Data integration

Why database integration?

- Combine database when two companies merge
- enhance information using data from different sources
- access data in legacy databases

Related Issues

- Data linkage
  - Identifying records referring to the same real-world entity
  - Computing data similarity
  - Applicability of different similarities
- Info fusion — Extract info from different data source
- Data cleaning
  - Remove noise in original data
  - Remove noise in integrated data — inconsistency and redundancy
- Data quality
  - Data augmentaion
  - Data constraints
  - Data provenance
- Data privacy — Share data with the assurance that private info can not be derived

Difference between Data integration and Info fusion

- Common goal: Integrate and organize data from multiple sources in order to present a unified view of data to derive actionable insights
- Data integration: focuses on combining data to create a bigger and consistent data set
- Info fusion: focuses on deriving insight from real-time streaming data with semantic context from other big data source

Global Info systems

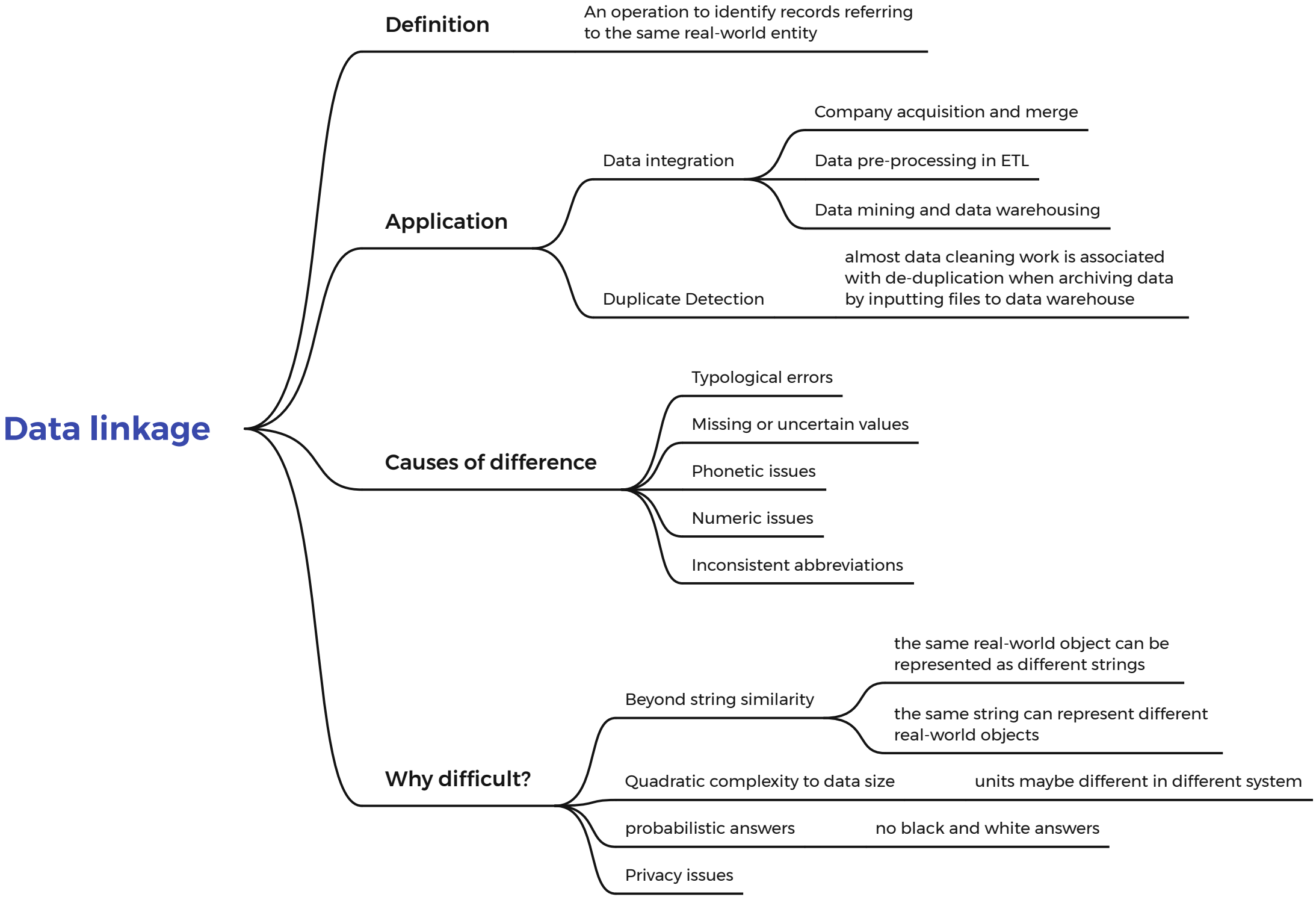
- Federated database(FDB)
  - Semi-autonomous database systems
  - global view is provided
- Multi-databases(MDB) — Autonomous database systems, no or limited global view is provided
- Interoperable info systems
  - Definition
    - Only API provided to communicate with different database
    - No global virtual view
  - Interoperability
    - Ability for an application to access multiple distinct systems
    - Not necessarily for database only
  - Interoperable systems
    - Exchange messages and requests
    - Receive services and operate as a unit towards a common goal
  - Different types of interoperability
    - Syntactic — languages and data formats
    - Semantic — meaning
    - System — machines, networks, database models
    - Structural — data structures and data models

Challenges in DB integration

- Schema heterogeneity
- Type heterogeneity
- Value heterogeneity
- Semantic heterogeneity

Process of DB Integration

- Schema mapping — mapping of structures
- Data mapping — matching based on content
- Data fusion — reconciliation of mismatching content


Data linkage
Definition
An operation to identify records referring to the same real-world entity
Application
Data integration
Company acquisition and merge
Data pre-processing in ETL
Data mining and data warehousing
Duplicate Detection
almost data cleaning work is associated with de-duplication when archiving data by inputting files to data warehouse
Causes of difference
Typological errors
Missing or uncertain values
Phonetic issues
Numeric issues
Inconsistent abbreviations
Why difficult?
Beyond string similarity
the same real-world object can be represented as different strings
the same string can represent different real-world objects
Quadratic complexity to data size
units maybe different in different system
probabilistic answers
no black and white answers
Privacy issues

- **Data quality**
  - **Governance**
    - What the organisation does and what it should become in the future
  - **Management**
    - How the organisation will reach those goals and aspirations
      - solutions
        - organizational
        - architectural
        - computational
          - defining and enforcing data constriants
          - Doing entity resolution effectively and efficiently
          - managing missing values and uncertain data
  - **Data quality dimensions**
    - Integrity
      - Meaningless
    - Accuracy
      - Erroneous
      - the closeness between a value v and value v' considered as the correct representation
    - Completeness
      - Missing
      - the sufficiency of data for the task at hand
    - Currency
      - Obsolete, out of date
      - How promptly data is updated
      - Volatility: frequency with which data vary in time
      - Timeliness: how current the data are for the task at hand
    - Representational consistency
      - inconsistent
    - Accessibility
      - unavailable
    - Uncertainty
      - reliability, trust
  - **Basic steps for governance**
    - Recognize the problem
      - Data quality problem
        - Data acquisition
        - Data integration
        - Data utilization
    - Measure its cost
      - How many errors?
        - Data quality cost
          - Costs caused by low Data quality
            - Direct costs
              - verification costs
              - Re-entry costs
              - Compensation costs
            - Indirect costs
              - Costs based on lower reputation
              - Costs based on wrong Decisions or actions
              - Sunk investment costs
          - Costs of improving or assuring data quality
            - Prevention cost
              - training costs
              - monitoring costs
              - standard development and deployment cost
            - Detection costs
              - Analysis costs
              - reporting costs
            - Repair costs
              - Repair planning costs
              - Repair implementation costs
    - Devise strategy for improvement
      - Short term
        - Define metrics and its relationship to business impact to identify which data to improve
        - Produce baseline
        - Use same metric to measure change from baseline
        - Sustain improvements through ongoing monitoring
      - long term
        - Establish process owner and management team
        - describe process qualitatively and understand requirements
        - Establish measurement system
        - Establish process control and conformance to requirements
        - Identify improvement opportunities
        - Select opportunities and set objectives
        - make and sustain improvements
    - Aim towards the governance maturity

Data warehouse

What is data warehouse

- Definition of DW
  - A storage architecture designed to hold data extracted from transaction systems, operational data stores and external sources
    - data are combined in an aggregate, summary form suitable for enterprise-wide data analysis and reporting for predefined business needs.
    - data are periodically updated and stored for read-only applications
  - A data warehouse is a set of facts perceived by a number of dimensions
- Characteristic of DW
  - Subject oriented
  - Nonvolatile — not usually subject tot changes
  - integrated — data is consistent and integrated from multiple courses
  - Time variant — historical data is recorded for analytical applications

Why we need data warehouse

- Traditional database issues
  - Traditional database applications consist of both updates and queries, some queries are large scale aggregation reports which can take long time to generate on-the-fly
  - Database updates and queries must lock data resources, large scale aggregation reports lock many resources for a long time
  - Organizations are analyzing current and historical data to identify useful patterns and support business strategies
- Benefits of data warehouse
  - Emphasis is on complex,interactive, exploratory analysis of very large datasets created by integrating data from across all parts of an enterprise
- Difference between DW and DB
  - Integrated data spanning long time periods, often augmented with summary information
  - very large volume
  - Interactive response times expected for complex queries
  - Ad-hoc updated uncommon(write-once and Read forever)

DW design

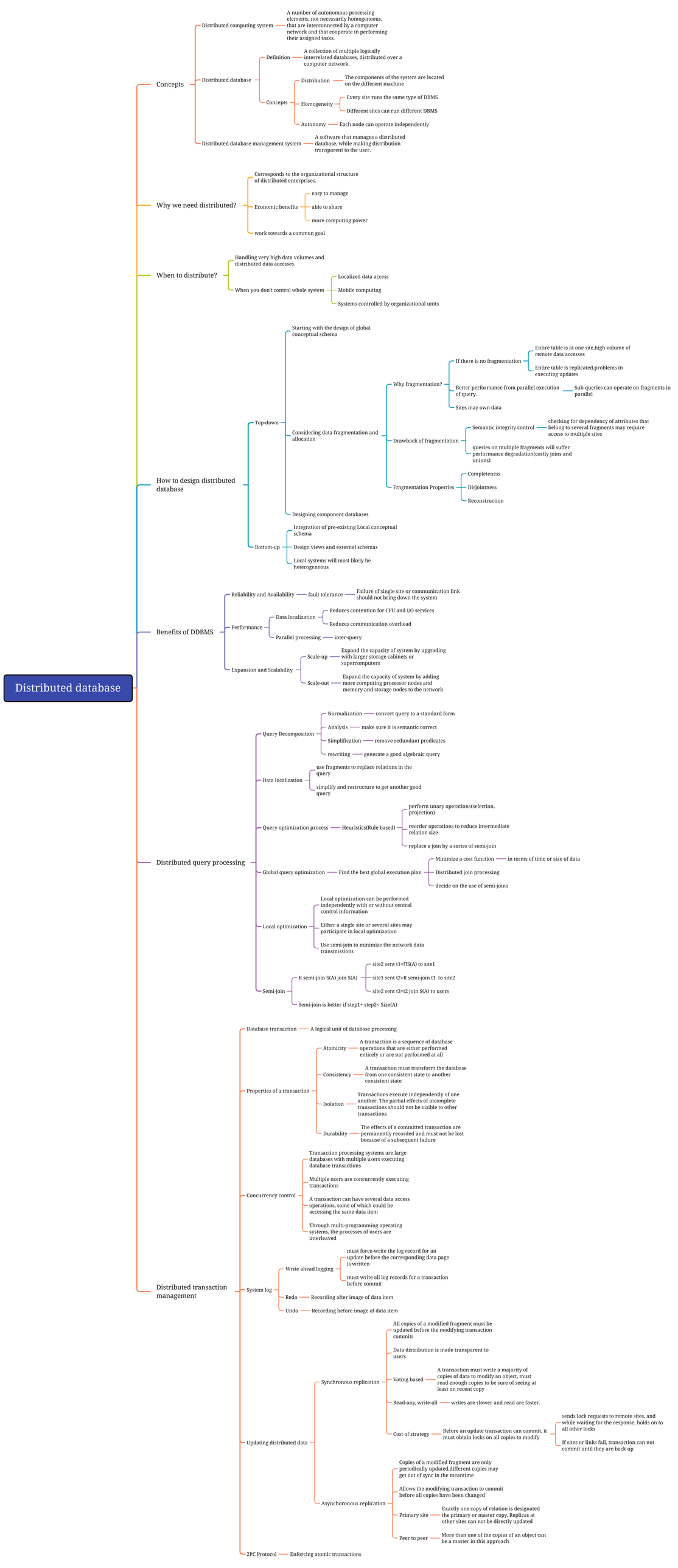
- Star schema — advantage — faster query processing speed
- Snowflake schema
  - fact constellation — A set of fact tables that share some dimension tables
  - advantage
    - less data integrity problem
    - less space consumption

DW implementation

- Data cube
  - OLAP
    - Pivot — Rotate data cube to show a different orientation of axes
    - Roll up — Move up concept hierarchy, grouping into larger units along a dimension with generalization
    - Drill-down — Disaggregate to a finer-grained view to show more details
    - Slice and dice — Perform projection operations on the dimensions
  - Cuboid — A cuboid is a denotation of one of the 2^d summarized views, which can be used for materialized view
- Materialized view
  - Advantages
    - OLAP queries are typically aggregate queries
    - Pre-aggregation is essential for interactive response time
      - Pre-calculate expensive joins
      - Speed up online OLAP queries
  - Disadvantages
    - It increases storage cost
    - The content of the materialized views must be maintained when the underlying details tables are modified
    - Trade off between query performance and accessibility to up-to-date data

Database application

- Big data application
  - Connecting Dots — From small to big
  - Discovering specifics — From big to small — How do we find outliers, predict trends, provide summaries, and give explanations from a dataset.
  - Inferencing — Knowing unknown — Everything is related to everything else
  - Key values — The smallest unit of big data
    - Every key-value is unique with timestamp
    - Key values are associated to each other
    - All key values can be drawn as a
- Network science
  - The curse of dimensionality — Adding extra dimension to a data space will exponentially increase the volume of data space
  - Scale-free Network — the characteristic of network are independent of the size of network
  - Different type of networks
  - Computing issues of complex networks
- Explaining outliers in aggregate queries
- Effective storage of big data
  - Column-based VS Row-based data database stroages
  - Applicability of row and column storages
  - Compare and contrast of row and column storages
    - Row storage
      - Efficient when many columns of a single row are required at the same time
      - Well-suited for OLTP-liked workloads which are more heavily loaded with interactive transactions
    - Column storage
      - Efficient on aggregation operation over many row but only for a smaller subset of all columns of data
      - Efficient when inserting new values of a column for all rows at once
      - Well-suited for OLAP-like workloads
  - NoSQL(not only SQL)
    - Technology
      - Cloud Platform — A viable alternative to relational databases operating on cluster servers
      - No schema — Different types of data is collected, stored, accessed without a schema
      - Data fusion — A data integration technique for multi-source data
      - Flexible access — A query model accessing data without using traditional SQL
    - Property
      - To add/delete/query massive arrays and still allow for persistence and fault tolerance
      - To store objects using key-values
      - To implement large data query on MapReduce framework
    - System
      - CouchDB — A document-oriented database that can be queried and indexed in a MapReduce fashion using JavaScript.
      - MongoDB — A scalable, high-performance, open source, document-oriented database system
      - Hadoop — Hadoop develops open-source software for reliable, scalable, distributed computing
      - HBase — Hadoop database system supports random, real-time, read/write access to big data.


Distributed database
Concepts
Distributed computing system
A number of autonomous processing elements, not necessarily homogeneous, that are interconnected by a computer network and that cooperate in performing their assigned tasks.
Distributed database
Definition
A collection of multiple logically interrelated databases, distributed over a computer network.
Concepts
Distribution
The components of the system are located on the different machine
Homogeneity
Every site runs the same type of DBMS
Different sites can run different DBMS
Autonomy
Each node can operate independently
Distributed database management system
A software that manages a distributed database, while making distribution transparent to the user.
Why we need distributed?
Corresponds to the organizational structure of distributed enterprises.
Economic benefits
easy to manage
able to share
more computing power
work towards a common goal
When to distribute?
Handling very high data volumes and distributed data accesses.
When you don't control whole system
Localized data access
Mobile computing
Systems controlled by organizational units
How to design distributed database
Top-down
Starting with the design of global conceptual schema
Considering data fragmentation and allocation
Why fragmentation?
If there is no fragmentation
Entire table is at one site,high volume of remote data accesses
Entire table is replicated,problems in executing updates
Better performance from parallel execution of query.
Sub-queries can operate on fragments in parallel
Sites may own data
Drawback of fragmentation
Semantic integrity control
checking for dependency of attributes that belong to several fragments may require access to multiple sites
queries on multiple fragments will suffer performance degradation(costly joins and unions)
Fragmentation Properties
Completeness
Disjointness
Reconstruction
Designing component databases
Bottom-up
Integration of pre-existing Local conceptual schema
Design views and external schemas
Local systems will most likely be heterogeneous
Benefits of DDBMS
Reliability and Availability
fault tolerance
Failure of single site or communication link should not bring down the system
Performance
Data localization
Reduces contention for CPU and I/O services
Reduces communication overhead
Parallel processing
inter-query
Expansion and Scalability
Scale-up
Expand the capacity of system by upgrading with larger storage cabinets or supercomputers
Scale-out
Expand the capacity of system by adding more computing processor nodes and memory and storage nodes to the network
Distributed query processing
Query Decomposition
Normalization
convert query to a standard form
Analysis
make sure it is semantic correct
Simplification
remove redundant predicates
rewriting
generate a good algebraic query
Data localization
use fragments to replace relations in the query
simplify and restructure to get another good query
Query optimization process
Heuristics(Rule based)
perform unary operations(selection, projection)
reorder operations to reduce intermediate relation size
replace a join by a series of semi-join
Global query optimization
Find the best global execution plan
Minimize a cost function
in terms of time or size of data
Distributed join processing
decide on the use of semi-joins
Local optimization
Local optimization can be performed independently with or without central control information
Either a single site or several sites may participate in local optimization
Use semi-join to minimize the network data transmissions
Semi-join
R semi-join S(A) join S(A)
site2 sent t1=ΠS(A) to site1
site1 sent t2=R semi-join t1 to site2
site2 sent t3=t2 join S(A) to users
Semi-join is better if step1+ step2< Size(A)
Distributed transaction management
Database transaction
A logical unit of database processing
Properties of a transaction
Atomicity
A transaction is a sequence of database operations that are either performed entirely or are not performed at all
Consistency
A transaction must transform the database from one consistent state to another consistent state
Isolation
Transactions execute independently of one another. The partial effects of incomplete transactions should not be visible to other transactions
Durability
The effects of a committed transaction are permanently recorded and must not be lost because of a subsequent failure
Concurrency control
Transaction processing systems are large databases with multiple users executing database transactions
Multiple users are concurrently executing transactions
A transaction can have several data access operations, some of which could be accessing the same data item
Through multi-programming operating systems, the processes of users are interleaved
System log
Write ahead logging
must force-write the log record for an update before the corresponding data page is written
must write all log records for a transaction before commit
Redo
Recording after image of data item
Undo
Recording before image of data item
Updating distributed data
Synchronous replication
All copies of a modified fragment must be updated before the modifying transaction commits
Data distribution is made transparent to users
Voting based
A transaction must write a majority of copies of data to modify an object, must read enough copies to be sure of seeing at least on recent copy
Read-any, write-all
writes are slower and read are faster.
Cost of strategy
Before an update transaction can commit, it must obtain locks on all copies to modify
sends lock requests to remote sites, and while waiting for the response, holds on to all other locks
If sites or links fail, transaction can not commit until they are back up
Asynchoronous replication
Copies of a modified fragment are only periodically updated,different copies may get out of sync in the meantime
Allows the modifying transaction to commit before all copies have been changed
Primary site
Exactly one copy of relation is designated the primary or master copy. Replicas at other sites can not be directly updated
Peer to peer
More than one of the copies of an object can be a master in this approach
2PC Protocol
Enforcing atomic transactions