Data Management and Visualization

Dr. Ashish Kumar Jha

Session 6

Advanced concepts in Database

Agenda

- Database Security
- Need and types of DB Security
- Business Intelligence
- NOSQL

Database security

Mechanisms that protect the database against intentional or accidental threats.

Not only apply to the data held in a database. Breaches of security may affect other parts of the system, which may in turn affect the database.

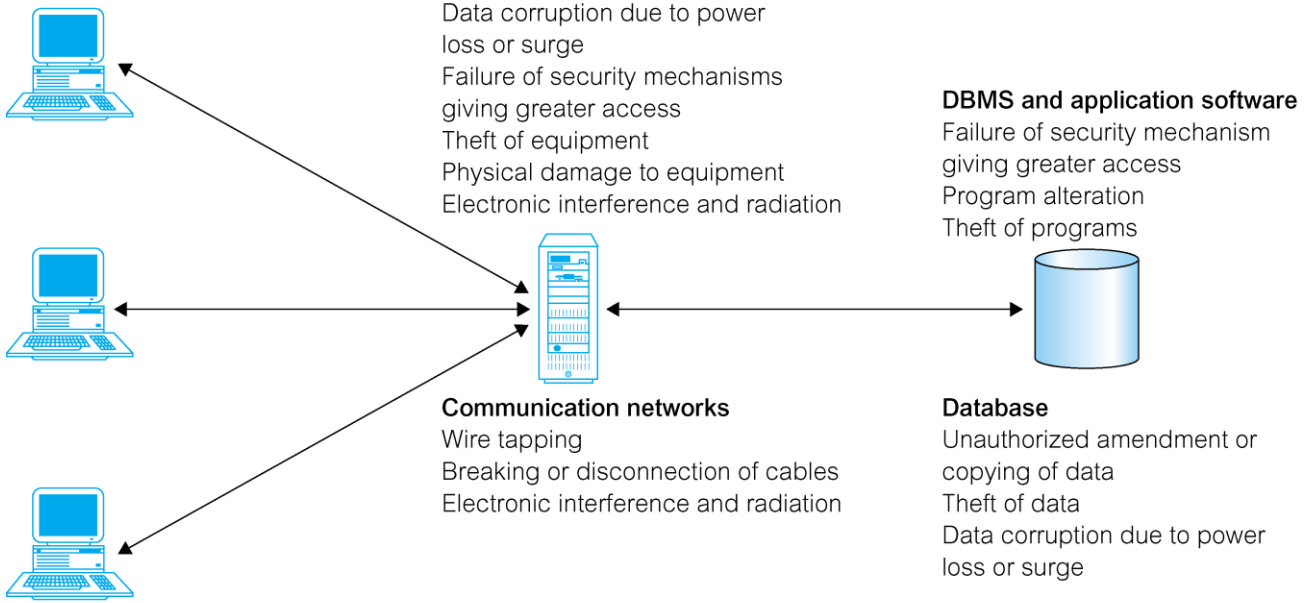
Includes hardware, software, people, and data.

Growing importance of security is the increasing amounts of crucial corporate data being stored on computer.

Examples of threats and possible outcomes

Threat	*Theft and fraud*	*Loss of confidentiality*	*Loss of privacy*	*Loss of integrity*	*Loss of availability*
Using another person's means of access	√	√	√		
Unauthorized amendment or copying of data	√			√	
Program alteration	√			√	√
Inadequate policies and procedures that allow a mix of confidential and normal output	√	√	√		
Wire tapping	√	√	√		
Illegal entry by hacker	√	√	√		
Blackmail	√	√	√		
Creating 'trapdoor' into system	√	√	√		
Theft of data, programs, and equipment	√	√	√		√
Failure of security mechanisms, giving greater access than normal	√	√	√		
Staff shortage or strikes				√	√
Inadequate staff training		√	√	√	√
Viewing and disclosing unauthorized data	√	√	√		
Electronic interference and radiation				√	√
Data corruption due to power loss or surge				√	√
Fire (electrical fault, lightning strike, arson), flood, hurricane, bomb				√	√
Physical damage to equipment				√	√
Breaking cables or disconnection of cables				√	√
Software (DBMS) and Operating System crashes				√	√
Exposure to viruses				√	√

Summary of threats to computer systems



Hardware
Fire/flood/bombs
Data corruption due to power loss or surge
Failure of security mechanisms giving greater access
Theft of equipment
Physical damage to equipment
Electronic interference and radiation

DBMS and application software
Failure of security mechanism giving greater access
Program alteration
Theft of programs

Communication networks
Wire tapping
Breaking or disconnection of cables
Electronic interference and radiation

Database
Unauthorized amendment or copying of data
Theft of data
Data corruption due to power loss or surge

Users
Using another person's means of access
Viewing and disclosing unauthorized data
Inadequate staff training
Illegal entry by hacker
Blackmail
Introduction of viruses

Programmers/operators
Creating trapdoors
Program alteration (such as creating software that is insecure)
Inadequate staff training
Inadequate security policies and procedures
Staff shortages or strikes

Data/database administrator
Inadequate security policies and procedures

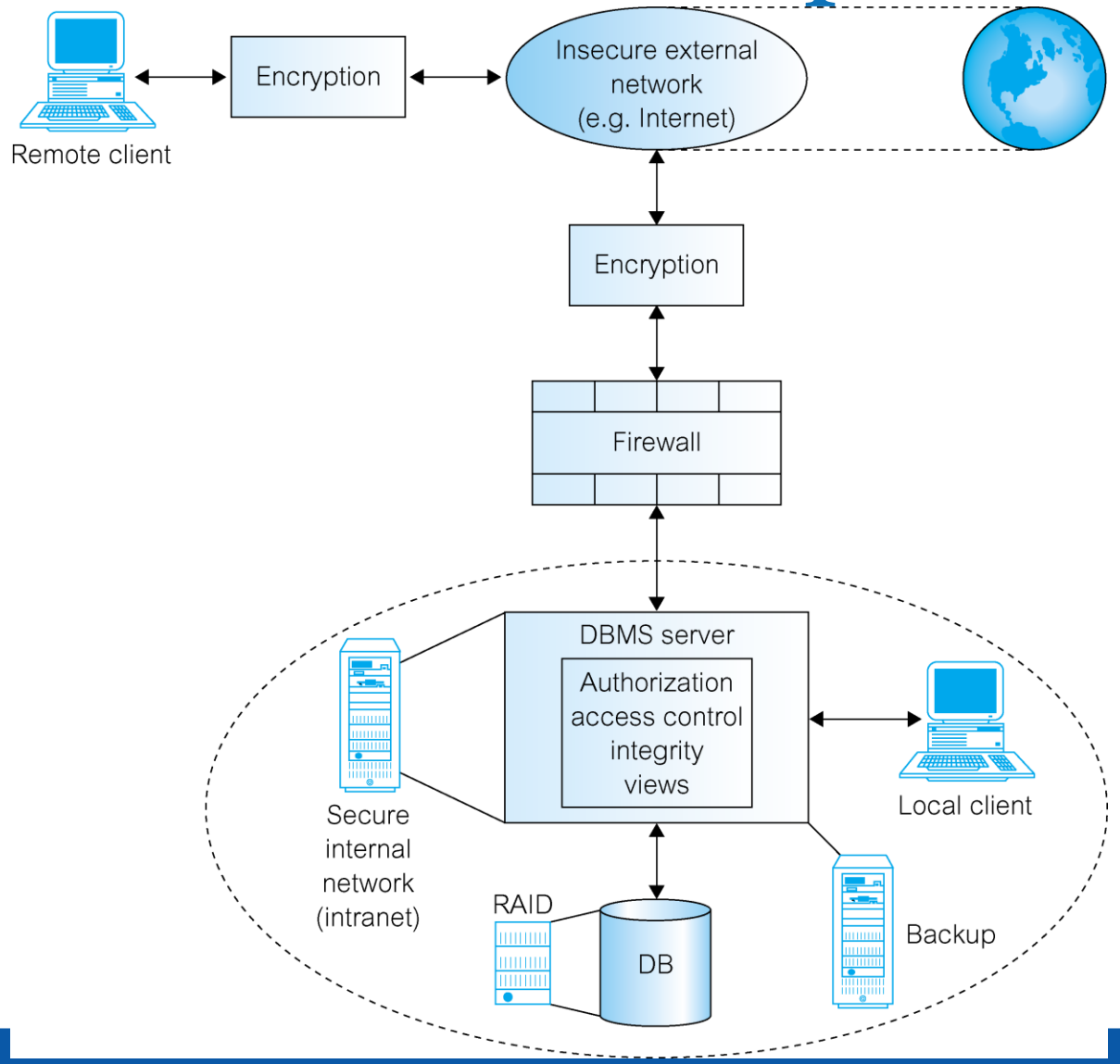
Database security

Threat is any situation or event, whether intentional or unintentional, that may adversely affect a system and consequently the organization.

Outcomes to avoid:

- theft and fraud,
- loss of confidentiality (secrecy),
- loss of privacy,
- loss of integrity,
- loss of availability.

Typical multi-user computer environment



Database security

Computer-based countermeasures include:

- authorization,
- views,
- backup and recovery,
- integrity,
- encryption,
- redundant array of independent disks (RAID).

Countermeasures - computer-based controls

Authorization

- The granting of a right or privilege that enables a subject to have legitimate access to a database system or a database system's object.

Authentication

- A mechanism that determines whether a user is, who he or she claims to be. failure.

Privilege

- A right granted by one user to allow another user or group of users access to a database system or an object in the database system.

Countermeasures - computer-based controls

Views

- A view is a virtual table that does not necessarily exist in the database but can be produced upon request by a particular user, at the time of request.

Backup and recovery

- Process of periodically taking a copy of the database and log file (and possibly programs) onto offline storage media.

Countermeasures - computer-based controls

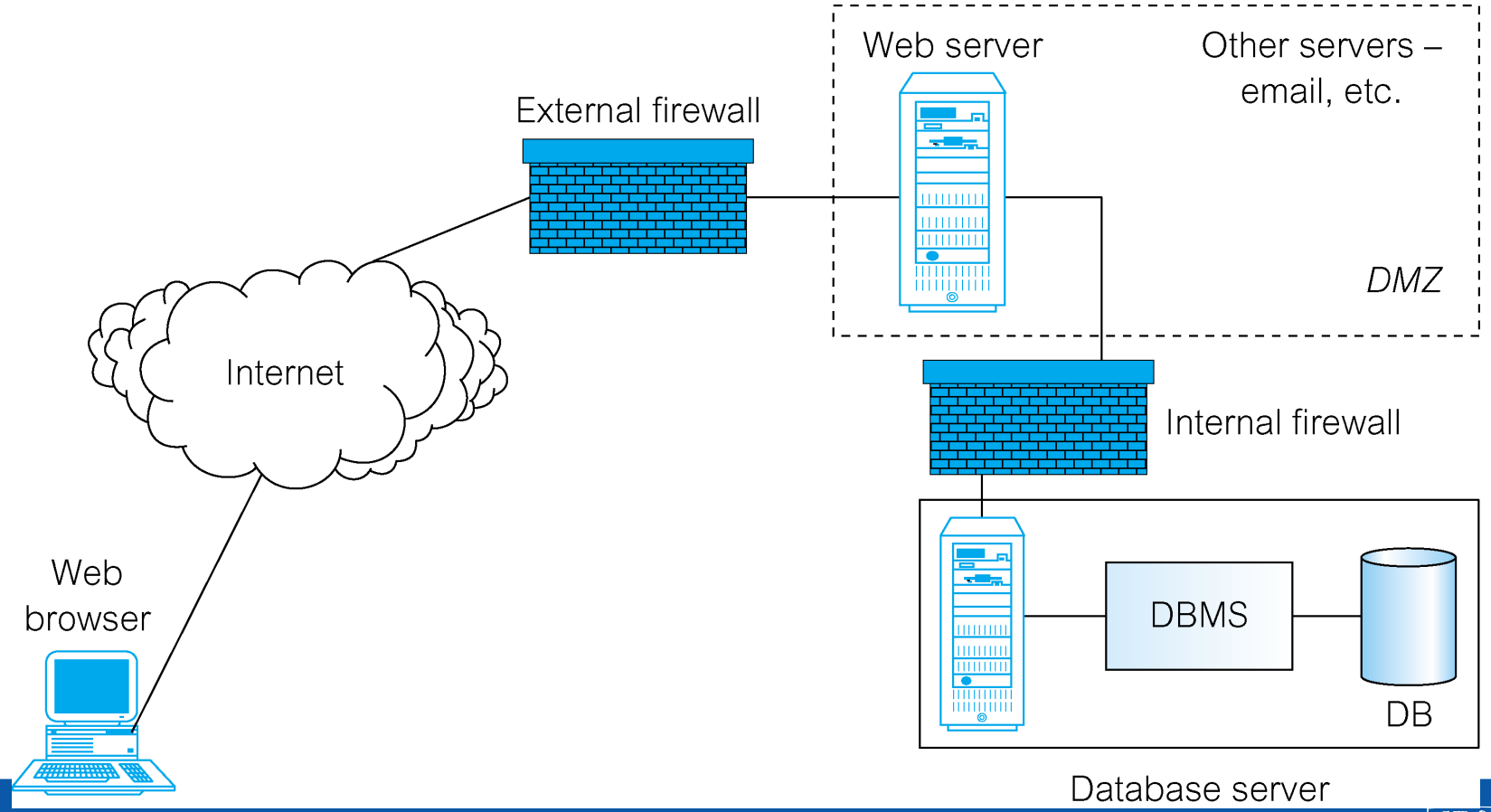
Integrity

- Prevents data from becoming invalid, and hence giving misleading or incorrect results.

Encryption

- Encoding the data by a special algorithm that renders the data unreadable by any program without the decryption key.

Network security architecture three tier database system architecture



Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Trinity Business School

Business Intelligence

Business intelligence

The processes for collecting and analyzing data, the technologies used in these processes, and the information obtained from these processes with the purpose of facilitating corporate decision–making.

The main technologies associated with business intelligence includes:

- data warehouse,
- online analytical processing (OLAP),
- data mining.

Data warehouse

A database system that is designed to support decision-making by presenting an integrated view of corporate data that is copied from disparate data sources.

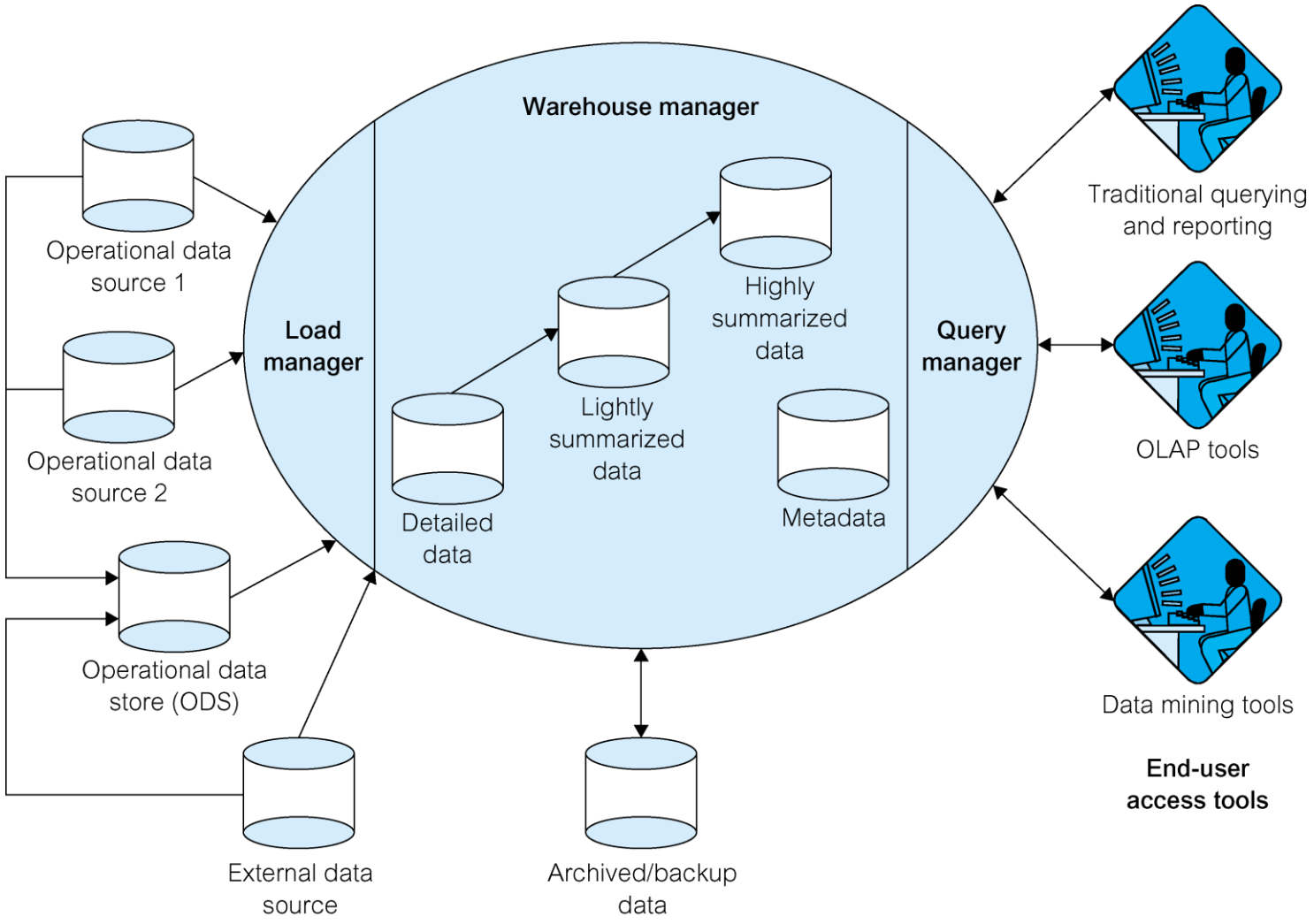
Data held in a data warehouse is described as being subject-oriented, integrated, time-variant, and non-volatile (Inmon, 1993).

The main source of data for the data warehouse are online transaction processing (OLTP) systems.

Comparison of OLTP with data warehousing

OLTP Systems	*Data Warehousing*
Holds current data	Holds historic data
Stores detailed data	Stores detailed, lightly, and highly summarized data
Data is dynamic	Data is largely static
Repetitive processing	*Ad hoc*, unstructured, and heuristic processing
High level of transaction throughput	Medium to low level of transaction throughput
Predictable pattern of usage	Unpredictable pattern of usage
Transaction driven	Analysis driven
Application oriented	Subject oriented
Supports day-to-day operational decisions	Supports tactical and strategic decisions
Serves large number of users	Serves lower number of users

Typical architecture of a data warehouse



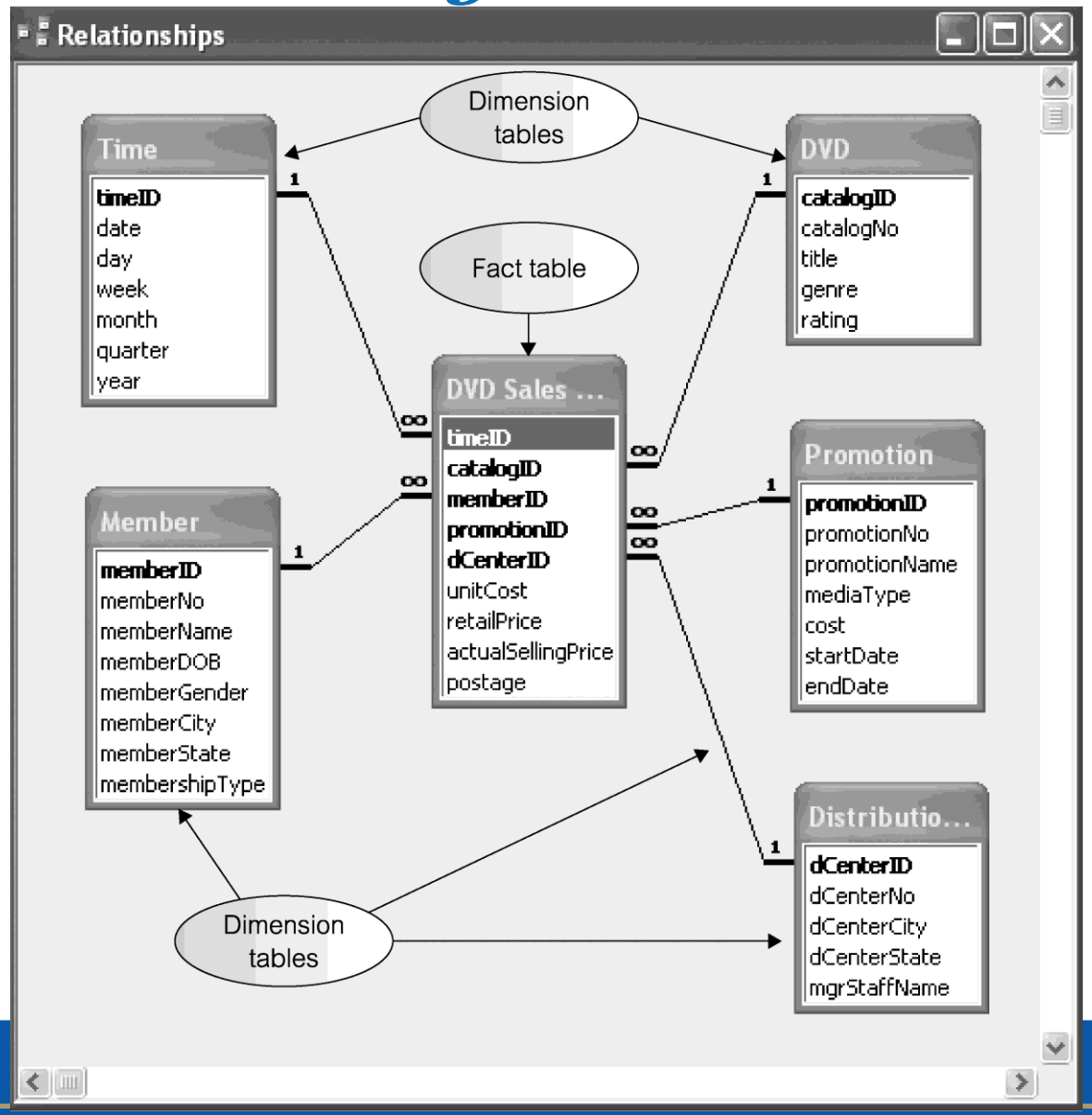
Dimensionality modeling

Creates a dimensional model (DM) called a star schema that has a fact table containing factual data in the center, surrounded by smaller dimension tables containing denormalized reference data.

As the bulk of data is represented as facts, the fact tables can be extremely large relative to the dimension tables.

Dimension tables contain descriptive textual information and are used as the constraints (search conditions) in queries on the fact data.

Star schema for *StayHome* DVD sales



Online analytical processing (OLAP)

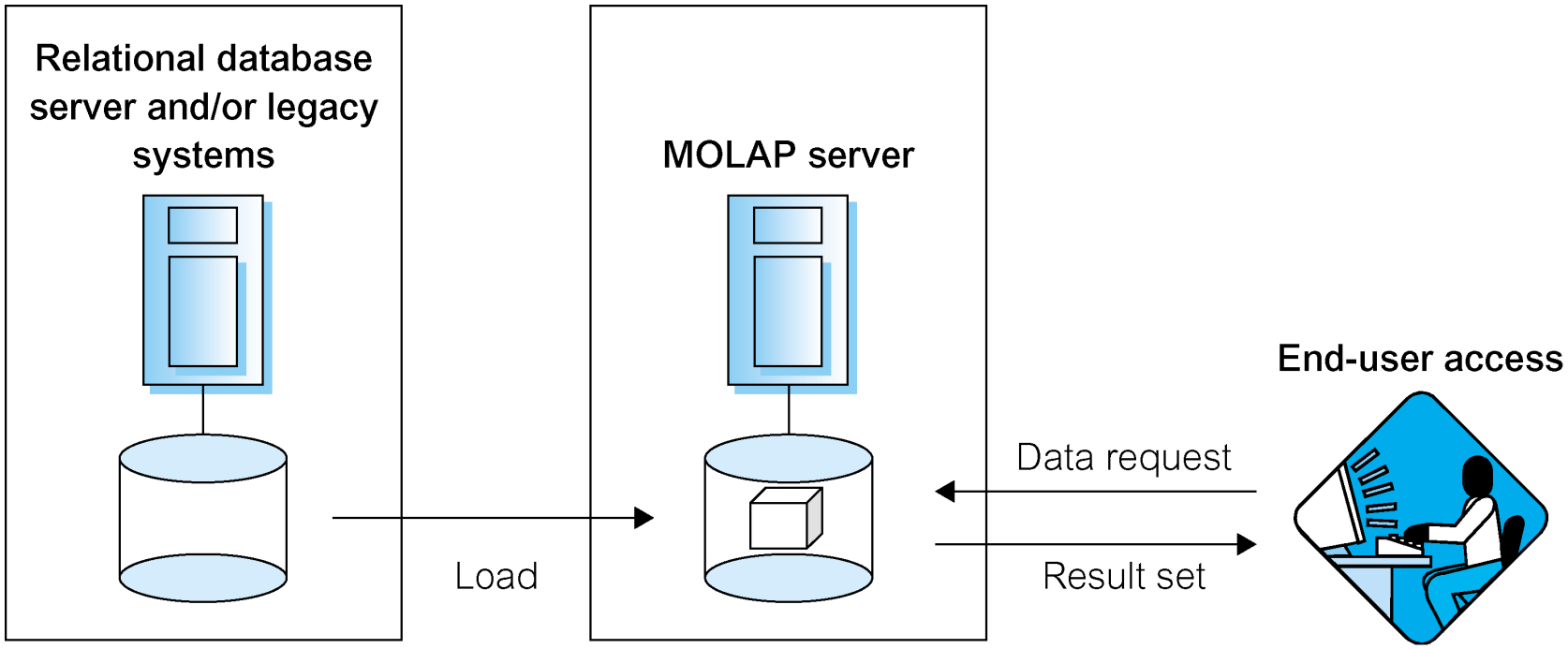
Stores large volumes of multi-dimensional data that is aggregated (summarized) to various levels of detail to support advanced analysis of this data.

Multi-dimensional data can be characterized through many different views. For example DVD sales can be viewed by product, customer, and/or sales channel.

Examples of OLAP applications

Business Area	*Examples of OLAP Applications*
Finance	Budgeting, activity-based costing, financial performance analysis, and financial modeling.
Sales	Sales analysis and sales forecasting.
Marketing	Market research analysis, sales forecasting, promotions analysis, customer analysis, and market/customer segmentation.
Manufacturing	Production planning and defect analysis.

Typical architecture for multi-dimensional OLAP (MOLAP)



Data mining

The process of extracting valid, previously unknown, comprehensible, and actionable knowledge from large databases and using it to provide decision-support.

Examples of data mining applications

Business Area	*Examples of OLAP Applications*
Retail / Marketing	Identifying buying patterns of customers Finding associations among customer demographic characteristics Predicting response to mailing campaigns Market basket analysis
Banking	Detecting patterns of fraudulent credit card use Identifying loyal customers Predicting customers likely to change their credit card affiliation Determining credit card spending by customer groups
Insurance	Claims analysis Predicting which customers will buy new policies
Medicine	Characterizing patient behavior to predict surgery visits Identifying successful medical therapies for different illnesses

Data warehousing and data mining

Major challenge to exploit data mining is identifying suitable data to mine.

Data mining requires a single, separate, clean, integrated, and self-consistent source of data.

A data warehouse is well equipped for providing data for mining.

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Trinity Business School

NO SQL


I THINK WE SHOULD BUILD AN SQL DATABASE.
UH-OH
DOES HE UNDERSTAND WHAT HE SAID OR IS IT SOMETHING HE SAW IN A TRADE MAGAZINE AD?
WHAT COLOR DO YOU WANT THAT DATABASE?
I THINK MAUVE HAS THE MOST RAM.
E-mail: SCOTTADAMS@AOL.COM
© 1995 United Feature Syndicate, Inc.(NYC)

History of the World

Relational Databases – mainstay of business

Web-based applications caused spikes

- Especially true for public-facing e-Commerce sites

Developers begin to front RDBMS with memcache or integrate other caching mechanisms within the application

SQL

Specialized data structures

Shines with complicated queries

Focus on fast query & analysis

- Not necessarily on large datasets

What is NoSQL?

Stands for Not Only SQL

Class of non-relational data storage systems

Usually do not require a fixed table schema nor do they use the concept of joins

All NoSQL offerings relax one or more of the ACID properties (will talk about the CAP theorem)

NoSQL

Non relational

Scalability

- Vertically
  - Add more data
- Horizontally
  - Add more storage

Collection of structures

- Hashtables, maps, dictionaries

No pre-defined schema

No join operations

CAP not ACID

- Consistency, Availability and Partitioning (but not all three at once!)
- Atomicity, Consistency, Isolation and Durability

Advantages of NoSQL

Cheap, easy to implement

Data are replicated and can be partitioned

Easy to distribute

Don't require a schema

Can scale up and down

Quickly process large amounts of data

Relax the data consistency requirement (CAP)

Can handle web-scale data, whereas Relational DBs cannot

Disadvantages of NoSQL

Data is generally duplicated, potential for inconsistency

No standardized schema

No standard format for queries

No standard language

Difficult to impose complicated structures

Depend on the application layer to enforce data integrity

No guarantee of support

Too many options, which one, or ones to pick

How did we get here?

Explosion of social media sites (Facebook, Twitter) with large data needs

Rise of cloud-based solutions such as Amazon S3 (simple storage solution)

Just as moving to dynamically-typed languages (Ruby/Groovy), a shift to dynamically-typed data with frequent schema changes

Open-source community

More Programming and Less Database Design

Alternative to traditional relational DBMS

+ Flexible schema
+ Quicker/cheaper to set up
+ Massive scalability
+ Relaxed consistency → higher performance & availability

– No declarative query language → more programming
– Relaxed consistency → fewer guarantees

Challenge: Coordination

The solution to availability and scalability is to decentralize and replicate functions and data…but how do we coordinate the nodes?

- data consistency
- update propagation
- mutual exclusion
- consistent global states
- group membership
- group communication
- event ordering
- distributed consensus
- quorum consensus

Dynamo and BigTable

Three major papers were the seeds of the NoSQL movement

- BigTable (Google)
- Dynamo (Amazon)
  - Gossip protocol (discovery and error detection)
  - Distributed key-value data store
  - Eventual consistency
- CAP Theorem

CAP Theorem

Proposed by Eric Brewer (Berkeley)

Subsequently proved by Gilbert and Lynch

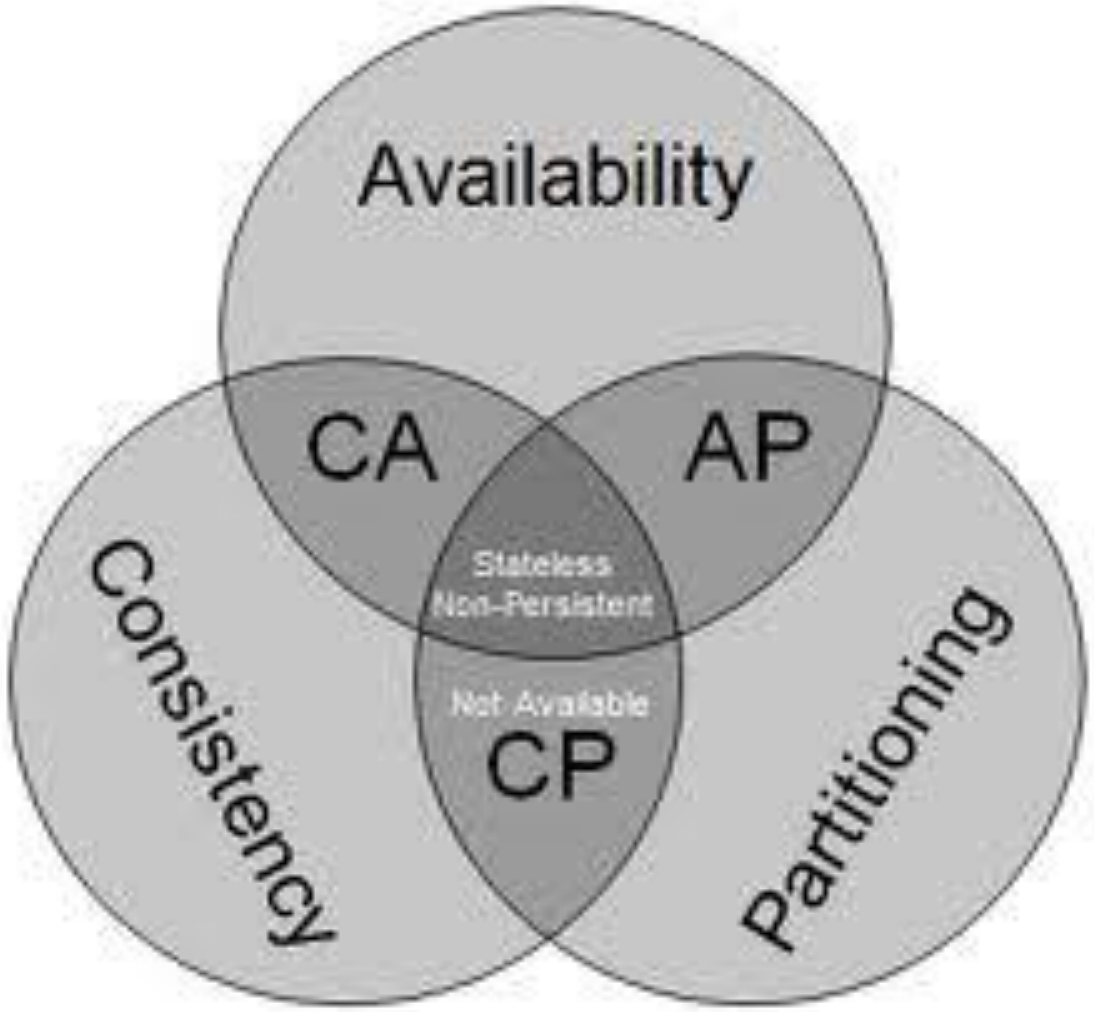
In a distributed system you can satisfy at most 2 out of the 3 guarantees

1. **Consistency**: all nodes have same data at any time
2. **Availability**: the system allows operations all the time
3. **Partition-tolerance**: the system continues to work in spite of network partitions

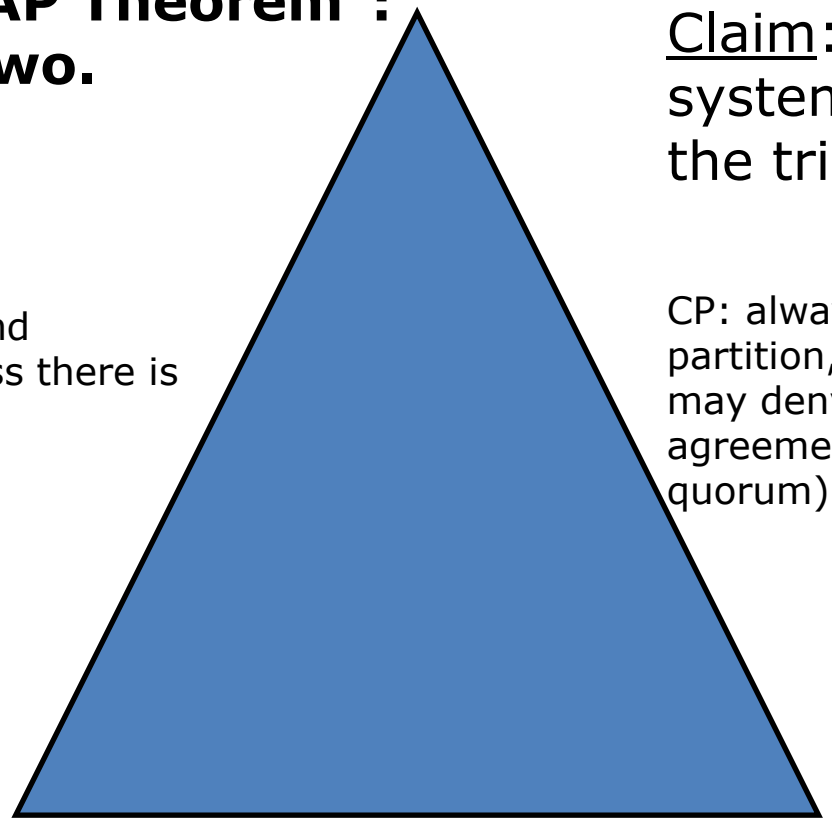
CAP Theorem

- **Three properties of a system: consistency, availability and partitions**
- **"You can have at most two of these three properties for any shared-data system"**
- **To scale out, you have to partition. That leaves either consistency or availability to choose from**
  - In almost all cases, you would choose availability over consistency

CAP Theorem


Availability
CA
AP
Stateless
Non-Persistent
Consistency
Not Available
CP
Partitioning

Fox&Brewer "CAP Theorem": C-A-P: choose two.



Claim: every distributed system is on one side of the triangle.

CA: available, and consistent, unless there is a partition.

CP: always consistent, even in a partition, but a reachable replica may deny service without agreement of the others (e.g., quorum).

AP: a reachable replica provides service even in a partition, but may be inconsistent if there is a failure.

Availability

Traditionally, thought of as the server/process available five 9's (99.999 %).

However, for large node system, at almost any point in time there's a good chance that a node is either down or there is a network disruption among the nodes.

- Want a system that is resilient in the face of network disruption

Consistency Model

A consistency model determines rules for visibility and apparent order of updates. For example:

- Row X is replicated on nodes M and N
- Client A writes row X to node N
- Some period of time t elapses.
- Client B reads row X from node M
- Does client B see the write from client A?
- Consistency is a continuum with tradeoffs
- For NoSQL, the answer would be: maybe
- CAP Theorem states: Strict Consistency can't be achieved at the same time as availability and partition-tolerance.

Eventual Consistency

When no updates occur for a long period of time, eventually all updates will propagate through the system and all the nodes will be consistent

For a given accepted update and a given node, eventually either the update reaches the node or the node is removed from service

Known as BASE (Basically Available, Soft state, Eventual consistency), as opposed to ACID

- Soft state: copies of a data item may be inconsistent
- Eventually Consistent – copies becomes consistent at some later time if there are no more updates to that data item
- Basically Available – possibilities of faults but not a fault of the whole system

What kinds of NoSQL

NoSQL solutions fall into two major areas:

- Key/Value or 'the big hash table'.
  - Amazon S3 (Dynamo)
  - Voldemort
  - Scalaris
- Schema-less which comes in multiple flavors, column-based, document-based or graph-based.
  - Cassandra (column-based)
  - CouchDB (document-based)
  - Neo4J (graph-based)
  - HBase (column-based)