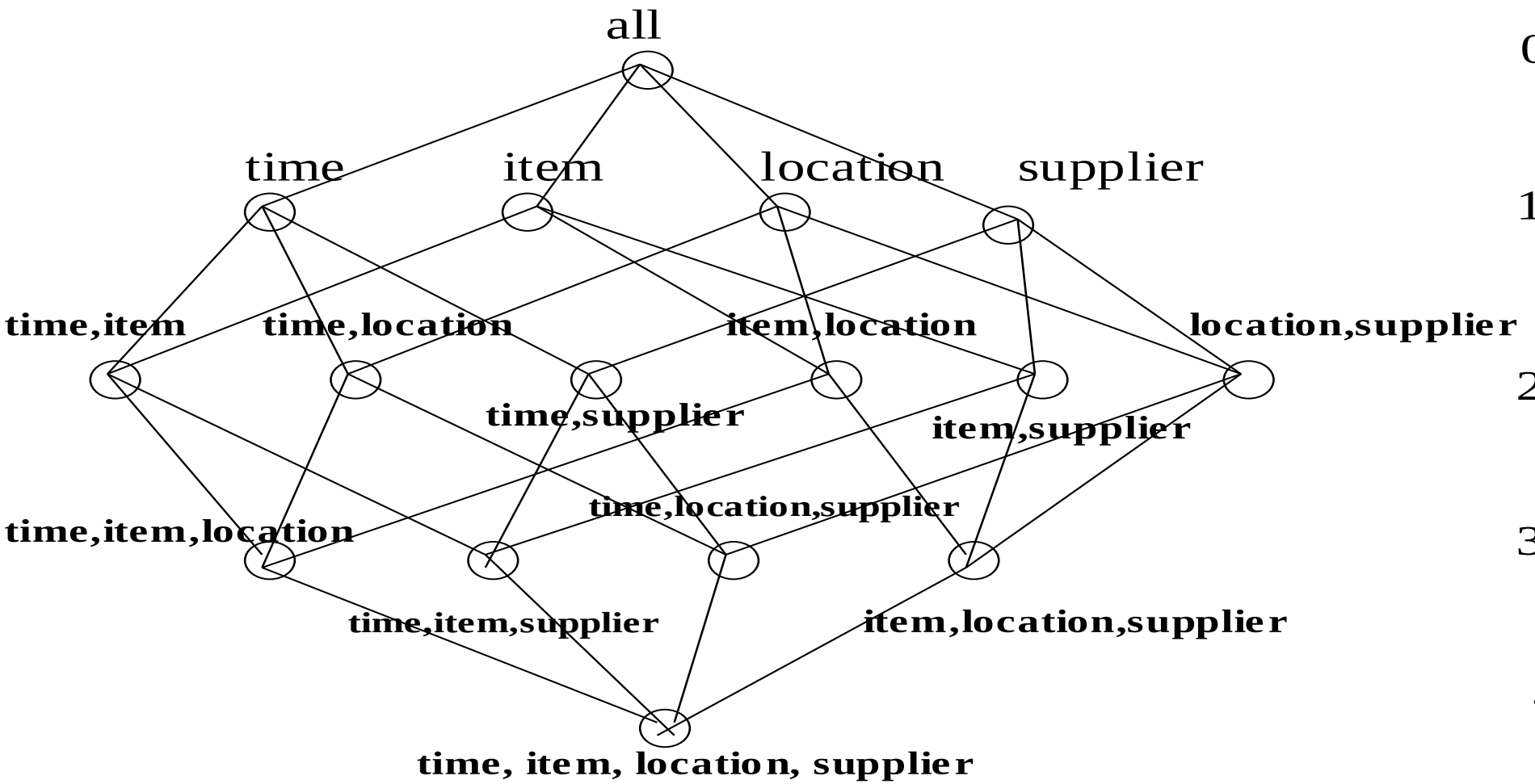
- Sales volume as a function of product, month, and region



Dimensions: Product, Location, Time
Hierarchical summarization paths



Cube: A Lattice of Cuboids



0-D(apex) cuboid

1-D cuboids

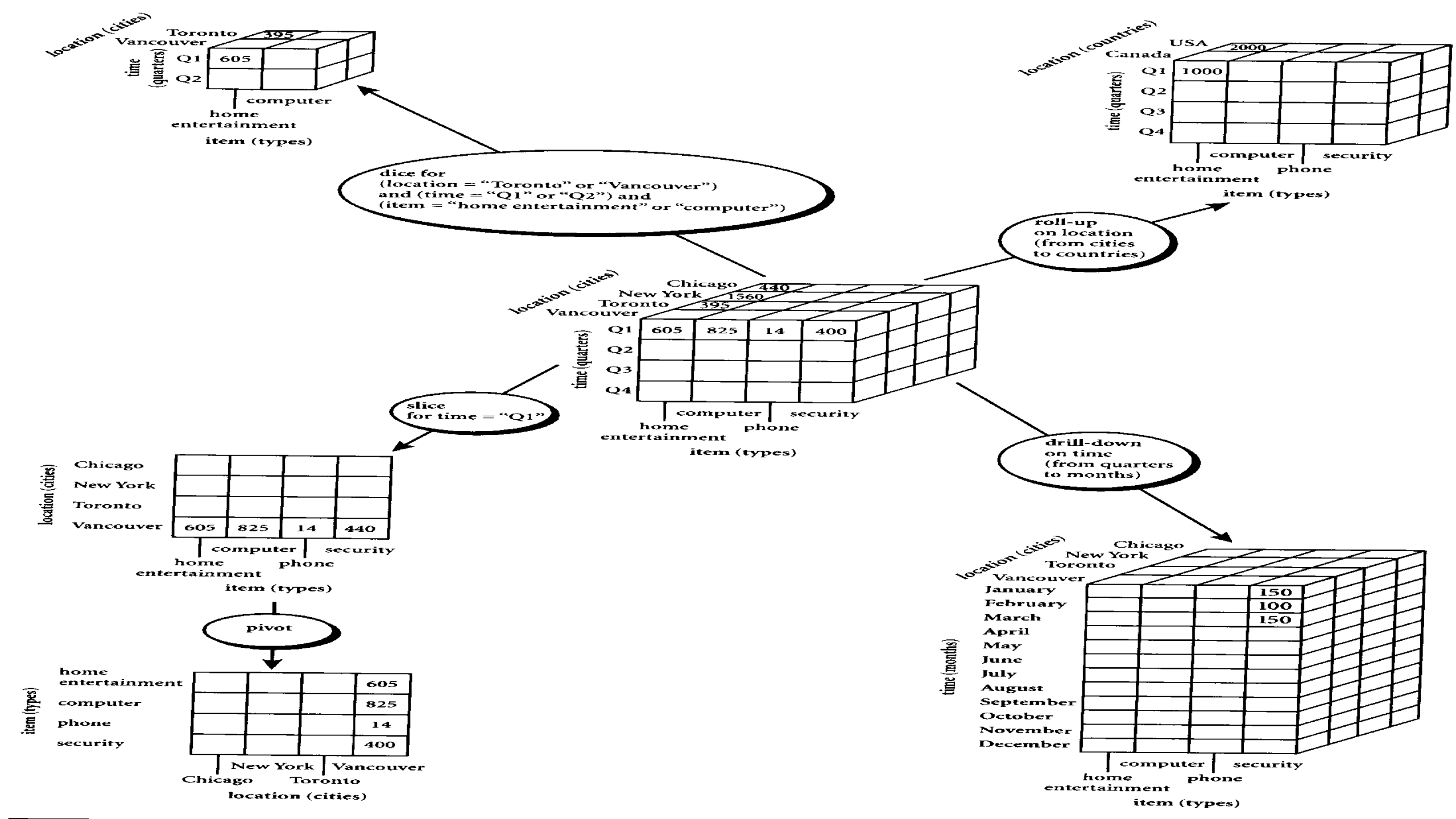
2-D cuboids

3-D cuboids

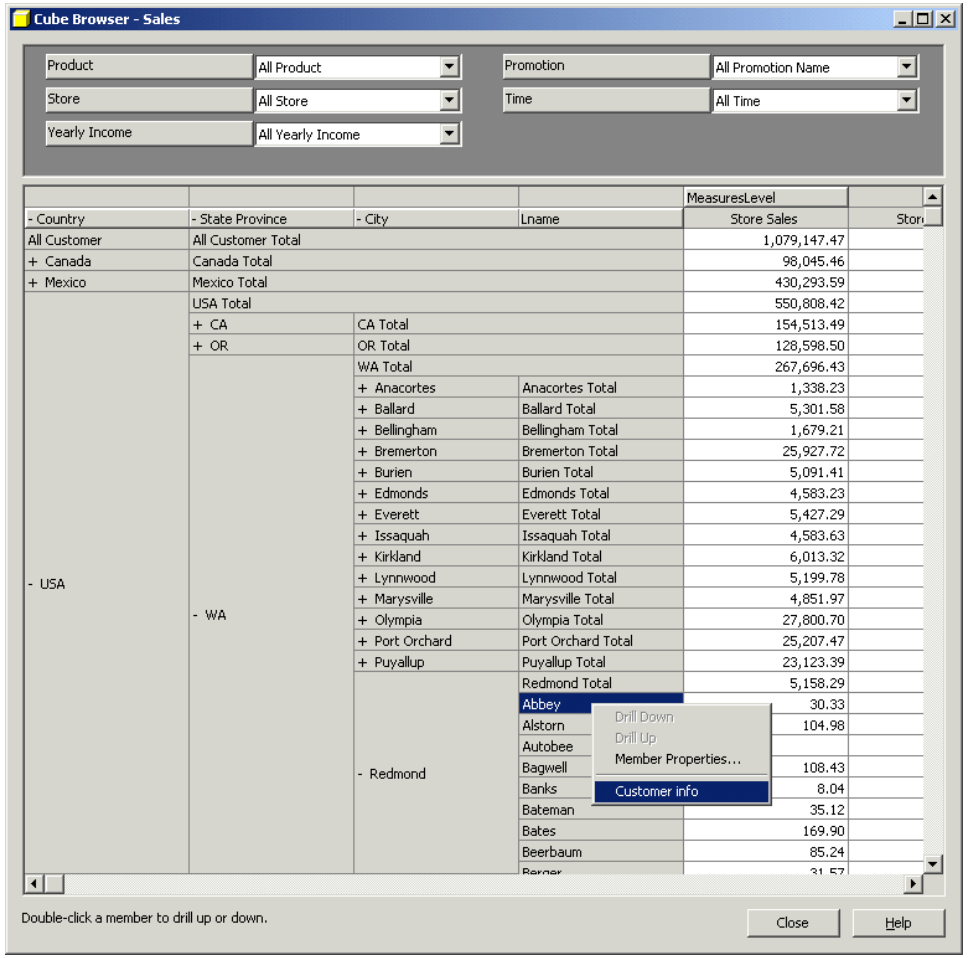
4-D(base) cuboid

Typical OLAP Operations

- **Roll up (drill-up):** summarize data
  - *by climbing up hierarchy or by dimension reduction*
- **Drill down (roll down):** reverse of roll-up
  - *from higher level summary to lower level summary or detailed data, or introducing new dimensions*
- **Slice and dice:**
  - *project and select*
- **Pivot (rotate):**
  - *reorient the cube, visualization, 3D to series of 2D planes.*
- Other operations
  - ***drill through**: through the bottom level of the cube to its back-end relational tables (using SQL)*


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

Sample OLAP Drill down online report



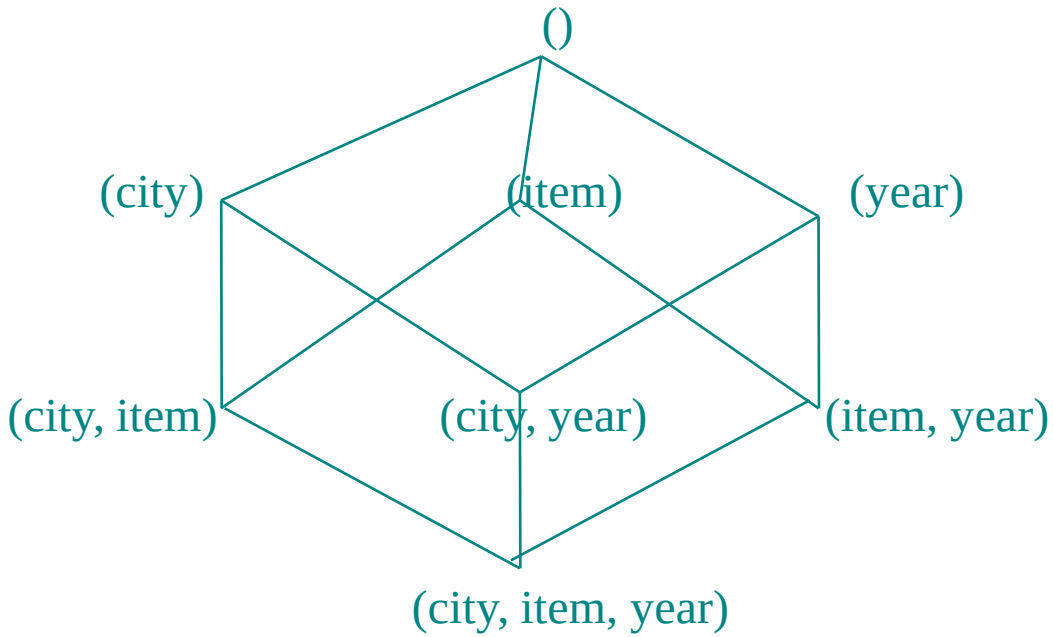
Cube Operation

- Cube definition and computation in OLAP
  1. define cube sales[item, city, year]: sum(sales_in_dollars)
  2. compute cube sales
- Transform it into a SQL-like language (with a new operator cube by)

```
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)
()



Roll-up and Drill-down

The roll-up operation performs aggregation on a data cube, either by climbing up a concept hierarchy for a dimension or by dimension reduction such that one or more dimensions are removed from the given cube.

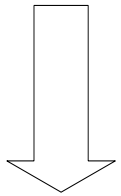
Drill-down is the reverse of roll-up. It navigates from less detailed data to more detailed data. Drill-down can be realized by either stepping down a concept hierarchy for a dimension or introducing additional dimensions.

Slice and dice

The slice operation performs a selection on one dimension of the given cube, resulting in a sub_cube.

The dice operation defines a sub_cube by performing a selection on two or more dimensions.

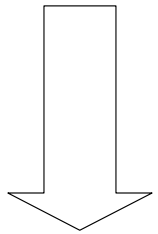
	Food Line	Outdoor Line	CATEGORY_total
Asia	59,728	151,174	210,902



Drill-Down

	Food Line	Outdoor Line	CATEGORY_total
Malaysia	618	9,418	10,036
China	33,198.5	74,165	107,363.5
India	6,918	0	6,918
Japan	13,871.5	34,965	48,836.5
Singapore	5,122	32,626	37,748
Belgium	7797.5	21,125	28,922.5

	Food Line	Outdoor Line	CATEGORY_total
Canada	29,116.5	69,310	98,426.5
Mexico	12,743.5	24,284	37,027.5
United States	102,561.5	232,679	335,240.5

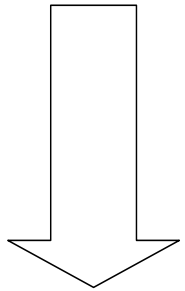


Roll-Up

	Food Line	Outdoor Line	CATEGORY_total
North America	144,421.5	326,273	470,694.5

	Food Line	Outdoor Line	**CATEGORY_total**
Asia	59,728	151,174	210,902
Europe	97,580.5	213,304	310,884.5
North America	144,421.5	326,273	470,694.5
REGION_total	301,730	690,751	992,481

	Food Line	Outdoor Line	CATEGORY_total
Canada	29,116.5	69,310	98,426.5
Mexico	12,743.5	24,284	37,027.5
United States	102,561.5	232,679	335,240.5



Dice

	Food Line	Outdoor Line
Mexico	12,743.5	24,284
United States	102,561.5	232,679

Querying with MDX (Multidimensional Expressions)

The select clause defines axis dimensions on COLUMNS and on ROWS, where clause supplies slicer dimensions, and Cube is the name of the data cube.

Select axis [, axis]

From Cube

Where slicer [, slicer]

The Data Hierarchy

- For the majority of MDX statements, the context of the query will be limited to a single cube. It is important to know how all data within a cube is divided into the following relationship:

Dimensions

Hierarchies

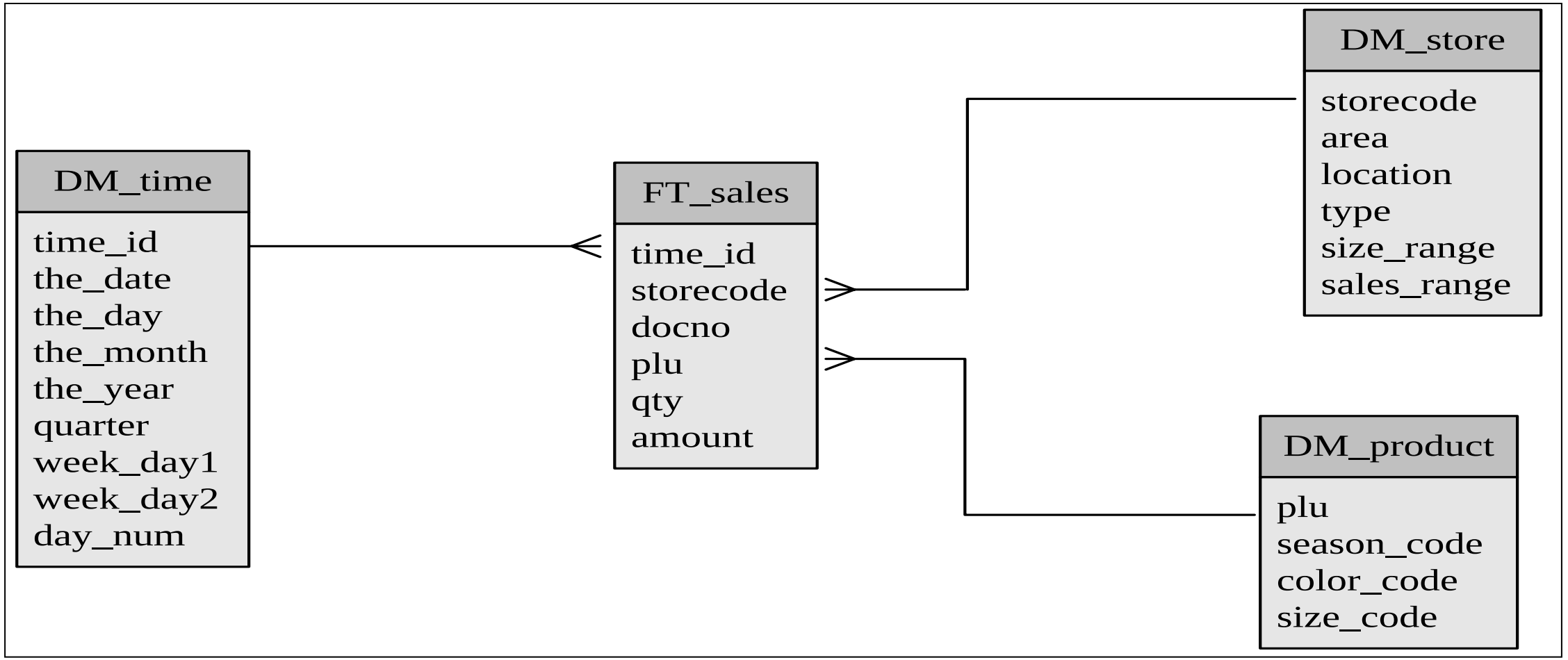
Levels

Members

Sample MDX where italic are default

```
SELECT
 {[Gender].[Gender].Members} ON COLUMNS,
{[Product].[Product Family].Members} ON ROWS,
FROM [Sales]
WHERE
([Measures].[Unit Sales],
[Customers].[All Customers],
[Education Level].[All Education Level],
[Marital Status].[All Martial status],
[Promotions].[All Promotions],
[Store].[All Stores],
[Store Size in SQFT].[All],
[Store Type].[All],
[Yearly Income].[All Yearly Income]
```

Example on Star Schema



Example on Roll-up

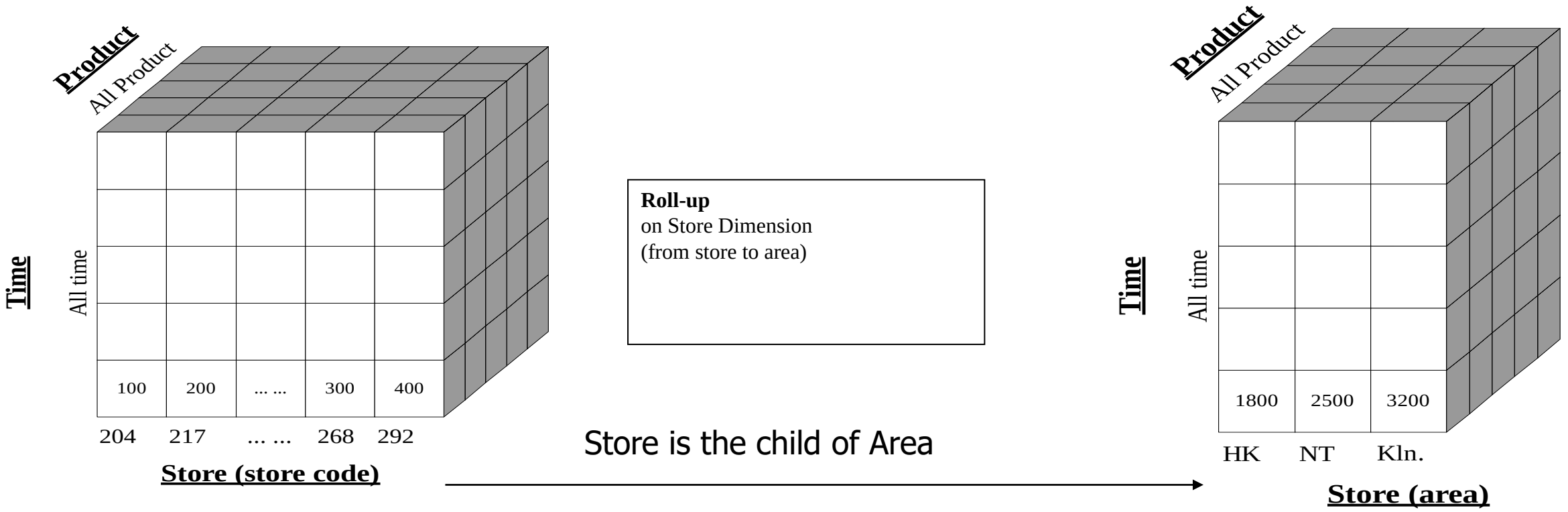
MDX

```
SELECT [SALES].[AMOUNT] ON COLUMNS,
[store].[Kowloon] ON ROWS
FROM SALES
```

SQL

```
select sum(amount), area
from SALES
where  (area='Kowloon') group by area
```

Graphical Description on Roll-up Example



Example on Drill-down

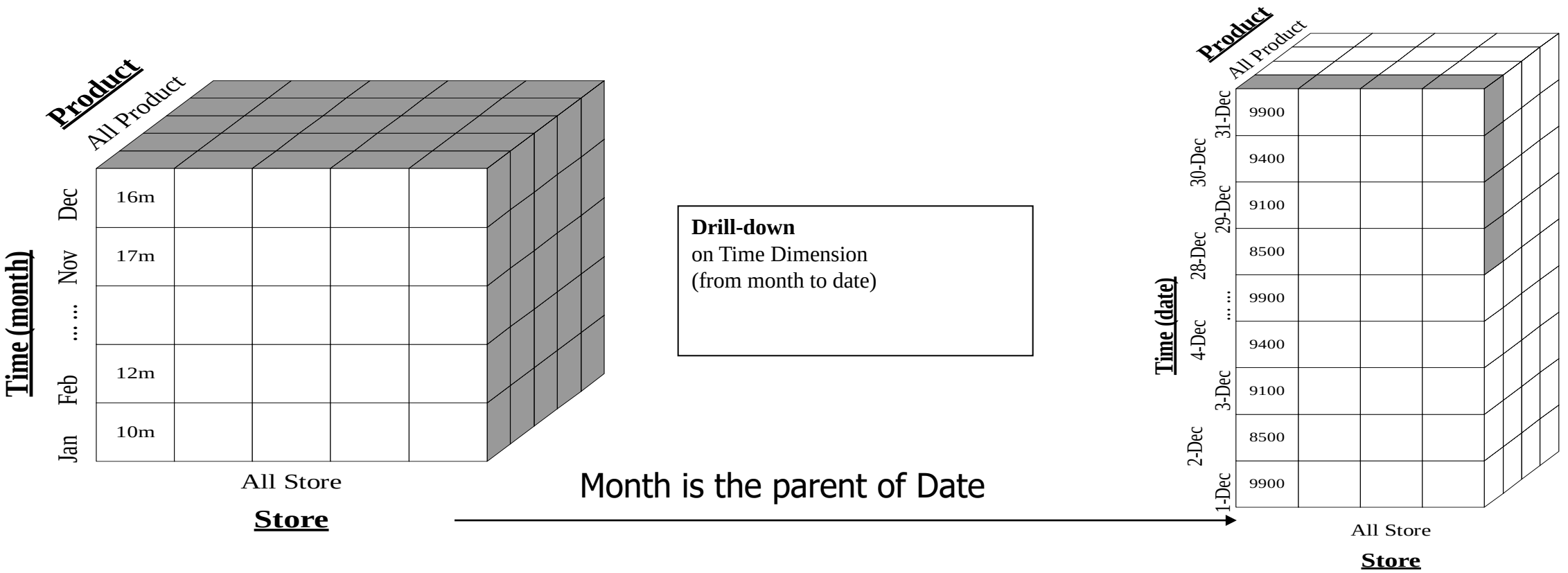
MDX

```
SELECT [SALES].[AMOUNT] ON COLUMNS,
[time].[2003].[Q4].[Dec].[31],
[time].[2003].[Q4].[Dec].[30],… …,
[time].[2003].[Q4].[Dec].[2],
[time].[2003].[Q4].[Dec].[1] ON ROWS FROM SALES
```

SQL

```
select sum(amount), the_date
from SALES
where  (the_date='2003-Dec-31')
or (the_date='2003-Dec-30')
or… …or (the_date='2003-Dec-2')
or (the_date='2003-Dec-1') group by the_date
```

Graphical Description of Drill-down Example



Example on Slice

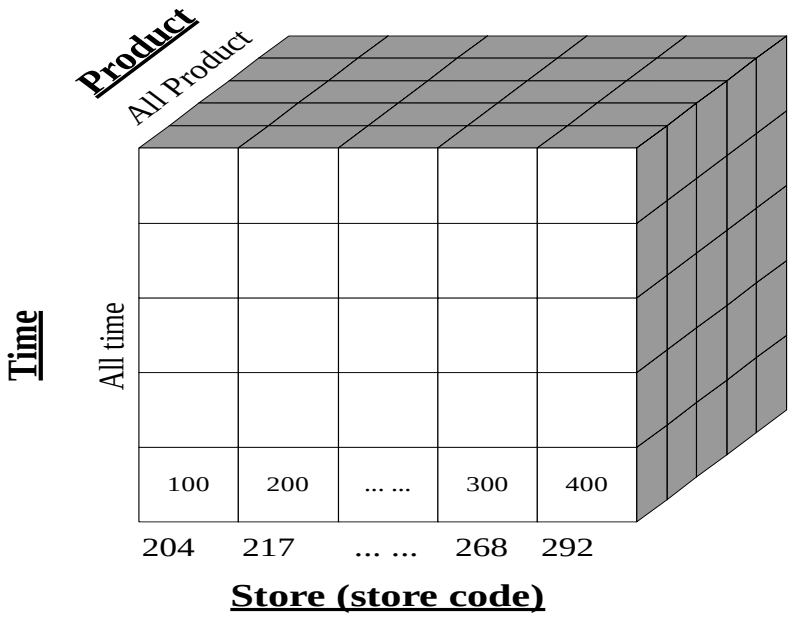
MDX

```
SELECT [SALES].[AMOUNT] ON COLUMNS,
[store].[Kowloon].[292] ON ROWS FROM SALES
```

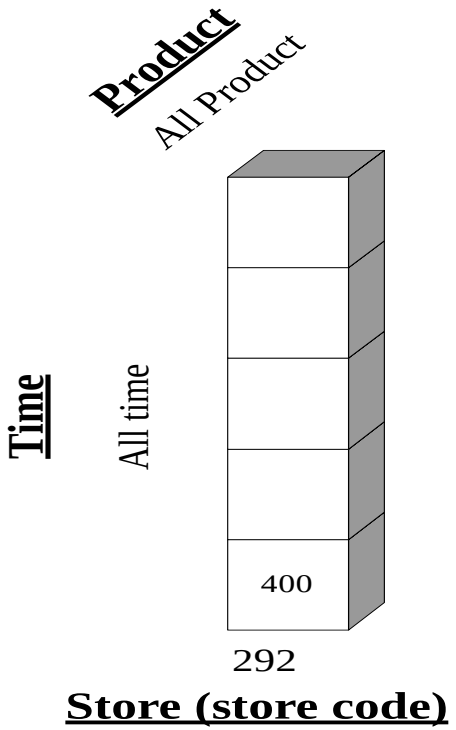
SQL

```
select sum(amount), storecode
from SALES
where  (storecode='292') group by storecode
```

Graphical Description of Slice



Slice
WHERE store = ‘292’



Example on Dice

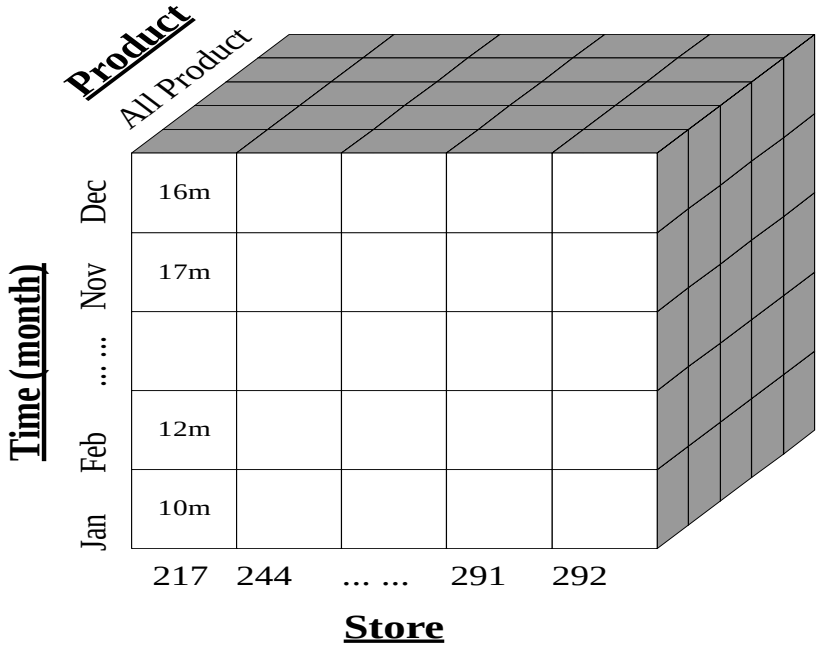
MDX

```
SELECT [SALES].[AMOUNT] ON COLUMNS,
[store].[HK],[store].[NT], [store].[Kowloon] ON ROWS
FROM SALES WHERE [time].[2003].[Q4].[Dec].[24]
```

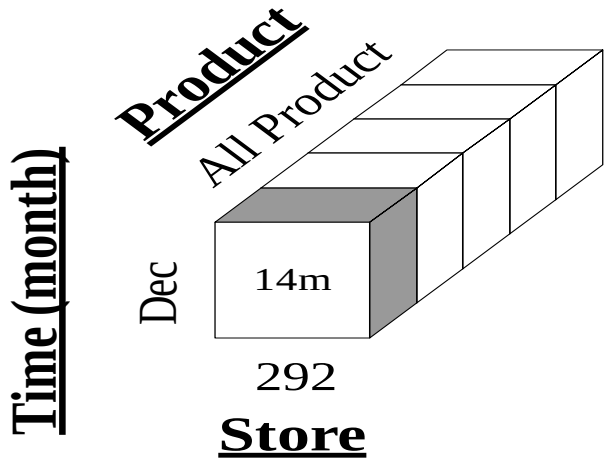
SQL

```
select sum(amount), area
from SALES
where ( (area='HK') or (area='NT') or (area='Kowloon'))
and (the_date='2003-Dec-24')
group by area
```

Graphical Description of Dice



Dice
WHERE (store = ‘292’) and (time = ‘Dec’) and product = ‘ALL’



The CrossJoin ( ) Function

The CrossJoin ( ) function is used to generate the cross-product of two input sets. If two sets exist in two independent dimensions, the CrossJoin operator creates a new set consisting of all of the combinations of the members in the two dimensions.

Filter ( ) Versus Slicer

In some respects, the Filter ( ) function and the slicer axis have similar purposes. The difference between the two is that the Filter ( ) function defines the members in a set, while slicers determine a slice of the cube returned from a query.

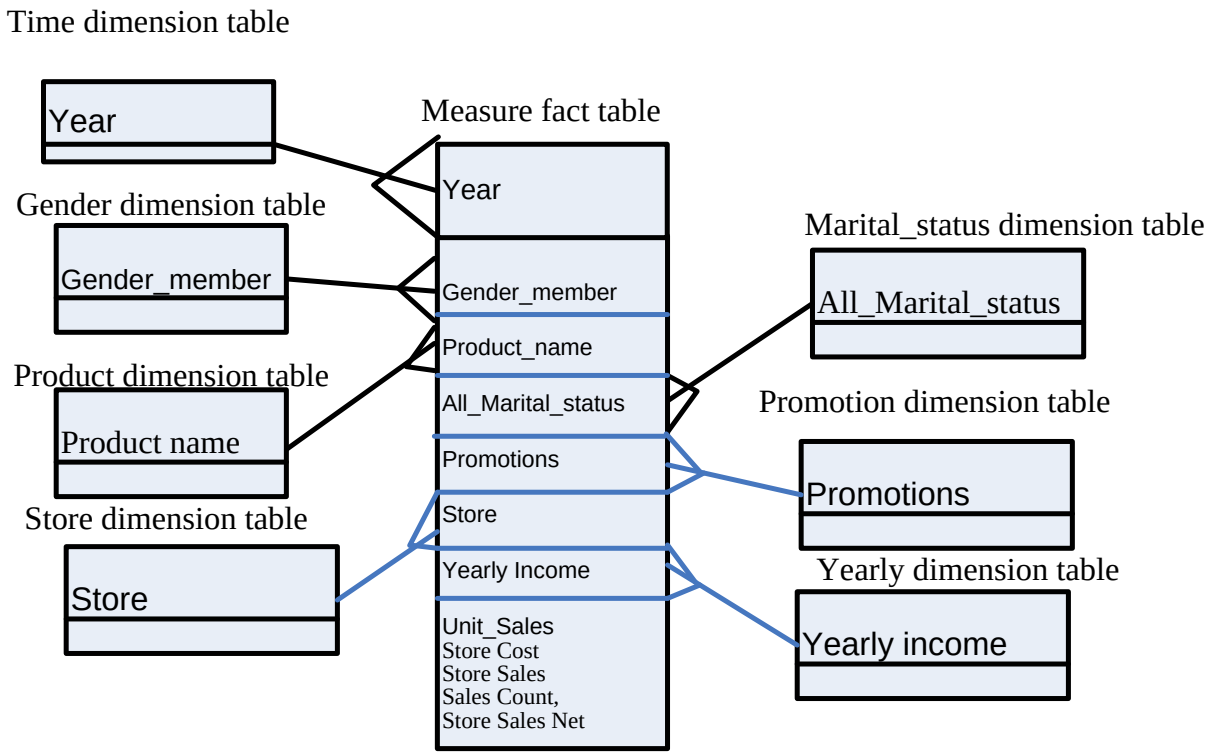
The Order ( ) Function

The Order ( ) function provides sorting capabilities within the MDX language. When the Order expression is used, it can either sort within the natural hierarchy (ASC and BDESC), or it can sort without the hierarchy (BASC and BDESC). The “B” indicates “break” hierachy.

TopCount ( ) and BottomCount ( ) Functions

The TopCount () and BottomCount() functions provide rank functionality critical in a decision support and data analysis environment. These expressions sort a set based on a numerical expression and pick the top index items based on rank order.

Sales Data Warehouse Star Schema of the SalesRecord



Sample Star Schema of Sales Record

Dimension tables:
[Gender].[Gender Members]
[Product].[Product Name]
[Marital Status].[All Martial status]
[Promotions].[All Promotions],
[Store].[All Stores],
[Store Size in SQFT].[All],
[Store Type].[All],
[Yearly Income].[All Yearly Income]
[Time].[Year]

Fact table:
[Measures].[Unit Sales],
[Measures].[Store Cost],
[Measures].[Store Sales],
[Measures].[Sales Count],
[Measures].[Store Sales Net]

Axis Dimensions in the Select Clause

```
Select
CrossJoin({[Gender]. [Gender]. Members},
{[Time].[Year]. Members}) ON COLUMNS,
{[Measures].Members} ON ROWS
FROM [Sales]
```

Where CrossJoin operator of source sets creates a new set consisting of all of the combinations of the members of the source sets.

This query crossjoins the Gender and the Time dimensions to produce data in which the data for each gender is broken into two years in this cube.

- The two specific sets that are within the CrossJoin are the two members of the gender level of the Gender dimension, and the two years in the year level of the Time dimension.
- The set of all members of the Measure dimension is included on the rows axis.
- There is no member explicitly added as a slicer in this query.

Output from the data cube of Slicer Function

	F		M	
	1997	1998	1997	1998
Unit Sales	131,558.00		135,215.00	
Store Cost	111,777.48		113,849.75	
Store Sales	280,228.21		285,011.92	
Sales count	428.31		440.06	
Store Sales Net	168,448.73		171,162.17	

Filter filers a set based on a particular condition

```
SELECT
{[Measures]. [Unit Sales]} ON COLUMNS,
{Filter({[Product]. [Product Department].Members},
([Gender]. [All Gender].[F],[Measures].[Unit Sales]) > 10000)} ON ROWS
FROM [Sales]
```

The Filter function produces a set of product departments meeting the Filter criteria

The results of this query show that the set returned on the rows axis consists of product departments for which unit sales to females is greater than $10000

Output from the data cube

	Unit Sales
Frozen Food	26,655.00
Produce	37,792.00
Snack Foods	30,545.00
Household	27,038.00

TopCount( ) and BottomCount( ) Functions

```
SELECT
{[Customers].[All Customers].[USA],
[Customers].[All Customers].[USA].Children}ON COLUMNS,
{TopCount({[Product].[Product Category].Members},
5, [Measures].[Unit Sales]),
BottomCount({[Product].[Product Category].Members},
5, [Measures].[Unit Sales])} ON ROWS
FROM [Sales]
```

Where TopCount is to request the highest count of the data as a result of the query. Similarly, BottomCount is to request the lowest count of the data as a res
of the query.

The columns axis contains the members from the customers dimension. The single member, {[Customers].[All Customers].[USA] is specified and the children of USA, [Customers].[All Customers].[USA]. Children, are combined in a comma- separated list to make up the set.

The product categories are included on the rows axis in a comma-separated list where different operators are used to specify a particular subset of the [Product].[Product Category]. Members set. Unit sales is used as the measure with which to select the top five product categories and the bottom five product categories.

Output from the data cube

	USA	CA	OR	WA
Snack Foods	30,545.00	8,543.00	7,789.00	14,213.00
Vegetables	20,733.00	5,506.00	5,447.00	9,306.00
Dairy	12,885.00	3,534.00	3,131.00	6,220.00
Jams & Jelies	11,888.00	3,343.00	2,877.00	5,868.00
Fruit	11,767.00	3,184.00	3,008.00	5,575.00
Canned Oysters	708.00	220.00	182.00	296.00
Canned Shimp	804.00	231.00	173.00	400.00
Hardware	810.00	281.00	215.00	334.00
Candies	815.00	286.00	248.00	303.00
Canned Food	819.00	234.00	215.00	370.00

The Order () Function

```
Select
{[Marital Status].[All Marital Status].[S]} ON COLUMNS,
{Order ({[Promotion Media].[Media Type].Members},
[Unit Sales], BDESC)} ON ROWS
FROM [Sales]
```

Where BDESC means sort descending without hierarchy.

The Order function provides sorting capabilities within the MDX language in ASC, DESC, BASC and BDESC where “B” indicates “break” hierarchy.

The sort was performed using the [Marital Status] .[All Marital Status].[S] member in descending order without hierarchy of the unit sales.

Output from the data cube

Marital Status	**S**
No Media	95.970.00
Daily Paper, Radio, TV	4,787.00
Daily Paper	3,559.00
Product Attachment	3,352.00
Daily Paper, Radio	3,572.00
Cash Register Handout	3,567.00
Sunday Paper, Radio	3,285.00
Street Handout	2,921.00
Sunday Paper	2,098.00
Bulk Mail	2,271.00
In Store Coupon	1,829.00
TV	1,873.00
Sunday Paper, Radio, TV	1,378.00
Radio	1,298.00

Filter Function

```
Select
{[Gender], Members} ON COLUMNS,
{TopCount ({[Product].[Product Name].Members},10,
([Gender].[Gender].[F], [Measures].[Unit Sales]))} ON ROWS
FROM [Sales]
WHERE ([Marital Status].[All Marital Status].[M],
[Measures].[Unit Sales])
```

This query is motivated by a desire to determine which products married women are most likely to purchase and the sales of these same products to married men.

- The columns axis contains all members of the gender dimension, [All Gender], [F]. and [M]. [All Gender] is included because the .Members function was placed on the gender dimension instead of on the [Gender].[Gender] level.
- The fundamental set in the rows axis consists of names of products (members of the [Product].[Product Name] level). In this query the TopCount ( ) function is used to examine some of the products. Of specific interest here are the top 10 products in unit sales pruchased by females. Therefore, the index in the TopCount ( ) function is 10, and the numeric expression is the tuple ([Gender].[Gender].[F], [Measures].[Unit Sales]).

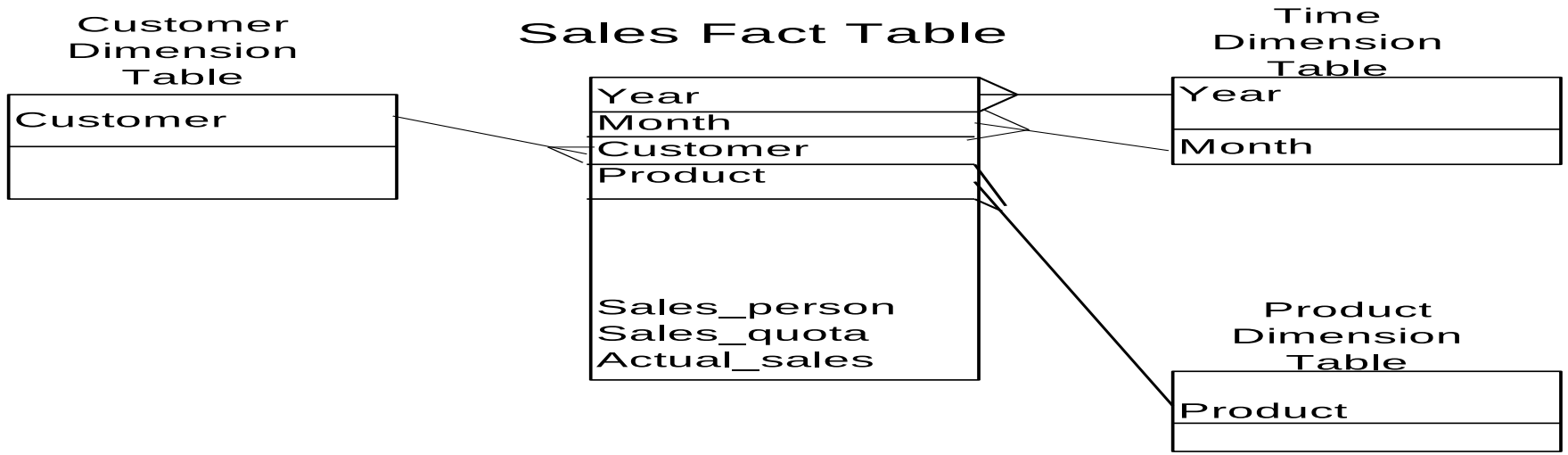
- The slicer contains the two members explicitly defined, [Marital Status].[All Marital Status].[M] and [Measures].[Unit Sales], because only data with these characteristics is desired.

Output from the data cube

	All Gender	F	M
Fabulous Berry Juice	127.00	87.00	40.00
Fast Beef Jerky	134.00	87.00	47.00
BBB Best Pepper	134.00	82.00	52.00
Ebony Cantelope	130.00	80.00	50.00
Peart Cheable Wine	117.00	79.00	38.00
Skinner Diel Cola	115.00	78.00	37.00
Shdy Lake Manicotti	106.00	78.00	28.00
Pearl Light Beer	130.00	77.00	53.00
Shady Lake Rice Medly	131.00	77.00	54.00
TriState Potatos	108.00	76.00	32.00

Example of OLAP

Starting with the base cuboid [Year, Month, Customer, Product, Sales-person, Sales-quota, Actual-sales], what specific OLAP operation should be performed in order to list the total Actual Sales by Customer in year 2000?



The requested SQL statement is:
Select Customer, Sum(Actual_sales) From Sales Where year = '2000' Group by customer

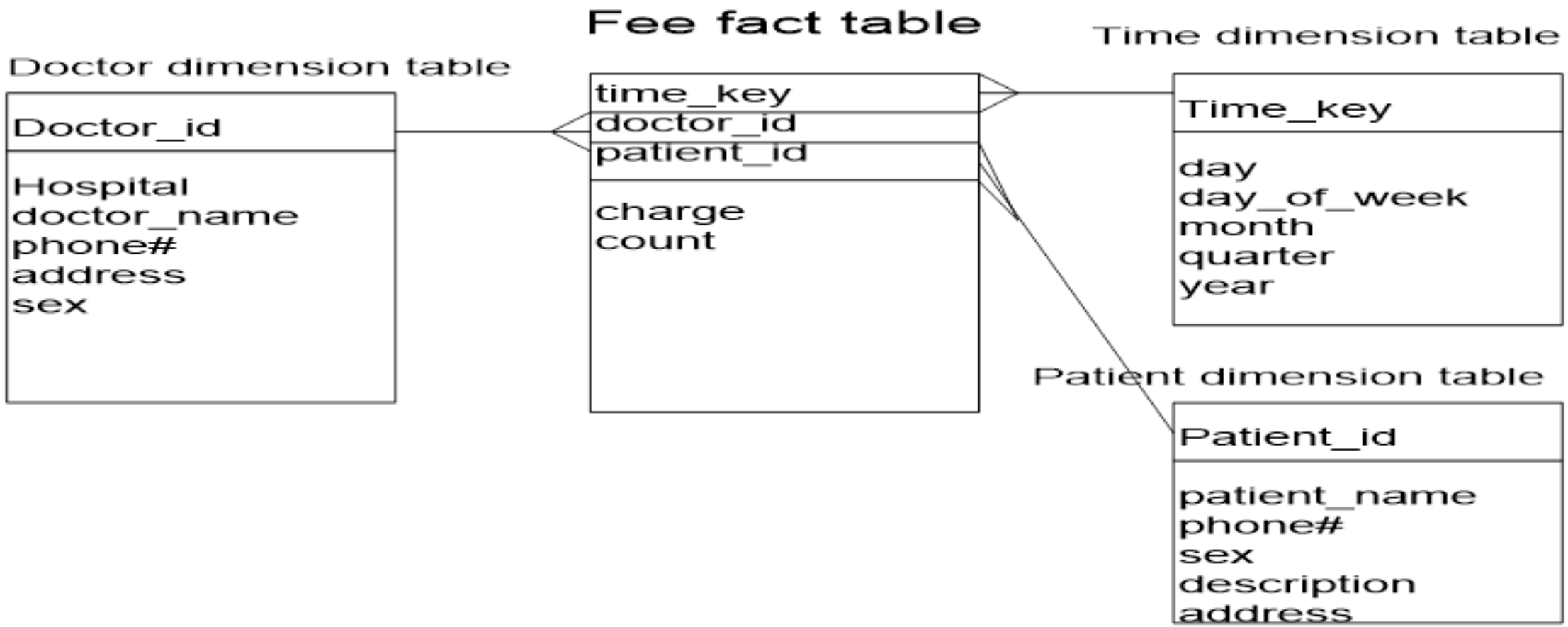
The requested MDX statement is:
Select{[Sales].Actual_sales}on Columns
({[Customer].Customer_name},{[Time].Year}) on Rows
from Cuboid
Where ([Time].[Year].[2000])

Tutorial Question 5

Suppose that a data warehouse consists of the three dimensions *time, doctor,* and *patent,* and the two measures *count* and *charge*, where charge is the fee that a doctor charges a patient for a visit. Starting with the base *cuboid* [*day, doctor, patient*], provide a MDX (Multidimensional Expression) query to list the total fee collected by each doctor in 2000?

To obtain the same list, write an SQL query assuming the data is stored in a relational database with the table *fee (day, month, year, doctor, hospital, patient, count, charge).*

1. *Starting with the base cuboid [day, doctor, patient], provide a MDX (Multidimensional Expression) query to list the total fee collected by each doctor in 2000?*
2. *To obtain the same list, write an SQL query assuming the data is stored in a relational database with the table fee (day, month, year, doctor, hospital, patient, count, charge).*


Fee fact table
Doctor dimension table
Doctor_id
Hospital
doctor_name
phone#
address
sex
time_key
doctor_id
patient_id
charge
count
Time dimension table
Time_key
day
day_of_week
month
quarter
year
Patient dimension table
Patient_id
patient_name
phone#
sex
description
address

Data Warehouse Design and Usage

Data Warehouse Usage

Introduction

- Data warehouses and data marts are used in a wide range of applications.
- Business executives use the data in data warehouses and data marts to perform data analysis and make strategic decisions.
- Data warehouses are used extensively in banking and financial services, consumer goods and retail distribution sectors, and controlled manufacturing such as demand-based production.

- Initially, the data warehouse is mainly used for generating reports and answering predefined queries.
- Progressively, it is used to analyze summarized and detailed data, where the results are presented in the form of reports and charts.
- Later, the data warehouse is used for strategic purposes, performing multidimensional analysis and sophisticated slice-and-dice operations.
- Finally, the data warehouse may be employed for knowledge discovery and strategic decision making using data mining tools.

Data Warehouse Applications

There are three kinds of data warehouse applications:

I. **information processing,**
II. **analytical processing, and**
III. **data mining.**

Data Warehouse Applications

- **Information processing**
  - supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs
- **Analytical processing**
  - It generally operates on historic data in both summarized and detailed forms.
  - supports basic OLAP operations, slice-dice, drilling, pivoting.
  - The major strength of online analytical processing over information processing is the multidimensional data analysis of data warehouse data.
- **Data mining**
  - knowledge discovery from hidden patterns
  - supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools

“How does data mining relate to information processing and online analytical processing? ”

- ✓ Information processing, based on queries, can find useful information. However, answers to such queries reflect the information directly stored in databases or computable by aggregate functions.
- ✓ It do not reflect hidden patterns or regularities buried in the database.
- ✓ Therefore, information processing is not data mining.

“Do OLAP systems perform data mining? Are OLAP systems actually data mining systems?”

- The functionalities of OLAP and data mining can be viewed as disjoint.
- OLAP is a data summarization/aggregation tool that helps simplify data analysis.
- Data mining allows the automated discovery of implicit patterns and interesting knowledge hidden in large amounts of data.
- OLAP functions are essentially for user-directed data summarization and comparison.
- Data mining covers a much broader spectrum than simple OLAP operations, because it performs not only data summarization and comparison but also association, classification, prediction, clustering, time-series analysis, and other data analysis tasks.

- Data mining is not confined to the analysis of data stored in data warehouses.
- It may also analyze transactional, spatial, textual, and multimedia data that are difficult to model with current multidimensional database technology.

From Online Analytical Processing to Multidimensional Data Mining

- Multidimensional data mining (also known as exploratory multidimensional data mining, online analytical mining, or OLAM) integrates OLAP with data mining to uncover knowledge in multidimensional databases.

- Among the many different paradigms and architectures of data mining systems, multidimensional data mining is particularly important for the following reasons:

From Online Analytical Processing to Multidimensional Data Mining

High quality of data in data warehouses:

Most data mining tools need to work on integrated, consistent, and cleaned data, which requires costly data cleaning, data integration, and data transformation as preprocessing steps.

A data warehouse constructed by such preprocessing serves as a valuable source of high-quality data for OLAP as well as for data mining.

From Online Analytical Processing to Multidimensional Data Mining

Available information processing infrastructure surrounding data warehouses:

Comprehensive information processing and data analysis infrastructures have been or will be systematically constructed surrounding data warehouses, which include accessing, integration, consolidation, and transformation of multiple heterogeneous databases, ODBC/OLEDB connections, Web accessing and service facilities, and reporting and OLAP analysis tools.

It is prudent to make the best use of the available infrastructures rather than constructing everything from scratch.

From Online Analytical Processing to Multidimensional Data Mining

OLAP-based exploration of multidimensional data:

Effective data mining needs exploratory data analysis. A user will often want to traverse through a database, select portions of relevant data, analyze them at different granularities, and present knowledge/results in different forms.

Online selection of data mining functions:

Users may not always know the specific kinds of knowledge they want to mine. By integrating OLAP with various data mining functions, multidimensional data mining provides users with the flexibility to select desired data mining functions and swap data mining tasks dynamically.

Data Warehouse Implementation

Data Warehouse Implementation

- Data warehouses contain huge volumes of data.
- OLAP servers demand that decision support queries be answered in the order of seconds.
- Therefore, it is crucial for data warehouse systems to support highly efficient cube computation techniques, access methods, and query processing techniques.

Efficient Data Cube Computation

- One approach to cube computation extends SQL so as to include a **compute cube** operator.
- The compute cube operator computes aggregates over all subsets of the dimensions specified in the operation.
- This can require excessive storage space, especially for large numbers of dimensions.

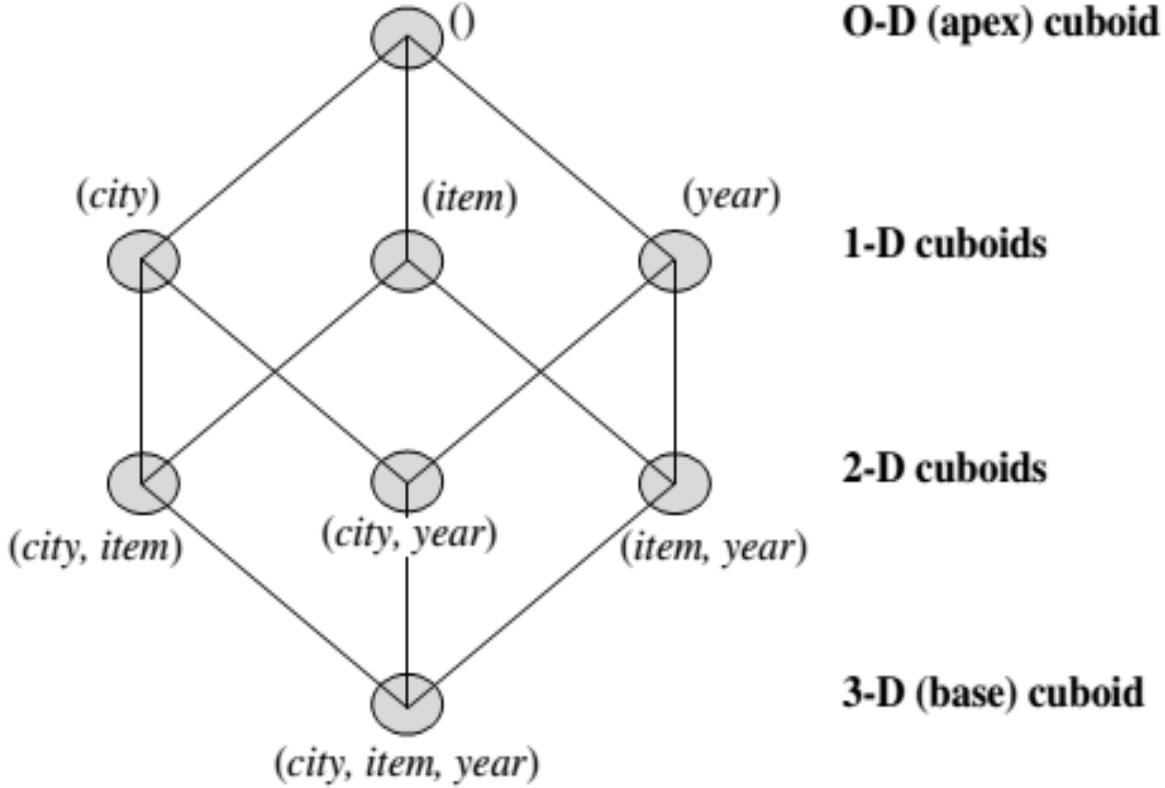
EXAMPLE

- Suppose that you want to create a data cube for AllElectronics sales that contains the following: city, item, year, and sales_in_dollars.
- Taking the three attributes, city, item, and year, as the dimensions for the data cube, and sales_in_dollars as the measure.
- The total number of cuboids, that can be computed for this data cube is $2^3 = 8$

{(city, item, year), (city, item), (city, year), (item, year), (city), (item), (year), () }

EXAMPLE

If we start at the **apex cuboid** and explore downward in the lattice, this is equivalent to **drilling down** within the data cube. If we start at the **base cuboid** and explore upward, this is akin to **rolling up**.



- An SQL query containing no group-by (e.g., **“compute the sum of total sales”**) is a zero dimensional operation.
- An SQL query containing one group-by (e.g., **“compute the sum of sales, group-by city ”**) is a one-dimensional operation.
- A cube operator on n dimensions is equivalent to a collection of group-by statements, one for each subset of the n dimensions.

- Similar to the SQL syntax, the data cube can be defined as:

define cube sales_cube [city, item, year]: sum(sales_in_dollars)

- A statement such as

compute cube sales_cube

would explicitly instruct the system to compute the sales aggregate cuboids for all eight subsets of the set {city, item, year }, including the empty subset.

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
```

“How many cuboids are there in an n-dimensional data cube ???”

- If there were no hierarchies associated with each dimension, then the total number of cuboids for an n-dimensional data cube, as we have seen, is 2^n.
- However, in practice, many dimensions do have hierarchies.
- For example, time “day ≤ month ≤ quarter ≤ year.”

“How many cuboids are there in an n-dimensional data cube ???”

$$Total\ number\ of\ cuboids = \prod_{i=1}^{n}(L_i + 1),$$

$$5^{10} \approx 9.8 \times 10^6$$

- By now, you probably realize that it is unrealistic to pre-compute and materialize all of the cuboids that can possibly be generated for a data cube.
- If there are many cuboids, and these cuboids are large in size.
- A more reasonable option is partial materialization; that is, to materialize only some of the possible cuboids that can be generated.

Partial Materialization: Selected Computation of Cuboids

- There are three choices for data cube materialization given a base cuboid:
- **No materialization**: Do not pre-compute any of the "non-base" cuboids. This leads to computing expensive multidimensional aggregates on-the-fly, which can be extremely slow.
- **Full materialization:** Pre-compute all of the cuboids. It is referred to as the full cube. It requires huge amounts of memory space in order to store all of the pre-computed cuboids.

- **Partial materialization**:

We may compute a subset of the cube, which contains only those cells that satisfy some user-specified criterion. It is also referred to as **sub-cube**.

Indexing OLAP Data:

Bitmap Index and Join Index

- To facilitate efficient data accessing, most data warehouse systems support index structures and materialized views (using cuboids).
- Bitmap indexes have traditionally been considered to work well for data **such as gender**, which has a small number of distinct values, for example male and female, but many occurrences of those values.

- The bitmap indexing method is popular in OLAP because it allows quick searching in data cubes.
- If a given attribute's domain consists of n values, then n bits are needed for each entry in the bitmap index (i.e., there are n bit vectors).
- If the attribute has the **value v** for a given row in the data table, then the bit representing that value is **set to 1** in the corresponding row of the bitmap index. All other bits for that row are **set to 0**.

Base table

RID	*item*	*city*
R1	H	V
R2	C	V
R3	P	V
R4	S	V
R5	H	T
R6	C	T
R7	P	T
R8	S	T

item bitmap index table

RID	H	C	P	S
R1	1	0	0	0
R2	0	1	0	0
R3	0	0	1	0
R4	0	0	0	1
R5	1	0	0	0
R6	0	1	0	0
R7	0	0	1	0
R8	0	0	0	1

city bitmap index table

RID	V	T
R1	1	0
R2	1	0
R3	1	0
R4	1	0
R5	0	1
R6	0	1
R7	0	1
R8	0	1

Note: H for "home entertainment," C for "computer," P for "phone," S for "security," V for "Vancouver," T for "Toronto."

- Bitmap indexing is advantageous compared to hash and tree indices.
- It is especially useful for low-cardinality domains.
- Bitmap indexing leads to significant reductions in space and input/output (I/O) since a string of characters can be represented by a single bit.
- For higher-cardinality domains, the method can be adapted using compression techniques.

- Traditional indexing maps the value in a given column to a list of rows having that value.
- Join indexing registers the joinable rows of two relations from a relational database.

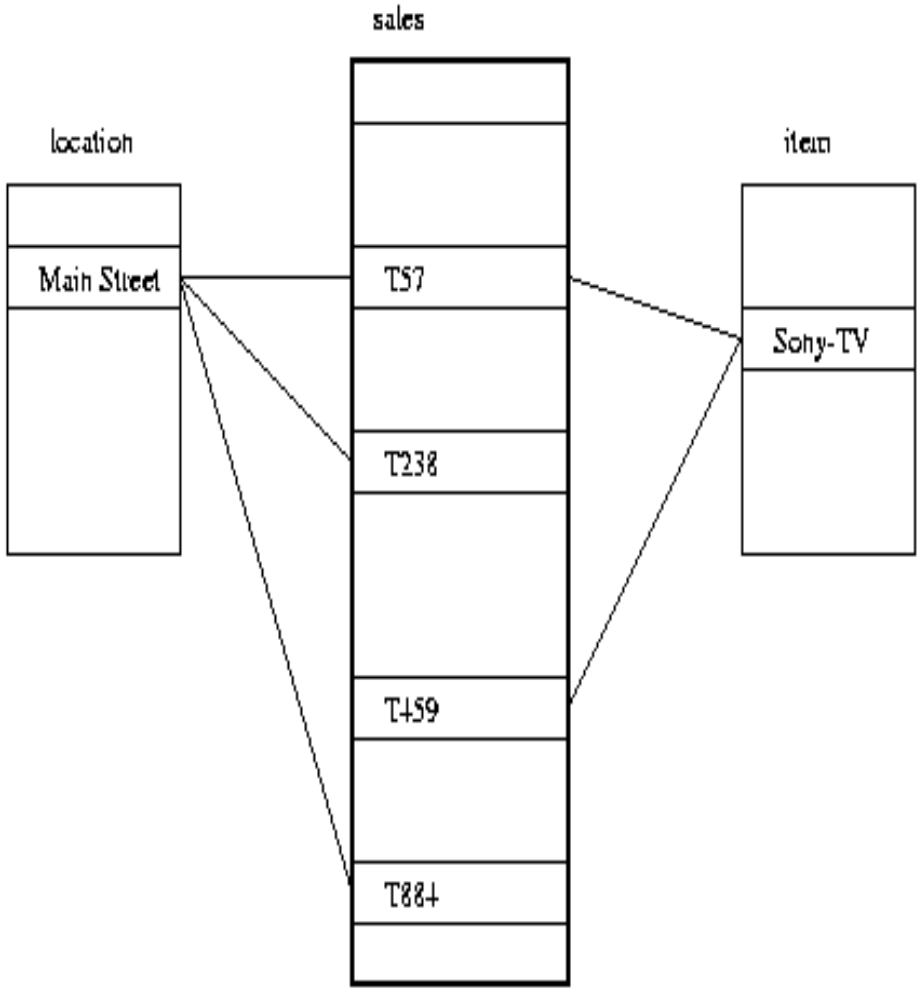
Join index: JI(R-id, S-id) where R (R-id, …) ⊳⊲ S (S-id, …)

- Join indexing is especially useful for maintaining the relationship between a foreign key and its matching primary keys, from the joinable relation.

- 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.*

- Join indices can span multiple dimensions.



Join index table for *location/sales*

location	*sales_key*
. . .	. . .
Main Street	T57
Main Street	T238
Main Street	T884
. . .	. . .

Join index table for *item/sales*

item	*sales_key*
. . .	. . .
Sony-TV	T57
Sony-TV	T459
. . .	. . .

Join index table linking *location* and *item* to *sales*

location	*item*	*sales_key*
. . .	. . .	. . .
Main Street	Sony-TV	T57
. . .	. . .	. . .

- The purpose of materializing cuboids and constructing OLAP index structures is to speed up query processing in data cubes. Given materialized views, query processing should proceed as follows:

1) Determine which operations should be performed on the available cuboids.
2) Determine to which materialized cuboid(s) the relevant operatiozns should be applied.

Suppose that we define a data cube for AllElectronics of the form

“sales cube [time, item, location]: sum(sales in dollars).”

The dimension hierarchies used are

“day < month < quarter < year ” for time ;

“item name < brand < type ” for item;

“street < city < province or state < country ” for location.

cuboid 1: {year, item_name, city}

cuboid 2: {year, brand, country }

cuboid 3: {year, brand, province_or_state}

cuboid 4: {item_name, province_or_state }, where year = 2010

“Which of these four cuboids should be selected to process the query?”

- Finer-granularity data cannot be generated from coarser-granularity data.
- Therefore, cuboid 2 cannot be used because country is a more general concept than province or state.
- Cuboids 1, 3, and 4 can be used to process the query

“How would the costs of each cuboid compare if used to process the query?”