Lecture 1: Introduction to Business Intelligent Analytics

What are Data Analytics?

- Analytic is the use of:
  - Data,
  - Information technology,
  - Statistical analysis,
  - Quantitative methods,
  - Mathematical or computer-based models.
- To help managers gain improved insight about their business operations and make better, fact-based decisions.
- Business Analytics (BA) is a subset of Data Analytics.

Business Analytics applications:

- Management of customer relationships.
- Financial and marketing activities.
- Supply chain management.
- Human resource planning.
- Pricing decisions.
- Sport team game strategies.

Importance of Business Analytics:

- There is a strong of relationship of BA with:
  - Profitability of businesses.
  - Revenue of businesses.
  - Shareholder return.
- BA enhances understanding of data.
- BA is vital for businesses to remain competitive.
- BA enables creation of informative reports.

Scope of Business Analytics

- Descriptive analytics:
  - Uses data to understand past and present.
- Predictive analytics:
  - Analyzes past performance.
- Prescriptive analytics:
  - Uses optimization techniques.

Scope of Business Analytics - Retail Markdown Decisions

- Most department stores clear seasonal inventory by reducing prices.
- The question is:
  - When to reduce the price and how much?

- Descriptive analytics: examine historical data for similar products (prices, unit sold, advertising, etc.)
- Predictive analytics: predict sales based on price.
- Prescriptive analytics: find the best sets of pricing and advertising to maximize sales revenues.

Data for Business Analytics

- DATA:
  - Collected facts and figures.
- DATABASE:
  - Collection of computer files containing data.
- INFORMATION:
  - Comes from analyzing data.

Data for Business Analytics

- Metrics are used to quantify performance.
- Measures are numerical values of metrics.
- Discrete metrics involve counting.
  - On time or not on time.
  - Number or proportion of on time deliveries.
- Continuous metrics are measured on a continuum.
  - Delivery time.
  - Package weight.
  - Purchase price.

Data for Business Analytics

- A Sales Transaction Database File:

	A	B	C	D	E	F	G	H
1	**Sales Transactions: July 14**							
2								
3	**Cust ID**	**Region**	**Payment**	**Transaction Code**	**Source**	**Amount**	**Product**	**Time Of Day**
4	10001	East	Paypal	93816545	Web	$20.19	DVD	22:19
5	10002	West	Credit	74083490	Web	$17.85	DVD	13:27
6	10003	North	Credit	64942368	Web	$23.98	DVD	14:27
7	10004	West	Paypal	70560957	Email	$23.51	Book	15:38
8	10005	South	Credit	35208817	Web	$15.33	Book	15:21
9	10006	West	Paypal	20978903	Email	$17.30	DVD	13:11
10	10007	East	Credit	80103311	Web	$177.72	Book	21:59
11	10008	West	Credit	14132683	Web	$21.76	Book	4:04
12	10009	West	Paypal	40128225	Web	$15.92	DVD	19:35
13	10010	South	Paypal	49073721	Web	$23.39	DVD	13:26

Records

Entities

Fields or Attributes

Figure 1.1

What is Big Data?

- Information from multiple internal and external sources:
  - Transactions.
  - Social media.
  - Enterprise content.

    - Sensors.
    - Mobile devices.
- Companies leverage data to adapt products and services to:
    - Meet customer needs.
    - Optimize operations.
    - Optimize infrastructure.
    - Find new resource of revenue.
    - Can reveal more patterns and anomalies.
- IBM estimates that by 2015 4.4 million jobs will be created globally to support Big Data.
    - 1.9 million of these jobs will be in United States.

Types of Data

- When collecting or gathering data we collect data from individuals' cases on variables.
- A variable is a unit of data collection whose value can vary.
- Variables can be defined into types according to the level of mathematical scaling that can carried out on the data.
- There are four types of data or levels of measurement:

1. Categorical (Nominal)	**2. Ordinal**
3. Interval	**4. Ratio**

Data for Business Analytics

- Classifying Data Elements in a Purchasing Database

	A	B	C	D	E	F	G	H	I	J
1	**Purchase Orders**									
2										
3	**Supplier**	**Order No**	**Item No.**	**Item Description**	**Item Cost**	**Quantity**	**Cost per order**	**A/P Terms (Month**	**Order Date**	**Arrival Date**
4	Spacetime Technologies	A0111	6489	O-Ring	$ 3.00	900	$ 2,700.00	25	10/10/11	10/18/11
5	Steelpin Inc.	A0115	5319	Shielded Cable/ft.	$ 1.10	17,500	$ 19,250.00	30	08/20/11	08/31/11
6	Steelpin Inc.	A0123	4312	Bolt-nut package	$ 3.75	4,250	$ 15,937.50	30	08/25/11	09/01/11
7	Steelpin Inc.	A0204	5319	Shielded Cable/ft.	$ 1.10	16,500	$ 18,150.00	30	09/15/11	10/05/11
8	Steelpin Inc.	A0205	5677	Side Panel	$195.00	120	$ 23,400.00	30	11/02/11	11/13/11
9	Steelpin Inc.	A0207	4312	Bolt-nut package	$ 3.75	4,200	$ 15,750.00	30	09/01/11	09/10/11
10	Alum Sheeting	A0223	4224	Bolt-nut package	$ 3.95	4,500	$ 17,775.00	30	10/15/11	10/20/11
11	Alum Sheeting	A0433	5417	Control Panel	$255.00	500	$ 127,500.00	30	10/20/11	10/27/11
12	Alum Sheeting	A0443	1243	Airframe fasteners	$ 4.25	10,000	$ 42,500.00	30	08/08/11	08/14/11
13	Alum Sheeting	A0446	5417	Control Panel	$255.00	406	$ 103,530.00	30	09/01/11	09/10/11
14	Spacetime Technologies	A0533	9752	Gasket	$ 4.05	1,500	$ 6,075.00	25	09/20/11	09/25/11
15	Spacetime Technologies	A0555	6489	O-Ring	$ 3.00	1,100	$ 3,300.00	25	10/05/11	10/10/11

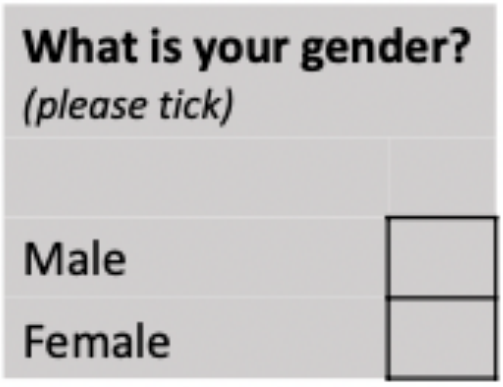
Figure 1.2

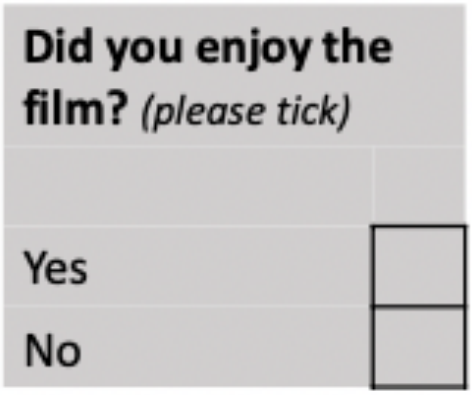
Categorical Categorical Categorical Categorical Ratio Ratio Ratio Ratio Interval Interval

Categorical (Nominal) data

- Nominal or categorical data is data that comprises of categories that cannot be rank ordered – each category is just different.
- The categories available cannot be placed in any order and no judgement can be made about the relative size or distance from one category to another.
    - Categories bear no quantitative relationship to one another.

    - Examples:
        - Customer's location (American, Europe, Asia).
        - Employee classification (manager, supervisor, associate).
- What does this mean? No mathematical operations can be performed on the data relative to each other.
- Therefore, nominal data reflect qualitative differences rather than quantitative.





- Systems for measuring nominal data must ensure that each category is mutually exclusive, and the system of measurement needs to be exhaustive.
- Variables that have only 2 responses, i.e., Yes or No, are known as dichotomies.

Ordinal data

- Ordinal data is data that comprises of categories that can be ranked ordered.
- Similarly with nominal data, the distance between each category cannot be calculated but the categories can be ranked above or below each other.
    - No fixed units of measurement.
    - Examples:
        - College football rankings.
        - Survey responses (poor, average, good, very good, excellent).
- What does this mean? Can make statical judgements and perform limited math.

How satisfied are you with the level of service you have received? *(please tick)*	
Very satisfied	
Somewhat satisfied	
Neutral	
Somewhat dissatisfied	
Very dissatisfied	

Interval and ratio data

- Both interval and ratio are examples of scale data.
- Scale data:
  - Data is in numeric format ($50, $100, $200).
  - Data that can be measured on a continuous scale.
  - The distance between each can be observed and as a result measured.
  - The data can be placed in rank order.

Interval data

- Ordinal data but with constant differences between observations.
- Ratios are not meaningful.
- Examples:
  - Time – moves along a continuous measure of seconds, minutes, and so on, and is without a zero point of time.
  - Temperature – moves along a continuous measure of degrees and is without a true zero.
  - SAT scores.

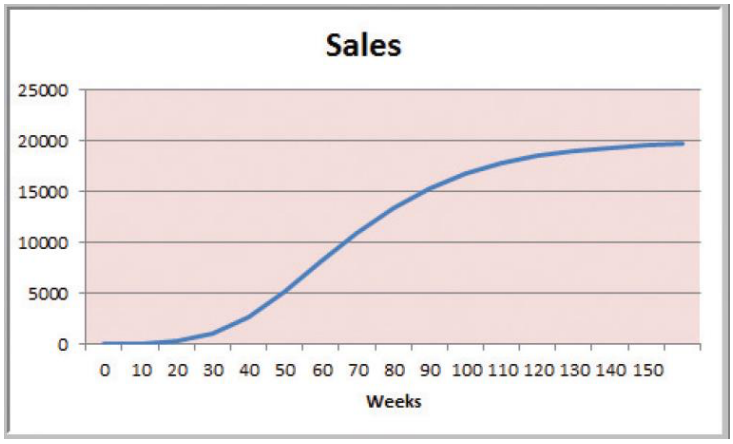
Ratio data

- Ratio data measured on a continuous scale and does have a natural zero point.
  - Ratios are meaningful.
  - Examples:
    - Monthly sales.
    - Delivery times.
    - Weight.
    - Height.

        - Age.

Types of analytics – Decision Models

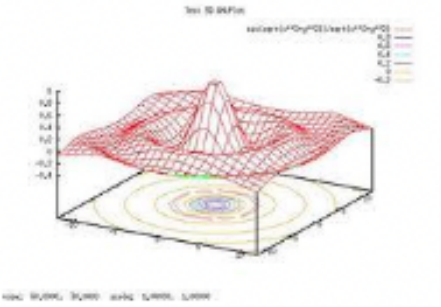
- Model:
    - An abstraction or representation of a real system, idea, or object.
    - Captures the most important features.
    - Can be:
        - written or verbal description,
        - a visual display,
        - a mathematical formula,
        - a spreadsheet representation.
- Decision Models:



    - Is a model used to understand, analyze, or facilitate decision making.
    - Types of model input:
        - Data.
        - Uncontrollable variables.
        - Decision variables (controllable).
- Descriptive Decision Models:
    - Simply tell “what is” and describe relationships.
    - Do not tell managers what to do.

Descriptive Analytics

- Description analytics, such as reporting/OLAP, dashboards, and data visualization, have been widely used for some time.
- They are the core of traditional BI.

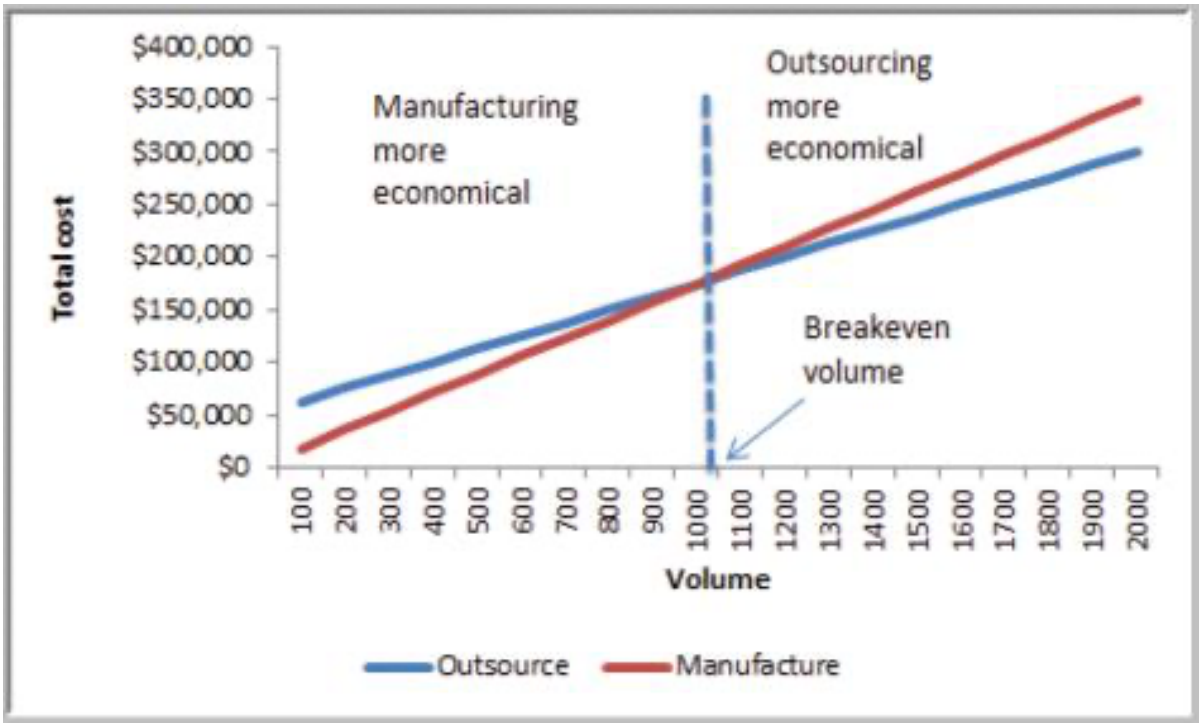


What has occurred?

Descriptive analytics, such as data visualization, is important in helping users interpret the output from predictive and predictive analytics.

Decision Models

- A Break-even Decision Model:
  - $TC(manufacturing) = \$50.000 + \$125 * Q$
  - $TC(outsourcing) = \$175 * Q$
- Break-even Point:
  - Set $TC(manufacturing) = TC(outsourcing)$.
  - Solve for Q = 1000 units.

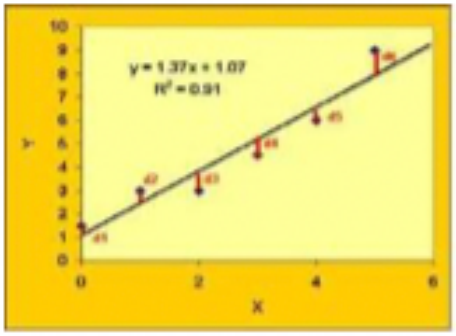


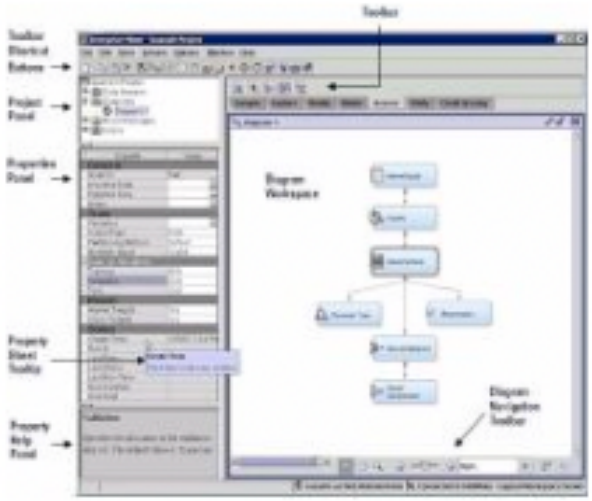
Predictive Decision Models

- Predictive Decision Models often incorporate uncertainty to help managers analyze risk.
- Aim to predict what will happen in the future.
- Uncertainty is imperfect knowledge of what will happen in the future.
- Risk is associated with the consequences of what happens.

Predictive Analytics

- Algorithms for predictive analytics, such as regression analysis, machine learning, and neural networks, have also been around for some time.
- Prescriptive analytics are often referred to as advanced analytics.





What will occur?

- Marketing is the target for many predictive analytics applications.
- Descriptive analytics, such as data visualization, is important in helping users interpret the output from predictive and prescriptive analytics.

Decision Models

- A Linear Demand Prediction Model:
    - As price increases, demand falls.

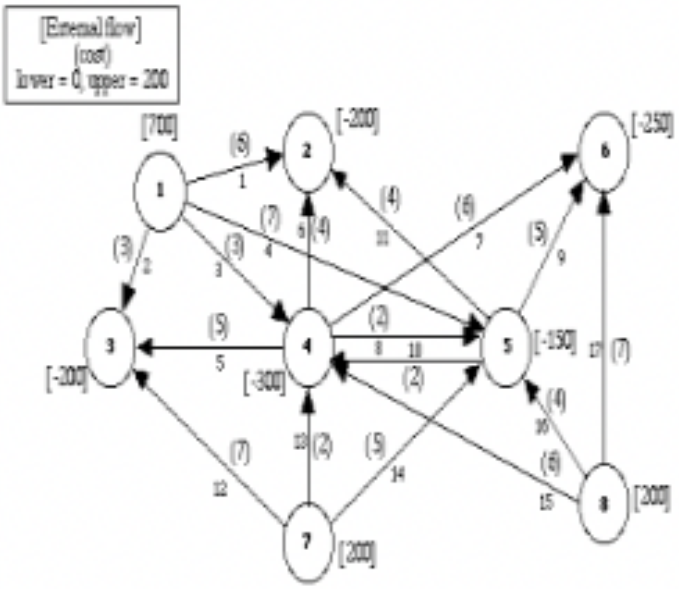
- A Non-Linear Demand Prediction Model:

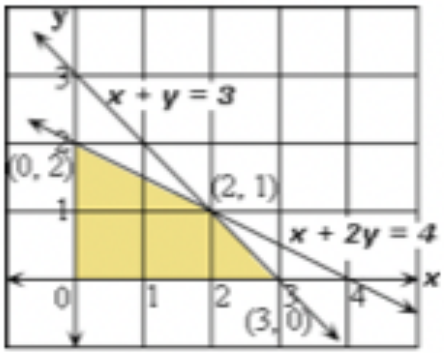
- Assumes price elasticity (constraint ratio of % change in demand to % change in price).

- Prescriptive Decision Models help decision makers identify the best solution.
  - Optimization – finding values for decision variables that minimize (or maximize) something such as cost (or profit).
  - Objective function – the equation that minimizes (or maximizes) the quantity of interest.
  - Constraints – limitation or restrictions.
  - Optimal solution – values of decision variables at the minimum (or maximum) point.

Prescriptive Analytics

- Prescriptive analytics are often referred to as advanced analytics.
- Regression analysis, machine learning and neural networks.
- Often for the allocation of scarce resources.





What should occur?

- For example, the use of mathematical programming for revenue management is common for organizations that have "perishable" good (e.g., rental cars, hotel rooms, airline seats).
- Harrah's has been using revenue management for hotel room pricing for some time.

Organizational Transformation

- Brought about by opportunity or necessity.
- The firm adopts a new business model enabled by analytics.
- Analytics are competitive requirements.

2013 Academic Research

- A 2011 TDWI report on Big Data Analytics found that 85% of respondents indicated that their firms would be using advanced analytics within 3 years.
- A 2011 IBM/MIT Sloan Management Review research study found that top performing companies in their industry are much more likely to use analytics rather than intuition across the widest range of possible decisions.

Conditions that Lead to Analytics-based Organizations

- The nature of the industry.
- Seizing an opportunity.
- Responding to a problem.

Complex Systems

- Tackle complex problems and provide individualized solutions.
- Products and services are organized around the needs of individual customers.
- Dollar value of interactions with each customer is high.
- There is considerable interaction with each customer.
- Examples: IBM, World Bank, Halliburton.

Volume Operations

- Serves high-volume markets through standardized products and services.
- Each customer interaction has a low dollar value.
- Customer interactions are generally conducted through technology rather than person-to-person.
- Are likely to be analytics-based.

- Examples: Amazon, eBay, Hertz.

The Nature of the Industry: Online Retailers

- BI Applications
    - Analysis of clickstream data.
    - Customer profitability analysis.
    - Customer segmentation analysis.
    - Product recommendations.
    - Campaign management.
    - Pricing.
    - Forecasting.
    - Dashboards.

The Nature of the Industry

- Online retailers like Amazon.com and Overstock.com are high volume operations who rely on analytics to complete.
- When you enter their sites, a cookie is placed on your PC and all clicks are recorded.
- Based on your blocks and any search terms, recommendation engines decide what products to display.
- After you purchase an item, they have additional information that is used in marketing campaigns.
- Customer segmentation analysis is used in deciding what promotions to send you.
- How profitable you are influencing how the customer care center treats you.
- A pricing team helps set prices and decides what prices are needed to clear out merchandise.
- Forecasting models are used to decide how many items to order for inventory.
- Dashboards monitor all aspects of organizational performance.

Analytics Help the Cincinnati Zoo Know Its Customers

- What management, organization, and technology factors were behind the Cincinnati Zoo losing opportunities to increase revenue?
- Why was replacing legacy point-of-sale systems and implementing a data warehouse essential to an information system solution?
- How did the Zoo benefit from business intelligence? How did it enhance operational performance and decision making? What role was played by predictive analytics?
- Visit the IBM Cognos Web site and describe the business intelligence tools that would be the most useful for the Zoo.

Table of Contents

Lecture 2: OLAP, Data Warehouse and Column Store

Why we still study OLAP/Data Warehouse in BI?

- Understand the Big Data history.
  - How does the requirement of (big) data analytics/business intelligence evolve over the time?
  - What are the architecture and implementation techniques being developed? Will they still be useful in Big Data?
  - Understand their limitation and what factors have changed from 90's to now?
- NoSQL is not only SQL.
- Hive/Impala aims to provide OLAP/BI for Big Data using Hadoop.

Highlights

- OLAP
  - Multi-relational Data model.
  - Operators.
  - SQL.
- Data warehouse (architecture, issues, optimizations).
- Join Processing.
- Column Stores (Optimized for OLAP workload).

Let's get back to the root in 70's: Relational Database

Basic Structure

- Formally, given sets D_1, D_2, …, D_n. A relation r is a subset of $D1 \ x \ D2 \ x \ \ldots \ x \ Dn$.
- Thus, a relation is a set of n-tuples $(a_1, a_2, \ldots, a_n)$ where each a_i belong to D_i.
- Example:

```
customer_name = {Jones, Smith, Curry, Lindsay}
customer_street = {Main, North, Park}
customer_city = {Harrison, Rye, Pittsfield}
```

- Then:

```
r = {
    (Jones, Main, Harrison),
    (Smith, North, Rye),
    (Curry, North, Rye),
    (Lindsay, Park, Pittsfield)
}
```

- Is a relation over:

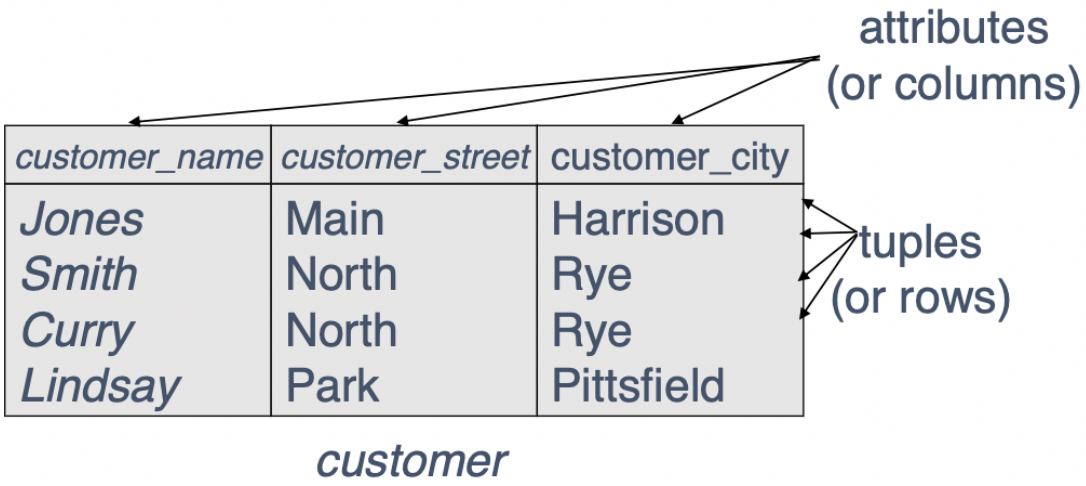
```
customer_name, customer_street, customer_city
```

Relation Schema

- A_1, A_2, …, A_n are attributes.
- $R = (A_1, A_2, …, A_n)$ is a relation schema.
  Example:
  - customer_schema = (customer_name, customer_street, customer_city)
- r(R) is a relation on the relation schema R.
  Example:
  - customer(customer_schema)

Relation Instance

- The current values (relation instance) of a relation are specified by a table.
- An element t of r is a tuple, represented by a row in a table.



Database

- A database consists of multiple relations.
- Information about an enterprise is broken up into parts, with each relation storing one part of the information.

```
account : stores information about accounts
depositor : stores information about which customer owns which account
customer : stores information about customers
```

- Storing all information as a single relation such as:

```
bank(account_number, balance, customer_name, ...)
```

  results in repetition of information (e.g., two customers own an account) and the need for null values (e.g., represent a customer without an account).

Banking Example

```
branch (branch-name, branch-city, assets)
customer (customer-name, customer-street, customer-city)
account (account-number, branch-name, balance)
loan (loan-number, branch-name, amount)
depositor (customer-name, account-number)
borrower (customer-name, loan-number)
```

Relational Algebra

- Primitives:

- Projection (π)
- Selection (σ)
- Cartesian product ($\times$)
- Set union ($\cup$)
- Set difference ($-$)
- Rename (ρ)

- Other operations:

- Join ($\bowtie$)
- Group by… aggregation
- …

What happens next?

- SQL.
- System R (DB2), INGRES, ORACLE, SQL-Server, Teradata.
    - B+-Tree (select).
    - Transaction Management.
    - Join Algorithm.

In early 90's: OLAP & Data Warehouse

Database Workloads

- OLTP (online transaction processing).
    - Typical applications: e-commerce, banking, airline reservations.
    - User-facing: real-time, low latency, highly concurrent.
    - Tasks: relatively small set of "standard" transactional queries.
    - Data access pattern: random reads, updates, writes (involving relatively small amounts of data).
- OLAP (online analytical processing).
    - Typical applications: business intelligence, data mining.
    - Back-end processing: batch workloads, less concurrency.
    - Tasks: complex analytical queries, often ad-hoc.
    - Data access pattern: table scans, large amounts of data involved per query.

OLTP

- Most database operations involve (On-Line Transaction Processing).
    - Short, simple, frequent queries and/or modifications, each involving a small number of tuples.
    - Examples: Answering queries from Web interface, sales at cash registers, selling airline tickets.

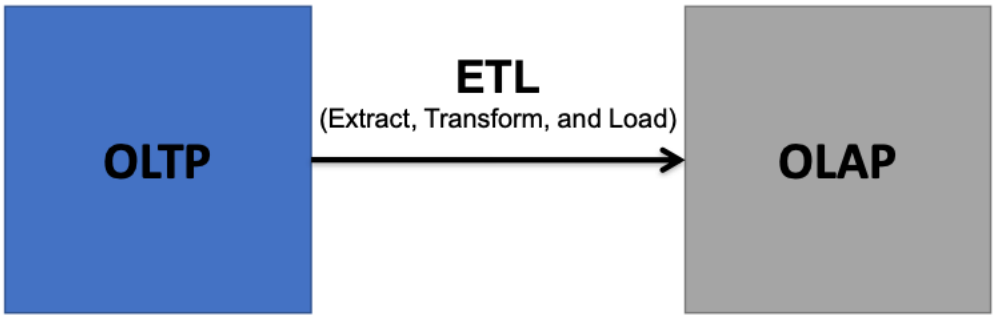
OLAP

- Of increasing importance are On-line Application Processing (OLAP) queries.
    - Few, but complex queries – may run for hours.
    - Queries do not depend on having an **ABSOLUTELY** up-to-date database.
- Examples:
    - 1. Amazon analyzes purchases by its customers to come up with an individual screen with products of likely interest to the customer.
    - 2. Analysts at Wal-Mart look for items with increasing sales in some region.

One Database or Two?

- Downsides of co-existing OLTP and OLAP workloads:
    - Poor memory management.
    - Conflicting data access patterns.
    - Variable latency.
- Solution: separate databases.
    - User-facing OLTP database for high-volume transaction.
    - Data warehouse for OLAP workloads
    - How do we connect the two?

OLTP/OLAP Architecture



OLTP/OLAP Integration

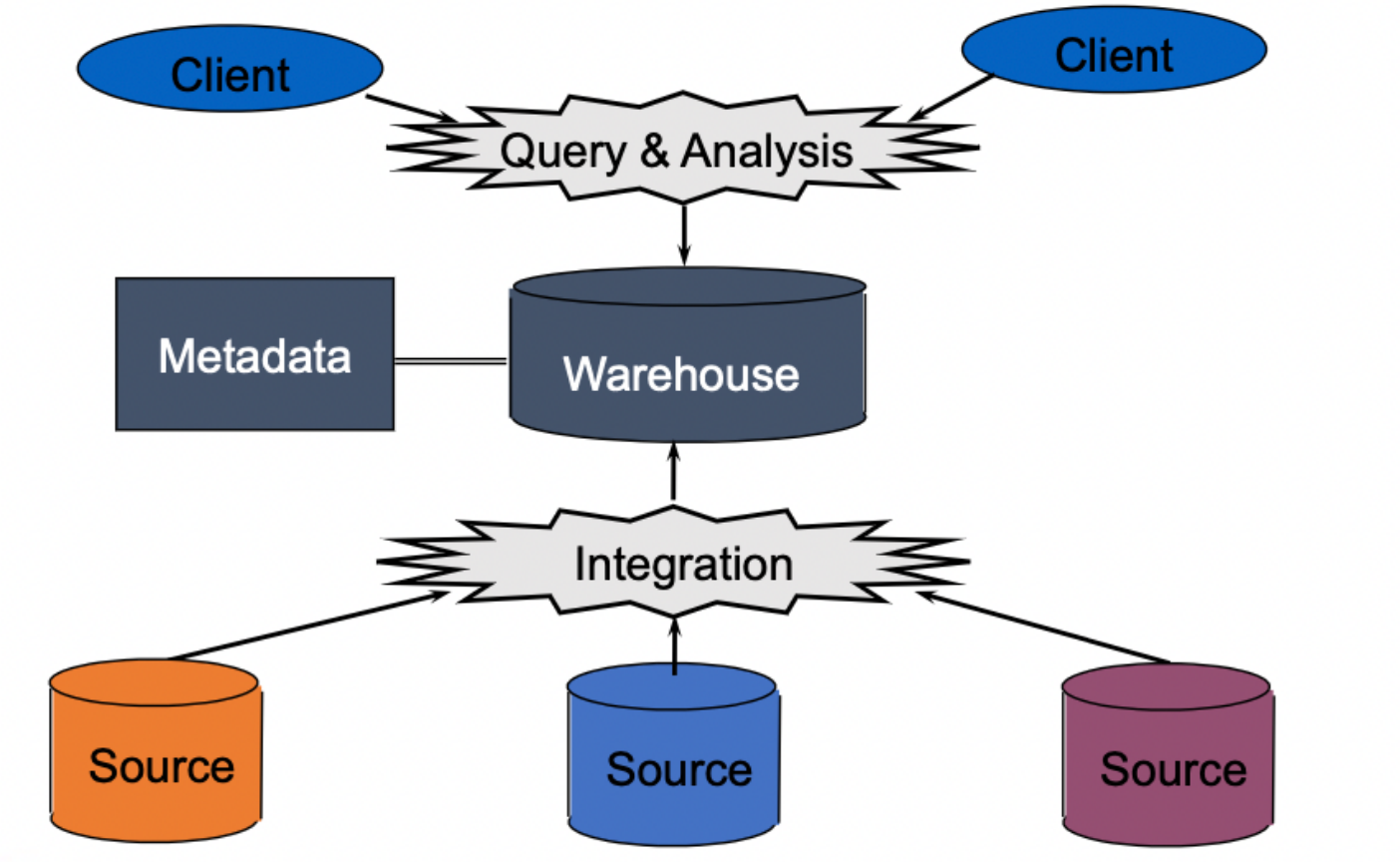
- OLTP database for user-facing transactions.
    - Retain records of all activity.
    - Periodic ETL (e.g., nightly).
- Extract-Transform-Load (ETL):
    - Extract records from source.
    - Transform: clean data, check integrity, aggregate, etc.
    - Load into OLAP database.
- OLAP database for data warehousing:

- BI: reporting, ad-hoc queries, data mining, etc.
- Feedback to improve OLTP services.

The Data Warehouse

- The most common form of data integration.
  - Copy sources into a single DB (warehouse) and try to keep it up to date.
  - Usual method: periodic reconstruction of the warehouse, perhaps overnight.
  - Frequently essential for analytic queries.

Warehouse Architecture



Star Schemas

- A star schema is a common organization for data at a warehouse. It consists of:
- 1. Fact table: a very large accumulation of facts such as sales.
  - Often “insert-only”.
- 2. Dimension tables: smaller, generally static information about the entities involved in the facts.

Example: Star Schema

- Suppose we want to record in a warehouse information about every beer sale: the bar, the brand of beer, the drinker who bought the beer, the day, the time, and the price charged.
- The fact table is a relation:

Sales(bar, beer, drinker, day, time, price)

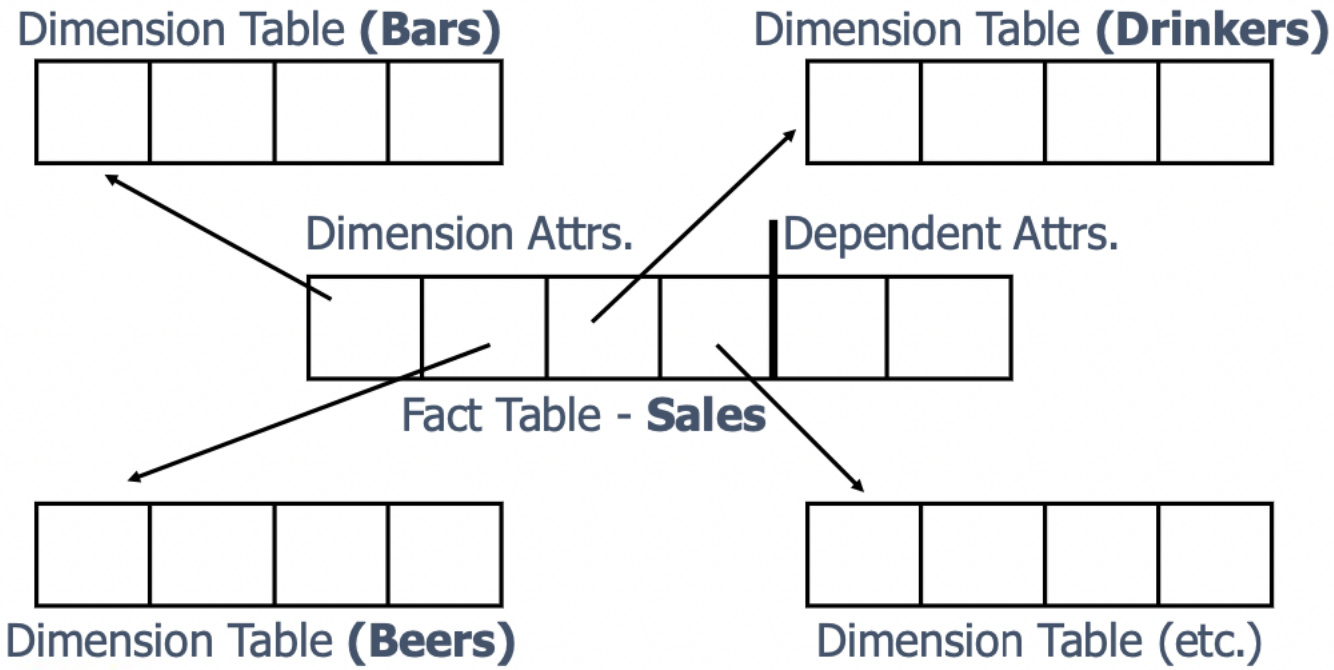
- The dimension tables include information about the bar, beer, and drinker “dimensions”:

Bars(bar, addr, license)

Beers(beer, manf)

Drinkers(drinker, addr, phone)

Visualization – Start Schema



Dimension and Dependent Attributes

- Two classes of fact-table attributes:
  - 1. Dimension attributes: the keys of dimension tables.
  - 2. Dependent attributes: a value determined by the dimension attributes of tuple.

Warehouse Models & Operators

- Data Models:
  - Relations.
  - Stars & snowflakes.
  - Cubes.
- Operators:
  - Slice & dice.
  - Roll-up, drill-down.
  - Pivoting.
  - Other.

Star

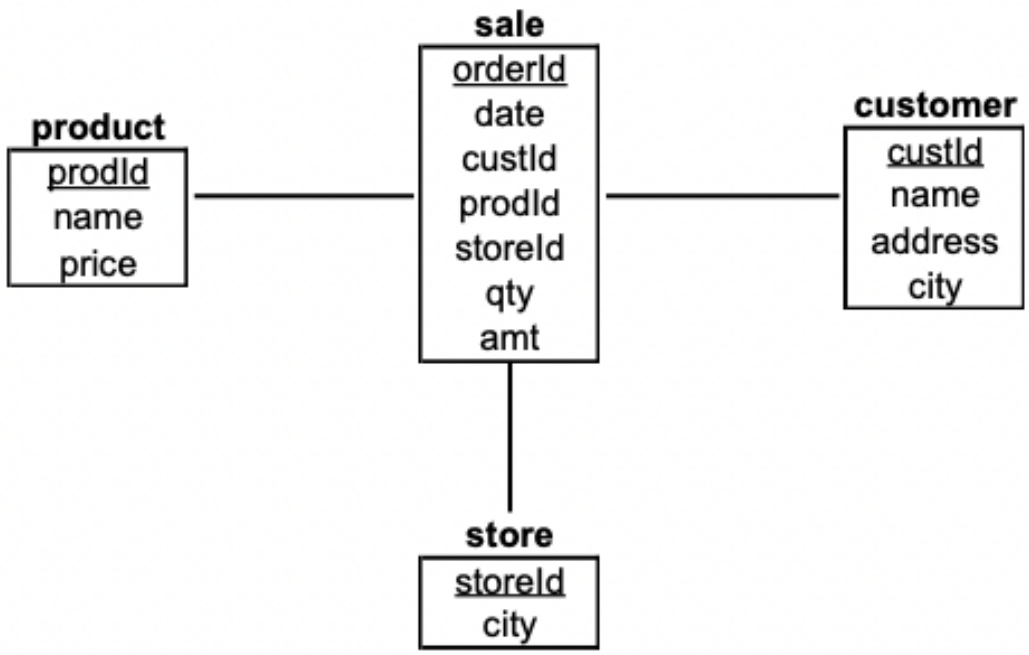
product	prodId	name	price
	p1	bolt	10
	p2	nut	5

store	storeId	city
	c1	nyc
	c2	sfo
	c3	la

sale	oderId	date	custId	prodId	storeId	qty	amt
	o100	1/7/97	53	p1	c1	1	12
	o102	2/7/97	53	p2	c1	2	11
	105	3/8/97	111	p1	c3	5	50

customer	custId	name	address	city
	53	joe	10 main	sfo
	81	fred	12 main	sfo
	111	sally	80 willow	la

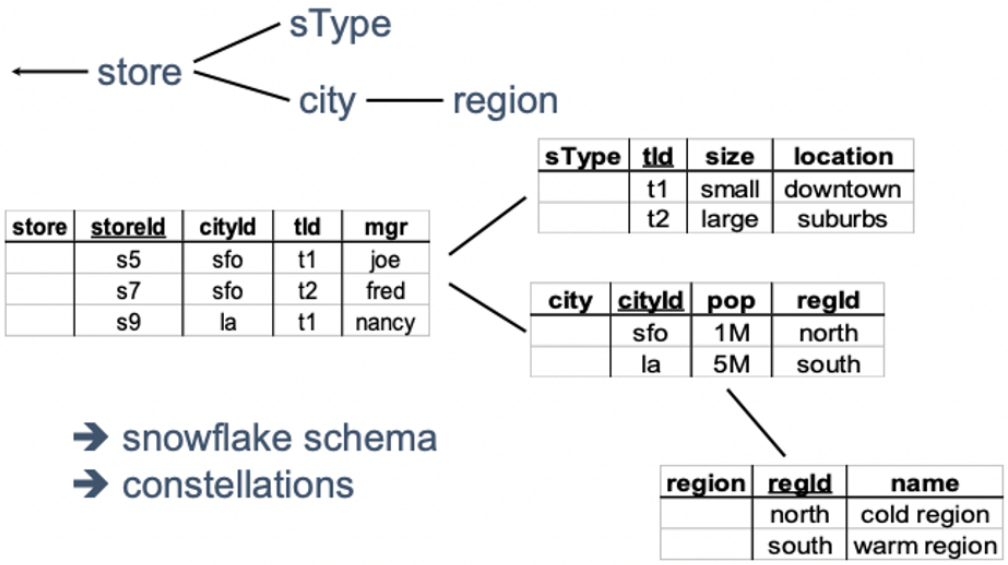
Star Schema



Terms

- Fact table.
- Dimension tables.
- Measures.

Dimension Hierarchies



Aggregates

- In SQL:

```
SELECT sum(amt) FROM SALE
  WHERE date = 1
```

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

→ 81

```
SELECT date, sum(amt) FROM SALE
  GROUP BY date
```

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

→

ans	date	sum
	1	81
	2	48

SQL:

```
SELECT date, sum(amt) FROM SALE
  GROUP BY date, prodId
```

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

→

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

rollup →

← drill-down

ROLAP VS. MOLAP

- ROLAP: Relational On-Line Analytical Processing
- MOLAP: Multi-Dimensional On-Line Analytical Processing

Cube

Fact table view:

sale	prodId	storeId	amt
	p1	c1	12
	p2	c1	11
	p1	c3	50
	p2	c2	8

Multi-dimensional cube:

	c1	c2	c3
p1	12		50
p2	11	8	

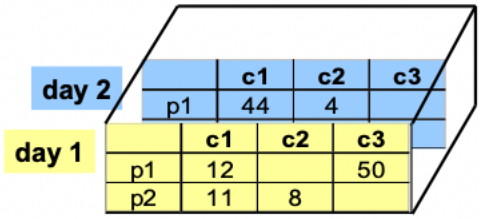
dimensions = 2

3-D Cube

Fact table view:

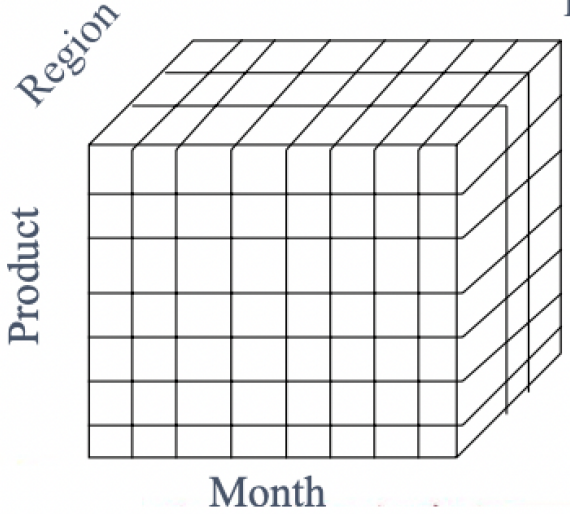
sale	prodId	storeId	date	amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4

Multi-dimensional cube:

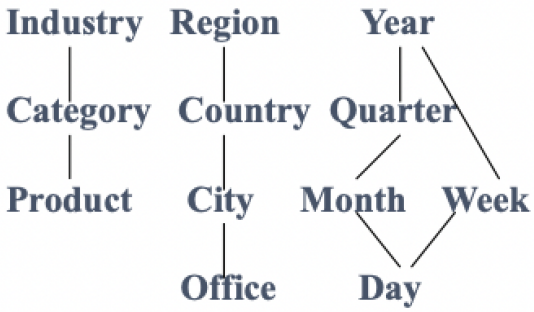


Multidimensional Data

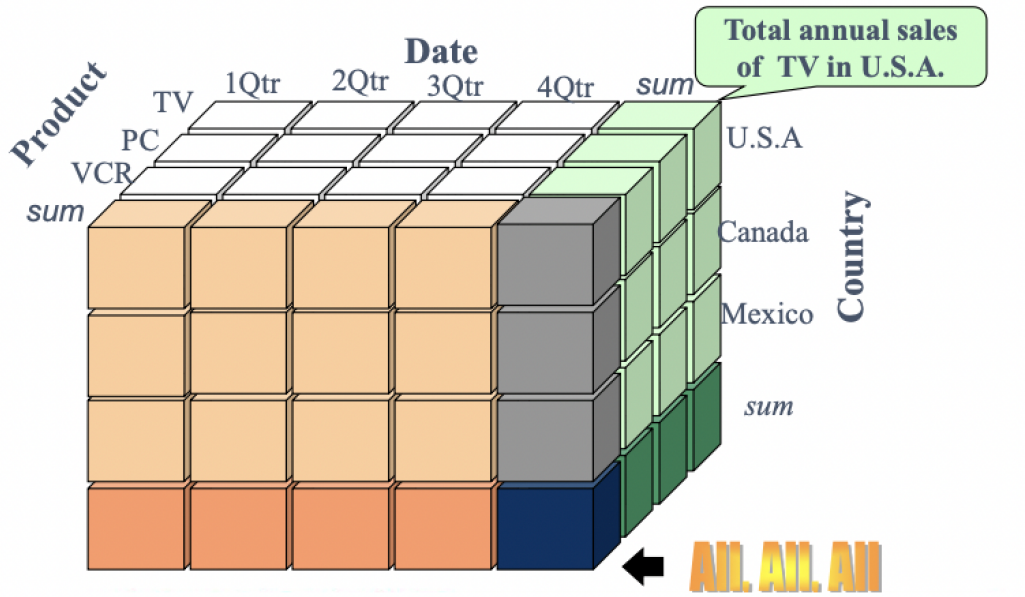
- Sales volume as a function of product, month, and region.



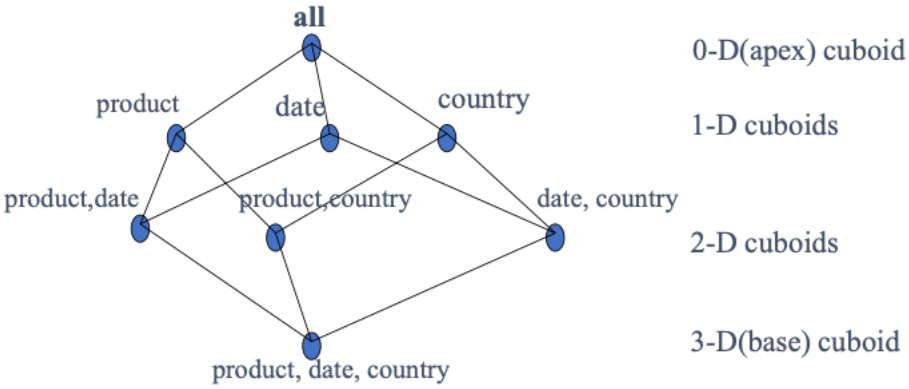
Dimensions: Product, Location, Time
Hierarchical summarization paths



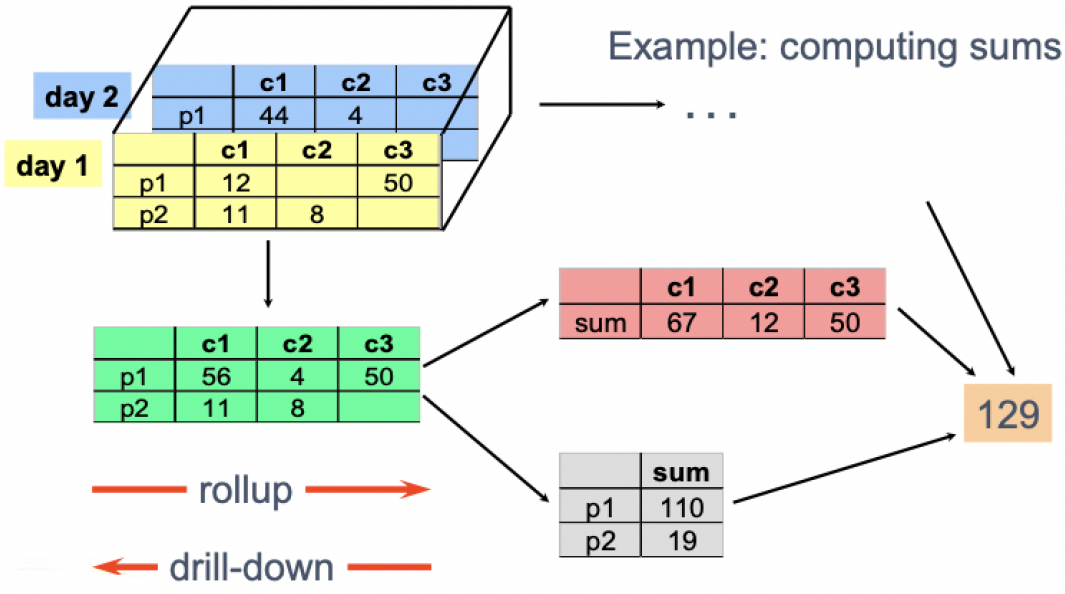
A Sample Data Cube



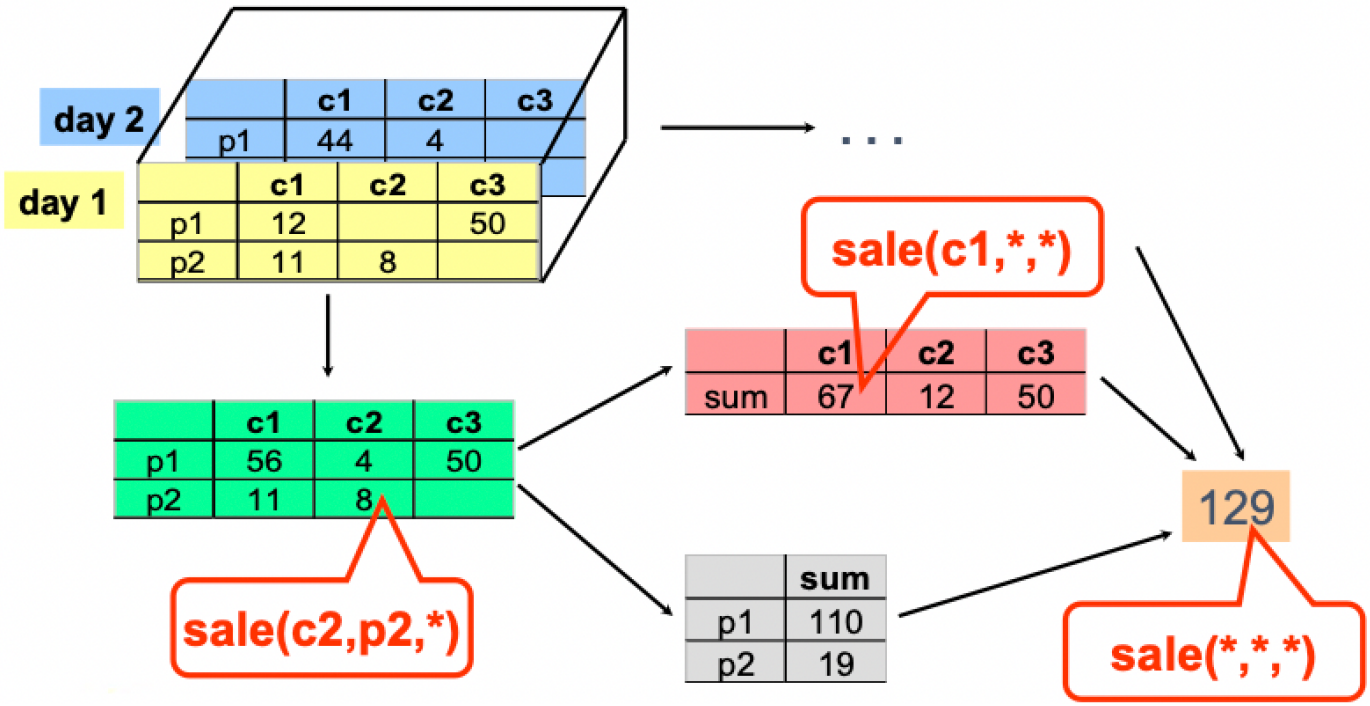
Cuboids Corresponding to the Cube



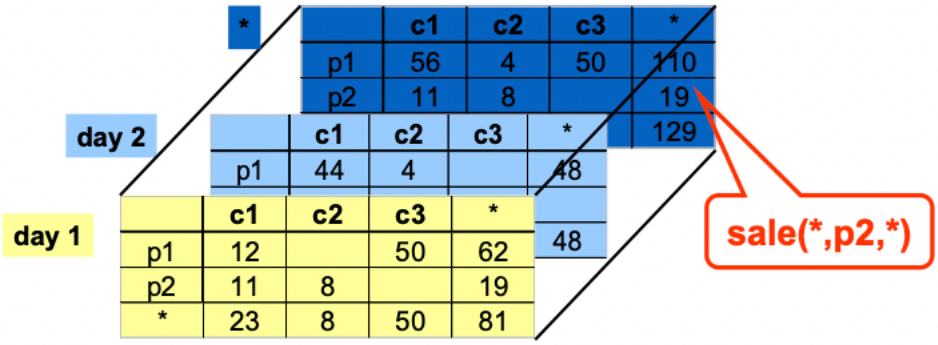
Cube Aggregation



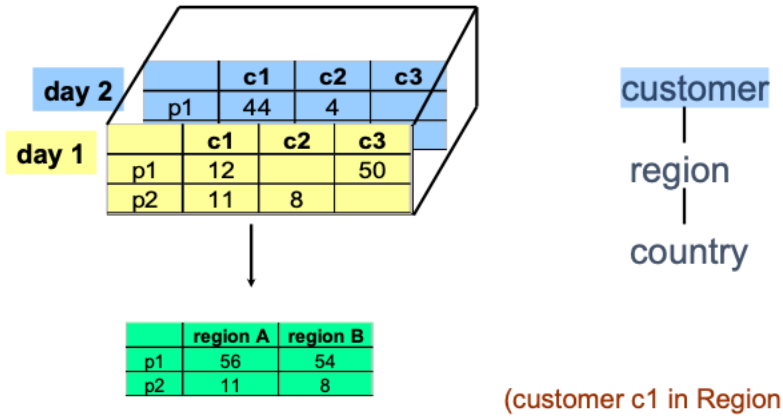
Cube Operators



Extended Cube



Aggregation Using Hierarchies

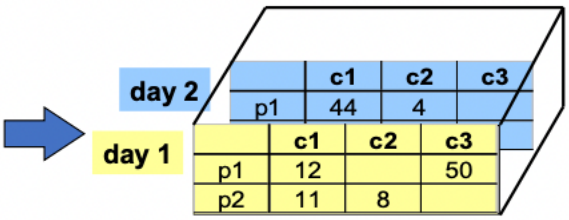


Pivoting

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	

CUBE Operator (SQL-99)

Chevy Sales Cross Tab				
Chevy	*1990*	*1991*	*1992*	*Total (ALL)*
black	50	85	154	289
white	40	115	199	354
Total (ALL)	90	200	353	1286

```
SELECT  model, year, color, sum(sales) as sales
FROM    sales
WHERE   model in ( 'Chevy' )
AND     year BETWEEN 1990 AND 1992
GROUP BY  CUBE (model, year, color);
```

- Computes union of 8 different groups:
  - {(model, year, color), (model, year), (model, color), (year, color), (model), (year), (color), ()}

Aggregates

• Operators: sum, count, max, min, median, average.
• "Having" clause.
• Cube (& Rollup) operator.
• Using dimension hierarchy.
  • average by region (within store).
  • maximum by month (within date).

Query & Analysis Tools

- Query Building.
- Report Writers (comparisons, growth, graphs, etc.)

- Spreadsheet Systems.
- Web Interfaces.
- Data Mining.

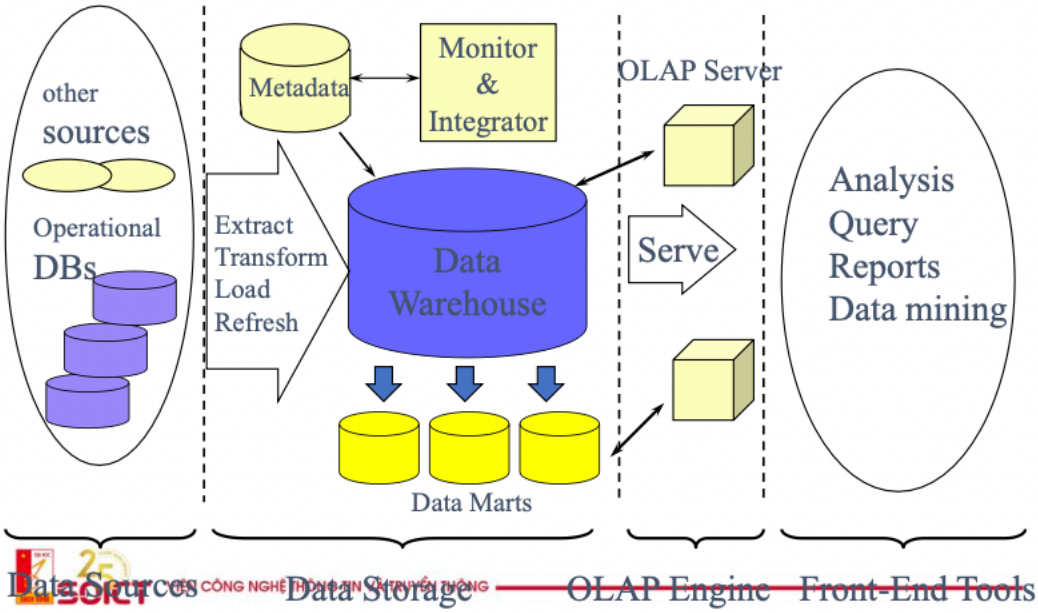
Other Operations

- Time functions.
  - E.g., time average.
- Computed Attributes:
  - E.g., commission = sales * rate.
- Text Queries:
  - E.g., find documents with words X AND B.
  - E.g., rank documents by frequency of words X, Y, Z.

Data Warehouse Implementation

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

Multi-Tiered Architecture



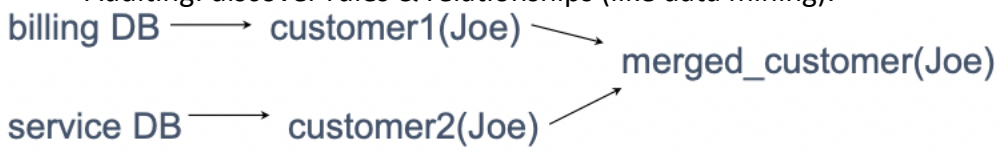
Monitoring

- Source Types: relational, flat file, IMS, VSAM, IDMS, WWW, newswire, etc.
- Incremental vs. Refresh.

customer	id	name	address	city
	53	joe	10 main	sfo
	81	fred	12 main	sfo
	111	sally	80 willow	la

Data Cleansing

- Migration (e.g., yen → dollars).
- Scrubbing: use domain-specific knowledge (e.g., social security numbers).
- Fusion (e.g., mail list, customer merging).
- Auditing: discover rules & relationships (like data mining).



Loading data

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

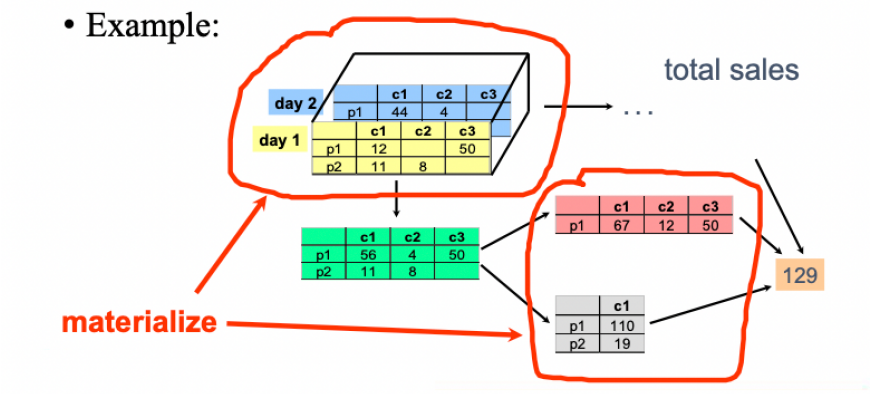
OLAP Implementation

Derived Data

- Derived Warehouse Data
  - Indexes
  - Aggregates
  - materialized views
- When to update derived data?
- Incremental vs. refresh.

What to Materialize

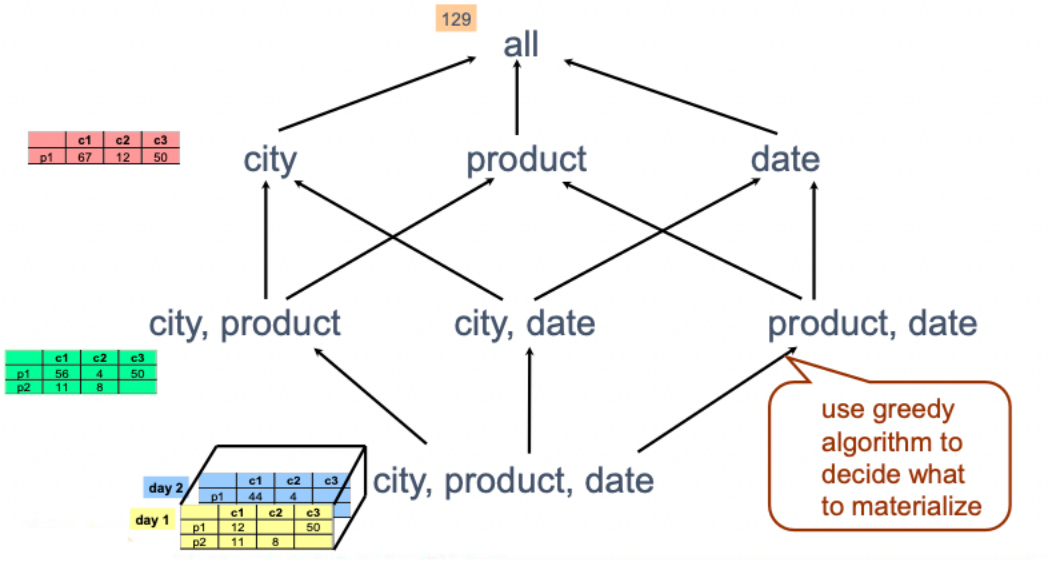
- Store in warehouse results useful for common queries.



Materialization Factors

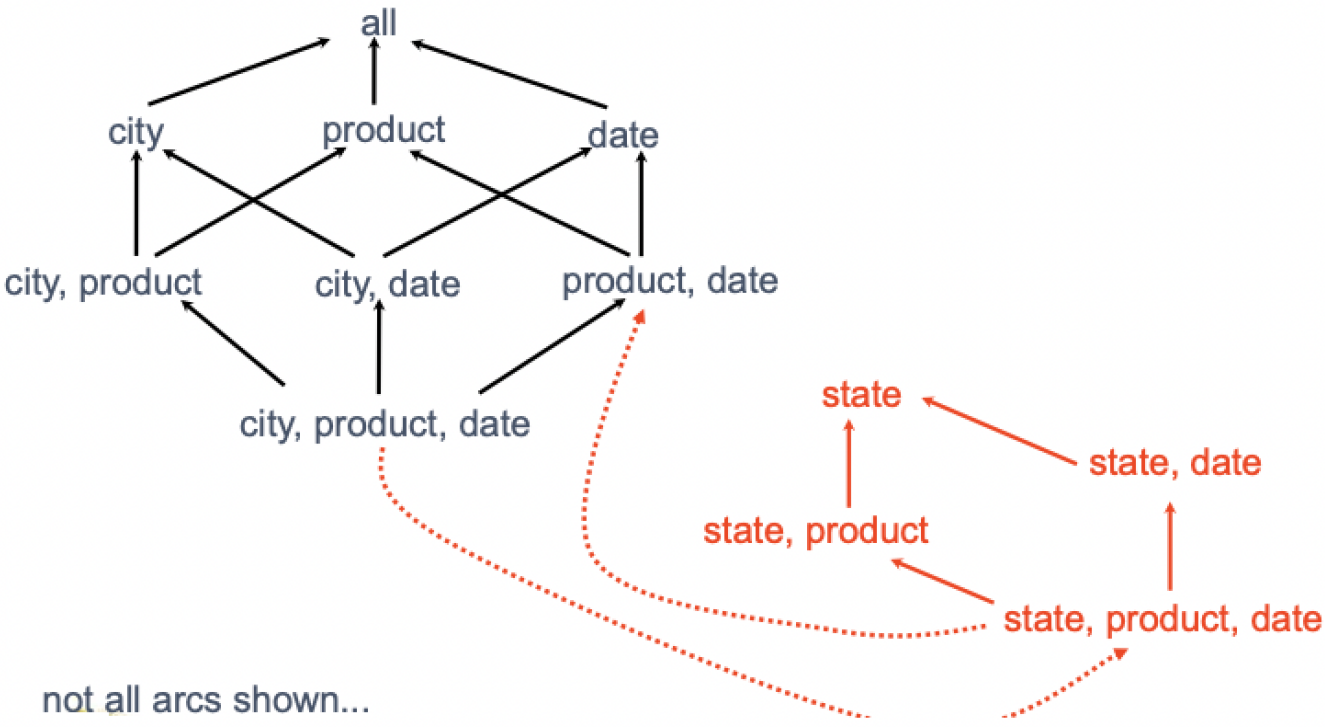
- Type/frequency of queries.
- Query response time.
- Storage Cost.
- Update Cost.

Cube Aggregates Lattice



Dimension Hierarchies

cities	city	state
	c1	CA
	c2	NY



Interesting Hierarchy

time	day	week	month	quarter	year
	1	1	1	1	2000
	2	1	1	1	2000
	3	1	1	1	2000
	4	1	1	1	2000
	5	1	1	1	2000
	6	1	1	1	2000
	7	1	1	1	2000
	8	2	1	1	2000

conceptual dimension table

Indexing OLAP Data: Bitmap Index

- 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

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

Join Processing

- How does DBMS join 2 tables?
- Sorting is one way.
- Database must choose the best way for each query.

Schema for Examples

Sailors (*sid*: integer, *sname*: string, *rating*: integer, *age*: real)
Reserves (*sid*: integer, *bid*: integer, *day*: dates, *rname*: string)

- Like old schema, `*rname*` added for variations.
- Reserves:
  - Each tuple is 40 bytes long.
  - 100 tuples per page.
  - M = 1000 pages total.
- Sailors:

- Each tuple is 50 bytes long.
- 80 tuples per page.
- N = 500 pages total.

Equality Joins with One Join Column

```
SELECT *
FROM   Reserves R1, Sailors S1
WHERE  R1.sid=S1.sid
```

- In algebra: R⋈S. Common! Must be carefully optimized. R × S is large; so, R × S followed by a selection is inefficient.
- Assume: M tuples in R, p_R tuples per page, N tuples in S, p_S tuples per page.
  - In our examples, R is Reserves and S is Sailors.
- We will consider more complex join conditions later.
- *Cost metric*: # of I/Os. We will ignore output costs.

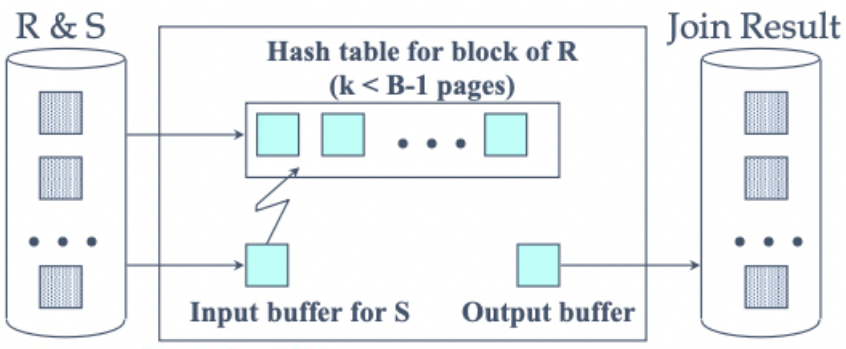
Simple Nested Loops Join

```
foreach tuple r in R do
    foreach tuple s in S do
        if ri == sj then add <r, s> to result
```

- For each tuple in the *outer* relation R, we scan the entire *inner* relation S.
  - Cost: $M + p_R * M * N = 1000 + 100*1000*500$ I/Os.
- Page-oriented Nested Loops join: For each *page* of R, get each *page* of S, and write out matching pairs of tuples <r, s>, where r is in R-page and S is in S-page.
  - Cost: $M + M*N = 1000 + 1000*500$
  - If smaller relation (S) is outer, cost = 500 + 500*1000

Block Nested Loops Join

- Use one page as an input buffer for scanning the inner S, one page as the output buffer, and use all remaining pages to hold ```block``` of outer R.
- For each matching tuple r in R-Block, s in S-page, add <r, s> to result. Then read next R-block, scan S, etc.



Examples of Block Nested Loops

- Cost: Scan of outer + #outer blocks * scan of inner
  - #outer blocks = $\lceil \#\ of\ pages\ of\ outer\ /\ blocksize \rceil$
- With Reserves (R) as outer, and 100 pages of R:
  - Cost of scanning R is 1000 I/Os; a total of 10 *blocks*.
  - Per block of R, we scan Sailors (S); 10*500 I/Os.
  - If space for just 90 pages of R, we would scan S 12 times.
- With 100-page block of Sailors as outer:
  - Cost of scanning S is 500 I/Os; a total of 5 blocks.
  - Per block of S, we scan Reserves; 5*1000 I/Os.
- With *sequential reads* considered, analysis changes: may be best to divide buffers evenly between R and S.

Index Nested Loops Join

```
foreach tuple r in R do
    foreach tuple s in S where r_i == s_j do
        add <r, s> to result
```

- If there is an index on the join column of one relation (say S), can make it the inner and exploit the index.
  - Cost: $M + ((M*p_R) *$ cost of finding matching S tuples)
- For each R tuple, cost of probing S index is about 1.2 for hash index, 2-4 for B+ tree. Cost of then finding S tuples (assuming Alt. (2) or (3) for data entries) depends on clustering.
  - Clustered index: 1 I/O (typical), unclustered: upto 1 I/O per matching S tuple.

Examples of Index Nested Loops

- Hash-index (Alt. 2) on *sid* of Sailors (as inner):
  - Scan Reserves: 1000 page I/Os, 100*1000 tuples.
  - For each Reserves tuple: 1.2 I/Os to get data entry in index, plus 1 I/O to get (the exactly one) matching Sailors tuple. Total: 220,000 I/Os.
- Hash-index (Alt. 2) on *sid* of Reserves (as inner):
  - Scan Sailors: 500 page I/Os, 80*500 tuples.
  - For each Sailors tuple: 1.2 I/Os to find index page with data entries, plus cost of retrieving matching Reserves tuples. Assuming uniform distribution, 2.5 reservations per sailor (100,000 / 40,000). Cost of retrieving them is 1 or 2.5 I/Os depending on whether the index is clustered.

Sort-Merge Join $(R \bowtie S)_{i=j}$

- Sort R and S on the join column, then scan them to do a ```merge``` (on join column), and input result tuples.
    - Advance scan of R until current R-tuple >= current S tuple, then advance scan of S until current S-tuple >= current R tuple; do this until current R tuple = current S tuple.
    - At this point, all R tuples with same value in Ri (current R group) and all S tuples with same value in Sj (current S group) match; output <r, s> for all pairs of such tuples.
    - • Then resume scanning R and S.
- R is scanned once; each S group is scanned once per matching R tuple. (Multiple scans of an S group are likely to find needed pages in buffer).

Example of Sort-Merge Join

sid	sname	rating	age
22	dustin	7	45.0
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

sid	bid	day	rname
28	103	12/4/96	guppy
28	103	11/3/96	yuppy
31	101	10/10/96	dustin
31	102	10/12/96	lubber
31	101	10/11/96	lubber
58	103	11/12/96	dustin

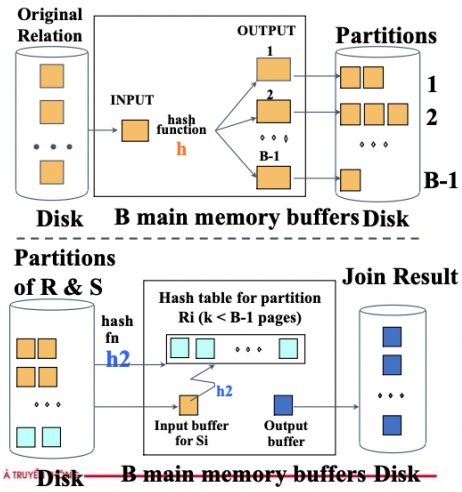
- Cost: M log M + N log N + (M+N)
  - The cost of scanning, M+N, could be M*N (very unlikely!)
- With 35, 100 or 300 buffer pages, both Reserves and Sailors can be sorted in 2 passes; total join cost: 7500.

Refinement of Sort-Merger Join

- We can combine the merging phases in the *sorting* of R and S with the merging required for the join.
  - With $B > \sqrt{L}$, where L is the size of the larger relation, using the sorting refinement that produces runs of length 2B in Pass 0, #runs of each relation is < B/2.
  - Allocate 1 page per run of each relation, and `merge' while checking the join condition.
  - Cost: read+write each relation in Pass 0 + read each relation in (only) merging pass (+ writing of result tuples).
  - In example, cost goes down from 7500 to 4500 I/Os.
- In practice, cost of sort-merge join, like the cost of external sorting, is *linear*.

Hash-Join

- Partition both relations using hash fn h: R tuples in partition I will only match S tuples in partition i.
- Read in a partition of R, hash it using h2 (<> h!). Scan matching partition of S, search for matches.



Observations in Hash-Join

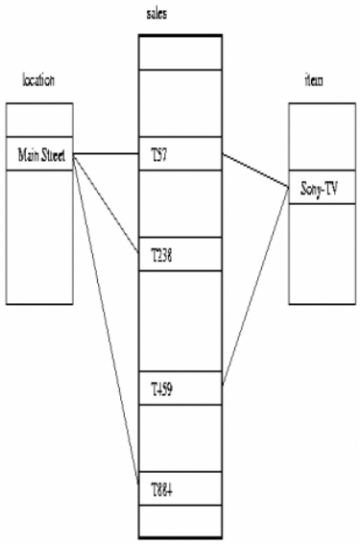
- #partitions k < B-1 (why?), and B-2 > size of largest partition to be held in memory. Assuming uniformly sized partitions, and maximizing k, we get:
  - k= B-1, and M/(B-1) < B-2, i.e., B must be > $\sqrt{M}$
- If we build an in-memory hash table to speed up the matching of tuples, a little more memory is needed.
- If the hash function does not partition uniformly, one or more R partitions may not fit in memory. Can apply hash-join technique recursively to do the join of this R-partition with corresponding S-partition.

Cost of Hash-Join

- In partitioning phase, read+write both relns; 2(M+N). In matching phase, read both relns; M+N I/Os.
- In our running example, this is a total of 4500 I/Os.
- Sort-Merge Join vs. Hash Join:
  - Given a minimum amount of memory (*what is this, for each?*) both have a cost of 3(M+N) I/Os. Hash Join superior on this count if relation sizes differ greatly. Also, Hash Join shown to be highly parallelizable.
  - Sort-Merge less sensitive to data skew; result is sorted.

Join Indices

- 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 start 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



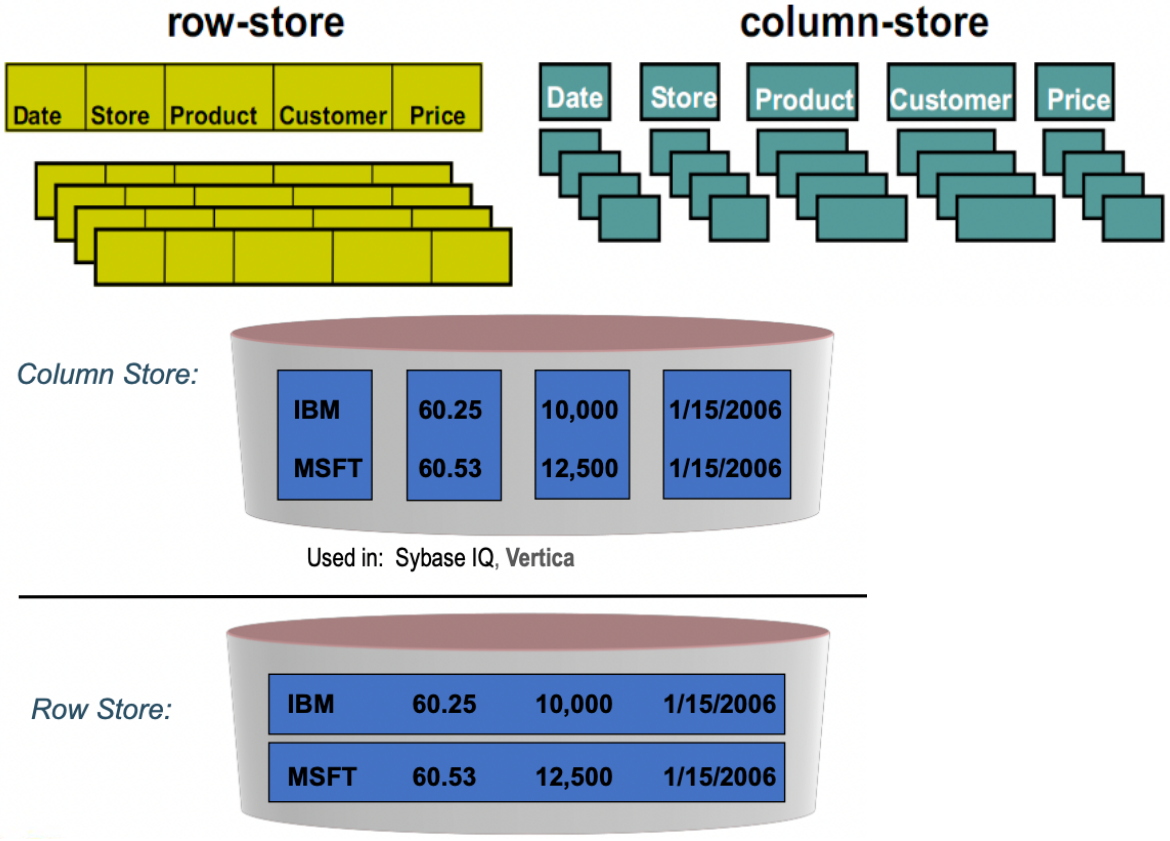
General Join Conditions

- Equalities over several attributes (e.g., *R.sid=S.sid* AND *R.rname=S.sname*):
  - For Index NL, build index on *<sid, sname>* (if S is inner); or use existing indexes on *sid* or *sname*.
  - For Sort-Merge and Hash Join, sort/partition on combination of the two join columns.
- Inequality conditions (e.g., *R.rname < S.sname*):
  - For Index NL, need (clustered!) B+ tree index.
    - Range probes on inner; # matches likely to be much higher than for equality joins.
  - Hash Join, Sort Merge Join not applicable.
  - Block NL quite likely to be the best join method here.

An invention in 2000s: Column Stores for OLAP

Row Store and Column Store

- In row store data are stored in the disk tuple by tuple.
- Where in column store data are stored in the disk column by column



For example the query

```
SELECT account.account_number,
        sum (usage.toll_airtime),
        sum (usage.toll_price)
FROM usage, toll, source, account
WHERE usage.toll_id = toll.toll_id
AND usage.source_id = source.source_id
AND usage.account_id = account.account_id
AND toll.type_ind in ('AE'. 'AA')
AND usage.toll_price > 0
AND source.type != 'CIBER'
AND toll.rating_method = 'IS'
AND usage.invoice_date = 20051013
GROUP BY account.account_number
```

Row-store: one row = 212 columns!
Column-store: 7 attributes

Row Store	Column Store
(+) Easy to add/modify a record	(+) Only need to read in relevant data
(-) Might read in unnecessary data	(-) Tuple writes require multiple accesses

- So, column storages are suitable for read-mostly, read intensive, large data repositories.

Column Stores: High Level

- Read only what you need:
  - "Fat" fact tables are typical.
  - Analytics read only a few columns.
- Better compression.
- Execute and compressed data.
- Materialized views help row stores and column stores about equally.

Data Model (Vertica/C-Store)

- Same as relational data model:
  - Tables, rows, columns.
  - Primary keys and foreign keys.
  - Projections:
    - From single table.
    - Multiple joined tables.
- Example:

Normal relational model

EMP(name, age, dept, salary)

DEPT(dname, floor)

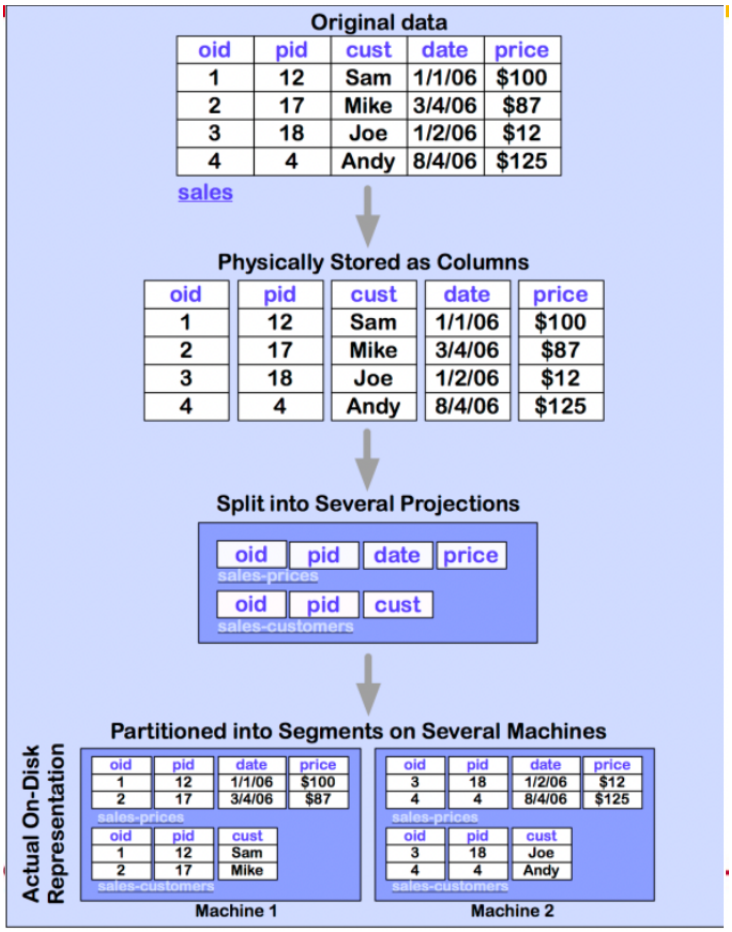
Possible C-store model

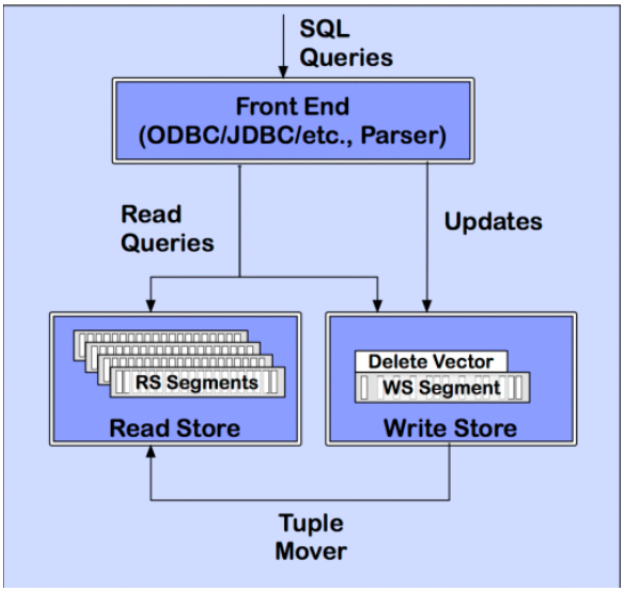
EMP1 (name, age)

EMP2 (dept, age, DEPT.floor)

EMP3 (name, salary)

DEPT1(dname, floor)





Read store: Column Encoding/Compression

- Use compression schemes and indices:
  - Null Suppression.
  - Dictionary encoding.
  - Run Length encoding.
  - Bit-Vector encoding.

– Self-order (key), few distinct values
  - (value, position, # items)
  - Indexed by clustered B-tree
– Foreign-order (non-key), few distinct values
  - (value, bitmap index)
  - B-tree index: position → values
– Self-order, many distinct values
  - Delta from the previous value
  - B-tree index
– Foreign-order, many distinct values
  - Unencoded

Compression

- Trades I/O for CPU
  - Increased column-store opportunities.
  - Higher data value locality in column stores.
  - Techniques such as run length encoding far more useful.

Write Store

- Same structure, but explicitly use (segment, key) to identify records.
  - Easier to maintain the mapping.
  - Only concerns the inserted records.
- Tuple mover:
  - Copies batch of records to RS.
- Delete record:
  - Mark it on RS.
  - Purged by tuple mover.

How to solve read/write conflict

- Situation: one transaction updates the record X, while another transaction reads X.
- Use snapshot isolation.

Query Execution – Operators

- Select: Same as relational algebra but produces a bit string.
- Project: Same as relational algebra.
- Join: Joins projections according to predicates.
- Aggregation: SQL like aggregates.
- Sort: Sort all columns of a projection.
- Decompress: Converts compressed column to uncompressed representation.
- Mask(Bitstring B, Projection Cs) => emit only those values whose corresponding bits are 1.
- Concat: Combines one or more projections sorted in the same order into a single projection.
- Permute: Permutes a projection according to the ordering defined by a join index.
- Bitstring operators: Band – Bitwise AND, Bor – Bitwise OR, Bnot – complement.

Benefits in Query Processing

- Selection – has more indices to use.
- Projection – some “projections” already defined.
- Join – some projections are materialized joins.
- Aggregations – works on required columns only.

Evaluation

- Use TPC-H – decision support queries.
- Storage

C-Store	Row Store	Column Store
1.987 GB	4.480 GB	2.650 GB

Query Performance

Query	C-Store	Row Store	Column Store
Q1	0.03	6.80	2.24
Q2	0.36	1.09	0.83
Q3	4.90	93.26	29.54
Q4	2.09	722.90	22.23
Q5	0.31	116.56	0.93
Q6	8.50	652.90	32.83
Q7	2.54	265.80	33.24

• Row store uses materialized views

Query	C-Store	Row Store	Column Store
Q1	0.03	0.22	2.34
Q2	0.36	0.81	0.83
Q3	4.90	49.38	29.10
Q4	2.09	21.76	22.23
Q5	0.31	0.70	0.63
Q6	8.50	47.38	25.46
Q7	2.54	18.47	6.28

Summary: The performance gain

- Column representation: avoid reads of unused attributes.
- Storing overlapping projections – multiple orderings of a column, more choices for query optimization.
- Compression of data – more orderings of a column in the same amount of space.
- Query operators operate on compressed representation.

Google's Dremel/Big Query: Interactive Analysis of Web-Scale Datasets

Big Query System

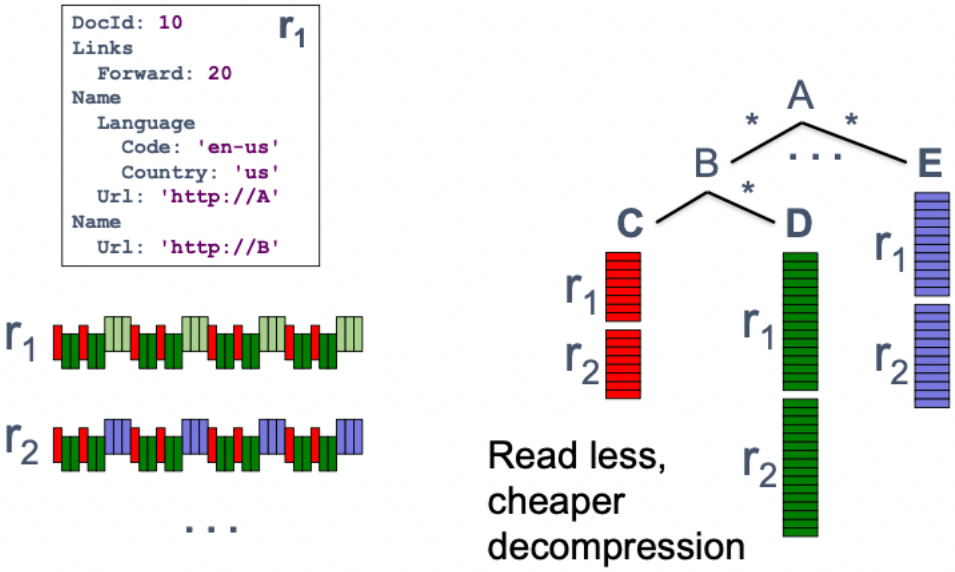
• Trillion-record, multi-terabyte datasets at interactive speed
    • Scales to thousands of nodes
    • Fault and straggler tolerant execution
• Nested data model
    • Complex datasets; normalization is prohibitive
    • Columnar storage and processing
• Tree architecture (as in web search)
• Interoperates with Google's data mgmt tools
    • In situ data access (e.g., GFS, Bigtable)
    • MapReduce pipelines

Widely Used inside Google

- Analysis of crawled web documents
- Tracking install data for applications on Android Market
- Crash reporting for Google products
- OCR results from Google Books
- Spam analysis
- Debugging of map tiles on Google Maps

- Tablet migrations in managed Bigtable instances
- Results of tests run on Google's distributed build system
- Disk I/O statistics for hundreds of thousands of disks
- Resource monitoring for jobs run in Google's data centers
- Symbols and dependencies in Google's codebase

Records vs. columns



Challenge: Preserve structure, reconstruct from a subset of fields.

Nested Data Model

http://code.google.com/apis/protocolbuffers

multiplicity:

```
message Document {
  required int64 DocId;              [1,1]
  optional group Links {
    repeated int64 Backward;         [0,*]
    repeated int64 Forward;
  }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country;       [0,1]
    }
    optional string Url;
  }
}
```

r_1

```
DocId: 10
Links
  Forward: 20
  Forward: 40
  Forward: 60
Name
  Language
    Code: 'en-us'
    Country: 'us'
  Language
    Code: 'en'
  Url: 'http://A'
Name
  Url: 'http://B'
Name
  Language
    Code: 'en-gb'
    Country: 'gb'
```

r_2

```
DocId: 20
Links
  Backward: 10
  Backward: 30
  Forward:  80
Name
  Url: 'http://C'
```


Column-striped representation

DocId

value	r	d
10	0	0
20	0	0

Name.Url

value	r	d
http://A	0	2
http://B	1	2
NULL	1	1
http://C	0	2

Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2
80	0	2

Links.Backward

value	r	d
NULL	0	1
10	0	2
30	1	2

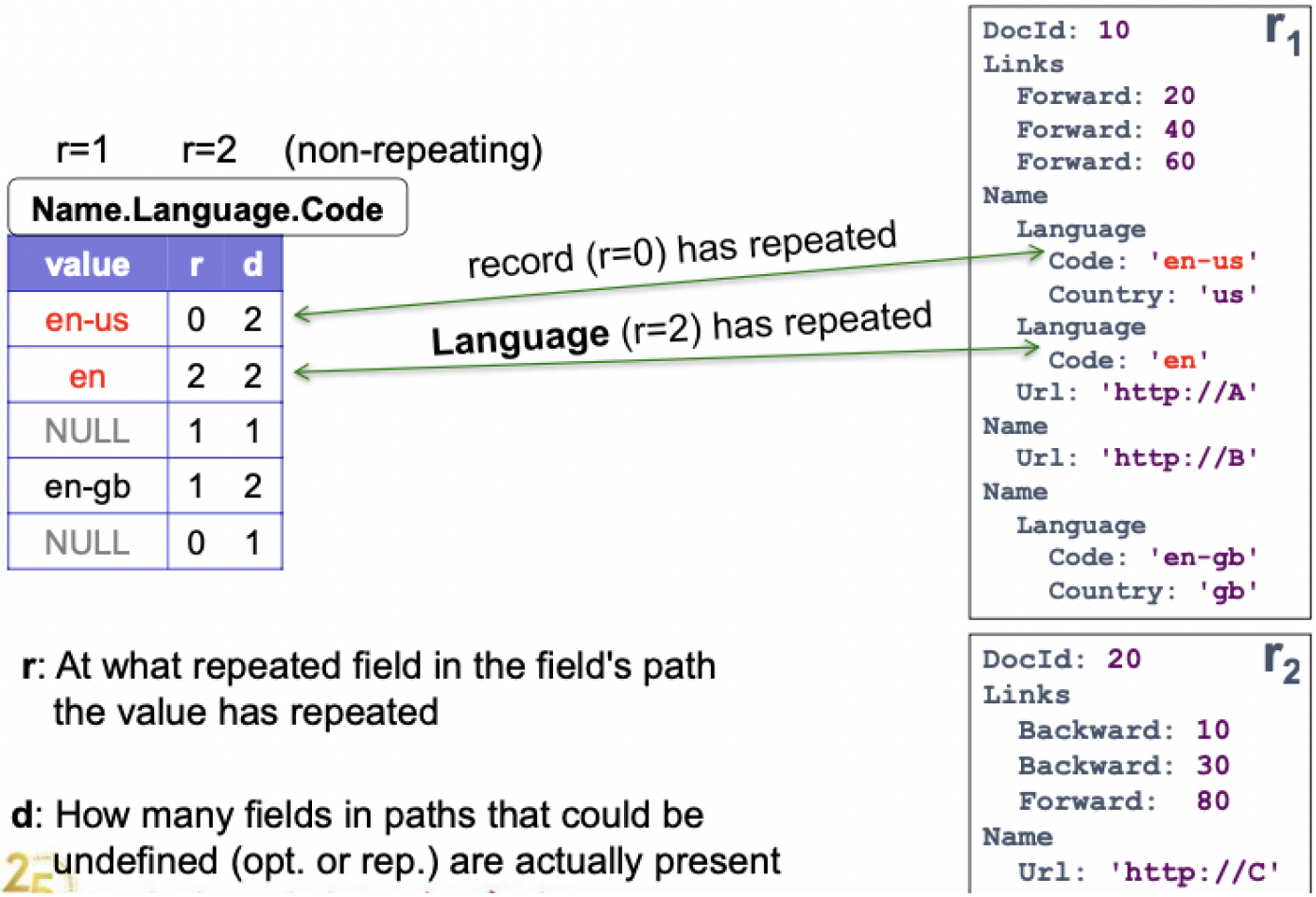
Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2
NULL	0	1

Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NULL	1	1
gb	1	3
NULL	0	1

Repetition and definition levels



Query processing

- Optimized for select-project-aggregate
  - Very common class of interactive queries
  - Single scan
  - Within-record and cross-record aggregation
- Approximations: count(distinct), top-k.
- Joins, temp tables, UDFs/TVFs, etc.

SQL dialect for nested data

```
SELECT DocId AS Id,
  COUNT(Name.Language.Code) WITHIN Name AS Cnt,
  Name.Url + ',' + Name.Language.Code AS Str
FROM t
WHERE REGEXP(Name.Url, '^http') AND DocId < 20;
```

Output table

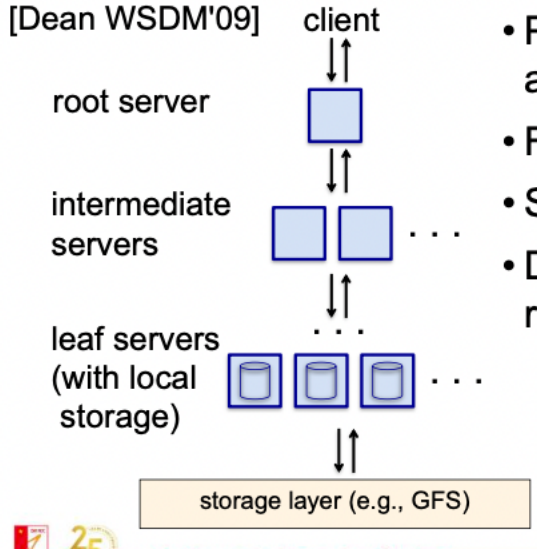
```
Id: 10
Name
  Cnt: 2
  Language
    Str: 'http://A,en-us'
    Str: 'http://A,en'
Name
  Cnt: 0
```

t1

Output schema

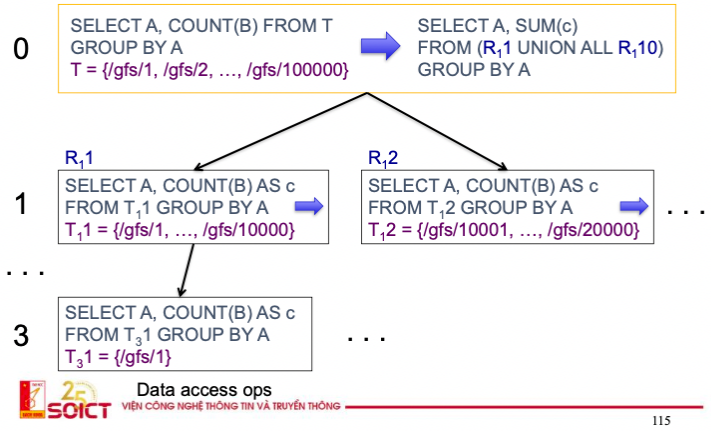
```
message QueryResult {
  required int64 Id;
  repeated group Name {
    optional uint64 Cnt;
    repeated group Language {
      optional string Str;
    }
  }
}
```

Serving tree



- Parallelizes scheduling and aggregation
- Fault tolerance
- Stragglers
- Designed for "small" results (<1M records)

Example: count()

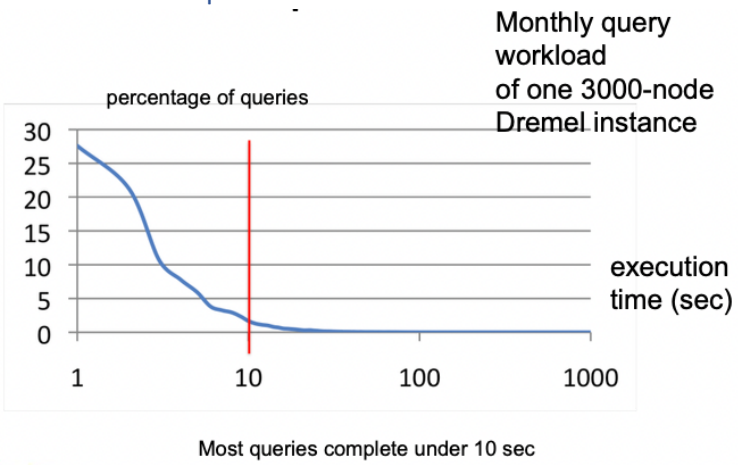


Experiments

- 1 PB of real data (uncompressed, non-replicated).
- 100k-800k tablets per table.
- Experiments run during business hours:

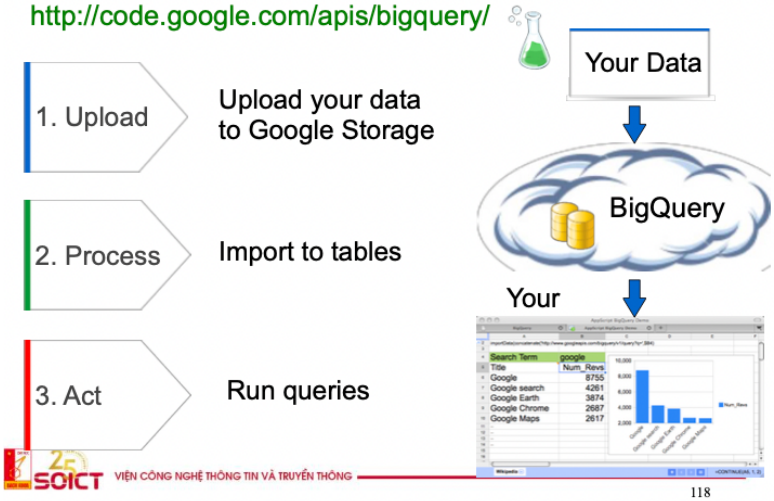
Table name	Number of records	Size (unrepl., compressed)	Number of fields	Data center	Repl. factor
T1	85 billion	87 TB	270	A	3×
T2	24 billion	13 TB	530	A	3×
T3	4 billion	70 TB	1200	A	3×
T4	1+ trillion	105 TB	50	B	3×
T5	1+ trillion	20 TB	30	B	2×

Interactive Speed



Most queries complete under 10 sec

Big Query: Powered by Dremel



List of Column Data Bases

- Vertica/C-Store
- SybaseIQ
- MonetDB
- LucidDB
- HANA
- Google's Dremel
- Parcell -> Redshit (Another Cloud-DB Service)

Take-home messages

- OLAP
    - Multi-relational Data model
    - Operators
    - SQL
- Data warehouse (architecture, issues, optimizations)
- Join Processing
- Column Stores (Optimized for OLAP workload)

Table of Contents


ĐẠI HỌC
BÁCH KHOA
25 YEARS ANNIVERSARY
SOICT
ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

1

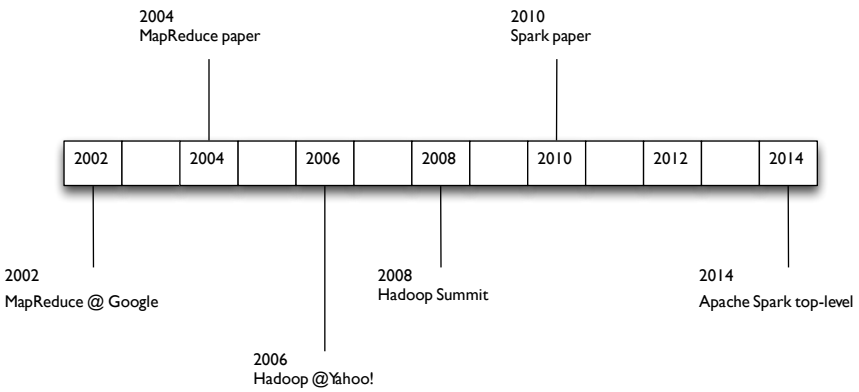

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
Spark Introduction

2

Agenda

- History of Spark
- Introduction
- Components of Stack
- Resilient Distributed Dataset – RDD

History of Spark



History of Spark

circa 1979 – **Stanford**, **MIT**, **CMU**, etc.
set/list operations in LISP, Prolog, etc., for parallel processing
www-formal.stanford.edu/ jmc/ history/ lisp/ lisp.htm

circa 2004 – **Google**
MapReduce: Simplified Data Processing on Large Clusters Jeffrey Dean and Sanjay Ghemawat
research.google.com/ archive/ mapreduce.html

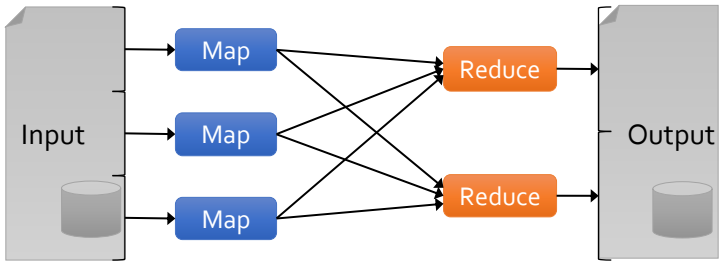
circa 2006 – **Apache**
Hadoop, originating from the Nutch Project Doug Cutting
research.yahoo.com/ files/ cutting.pdf

circa 2008 – **Yahoo**
web scale search indexing Hadoop Submit, HUG, etc.
developer.yahoo.com/hadoop/

circa 2009 – **Amazon AWS**
Elastic MapReduce
Hadoop modified for EC2/S3, plus support for Hive, Pig, Cascading, etc.
aws.amazon.com/ elasticmapreduce/

MapReduce

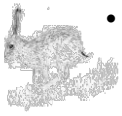
Most current cluster programming models are based on *acyclic data flow* from stable storage to stable storage



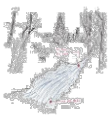
MapReduce

- Acyclic data flow is inefficient for applications that repeatedly reuse a *working set* of data:
  - **Iterative** algorithms (machine learning, graphs)
  - **Interactive** data mining tools (R, Excel, Python)

Data Processing Improvement Goals



- **Low latency (interactive) queries on historical data:** enable faster decisions
  - E.g., identify why a site is slow and fix it

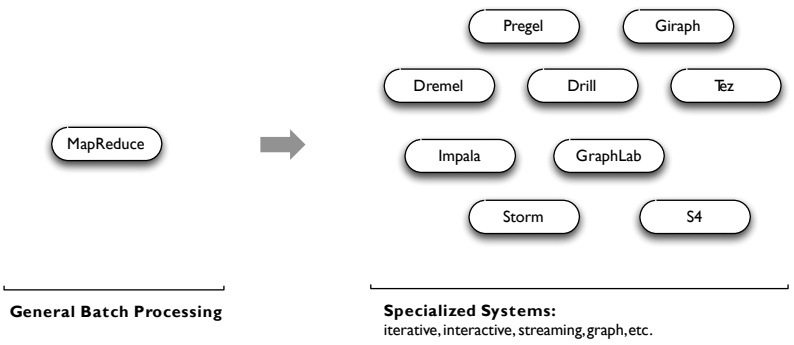


- **Low latency queries on live data (streaming):** enable decisions on real-time data
  - E.g., detect & block worms in real-time (a worm may infect **1mil** hosts in **1.3sec**)

- **Sophisticated data processing:** enable "better" decisions
  - E.g., anomaly detection, trend analysis

Therefore, people built specialized systems as workarounds…

Specialized Systems



The State of Spark, and Where We're Going Next
Matei Zaharia
Spark Summit (2013)
youtu.be/ nU6vO2EJAb4

Storage vs Processing Wars

NoSQL battles

Relational vs NoSQL

HBase vs Cassanrdra

Redis vs Memcached vs Riak

MongoDB vs CouchDB vs Couchbase

Neo4j vs Titan vs Giraph vs OrientDB

Solr vs Elasticsearch

Compute battles

MapReduce vs Spark

Spark Streaming vs Storm

Hive vs Spark SQL vs Impala

Mahout vs MLlib vs H20

Storage vs Processing Wars

NoSQL battles

Relational vs NoSQL

HBase vs Cassanrdra

Redis vs Memcached vs Riak

MongoDB vs CouchDB vs Couchbase

Neo4j vs Titan vs Giraph vs OrientDB

Solr vs Elasticsearch

Compute battles

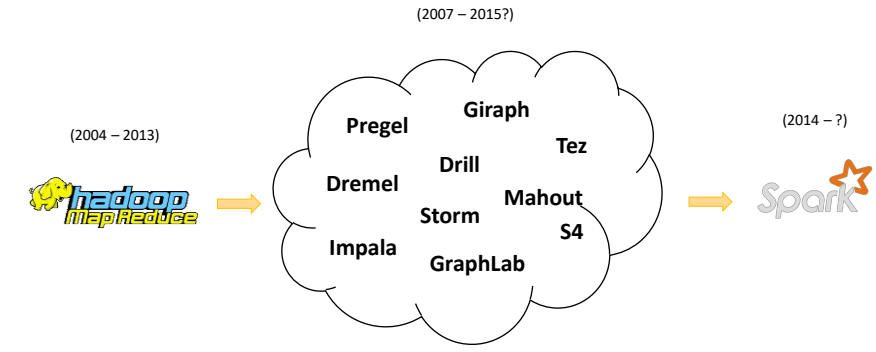
MapReduce vs Spark

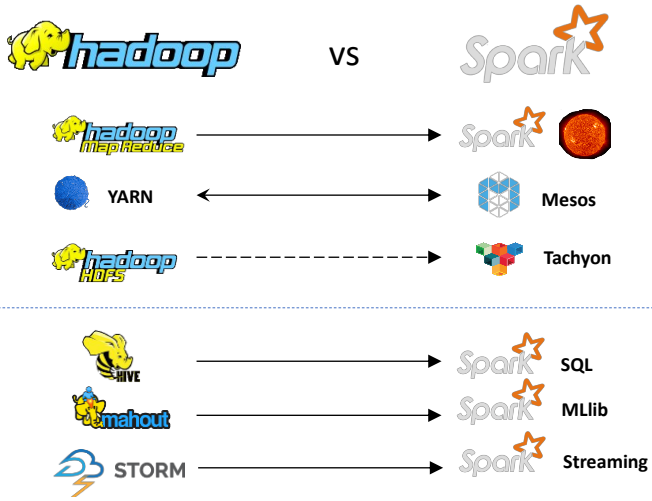
Spark Streaming vs Storm

Hive vs Spark SQL vs Impala

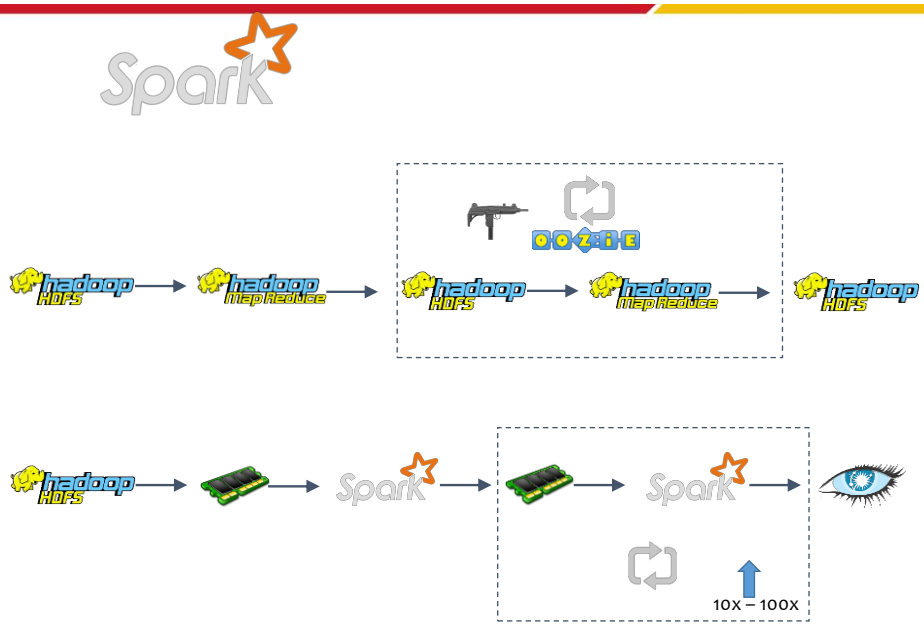
Mahout vs MLlib vs H20

Specialized Systems





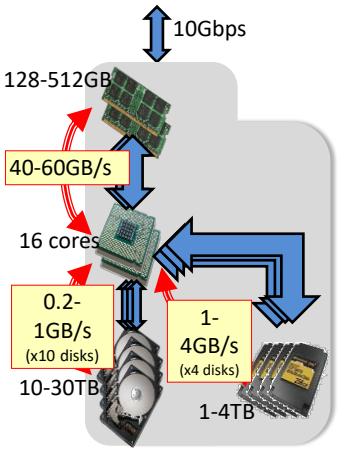
13



14

Support Interactive and Streaming Comp.

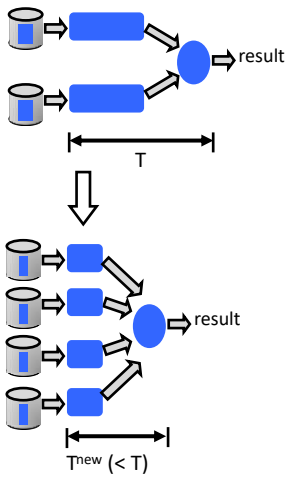
- Aggressive use of ***memory***
- Why?
  1. Memory transfer rates >> disk or SSDs
  2. Many datasets already fit into memory
     - Inputs of over 90% of jobs in Facebook, Yahoo!, and Bing clusters fit into memory
     - e.g., 1TB = 1 billion records @ 1KB each
  3. Memory density (still) grows with Moore's law
     - RAM/SSD hybrid memories at horizon



High end datacenter node

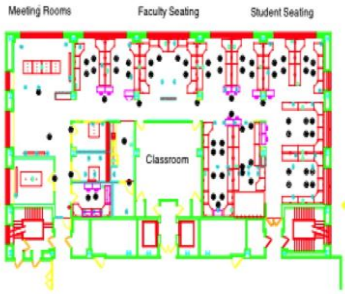
Support Interactive and Streaming Comp.

- Increase ***parallelism***
- Why?
  - Reduce work per node → improve latency
- Techniques:
  - Low latency parallel scheduler that achieve high locality
  - Optimized parallel communication patterns (e.g., shuffle, broadcast)
  - Efficient recovery from failures and straggler mitigation



Berkeley AMPLab

- "Launched" January 2011: 6 Year Plan
- 8 CS Faculty
- ~40 students
- 3 software engineers
- Organized for collaboration:



Berkeley AMPLab

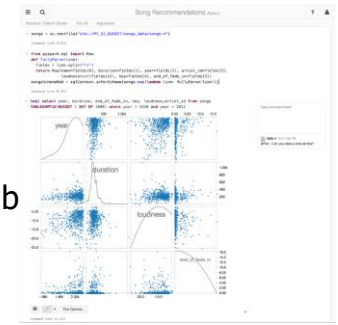
- Funding:
  - XData, CISE Expedition Grant
  - Industrial, founding sponsors
  - 18 other sponsors, including

Goal: Next Generation of Analytics Data Stack for Industry & Research:
• Berkeley Data Analytics Stack (BDAS)
• Release as Open Source

Databricks

databricks
making big data simple

- Founded in late 2013
- by the creators of Apache Spark
- Original team from UC Berkeley AMPLab
- Raised $47 Million in 2 rounds

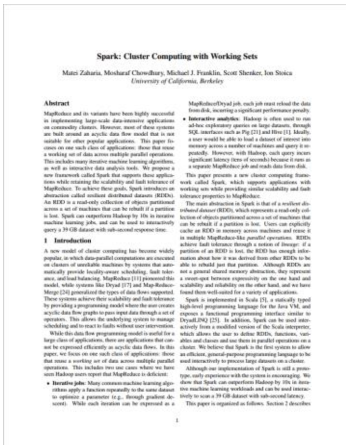


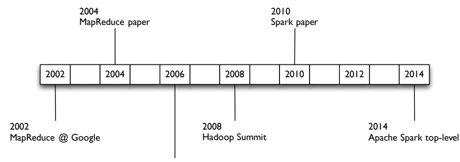
Databricks Cloud:
"A unified platform for building Big Data pipelines – from ETL to Exploration and Dashboards, to Advanced Analytics and Data Products."

The Databricks team contributed more than **75%** of the code added to Spark in the 2014

databricks

History of Spark


Spark: Cluster Computing with Working Sets
Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica
University of California, Berkeley



Spark: Cluster Computing with Working Sets
Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica
USENIX HotCloud (2010)
people.csail.mit.edu/matei/papers/2010/hotcloud_spark.pdf

Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing
Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica NSDI (2012)
usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf

History of Spark


Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing
Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica
University of California, Berkeley

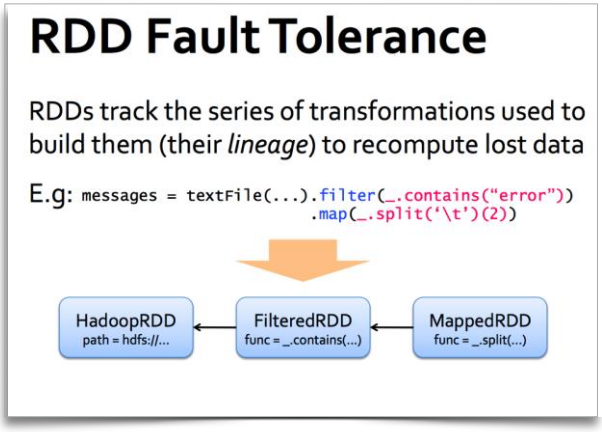
"We present Resilient Distributed Datasets (RDDs), a distributed memory abstraction that lets programmers perform in-memory computations on large clusters in a fault-tolerant manner.

RDDs are motivated by two types of applications that current computing frameworks handle inefficiently: iterative algorithms and interactive data mining tools.

In both cases, keeping data in memory can improve performance by an order of magnitude."

April 2012

History of Spark



The State of Spark, and Where We're Going Next
Matei Zaharia
Spark Summit (2013)
youtu.be/nU6vO2EJAb4

History of Spark

Spark STREAMING

Analyze real time streams of data in ½ second intervals

www.cs.berkeley.edu/~matei/papers/2013/sosp_spark_streaming.pdf

Discretized Streams: Fault-Tolerant Streaming Computation at Scale

Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, Ion Stoica
University of California, Berkeley

Abstract

Many "big data" applications must act on data in real time. Running these applications at ever-larger scales requires parallel platforms that automatically handle faults and stragglers. Unfortunately, current distributed stream processing models provide fault recovery in an expensive manner, requiring hot replication or long recovery times, and do not handle stragglers. We propose a new processing model, *discretized streams* (D-Streams), that overcomes these challenges. D-Streams enable a *parallel recovery* mechanism that improves efficiency over traditional replication and backup schemes, and tolerates stragglers. We show that they support a rich set of operators while attaining high per-node throughput similar to single-node systems, linear scaling to 100 nodes, sub-second latency, and sub-second fault recovery. Finally, D-Streams can easily be composed with batch and interactive query models like MapReduce, enabling rich applications that combine these modes. We implement D-Streams in a system called Spark Streaming.

1 Introduction

Much of "big data" is received in real time, and is most valuable at its time of arrival. For example, a social network may wish to detect trending conversation topics in *faults* and *stragglers* (slow nodes). Both problems are inevitable in large clusters [12], so streaming applications must recover from them quickly. Fast recovery is even more important in streaming than it was in batch jobs: while a 30 second delay to recover from a fault or straggler is a nuisance in a batch setting, it can mean losing the chance to make a key decision in a streaming setting.

Unfortunately, existing streaming systems have limited fault and straggler tolerance. Most distributed streaming systems, including Storm [37], TimeStream [33], MapReduce Online [11], and streaming databases [5, 9, 10], are based on a continuous operator model, in which long-running, stateful operators receive each record, update internal state, and send new records. While this model is quite natural, it makes it difficult to handle faults and stragglers.

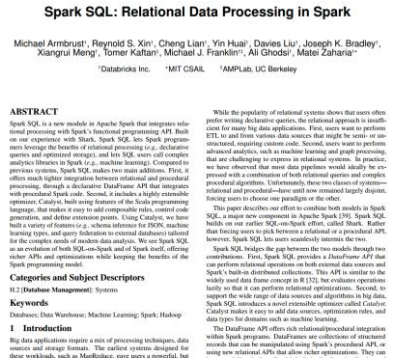
Specifically, given the continuous operator model, systems perform recovery through two approaches [20]: *replication*, where there are two copies of each node [5, 34], or *upstream backup*, where nodes buffer sent messages and replay them to a new copy of a failed node [33, 11, 37]. Neither approach is attractive in large clusters: replication costs 2× the hardware, while upstream backup takes a long time to recover, as the whole system must wait for a new node to serially rebuild the failed

```
TwitterUtils.createStream(...)
    .filter(_.getText.contains("Spark"))
    .countByWindow(Seconds(5))
```

History of Spark

Spark SQL

Seemlessly mix SQL queries with Spark programs.



Spark SQL: Relational Data Processing in Spark

Michael Armbrust[1], Reynold S. Xin[2], Cheng Lian[1], Yin Huai[1], Davies Liu[1], Joseph K. Bradley[1], Xiangrui Meng[1], Tomer Kaftan[3], Michael J. Franklin[2,1], Ali Ghodsi[1], Matei Zaharia[1,2]

[1]Databricks Inc. [2]MIT CSAIL [3]AMPLab, UC Berkeley

ABSTRACT

Spark SQL is a new module in Apache Spark that integrates relational processing with Spark's functional programming API. Built on our experience with Shark, Spark SQL lets Spark programmers leverage the benefits of relational processing (e.g., declarative queries and optimized storage), and lets SQL users call complex analytics libraries in Spark (e.g., machine learning). Compared to previous systems, Spark SQL makes two main additions. First, it offers much tighter integration between relational and procedural processing, through a declarative DataFrame API that integrates with procedural Spark code. Second, it includes a highly extensible optimizer, Catalyst, built using features of the Scala programming language, that makes it easy to add composable rules, control code generation, and define extension points. Using Catalyst, we have built a variety of features (e.g., schema inference for JSON, machine learning types, and query federation to external databases) tailored for the complex needs of modern data analysis. We see Spark SQL as an evolution of both SQL-on-Spark and of Spark itself, offering richer APIs and optimizations while keeping the benefits of the Spark programming model.

Categories and Subject Descriptors

H.2 [Database Management]: Systems

Keywords

Databases; Data Warehouse; Machine Learning; Spark; Hadoop

1 Introduction

Big data applications require a mix of processing techniques, data sources and storage formats. The earliest systems designed for these workloads, such as MapReduce, gave users a powerful, but

While the popularity of relational systems shows that users often prefer writing declarative queries, the relational approach is insufficient for many big data applications. First, users want to perform ETL to and from various data sources that might be semi- or unstructured, requiring custom code. Second, users want to perform advanced analytics, such as machine learning and graph processing, that are challenging to express in relational systems. In practice, we have observed that most data pipelines would ideally be expressed with a combination of both relational queries and complex procedural algorithms. Unfortunately, these two classes of systems—relational and procedural—have until now remained largely disjoint, forcing users to choose one paradigm or the other.

This paper describes our effort to combine both models in Spark SQL, a major new component in Apache Spark [39]. Spark SQL builds on our earlier SQL-on-Spark effort, called Shark. Rather than forcing users to pick between a relational or a procedural API, however, Spark SQL lets users seamlessly intermix the two.

Spark SQL bridges the gap between the two models through two contributions. First, Spark SQL provides a *DataFrame API* that can perform relational operations on both external data sources and Spark's built-in distributed collections. This API is similar to the widely used data frame concept in R [32], but evaluates operations lazily so that it can perform relational optimizations. Second, to support the wide range of data sources and algorithms in big data, Spark SQL introduces a novel extensible optimizer called Catalyst. Catalyst makes it easy to add data sources, optimization rules, and data types for domains such as machine learning.

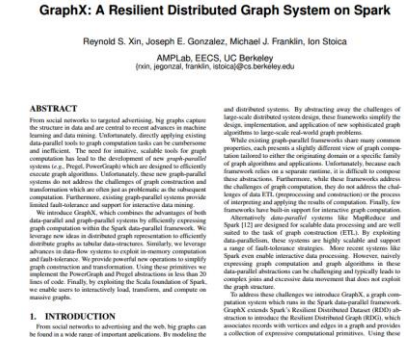
The DataFrame API offers rich relational/procedural integration within Spark programs. DataFrames are collections of structured records that can be manipulated using Spark's procedural API, or using new relational APIs that allow richer optimizations. They can

```
sqlCtx = new HiveContext(sc)
results = sqlCtx.sql(
  "SELECT * FROM people")
names = results.map(lambda p:
p.name)
```


History of Spark

Spark GRAPHX

Analyze networks of nodes and edges using graph processing



GraphX: A Resilient Distributed Graph System on Spark

Reynold S. Xin, Joseph E. Gonzalez, Michael J. Franklin, Ion Stoica

AMPLab, EECS, UC Berkeley
(rxin, jegonzal, franklin, istoica)@cs.berkeley.edu

ABSTRACT

From social networks to targeted advertising, big graphs capture the structure in data and are central to recent advances in machine learning and data mining. Unfortunately, directly applying existing data-parallel tools to graph computation tasks can be cumbersome and inefficient. The need for intuitive, scalable tools for graph computation has lead to the development of new *graph-parallel* systems (e.g., Pregel, PowerGraph) which are designed to efficiently execute graph algorithms. Unfortunately, these new graph-parallel systems do not address the challenges of graph construction and transformation which are often just as problematic as the subsequent computation. Furthermore, existing graph-parallel systems provide limited fault-tolerance and support for interactive data mining.

We introduce GraphX, which combines the advantages of both data-parallel and graph-parallel systems by efficiently expressing graph computation within the Spark data-parallel framework. We leverage new ideas in distributed graph representation to efficiently distribute graphs as tabular data-structures. Similarly, we leverage advances in data-flow systems to exploit in-memory computation and fault-tolerance. We provide powerful new operations to simplify graph construction and transformation. Using these primitives we implement the PowerGraph and Pregel abstractions in less than 20 lines of code. Finally, by exploiting the Scala foundation of Spark, we enable users to interactively load, transform, and compute on massive graphs.

1. INTRODUCTION

From social networks to advertising and the web, big graphs can be found in a wide range of important applications. By modeling the

and distributed systems. By abstracting away the challenges of large-scale distributed system design, these frameworks simplify the design, implementation, and application of new sophisticated graph algorithms to large-scale real-world graph problems.

While existing graph-parallel frameworks share many common properties, each presents a slightly different view of graph computation tailored to either the originating domain or a specific family of graph algorithms and applications. Unfortunately, because each framework relies on a separate runtime, it is difficult to compose these abstractions. Furthermore, while these frameworks address the challenges of graph computation, they do not address the challenges of data ETL (preprocessing and construction) or the process of interpreting and applying the results of computation. Finally, few frameworks have built-in support for interactive graph computation.

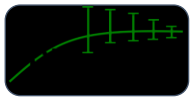
Alternatively *data-parallel* systems like MapReduce and Spark [12] are designed for scalable data processing and are well suited to the task of graph construction (ETL). By exploiting data-parallelism, these systems are highly scalable and support a range of fault-tolerance strategies. More recent systems like Spark even enable interactive data processing. However, naïvely expressing graph computation and graph algorithms in these data-parallel abstractions can be challenging and typically leads to complex joins and excessive data movement that does not exploit the graph structure.

To address these challenges we introduce GraphX, a graph computation system which runs in the Spark data-parallel framework. GraphX extends Spark's Resilient Distributed Dataset (RDD) abstraction to introduce the Resilient Distributed Graph (RDG), which associates records with vertices and edges in a graph and provides a collection of expressive computational primitives. Using these

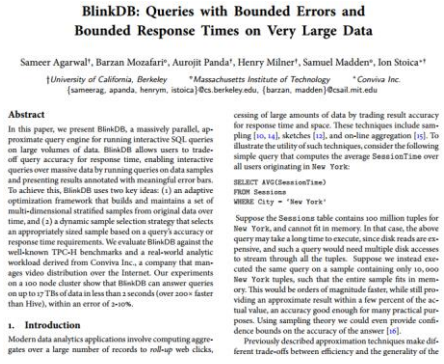
```
graph = Graph(vertices, edges)
messages =
spark.textFile("hdfs://...")
graph2 =
graph.joinVertices(messages) {
  (id, vertex, msg) => ...
}
```

https://amplab.cs.berkeley.edu/wp-content/uploads/2013/05/grades-graphx_with_fonts.pdf

History of Spark



SQL queries with Bounded Errors and Bounded Response Times



BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data

Sameer Agarwal[†], Barzan Mozafari[*], Aurojit Panda[†], Henry Milner[†], Samuel Madden[*], Ion Stoica[*†]

[†]*University of California, Berkeley* [*]*Massachusetts Institute of Technology* [*]*Conviva Inc.*

{sameerag, apanda, henrym, istoica}@cs.berkeley.edu, {barzan, madden}@csail.mit.edu

Abstract

In this paper, we present BlinkDB, a massively parallel, approximate query engine for running interactive SQL queries on large volumes of data. BlinkDB allows users to trade-off query accuracy for response time, enabling interactive queries over massive data by running queries on data samples and presenting results annotated with meaningful error bars. To achieve this, BlinkDB uses two key ideas: (1) an adaptive optimization framework that builds and maintains a set of multi-dimensional stratified samples from original data over time, and (2) a dynamic sample selection strategy that selects an appropriately sized sample based on a query's accuracy or response time requirements. We evaluate BlinkDB against the well-known TPC-H benchmarks and a real-world analytic workload derived from Conviva Inc., a company that manages video distribution over the Internet. Our experiments on a 100 node cluster show that BlinkDB can answer queries on up to 17 TBs of data in less than 2 seconds (over 200× faster than Hive), within an error of 2-10%.

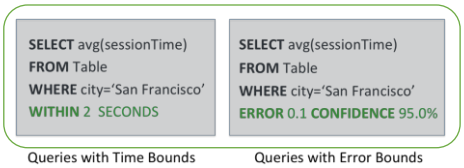
1. Introduction

Modern data analytics applications involve computing aggregates over a large number of records to *roll-up* web clicks, cessing of large amounts of data by trading result accuracy for response time and space. These techniques include sampling [10, 14], sketches [12], and on-line aggregation [15]. To illustrate the utility of such techniques, consider the following simple query that computes the average `SessionTime` over all users originating in `New York`:

```
SELECT AVG(SessionTime)
FROM Sessions
WHERE City = 'New York'
```

Suppose the `Sessions` table contains 100 million tuples for `New York`, and cannot fit in memory. In that case, the above query may take a long time to execute, since disk reads are expensive, and such a query would need multiple disk accesses to stream through all the tuples. Suppose we instead executed the same query on a sample containing only 10,000 `New York` tuples, such that the entire sample fits in memory. This would be orders of magnitude faster, while still providing an approximate result within a few percent of the actual value, an accuracy good enough for many practical purposes. Using sampling theory we could even provide confidence bounds on the accuracy of the answer [16].

Previously described approximation techniques make different trade-offs between efficiency and the generality of the



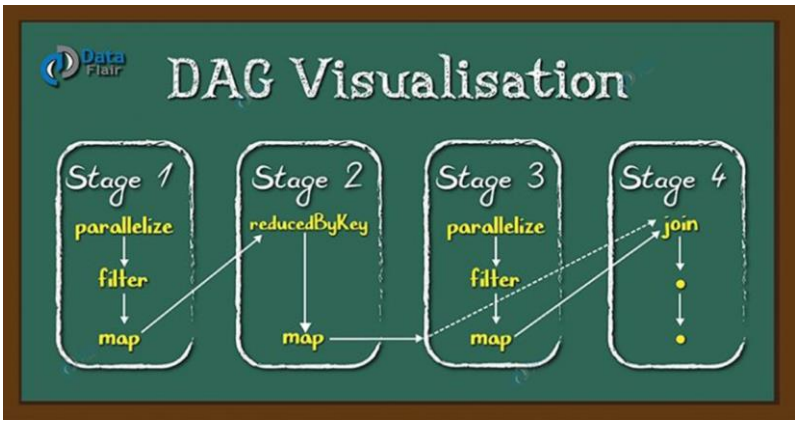
Queries with Time Bounds | Queries with Error Bounds

https://www.cs.berkeley.edu/~sameerag/blinkdb_eurosys13.pdf

History of Spark

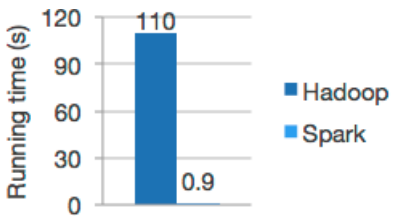
- Unlike the various specialized systems, Spark's goal was to *generalize* MapReduce to support new apps within same engine
- Two reasonably small additions are enough to express the previous models:
  - *fast data sharing*
  - *general DAGs*
- This allows for an approach which is more efficient for the engine, and much simpler for the end users

Directed Acyclic Graph - DAG



What is Apache Spark

- Spark is a unified analytics engine for large-scale data processing
- Speed: run workloads 100x faster
  - High performance for both batch and streaming data
  - Computations run in memory



Logistic regression in Hadoop and Spark

What is Apache Spark

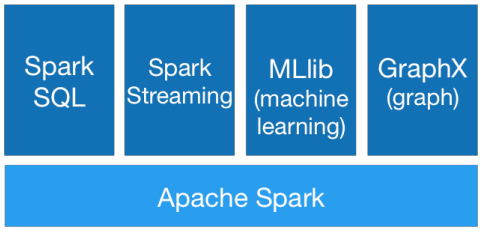
- Ease of Use: write applications quickly in Java, Scala, Python, R, SQL
  - Offer over 80 high-level operators
  - Use them interactively form Scala, Python, R, and SQL

```
df = spark.read.json("logs.json")
df.where("age > 21")
select("name.first").show()
```

Spark's Python DataFrame API
Read JSON files with automatic schema inference

What is Apache Spark

- Generality: combine SQL, Streaming, and complex analytics
  - Provide libraries including SQL and DataFrames, Spark Streaming, MLib, GraphX
  - Wide range of workloads e.g., batch applications, interactive algorithms, interactive queries, streaming



What is Apache Spark

- Run Everywhere:
  - run on Hadoop, Apache Mesos, Kubernetes, standalone or in the cloud.
  - access data in HDFS, Aluxio, Apache Cassandra, Apache Hbase, Apache Hive, etc.

Comparison between Hadoop and Spark

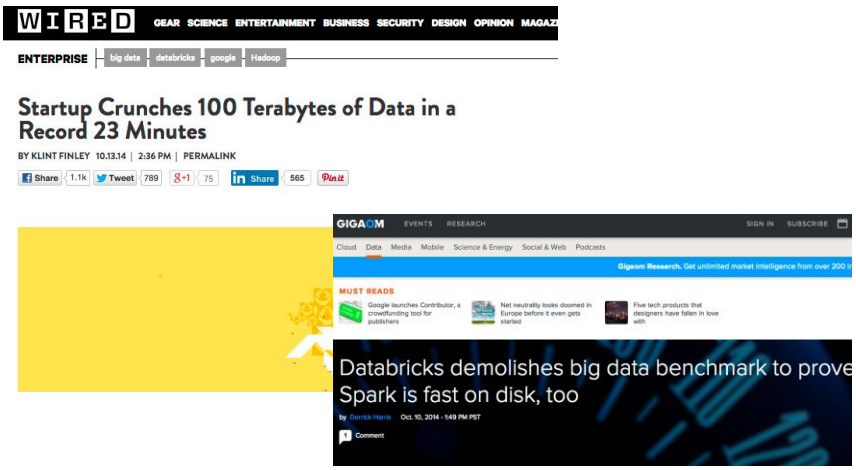
	hadoop MapReduce	Spark
Strengths	▪ Can collect any data ▪ Limitless in size	▪ Can work off any Hadoop collection ▪ Runs on Hadoop, or other clusters ▪ In-memory processing makes it very fast ▪ Supports Java, Scala, Python, and R*, and can be used with SQL.
Used for	▪ Initial data ingestion ▪ Data curation ▪ Large-scale "boil the ocean" analytics ▪ Data archiving	▪ Complex query processing of large amounts of data quickly ▪ Can handle ad hoc queries
Limitations	▪ MapReduce is hard to program ▪ Disk-based batch nature limits speed, agility.	▪ Limited only by processor speed, available memory, cores, and cluster size.

100TB Daytona Sort Competition

	Hadoop MR Record	Spark Record	Spark 1 PB
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**	**22.5 GB/min**

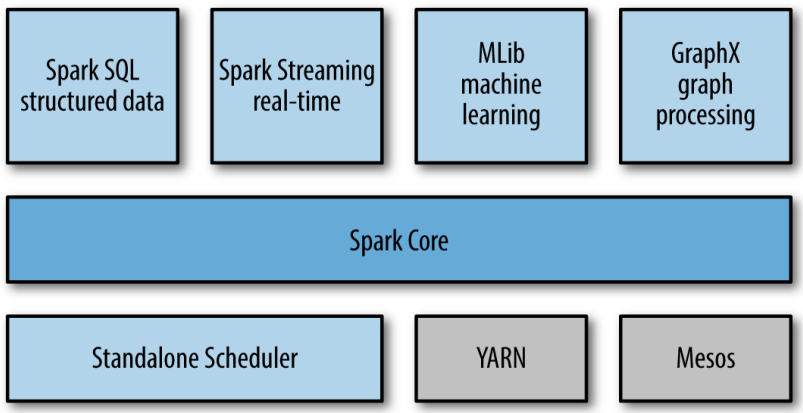
Spark sorted the same data **3X faster** using **10X fewer machines** than Hadoop MR in 2013.

All the sorting took place on disk (HDFS) without using Spark's in-memory cache!



Components of Stack

The Spark stack



The Spark stack

- Spark Core:
  - contain basic functionality of Spark including task scheduling, memory management, fault recovery, etc.
  - provide APIs for building and manipulating RDDs
- SparkSQL
  - allow querying structured data via SQL, Hive Query Language
  - allow combining SQL queries and data manipulations in Python, Java, Scala

The Spark stack

- Spark Streaming: enables processing of live streams of data via APIs
- Mlib:
  - contain common machine language functionality
  - provide multiple types of algorithms: classification, regression, clustering, etc.
- GraphX:
  - library for manipulating graphs and performing graph-parallel computations
  - extend Spark RDD API

The Spark stack

- Cluster Managers
  - Hadoop Yarn
  - Apache Mesos, and
  - Standalone Schedular (simple manager in Spark).

Resilient Distributed Dataset – RDD

- RDD Basics
- Creating RDDs
- RDD Operations
- Common Transformation and Actions
- Persistence (Caching)

RDD Basics

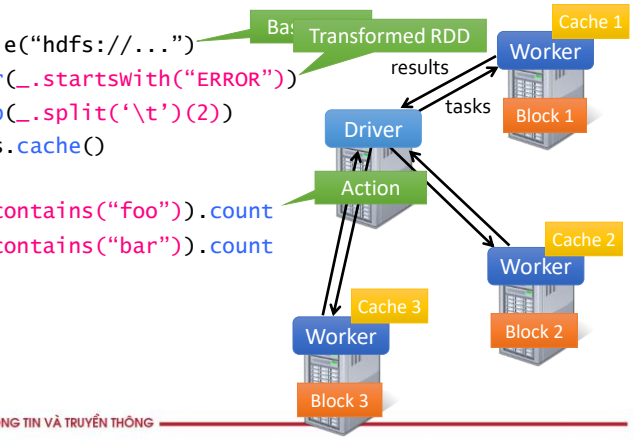
- RDD:
  - Immutable distributed collection of objects
  - Split into multiple partitions => can be computed on different nodes
- All work in Spark is expressed as
  - creating new RDDs
  - transforming existing RDDs
  - calling actions on RDDs

Example

Load error messages from a log into memory, then interactively search for various patterns

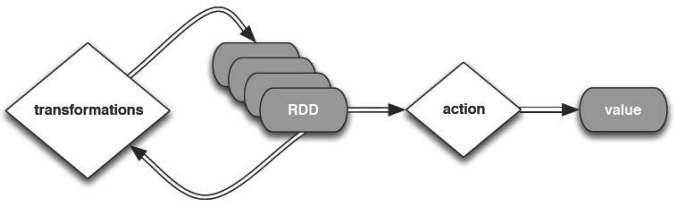
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(_.startsWith("ERROR"))
messages = errors.map(_.split('\t')(2))
cachedMsgs = messages.cache()

cachedMsgs.filter(_.contains("foo")).count
cachedMsgs.filter(_.contains("bar")).count
. . .
```



RDD Basics

- Two types of operations: transformations and actions
- Transformations: construct a new RDD from a previous one e.g., filter data
- Actions: compute a result base on an RDD e.g., count elements, get first element



45

Transformations

- Create new RDDs from existing RDDs
- Lazy evaluation
  - See the whole chain of transformations
  - Compute just the data needed
- Persist contents:
  - persist an RDD in memory to reuse it in future
  - persist RDDs on disk is possible

46

Typical works of a Spark program

1. Create some input RDDs form external data
2. Transform them to define new RDDs using transformations like filter()
3. Ask Spark to persist() any intermediate RDDs that will need to be reused
4. Launch actions such as count(), first() to kick off a parallel computation

Resilient Distributed Dataset – RDD

- RDD Basics
- Creating RDDs
- RDD Operations
- Common Transformation and Actions
- Persistence (Caching)

Two ways to create RDDs

1. Parallelizing a collection: uses parallelize()

- Python

```
lines = sc.parallelize(["pandas", "i like pandas"])
```

- Scala

```
val lines = sc.parallelize(List("pandas", "i like pandas"))
```

- Java

```
JavaRDD<String> lines = sc.parallelize(Arrays.asList("pandas", "i like pandas"));
```


Two ways to create RDDs

2. Loading data from external storage

- Python

```
lines = sc.textFile("/path/to/README.md")
```

- Scala

```
val lines = sc.textFile("/path/to/README.md")
```

- Java

```
JavaRDD<String> lines = sc.textFile("/path/to/README.md");
```

Resilient Distributed Dataset – RDD

- RDD Basics
- Creating RDDs
- RDD Operations
- Common Transformation and Actions
- Persistence (Caching)

RDD Operations

- Two types of operations
  - Transformations: operations that return a new RDDs e.g., map(), filter()
  - Actions: operations that return a result to the driver program or write it to storage such as count(), first()
- Treated differently by Spark
  - Transformation: lazy evaluation
  - Action: execution at any time

Transformation

- Example 1. Use filter()
- Python

```
inputRDD = sc.textFile("log.txt")
errorsRDD = inputRDD.filter(lambda x: "error" in x)
```

- Scala

```
val inputRDD = sc.textFile("log.txt")
val errorsRDD = inputRDD.filter(line =>
line.contains("error"))
```

- Java

```
JavaRDD<String> inputRDD = sc.textFile("log.txt");
JavaRDD<String> errorsRDD = inputRDD.filter(
new Function<String, Boolean>() {
public Boolean call(String x) {
        return x.contains("error"); }}
  });
```


Transformation

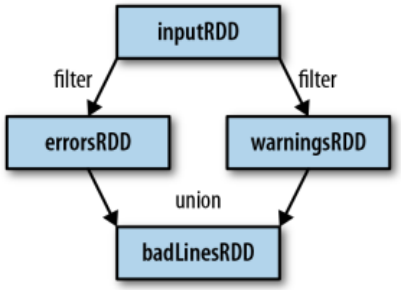
- filter()
  - does not change the existing *inputRDD*
  - returns a pointer to an entirely new RDD
  - *inputRDD* still can be reused
- union()

```
errorsRDD = inputRDD.filter(lambda x: "error" in x)
warningsRDD=inputRDD.filter(lambda x: "warning" in x)
badLinesRDD = errorsRDD.union(warningsRDD)
```

- transformations can operate on any number of input RDDs

Transformation

- Spark keeps track dependencies between RDDs, called the lineage graph
- Allow recovering lost data



Actions

- Example. count the number of errors
- Python

```
print "Input had " + badLinesRDD.count() + " concerning lines"
print "Here are 10 examples:"
for line in badLinesRDD.take(10):
print line
```

- Scala

```
println("Input had " + badLinesRDD.count() + " concerning lines")
println("Here are 10 examples:")
badLinesRDD.take(10).foreach(println)
```

- Java

```
System.out.println("Input had " + badLinesRDD.count() + " concerning
lines")
System.out.println("Here are 10 examples:")
for (String line: badLinesRDD.take(10)) {
System.out.println(line);
}
```

Resilient Distributed Dataset – RDD

- RDD Basics
- Creating RDDs
- RDD Operations
- Common Transformation and Actions
- Persistence (Caching)

RDD Basics

Transformations	Actions
map flatMap filter sample union groupByKey reduceByKey join cache …	reduce collect count save lookupKey …

Transformations

transformation	*description*
map(*func*)	return a new distributed dataset formed by passing each element of the source through a function *func*
filter(*func*)	return a new dataset formed by selecting those elements of the source on which *func* returns true
flatMap(*func*)	similar to map, but each input item can be mapped to 0 or more output items (so *func* should return a Seq rather than a single item)
sample(*withReplacement*, *fraction*, *seed*)	sample a fraction *fraction* of the data, with or without replacement, using a given random number generator *seed*
union(*otherDataset*)	return a new dataset that contains the union of the elements in the source dataset and the argument
distinct([*numTasks*]))	return a new dataset that contains the distinct elements of the source dataset

Transformations

transformation	*description*
groupByKey([*numTasks*])	when called on a dataset of `(K, V)` pairs, returns a dataset of `(K, Seq[V])` pairs
reduceByKey(*func*, [*numTasks*])	when called on a dataset of `(K, V)` pairs, returns a dataset of `(K, V)` pairs where the values for each key are aggregated using the given reduce function
sortByKey([*ascending*], [*numTasks*])	when called on a dataset of `(K, V)` pairs where `K` implements `Ordered`, returns a dataset of `(K, V)` pairs sorted by keys in ascending or descending order, as specified in the boolean ascending argument
join(*otherDataset*, [*numTasks*])	when called on datasets of type `(K, V)` and `(K, W)`, returns a dataset of `(K, (V, W))` pairs with all pairs of elements for each key
cogroup(*otherDataset*, [*numTasks*])	when called on datasets of type `(K, V)` and `(K, W)`, returns a dataset of `(K, Seq[V], Seq[W])` tuples – also called `groupWith`
cartesian(*otherDataset*)	when called on datasets of types `T` and `U`, returns a dataset of `(T, U)` pairs (all pairs of elements)

Actions

action	description
reduce(*func*)	aggregate the elements of the dataset using a function *func* (which takes two arguments and returns one), and should also be commutative and associative so that it can be computed correctly in parallel
collect()	return all the elements of the dataset as an array at the driver program – usually useful after a filter or other operation that returns a sufficiently small subset of the data
count()	return the number of elements in the dataset
first()	return the first element of the dataset – similar to *take(1)*
take(*n*)	return an array with the first *n* elements of the dataset – currently not executed in parallel, instead the driver program computes all the elements
takeSample(*withReplacement, fraction, seed*)	return an array with a random sample of *num* elements of the dataset, with or without replacement, using the given random number generator seed

Actions

action	description
saveAsTextFile(*path*)	write the elements of the dataset as a text file (or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system. Spark will call `toString` on each element to convert it to a line of text in the file
saveAsSequenceFile(*path*)	write the elements of the dataset as a Hadoop `SequenceFile` in a given path in the local filesystem, HDFS or any other Hadoop-supported file system. Only available on RDDs of key-value pairs that either implement Hadoop's `Writable` interface or are implicitly convertible to `Writable` (Spark includes conversions for basic types like `Int`, `Double`, `String`, etc).
countByKey()	only available on RDDs of type `(K, V)`. Returns a `Map` of `(K, Int)` pairs with the count of each key
foreach(*func*)	run a function *func* on each element of the dataset – usually done for side effects such as updating an accumulator variable or interacting with external storage systems

Resilient Distributed Dataset – RDD

- RDD Basics
- Creating RDDs
- RDD Operations
- Common Transformation and Actions
- Persistence (Caching)

Persistence levels

Level	Space used	CPU time	In memory	On disk	Comments
MEMORY_ONLY	High	Low	Y	N	
MEMORY_ONLY_SER	Low	High	Y	N	
MEMORY_AND_DISK	High	Medium	Some	Some	Spills to disk if there is too much data to fit in memory.
MEMORY_AND_DISK_SER	Low	High	Some	Some	Spills to disk if there is too much data to fit in memory. Stores serialized representation in memory.
DISK_ONLY	Low	High	N	Y	

Persistence

- Example

```
val result = input.map(x => x * x)
result.persist(StorageLevel.DISK_ONLY)
println(result.count())
println(result.collect().mkString(","))
```


Acknowledgement and References

Books:

- Holden Karau, Andy Konwinski, Patrick Wendell & Matei Zaharia. Learning Spark. Oreilly
- TutorialsPoint. Spark Core Programming

Slides:

- Paco Nathan. Intro to Apache Spark
- Harold Liu. Berkely Data Analytics Stack
- DataBricks. Intro to Spark Development

Q&A

67

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

68

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!

soict.hust.edu.vn/ fb.com/groups/soict

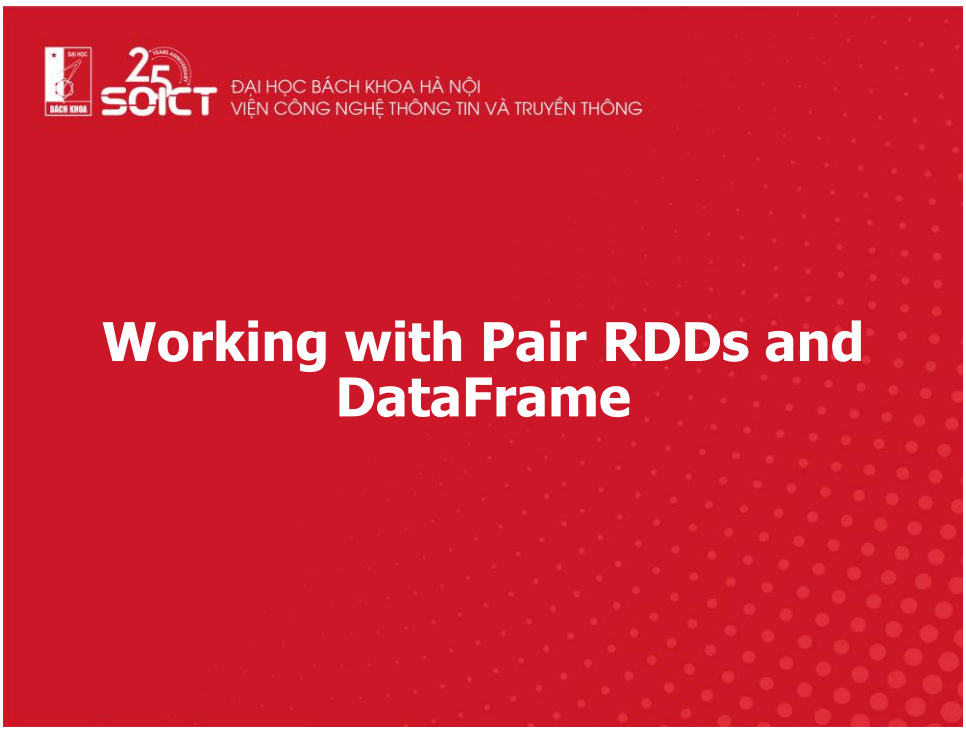
69

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

70


ĐẠI HỌC
BÁCH KHOA
25 YEARS ANNIVERSARY
SOICT
ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

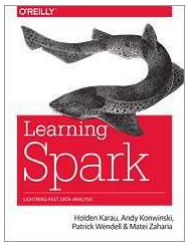
1


ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
Working with Pair RDDs and DataFrame

2

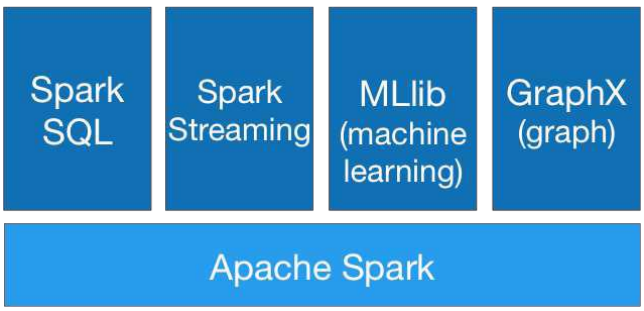
From where to learn Spark ?

http://spark.apache.org/



http://shop.oreilly.com/product/0636920028512.do

Spark architecture



Easy ways to run Spark ?

- ★ your IDE (ex. Eclipse or IDEA)
- ★ Standalone Deploy Mode: simplest way to deploy Spark on a single machine
- ★ Docker & Zeppelin
- ★ EMR
- ★ Hadoop vendors (Cloudera, Hortonworks)

Supported languages

RDD (Resilient Distributed Dataset)

RDD (Resilient Distributed Dataset)

- Resilient: If data in memory is lost, it can be recreated
- Distributed: Processed across the cluster
- Dataset: Initial data can come from a source such as a file, or it can be created programmatically

- **RDDs are the fundamental unit of data in Spark**
- **Most Spark programming consists of performing operations on RDDs**

Creating RDD (I)

- Python
- lines = sc.parallelize([“workshop”, “spark”])

- Scala
- val lines = sc.parallelize(List(“workshop”, “spark”))

- Java
- JavaRDD<String> lines = sc.parallelize(Arrays.asList(“workshop”, “spark”))

Creating RDD (II)

```
Python
lines = sc.textFile("/path/to/file.txt")

Scala
val lines = sc.textFile("/path/to/file.txt")

Java
JavaRDD<String> lines = sc.textFile("/path/to/file.txt")
```


RDD persistence

```
MEMORY_ONLY
MEMORY_AND_DISK
MEMORY_ONLY_SER
MEMORY_AND_DISK_SER
DISK_ONLY
MEMORY_ONLY_2
MEMORY_AND_DISK_2
OFF_HEAP
```

Working with RDDs

RDDs

RDDs can hold any serializable type of element

- –Primitive types such as integers, characters, and booleans
- –Sequence types such as strings, lists, arrays, tuples, and dicts (including nested data types)
- –Scala/Java Objects (if serializable)
- –Mixed types

§ **Some RDDs are specialized and have additional functionality**

- –Pair RDDs
- –RDDs consisting of key-value pairs
- –Double RDDs
- –RDDs consisting of numeric data

Creating RDDs from Collections

You can create RDDs from collections instead of files
–sc.parallelize(*collection*)

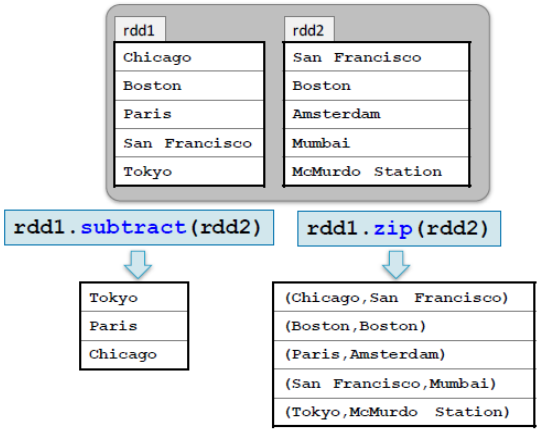
```
myData = ["Alice","Carlos","Frank","Barbara"]
> myRdd = sc.parallelize(myData)
> myRdd.take(2) ['Alice', 'Carlos']
```


Creating RDDs from Text Files (1)

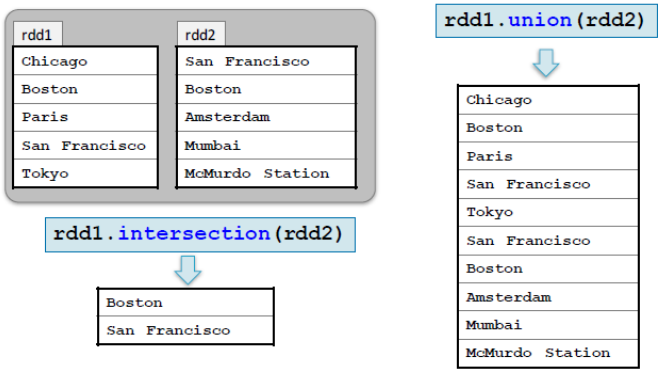
For file-based RDDs, use SparkContext.textFile
– Accepts a single file, a directory of files, a wildcard list of files, or a comma-separated list of files. Examples:
–sc.textFile("myfile.txt")
–sc.textFile("mydata/")
–sc.textFile("mydata/*.log")
–sc.textFile("myfile1.txt,myfile2.txt")
–Each line in each file is a separate record in the RDD

Files are referenced by absolute or relative URI
–Absolute URI:
–file:/home/training/myfile.txt
–hdfs://nnhost/loudacre/myfile.txt

Examples: Multi-RDD Transformations (1)



Examples: Multi-RDD Transformations (2)



Some Other General RDD Operations

Other RDD operations

- –first returns the first element of the RDD
- –foreach applies a function to each element in an RDD
- –top(*n*) returns the largest *n* elements using natural ordering

Sampling operations

- –sample creates a new RDD with a sampling of elements
- –takeSample returns an array of sampled elements

Other data structures in Spark

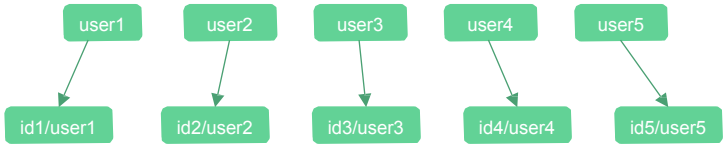
★ Paired RDD

★ DataFrame

★ DataSet

Paired RDD

Paired RDD = an RDD of key/value pairs



Pair RDDs

Pair RDDs

§ **Pair RDDs are a special form of RDD**
- –Each element must be a key-value pair (a two-element *tuple*)
- –Keys and values can be any type

§ **Why?**
- –Use with map-reduce algorithms
- –Many additional functions are available for common data processing needs
- –Such as sorting, joining, grouping, and counting

Pair RDD

(key1,value1)
(key2,value2)
(key3,value3)
...

Creating Pair RDDs

The first step in most workflows is to get the data into key/value form

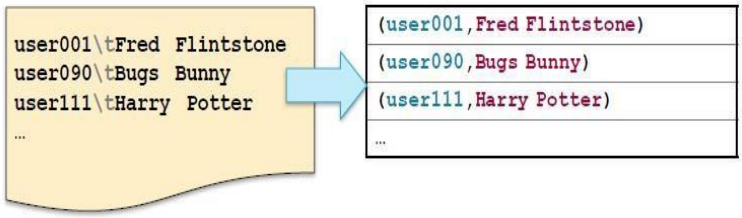
- –What should the RDD should be keyed on?
- –What is the value?

Commonly used functions to create pair RDDs
- **–map**
- **–flatMap / flatMapValues**
- **–keyBy**

Example: A Simple Pair RDD

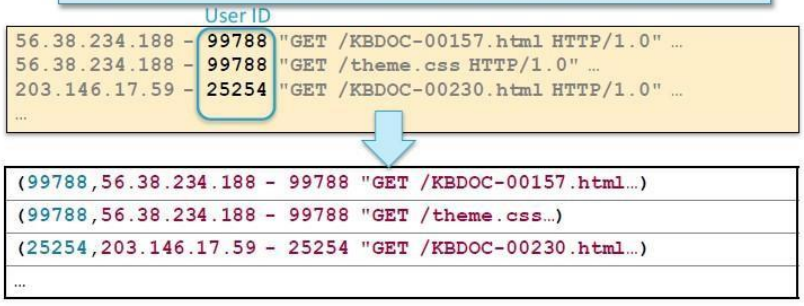
Example: Create a pair RDD from a tab-separated file

```
> val users = sc.textFile(file).
   map(line => line.split('\t').
   map(fields => (fields(0),fields(1)))
```



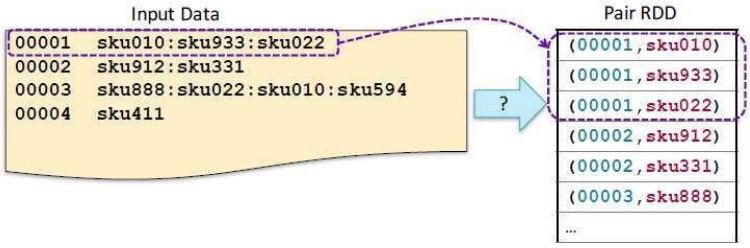
Example: Keying Web Logs by User ID

```
> sc.textFile(logfile).
    keyBy(line => line.split(' ')(2))
```



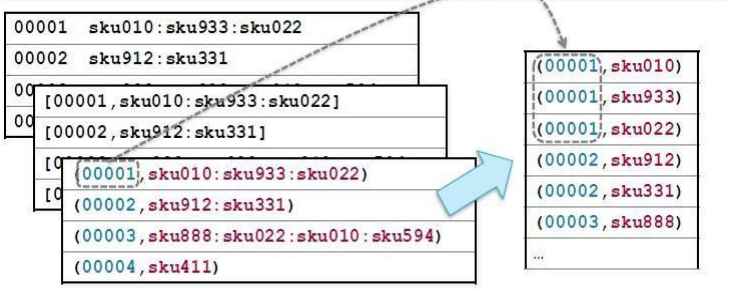
Mapping Single Rows to Multiple Pairs

- **How would you do this?**
  - Input: order numbers with a list of SKUs in the order
  - Output: order (key) and sku (value)



Answer : Mapping Single Rows to Multiple Pairs

```
> sc.textFile(file) \
  .map(lambda line: line.split('\t')) \
  .map(lambda fields: (fields[0],fields[1]))
  .flatMapValues(lambda skus: skus.split(':'))
```



Map-Reduce

§ **Map-reduce is a common programming model**
- –Easily applicable to distributed processing of large data sets

§ **Hadoop MapReduce is the major implementation**
- –Somewhat limited
- –Each job has one map phase, one reduce phase
- –Job output is saved to files

§ **Spark implements map-reduce with much greater flexibility**
- –Map and reduce functions can be interspersed
- –Results can be stored in memory
- –Operations can easily be chained

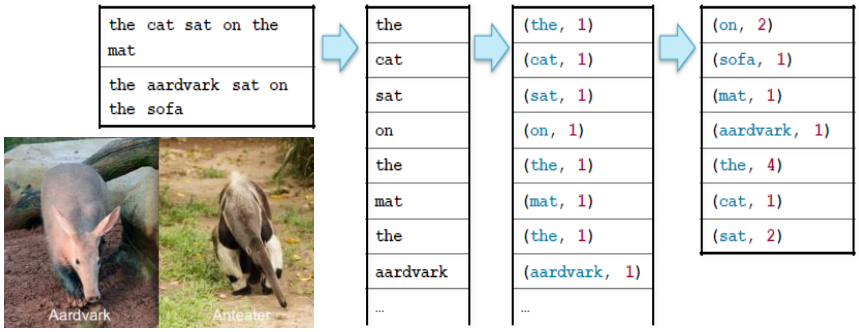
Map-Reduce in Spark

§ **Map-reduce in Spark works on pair RDDs**

§ **Map phase**
- –Operates on one record at a time
- –"Maps" each record to zero or more new records
- –Examples: **map, flatMap, filter, keyBy**

§ **Reduce phase**
- –Works on map output
- –Consolidates multiple records
- –Examples: **reduceByKey, sortByKey, mean**

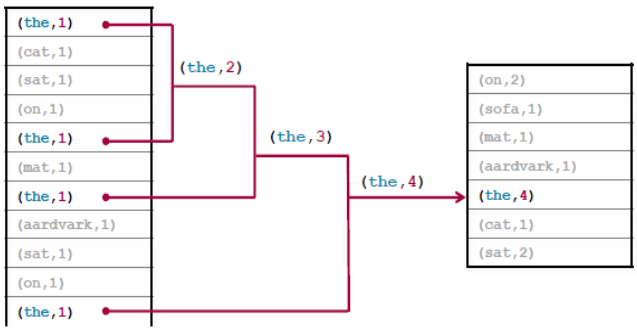
Example: Word Count



reduceByKey

The function passed to reduceByKey combines values from two keys

– Function must be binary



```
> val counts = sc.textFile(file). flat.Map
  (line => line.split(' ')). map(word => (word
  ,1)). reduceByKey((v1,v2) => v1+v2)
```

OR

```
> val counts = sc.textFile(file). flat.Map
  (_.split(' '),.
  map((_,1)).
  reduceByKey(_+_)
```

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

32

Pair RDD Operations

§ **In addition to map and reduceByKey operations, Spark has several operations specific to pair RDDs**

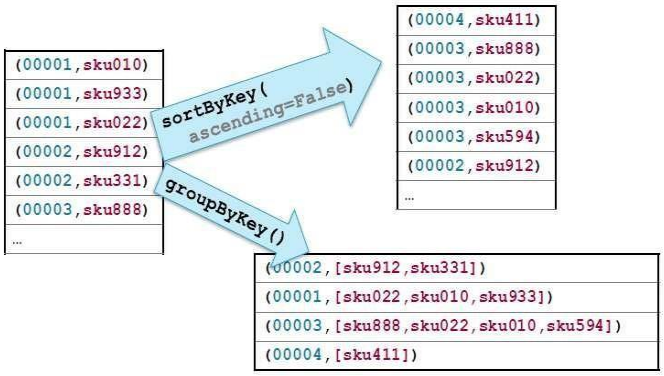
§ **Examples**

- –**countByKey** returns a map with the count of occurrences of each key
- –**groupByKey** groups all the values for each key in an RDD
- –**sortByKey** sorts in ascending or descending order
- –**join** returns an RDD containing all pairs with matching keys from two RDD

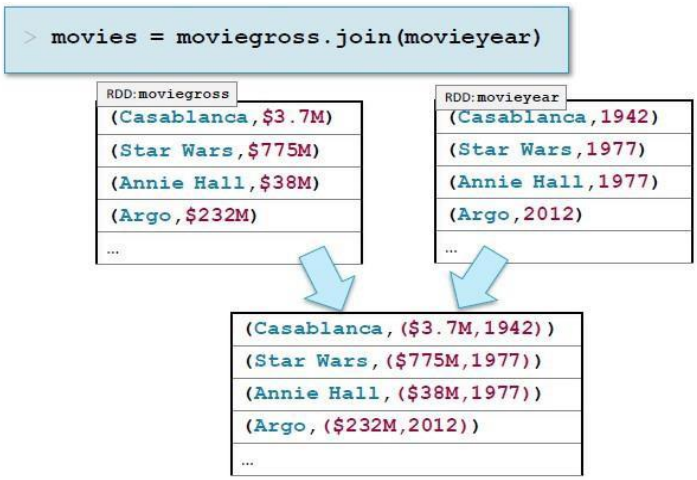
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

33

Example: Pair RDD Operations



Example: Joining by Key



Other Pair Operations

§ **Some other pair operations**

–**keys** returns an RDD of just the keys, without the values

–**values** returns an RDD of just the values, without keys

–**lookup(*key*)** returns the value(s) for a key

–**leftOuterJoin**, **rightOuterJoin** , **fullOuterJoin** join two RDDs, including keys defined in the left, right or either RDD respectively

–**mapValues**, **flatMapValues** execute a function on just the values, keeping the key the same

DataFrames and Apache Spark SQL

What is Spark SQL?

§ **What is Spark SQL?**
- –Spark module for structured data processing
- –Replaces Shark (a prior Spark module, now deprecated)
- –Built on top of core Spark

§ **What does Spark SQL provide?**
- –The DataFrame API—a library for working with data as tables
- –Defines DataFrames containing rows and columns
- –DataFrames are the focus of this chapter!
- –Catalyst Optimizer—an extensible optimization framework
- –A SQL engine and command line interface

SQL Context

§ **The main Spark SQL entry point is a SQL context object**
- –Requires a **SparkContext** object
- –The SQL context in Spark SQL is similar to Spark context in core Spark

§ **There are two implementations**
- –**SQLContext**
- –Basic implementation
- –**HiveContext**
- –Reads and writes Hive/HCatalog tables directly
- –Supports full HiveQL language
- –Requires the Spark application be linked with Hive libraries
- –Cloudera recommends using **HiveContext**

Creating a SQL Context

§ **The Spark shell creates a HiveContext instance automatically**
- –Call **sqlContext**
- –You will need to create one when writing a Spark application
- –Having multiple SQL context objects *is* allowed

§ **A SQL context object is created based on the Spark context**

Language: Scala

```
import org.apache.spark.sql.hive.HiveContext
val sqlContext = new HiveContext(sc)
import sqlContext.implicits._
```

40

DataFrames

§ **DataFrames are the main abstraction in Spark SQL**

- –Analogous to RDDs in core Spark
- –A distributed collection of structured data organized into Named columns
- –Built on a *base RDD* containing **Row** objects

41

Creating a DataFrame from a Data Source

§ **sqlContext.read returns a DataFrameReader object**

§ **DataFrameReader provides the functionality to load data into a DataFrame**

§ **Convenience functions**
- **–json(*filename*)**
- **–parquet(*filename*)**
- **–orc(*filename*)**
- **–table(*hive-tablename*)**
- **–jdbc(*url,table,options*)**

Example: Creating a DataFrame from a JSON File

Language: Scala

```
val sqlContext = new HiveContext(sc)
import sqlContext.implicits._
val peopleDF = sqlContext.read.json("people.json")
```

File: people.json

```
{"name":"Alice", "pcode":"94304"}
{"name":"Brayden", "age":30, "pcode":"94304"}
{"name":"Carla", "age":19, "pcode":"10036"}
{"name":"Diana", "age":46}
{"name":"Étienne", "pcode":"94104"}
```

age	name	pcode
null	Alice	94304
30	Brayden	94304
19	Carla	10036
46	Diana	null
null	Étienne	94104

Example: Creating a DataFrame from a Hive/Impala Table

Language: Scala

```
val sqlContext = new HiveContext(sc)
import sqlContext.implicits._
val customerDF = sqlContext.read.table("customers")
```

Table: customers

cust_id	name	country
001	Ani	us
002	Bob	ca
003	Carlos	mx
…	…	…

cust_id	name	country
001	Ani	us
002	Bob	ca
003	Carlos	mx
…	…	…

Loading from a Data Source Manually

§ **You can specify settings for the DataFrameReader**
–**format**: Specify a data source type
–**option**: A key/value setting for the underlying data source
–**schema**: Specify a schema instead of inferring from the data source
§ **Then call the generic base function load**

```
sqlContext.read.
   format("com.databricks.spark.avro").
   load("/loudacre/accounts_avro")
```

```
sqlContext.read.
   format("jdbc").
   option("url","jdbc:mysql://localhost/loudacre").
   option("dbtable","accounts").
   option("user","training").
   option("password","training").
   load()
```

Data Sources

§ **Spark SQL 1.6 built-in data source types**
- –**table**
- –**json**
- –**parquet**
- –**jdbc**
- –**orc**

§ **You can also use third party data source libraries, such as**
- –Avro (included in CDH)
- –HBase
- –CSV
- –MySQL
- –and more being added all the time

DataFrame Basic Operations

§ Basic operations deal with DataFrame metadata (rather than its data)

§ Some examples
- –schema returns a schema object describing the data
- –printSchema displays the schema as a visual tree
- –cache / persist persists the DataFrame to disk or memory
- –columns returns an array containing the names of the columns
- –dtypes returns an array of (column name,type) pairs
- –explain prints debug information about the DataFrame to the console

DataFrame Basic Operations

Language: Scala

```
> val peopleDF = sqlContext.read.json("people.json")
> peopleDF.dtypes.foreach(println)
(age,LongType)
(name,StringType)
(pcode,StringType)
```


DataFrame Actions

- Some DataFrame actions
  - –collect returns all rows as an array of Row objects
  - –take(*n*) returns the first n rows as an array of Row objects
  - –count returns the number of rows
  - –show(*n*)displays the first n rows (default=20)

Language: Scala

```
> peopleDF.count()
res7: Long = 5

> peopleDF.show(3)
age  name    pcode
null Alice   94304
30   Brayden 94304
19   Carla   10036
```

DataFrame Queries

§ **DataFrame query methods return new DataFrames**
– Queries can be chained like transformations
§ **Some query methods**
–**distinct** returns a new DataFrame with distinct elements of this DF
–**join** joins this DataFrame with a second DataFrame
– Variants for inside, outside, left, and right joins
–**limit** returns a new DataFrame with the first **n** rows of this DF
–**select** returns a new DataFrame with data from one or more columns of the base DataFrame
–**where** returns a new DataFrame with rows meeting specified query criteria (alias for **filter**)

DataFrame Query Strings

- **Some query operations take strings containing simple query expressions**
  - Such as `select` and `where`
- **Example: `select`**

age	name	pcode
null	Alice	94304
30	Brayden	94304
19	Carla	10036
46	Diana	null
null	Étienne	94104

`peopleDF.select("age")`

age
null
30
19
46
null

`peopleDF.select("name","age")`

name	age
Alice	null
Brayden	30
Carla	19
Diana	46
Étienne	null

Querying DataFrames using Columns

§ **Columns can be referenced in multiple ways**

age	name	pcode
null	Alice	94304
30	Brayden	94304
19	Carla	10036
46	Diana	null
null	Étienne	94104

▪ Scala

```
val ageDF = peopleDF.select(peopleDF("age"))
```

```
val ageDF = peopleDF.select($"age")
```

age
null
30
19
46
null

Joining DataFrames

§ **A basic inner join when join column is in both DataFrames**

age	name	pcode
null	Alice	94304
30	Brayden	94304
19	Carla	10036
46	Diana	null
null	Étienne	94104

pcode	city	state
10036	New York	NY
87501	Santa Fe	NM
94304	Palo Alto	CA
94104	San Francisco	CA

Language: Python/Scala

```
peopleDF.join(pcodesDF, "pcode")
```

pcode	age	name	city	state
94304	null	Alice	Palo Alto	CA
94304	30	Brayden	Palo Alto	CA
10036	19	Carla	New York	NY
94104	null	Étienne	San Francisco	CA

Joining DataFrames

- Specify type of join as `inner` (default), `outer`, `left_outer`, `right_outer`, or `leftsemi`

age	name	pcode
null	Alice	94304
30	Brayden	94304
19	Carla	10036
46	Diana	null
null	Étienne	94104

pcode	city	state
10036	New York	NY
87501	Santa Fe	NM
94304	Palo Alto	CA
94104	San Francisco	CA

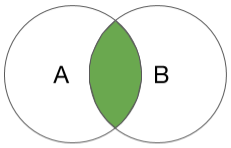
Language: Python

```
peopleDF.join(pcodesDF, "pcode",
"left_outer")
```

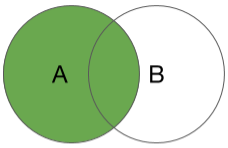
Language: Scala

```
peopleDF.join(pcodesDF,
Array("pcode"), "left_outer")
```

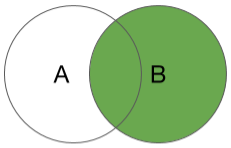
pcode	age	name	city	state
94304	null	Alice	Palo Alto	CA
94304	30	Brayden	Palo Alto	CA
10036	19	Carla	New York	NY
null	46	Diana	null	null
94104	null	Étienne	San Francisco	CA



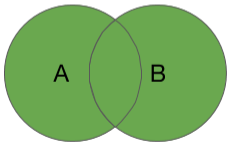
INNER JOIN



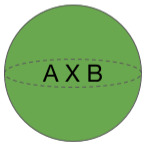
LEFT OUTER JOIN



RIGHT OUTER JOIN



FULL OUTER JOIN



CARTESIAN (CROSS) JOIN

SQL Queries

§ **When using HiveContext, you can query Hive/Impala tables using HiveQL**

– Returns a DataFrame

Language: Python/Scala

```
sqlContext.
  sql("""SELECT * FROM customers WHERE name LIKE "A%" """)
```

Table: customers

cust_id	name	country
001	Ani	us
002	Bob	ca
003	Carlos	mx
…	…	…

cust_id	name	country
001	Ani	us

Saving DataFrames

§ **Data in DataFrames can be saved to a data source**

§ **Use DataFrame.write to create a DataFrameWriter**

§ **DataFrameWriter provides convenience functions to externally save the data represented by a DataFrame**

–**jdbc** inserts into a new or existing table in a database

–**json** saves as a JSON file

–**parquet** saves as a Parquet file

–**orc** saves as an ORC file

–**text** saves as a text file (string data in a single column only)

–**saveAsTable** saves as a Hive/Impala table (**HiveContext** only)

Language: Python/Scala

```
peopleDF.write.saveAsTable("people")
```

Options for Saving DataFrames

- § DataFrameWriter option methods
  - –format specifies a data source type
  - –mode determines the behavior if file or table already exists:
  - overwrite, append, ignore or error (default is error)
  - –partitionBy stores data in partitioned directories in the form
  - *column=value* (as with Hive/Impala partitioning)
  - –options specifies properties for the target data source
  - –save is the generic base function to write the data

Language: Python/Scala

```
peopleDF.write.
  format("parquet").
  mode("append").
  partitionBy("age").
  saveAsTable("people")
```


DataFrames and RDDs

§ **DataFrames are built on RDDs**

–Base RDDs contain **Row** objects

–Use **rdd** to get the underlying RDD

```
peopleRDD = peopleDF.rdd
```

peopleDF

age	name	pcode
null	Alice	94304
30	Brayden	94304
19	Carla	10036
46	Diana	null
null	Étienne	94104

peopleRDD

Row[null,Alice,94304]
Row[30,Brayden,94304]
Row[19,Carla,10036]
Row[46,Diana,null]
Row[null,Étienne,94104]

DataFrames and RDDs

§ **Row RDDs have all the standard Spark actions and transformations**
–Actions: **collect**, **take**, **count**, and so on
–Transformations: **map**, **flatMap**, **filter**, and so on

§ **Row RDDs can be transformed into pair RDDs to use map-reduce methods**

§ **DataFrames also provide convenience methods (such as map, flatMap,and foreach)for converting to RDDs**

Working with **Row** Objects

–Use **Array**-like syntax to return values with type **Any**
–**row(*n*)** returns element in the *n*th column
–**row.fieldIndex("age")**returns index of the **age** column
–Use methods to get correctly typed values
–**row.getAs[Long]("age")**
–Use type-specific **get** methods to return typed values
–**row.getString(*n*)** returns *n*th column as a string
–**row.getInt(*n*)** returns *n*th column as an integer
–And so on

Prerequisites

- Docker
- Zeppelin Docker Container
- Terminal Tools (Command Prompt, PowerShell)

Working with Spark RDDs, Pair-RDDs

RDD Operations

Transformations	Actions
map()	count()
flatMap()	collect()
filter()	first(), top(n)
union()	take(n), takeOrdered(n)
intersection()	countByValue()
distinct()	reduce()
groupByKey()	foreach()
reduceByKey()	…
sortByKey()	
join()	
…	

Lambda Expression

PySpark WordCount example:

```
input_file = sc.textFile("/path/to/text/file")
map = input_file.flatMap(lambda line: line.split(" ")) \
                        .map(lambda word: (word, 1))
counts = map.reduceByKey(lambda a, b: a + b)
counts.saveAsTextFi
```

lambda arguments: expression

PySpark RDD API

https://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

map(*f, preservesPartitioning=False*) [source]

Return a new RDD by applying a function to each element of this RDD.

```
>>> rdd = sc.parallelize(["b", "a", "c"])
>>> sorted(rdd.map(lambda x: (x, 1)).collect())
[('a', 1), ('b', 1), ('c', 1)]
```

flatMap(*f, preservesPartitioning=False*) [source]

Return a new RDD by first applying a function to all elements of this RDD, and then flattening the results.

```
>>> rdd = sc.parallelize([2, 3, 4])
>>> sorted(rdd.flatMap(lambda x: range(1, x)).collect())
[1, 1, 1, 2, 2, 3]
>>> sorted(rdd.flatMap(lambda x: [(x, x), (x, x)]).collect())
[(2, 2), (2, 2), (3, 3), (3, 3), (4, 4), (4, 4)]
```


68

Practice with flight data (1)

Data: **airports.dat** (https://openflights.org/data.html)

[Airport ID, Name, City, Country, IATA, ICAO, Latitude, Longitude, Altitude, Timezone, DST, Tz database, Type, Source]

Try to do somethings:

- Create RDD from textfile
- Count the number of airports
- Filter by country
- Group by country
- Count the number of airports in each country

69

Practice with flight data (2)

- Data: **airports.dat** (https://openflights.org/data.html)

[Airport ID, Name, City, Country, IATA, ICAO, Latitude, Longitude, Altitude, Timezone, DST, Tz database, Type, Source]

- Data: **routes.dat**

[Airline, Airline ID, Source airport, Source airport ID, Destination airport, Destination airport ID, Codeshare, Stops, Equipment]

Try to do somethings:

- Join 2 RDD
- Count the number of flights arriving in each country

Working with DataFrame and Spark SQL

Creating a DataFrame(1)

```
%pyspark
from pyspark.sql import *

Employee = Row("firstName", "lastName", "email", "salary")

employee1 = Employee('Basher', 'armbrust', 'bash@edureka.co', 100000)
employee2 = Employee('Daniel', 'meng', 'daniel@stanford.edu', 120000 )
employee3 = Employee('Muriel', None, 'muriel@waterloo.edu', 140000 )
employee4 = Employee('Rachel', 'wendell', 'rach_3@edureka.co', 160000 )
employee5 = Employee('Zach', 'galifianakis', 'zach_g@edureka.co', 160000 )

employees = [employee1,employee2,employee3,employee4,employee5]

print(Employee[0])

print(employees)

dframe = spark.createDataFrame(employees)
dframe.show()
```


Creating a DataFrame

From CSV file:

```
%pyspark
flightData2015 = spark\
  .read\
  .option("inferSchema", "true")\
  .option("header", "true")\
  .csv("/usr/zeppelin/module9/2015-summary.csv")

flightData2015.show()
```

```
%pyspark
from pyspark.sql import *
list = [('Ankit',25),('Jalfaizy',22),('saurabh',20),('Bala',26)]
rdd = sc.parallelize(list)
people = rdd.map(lambda x: Row(name=x[0], age=int(x[1])))
df = spark.createDataFrame(people)

df.show()
```

DataFrame APIs

- **DataFrame**: show(), collect(), createOrReplaceTempView(), distinct(), filter(), select(), count(), groupBy(), join()…
- **Column**: like()
- **Row**: row.key, row[key]
- **GroupedData**: count(), max(), min(), sum(), …

https://spark.apache.org/docs/latest/api/python/pyspark.sql.html

Spark SQL

- Create a temporary view
- Query using SQL syntax

```
%pyspark
flightData2015.createOrReplaceTempView("flight_data_2015")

maxSql = spark.sql("""
SELECT DEST_COUNTRY_NAME, sum(count) as destination_total
FROM flight_data_2015
GROUP BY DEST_COUNTRY_NAME
ORDER BY sum(count) DESC
LIMIT 5
""")

maxSql.show()
```

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

76

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!

soict.hust.edu.vn/ fb.com/groups/soict

77


ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

78

Lecture 5: Data Preparation

Data Preparation

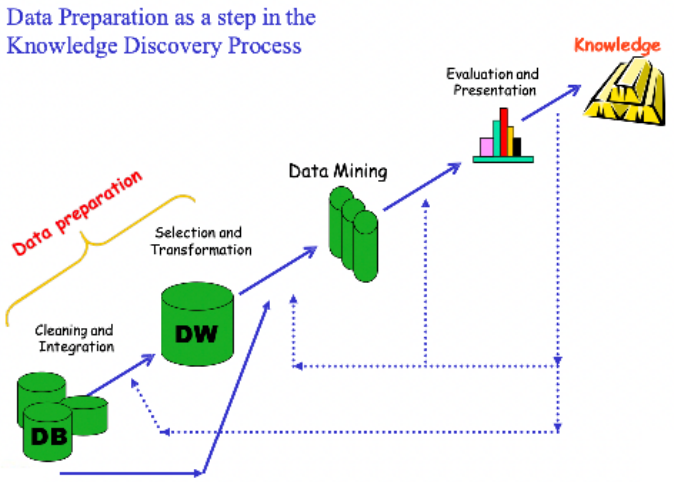
- Introduction to Data Preparation.
- Types of Data.
- Outliers.
- Data Transformation.
- Missing Data.

Why Prepare Data

- Some data preparation is needed for all mining tools.
- The purpose of preparation is to transform datasets so that their information content is best exposed to the mining tool.
- Error prediction rate should be lower (or the same) after the preparation as before it.
- Preparing data also prepares the miner so that when using prepared data the miner produces better models, faster.
- Good data is a prerequisite for producing effective models of any type.
- Data need to be formatted for a given software tool.
- Data need to be made adequate for a given method.
- Data in the real world is dirty.
    - **incomplete**: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data.
        - e.g., occupation="".
    - **noisy**: containing errors or outliers.
        - e.g., Salary = "-10", Age = "222".
    - **inconsistent**: containing discrepancies in codes or names.
        - e.g., Age= "42", Birthday = "03/07/1997".
        - e.g., Was rating "1,2,3", now rating "A, B, C".
        - e.g., discrepancy between duplicate records.

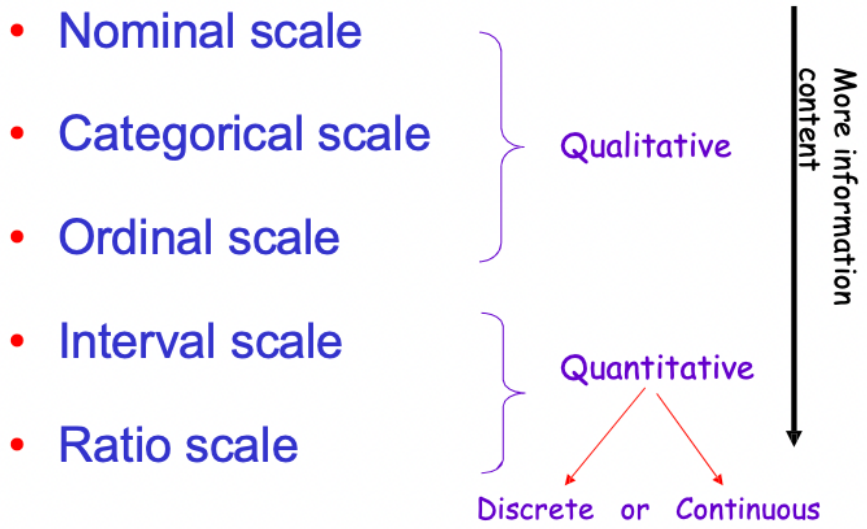
Major Tasks in Data Preparation

- Data discretization:
    - Part of data reduction but with particular importance, especially for numerical data.
- Data cleaning:
    - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies.
- Data integration:
    - Integration of multiple databases, data cubes, or files.
- Data transformation:
    - Normalization and aggregation.
- Data reduction:
    - Obtains reduced representation in volume but produces the same or similar analytical results.



Types of Data

Types of Measurements



Types of Measurements: Example

- Nominal:
  - ID numbers, Names of people
- Categorical:
  - eye color, zip codes
- Ordinal:
  - rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in {tall, medium, short}
- Interval:
  - calendar dates, temperatures in Celsius or Fahrenheit, GRE (Graduate Record Examination) and IQ scores
- Ratio:
  - temperature in Kelvin, length, time, counts

Data Conversion

- Some tools can deal with nominal values but other need fields to be numeric.
- Convert ordinal fields to numeric to be able to use “>” and “<” comparisons on such fields.
  - A → 4.0
  - A- → 3.7
  - B+ → 3.3
  - B → 3.0
- Multi-valued, unordered attributes with small no. of values:
  - E.g., Color=Red, Orange, Yellow, ..., Violet

- For each value v create a binary “flag” variable C_v , which is 1 if Color=v, 0 otherwise.

Conversion: Nominal, Many Values

- Examples:
  - US State Code (50 values).
  - Profession Code (7,000 values, but only few frequent).
- Ignore ID-like fields whose values are unique for each record.
- For other fields, group values “naturally”:
  - E.g., 50 US States => 3 or 5 regions.
  - Profession – select most frequent ones, group the rest.
- Create binary flag-fields for selected values.

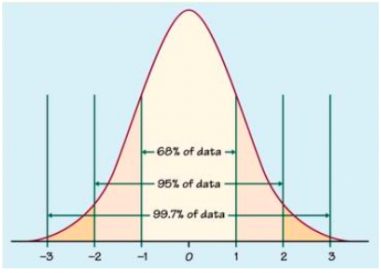
Outliers

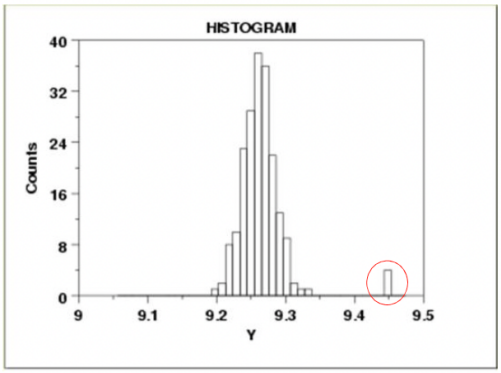
- Outliers are values thought to be out of range.
  - **“An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism”.**
  - Can be detected by standardizing observations and label the standardized values outside a predetermined bound as outliers.
  - Outlier detection can be used for fraud detection or data cleaning.
- Approaches:
  - Do nothing.
  - Enforce upper and lower bounds.
  - Let binning handle the problem.

Outlier Detection

- Univariate:
  - Compute mean and std. deviation. For k=2, k=3, x is an outlier if outside limits (normal distribution assumed).

$$(x - ks, x + ks)$$

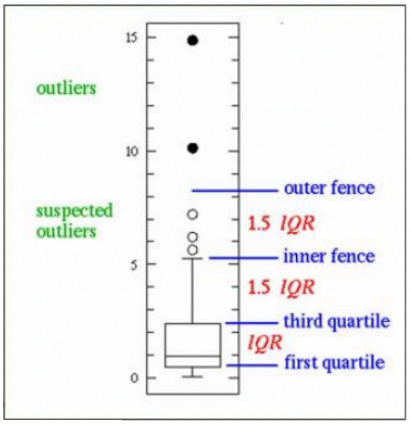
  - 

  - 

  - Boxplot: An observation in an extreme outlier if:

(Q1-3×IQR, Q3+3×IQR), where IQR=Q3-Q1

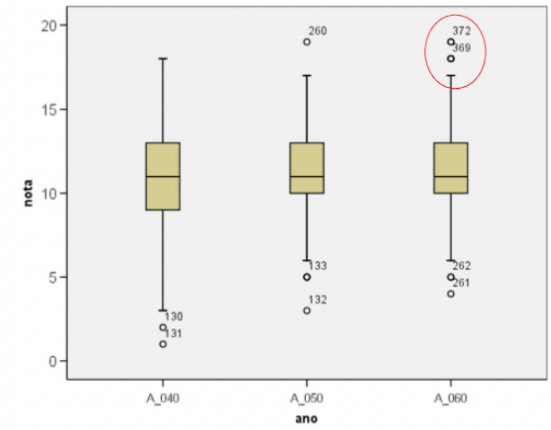
(IQR = Inter Quartile Range)

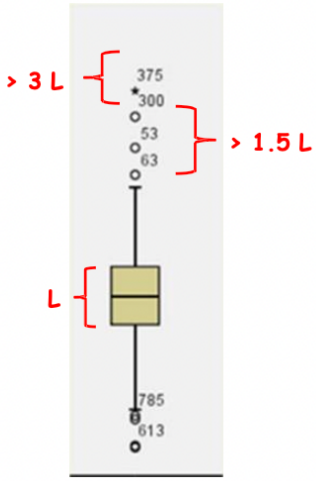


and declared a mild outlier if it lies outside of the interval

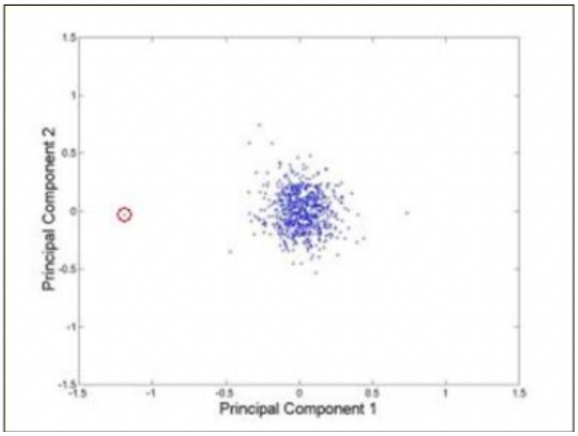
(Q1-1.5×IQR, Q3+1.5×IQR).

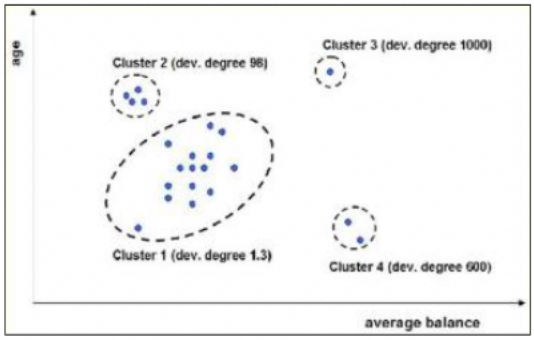
http://www.physics.csbsju.edu/stats/box2.html





- Multivariate:
  - Clustering:
    - Very small clusters are outliers.

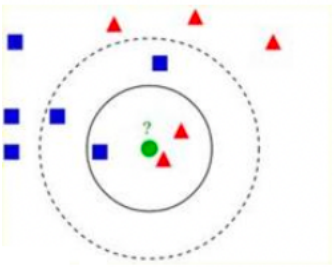


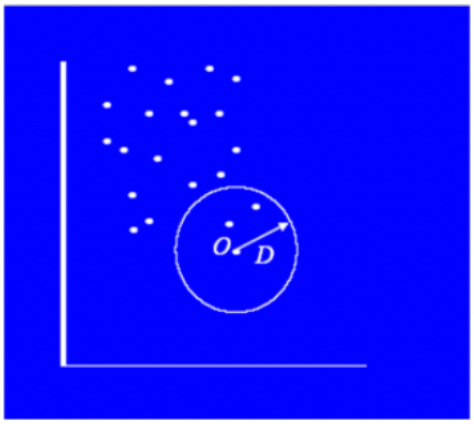


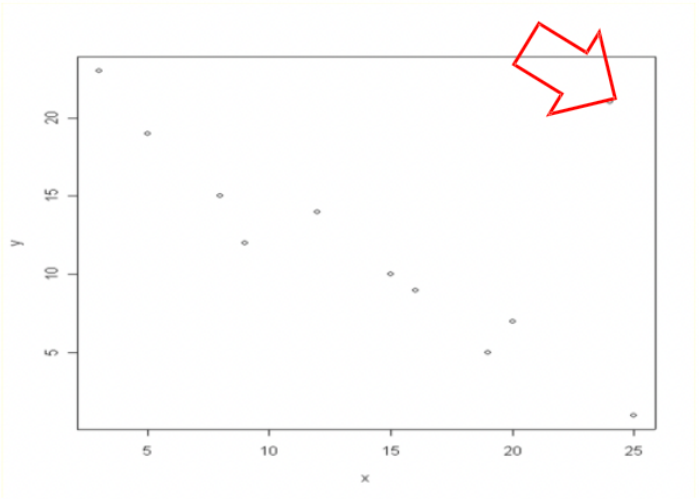
http://www.ibm.com/developerworks/data/library/techarticle/dm-0811wurst/

  - Distance based:
    - An instance with very few neighbors within D is regarded as an outlier.

Knn algorithm







A bi-dimensional outlier that is not an outlier in either of its projections.

Data Transformation

Normalization

- For distance-based methods, normalization helps to prevent that attribute with large ranges out-weight attributes with small ranges.
  - min-max normalization

$$v' = \frac{v - \min_v}{\max_v - \min_v}(new\_max_v - new\_min_v) + new\_min_v$$

  - z-score normalization

$$v' = \frac{v - \bar{v}}{\sigma_v}$$ does not eliminate outliers

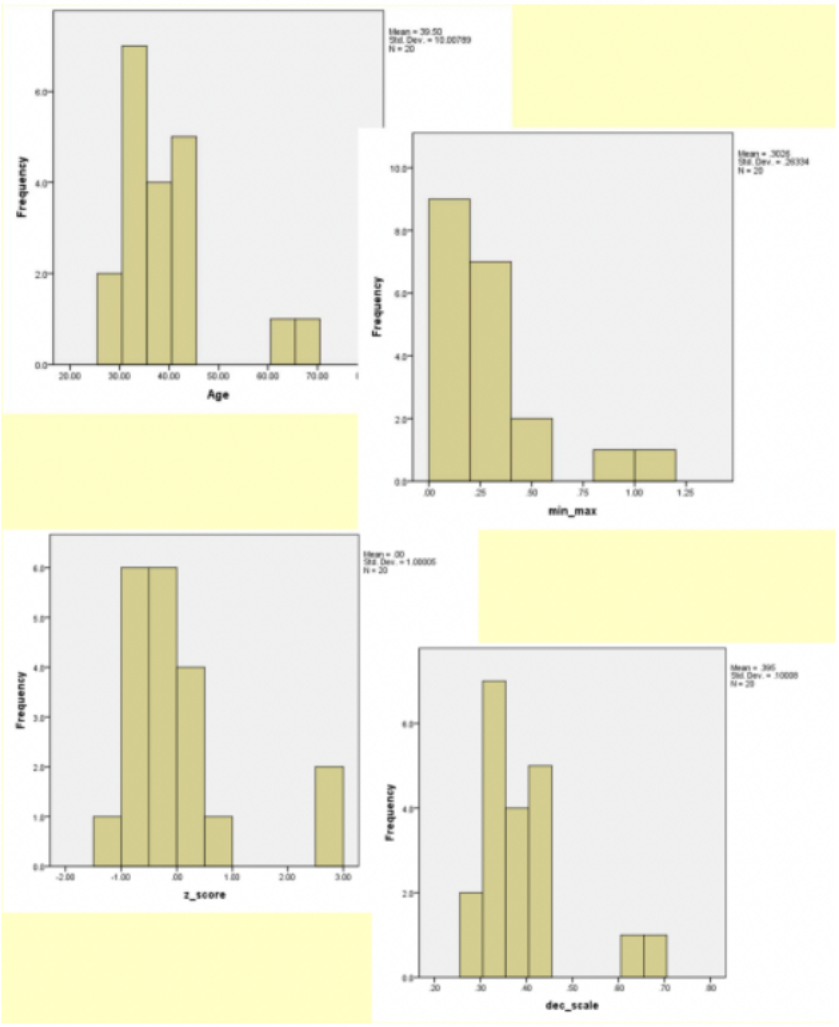
  - normalization by decimal scaling

$$v' = \frac{v}{10^j}$$

Where j is the smallest integer such that Max(| v' |)<1

range: -986 to 917 => j=3 -986 -> -0.986 917 -> 0.917

Age	min-max (0-1)	z-score	dec. scaling
44	0.421	0.450	0.44
35	0.184	-0.450	0.35
34	0.158	-0.550	0.34
34	0.158	-0.550	0.34
39	0.289	-0.050	0.39
41	0.342	0.150	0.41
42	0.368	0.250	0.42
31	0.079	-0.849	0.31
28	0.000	-1.149	0.28
30	0.053	-0.949	0.3
38	0.263	-0.150	0.38
36	0.211	-0.350	0.36
42	0.368	0.250	0.42
35	0.184	-0.450	0.35
33	0.132	-0.649	0.33
45	0.447	0.550	0.45
34	0.158	-0.550	0.34
65	0.974	2.548	0.65
66	1.000	2.648	0.66
38	0.263	-0.150	0.38
28	minimun		
66	maximum		
39.50	avgerage		
10.01	standard deviation		



Data Transformation

- It is the process to create new attributes:
  - Often called transforming the attributes or the attribute set.
- Data transformation usually combines the original raw attributes using different mathematical formulas originated in business models or pure mathematical formulas.
- Linear Transformations:
  - Normalizations may not be enough to adapt the data to improve the generated model.
  - Aggregating the information contained in various attributes might be beneficial.
  - If B is an attribute subset of the complete set A, A new attribute Z can be obtained by a linear combination:

$$Z = r_1 B_1 + r_2 B2 + \cdots + r_m B_M$$

- Quadratic Transformations:
  - In Quadratic Transformations, a new attribute is built as follows:

$$Z = r_{1,1} B_1^2 + r_{1,2} B_1 B_2 + \cdots + r_{m-1,m} B_{m-1} B_m + r_{m,m} B_m^2,$$

- These kinds of transformations have been thoroughly studied and can help to transform data to make it separable.

- Non-polynomial Approximations of Transformations:
  - Sometimes polynomial transformations are not enough.
  - For examples, guessing whether a set of triangles are congruent is not possible by simply observing their coordinates.
    - Computing the length of their segments will easily solve the problem → non polynomial transformation.

$$A = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$$

- Polynomial Approximations of Transformations:
  - We have observed that specific transformations may be needed to extract knowledge.
    - But help from an expert is not always available.
  - When no knowledge is available, a transformation f can be approximated via a polynomial transformation using a brute search with one degree at a time.
    - Using the Weistrass approximation, there is a polynomial function f that takes the value Yi for each instance Xi.

$$Y = f(X_1, X_2, \ldots, X_n)$$

  - There are many polynomials verifying Y=f(X) as we want.
  - As the number of instances in the data set increases, the approximations will be better.
  - We can use computer assistance to approximate the intrinsic information.
  - When the intrinsic transformation is polynomial, we need to add the cartesian product of the attributes needed for the polynomial degree approximation.
  - Sometimes the approximation obtained must be rounded to avoid the limitations of the computer digital precision.

- Rank transformation:
  - A change in an attribute distribution can result in a change of model performance.
  - The simplest transformation to accomplish this in numerical attributes is to replace the value of an attribute with its rank.
  - The attribute will be transformed into a new attribute containing integer values ranging from 1 to m, being m the number of instances in the data set.
  - Next, we can transform the ranks to normal scores representing their probabilities in the normal distribution by spreading these values on the Gaussian curve using a simple transformation given by:

$$y = \Phi^{-1}\left(\frac{r_i - \frac{3}{8}}{m + \frac{1}{4}}\right)$$

  - Being ri the rank of the observation i and Φ the cumulative normal function.
  - Note: This transformation cannot be applied separately to the training and test partitions.

- Box-Cox Transformations:
  - When selecting the optimal transformation for an attribute is that we do not know in advance which transformation will be the best.
  - The Box-Cox transformation aims to transform a continuous variable into an almost normal distribution.
  - This can be achieved by mapping the values using following set of transformations:

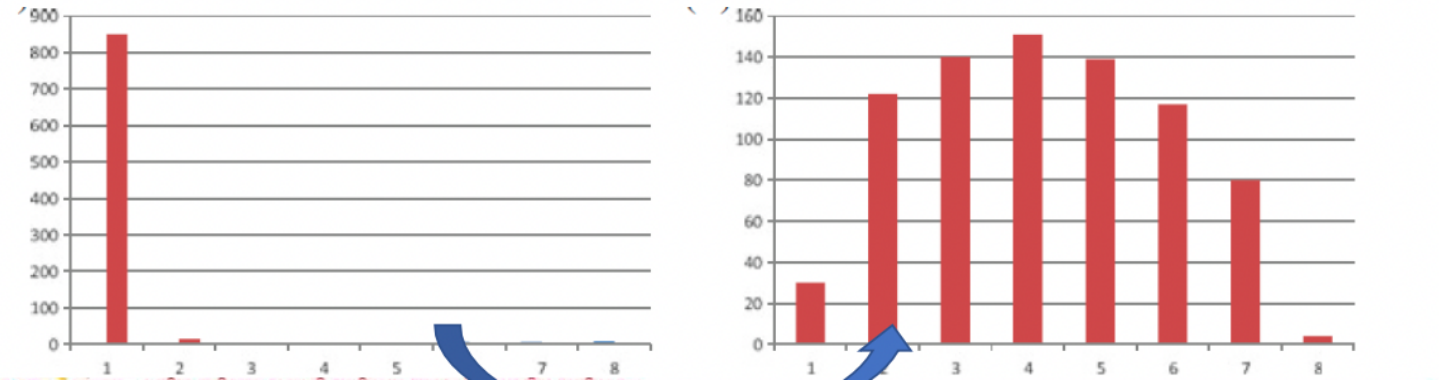
$$y = \begin{cases} x^{\lambda-1}/\lambda, & \lambda \neq 0 \\ log(x), & \lambda = 0 \end{cases}$$

- All linear, inverse, quadratic and similar transformations are special cases of the Box-Cox transformations.
- Please note that all the values of variable x in the previous slide must be positive. If we have negative values in the attribute, we must add a parameter c to offset such negative values:

$$y = \begin{cases} (x+c)^{\lambda-1}/g\lambda, & \lambda \neq 0 \\ log(x+c)/g, & \lambda = 0 \end{cases}$$

- The parameter g is used to scale the resulting values, and it is often considered as the geometric mean of the data.
- The value of λ is iteratively found by testing different values in the range from −3.0 to 3.0 in small steps until the resulting attribute is as close as possible to the normal distribution.

- Spreading the Histogram:
  - Spreading the histogram is a special case of Box-Cox transformations.
  - As Box-Cox transforms the data to resemble a normal distribution, the histogram is thus spread as shown here.



  - When the user is not interested in converting the distribution to a normal one, but just spreading it, we can use two special cases of Box-Cox transformations.
    - 1. Using the logarithm (with an offset if necessary) can be used to spread the right side of the histogram: y = log(x).
    - 2. If we are interested in spreading the left side of the histogram, we can simply use the power transformation $y = x^g$.

- Nominal to Binary Transformation:
  - The presence of nominal attributes in the data set can be problematic, especially if the Data Mining (DM) algorithm used cannot correctly handle them.
  - The first option is to transform the nominal variable to a numeric one.
  - Although simple, this approach has two big drawbacks that discourage it:
    - With this transformation we assume an ordering of the attribute values.
    - The integer values can be used in operations as numbers, whereas the nominal values cannot.
  - To avoid the problems, a very typical transformation used for DM methods is to map each nominal attribute to a set of newly generated attributes.
  - If N is the number of different values the nominal attribute has, we will substitute the nominal variable with a new set of binary attributes, each one representing one of the N possible values.

    - For each instance, only one of the N newly created attributes will have a value of 1, while the rest will have the value of 0.
    - This transformation is also referred in the literature as 1-to-N transformation.
    - A problem with this kind of transformation appears when the original nominal attribute has a large cardinality:
        - The number of attributes generated will be large as well.
        - Resulting in a very sparse data set which will lead to numerical and performance problems.
- Transformations via Data Reduction:
    - When the data set is very large, performing complex analysis and DM can take a long computing time.
    - Data reduction techniques are applied in these domains to reduce the size of the data set while trying to maintain the integrity and the information of the original data set as much as possible.
    - Mining on the reduced data set will be much more efficient and it will also resemble the results that would have been obtained using the original data set.
    - The main strategies to perform data reduction are Dimensionality Reduction (DR) techniques.
    - They aim to reduce the number of attributes or instances available in the data set.
    - Well known attribute reduction techniques are Wavelet transforms or Principal Component Analysis (PCA).
    - Many techniques can be found for reducing the dimensionality in the number of instances, like the use of clustering techniques, parametric methods and so on.
    - The use of binning and discretization techniques is also useful to reduce the dimensionality and complexity of the data set.
    - They convert numerical attributes into nominal ones, thus drastically reducing the cardinality of the attributes involved.

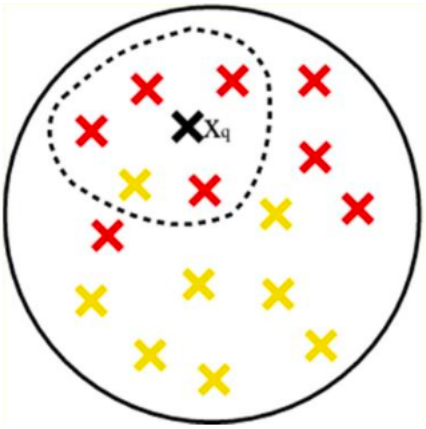
Missing Data

- Data is not always available.
    - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data.
- Missing data may be due to:
    - equipment malfunction.
    - inconsistent with other recorded data and thus deleted.
    - data not entered due to misunderstanding.
    - certain data may not be considered important at the time of entry.
    - not register history or changes of the data.
- Missing data may need to be inferred.
- Missing values may carry some information content.
    - E.g., a credit application may carry information by noting which field the applicant did not complete.
- There are always MVs in a real dataset.
    - MVs may have an impaction modelling, in fact, they can destroy it!
    - Some tools ignore missing values, others use some metric to fill in replacements.
    - The modeler should avoid default automated replacement techniques.
    - Difficult to know limitations, problems and introduced bias.
    - Replacing missing values without elsewhere capturing that information removes information from the dataset.

How to Handle Missing Data?

- Ignore records (use only cases with all values)
    - Usually done when class label is missing as most prediction methods do not handle missing data well.
    - Not effective when the percentage of missing values per attribute varies considerably as it can lead to insufficient and/or biased sample sizes.
- Ignore attributes with missing values.
- Use only features (attributes) with all values (may leave out important features).
- Fill in the missing value manually.
    - tedious + infeasible?
- Use a global constant to fill in the missing value.
    - E.g., “unknown”. (May create a new class).
- Use the attribute mean to fill in the missing value.
    - It will do the least harm to the mean of existing data.
    - If the mean is to be unbiased.
    - What if the standard deviation is to be unbiased?
- Use the attribute mean for all samples belonging to the same class to fill in the missing value.
- Use the most probable value to fill in the missing value.
    - Inference-based such as Bayesian formula or decision tree.
    - Identify relationships among variables.
        - Linear regression, Multiple linear regression, Non-linear regression.
    - Nearest-Neighbor estimator
        - Finding the k neighbors nearest to the point and fill in the most frequent value or the average value.
        - Finding neighbors in a large dataset may be slow.

Nearest-neighbor



How to Handle Missing Data?

- Note that, it is as important to avoid adding bias and distortion to the data as it is to make the information available.
    - Bias is added when a wrong value is filled in.
- No matter what techniques you use to conquer the problem, it comes at a price. The more guessing you must do, the further away from the real data the database becomes. Thus, in turn, it can affect the accuracy and validation of the mining results.

Summary

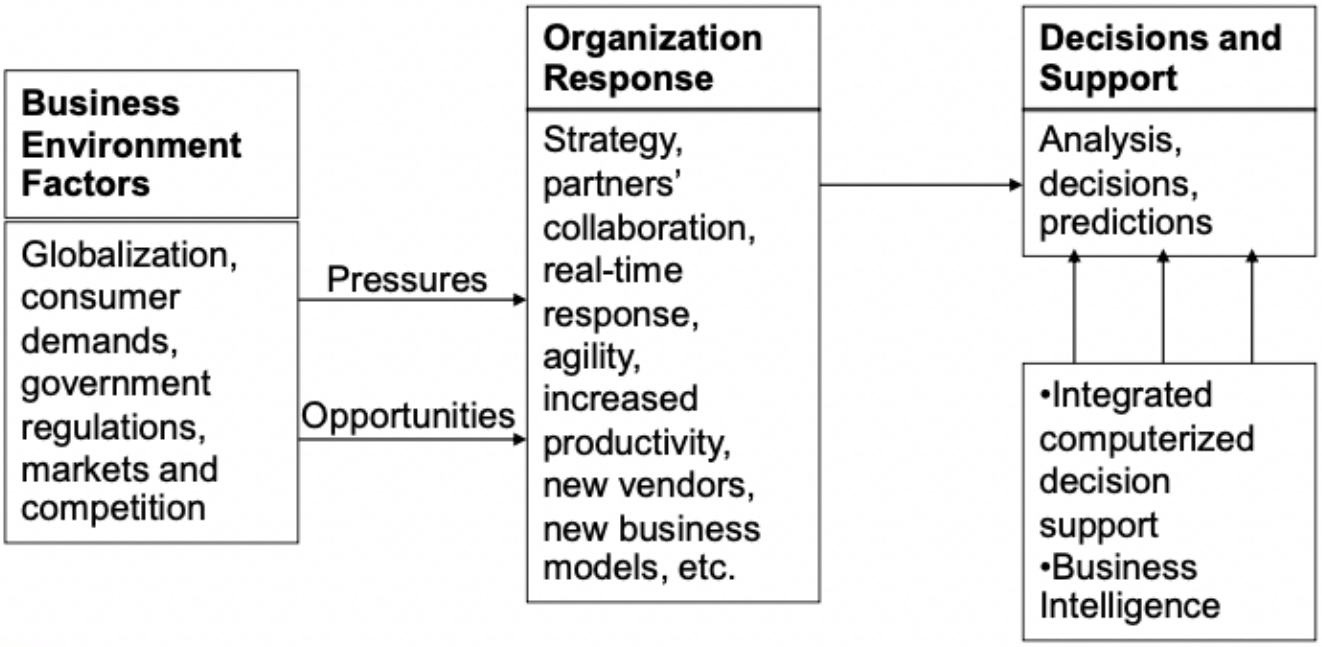
• Every real-world data set needs some kind of data pre-processing.
• Deal with missing values.
• Correct erroneous values.
• Select relevant attributes.
• Adapt data set format to the software tool to be used.
• In general, data pre-processing consumes more than 60% of a data mining project effort.

Table of Contents

Lecture 6.1: Introduction to Decision Support Systems

Decision Making

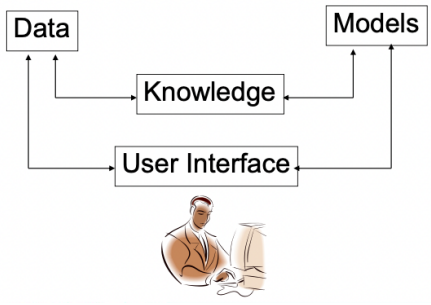
- Business Environment Factors.
  - Markets: strong competition, global markets, market on Internet.
  - Consumer demands: customization, quality, diversity, delivery.
  - Technology: more innovations, more obsolescence rate, more information overload.
  - Societal: more regulation and deregulation, more diversified workforce, more social responsibility.
- Business Pressure-Response-Support Model



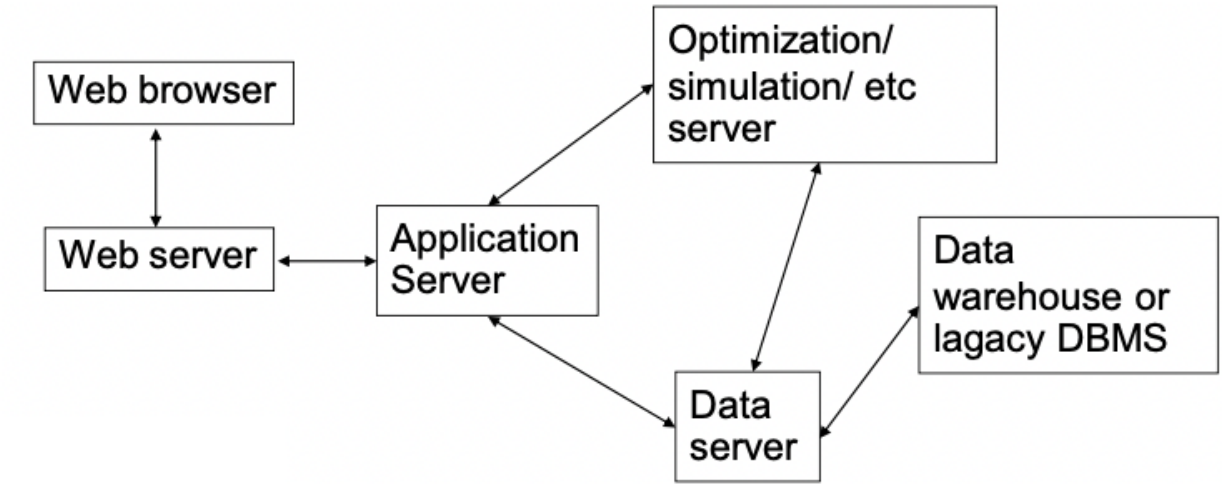
- Process of Decision Making:
  - Define the problem (i.e., a decision situation that may deal with some difficulty or with an opportunity).
  - Construct a model that describes the real-world problem.
  - Identify possible solutions to the modeled problem and evaluate the solutions.

Decision Support Systems

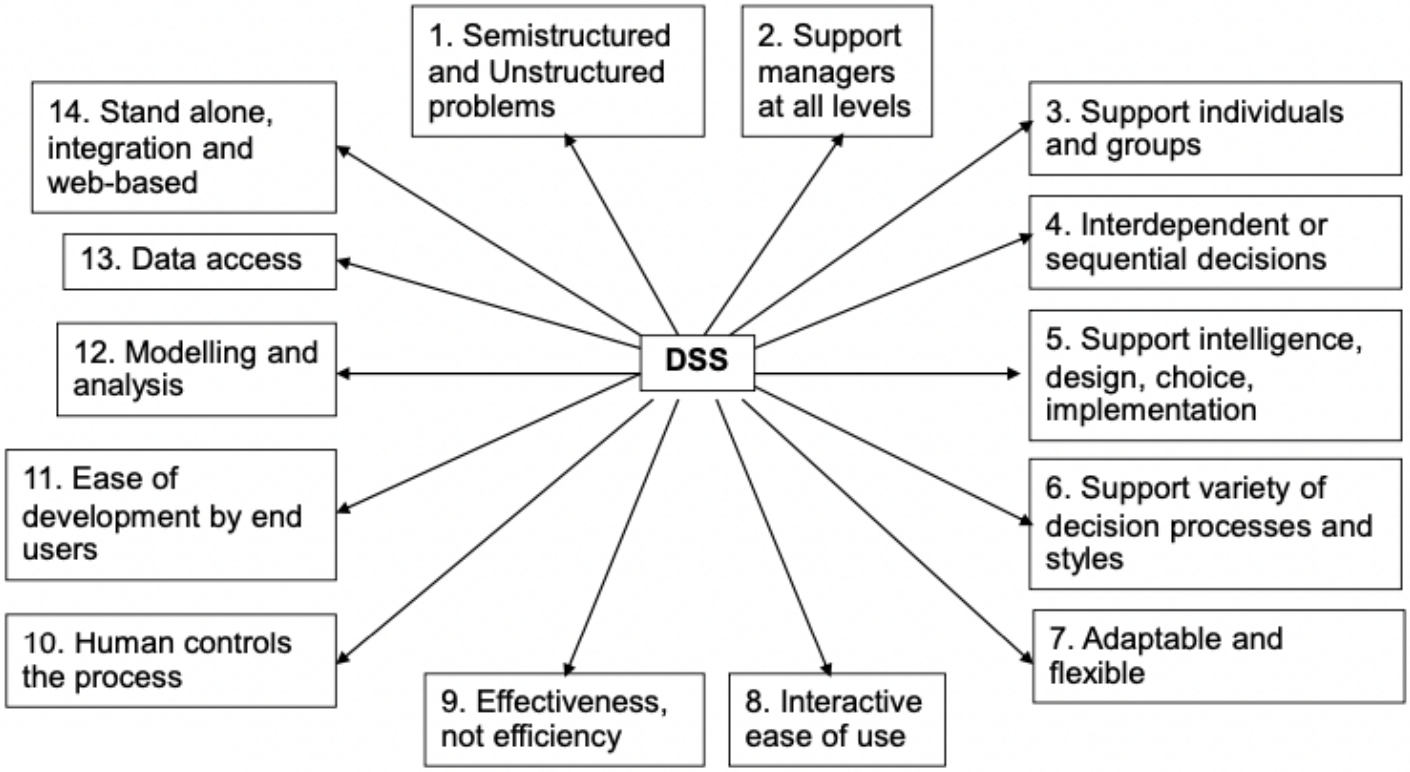
- Definition I (Keen and Scott-Morton):
  - Decision Support Systems couple the intellectual resources of individuals with the capabilities of the computer to improve the quality of decisions.
  - It is a computer-based support system for management decision makers who deal with semi-structured problems.
- High-level Architecture of DSS:



- Multitiered architecture for incorporating optimization, simulation, and other models into web based DSS:

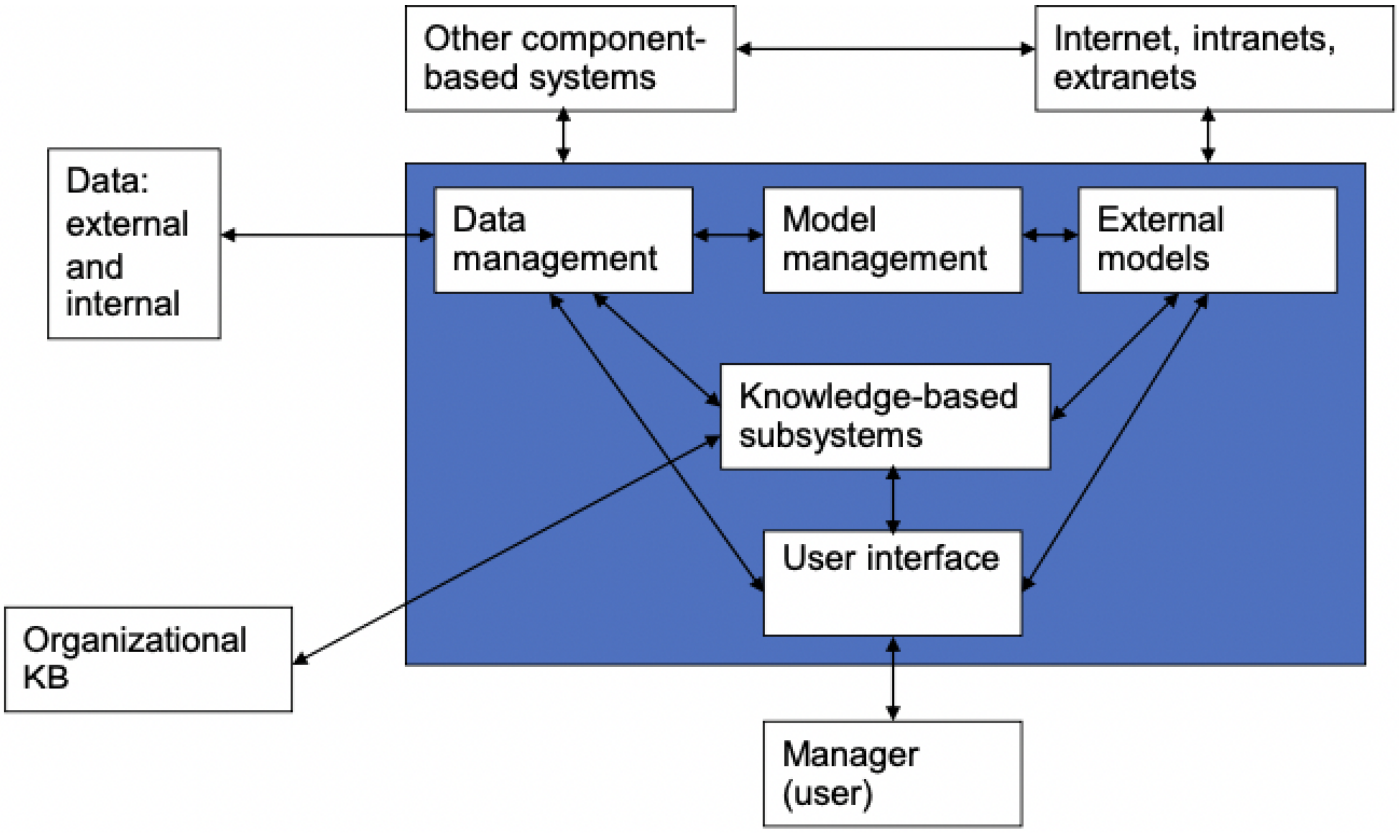


- Key characteristics and capabilities of DSS:

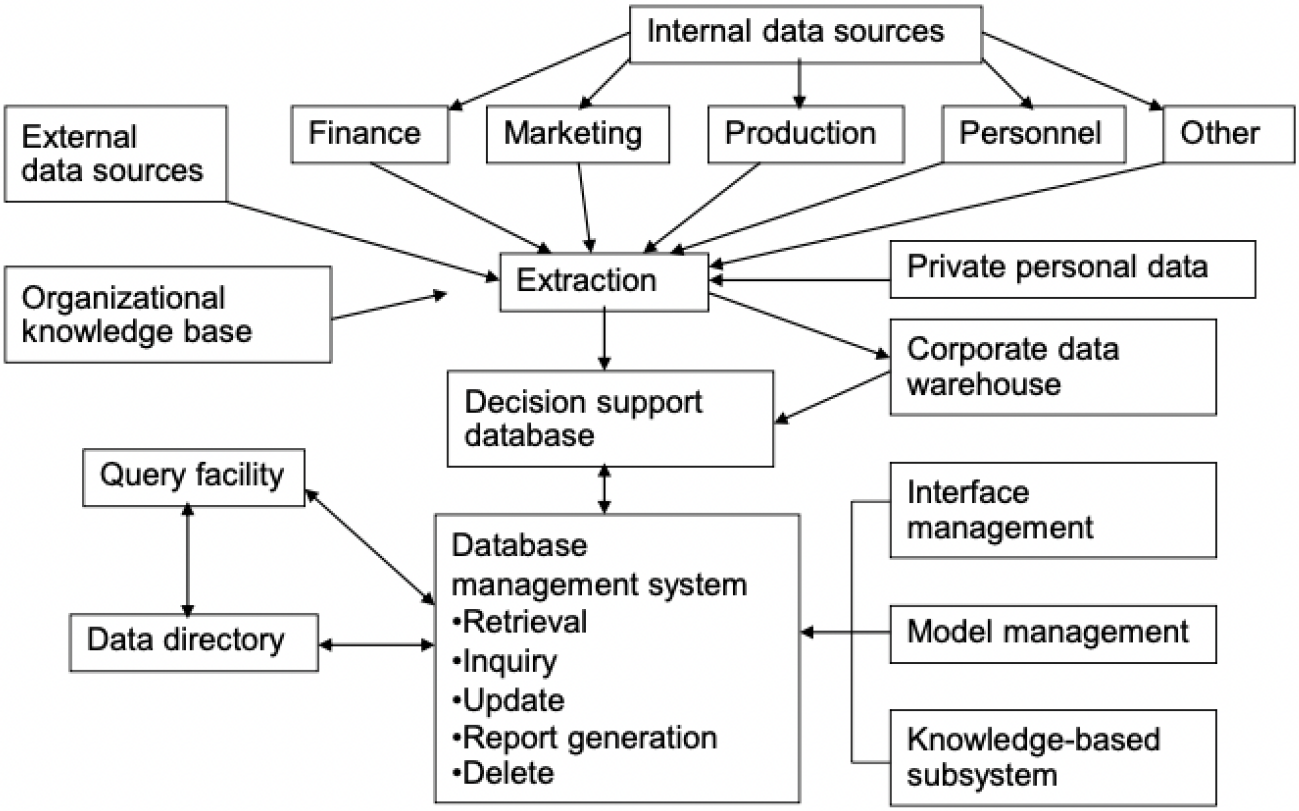


- Components of DSS:
  - Data Management Subsystems: include Database Management System (DBMS) and data warehouse.
  - Model Management Subsystems: include financial, statistical, management science, or other quantitative models that provide the analytical capabilities (also called model base management system MDMS).
  - User Interface Subsystems: include graphical user interface (GUI) that allows users to communicate with the system.

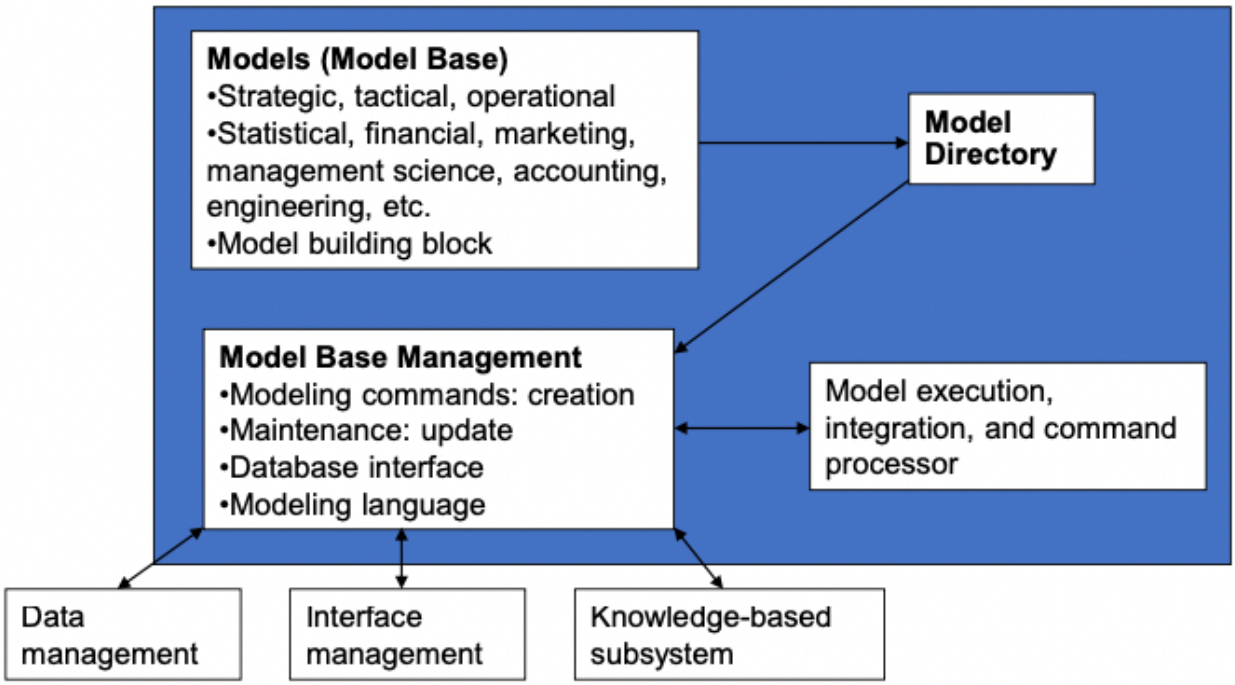
- Schematic View of DSS:



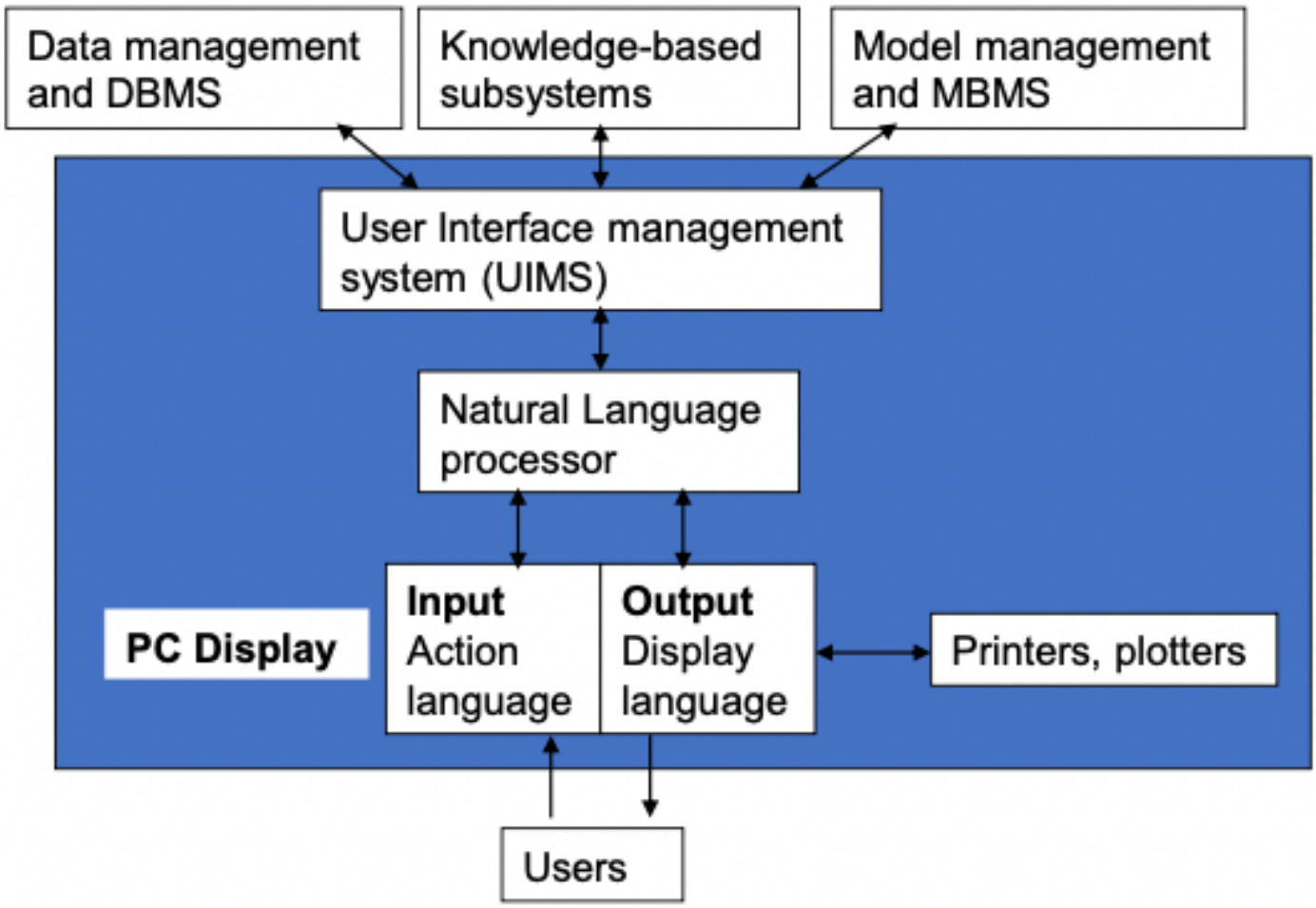
- The Structure of the DMS:



- The Structure of MMS:



- Schematic View of the User Interface Systems:



Modeling

- Static and Dynamic Models
- Certainty, Uncertainty, and Risk
- Modeling with Spreadsheets
- Decision Tables and Decision Trees
- The Structure of Mathematical Models
- Mathematical Programming Optimization
- Multiple Goals, Sensitivity, What-if and Goal Seeking
- Problem Solving Search Methods

Static and Dynamic Models

- A static model takes a single snapshot of the system. The decision is made on that snapshot.
- A decision where to buy a product, a quarterly or annual income statement, the decision to invest are static.
- A dynamic model is time dependent.
- Determining how many checkout points should be open in a supermarket – this needs to consider the time of day because different numbers of customers arrive during each hour.

Certainty, Uncertainty, and Risk

- In decision making under certainty, it is assumed that complete knowledge is available so that the decision maker knows exactly what the outcome of each course of action will be.
- Certainty models are relatively easy to develop and solve, and they can yield optimal solutions.
- In decision making under uncertainty, the decision maker consider situations in which several outcomes are possible for each course of action.
- The decision maker does not know, or cannot estimate, the probability of occurrence of the possible outcomes.
- It is more difficult than making decision under certainty because there is insufficient information.
- A decision made under risk (also known as a probabilistic, or stochastic) is one in which the decision maker must consider several possible outcomes for each alternative, each with a given probability of occurrence.
- Risk analysis is a decision-making method that analyzes the risk (based on assumed known probabilities) associated with different alternatives.

Modeling with Spreadsheets

- Spreadsheet packages were quickly recognized as easy-to-use implementation software for the development of a wide range of applications in business, engineering, mathematics, and science.
- Spreadsheets include extensive statistical, forecasting, and other modeling and database management capabilities, functions, and routines.
- These DSS-related spreadsheets include solver (solver.com), What's best (lindo.com), Braincel (promland.com), NeuralTools, Evolver, @RISK (palisade.com), and GRG-2 (MS Excel).

Decision Tables and Trees

- Decision tables conventionally organize information and knowledge in a systematic tabular manner to prepare for its analysis.
- Decision under certainty:

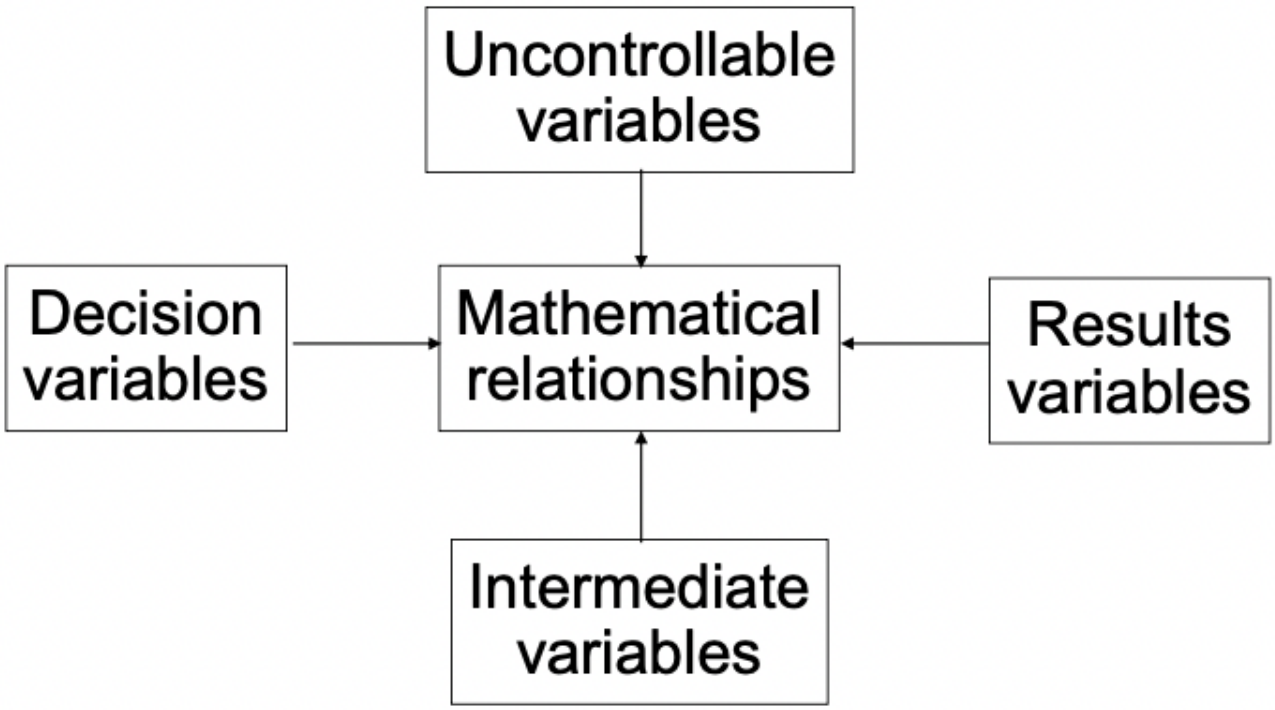
	State of Nature (Uncontrollable Variables)		
Alternative	Solid Growth(%)	Stagnation(%)	Inflation(%)
Bonds	12	6	3
Stocks	15	3	-2
CDs	6.5	6.5	6.5

- Decision under risk:

	Solid Growth	Stagnation	Inflation	Expected Value
Alternative	.5 (%)	.3 (%)	.2 (%)	(%)
Bonds	12	6	3	8.4
Stocks	15	3	-2	8.0
CDs	6.5	6.5	6.5	6.5

- A decision tree shows the relationships of the problem graphically and can handle complex situation in a compact form.
- TreeAge Pro (treeage.com), PrecisionTree (palisade.com), psychwww.com/mtsite/dectree.html and Mind Tools (mindtools.com).

Structure of Mathematical Model



Example of the Components of Models

Area	Decision variables	Result variables	Uncontrollable variables and parameters
Financial investment	Investment alternatives and amounts	Total profit, risk Rate of return on investment (ROI) Earning per share Liquidity level	Inflation rate Prime rate Competition
Marketing	Advertising budget Where to advertise	Market share Customer satisfaction	Customer's income Competitor's action
Manufacturing	What and how much to produce Inventory levels Compensation programs	Total cost Quantity level Employee satisfaction	Machine capacity Technology Material prices
Accounting	Use of computers Audit schedule	Data processing cost Error rate	Computer technology Tax rates Legal requirements
Transportation	Shipments schedule Use of smart cards	Total transport cost Payment float time	Delivery distance Regulations
Services	Staffing levels	Customer satisfaction	Demand for services

Mathematical Programming Optimization

- Linear Programming.
  - Product Mix.
  - Transportation Problem.
- Non-Linear Programming.
  - Travelling salesman.
  - Vehicle routing problem.

Multiple Goals

- Managers want to attain simultaneous goals, some of which may conflict.
- In addition to earning money, the company wants to grow, develop its products and employees, provide job security to its workers.
- Managers want to satisfy the shareholders and at the same time enjoy high salaries and expense accounts, and employees want to increase their take-home pay and benefits.
- To solve this kind of problems, common methods are:
  - Utility theory.
  - Goal programming.
  - Expression of goals and constraints, using LP.
  - A point system.

Sensitivity Analysis

- Sensitivity analysis attempts to assess the impact of a change in the input data or parameters on the proposed solution.
- Sensitivity allows flexibility and adaptation to changing conditions and to the requirements of different decision-making situations.
- Sensitivity analysis tests relationships such as the following:
    - The impact of changes in external (uncontrollable) variables and parameters on the outcome variables.
    - The impact of changes in decision variables on the outcome variables.
    - The effect of uncertainty in estimating external variables.
    - The effects of different dependent interactions among variables.
    - The robustness of decisions under changing condition.

What-if Analysis

- What-if analysis is structured as What will happen to the solution if an input variable, an assumption, or a parameter value is changed.
- For example, what will happen to the total inventory cost if the cost of the carrying inventories increases by 10 percent?
- A spreadsheet tool is a good example. A manager can analyze a cash flow problem by changing parameter's values and see the differences without any involvement computer programmers.

Goal Seeking

- Goal seeking calculates the values of the inputs necessary to achieve a desired level of an output (goal). It represents a backward solution approach.
- For example, what annual R&D budget is needed for an annual growth rate of 15 percent by 2012?

Problem Solving Search Method

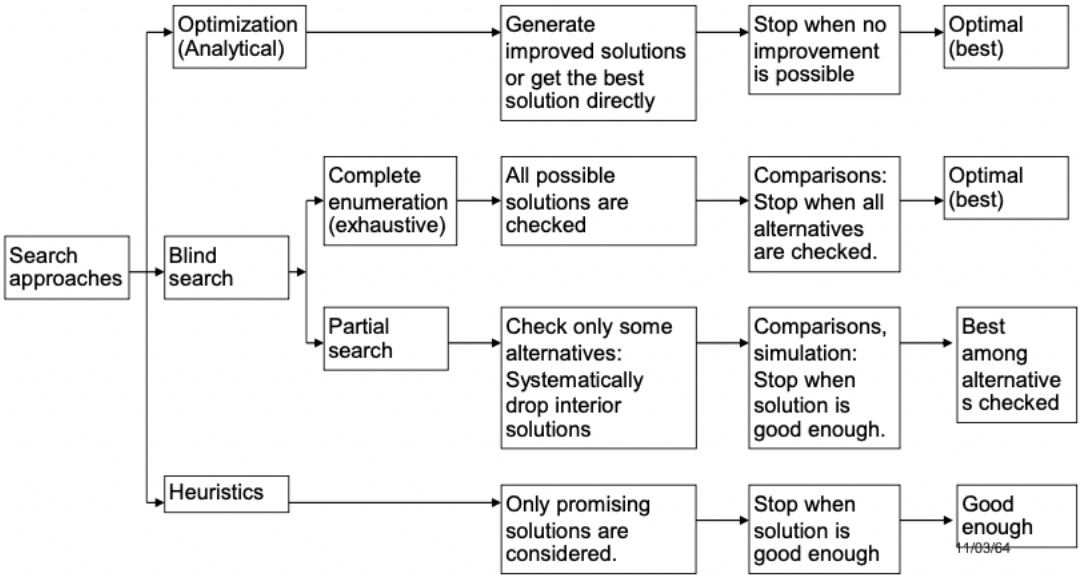


Table of Contents

Lecture 6.2: Introduction to Optimization

Learning Objectives

- Recognize decision-making situations which that may benefit from an optimization modeling approach.
- Formulate algebraic models for linear programming problems.
- Develop spreadsheet models for linear programming problems.
- Use Excel's Solver Add-In to solve linear programming problems.
- Interpret the results of models and perform basic sensitivity analysis.

Optimization: Basic Ideas

- Major field within the discipline of Data Analytics, Operations Research and Management Science.
- Optimization Problem Components:
    - Decision Variables.
    - Objective Function (to maximize or minimize).
    - Constraints (requirements or limitations).
- Basic Idea:
    - Find the values of the decision variables that maximize (minimize) the objective function value, while staying within the constraints.

Linear Programming (LP)

- If the objective function and all constraints are linear functions of the decision variables (e.g., no squared terms, trigonometric functions, ratios of variables), then the problem is called a Linear Programming (LP) problem. LPs are much easier to solve by computer than problems involving nonlinear functions.
- Real-world LPs are solved which contain hundreds of thousands to millions of variables (with specialized software!). Our problems are obviously much smaller, but the basic concepts are much the same.

Real-World Examples

- Dynamic and Customized Pricing.
- Product Mix.
- Scheduling/Allocation.
- Routing/Logistics.
- Supply Chain Optimization.
- Facility Location.
- Financial Planning/Asset Management.
- Etc.

Solving Optimization Problems

- Understand the problem; draw a diagram.
- Write a problem formulation in words:
- Write the algebraic formulation of the problem:
    - Define the decision variables.
    - Write the objective function in terms of the decision variables.
    - Write the constraints in terms of the decision variables.
- Develop Spreadsheet Model.
- Setup the Solver settings and solve the problem.
- Examine results, make corrections to model and/or Solver settings.

- Interpret the results and draw insights.

Solver

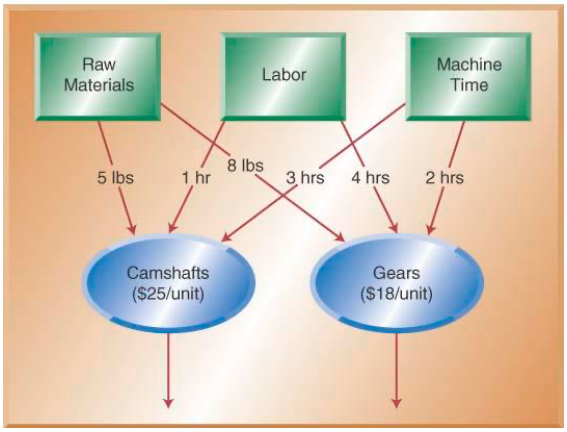
- We will use Solver as simple tool to illustrate the example, an Excel add-in, to solve Linear Programming problems.
- Add-In: An additional piece of software that Excel loads into memory when needed.

Example: Product Mix Decision

- DJJ Enterprises makes automotive parts, Camshafts & Gears.
- Unit Profit: Camshafts $25/unit, Gears $18/unit.
- Resources needed: Steel, Labor, Machine Time. In total, 5000 lbs. steel available, 1500 hours labor, and 1000 hours machine time.
- Camshafts need 5 lbs. steel, 1 hour labor, 3 hours machine time.
- Gears need 8 lbs. steel, 4 hours labor, 2 hours machine time.
- How many camshafts & gears to make to maximize profit?

Understanding the Problem

- Text-Based Formulation:
  - Decision Variables: Number of camshafts to make, number of gears to make.
  - Objective Function: Maximize profit.
  - Constraints: Don't exceed amounts available of steel, labor, and machine time.



Algebraic Formulation

- Decision Variables:
  - C = number of camshafts to make.
  - G = number of gears to make.
- Objective Function:
  - Maximize 25C + 18G (profit in $).
- Constraints:
  - 5C + 8G <= 5000 (steel in lbs.)
  - 1C + 4G <= 1500 (labor in hours)
  - 3C + 2G <= 1000 (machine time in hours)
  - C >= 0, G >= 0 (non-negativity)

Important Concepts

- Linear Program: The objective function and constraint are linear functions of the decision variables. Therefore, this is a Linear Program.
- Feasibility:
  - Feasible Solution. A solution is feasible for an LP if all constraints are satisfied.
  - Infeasible Solution. A solution is infeasible if one or more constraints is violated.
  - Check the solutions C=75, G=200; and C=300, G=200 for feasibility.
- Optimal Solution:
  - The optimal solution is the feasible solution with the largest (for a max problem) objective value (smallest for a min problem).

Solving Linear Programming Problems

- Trial and error:
  - Possible for very small problems.
  - Virtually impossible for large problems.
- Graphical approach:
  - It is possible to solve a 2-variable problem graphically to find the optimal solution (not shown).
- Simplex Method:
  - This is a mathematical approach developed by George Dantzig.
  - Can solve small problems by hand.
- Computer Software:
  - Most optimization software uses the Simplex Method to solve the problems.
  - Excel's Solver Add-In is an example of such software.
- Solver can solve LPs of up to 200 variables.

Spreadsheet Model Development

- Develop correct, flexible, documented model.
- Sections for decision variables, objective function, and constraints.
- Use algebraic formulation and natural structure of the problem to guide structure of the spreadsheet.
- Use one cell for each decision variable.
- Store objective function coefficients in separate cells and use another cell to compute the objective function value.
- Store constraint coefficients in cells, compute the LHS value of each constraint, for comparison to the RHS value.

Spreadsheet Model

- C=75, G=200 (cells B5, C5) entered as trial values. This is a feasible, but not the optimal, solution.
- Note close relationship to algebraic formulation.
- Note that only one distinct formula needed to be entered; once entered in Cell D8, it was copied to Cells D11:D13.
- This is possible because the coefficients were stored separately in a specific structure.

	A	B	C	D	E	F	G
1	**Example B.1**						
2	**DJJ Enterprises Production Planning**						
3							
4	**Decision Variables**	**Camshafts**	**Gears**				
5	Units to Make	75	200				
6							
7	**Objective**			**Total**			
8	Profit	$25	$18	$5,475			
9							
10	**Constraints**			**Used**		**Available**	
11	Steel (lbs)	5	8	1975	<=	5000	
12	Labor (hrs)	1	4	875	<=	1500	
13	Machine Time (hrs)	3	2	625	<=	1000	

D8: =B8*B$5+C8*C$5 (copied to D11:D13)

Solver Basics

- Don't even think about using Solver until you have a working, flexible spreadsheet model that you can use as a "what if" tool!
- Solver Settings
  - Specify Objective Cell (objective function).
  - Specify Changing Cells (decision variables).
  - Specify Constraints.
  - Specify Solver Settings
- Solve Problem to find Optimal Solution.

Solver Settings: Target Cell and Changing Cells

- Specify Target Cell: D8.
  - Equal to: Max.
- Changing Cells (decision variables)
  - B5, C5.

Solver Settings: Constraints

- Click “Add” to add constraints.
- Select LHS Cell (D11), relationship (<=), and RHS Cell (F11).
- LHS Cell should contain a formula which computes the LHS Value of the constraint.
- Typically, RHS Cell should contain a fixed value, but this is not absolutely required.
- Repeat for the other two constraints (for labor and machine time).
- OR use the range for the LHS and the RHS (see next slide).

Solver Options

- Select Simples LP”
- Tells Solver to use the Simplex Method, which is faster and is Solver’s default optimization method.
- Check “Make unconstrained Variables Non-negative”.
- Tells Solver that the decision variables (B5:C5, representing the number of Camshafts and Gears) must be ≥ 0 in any feasible solution.
- Leave other settings at their defaults.

Completed Solver Box

- Click “Solve” to tell Solver to find the Optimal Solution.

Solved Spreadsheet (Optimal Solution)

- Optimal Solution: Make 100 camshafts, 350 gears.
- Optimal Objective Value: $8800 profit.
- Both pieces of information are important. Knowing the optimal objective value is useless without knowing how that value can be attained.

	A	B	C	D	E	F
1	**Example B.1**					
2	**DJJ Enterprises Production Planning**					
3						
4	**Decision Variables**	**Camshafts**	**Gears**			
5	Units to Make	100	350			
6						
7	**Objective**			**Total**		
8	Profit	$25	$18	**$8,800**		
9						
10	**Constraints**			**Used**		**Available**
11	Steel (lbs)	5	8	3300	<=	5000
12	Labor (hrs)	1	4	1500	<=	1500
13	Machine Time (hrs)	3	2	1000	<=	1000

Interpreting the Solution

- Which constraints are actively holding us back from making even more profit?
  - Binding Constraints: Constraints at the optimal solution with the LHS equal to the RHS; equivalently for a resource, a binding constraint is one in which all the resource is used up.
  - Labor (all 1500 hours used), Machine Time (all 1000 hours used).
- Non-Binding Constraints:
  - Steel: Have 5000 lbs. available, but only need 3300 lbs. for this solution.
- What-If Analysis. Once the Spreadsheet and Solver model is set up, it is easy to change one or more input values and re-solve the problem.
  - This interactive use can be a very powerful way to use optimization.

Solution Reports

- Solver can generate three solution reports:
  - Answer Report.
  - Sensitivity Report.
  - Limits Report: Not covered here.
- The Answer Report presents in a standard format the Solver Settings and the optimal solution.
- The Sensitivity Report shows what will happen if certain problem parameters are changed from their current values.

Answer Report

- Tightening a binding constraint can only worsen the objective function value, and loosening a binding constraint can only improve the objective function value. As such, once an optimal solution is found, managers can seek to improve that solution by finding ways to relax binding constraints.
- Three sections: Objective Cell, Changing Cells, Constraints.
- "Final Value" indicates optimal solution.
- For each constraint, binding/non-binding status, and "slack" (difference between LHS and RHS), slack is zero for binding constraints.
- Also note that the Solver Settings for Objective Cell, Changing Cells, and Constraints are reported here. This can be a useful debugging tool.

Objective Cell (Max)

Cell	Name	Original Value	Final Value
$D$8	Profit Total	$0	$8,800

Variable Cells

Cell	Name	Original Value	Final Value	Integer
$B$5	Units to Make Camshafts	0	100	Contin
$C$5	Units to Make Gears	0	350	Contin

Constraints

Cell	Name	Cell Value	Formula	Status	Slack
$D$11	Steel (lbs) Used	3300	$D$11<=$F$11	Not Binding	1700
$D$12	Labor (hrs) Used	1500	$D$12<=$F$12	Binding	0
$D$13	Machine Time (hrs) Used	1000	$D$13<=$F$13	Binding	0

Sensitivity Report

Variable Cells

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
$B$5	Units to Make Camshafts	100	0	25	2	20.5
$C$5	Units to Make Gears	350	0	18	82	1.333333333

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
$D$11	Steel (lbs) Used	3300	0	5000	1E+30	1700
$D$12	Labor (hrs) Used	1500	0.4	1500	500	1166.666667
$D$13	Machine Time (hrs) Used	1000	8.2	1000	1416.666667	250

- Note two sections:
  - Variable Cells.
  - Constraints.
- Constraint Section
  - Shadow Price: This is the amount the optimal objective value will change by if the RHS of the constraint is increased by one unit.
  - Units of shadow price: (objective function units/constraint units). Example:
    - For the steel constraint, the units are $/lb.
    - For the labor constraint, the units are $/hour.

    - Allow able Increase/Decrease: Provides the increase/decrease of the RHS of the constraint for which the shadow price stays the same; that is, the effect on the objective value stays the same in this range.
- Example:
    - Shadow Price of Machine Time is $8.20/hour, valid for an increase of 1416 hours or a decrease of 250 hours.
    - If we make 200 additional machine time hours available, profit can be increased by (200 hours) ($8.20/hour) = $1640.
- What is the shadow price of a non-binding constraint? Why? Will this always be the case?
- Variable Cells Section
- Allowable Increase/Decrease: This is the amount the objective coefficient for a decision variable can be increased/decreased without changing the optimal solution.
- Example
    - Gears currently have a profit of $18/unit (objective coefficient).
    - Allowable Increase/Decrease is $82 and $1.33, respectively.
    - Interpretation: If the unit profit for gears is between $16.67 and $100, the optimal solution (production plan) will be to make 100 camshafts and 350 gears.
    - Note: The optimal solution (production plan) stays the same within this range, but the optimal objective value (profit) changes since the unit profit is changing. So, if the profit on gears goes up to $20/unit, profit will increase by ($20-$18) (350) = $700.

Sensitivity Report: Caveat

- Effects identified in the Sensitivity Report need to be carefully interpreted.
- Specifically, the effect of the change of one parameter (e.g., a RHS value or an objective coefficient) assume that all other parameters of the model stay at their base case values.
- For example, the Sensitivity Report does not tell us what happens if additional quantities of both Labor and Machine Time become available. In this scenario, we would need to enter the new values and re-solve the model.

Highlights

- People use informal “optimization” to make decisions almost every day.
- Organizations use formal optimization methods to address problems across the organization, from optimal pricing to locating a new facility.
- The algebraic formulation of an LP comprises the definitions of the decision variables, an algebraic statement of the objective function, and algebraic statements of the constraints.
- The spreadsheet model for an optimization problem should be guided by the algebraic formulation.
- Solver, an Excel Add -In, can solve both linear and nonlinear problems. This lecture focuses on solving linear problems.
- After solving an LP, you must interpret the results to see if they make sense, fix problems with the model, and find the insights useful for management.
- Solver can generate the Answer and Sensitivity Reports. The Sensitivity Report provides additional information about what happens to the solution when certain coefficients of the problem are changed.

Table of Contents

Lecture 7: Introduction to Data Mining

Introduction

- Data is growing at a phenomenal rate.
- Users expect more sophisticated information.
- How?

UNCOVER HIDDEN INFORMATION
DATA MINING

Data Mining – Definition

- Finding hidden information in a database.
- Fit data to a model.
- Similar terms:
  - Exploratory data analysis.
  - Data driven discovery.
  - Deductive learning.

Data Mining Algorithm

- Objective: Fit Data to a Model:
  - Descriptive.
  - Predictive.
- Preference – Technique to choose the best model.
- Search – Technique to search the data.
  - "Query".

Database Processing vs. Data Mining Processing

• Query
  • Well defined
  • SQL
	• Query
	  • Poorly defined
	  • No precise query language

■ Data
  – Operational data
	■ Data
	  – Not operational data

■ Output
  – Precise
  – Subset of database
	■ Output
	  – Fuzzy
	  – Not a subset of database

Query Examples

- Database:
  - Find all credit applicants with last name of Smith.
  - Identify customers who have purchased more than $10.000 in the last month.
- Data Mining:
  - Find all credit applicants who are poor credit risks (classification).
  - Identify customers with similar buying habits (clustering).
  - Find all items which are frequently purchased with milk (association rules).

Basic Data Mining Tasks

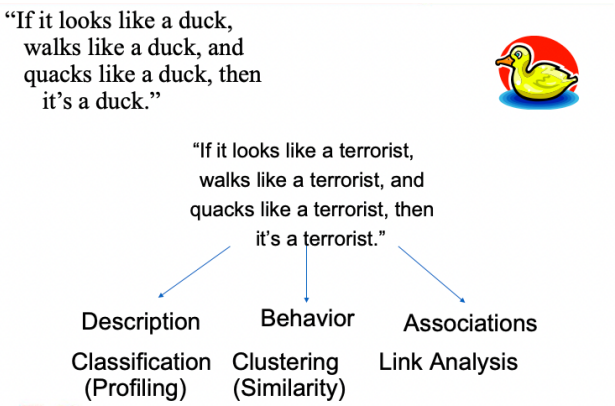
- Classification maps data into predefined groups or classes.
  - Supervised learning.
  - Prediction.
  - Regression.
- Clustering groups similar data together into clusters.
  - Unsupervised learning.
  - Segmentation.
  - Partitioning.
- Link Analysis uncovers relationships among data.
  - Affinity Analysis.
  - Association Rules.
  - Sequential Analysis determines sequential patterns.

Classification

- Assign data into predefined groups or classes.

But It Isn't Magic

- You must know what you are looking for.
- You must know how to look for you.
- Suppose you know that a specific cave had gold:
  - What would you look for?
  - How would you look for it?
  - Might need an expert miner.



Classification Example

- Grading.
- Given a collection of annotated data (in this case 5 instances of Katydids and five of Grasshoppers), decide what type of insect the unlabeled example is.
- The classification problem can now be expressed as:
  - Given a training database, predict the class label of previously unseen instance.
- Facial Recognition.
- Handwriting Recognition.
- Anomaly Detection.

Clustering

- Partition data into previously undefined groups.

Two Types of Clustering

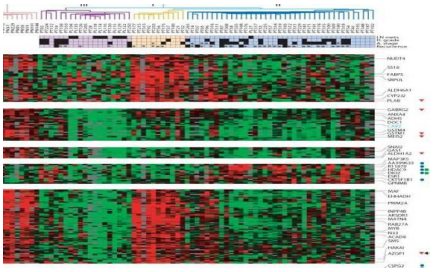
- Hierarchical.
- Partitional.

Hierarchical Clustering Example

- Iris Data Set.

Microarray Data Analysis

- Each probe location associated with gene.
- Color indicates degree of gene expression.
- Compare different samples (normal/disease).
- Track same sample over time.
- Questions:
  - Which genes are related to this disease?
  - Which genes behave in a similar manner?
  - What is the function of a gene?
- Clustering:
  - Hierarchical.
  - K-means.
- Gene Expression Profiling identifiers clinically relevant subtypes of prostate cancer.



Association Rules/Link Analysis

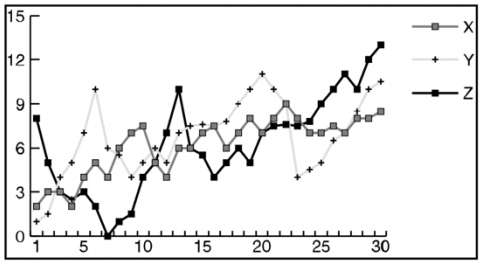
- Find relationships between data.

Association Rules Examples

- People who buy diapers also buy beer.
- If gene A is highly expressed in this disease, then gene A is also expressed.
- Relationships between people.
- Book Stores.
- Department Stores.
- Advertising.
- Product Placement.

Example: Stock Market Analysis

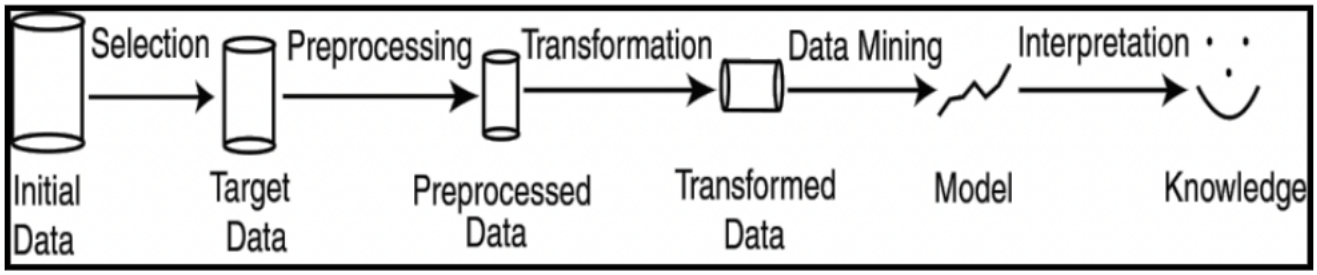
- Example: Stock Market.
- Predict future values.
- Determine similar patterns over time.
- Classify behavior.



Data Mining vs. KDD

- Knowledge Discovery in Databases (KDD): process of finding useful information and patterns in data.
- Data Mining: Use of algorithms to extract the information and patterns derived by the KDD process.

KDD Process



- Selection: Obtain data from various sources.
- Preprocessing: Cleanse data.
- Transformation: Convert to common format. Transform to new format.
- Data Mining: Obtain desired results.
- Interpretation/Evaluation: Present results to user in meaningful manner.

KDD Process Example: Web Log

- Selection:
  - Select log data (dates and locations) to use.
- Preprocessing:
  - Remove identifying URLs. Remove error logs.
- Transformation:
  - Sectionize (Sort and Group).
- Data Mining:
  - Identify and count patterns. Construct data structure.
- Interpretation/Evaluation:
  - Identify and display frequently accessed sequences.
- Potential User Applications:
  - Cache prediction.
  - Personalization.

Related Topics

- Databases.
- OLTP.
- OLAP.
- Information Retrieval.

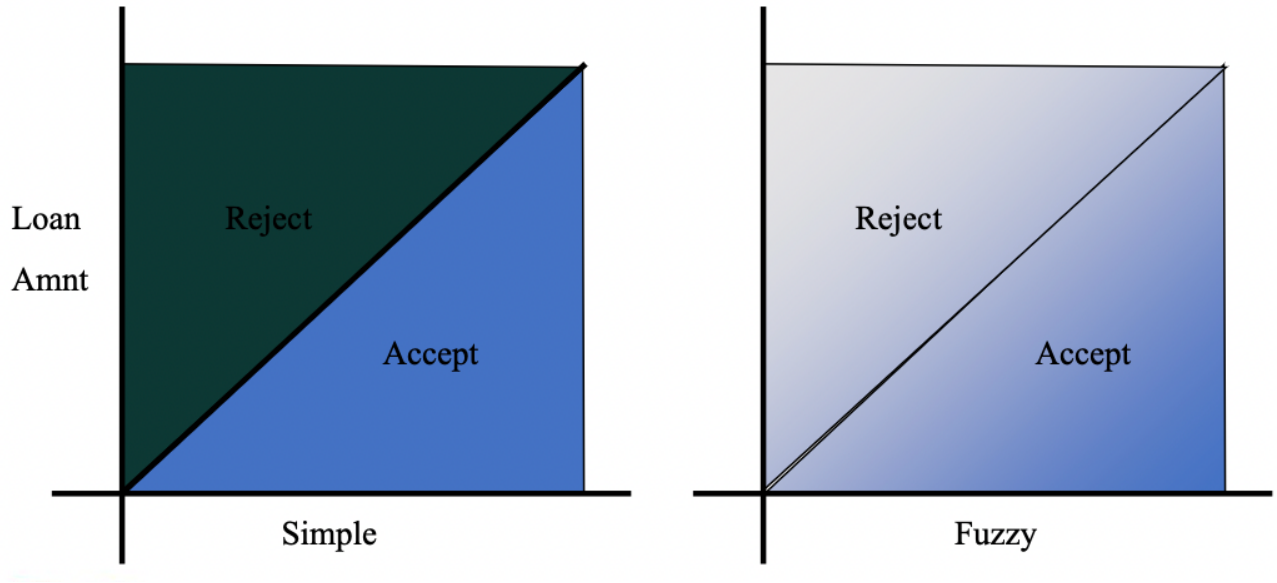
DB & OLTP Systems

- Schema:
  - (ID, Name, Address, Salary, JobNo).
- Data Model:
  - Entity Relationship.
  - Relational.
- Transaction.
- Sample query:

```
SELECT Name
FROM T
WHERE Salary > 100000
```

DM: Only imprecise queries

Classification/Prediction is Fuzzy



Information Retrieval

- Information Retrieval (IR): retrieving desired information from textual data.
- Library Science.
- Digital Libraries.
- Web Search Engines.
- Traditionally keyword based.
- Sample query:
    - Find all documents about “data mining”.

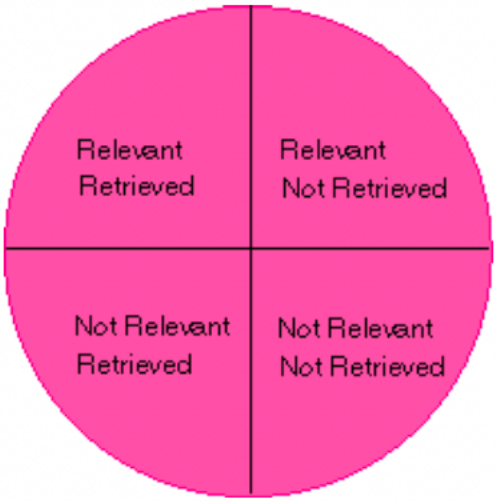
DM: Similarity measures;
Mine text/Web data.

- Similarity: measure of how close a query is to a document.
- Documents which are “close enough” are retrieved.
- Metrics:

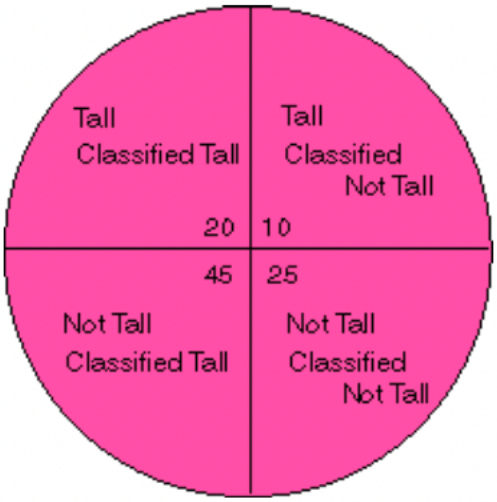
$$\bullet\ \textit{Precision} = \frac{|\text{Relevant and Retrieved}|}{|\text{Retrieved}|}$$

$$\bullet\ \textit{Recall} = \frac{|\text{Relevant and Retrieved}|}{|\text{Relevant}|}$$

IR Query Result Measures and Classification



IR



Classification

OLAP

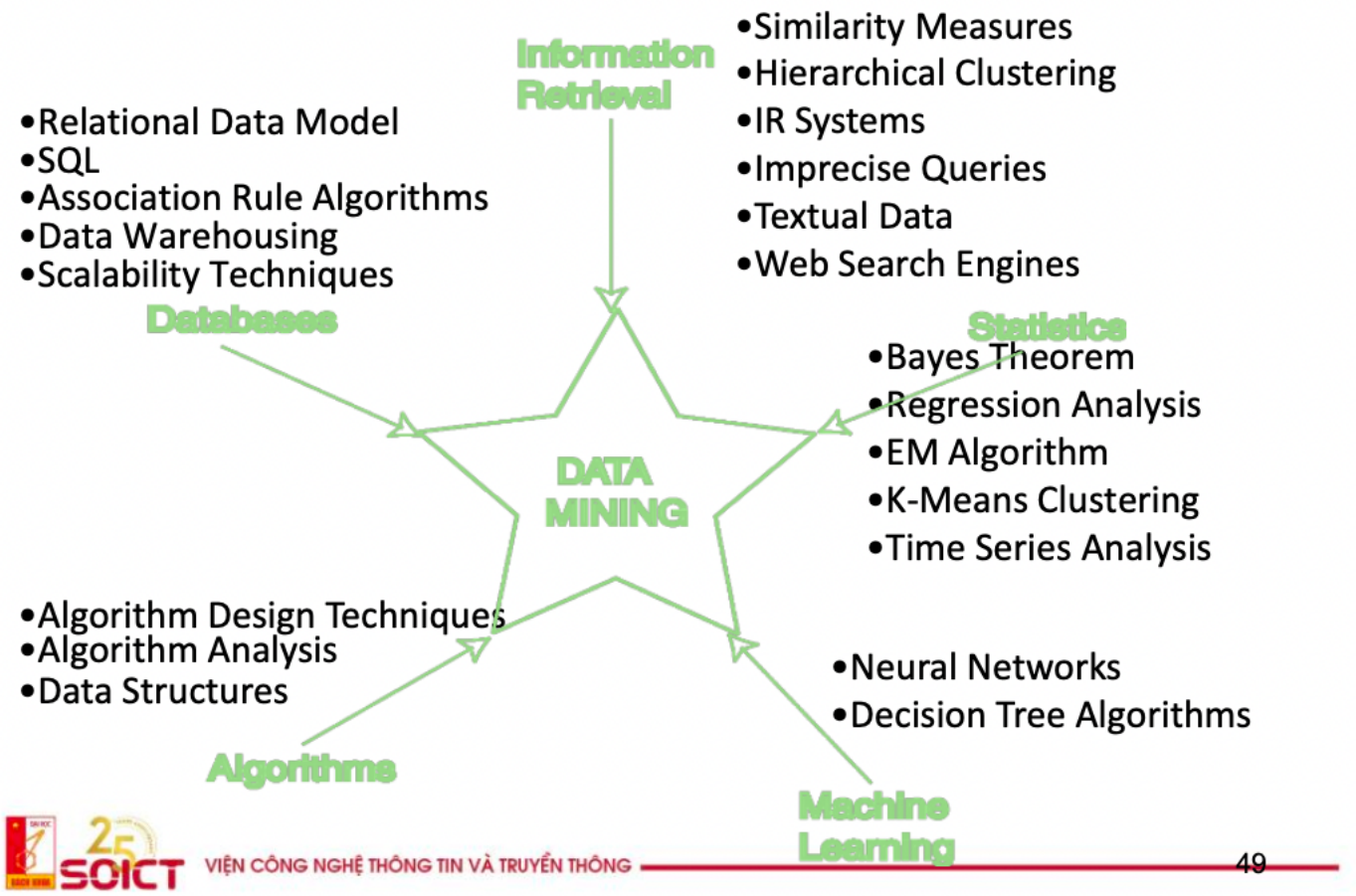
- Online Analytic Processing (OLAP): provides more complex queries than OLTP.
- Online Transaction Processing (OLTP): traditional database/transaction processing.
- Dimensional data; cube view.
- Visualization of operations:
  - Slice: examine sub-cube.
  - Dice: rotate cube to look at another dimension.
  - Roll Up/Drill Down.

DM: May use OLAP queries.

DM vs. Related Topics

Area	Query	Data	Results	Output
DB/OLTP	Precise	Database	Precise	DB Objects or Aggregation
IR	Precise	Documents	Vague	Documents
OLAP	Analysis	Multidimensional	Precise	DB Objects or Aggregation
DM	Vague	Preprocessed	Vague	KDD Objects

Data Mining Development



KDD Issues

- Human Interaction. - Overfitting. - Outliers. - Interpretation. - Visualization. - Large Datasets. - High Dimensionality.	- Multimedia Data. - Irrelevant Data. - Noisy Data. - Changing Data. - Integration. - Application.

Warning

- With data mining, you don’t always know what you are looking for.
- There is not one right answer.
- The data you are using is noisy.
- Data Mining is a very applied discipline.
- A data mining course provides you tools to use to analyze data.
- Experience provides you knowledge of how to use these tools.

Social Implications of DM

- Privacy. - Profiling.
- Unauthorized use. - Invalid results and claims.

Data Mining Metrics

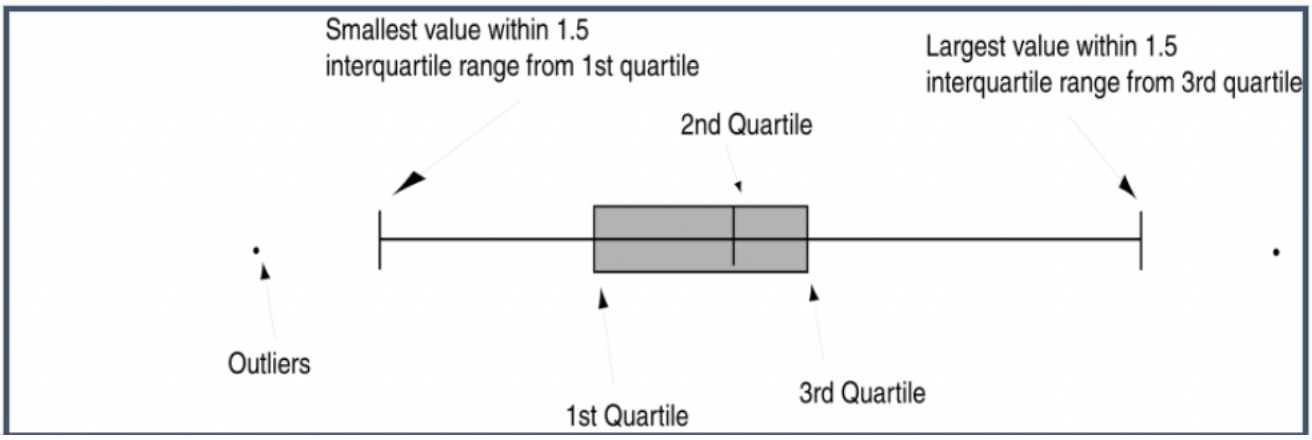
- Usefulness.
- Return on Investment (ROI).
- Accuracy.
- …
- Space/Time.

Visualization Techniques

- Graphical.
- Geometric.
- Icon-based.
- Pixel-based.
- Hierarchical.
- Hybrid.

Models Based on Summarization

- Visualization: Frequency distribution, mean, variance, median, mode, etc.
- Box Plot:



DM Tools

- XLMiner – Easy addin to Excel: http://www.solver.com/xlminer/index.html
- Webka – Open Source; Visualization, Functionality, Interface: http://www.cs.waikato.ac.nz/ml/weka/
- SAS (JMP) – Commercial Product.
- SPSS – Commercial Product.
- MATLAB – Statistical/Math Applications.
- R – Programming.

Table of Contents

Lecture 8: Classification

Classification: Definition

- Given a collection of records (training set).
  - Each record is characterized by a tuple (x, y) where x is the attribute set and y is the class label.
- Task: Learn a model that maps each attribute set x into one of the predefined class labels y.
- Example:

Task	Attribute set, $\boldsymbol{x}$	Class label, y
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from x-rays or MRI scans	malignant or benign cells
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

General Approach for Building Classification Model

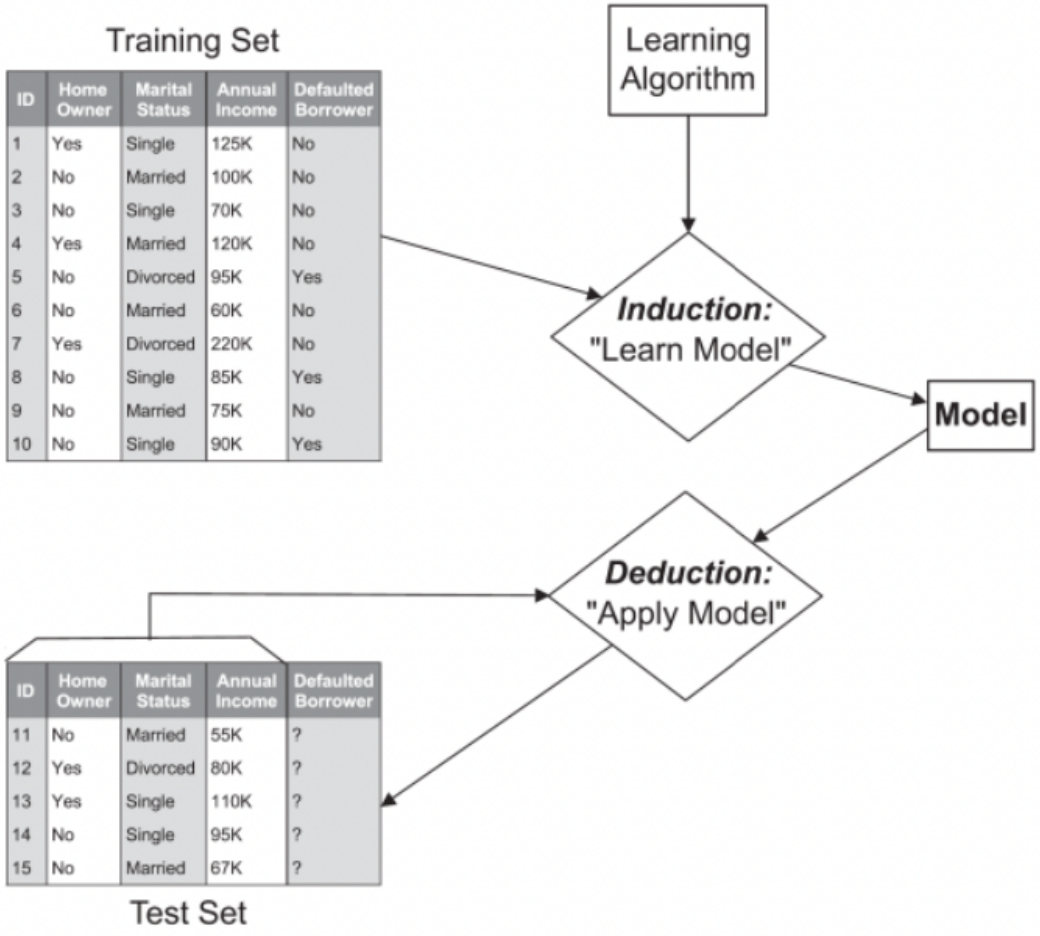


Figure 3.3. General framework for building a classification model.

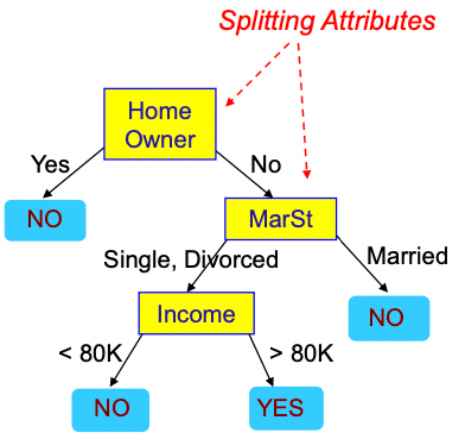
Classification Techniques

- Base Classifiers:
  - Decision Tree based Methods.
  - Rule-based Methods.
  - Nearest-neighbor.
  - Naïve Bayes and Bayesian Belief Networks.
  - Support Vector Machines.
- Ensemble Classifiers:
  - Booting.
  - Bagging.
  - Random Forests.

Example of a Decision Tree

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
	categorical	categorical	continuous	class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

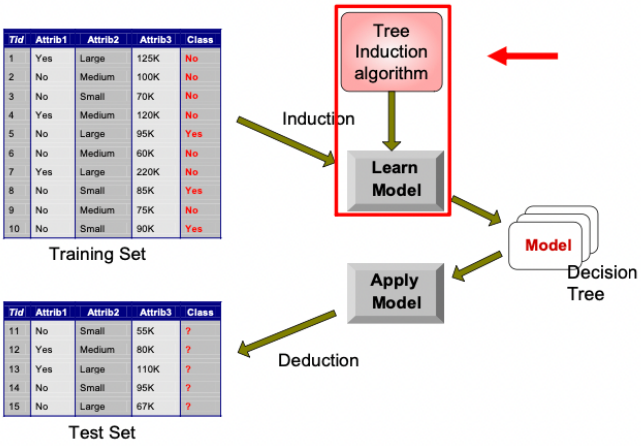
Training Data



Model: Decision Tree

- Apply Model to Test Data?
- There could be more than one tree that fits the same data.

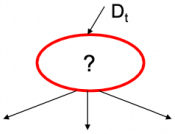
Decision Tree Classification Task



Decision Tree Induction

- Many Algorithms:
  - Hunt's Algorithm (one of the earliest).
  - CART.
  - ID3, C4.5.
  - SLIQ, SPRINT.

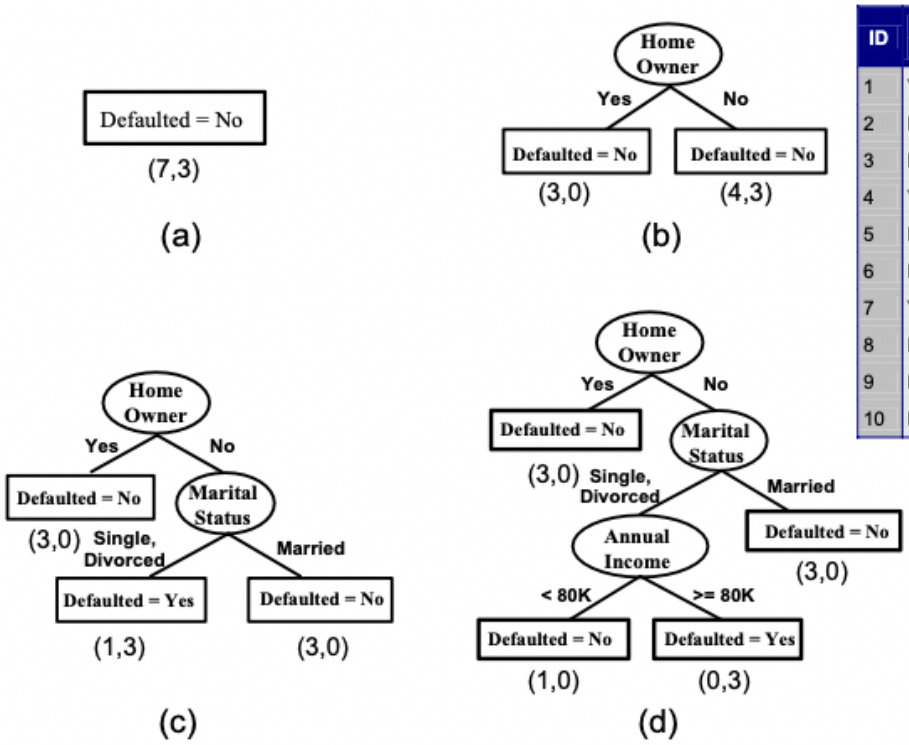
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



General Structure of Hunt's Algorithm

- Let D_t be the set of training records that reach a node t.
- General Procedure:
  - If D_t contains records that belong the same class y_t then t is a leaf node labeled as y_t.
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets.
  - Recursively apply the procedure to each subset.

Hunt's Algorithm



ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Design Issues of Decision Tree Induction

- How should training records be split?
  - Method for expressing test condition.
    - Depending on attribute types.
  - Measure for evaluating the goodness of a test condition.
- How should the splitting procedure stop?
  - Stop splitting if all the records belong to the same class or have identical attribute values.
  - Early termination.

Methods for Expressing Test Conditions

- Depends on attribute types.
  - Binary.
  - Nominal.
  - Ordinal.
  - Continuous.

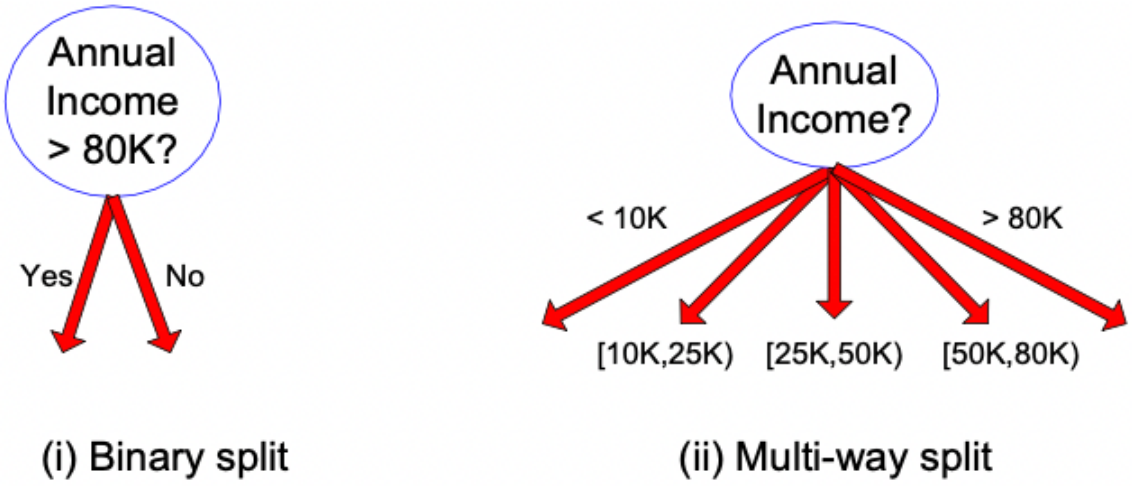
Test Condition for Nominal Attributes

- Multi-way split:
  - Use as many partitions as distinct values.
- Binary split:
  - Divides values into 2 subsets.

Test Condition for Ordinal Attributes

- Multi-way split:
  - Use as many partitions as distinct values.
- Binary split:
  - Divides values into 2 subsets.
  - Preserve order property among attribute values.

Test Condition for Continuous Attributes



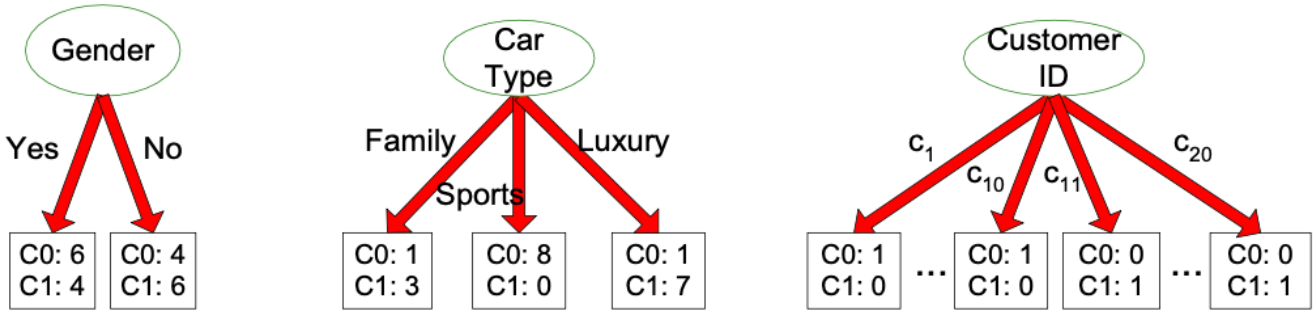
Splitting Based on Continuous – Attributes

- Different ways of handling:
  - Discretization to form an ordinal categorical attribute.
    - Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
      - Statistic: discretize once at beginning.
      - Dynamic: repeat at each node.
  - Binary Decision: $(A < v)$ or $(A \geq v)$.
    - Consider all possible splits and finds the best cut.
    - Can be more compute intensive.

How to determine the Best Split

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

- Before Splitting: 10 records of class 0, 10 records class of 1.



- What test condition is the best?
- Greedy approach:
  - Nodes with purer class distribution are preferred.
- Need a measure of node impurity.

C0: 5
C1: 5

High degree of impurity

C0: 9
C1: 1

Low degree of impurity

Measures of Node Impurity

- Gini Index

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where $\boldsymbol{p_i(t)}$ is the frequency of class $\boldsymbol{i}$ at node t, and $\boldsymbol{c}$ is the total number of classes

- Entropy

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

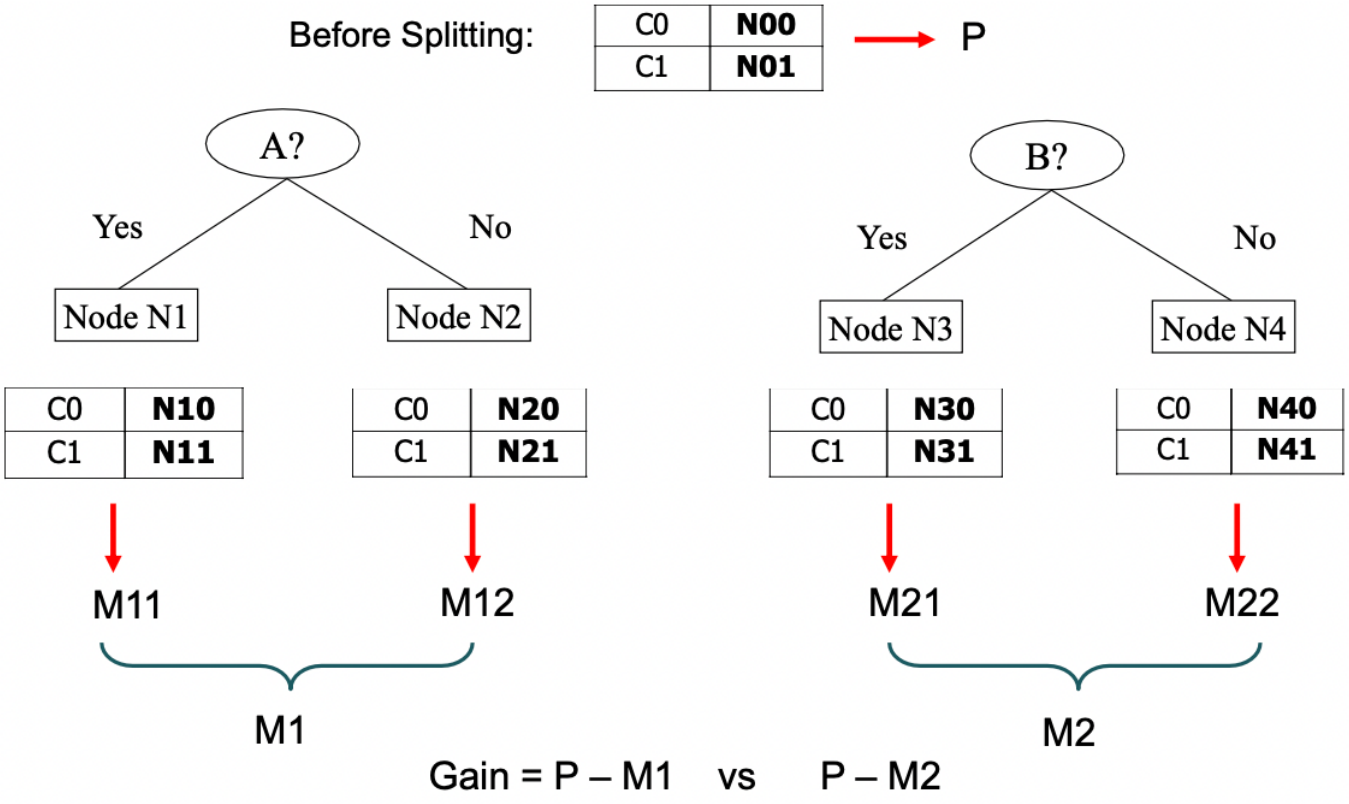
- Misclassification error

$$Classification\ error = 1 - \max[p_i(t)]$$

Finding the Best Split

1. Compute impurity measure (P) before splitting.
2. Compute impurity measure (M) after splitting.
   - Compute impurity measure of each child node.
   - M is the weighted impurity of child nodes.
3. Choose the attribute test condition that produces the highest gain.
   - Gain = P – M

   Or equivalently, lowest impurity measure after splitting (M).



Measure of Impurity

- Gini Index for a given node t

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where $p_i(t)$ is the frequency of class i at node t, and c is the total number of classes

- Maximum of 1-1/c when records are equally distributed among all classes, implying the least beneficial situation for classification.
- Minimum of 0 when all records belong to one class, implying the most beneficial situation for classification.
- Gini index is used in decision tree algorithms such as CART, SLIQ, SPRINT.
- For 2-class problem (p, 1-p):
  - GINI = 1 - p^2 - $(1-p)^2$ = 2p(1-p)

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Computing GINI Index of a Single Node

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

C1	0
C2	6

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – $P(C1)^2$ – $P(C2)^2$ = 1 – 0 – 1 = 0

C1	1
C2	5

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

C1	2
C2	4

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

Computing GINI Index for a Collection of Nodes

When a node p is split into k partitions (children)

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

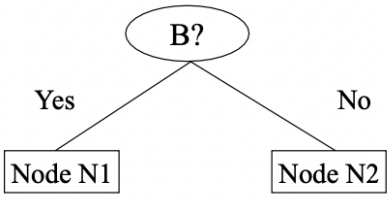
where, n_i = number of records at child i,

n = number of records at parent node p.

Binary Attributes: Computing GINI Index

- Splits into 2 partitions (child nodes).
- Effect of Weight partitions:
  - Larger and purer partitions are sought.
- Gini(N1) = $1 - (5/6)^2 - (1/6)^2 = 0.278$
- Gini(N2) = $1 - (2/6)^2 - (4/6)^2 = 0.444$

	Parent
C1	7
C2	5
Gini = 0.486	



	N1	N2
C1	5	2
C2	1	4
Gini=0.361		

Weighted Gini of N1 N2
= 6/12 * 0.278 +
6/12 * 0.444
= 0.361

Gain = 0.486 – 0.361 = 0.125

Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the dataset.
- Use the count matrix to make decisions.

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	0.163		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

- Which of these is the best?

Continuous Attributes: Computing Gini Index

- Use Binary Decision based on one value.
- Several Choices for the splitting value.
  - Number of possible splitting values = Number of distinct values.
- Each Splitting value has a count matrix associate with it.
  - Class counts in each of the partitions, A ≤ v and A > v.
- Simple method to choose the best v:
  - For each v, scan the database to gather count matrix and compute its Gini index.
  - Computationally inefficient! Repetition of work.
- For efficient computation: For each attribute,
  - Sort the attribute on values.
  - Linearly scan these values, each time updating the count matrix and computing GINI index.
  - Choose the split position that has the least GINI index.

ID	Home Owner	Marital Status	Annual Income	Defaulted
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Cheat	No		No		No		Yes		Yes		Yes		No		No		No		No	
Annual Income																				
Sorted Values →	60		70		75		85		90		95		100		120		125		220	
Split Positions →	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.400		0.420	

Measure of Impurity: Entropy

- At each node t:

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

- Where $p_i(t)$ is the frequency of class i at node t, and c is total number of classes.
  - Maximum of $log_2 c$ when records are equally distributed among all classes, implying the least beneficial situation for classification.
  - Minimum of 0 when all records belong to one class, implying most beneficial situation of classification.
- Entropy based computations are quite like the GINI index computations.

Computing Entropy of a Single Node

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Computing Information Gain After Splitting

- Information Gain:

$$Gain_{split} = Entropy(p) - \sum_{i=1}^{k} \frac{n_i}{n} Entropy(i)$$

  - Parent Node, p is split into k partitions (children).
  - n_i is number of records in child node i.
- Choose the split that achieve most reduction (maximizes GAIN).
- Used in the ID3 and C4.5 decision tree algorithm.
- Information gain is the mutual information between the class variable and the splitting variable.

Problem with large number of partitions

- Node impurity measures tend to prefer splits that result in large number of partitions, each being small but pure.
- Customer ID has highest information gain because entropy for all the children is zero.

Gain Ratio

$$Gain\ Ratio = \frac{Gain_{split}}{Split\ Info} \qquad Split\ Info = -\sum_{i=1}^{k} \frac{n_i}{n} log_2 \frac{n_i}{n}$$

  - Parent Node, p is split into k partitions (children).
  - n_i is the number of records in child node i.
- Adjusts Information Gain by the entropy of the partitioning (Split Info).

- Higher entropy partitioning (large number of small partitions) is penalized.
- Used in C4.5 algorithm.
- Designed to overcome the disadvantage of Information Gain.

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	0.163		

SplitINFO = 1.52

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

SplitINFO = 0.72

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

SplitINFO = 0.97

Measure of Impurity: Classification Error

- Classification error at a node t:

$$Error(t) = 1 - \max_i [p_i(t)]$$

- Maximum of 1 – 1/c when records are equally distributed among all classes, implying the least interesting situation.
- Minimum of 0 when all records belong to one class, implying the most interesting situation.

Computing Error of a Single Node

$$Error(t) = 1 - \max_i [p_i(t)]$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

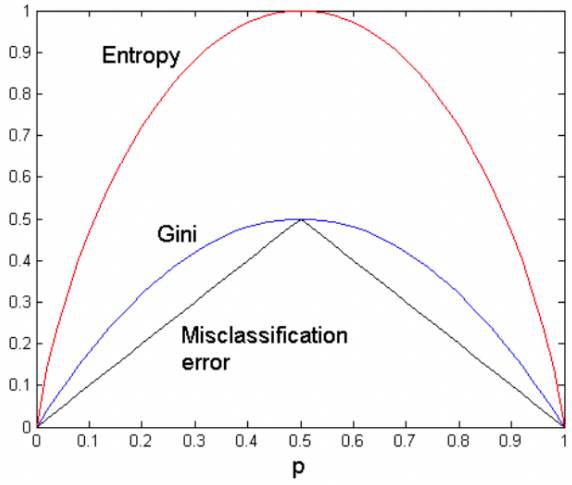
C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

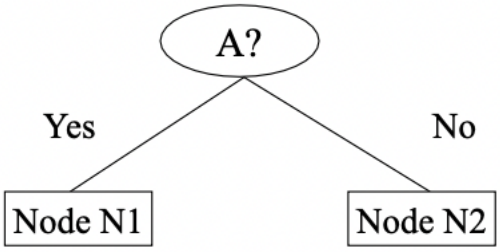
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Impurity Measures

For a 2-class problem:



Misclassification Error vs. Gini Index



	Parent
C1	7
C2	3
Gini = 0.42	

	N1	N2
C1	3	4
C2	0	3
Gini=0.342		

- Gini improves but error maintains the same.

	N1	N2
C1	3	4
C2	0	3
Gini=0.342		

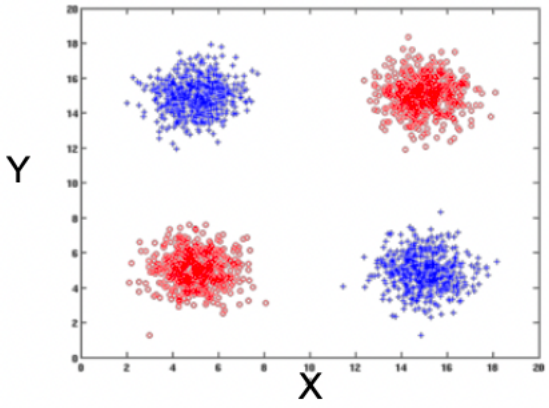
	N1	N2
C1	3	4
C2	1	2
Gini=0.416		

- Misclassification error for all three cases = 0.3!

Decision Tree Based Classification

- Advantages:
  - Relatively inexpensive to construct.
  - Extremely fast at classifying unknown records.
  - Easy to interpret for small-sized trees.
  - Robust to noise (especially when methods to avoid overfitting are employed).
  - Can easily handle redundant attributes.
  - Can easily handle irrelevant attributes (unless the attributes are interacting).
- Disadvantages:
  - Due to the greedy nature of splitting criterion, interacting attributes (that can distinguish between classes together but not individually) may be passed over in favor of other attributed that are less discriminating.
  - Each decision boundary involves only a single attribute.

Handling Interactions

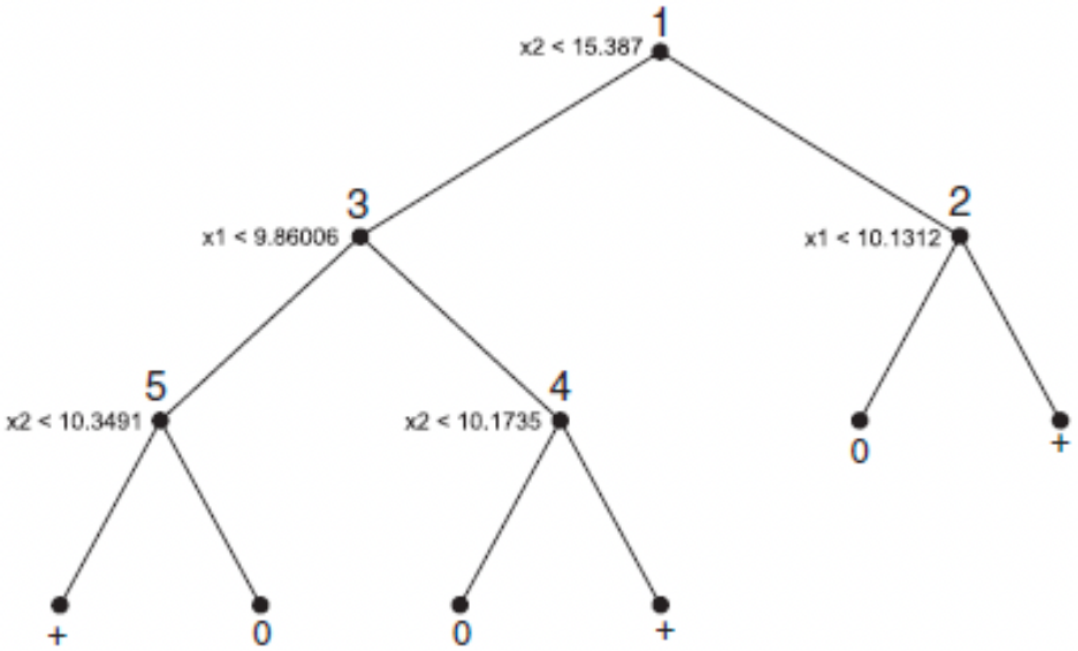


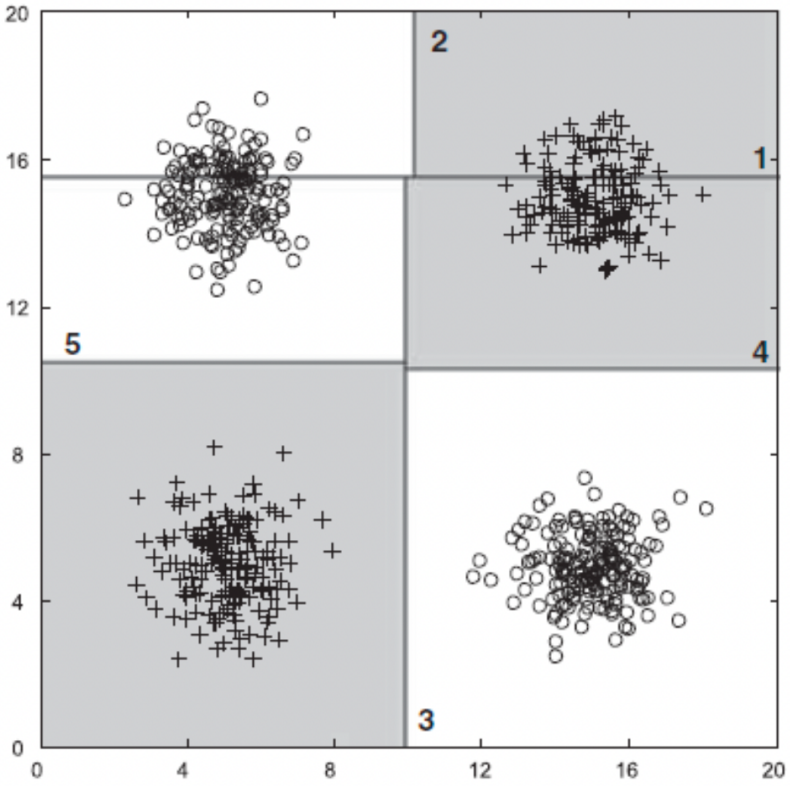
+ : 1000 instances

o : 1000 instances

Entropy (X) : 0.99
Entropy (Y) : 0.99

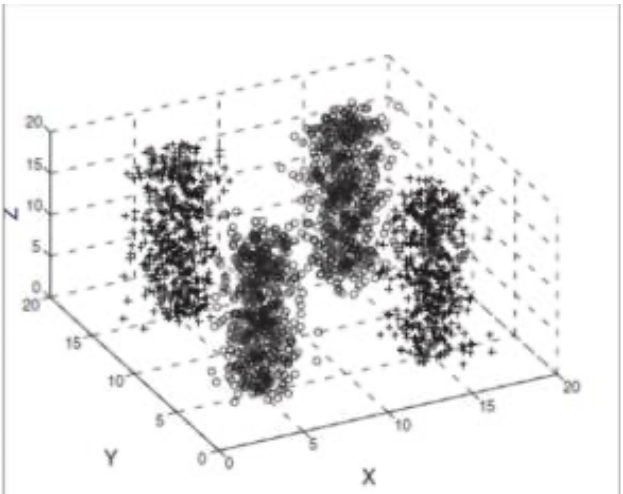
Handling Interactions – 6 leaf nodes





Handling Interactions Given Irrelevant Attributes

- Adding Z as a noisy attribute generated from a uniform distribution.
- Attribute Z will be chosen for splitting.



(a) Three-dimensional data with attributes X, Y, and Z.

Entropy (X) : 0.99
Entropy (Y) : 0.99
Entropy (Z) : 0.98

- Both positive (+) and negative (o) classes generated from skewed Gaussians with centers at (8, 8) and (12, 12) respectively.

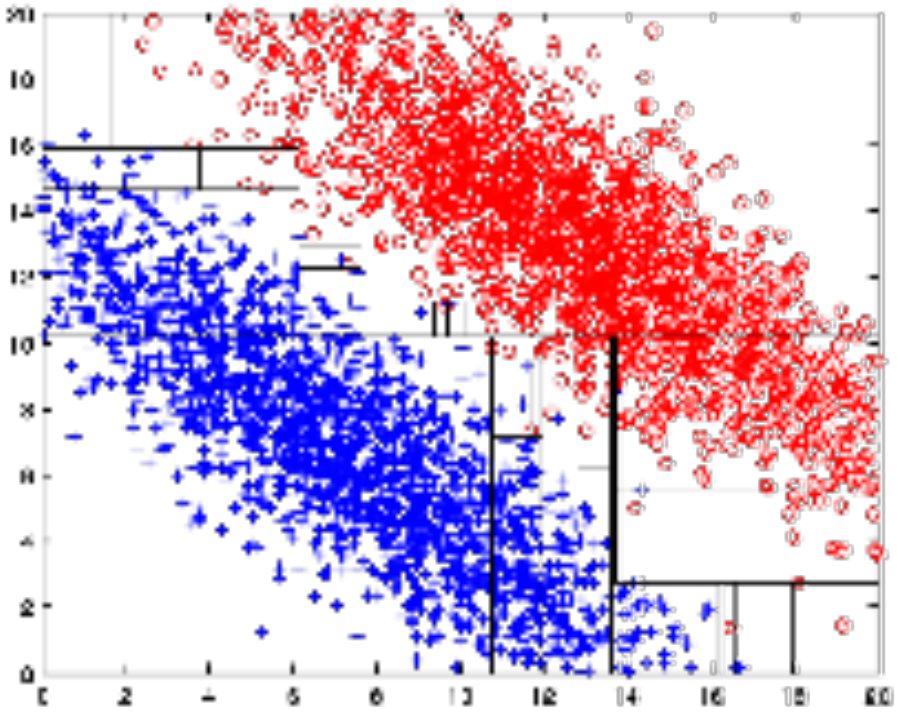


Table of Contents

Data Mining: Cluster Analysis: Basic Concepts and Methods

What is Cluster Analysis?

- Cluster: A collection of data objects.
  - Similar (or related) to one another within the same group.
  - Dissimilar (or unrelated) to the objects in other groups.
- Cluster Analysis (or clustering, data segmentation, etc.):
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters.
- Unsupervised learning: no predefined classes (i.e., *learning by observations* vs. *learning by examples*: supervised).
- Typical applications:
  - As a stand-alone tool to get insight into data distribution.
  - As a preprocessing step for other algorithms.

Applications of Cluster Analysis

- Data reduction:
  - Summarization: Preprocessing for regression, PCA, classification, and association analysis.
  - Compression: Image processing: vector quantization.
- Hypothesis generation and testing.
- Prediction based on groups.
  - Cluster & find characteristics/patterns for each group.
- Finding K-nearest Neighbors.
  - Localizing search to one or a small number of clusters.
- Outlier detection: Outliers are often viewed as those “far away” from any cluster.

Clustering: Application Examples

- Biology: taxonomy of living things: kingdom, phylum, class, order, family, genus, and species.
- Information retrieval: document clustering.
- Land use: Identification of areas of similar land use in an earth observation database.
- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs.
- City-planning: Identifying groups of houses according to their house type, value, and geographical location.
- Earth-quake studies: Observed earthquake epicenters should be clustered along continent faults.
- Climate: understanding earth climate, find patterns of atmospheric and ocean.
- Economic Science: market research.

Basic Steps to Develop a Cluster Task

- Feature Selection:
  - Select info concerning the task of interest.
  - Minimal information redundancy.
- Proximity measure:
  - Similarity of two feature vectors.
- Clustering criterion:
  - Expressed via a cost function or some rules.
- Clustering algorithms:
  - Choice of algorithms.
- Validation of the results:
  - Validation test (also, clustering tendency test).
- Interpretation of the results:
  - Integration with applications.

Quality: What is Good Clustering?

- A good clustering method will produce high quality clusters:
  - High intra-class similarity: cohesive within clusters.
  - Low inter-class similarity: distinctive between clusters.
- The quality of a clustering method depends on:
  - The similarity measure used by the method.
  - Its implementation, and
  - Its ability to discover some or all the hidden patterns.

Measure the Quality of Clustering

- Dissimilarity/Similarity metric:
  - Similarity is expressed in terms of a distance function, typically metric: d(i, j).
  - The definitions of distance functions are usually rather different for interval-scaled, Boolean, categorical, ordinal ratio, and vector variables.
  - Weights should be associated with different variables based on applications and data semantics.
- Quality of clustering:
  - There is usually a separate “quality” function that measures the “goodness” of a cluster.
  - It is hard to define “similar enough” or “good enough”. The answer is typically highly subjective.

Considerations for Cluster Analysis

- Partitioning criteria:
  - Single level.
  - Hierarchical partitioning (often, multi-level hierarchical partitioning is desirable).
- Separation of clusters:
  - Exclusive (e.g., one customer belongs to only one region).
  - Non-exclusive (e.g., one document may belong to more than one class).
- Similarity measure:
  - Distance-based (Euclidian, road network, vector).
  - Connectivity-based (density, contiguity).
- Clustering space:
  - Full space (often when low dimensional).
  - Subspaces (often in high-dimensional clustering).

Requirements and Challenges

- Scalability:
  - Clustering all the data instead of only on samples.
- Ability to deal with different types of attributes:
  - Numerical, binary, categorical, ordinal, linked, and mixture of these.
- Constraint-based clustering:
  - User may give inputs on constraints.
  - Use domain knowledge to determine input parameters.
- Interpretability and usability.
- Others:
  - Discovery of clusters with arbitrary shape.
  - Ability to deal with noisy data.
  - Incremental clustering and insensitivity to input order.
  - High dimensionality.

Major Clustering Approaches

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion.
  - E.g., minimizing the sum of square errors.
  - Typical methods: k-means, k-medoids, CLARANS.
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion.
  - Typical methods: Diana, Agnes, BIRCH, CAMELEON.
- Density-based approach:
  - Based on connectivity and density functions.
  - Typical methods: DBSACN, OPTICS, Den Clue.
- Grid-based approach:
  - Based on a multiple-level granularity structure.
  - Typical methods: STING, Wave Cluster, CLIQUE.
- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other.
  - Typical methods: EM, SOM, COBWEB.
- Frequent pattern-based:
  - Based on the analysis of frequent patterns.
  - Typical methods: p-Cluster.
- User-guided or constraint-based:
  - Clustering by considering user-specified or application-specific constraints.
  - Typical methods: COD (obstacles), constrained clustering.
- Link-based clustering:
  - Objects are often linked together in various ways.
  - Massive links can be used to cluster objects: SimRank, LinkClus.

Partitioning Algorithms: Basic Concept

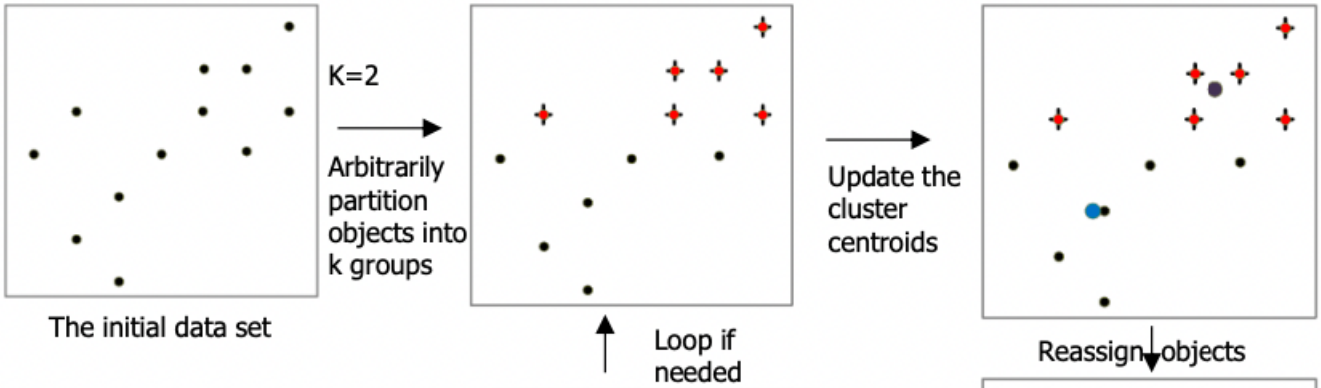
- Partitioning method: Partitioning a database D of n objects into a set of k clusters, such that the sum of squared distances is minimized (where c_i is the centroid of medoid of cluster C_i).

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} (d(p, c_i))^2$$

- Given k, find a partition of k clusters that optimized the chosen partitioning criterion.
  - Global optimal: exhaustively enumerate all partitions.
  - Heuristic methods: k-means and k-medoids algorithms.
  - K-means (MacQueen'67, Klogd'57/'82): Each cluster is represented by the center of the cluster.
  - K-medoids or PAM (Partition Around Medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster.

The K-Means Clustering Method.

- Given k, the k-means algorithm is implemented in four steps:
  - Partition objects into k non-empty subsets.
  - Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., mean point of the cluster).
  - Assign each object to the cluster with the nearest seed point.
  - Go back to step 2, stop when the assignment does not change.

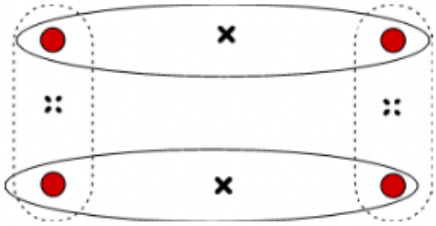


- Partition objects into *k* nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

Comments on the K-means Method

- Strength: Efficient O(t*k*n), where n is number of objects, k is number of clusters, and t is number of iterations. Normally k, t ≪ n.
    - PAM: $O(k(n-k)^2)$
    - CLARA: $O(ks^2 + k(n-k))$
- Comment: Often terminates at a local optimal.
- Weakness:
    - Applicable only to objects in a continuous n-dimensional space.
        - Using the k-modes method for categorical data.
        - In comparison, k-medoids can be applied to a wide range of data.
    - Need to specify k_i the number of clusters, in advance (there are ways to automatically determine the best k (see Hastie et al., 2009)).
    - Sensitive to noisy data and outliers.
    - Not suitable to discover clusters with non-convex shapes.

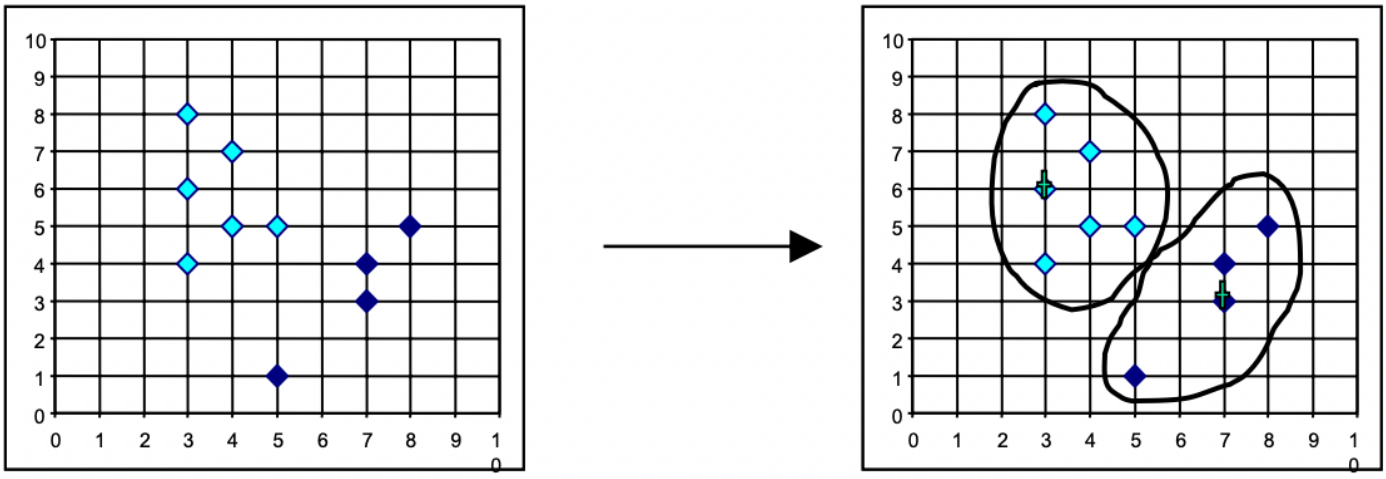
Variations of the K-Means Method



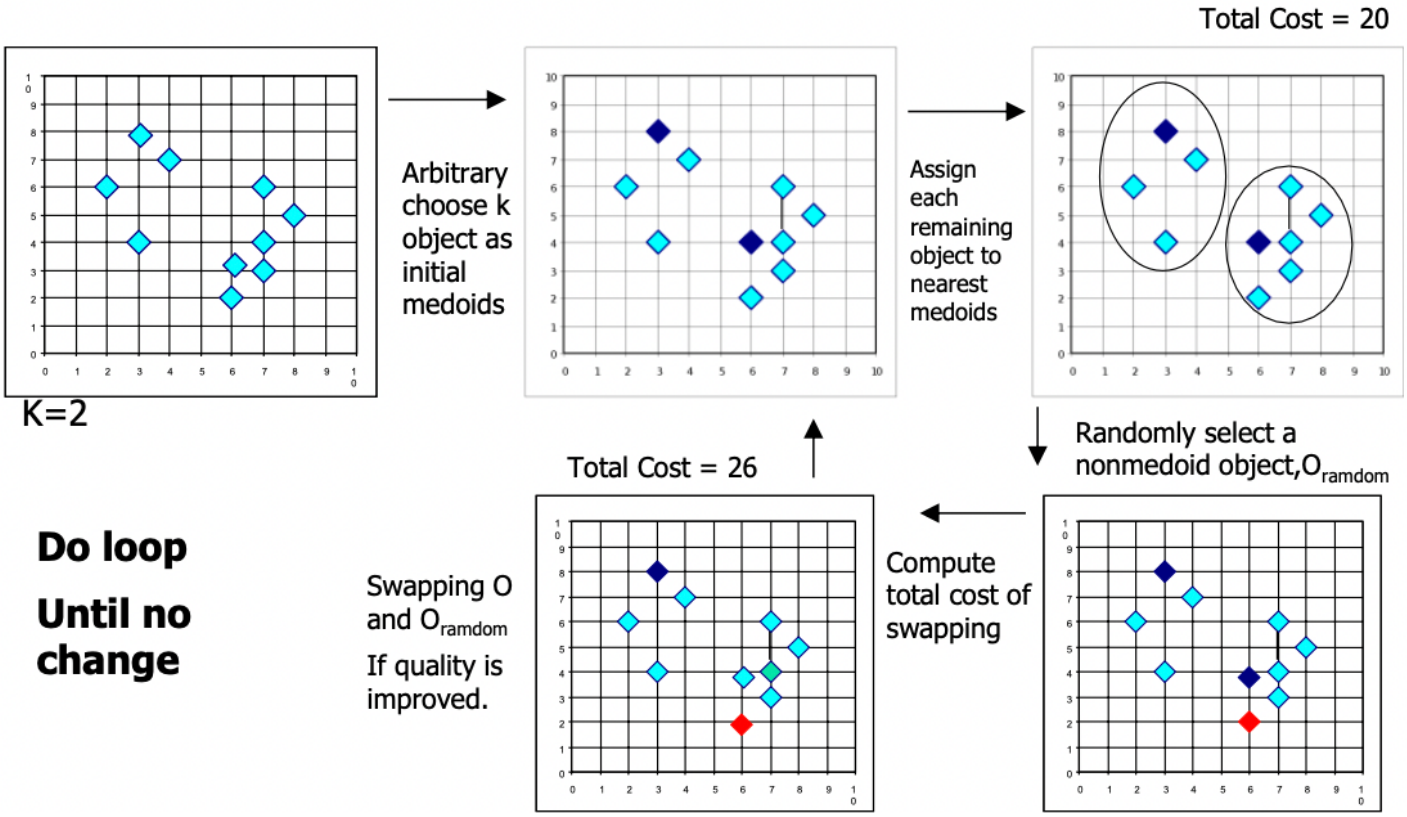
- Most of the variants of the k-means which differ in:
    - Selection of the initial k means.
    - Dissimilarity calculations.
    - Strategies to calculate cluster means.
- Handling categorical data: k-modes.
    - Replacing means of clusters with modes.
    - Using new dissimilarity measures to deal with categorical objects.
    - Using a frequency-based method to update modes of clusters.
    - A mixture of categorical and numerical data: k-prototype method.

What is the Problem of the K-Means Method

- The k-means algorithm is sensitive to outliers.
    - Since an object with an extremely large value may substantially distort the distribution of the data.
- K-Medoids: Instead of taking the mean value of the object in a cluster as a reference point, medoids can be used, which is the most centrally located object in a cluster.

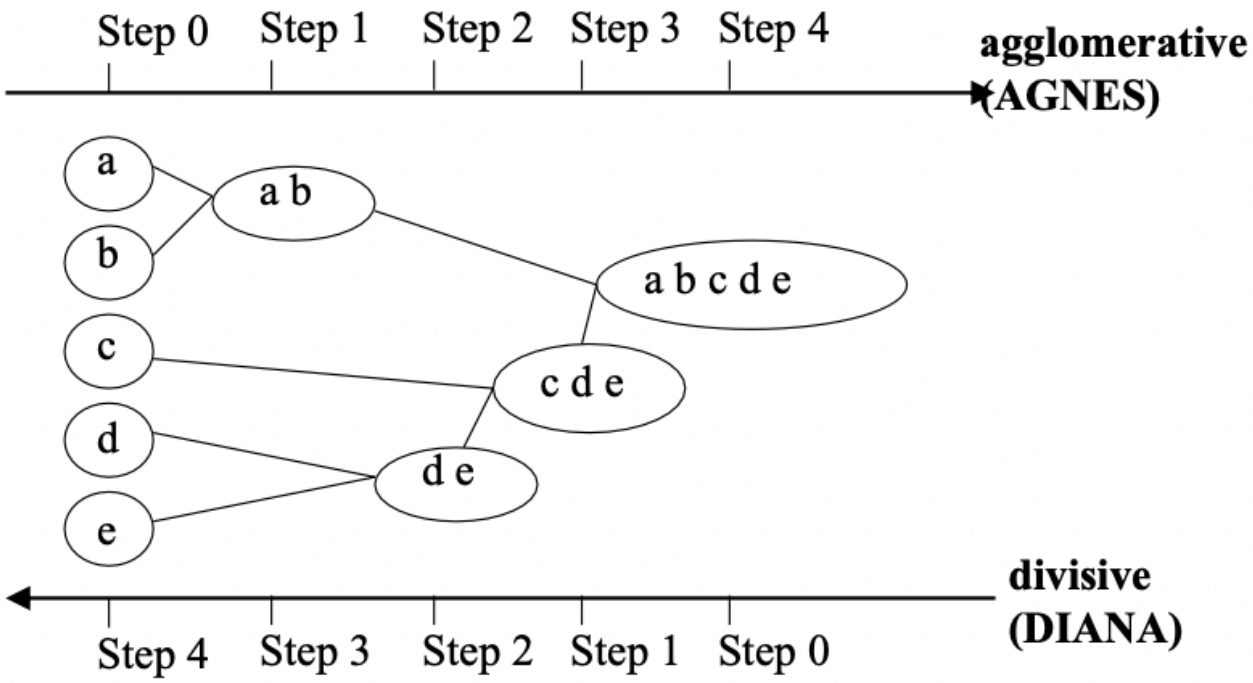


PAM: A Typical K-Medoids Algorithm



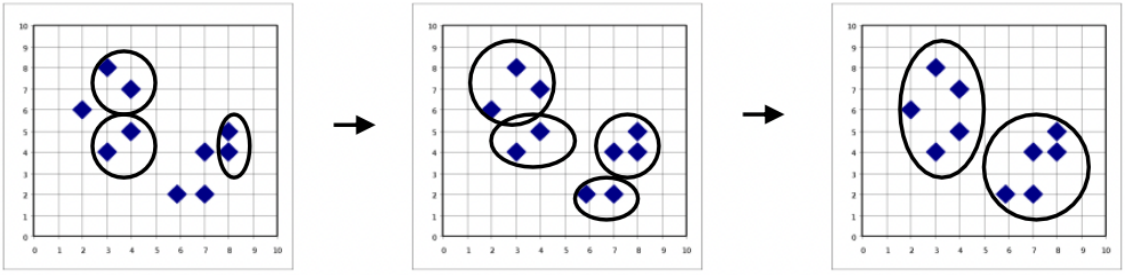
Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters k as an input but needs a termination condition.

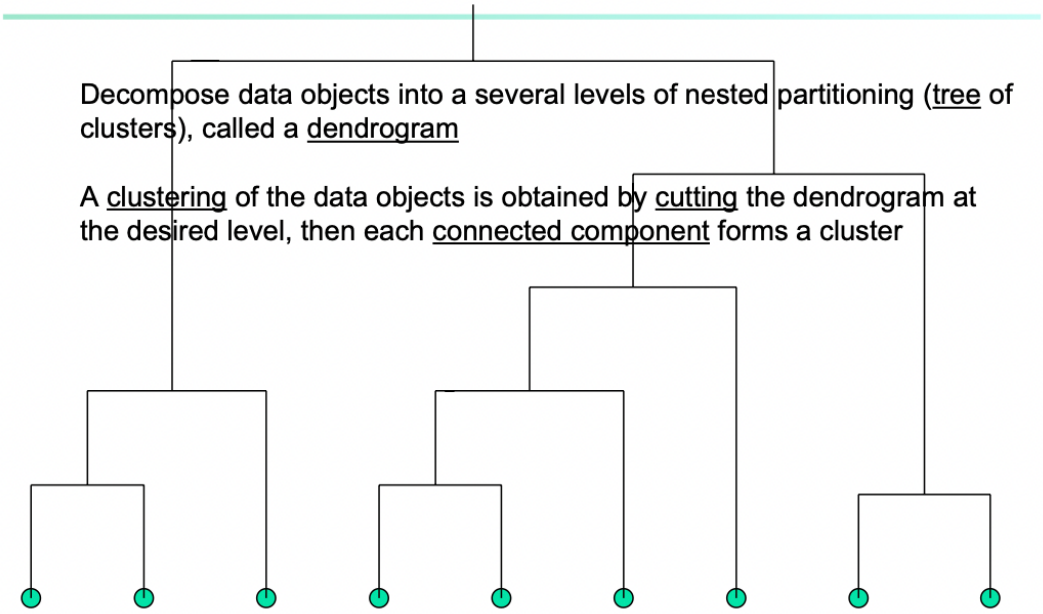


AGNES (Agglomerative Nesting)

- Introduced in Kaufmann and Rousseau (1990).
- Implemented in statistical packages, e.g., Splus.
- Use the single-link method and the dissimilarity matrix.
- Merge nodes that have the least dissimilarity.
- Go on in a non-descending fashion.
- Eventually all nodes belong to the same cluster.

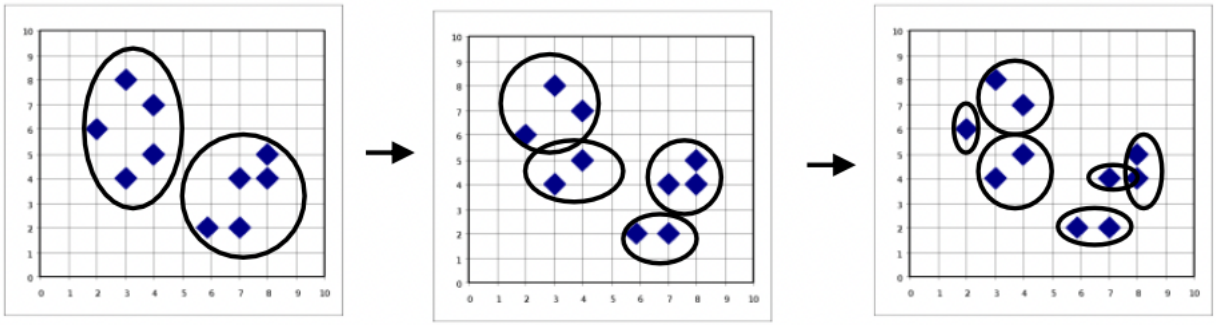


Dendrogram: Shows how Clusters Are Merged



DIANA (Divisive Analysis)

- Introduced in Kaufmann and Rousseau (1990).
- Implemented in statistical analysis package, e.g., Splus.
- Inverse order of AGNES.
- Eventually each node forms a cluster on its own.



Distance Between Clusters

- Single link: smallest distance between an element in one cluster and an element in the other:
  - dist(K_i, K_j) = min(t_{ip}, t_{jq})
- Complete link: largest distance between an element in one cluster and an element in the other:
  - dist(K_i, K_j) = max(t_{ip}, t_{jq})
- Average: avg distance between an element in one cluster and an element in the other:
  - dist(K_i, K_j) = avg(t_{ip}, t_{jq})
- Centroid: distance between the centroids of 2 clusters:
  - dist(K_i, K_j) = dist(C_i, C_j)
- Medoid: distance between the medoids of two clusters:
  - dist(K_i, K_j) = dist(M_i, M_j)
  - Medoid: a chosen, centrally located object in the cluster.

Centroid, Radius, and Diameter of a Cluster (for numerical data sets)

- Centroid: the "middle" of a cluster.

$$C_m = \frac{\sum_{i=1}^{N}(t_{ip})}{N}$$

- Radius: Square root of average distance from any point of the cluster to its centroid.

$$R_m = \sqrt{\frac{\sum_{i=1}^{N}(t_{ip} - c_m)^2}{N}}$$

- Diameter: Square root of average mean squared distance between all pairs of points in the cluster.

$$D_m = \sqrt{\frac{\sum_{i=1}^{N}\sum_{i=1}^{N}(t_{ip} - t_{iq})^2}{N(N-1)}}$$

Density-Based Clustering Methods

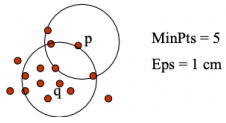
- Clustering based on density (local cluster criterion), such as density-connected points.
- Major features:
  - Discover clusters or arbitrary shape.
  - Handle noise.
  - One scan.
  - Need density parameters as termination condition.
- Several interesting studies:

- DBSCAN: Ester, et al. (KDD'96)
- OPTICS: Ankerst, et al (SIGMOD'99).
- DENCLUE: Hinneburg & D. Keim (KDD'98)
- CLIQUE: Agrawal, et al. (SIGMOD'98) (more grid-based)

Density-Based Clustering: Basic Concepts

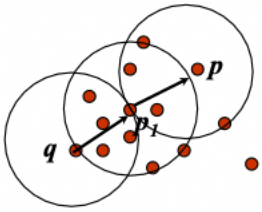
- 2 parameters:
  - Eps: Maximum radius of the neighborhood.
  - MinPts: Minimum number of points in an Eps-neighborhood of that point.
- $N_{Eps}(q)$: {p belongs to D | dist(p, q) ≤ Eps}
- Directed density-reachable: A point p is directly density-reachable from a point q w.r.t Eps, MinPts if:
  - p belongs to $N_{eps}(q)$
  - core point condition:

$$|N_{Eps}(q)| \geq MinPts$$

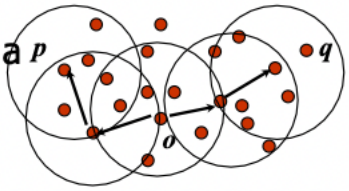


Density-Reachable and Density-Connected

- Density-reachable:
  - A point *p* is density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if there is a chain of points $p_1, \ldots, p_n, p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i

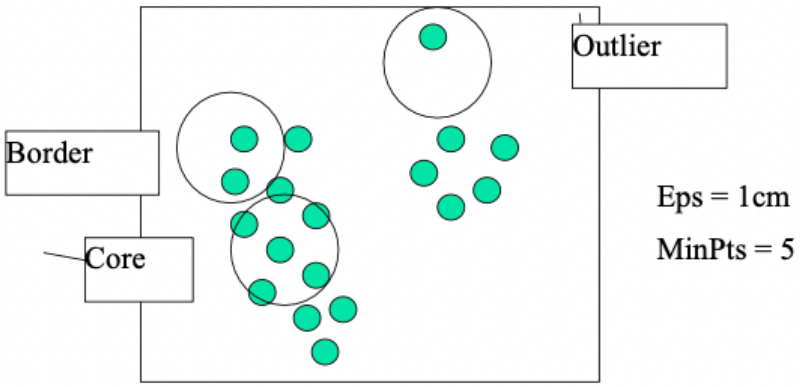


- Density-connected
  - A point *p* is density-connected to a point *q* w.r.t. *Eps*, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* w.r.t. *Eps* and *MinPts*



DBSCAN: Density-Based Spatial Clustering of Applications with Noise

- Relies on a *density-based* notion of cluster: A *cluster* is defined as a maximal set of density-connected points
- Discovers clusters of arbitrary shape in spatial databases with noise



DBSCAN: Sensitive to Parameters

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

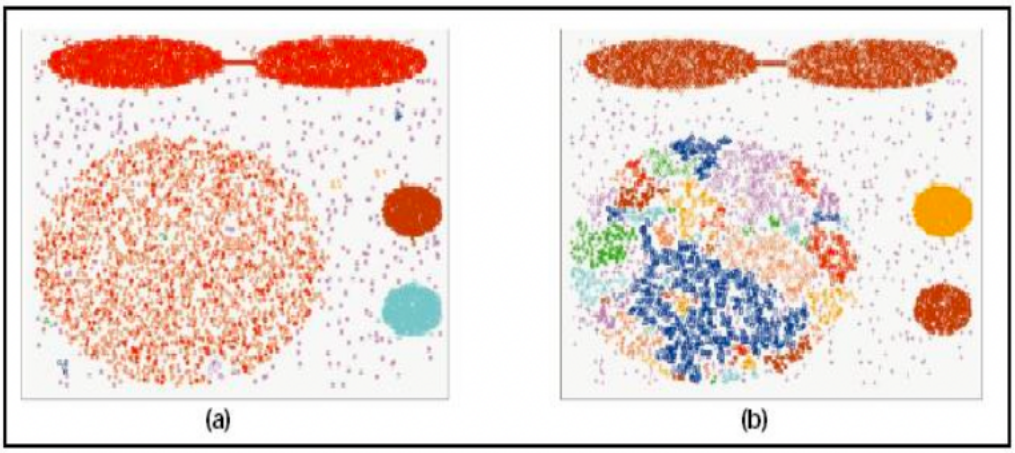
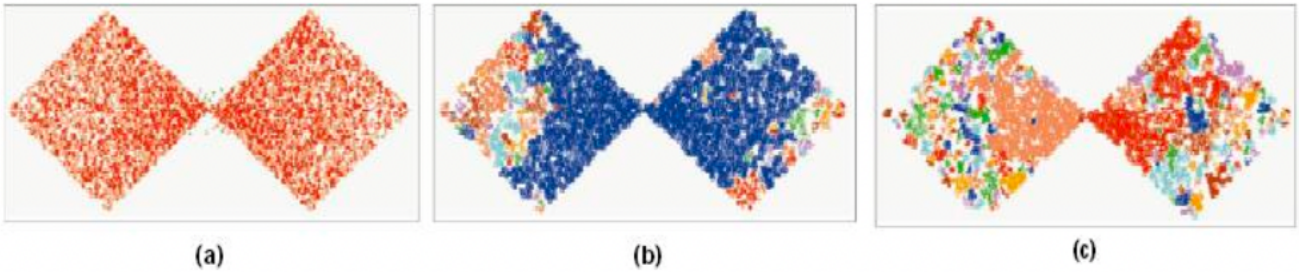


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



OPTICS: A Cluster-Ordering Method (1999)

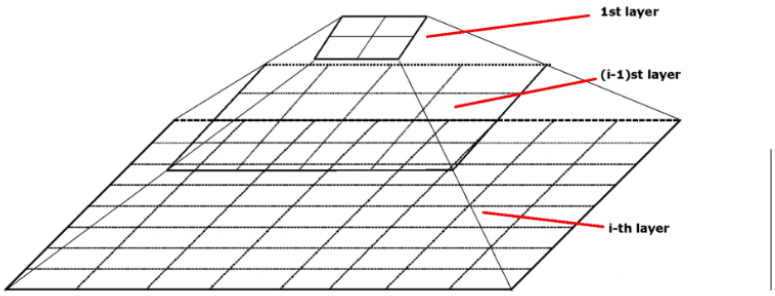
- OPTICS: Ordering Points To Identify the Clustering Structure
  - Ankerst, Breunig, Kriegel, and Sander (SIGMOD'99)
  - Produces a special order of the database wrt its density-based clustering structure
  - This cluster-ordering contains info equiv to the density-based clusterings corresponding to a broad range of parameter settings
  - Good for both automatic and interactive cluster analysis, including finding intrinsic clustering structure
  - Can be represented graphically or using visualization techniques

Grid-Based Clustering Method

- Using multi-resolution grid data structure
- Several interesting methods
  - STING (a STatistical INformation Grid approach) by Wang, Yang and Muntz (1997)
  - CLIQUE: Agrawal, et al. (SIGMOD'98)
    - Both grid-based and subspace clustering
  - WaveCluster by Sheikholeslami, Chatterjee, and Zhang (VLDB'98)
    - A multi-resolution clustering approach using wavelet method

STING: A Statistical Information Grid Approach

- Wang, Yang and Muntz (VLDB'97)
- The spatial area is divided into rectangular cells
- There are several levels of cells corresponding to different levels of resolution



The STING Clustering Method

- Each cell at a high level is partitioned into a number of smaller cells in the next lower level
- Statistical info of each cell is calculated and stored beforehand and is used to answer queries
- Parameters of higher level cells can be easily calculated from parameters of lower level cell
  - *count*, *mean*, *s*, *min*, *max*
  - type of distribution—*normal*, *uniform*, etc.
- Use a top-down approach to answer spatial data queries
- Start from a pre-selected layer—typically with a small number of cells
- For each cell in the current level compute the confidence interval

STING Algorithm and Its Analysis

- Remove the irrelevant cells from further consideration
- When finish examining the current layer, proceed to the next lower level
- Repeat this process until the bottom layer is reached
- Advantages:
  - Query-independent, easy to parallelize, incremental update
  - *O(K),* where *K* is the number of grid cells at the lowest level
- Disadvantages:
  - All the cluster boundaries are either horizontal or vertical, and no diagonal boundary is detected

CLIQUE (Clustering In QUEst)

- Agrawal, Gehrke, Gunopulos, Raghavan (SIGMOD'98)
- Automatically identifying subspaces of a high dimensional data space that allow better clustering than original space
- CLIQUE can be considered as both density-based and grid-based
  - It partitions each dimension into the same number of equal length interval
  - It partitions an m-dimensional data space into non-overlapping rectangular units
  - A unit is dense if the fraction of total data points contained in the unit exceeds the input model parameter
  - A cluster is a maximal set of connected dense units within a subspace

CLIQUE: The Major Steps

- Partition the data space and find the number of points that lie inside each cell of the partition.
- Identify the subspaces that contain clusters using the Apriori principle
- Identify clusters
  - Determine dense units in all subspaces of interests
  - Determine connected dense units in all subspaces of interests.
- Generate minimal description for the clusters
  - Determine maximal regions that cover a cluster of connected dense units for each cluster
  - Determination of minimal cover for each cluster

Determine The Number Of Clusters

- Empirical method
  - # of clusters: k ≈√n/2 for a dataset of n points, e.g., n = 200, k = 10
- Elbow method
  - Use the turning point in the curve of sum of within cluster variance w.r.t the # of clusters
- Cross validation method
  - Divide a given data set into *m* parts
  - Use *m* – 1 parts to obtain a clustering model
  - Use the remaining part to test the quality of the clustering
    - E.g., For each point in the test set, find the closest centroid, and use the sum of squared distance between all points in the test set and the closest centroids to measure how well the model fits the test set
  - For any k > 0, repeat it *m* times, compare the overall quality measure w.r.t. different *k's*, and find # of clusters that fits the data the best

Measuring Clustering Quality

- 3 kinds of measures: External, internal and relative
- External: supervised, employ criteria not inherent to the dataset
  - Compare a clustering against prior or expert-specified knowledge (i.e., the ground truth) using certain clustering quality measure
- Internal: unsupervised, criteria derived from data itself
  - Evaluate the goodness of a clustering by considering how well the clusters are separated, and how compact the clusters are, e.g., Silhouette coefficient
- Relative: directly compare different clusterings, usually those obtained via different parameter settings for the same algorithm

Some Commonly Used External Measures

- Matching-based measures
  - Purity, maximum matching, F-measure
- Entropy-Based Measures
  - Conditional entropy, normalized mutual information (NMI), variation of information
- Pair-wise measures
  - Four possibilities: True positive (TP), FN, FP, TN
  - Jaccard coefficient, Rand statistic, Fowlkes-Mallow measure
- Correlation measures
  - Discretized Huber static, normalized discretized Huber static

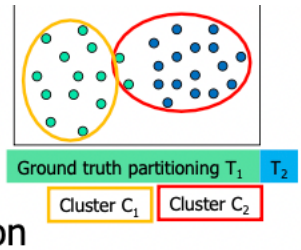


Table of Contents

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY

SOICT

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

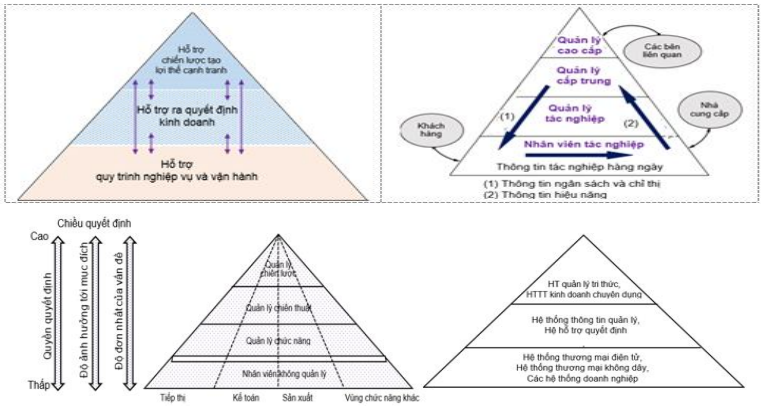
1

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Hệ thống hỗ trợ thông tin và quyết định

2

CÁC HTTT DOANH NGHIỆP



HTTT trong các tổ chức kinh doanh theo ba mức:

Mức trên: Hệ thống quản lý tri thức và hệ thống thông tin kinh doanh chuyên ngành . QL chiến lược

Mức giữa: HT thông tin quản lý và Hệ hỗ trợ quyết định. QL chiến thuật

Mức dưới: Thương mại điện tử, thương mại không dây (M-commerce: Mobile-commerce) và các hệ thống doanh nghiệp. QL chức năng (tác nghiệp)

Nội dung

1. Ra quyết định và giải quyết vấn đề
2. Khái quát về HTTT quản lý
3. Các HTTT quản lý chức năng
4. Khái quát về hệ hỗ trợ quyết định
5. Các thành phần của hệ hỗ trợ quyết định
6. Hệ thống hỗ trợ nhóm
7. Hệ thống hỗ trợ điều hành
8. Dẫn luận: Công ty Generals Mills, Mỹ
9. C/ty dược phẩm AstraZeneca giảm thời gian ra thị trường
10. Bốn nguyên lý và mục tiêu học tập

1. Ra quyết định và giải quyết vấn đề

- Giới thiệu
  - Mọi tổ chức cần ra quyết định hiệu quả
  - Các khóa học ra quyết định
    - Nhân viên và đơn vị kinh doanh
    - Hoàn thành mục tiêu và mục đích
  - HTTT hỗ trợ **giải quyết vấn đề**:
    - Giúp ra quyết định tốt hơn và tiết kiệm hơn
  - Ví dụ HTTT Trung tâm Y tế ĐH Hackensack
    - Phân tích tương tác thuốc tiềm năng "thuốc – phản ứng".

      Chủ đề nghiên cứu "phân tích quan hệ ngữ nghĩa ("thuốc – phản ứng") trong văn bản y sinh.

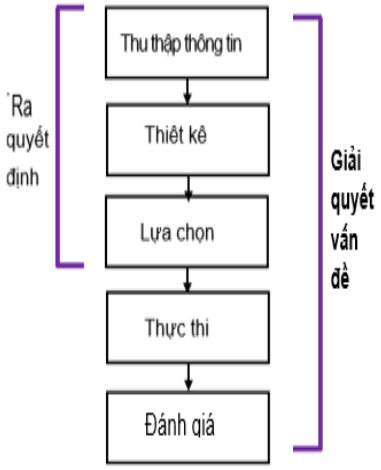
      Cụ thể: Thuốc trầm cảm cho bệnh nhân AIDS
    - Đầu tư hàng triệu đô la cho HTTT

Các kiểu vấn đề

- Vấn đề cấu trúc được và không cấu trúc
  - Vấn đề cấu trúc được (structured problem)
    - Quen thuộc, đơn giản, và các yêu cầu thông tin rõ ràng.
    - "Doanh số tuần này có cao hơn tuần trước"?
    - Chia nhỏ được thành chuỗi các bước đã được xác định tốt
    - **Tương ứng với "thuật toán hóa": Lời giải lập trình được**
  - Vấn đề không cấu trúc được (unstructured problem)
    - Mơ hồ do thiếu thông tin
    - "Đặc trưng khách hàng mua nhiều hàng tuần này" ?
    - Không thể chia nhỏ được thành chuỗi bước được xác định tốt
    - Cần sử dụng trực giác, lý luận, và ghi nhớ
    - **Lời giải không lập trình được**

Ra quyết định: thành phần của giải vấn đề

- Giải vấn đề
  - Hoạt động quan trọng của mọi tổ chức kinh doanh
  - Người "giải vấn đề thực sự"
- Mô hình giải vấn đề
  - Mô hình ra quyết định Herbert Simon
    - Nổi tiếng
    - Ba giai đoạn: thu thập thông tin (intelligence), thiết kế (design), chọn lựa (choice)
  - Mô hình giải vấn đề
    - George Huber mở rộng mô hình trên
    - Thi hành (implementation), Giám sát (monitoring) kết quả giải vấn đề



Các giai đoạn giải vấn đề: ra quyết định

- **Ví dụ: muốn bán vải thiều Mai Siu tại Hà Nội**
- Thu thập thông tin
  - Nhận dạng và xác định vấn đề hoặc cơ hội tiềm năng
  - Điều tra tài nguyên và ràng buộc môi trường
    - Vấn đề: vải thiều dễ hỏng;
    - Cơ hội: giá bán buôn vải ở Hà Nội cao
- Thiết kế
  - Các giải pháp thay thế nhau (nên **vài ba** giải pháp) **"Mọi mô hình đều sai và có một vài mô hình dùng được"**
    - Thuê ô tô riêng / đi ô tô khách / đi bằng xe máy
    - Thời gian: lộ trình ? Chi phí ?
- Chọn lựa
  - Chọn giải pháp khả thi nhất từ các giải pháp thay thế nhau
    - Thuê ô tô riêng/đi ô tô khách/đi bằng xe máy

Hai giai đoạn thi hành quyết định

- **Ví dụ: muốn bán vải thiều Mai Siu tại Hà Nội**
- Thực thi
  - Thực thi giải pháp đã lựa chọn (vận chuyển vải bằng xe máy)
  - Thông báo khách hàng, vận chuyển vải, giao quả vải, nhận tiền
- Giám sát
  - Có thông tin kết quả thực thi: thông tin phản hồi
  - Người ra quyết định tốt đánh giá giải pháp được chọn
  - Thông tin phản hồi→ Điều chỉnh giải pháp được chọn
    - ví dụ, điều chỉnh lịch trình vận chuyển, cách đặt vải thiều vào sọt…
  - Thay đổi giải pháp: chọn giải pháp thay thế phù hợp

Quyết định lập trình được

- Chọn lựa: nhiều nhân tố tác động đến chọn giải pháp
- Một nhân tố: quyết định lập trình được hay không
- Quyết định lập trình được
  - Có được với một quy tắc/thủ tục/phương pháp định lượng
    - Ví dụ: "hàng trong kho dưới 100 đơn vị thì cần được đặt hàng" là quyết định lập trình được vì tuân theo một quy tắc
  - Dễ dàng tin học hóa khi dùng HTTT truyền thống
    - Dễ lập trình khi số hàng trong kho <= 100 đơn vị thì đặt hàng
  - Mối quan hệ giữa các thành phần trong HT là cố định
  - Một dạng QĐ lập trình được: cung cấp báo cáo vấn đề thường xuyên mà mối quan hệ được xác định
- Giải pháp
  - Hầu hết quá trình tự động hóa ở hệ thống hoạch định tài nguyên doanh nghiệp / Hệ thống xử lý giao dịch
  - HTTT quản lý: các báo cáo mức cao hơn

Quyết định không lập trình được

- Tình huống
  - Các tình huống bất thường hoặc đặc thù
    - Xác định chương trình đào tạo cho một nhân viên mới
    - Quyết định phát triển một sản phẩm/dịch vụ mới
    - cận nhắc lợi ích và hạn chế lắp đặt một hệ thống kiểm soát ô nhiễm nâng cấp
    - Hệ thống thông tin soạn thảo, thi hành luật
  - Quyết định rất khó định lượng
  - Quyết định có tính độc đáo
  - Không áp dụng được các quy tắc, thủ tục chuẩn
- Giải pháp
  - Hệ hỗ trợ quyết định

Tiếp cận tối ưu hóa

- Mô hình tối ưu hóa
  - Hệ HT quyết định tin học hóa là tối ưu hoặc đáp ứng
  - Một quá trình tìm giải pháp tốt nhất (một trong các tốt nhất) giúp tổ chức đạt được mục tiêu của mình.
  - mô hình tối ưu hóa tìm thấy giải pháp tốt nhất
    - điều kiện và giả định nhất định cho trước
    - sử dụng ràng buộc vấn đề
    - một mô hình tối ưu hóa tìm thấy lượng sản phẩm thích hợp mà tổ chức phải sản xuất để đáp ứng mục tiêu lợi nhuận

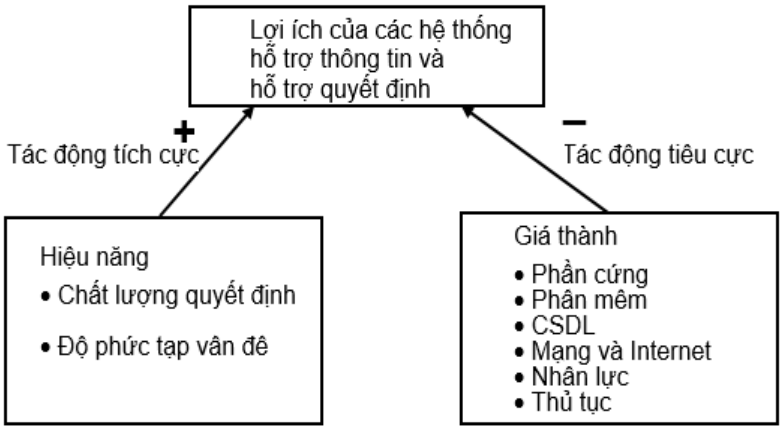
Tiếp cận đáp ứng

- Khái niệm
  - Mô hình đáp ứng tìm được giải pháp tốt song không phải là tốt nhất
    - Có phương pháp tìm được giải pháp tốt
    - Rất khó đánh giá tốt nhất ?
  - Không xem xét được mọi khả năng mà xem một vài khả năng tốt
  - Ví dụ
    - Lựa chọn vị trí đặt cửa hàng
    - Tốt nhất: xem mọi tình huống nhưng không thể
    - Đáp ứng: khoang vùng được tốt/tốt nhất rồi mới tìm kiếm
  - "Đáp ứng" là phương pháp mô hình hóa thay thế tốt
    - Quá đắt để phân tích mọi lựa chọn để lựa chọn tốt nhất,

Heristic (tự khám phá) và phản hồi

- Heuristics
  - Quy tắc ngón tay cái: rules of thumb
  - Chấp nhận hướng dẫn/thủ tục tìm giải pháp tốt
  - Thường dùng khi ra quyết định
    - Đặt hàng trước 4 tháng khi số hàng <=20
  - Phần mềm chống thư rác
    - Heuristic dựa theo luật/phân lớp
    - Tìm ra thư có khả năng nhất
    - Không tìm tất cả các thư
- Cảm nhận và phản hồi
  - Sense and Respond
  - xác định vấn đề/cơ hội (cảm nhận) và phát triển hệ thống để giải vấn đề /tận dụng cơ hội (phản hồi)
  - SAR thường đòi hỏi phải tổ chức linh hoạt để thay thế dòng truyền thống

Lợi ích HTTT QL và HTHTQĐ



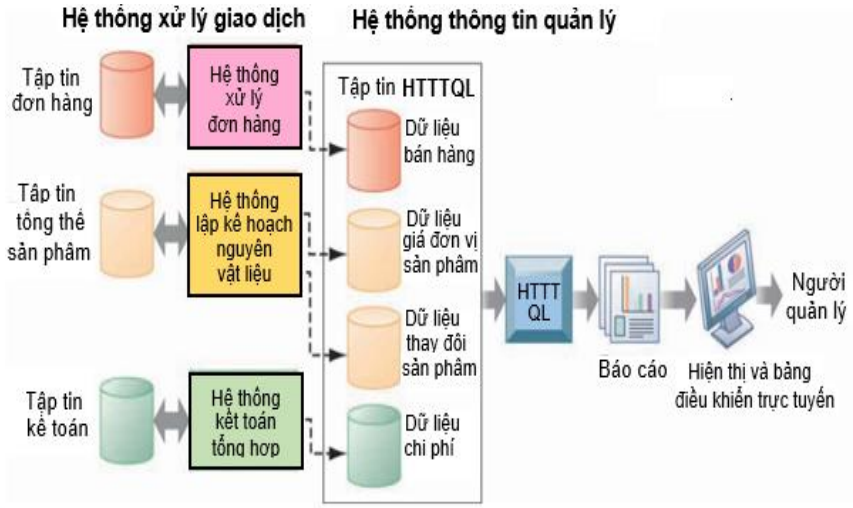
15

2. Tổng quan về HTTT quản lý

- Khái niệm
  - Khái niệm
    - con người, thủ tục, cơ sở dữ liệu, và các thiết bị
    - cung cấp thông tin cho các nhà quản lý ra quyết định
    - giúp đạt được mục tiêu của tổ chức
  - Lợi thế cạnh tranh: thông tin chính xác, đúng người, đúng lúc
- Các khía cạnh
  - Ngắn hạn: các báo cáo phản hồi hoạt động hàng ngày
  - Mọi cấp trong toàn tổ chức The Truth About Your Future

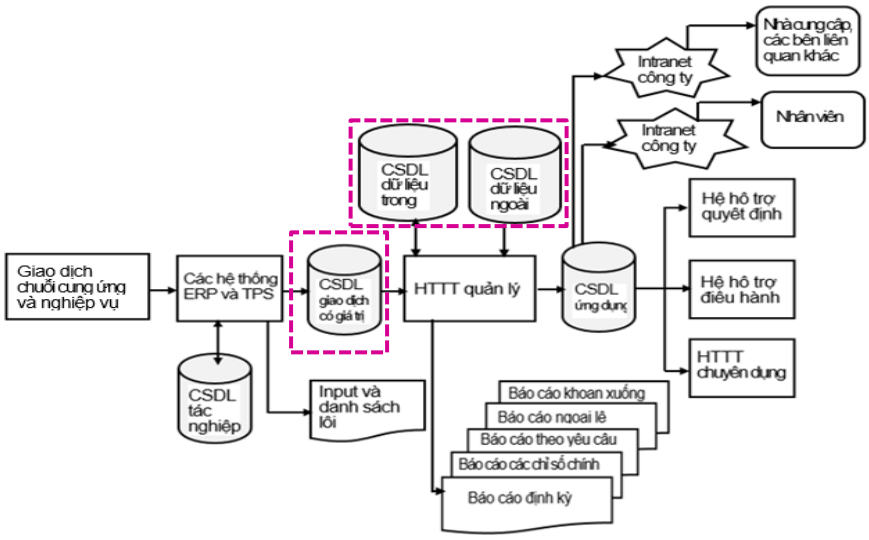
16

HTTT quản lý



17

HTTT quản lý: Đầu vào



HTTTQL chỉ là một trong nhiều nguồn TT quản lý (Hệ HTQĐ, hệ HTĐH và hệ CG cũng hỗ trợ việc ra quyết định). *Từ giao dịch chuỗi cung ứng và kinh doanh*

18

HTTT QL: Đầu vào

- bên trong và bên ngoài, bao gồm chuỗi cung ứng (supply chain)
- được xử lý thành báo cáo dễ sử dụng cho các nhà quản lý

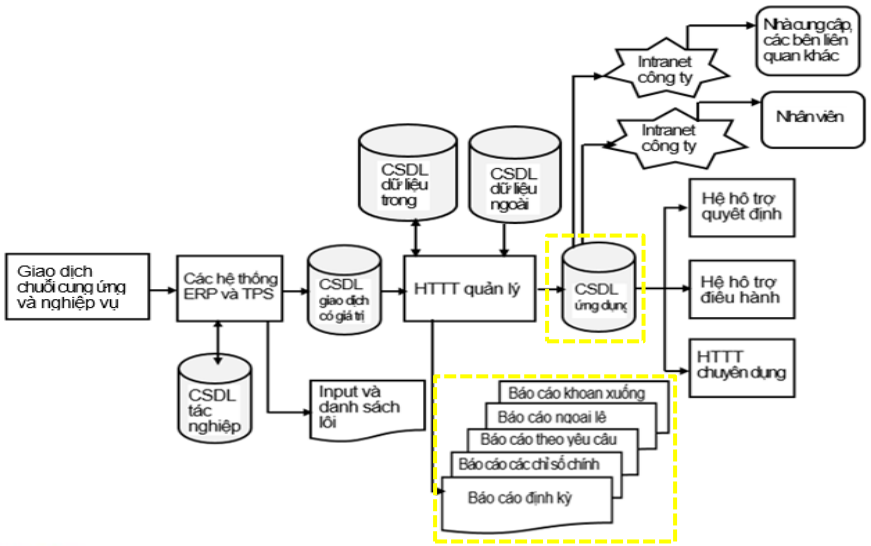
• Bên trong

- Nguồn quan trọng nhất các TPS, HT ERP và CSDL liên quan
- Kho DL, kho DL chuyên (Data mart): thông tin kinh doanh giá trị, thông minh kinh doanh
- DL từ các khu vực chức năng khác

• Bên ngoài

- DL về khách hàng, nhà cung cấp, đối thủ cạnh tranh, cổ đông, DL khác (Internet)
- Nhóm doanh nghiệp kết nối với nhau trao đổi DL

HTTT quản lý: Đầu ra



HTTT QL: Đầu ra

- tập kiểu các báo cáo: cung cấp từng người kịp thời
- định kỳ (scheduled), chỉ số chính (key indicator), theo yêu cầu (demand), ngoại lệ (exception), khoan xuống (drill down)

• Các loại báo cáo

- Báo cáo định kỳ: được tạo ra theo định kỳ, hoặc theo một lịch trình, chẳng hạn như hàng ngày, hàng tuần, hoặc hàng tháng.
- Báo cáo chỉ số chính: tóm tắt các hoạt động quan trọng của ngày trước đó, sẵn sàng đầu ngày làm việc của nhà quản lý- điều hành.
- Báo cáo theo yêu cầu: báo cáo được tạo ra để cung cấp thông tin nào đó theo yêu cầu của một người (điều hành, nhà cung cấp, khách hàng).
- Báo cáo ngoại lệ: được tự động tạo ra khi một tình huống bất thường hoặc theo đòi hỏi hành động quản lý (người quản lý đặt tham số để tạo một báo cáo về mọi sản phẩm tồn kho ít hơn lượng năm ngày bán hàng hiện hành)
- Báo cáo khoan xuống (chi tiết hóa) cung cấp dữ liệu chi tiết hơn về một tình huống. *Khoan xuống là một kỹ thuật trong kho dữ liệu*

Đặc điểm của HTTTQL

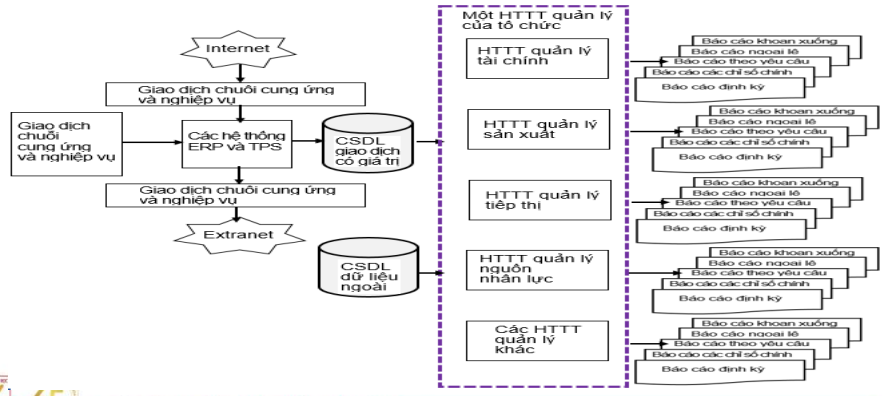
- Báo cáo định kỳ , chỉ số chính, theo yêu cầu, ngoại lệ, và khoan xuống giúp các nhà quản lý và điều hành ra quyết định tốt hơn, kịp thời hơn.

• Các đặc điểm HTTTQL

- Cung cấp các báo cáo với các định dạng cố định và chuẩn
- Tạo ra các báo cáo bản cứng và bản mềm
- Dùng dữ liệu nội bộ được lưu trong hệ thống máy tính
- Cho phép người dùng xây dựng báo cáo của riêng họ
- Phụ thuộc yêu cầu người dùng tới các báo cáo được nhân viên hệ thống phát triển

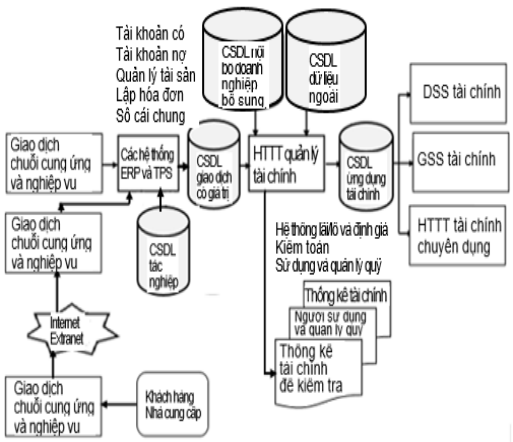
3. Khía cạnh chức năng của các HTTTQL

- Đặt vấn đề
  - Tổ chức được cấu trúc theo tuyến hoặc vùng chức năng: phân cấp theo vai trò hoặc vị trí
  - MIS theo vùng chức năng truyền thống: tài chính, sản xuất, tiếp thị, nguồn nhân lực, khác



HTTT quản lý tài chính

- Sơ bộ
  - Phù hợp HTTTQL chung
  - Yêu cầu khách hàng
  - Các báo cáo
  - Hệ thống lợi nhuận/ chi phí, giá
  - Kiểm toán
  - Dùng và quản lý quỹ
  - DSS TC, GSS TC, HTTTTC chuyên sâu



- HTTT quản lý tài chính
  - Cung cấp TT tài chính không chỉ người điều hành mà cho tập rộng rãi người cần ra quyết định tốt hơn hàng ngày (đầu tư cổ phiếu…)
  - Cho phép tự động gắn máy kinh doanh

HTTT quản lý tài chính: chức năng

- Chức năng
  - Tích hợp thông tin tài chính và hoạt động từ nhiều nguồn (cả Internet) vào một hệ thống duy nhất
  - Cung cấp sự dễ dàng truy cập dữ liệu cho người sử dụng cả về tài chính và phi tài chính, thường dùng mạng nội bộ công ty để truy cập các trang web của công ty dữ liệu và thông tin tài chính
  - Tạo sự sẵn có tức thời dữ liệu tài chính để rút ngắn thời gian chu kỳ phân tích
  - Cho phép phân tích dữ liệu tài chính theo nhiều chiều: thời gian, địa lý, sản phẩm, nhà máy, và khách hàng
  - Phân tích hoạt động tài chính lịch sử và hiện tại
  - Theo dõi và kiểm soát việc sử dụng quỹ theo thời gian
- Cấu trúc điển hình
  - đầu vào, các hệ thống con đầu ra: xem hình vẽ trang trước
  - Các hệ thống con: lợi nhuận/chi phí và giá, kiểm toán, sử dụng-quản lý quỹ

HTTT quản lý tài chính: các hệ thống con

- HT con lãi/chi phí và định giá
  - Các trung tâm lợi nhuận ở nhiều bộ phận: hướng tới sinh lợi nhuận. Ví dụ: phòng đầu tư của một Cty bảo hiểm và thẻ tín dụng lớn
  - Các trung tâm doanh số: hướng tới tăng doanh số như các bộ phận bán hàng, tiếp thị
  - Các trung tâm giá: hướng tới giảm giá như bộ phận sản xuất, R&D
  - Htcon lợi nhuận, chi phí, doanh số, **định giá**
- HT con kiểm toán
  - quá trình có tính hệ thống, độc lập và được làm tài liệu để có bằng chứng kiểm toán (hồ sơ, báo cáo về sự kiện/thông tin khác có liên quan và kiểm chứng được) và đánh giá nó một cách khách quan nhằm xác định mức độ các tiêu chuẩn kiểm toán (tập các chính sách, thủ tục hoặc yêu cầu) được đáp ứng. Phân tích điều kiện tài chính
  - Xác định thông báo và báo cáo tài chính do MIS tạo ra có chính xác
  - Kiểm toán nội bộ (trong): Bộ phận tài chính dùng nhân viên
  - Kiểm toán độc lập (ngoài): Big Four gồm Deloitte, PwC, Ernst&Young, KPMG (USD23.03 billion in the 2012 fiscal year)

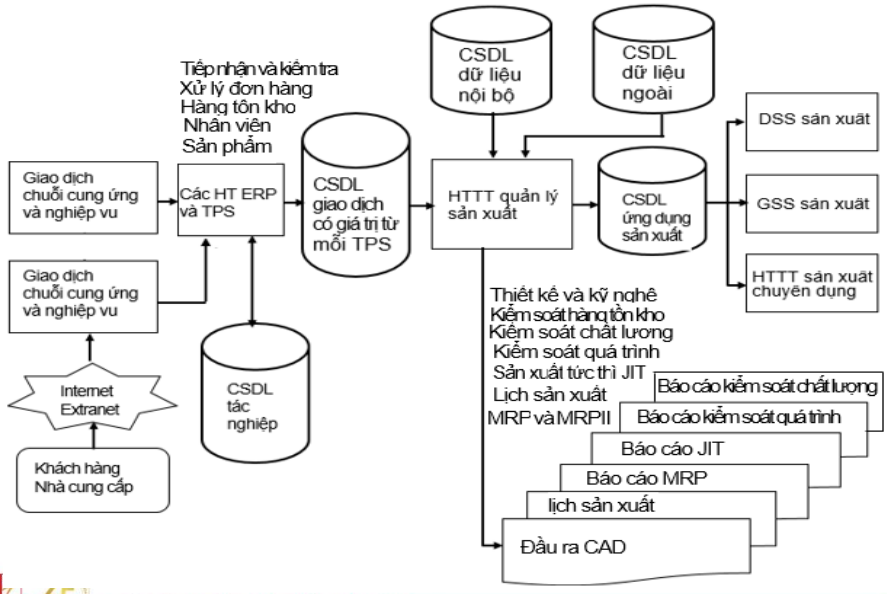
HTTT quản lý tài chính: HT con quỹ

- Dùng quỹ nội bộ
  - mua hàng tồn kho bổ sung,
  - nâng cấp nhà máy và thiết bị,
  - thuê nhân viên mới,
  - mua lại các công ty khác, mua hệ thống máy tính mới,
  - tăng tiếp thị và quảng cáo, mua nguyên liệu hoặc đất đai,
  - đầu tư vào sản phẩm mới, và tăng nghiên cứu và phát triển
- Dùng quỹ ra ngoài
  - Đầu tư kinh phí dư thừa ra bên ngoài
  - Tài khoản ngân hàng, cổ phiếu, trái phiếu, tín phiếu, danh tiếng, cho tương lai, quyền mua bán, ngoại tệ
  - Tạo khoản vay trên Internet:

HTTT quản lý sản xuất

- Giới thiệu
  - HTTTQL sản xuất: "cách mạng hóa" sản xuất bằng HTTTQL
  - Cải thiện đáng kể nhiều hoạt động sản xuất
  - Nhấn mạnh chất lượng và năng suất hơn → một quá trình sản xuất hiệu quả ngày càng trở nên quan trọng hơn
  - Tin học hóa: từ quầy hàng tới lãnh đạo cấp cao.
  - Càng nhiều công ty gia công phần mềm quá trình sản xuất
- Nội dung sơ bộ
  - HTTTQL sản xuất và kết quả ra để theo dõi & quản lý dòng chảy vật liệu, sản phẩm xuyên qua công ty.
  - HTTTQL SX mọi khâu chuyển đổi nguyên vật liệu tới thành phẩm
  - Công nghệ mới (chip): tạo dễ dàng dòng chuyển này
  - Thành công của tổ chức phụ thuộc vào chức năng sản xuất
  - Nhiều HTTTQL con.

HTTTQL sản xuất: sơ đồ chung



HTTTQL sản xuất: Các HT con

- Thiết kế và kỹ nghệ
  - HTTT thiết kế nhờ MT (computer-aided design: CAD) sản phẩm mới hoặc hiện có
  - Dùng CAD phát triển và thiết kế sản phẩm/cấu trúc phức tạp
  - Ví dụ Boeing
- Điều khiển lịch sản xuất và kiểm soát hàng tồn kho
  - Lập kế hoạch SX và kiểm soát hàng tồn kho: rất quan trọng cho mọi công ty sản xuất
  - Mục tiêu của điều khiển lập lịch: cung cấp KH chi tiết cả lịch biểu ngắn hạn và dài hạn của các CSSX
  - Nhiều kỹ thuật: giảm thiểu chi phí hàng tồn kho. Khi nào và bao nhiêu hàng tồn kho cần đặt hàng ? Câu trả lời reorder point: ROP.

HTTTQL sản xuất: HT con điều khiển QT

- Khái niệm
  - Theo dõi và sắp xếp dòng quá trình SX (KPQT)
  - Trực tiếp điều khiển thiết bị SX: SX có máy tính hỗ trợ (computer-assisted manufacturing: CAM). Hệ thống CAM điều khiển máy khoan, dây chuyền lắp ráp, và nhiều ứng dụng khác
  - Sản xuất tích hợp máy tính (Computer-integrated manufacturing: CIM): dùng máy tính liên kết các thành phần SX.

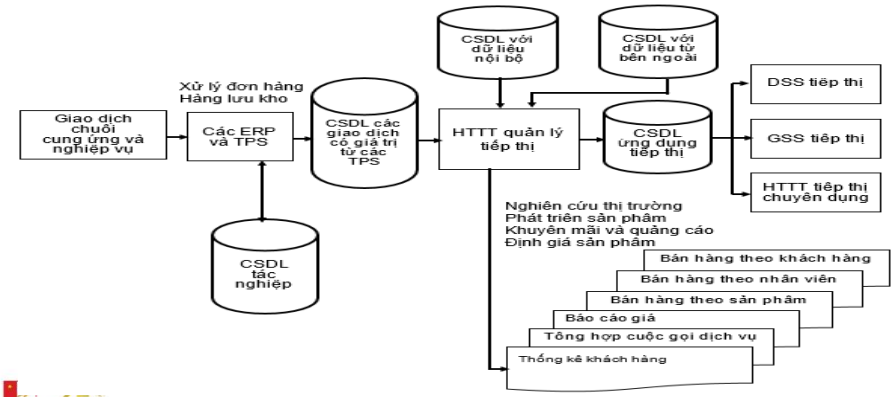
    Mục tiêu của CIM: kết hợp chặt chẽ mọi khía cạnh sản xuất, bao gồm xử lý đơn hàng, thiết kế sản phẩm, sản xuất, kiểm tra, kiểm soát chất lượng, và vận chuyển.
  - Hệ thống SX linh hoạt (flexible manufacturing system: FMS): cho phép cơ sở sản xuất để nhanh chóng và hiệu quả từ đã có một sản phẩm khác. Thay đổi quá trình

HTTTQL sản xuất: Các HT con

- Điều khiển quy trình
  - Phương pháp lượng đặt hàng kinh tế (economic order quantity: EOQ). Khi hàng nhỏ hơn ngưỡng (reorder point: ROP)
  - Kỹ thuật lập kế hoạch yêu cầu vật liệu (MRP): tập kỹ thuật kiểm soát hàng tồn kho trợ giúp phối hợp hàng ngàn mặt hàng tồn kho khi nhu cầu của một mặt hàng phụ thuộc vào nhu cầu khác
  - Kỹ thuật hàng tồn kho và sản xuất Just-in- time (JIT): Duy trì hàng tốn kho ở mức thấp nhất mà không mất tính sẵn có của SP hoàn chỉnh.
  - JIT: hết hàng tồn kho khi có đợt mua hàng nhiều
- Kiểm soát chất lượng và kiểm thử
  - Kiểm soát chất lượng: sản phẩm đáp ứng yêu cầu người dùng ?

HTTT quản lý tiếp thị

- Khái niệm
  - hỗ trợ quản lý phát triển sản phẩm, phân phối, quyết định giá cả, hiệu quả quảng cáo, và dự báo bán hàng.
  - Xu thế sử dụng trên Internet
  - Các HTTT con: nghiên cứu thị trường, phát triển sản phẩm, khuyến mãi và quảng cáo, và giá cả sản phẩm



HTTT quản lý tiếp thị

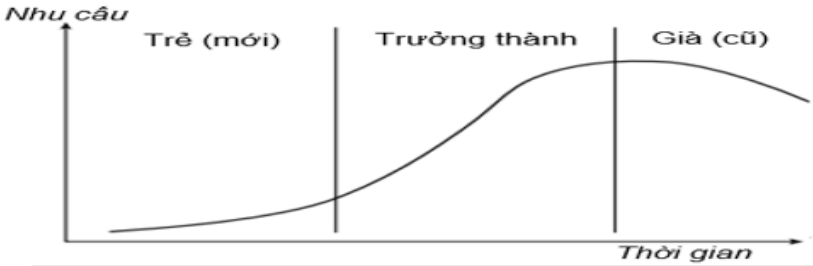
- HT con nghiên cứu thị trường
  - Tiến hành một nghiên cứu chính quy về thị trường và sở thích của khách hàng
  - *Chương trình máy tính* giúp tiến hành và phân tích kết quả điều tra, bảng hỏi, nghiên cứu thí điểm, và các cuộc phỏng vấn
- Phát triển sản phẩm
  - chuyển đổi nguyên liệu vào hàng hóa và dịch vụ hoàn thiện: tập trung chủ yếu vào các thuộc tính vật chất của sản phẩm
  - Các yếu tố: năng suất máy móc, kỹ năng lao động, các yếu tố kỹ thuật, và các tài liệu
  - *chương trình máy tính* phân tích các yếu tố khác nhau và lựa chọn sự pha trộn thích hợp của lao động, vật tư, máy móc thiết bị, và thiết kế kỹ thuật

N/C thị trường: sản phẩm/thị trường

	Thị trường hiện thời	Thị trường mới
Sản phẩm mới	Nghiên cứu thị trường có thể chỉ ra khả năng tiếp nhận sản phẩm mới	Nghiên cứu thị trường có thể chỉ ra nhu cầu chưa được đáp ứng và cung cấp hiểu biết về các thị trường mới
Sản phẩm hiện có	Nghiên cứu thị trường có thể đo lường sự hài lòng khách hàng để tìm được cách thức duy trì lợi thế cạnh tranh	Nghiên cứu thị trường có thể tìm ra các vũng lãnh thổ mới cho các sản phẩm hoặc dịch vụ

35

N/C thị trường: nhu cầu/thời gian



Nghiên cứu thị trường khám phá các nhu cầu chưa được đáp ứng cho sản phẩm mới và giúp ước tính nhu cầu có thể. Nó có thể được sử dụng để định giá và định hình các đặc điểm kỹ thuật của sản phẩm	Nghiên cứu thị trường chỉ ra cách xây dựng thương hiệu và cạnh tranh. Nghiên cứu sự hài lòng của khách hàng chỉ ra các điểm mạnh cần được xây dựng, các điểm yếu cần được sửa chữa	Nghiên cứu thị trường chỉ ra khả năng trẻ hoá sản phẩm, có thể bằng cách kết hợp các tính năng mới hoặc tìm kiếm các thị trường mới

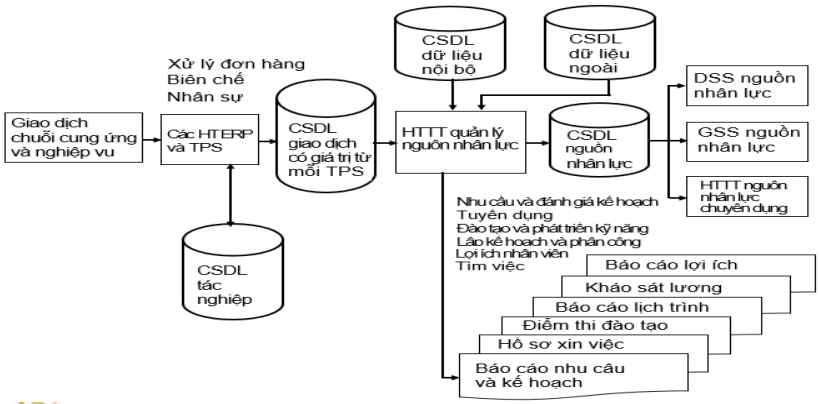
36

HTTT quản lý tiếp thị

- Xúc tiến và quảng cáo
  - chức năng quan trọng nhất của mọi nỗ lực tiếp thị
  - sử dụng Internet để quảng cáo và bán sản phẩm và dịch vụ
  - quảng cáo truyền hình và Internet
- Định giá SP
  - một chức năng tiếp thị quan trọng và phức tạp: giá bán lẻ, giá bán buôn, giảm giá
  - phát triển chính sách giá cả để tối đa tổng doanh thu bán hàng: *chương trình máy tính* phân tích mối quan hệ giữa giá và tổng doanh thu
- Phân tích bán hàng
  - rất quan trọng để xác định sự đóng góp của sản phẩm, nhân viên bán hàng, và khách hàng đóng góp vào lợi nhuận
  - Một số báo cáo được tạo ra giúp đưa ra quyết định bán hàng tốt
    - Báo cáo bán hàng theo sản phẩm chính
    - Báo cáo bán hàng theo nhân viên bán hàng
    - Báo cáo bán hàng theo khách hàng

HTTT quản lý nhân lực

- Khái niệm
  - quan tâm tới các hoạt động liên quan đến người lao động và người lao động tiềm năng
  - Thời gian: quá khứ, hiện tại và tương lai
  - Đóng vai trò quan trọng đảm bảo thành công của tổ chức



HTTT quản lý nhân lực

- Một số nội dung
  - Lập kế hoạch nguồn nhân lực: khía cạnh đầu tiên xác định nhân viên và nhu cầu nhân lực
  - Lựa chọn và tuyển dụng: chương trình máy tính sắp xếp các nỗ lực tuyển dụng, kiểm tra kỹ năng nhân viên tiềm năng, sàng lọc người xin việc qua Internet
  - Đào tạo và phát triển kỹ năng: yêu cầu đào tạo rất cụ thể cho nhân viên mới như văn hóa tổ chức, định hướng, tiêu chuẩn, và mong đợi của tổ chức.
  - Lập kế hoạch và sắp xếp công việc
  - Tiền lương và quản lý tiền lương
  - Tìm việc. Nhân viên rời khỏi công ty vì nhiều lý do. Công ty cung cấp dịch vụ tìm việc giúp nhân viên thực hiện quá trình chuyển đổi.

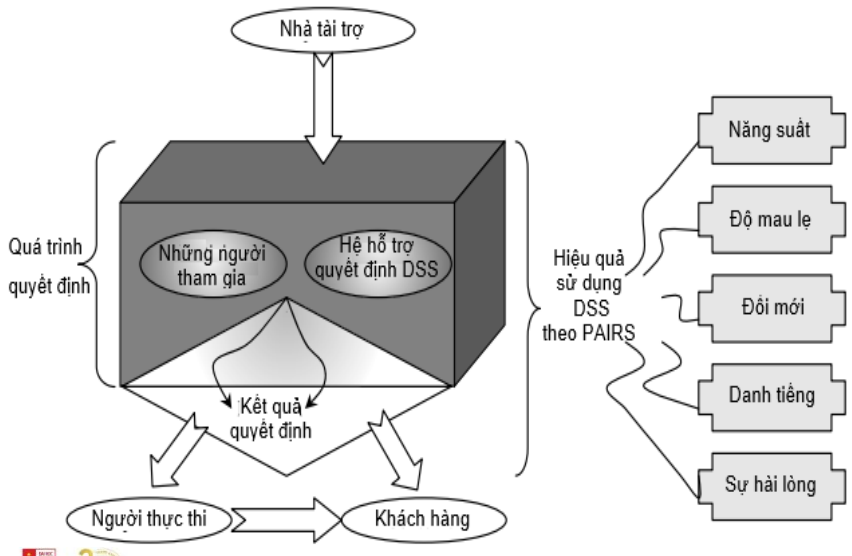
HTTT quản lý khác

- HTTT QL kế toán
  - Liên quan tới MIS tài chính
  - thực hiện một số hoạt động quan trọng, cung cấp thông tin tổng hợp về các khoản phải trả, các khoản phải thu, biên chế, và nhiều ứng dụng khác
- Hệ thống thông tin địa lý (GIS)
  - Geographic Information Systems
  - Trực quan hóa dữ liệu dưới dạng đồ họa
  - hệ thống máy tính có khả năng lắp ráp, lưu trữ, thao tác, và hiển thị thông tin địa lý tham chiếu:dữ liệu xác định theo vị trí của nó

4. Khái quát về HHTQĐ

- Hệ HTQĐ nhiều đặc trưng ⇒ trở thành công cụ hỗ trợ QL hiệu quả
- Đặc trưng của HHTQĐ
  - Cung cấp truy cập nhanh đến thông tin.
  - Xử lý một lượng lớn dữ liệu từ các nguồn khác nhau
  - Cung cấp và trình bày linh hoạt các báo cáo
  - Cung cấp cả hai định hướng văn bản và đồ họa
  - Hỗ trợ phân tích khoan xuống
  - Thực hiện phân tích và so sánh phức tạp, tinh vi sử dụng các gói phần mềm tiên tiến
  - Hỗ trợ tối ưu hóa, pháp thoả mãn, và cách tiếp cận heuristic
  - Thực hiện phân tích mô phỏng: khả năng của HHTQĐ để sao chép các tính năng của một hệ thống thực, nơi có liên quan tới tính khả năng hoặc tính không chắc chắn

Vai trò của HHTQĐ



HHTQĐ: Một số ví dụ

Tổ chức hoặc ứng dụng	*Mô tả*
ING Direct	Công ty dịch vụ tài chính dùng DSS để tổng hợp hiệu năng tài chính của một ngân hàng. Ngân hàng cần có cơ chế đo lường và theo dõi để xác định nó thành công ra sao và cách cải tiến để lên kế hoạch thời gian thực
Cinergy Corporation	Tiện ích số hóa phát triển một HHTQĐ để rút gọn thời gian và nỗ lực đòi hỏi ra quyết định
U.S. Army	Lực lượng quân đội Mỹ phát triển một HHTQĐ để trợ giúp tân binh, huấn luyện, và đào tạo nhập ngũ quân đội. HHTQĐ này sử dụng mô phỏng kết hợp các đặc trưng What-If.
National Audubon Society	Tổ chức phát triển một HHTQĐ được gọi là Energy Plan (EPLAN) để phân tích ảnh hưởng của chính sách năng lượng Mỹ tới môi trường
Hewlett-Packard	Công ty máy tính phát triển một HHTQĐ được gọi là Quality Decision Management để hỗ trợ nâng cao chất lượng sản phẩm và dịch vụ
State of Virginia	Bang Virginia phát triển HHTQĐ đánh giá vận chuyển (the Transportation Evacuation Decision Support System: TEDSS) để xác định con đường tốt nhất để người dân tránh được tai họa hạt nhân từ các nhà máy điện hạt nhân.

Năng lực của HHTQĐ

- Giới thiệu
  - Phát triển HHTQĐ nhằm mục đích:
    - Linh hoạt hơn HTTTQL
    - Tăng cường năng lực hỗ trợ ra quyết định trong nhiều tình huống
  - HHTQĐ hỗ trợ giải vấn đề
    - toàn bộ/hầu hết các khâu
    - Kiểu thường xuyên quyết định
    - Kiểu cấu trúc vấn đề
  - Đối tượng sử dụng HHTQĐ: người ra quyết định ở mọi mức
- Tiếp cận HHTQĐ
  - Hỗ trợ mọi mức của quá trình ra quyết định
  - HHTQĐ chỉ thi hành một vài năng lực
  - Mục đích và phạm vi sử dụng một HHTQĐ quy định tập con năng lực của HHTQĐ đó

HHTQĐ: Tập năng lực

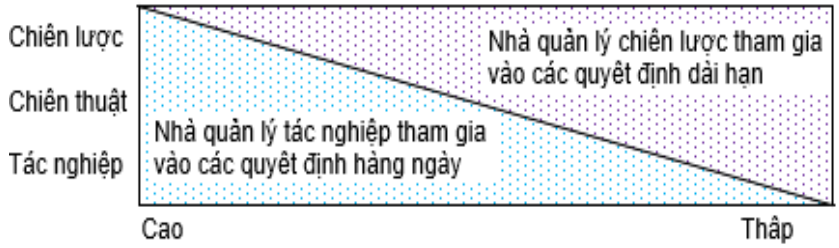
- Hỗ trợ các giai đoạn giải vấn đề
  - Hỗ trợ một vài pha trong thu thập TT, thiết kế, lựa chọn, thực thi, giám sát
  - Hỗ trợ nhiều tiếp cận trong mỗi pha: linh hoạt cho người ra QĐ
- Hỗ trợ ra quyết định thường xuyên khác nhau
  - Từ QĐ đơn nhất (one-of-a-kind) tới QĐ được lặp đi lặp lại
  - QĐ đơn nhất:
    - xuất hiện chỉ một vài lần trong cuộc sống của tổ chức
    - doanh nghiệp nhỏ: có thể chỉ xảy ra một lần
    - Xây dựng nhà máy tài khu vực khác trong nước
    - HHTQD chuyên biệt (ad hoc DSS)
  - QĐ lặp đi lặp lại:
    - Vài lần hoặc hơn trong một năm
    - HHTQĐ tổ chức (institutional DSS)
    - Vài lần/năm và được tinh chỉnh theo thời gian: Vấn đề Danh mục đầu tư, quyết định đầu tư, lập lịch sản xuất
    - Vài lần/ngày: Giải vấn đề dựa vào máy tính, DSS giám sát từng giây

HHTQĐ: Tập năng lực (tiếp)

- Hỗ trợ giải vấn đề với nhiều mức cấu trúc
  - Cấu trúc được và lập trình được cao → không cấu trúc và không lập trình được
  - Vấn đề cấu trúc được: các sự kiện và quan hệ đã biết → HHTQĐ đơn giản
  - Vấn đề không cấu trúc và nửa cấu trúc → HHTQĐ phức tạp
    - quan hệ giữa các DL không tường minh,
    - DL ở nhiều định dạng khó thao tác
    - Yêu cầu thông tin quyết định có thể chưa biết trước
    - DSS hỗ trợ phân tích đầu tư tinh vi và không cấu trúc → tạo lợi nhuận đáng kể cho thương nhân và nhà đầu tư
    - Một vài phần mềm DSS lập trình đặt lệnh mua và bán tự động

HHTQĐ: Tập năng lực (tiếp)

- Hỗ trợ nhiều cấp độ ra quyết định
  - Trong một tổ chức: một HHTQĐ hỗ trợ nhiều cấp
  - Mức quản lý tác nghiệp: ra QĐ hàng ngày và thường xuyên
  - Mức QĐ chiến thuật: lập kế hoạch và kiểm soát đúng cách
  - Mức QĐ chiến lược: thông tin dài hạn



47

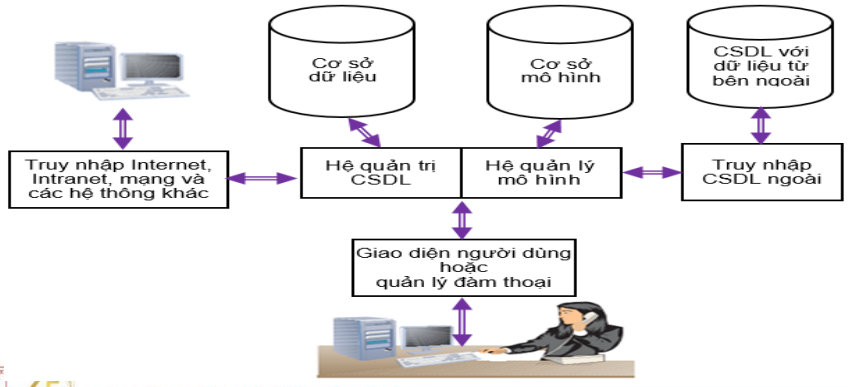
So sánh HHTQĐ với HTTT quản lý

Yếu tố	*HHTQĐ*	*HTTTQL*
Loại vấn đề	HHTQĐ có thể giải quyết vấn đề phi cấu trúc mà không dễ dàng lập trình	HTTTQL thường chỉ được dùng để giải quyết vấn đề cấu trúc được
Kiểu người sử dụng	Một HHTQĐ hỗ trợ cá nhân, nhóm nhỏ, và toàn bộ tổ chức. Người sử dụng thường có quyền kiểm soát nhiều hơn đối với HHTQĐ	HTTTQL chủ yếu hỗ trợ tổ chức. Người sử dụng ít có quyền kiểm soát HTTTQL
Kiểu hỗ trợ	HHTQĐ hỗ trợ mọi khía cạnh và mọi giai đoạn ra quyết định, nó không thay thế việc ra quyết định, con người phải ra quyết định	Một vài HTTTQL ra quyết định tự động và thay thế người ra quyết định.
Nhấn mạnh	HHTQĐ nhấn mạnh quyết định thực tiễn và phong cách ra quyết định	HTTTQL thường chỉ nhấn mạnh tới thông tin.
Tiếp cận	HHTQĐ là một hệ hỗ trợ trực tiếp, cung cấp báo cáo tương tác trên màn hình máy tính	HTTTQL thường là hệ hỗ trợ gián tiếp khi dùng báo cáo ra thường xuyên.
Hệ thống	Thiết bị cung cấp HTQĐ thường là trực tuyến (kết nối trực tiếp với hệ thống máy tính), liên quan thời gian thực (cung cấp kết quả ngay lập tức). Trạm cuối và màn hình hiển thị: ví dụ cung cấp thông tin trả lời tức thời cho câu hỏi	HTTTQL dùng các báo cáo được in ra, được chuyển giao cho người quản lý một lần mỗi tuần, không cung cấp kết quả ngay lập tức.
Tốc độ	Vì HHTQĐ là linh hoạt, được thực hiện bởi người sử dụng, nó thường mất ít thời gian hơn để phát triển và có khả năng đáp ứng yêu cầu người sử dụng tốt hơn	Thời gian đáp ứng của HTTTQL thường là dài hơn.
Đầu ra	Báo cáo từ HHTQĐ thường định hướng màn hình, cùng năng lực tạo báo cáo tại máy in	HTTTQL thường được định hướng in ra các báo cáo và tài liệu.
Phát triển	Người sử dụng HHTQĐ thường liên quan trực tiếp hơn tới sự phát triển HHTQĐ. Sự tham gia của người sử dụng thường theo nghĩa hệ thống tốt hơn để cung cấp hỗ trợ tốt hơn. Với mọi HHTQĐ, sự tham gia của người sử dụng là yếu tố quan trọng nhất cho sự phát triển thành công hệ thống	HTTTQL thường có vòng đời nhiều năm và thường được phát triển cho những người không còn thực thi các công việc được hỗ trợ bởi HTTTQL đó

48

5. Các thành phần của HHTQĐ

- Thành phần chung
  - Cốt lõi: CSDL và Cơ sở mô hình
  - Giao diện người dùng (bộ quản lý hội thoại: dialogue manager)
  - Hình vẽ: Mô hình khái niệm HHTQĐ



HHTQĐ: Cơ sở dữ liệu

- Đặc trưng
  - Cho phép phân tích định tính để ra quyết định
  - DL nội bộ đa dạng của công ty: CSDL, kho DL, kho DL chuyên đề
- HHTQĐ định hướng DL
  - Data-driven DSS (DDSS)
  - Lấy thông tin hàng tồn kho, bán hàng, nhân viên, sản xuất, tài chính, kế toán…hỗ trợ quyết định để giảm chi phí hàng tồn kho
  - Khai phá dữ liệu và thông minh kinh doanh (BI)
  - DDSS y tế: bác sĩ truy cập hồ sơ y tế đầy đủ của bệnh nhân
  - Lưu ý vấn đề riêng tư
  - Có thể kết nối lấy DL ngoài

HHTQĐ: Cơ sở mô hình

- Đặc trưng
  - Cho phép phân tích định lượng để ra quyết định
  - Dữ liệu nội bộ và bên ngoài
  - Cơ sở mô hình: các mô hình miền bài toán
- HHTQĐ định hướng mô hình
  - Model-driven DSS (MDSS)
  - Quản lý mô hình cho phép người dùng truy cập nhiều mô hình, tạo kịch bản theo mô hình và trực quan hóa kết quả
  - Procter & Gamble: MDSS sắp xếp hợp lý hóa dòng chảy nguyên vật liệu và sản phẩm từ các nhà cung cấp tới khách hàng: tiết kiệm được hàng trăm triệu US$ chi phí chuỗi cung ứng.
  - Tiện lợi cho dự đoán hành vi khách hàng
  - LoanPerformance (www.loanperformance.com): dùng MDSS hỗ trợ dự báo khách hàng có thể trễ hoặc vỡ nợ
  - Highmark (bảo hiểm y tế): dùng MDSS để dự đoán gian lận

HHTQĐ: Cơ sở mô hình (tiếp)

- Phần mềm quản lý mô hình
  - Model management software | hệ thống quản lý mô hình
  - các mô hình: tài chính, thống kê, đồ họa, quản lý dự án
  - Sử dụng một hoặc phối hợp nhiều mô hình theo nhu cầu

Loại mô hình	*Mô tả*	*Phần mềm*
Tài chính	Cung cấp dòng tiền mặt, tỷ lệ hoàn vốn nội bộ, và phân tích đầu tư khác	Bảng tính như Microsoft Excel
Thống kê	Cung cấp thống kê tổng hợp, dự báo xu hướng, kiểm tra giả thuyết, v.v.	Chương trình thống kê như SPSS hoặc SAS
Đồ họa	Hỗ trợ ra quyết định trong thiết kế, phát triển và dụng màn hình đồ họa dữ liệu và thông tin	Chương trình đồ họa như Microsoft PowerPoint
Quản lý dự án	Xử lý & điều phối các dự án lớn, cũng dùng để xác định các hoạt động & bài toán then chốt mà có thể trì hoãn hoặc gây nguy hiểm cho toàn bộ dự án nếu không được hoàn thành kịp thời và chi phí hiệu quả	Phần mềm quản lý dự án như Microsoft Project

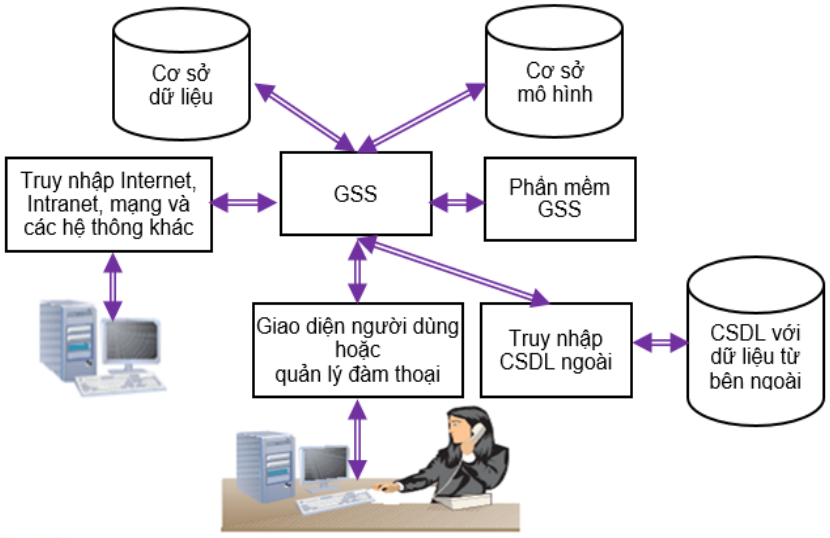
HHTQĐ: Giao diện người dùng

- Vai trò
  - user interface / dialogue manager
  - Cho phép tương tác người dùng với DSS để nhận thông tin
  - Hỗ trợ mọi phương diện truyền thông giữa người dùng và phần cứng &phần mềm tạo thành DSS
  - Quan niệm người dùng: Giao diện chính là DSS
- Người ra quyết định mức trên
  - ít quan tâm: nơi mà thông tin đến hoặc cách thông tin được thu thập
  - Quan tâm nhiều: thông tin dễ hiểu và dễ truy cập
  - Giao diện thân thiện, phù hợp người dùng

6. Hệ hỗ trợ làm việc nhóm

- GSS và vai trò
  - Group Support System | Group Decision Support System
  - DSS hỗ trợ ra quyết định ở mức cá nhân
  - Đặc trưng của tổ chức: hoạt động nhóm
  - Ra quyết định ở mức trên (chiến lược và chiến thuật) cần làm việc nhóm
  - GSS hỗ trợ ra quyết định ở mức nhóm làm việc trên máy tính
  - GSS = DSS + phần mềm hỗ trợ hiệu quả môi trường ra quyết định nhóm
- Ứng dụng
  - dùng trong hầu hết ngành công nghiệp, chính phủ và quân đội
  - Kiến trúc sư+kiến trúc sư và nhà xây dựng tạo kế hoạch tốt nhất và hợp đồng cạnh tranh
  - Nhà sản xuất: DSS nối nhà cung cấp nguyên liệu tới HT của họ
  - Mathcad Enterprise: cho phép tạo, chia sẻ, và tái sử dụng tính toán
  - Có thể dùng phương tiện xã hội cho GSS

Mô hình GSS = DSS + phần mềm GSS



55

Đặc trưng GSS: nâng cao ra quyết định

- Nâng cao quyết định
  - "Hai cái đầu tốt hơn một"; "Một cây – không, ba cây – hòn núi cao"
  - Độc đáo để có quyết định tốt hơn
  - DSS hỗ trợ cá nhân + độc đáo làm việc nhóm
  - Độc đáo: có thể không là mặt đối mặt
- Đặc trưng: Thiết kế đặc biệt
  - Thủ tục đặc biệt với thiết bị, tiếp cận độc đáo
  - Thủ tục: thúc đẩy tư duy sáng tạo, truyền thông hiệu quả, và kỹ thuật ra quyết định nhóm tốt
- Dễ sử dụng
  - GSS phải dễ hiểu và dễ sử dụng
- Linh hoạt
  - Thành viên nhóm: phong cách và sở thích riêng. Một mặt cần nâng cao kỹ năng làm việc nhóm song GSS cũng cần linh hoạt

56

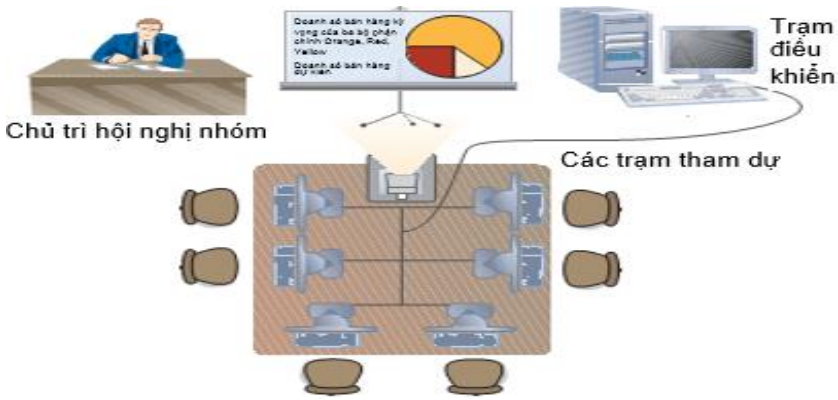
Các đặc trưng GSS

- Hỗ trợ ra quyết định
  - hỗ trợ các phương pháp ra quyết định khác nhau: *phương pháp Delphi, phương pháp khơi nguồn ý tưởng* (*brainstorming*), *đồng thuận nhóm* (*group consensus approach*), *kỹ thuật nhóm danh nghĩa* (*nominal group technique*),
- Đầu vào ẩn danh
  - *anonymous*: người nạp dữ liệu được "ẩn danh" với các thành viên khác trong nhóm (không biết ai đã nạp dữ liệu).
  - Đầu vào ẩn danh: người ra quyết định nhóm tập trung vào giá trị đầu vào mà không xem xét người nào đã ra mỗi quyết định
  - Bỏ phiếu kín
  - sử dụng đầu vào ẩn danh đưa ra quyết định tốt hơn <> có thể dẫn đến tình huống "thiêu đốt"(*flaming*), thành viên nhóm ẩn danh viết ra lời lăng mạ hoặc thậm chí những lời tục tĩu trên GSS

Các đặc trưng GSS

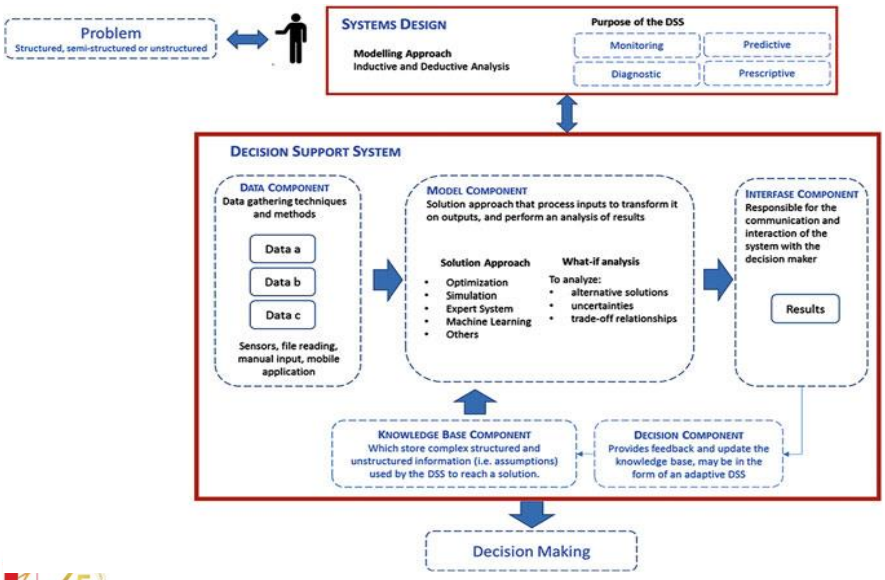
- Giảm thiểu hành vi nhóm tiêu cực
  - thiêu đốt khi đầu vào ẩn danh: một dạng hành vi nhóm tiêu cực
  - Thông đồng, cá nhân chi phối
  - tư duy tập thể (*groupthink*)
  - Thủ tục lập kế hoạch và quản lý các cuộc họp nhóm hiệu quả
- Truyền thông song song và đơn nhất
  - mọi thành viên giải quyết vấn đề/có ý kiến cùng một lúc bằng cách nhập ý kiến từ máy tính cá nhân hoặc máy trạm
  - nhận định và vấn đề được hiển thị tức thì trên máy tính cá nhân hoặc trạm làm việc
  - Truyền thông song song (*parallel communication*) , truyền thông hợp nhất (*unified communications*)
- Lưu trữ hồ sơ tự động
  - lưu giữ tự động hồ sơ chi tiết của cuộc họp nhóm
  - bình luận, ý kiến được nhập vào từ máy tính hoặc máy trạm của một nhóm thành viên có thể được ghi ẩn danh

Phần mềm hệ hỗ trợ nhóm



- Phần mềm GSS (*groupware/ workgroup software*).
- Lịch biểu điện tử dùng chung (*shared electronic calendars*)
- Phần mềm GSS ngày càng được tích hợp vào gói phần mềm hiện có (gói HTXLGD và hệ thống ERP)

Hệ HTQĐ ví dụ



Các thành phần cốt lõi của một DSS hậu cần cảng và chuỗi cung ứng hàng hải

7. HT hỗ trợ điều hành

- Khái niệm
  - *Executive Support System*: ESS
  - HHTQĐ chuyên dụng hỗ trợ giám đốc điều hành cấp cao
  - ESS, còn được gọi là hệ thống thông tin điều hành (*Executive Information System*: EIS), hỗ trợ hoạt động ra quyết định của các thành viên Ban giám đốc, người chịu trách nhiệm cho các cổ đông
  - Các tầng như hình vẽ

Yêu cầu và khả năng HT hỗ trợ điều hành

- Yêu cầu
  - Phù hợp với cá nhân giám đốc điều hành,
  - Dễ sử dụng,
  - Có khả năng khoan xuống,
  - Hỗ trợ nhu cầu về dữ liệu bên ngoài,
  - Trợ giúp trong các tình huống có mức độ không chắc chắn cao,
  - Có định hướng tương lai,
  - Được liên kết với các quá trình kinh doanh giá trị gia tăng
- Khả năng
  - Hỗ trợ xác định một tầm nhìn tổng thể
  - Hỗ trợ lập kế hoạch chiến lược (*Strategic planning*)
  - Hỗ trợ tổ chức và nhân sự chiến lược
  - Hỗ trợ kiểm soát chiến lược
  - Hỗ trợ quản lý khủng hoảng

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

63

25 SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!

soict.hust.edu.vn/ fb.com/groups/soict

64