Push: Source will send data to Target. Apache Kafka, MQTT. Pull: file download or database querying. Poll: Target periodically check the Source for change, if it changes, then read again. Web crawling.

Bounded Data: discrete datasets or batches; Unbounded Data: continuous stream

Throughput, Latency and Scalability: One-off tasks 不需要考虑 latency;在 unbounded data stream 要考虑 latency;当 data volume grow 必须考虑 throughput 和 scalability

ETL: 加载前转换,适用于**中小**数据,数据仓库, Data analysts excluded from ETL processes,结构化数据

ELT: 加载后转换,适用于**大**数据,数据湖,非结构化数据

Orchestration: Manages the flow of data, automating extraction, transformation, and loading (ETL).

Data Quality: Believable(subjective), Value-added, Relevant, Accurate(objective), Interpretable

Data Cleaning: Missing(removed or imputed), Default, Incorrect(remove or adjust), Inconsistent(reconcile)

csv.reader reads rows into arrays

csv.DictReader reads rows into dictionaries

Bar Charts for Categorical/Nominal Data

Histograms for Numerical Data

Scatter Plots for comparing two continuing variables

自动生成 Schema：约束弱，没有 FK 和 Integrity Constraints; 自定义 Schema：约束强，速度慢

特性	Operational Systems	Analytical Systems
访问模式	OLTP (Online Transaction Processing)	OLAP (Online Analytical Processing)
核心任务	事务处理（Transaction）	分析与统计（Analysis & Aggregation）
操作类型	插入 / 更新 / 删除（Insert / Update / Delete）	通常只读（Read-only）
用户	外部用户（External users）	业务分析师 / 数据科学家
查询方式	查少量记录 → **Point Query**	扫描大量记录 → **Aggregate Query**（count, sum, avg, max, min, group by）
数据量	小、实时	大、历史数据全量

OLAP: Roll-up:向上汇总数据; Drill-down:小粒度;Slice: select a single dimension from a cube; Dice:create a smaller sub-cube from an OLAP cube.Pivot:改变观察角度,行列互换

Feature	Database (OLTP)	Data Warehouse (OLAP)
Main read pattern	Point queries (fetch individual records by key)	Aggregations over a large number of records
Main write pattern	**Create, update,** and **delete** individual records	Bulk import (ETL) or event stream **聚合、统计、查询**
Typical human user	End user of web/mobile application	Internal analyst (decision support)
Machine use case	Checking if an action is authorized; 用户系统、银行系统	Detecting fraud/abuse patterns; BI 和大数据分析
Data represents	Latest state of data (current point in time)	Historical events accumulated over time
Dataset size	Gigabytes to terabytes	Terabytes to petabytes

Feature	Data Lake (ELT)	Data Warehouse (ETL)
Data type	Structured, semi-structured (JSON, XML), unstructured (images, audio, logs)	Primarily Structured data (tables, SQL)
Schema	Schema-on-read (defined at query time)	Schema-on-write (defined before writing)
Storage & Cost	Distributed cheap storage (HDFS, S3), low cost	Specialized storage, high performance, high cost
Data processing	Raw data stored; flexible processing for analytics, ML, and exploration	Cleaned, transformed, indexed; optimized for analytics
Typical use cases	Big data analysis, machine learning, real-time processing	BI reports, analytics, decision support
Agility	More agile as it accepts raw data without a predefined structure	Less agile due to predefined schema

特性	Star Schema（星型模式）	Snowflake Schema（雪花模式）
结构	一个事实表 + 多个非规范化维度表	一个事实表 + 多个规范化维度表（维度拆成子维度表）
维度表特点	宽大、包含完整信息	层次拆分、减少冗余
适用场景	**BI 报表、实时看板、非技术用户查询**	**数据量极大、维度层次复杂、对存储和数据一致性要求高**
查询性能	**高，现代云数仓对宽表优化好**	**相对低，需要多次 JOIN**
维护难度	易于理解和维护	较复杂，理解和维护难度大
存储占用	**占用较多**	**节省存储空间**
设计理念	**以空间换时间 → 提升查询速度**	**以时间换空间 → 提升存储效率和数据整洁**

Data Warehouse: contains a read-only copy of data from various OLTP; Can be both source or sink; 有 Data Silo; Fact Table 通常包含度量和外键维度, 通常是数字，可以进行聚合; Dimension Table.

特性	Web Scraping	Web API
返回数据类型	HTML / 网页内容（可能还带 CSS/JS）	JSON / XML / 半结构化数据
数据结构	非结构化或半结构化，需要解析 DOM	结构化或半结构化，直接可用
稳定性	易受页面改版影响	相对稳定，只要 API 不变
速度	相对慢，需要解析网页	快，直接返回数据
面向对象	面向人类用户（页面展示）	面向程序（机器可直接处理）
依赖技术	解析 HTML / DOM / XPath / CSS Selector	HTTP 请求 + JSON/XML 解析
适用场景	无 API 或网页需要抓取	官方提供 API，可直接获取数据

Web Scraping: Reconnaissance(Check terms-of-service and robots.txt; if in doubt, contact owner); Webpage Retrieval; Data Extraction; Data Cleaning transformation; Data Storage, Analysis

URL: protocol://site/path_to_resource

Scrapy: 专业的 Python 爬虫框架

Selenium: 可编程的浏览器

HTTP:Head 部分用户看不到,Body 部分是网页内容.DOM is an element tree, Sibling tags have an order from left to right!方法有下面 2 个

请求方式	参数位置	常用场景
GET	URL（?key=value）	获取数据（搜索、下载网页）
POST	请求体（body）	提交数据（登录、上传表单）

RESTful APIs: Stateless,使用上面 2 个 HTTP 方法; XML Web Services:基于 SOAP,也是一种 Web API

Semi-structured data: Nesting, Heterogeneous collections; HTML, XML and JSON belong to it.

HTML:主要用于网页设计和展示,pre-defined tags,宽松语法,浏览器容错率高,即使写错也能渲染

XML:主要用于数据交换和存储,user-defined tags,严格语法,必须成对闭合<title></title>,区分大小写,Empty elements <AUD/>,

Well-Formed XML 文档符合 XML 的基本语法规则,Valid XML 文档不仅格式正确（well-formed），还要符合一个预定义的约束规则

DTD:定义 XML 文档的语法规则,?0 或 1,*0 或多,+至少 1 次.属性 genre 在<>里面,子元素在<>外.

<book> 元素由 title, author, price 组成。`title` 必须出现 1 次；`author+` 表示 至少一个作者；`price?` 表示 价格可选（0 或 1 次）

<!ELEMENT book (title,author+,price?)>

<book> 元素必须有一个属性 genre: 属性类型为 CDATA（字符串类型）

<!ATTLIST book genre CDATA #REQUIRED>

Feature	JSON (JavaScript Object Notation)	XML (eXtensible Markup Language)
Structure	Key–value pairs and arrays (map-like)	Tree structure with tags and attributes
Data Types	Native support (string, number, boolean, null)	Everything is string (types via schema)
Readability	High (compact and clean)	Moderate (verbose with closing tags)
Parsing	Fast (lightweight, widely supported)	Slower (DOM / SAX parsing)
Metadata	Not built-in	Supports XSD and XSLT
Size	Small	Large (tag overhead)

XPath	CSS Selectors
单个值、节点、完整子树	HTML 元素 (Element)
双向（可向上找父节点，向下找子节点）	单向（只能向下/向后找）
可以通过文字内容查找 (text)	无法通过文字内容查找
支持函数（如 count, sum, contains）	仅支持简单的逻辑伪类
XML、HTML	主要是 HTML

在 XML DOM 里面，root 不对应文档里面的元素.

保存爬虫数据方法:File systems (CSV, XML or JSON files), HTML 的核心目的是展示网页内容，不是为了存储结构化数据而设计。

Database JSONB does not keep duplicate, JSON 在 Database 里面不能有相同的 key.

Schema-First	Schema-Late /Schema-On-Read)
必须定义	可先插入、后解析
高	取决于读取时处理
低	高
关系型数据库	JSON/XML 数据库、NoSQL、数据湖

Relational Database	MongoDB
Database	Database
Table	Collection
Row (tuple, record)	Document
Column (attribute)	Field
Schema First: tuples in a relation follow the same schema	Flexible schema: documents in a collection do not need to have the same set of fields
Normalised Schema: 1NF	Denormalised data model: documents can be nested and can contain arrays
More functionality means more overhead	less functionality means less overhead

MongoDB shell 不要求双引号 around keys and string values;

MongoDB compass 要求双引号,错误{ name: 'Alice', age: 25 }

MongoDB 更容易 Scalability: Automatic Data Partitioning(把一个大集合的数据分布到多台机器上,horizontal range partitioning per 'table'); Single-Leader Replication(每个 shard 内部有多个副本 replica set 但只有一个 primary 主节点接收写入,保证数据可靠性和高可用), Load Balancing(将请求均匀分配到多个节点,负载均衡)

Graph Database: Neo4j,节点表示实体,边表示实体之间的关系,属性节点或边的元数据或属性.优势为高效关系处理,灵活性,性能.支持 ACID,高可用,使用 Cypher(a declarative graph query).

Python 虚拟环境:隔离依赖,保证每个项目可以独立运行,不受全局或其他项目的影响,python -m venv myenv

Spatial Database Management System (SDBMS):Handle a large amount of spatial data stored in secondary storage.

Geographic Information System (GIS):SDBMS Client, Information in GIS is typically organized in "layers". A map has a layer of "roads"

点数据 Point Data	区域数据 Region Data
单个位置：**离散的**	有边界的区域/形状：**连续的**
典型的栅格数据是一个巨大的像素矩阵或网格，每个像素都有一个值	折线、Polygon、MultiPolygon
Raster data	Vector data
Object-based (实体)	**Field-based (场)**
Object Viewpoint of Forest Stands Area-ID / Dominant Tree Species / Area/Boundary FS1 / Pine / [(0,2),(4,2),(4,4),(0,4)] FS2 / Fir / [(0,0),(2,0),(2,2),(0,2)] FS3 / Oak / [(2,0),(4,0),(4,2),(2,2)]	Field Viewpoint of Forest Stands $f(x,y) =$ "Pine," $2<x\le4; 2<y\le4$ "Fir," $0\le x\le2; 0\le y\le2$ "Oak," $2<x\le4; 0\le y\le2$
离散的（Discrete），适合 point data，特点是[(0,2),(4,2),(4,4),(0,4)]	连续的（Continuous），适合 region data，特点是<
矢量（Vector: Point, Line, Polygon）	栅格（Raster: Pixels / Cells）

SRID	描述
Cartesian	在平面上以定义的原点沿轴的点位置（二维或三维）
Geographic	(Geodetic)地理角度坐标：经度、纬度 longitude and latitude
Geocentric	三维直角坐标，地球中心为原点 three-dimensional Cartesian coordinates that model the earth as a three-dimensional object.
Projected	将地理坐标投影到平面上（UTM、墨卡托等）

对于 Geodetic 和 Geocentric 来说，北极点是不能一对一映射的，因为 geodetic 北极点有多个，比如(90°, 0°)(90°, 30°)(90°, 120°)

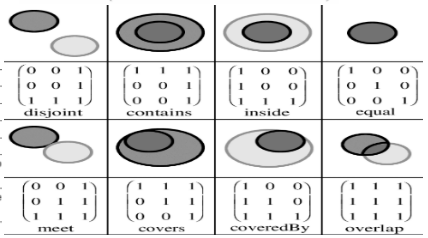
Mercator Projection:靠近北极的会被放大，introduces distortion (area distortion increases with latitude), preserves angles

世界通用 WGS84 == 4326 澳洲 GDA94

PostGIS: PostgreSQL 的空间数据库扩展,支持地理空间对象.R-Tree 索引基于 GiST 支持下面 2 种类型.

Geography Type	Geometry Type
球面/地球模型（经纬度）	平面坐标（二维笛卡尔平面）
两点之间的最短路径是大圆弧（circle arc）	**两点之间的最短路径是直线**
支持的函数较少，如 ST_Intersects(), ST_Area(), ST_Distance(), ST_Perimeter(), ST_CoveredBy()	PostGIS 提供几乎所有空间函数
较高，计算更复杂	较低
大范围或全球级别地理数据	小范围或局部空间分析
同上	可以随时使用 ::geometry 或 ::geography 转换
GiST 索引	GiST 索引

$$\Gamma_9(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$



GeoSeries:属于 Geopanda, GeoSeries 是一个"一维的空间对象集合"，里面的每个元素都是一个几何对象（Point / Line / Polygon），并且共享同一个坐标参考系（CRS）。

Point / Line / Polygon → 单个空间对象

GeoSeries → 一列空间对象

GeoDataFrame → 表（多列 + 一列 geometry）任一时刻只有一个 Geometry 列能被激活

Attribute Join	Spatial Join
字段值	空间关系
merge()	sjoin()
geometry 不参与	geometry 必须参与
SQL JOIN	GIS 专属
补名字、代码	点在哪个区

gs = gpd.GeoSeries([Point(-120,45), Point(-121.2,46), Point(-122.9,47.5)])

KML:OGC 标准的一种 XML 空间文件格式

Semi-Temporal Database:只记录有效时间（valid-time）的一部分，从某个时间开始到现在 [since, now)

Instant | Point: something happening at an instant of time; DATE ,TIME,TIMESTAMP(DATA+TIME)

Interval: a length of time; a duration 比如 3 天

Period: an anchored duration of time (a recurring interval of time),比如 2024 年.half-closed interval [start, end),开始和结束的 bound 必须相同类型.在 PostgreSQL 里面叫做 DATERANGE，也符合 half-closed。此外对于这种类型的列，我们要用 GiST 作为 index.

SQL supports time instants, intervals not support periods.

User-defined time: an uninterpreted time value. 用户自定义时间,数据库不解释其意义,不区分过去/现在/未来.

Valid/Fact time: application time. when a fact was true in the modelled reality.有效时间,事实在现实中成立的时间.可以向前修改,也可以向后修改.

Transaction time: system time, Database time. when a fact was stored in the database,事务时间,事实被记录入数据库的时间.只能向后修改.

Current: 只关心现在的状态,不关心历史变化. How many products do we currently have in stock?

Sequenced: at a specific instant or period of time,查询数据的历史变化序列,要关注顺序,Give the sequence of how many product were in stock.

Nonsequenced: ignoring time 查询过去曾经出现过的数据,关注数据的存在性,但不关注顺序. How many products A did we have at any time in stock?

transaction-time column 是 period type，即有开始和结束

valid-time column 是 period type，即有开始和结束

transaction-time table: A table with a transaction-time column

valid-time table: A table with a valid-time column

bitemporal table: A table with both a transaction-time and a valid-time column

nontemporal table: A table with neither a transaction-time nor a valid-time column

时间数据库如何表示"当前仍然有效?

- Minimum possible timestamp: 把 end_time 填成最小可能值, quite counter-intuitive
- NULL value: 把 end_time 填成空表示数据仍然有效,Use of NULL as 'unknown' is no longer possible
- Maximum possible timestamp: 给 end_time 设置最大值
- Point representation: Set start_time = end_time; 丢失了区间的含义

Timeseries Data

- Point-based representation: Multiple rows with atomic data types,用 B-Tree 作为 INDEX
- Sequence-based representation: Single row with array of time point data. Requires array datatype.用 GIN 作为 INDEX. Unnesting 把数组拆成多行; array_agg() 把多行合并成数组
- Dedicated time-series database: TimescaleDB, InfluxDB

Text: usually does not have a pre-defined data model and is unstructured.

Image: can be described as vector graphics or raster data.RGB 的 channel 就是 3 个.Number of bits per pixel (bpp) defines how many colors can be represented. 8 bpp=2^8=256 colors.

Gray-scale: Single channel, bpp defines how many shades of gray can be represented.

Binary image: Each pixel is just black (0) or white (1). Single channel.can be 1 bpp but also works with higher bpp. Referred as 'mask' in the image processing domain.

Supervised learning: Prediction, Classification, Regression.

Unsupervised learning: Clustering, Probability distribution estimation, Dimension reduction.

Unstructured Data Analysis Process

Data Acquisition→ Preprocessing→ Feature Extraction → Task

Feature Extraction: Unstructured data to vectors.

Word Embeddings: Using models like Word2Vec (word embeddings), GloVe, or BERT (contextual word embeddings) to represent text data in a dense vector format. Document Similarity 可以通过 Vector Distance 判断，比如 Cosine-Similarity 1 表示文档非常相似,0 表示文档无关,-1 表示文档内容相反.

TF: 词语在单篇文档中的出现频率；词出现次数/文档总词数

用 CountVectorizer 进行计算，可配置参数如下

ngram_range=(2,4)表示取 2-gram, 3-gram, 4-gram

binary=True 表示不再统计单词出现的次数，只记录该词是否出现过。

stop_words=['the','a'] 表示在构建词频矩阵时，手动排除列表中的高频但无实际含义的词（如 "the", "a", "is"）

min_df=0.1 表示忽略在少于 10% 的文档中出现的极罕见词

max_df=0.5 表示忽略在超过 50% 的文档中出现的极常见词

IDF：衡量词语在整个语料库中的稀有程度；log(总文档数/包含该词的文档数)

TF-IDF：TF 与 IDF 的乘积，值越大代表词语对文档的区分能力越强;.

BERT: pre-trained transformer-based neural network, uses a bi-directional approach considering both the left and right context of words in a sentence, returns vectors.

BERT	GPT
双向（Bidirectional）通过左右文预测被 Mask 掉的词	单向（Unidirectional）根据前文预测下一个词
Masked Language Model (MLM) Next Sentence Prediction (NSP)	Next Word Prediction (语言建模 LM)
Understanding and Analyzing text	Generating
文本分类、命名实体识别（NER）、情感分析、问答（QA）	文本生成、聊天、摘要、创作内容等
Labeled data	擅长少样本（few-shot）或零样本（zero-shot）学习，可快速泛化到新任务

TF-IDF (Term Frequency-Inverse Document Frequency)	BERT (Bidirectional Encoder Representations from Transformers)
Statistical (word frequency/importance)	Neural/Contextual (transformer-based)
No context. Treats "bank" (river) and "bank" (finance) the same	**Deep context.** Dynamically adjusts word meaning based on neighbors
Sparse. High-dimension, mostly zeros	**Dense.** Low-dimension, mathematically rich
Negligible; runs instantly on CPUs	Heavy; typically requires GPU for inference
Best for exact keyword matching	Superior for sentiment, tone, and paraphrasing

ROI = 图像中你特别关注的区

Image descriptors (white box algorithms) typically based on image gradient 主要依赖于 gradient.使用手工设计的算法提取图像特征

Neural networks (black box algorithms): 通过神经网络 NN 自动学习图像特征.

对比项	Image Descriptors（White-box）	Neural Networks（Black-box）
基本思路	手工设计特征 gradient	自动学习特征 NN
可解释性	高（过程透明）	低（内部不可见）
特征来源	图像梯度、颜色变化等	learning from data
常见特征	边缘、角点、纹理	High-level semantic features
典型方法	HOG、SIFT、边缘检测	CNN、深度神经网络
算法复杂度	较低	较高
数据需求	较少	较多
Python 支持	scikit-image（skimage）	TensorFlow / PyTorch

Metadata:

- GUI tools such as Photoshop, Lightroom, Preview
- Typically includes: GPS location, author, copyright
- in the form of key-value pairs
- 元数据可以被修改

命令	作用
exiftool **filename**	查看文件的 **所有元数据**
exiftool **-CreateDate filename**	查看文件中特定字段（如 **CreateDate**）
exiftool **"-GPS*" filename**	查看文件中 **所有 GPS 相关元数据**
exiftool **-common -csv=metadata.csv** *.jpg	将多张图片的 常用 EXIF 信息 导出为 **CSV 文件**
exiftool **-common -json=metadata.json *.jpg**	将多张图片的常用 EXIF 信息导出为 **JSON 文件**

Stream Processing: 处理正在流动的数据 data in motion 而不是先存储后处理,比如 Fraud detection, Cybersecurity

传统数据分析: Not agile enough, Queries are specified over stored data, data at rest: favors infrequent updates

Data Streams: unbounded.

Transactional data streams: log interactions between entities：记录实体之间发生的"事件 / 交互"的数据流.比如 credit card purchases by consumers from merchants
Measurement data streams: monitor evolution of entity states：持续监测某个实体"状态随时间变化"的数据流. IP network

Database Systems (DBMS)	Data Stream Systems (DSMS)
Persistent relations	Transient streams
set/bag of tuples	Sequence of tuples 有序
Bounded by stored data.静态	Unbound data;动态
modifications	appends
transient / one-time	continuous / persistent
exact	approximate
由 查询处理器和物理数据库设计 决定	不可预测，因为数据特性和到达模式不确定

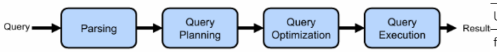
第一种 DSMS: Publish/Subscribe Messaging System ability to consume constant stream of events. with QoS guarantees. 布者不直接把消息发给具体接收者，而是按"主题（topic）"发布到 Broker：订阅者只从 broker 接收自己订阅的主题的消息。常见的有：Apache Kafka(Topics in Kafka are multi-subscribe)，MQTT. A broker mediates communication between producers and consumers, providing decoupling, scalability, and reliable message delivery in distributed systems

能力	解释
1. Publish and subscribe to streams of records	支持应用程序 发布消息到主题（topic），以及 订阅自己感兴趣的消息流、实现实时数据分发。
2. Can store streams of records in a fault-tolerant way	以容错性持久化消息，保证在系统故障时消息不丢失，典型实现：消息在 Broker 中持久化（如 Kafka 的 log 机制）。
3. Lets apps process streams of records as they occur	应用可以实时处理到达的数据流，无需等待数据全部存储完再分析，支持事件驱动或流式处理。

MQTT: 第一个是 subscriber，第二个是 publisher
subscribe.callback(print_msg, "MyTopic", hostname=broker)
print_msg：回调函数 = 系统帮你调用的函数，不是你手动调用
MyTopic：订阅 MyTopic 这个主题，当有新消息到来时，用 print_msg 这个函数来处理消息。Topics are case-sensitive
hostname=broker：是从 broker 订阅，而不是 Publisher
publish.single("MyTopic", "TheMessage", hostname=broker)
By default, MQTT broker does not store any message.只有在线的订阅者才能收到消息,但是可以设置为保留

Apache Kafka (Distributed Streaming Platform)	MQTT (IoT Messaging Protocol)
For large-scale stream data storage and real-time processing	For resource-constrained devices with low-bandwidth, real-time communication
Distributed commit log (append-only log); 保证 topic 不会丢失	Broker-based message forwarding model
Persistent by default: data stored on disk, replicated for fault tolerance, historical data can be replayed	Transient by default: messages usually deleted after forwarding (optional retained messages or offline cache)
Pull-based: consumers fetch messages from the queue in order, control their own offset	Push-based: broker actively pushes messages to subscribers for real-time delivery
Strictly guaranteed order within a partition	Depends on network and QoS level; global ordering hard to guarantee at scale
Very high: data replicated across multiple nodes, single-node failures do not affect safety	Lower: relies on broker stability, not focused on long-term persistence
大规模、可靠、持久化、企业级流处理；金融交易风控、网站日志分析	轻量级、低带宽、实时、IoT/事件推送；智能家居设备、传感器数据采集

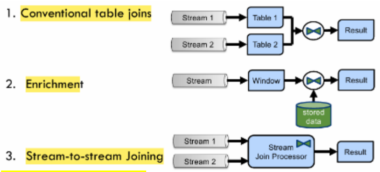
Quality of Service
-At most once (QoS 0：Default): every message is processed at most once, but no guarantee：尽力而为，发了就不管了，最快
-At least once (QoS 1): no message will get ignored：一定送到，但可能送两次
-Exactly once (QoS 2): system guarantees that every message is processed exactly once：一定送到，而且只送一次，最慢
第二种 DSMS: Data Stream Processors
ability to execute query processing on stream data with clear semantics.
Data Stream Query Processing



State refers to data that is retained and updated across multiple events during stream processing.
Stateless operators look at each event individually
Stateful operators output results based on multiple events

Stateful Applications	Stateless Applications
Retain user session information across requests	Do not retain user state between requests
Less scalable; require complex load balancing and session management	Highly scalable; each request is independent
Lower Fault Tolerance; server failure may cause session loss unless replication is used	Higher Fault Tolerance; server failure does not affect sessions
More Resource Need; require memory and processing for session handling	Less; no session data to manage
More complex; need careful session and state management	Simpler; no need to manage state across requests

Aggregation – need first to define a window on the data stream to process.
Agglomerative Window：incrementally aggregates incoming data and updates the result continuously without storing all events in the window.
Sliding Window: fixed size, moves forward at a fixed interval, overlap.
Tumbling Window: fixed-size, non-overlapping, 是特殊的 Sliding Window.
Tuple-Count-Based Windows: 包含 Sliding Window 和 Tumbling Window。基于 tuple 的数量进行 window 的划分. 有个问题就是 Tie，即当假设有 3 条数据时间戳都是 10:00:01，但窗口大小刚好在它们中间切断。这时候这次可能把 A 放入窗口，下次可能把 B 放入窗口.解决办法为在时间戳之外，增加一个唯一且稳定的字段 tie-breaker.
Punctuation-Based Windows：基于 punctuation 划分窗口，窗口长度不固定.
Event time：事实真正发生的时间。有个 Watermarks 的机制可以确保这个，它是一个时间戳 T，代表系统声明："所有 Event Time≤T 的数据都已经到齐了。因此可以通过窗口的 last timestamp ≤ watermark 来判断是否输出结果。
Ingestion time：数据进入系统边界的时刻。
Processing time：数据被算法/算子执行的时刻。



Conventional Table Joins 将两个已经存在的、完整的静态表（Table A 和 Table B）通过共有键（Join Key）合并。
Enrichment 将实时的"事实流"与相对静态的数据通过一个 window 进行匹配。这个 window 确保了即使维度表在不断变化（Stored Data updates），流数据也能找到它"应有"的历史镜像。因为查询数据库会导致延迟，有 2 种办法解决 In-memory Storage：通过把常用的存储在内存提升效率。External data Storage：通过 index 等提升查询效率
Stream-to-stream Joining 将两个实时产生的数据流，在某个时间窗口内进行匹配。
数据流处理工具 Recall：Kafka or MQTT are not data stream processors.

Apache Spark	Apache Flink
Set-oriented data; Lazy evaluation principle: Plan gets actually executed by Flink only when env.execute() is called	Transformations by iterating over collections with pipelining
Processing at Separate stages	Processing at Overlapping stages
Optimizer is SparkSQL	Optimizer is API
RDD	DataSet
Micro-batching (DStream)	Continuous pipelining (DataStream API)
Higher Latency	Low Latency
Use Case: Micro-batch	Use Case: pipelining
fixed, sliding, tumbling	fixed, sliding, tumbling, Agglomerative

Scale-Up: To scale with increasing load, buy more powerful, larger hardware
Scale-Out: need to scale-out to a cluster of multiple servers (nodes). shared-nothing architecture

Speed-Up	Scale-Up
数据量不变，资源增加，运行时间应按比例减少	数据量和资源同时按比例增加，运行时间应保持不变
固定	随资源增加而增加
增加计算资源（CPU/节点/内存）	增加计算资源与数据量成比例
减少处理时间	响应时间不变
时间按比例减少，实际收 overhead 影响	响应时间保持恒定

Scale-Agnostic Data Management 与规模无关的数据管理
数据分片（Sharding）：提升性能
数据复制（Replication）：保证可用性
应用透明：应用无需关心底层复杂性
Single Machine：比如 RAID 因为显示为一个逻辑磁盘.
Row-oriented databases organize data by record (row).Optimized for reading and writing on rows efficiently 适合 OLTP
Column-oriented databases organize data by field (column). Optimized for reading and computing on columns efficiently. 适合 OLAP.优点:不必读取不需要的属性, Better compression possibility,都是减少 IO. 缺点:不适合频繁更改，读取多行时的开销大，不适合 small table.
Cluster of Machines
Data Partitioning / Data Sharding: Storing subsets of the original data set at different places. Sharding：if each partition is stored on a different site. 分布 shard 方法如下: Round-robin：in rounds. Hash partitioning：hash function Range partitioning：range predicate；比如[0:1]放一起，[2:5]放一起. Inter-Query Parallelism：different Query independently on separate places Intra-Query Parallelism：same Query accesses several places in parallel
Data Replication: Storing copies ('replicas') of the same data at more than one place；适合 READ-ONLY. 主要是为了 Safety 和 availability. Lazy+Pri 最常见；Eager+Mult 是 IDEA
Synchronous (Eager) Replication: update all replicas inside original transaction
Asynchronous (Lazy) Replication: Update one copy，再扩散到别的
Primary Copy：永远先更新一个 replica
Multi Leader：可以同时更新多个不同的 replica；Eager 导致死锁；Lazy 导致 Conflict；适合在每个 transaction 负责 disjoint fragment 的情况
CAP Theorem: Consistency：All copies have the same data. Availability：A data system should always be up. Partition Tolerance: Ensures the system functions despite communication breakdowns. CP：适合 financial application. AP：适合 social media platform. CA：只存在于"没有网络分区"的系统中
Scale-Agnostic Data Processing 与规模无关的数据处理
大规模并行处理：跨数百或上千 CPU。
性能：并行查询/计算处理。
可用性：系统理想情况下永远在线，可透明处理故障。
弹性（Elasticity）：运行中可动态调整规模。
Data Processing: Spark, Hive, HBASE, Hadoop, MapReduce, ApacheFlink
Storage:HDFS,MongoDB,snowflake,AmazonS3
HDFS：Allow clients to access files on remote servers "transparently". One NameNode, Multiple DataNodes. Each block is replicated across multiple nodes default 3.
MapReduce 是实现这些目标的经典处理模型, Map: filtering and sorting sorting students by first name into queues. Reduce: summary operation counting the number of students in each queue. Hadoop= HDFS+YARN+MapReduce
Loosely-Coupled System: Separation of computer and storage nodes. The more nodes, the higher the probability of some failure, Latency can increase.
Data Architecture: Architecture first, technology second.
Operational Architecture: What needs to be done?
Technical Architecture: How data is ingested, stored, transformed, and served.
Batch-driven: **延时处理**,大数据架构的一种.
Data Lake: Dump all of your data, structured and unstructured, into a central location.容易导致 data swamp，缺乏治理、元数据和质量控制，数据湖变得混乱、难以理解、几乎无法使用的状态。
Feature Store Architecture: data pipeline server for training and serving machine learning models. Transformation, Storage, Serving, Monitoring, Feature Registry. Offline Serving: Access historical data; Online Serving: Provide fresh feature. Batch Transform 静态数据 Streaming Transform 动态数据

On-Demand Transform 只在预测的时候用. 替代方案: Custom ETL Pipelines：High flexibility；Increased complexity, lacks consistency. Data Warehouses & Lakes: Easy Integrated; Not suit real time ML. In-House Feature Management Systems: Tailored;扩展性差.
Lambda Architecture: Address latency by creating batch/cold layer 存入 serving layer 确保了准确性 and speed/hot layer 直接被 analytics 使用确保低 latency. 用户也可以通过 serving layer 结合两者。
Event-driven: **即时处理**,大数据架构的一种.Data Stream Processing Architectures. 例子 Kappa Architecture, Only one path.

维度	批次驱动 (Batch-driven)	事件驱动 (Event-driven)
设计哲学	效率优先，追求整体吞吐量	速度优先，追求极致响应时效
数据视图	静态（针对已保存的数据块）	动态（针对不断流动的事件流）
典型代表	ETL 报表、工资发放、财务对账	支付预警、自动驾驶、即时消息推送
系统解耦	弱（通常依赖中心化数据库）	强（通过消息中间件如 Kafka 实现异步）
复杂度	较低，错误容易重跑	较高，需处理事件乱序和补偿机制

Continuous Integration (CI): Automatically testing and integrating new code into a shared repository
Continuous Deployment (CD): Automatically deploying tested code to production environments.
DataOps is a set of collaborative data management practices designed to speed up data delivery, maintain quality, and foster collaboration.
-Key goal: Break Silo by Unifying Data, Collaborative Framework
-Key Functions:
Pipeline Orchestration：负责统一调度和管理数据与 ML 流水线
Data Quality Monitoring：持续监控数据和特征的完整性、准确性与一致性，及时发现缺失值、异常分布
Governance and Security：确保数据使用符合组织规范与安全要求
Self-Service Data Access：通过标准化接口、目录和文档，使数据科学家和业务人员能够自助发现、理解并使用数据与特征，减少对数据工程团队的依赖
-Lifecycle: Plan, Dev, Integrate, Test, Deploy, Monitor
-Data Curation: Automate data cleansing, transformation, and standardization to ensure high-quality data.
-Master Data Management：确保数据一致性。
-The Five Pillars：Freshness, Distribution(acceptable ranges),Volume(Monitor Missing), Schema(Track structural changes), Lineage(data move/transform across system)
3 种 Scaling ML 的方法如下
Feature Store: TFIDF 是一种 feature extraction 方法.
Data to ML/Data to Computation：适合小数据，不好扩展.Scale-up
ML to Data/Computation to Data：适合大数据，好扩展。典型代表是 **MADlib**(In-database analytic)，它还可以通 ARIMA 支持 time series(只有 point) 使用 Greenplum 则是 a shared-nothing database using postgres per each node 或 PostgreSQL. Scale-out
PII: Information that, when used alone or with other relevant data, can identify an individual.
Sensitive information: is personal information that includes information or an opinion about an individual. 比如信仰，性取向。在 PII 里的优先级最高。
Data Minimalism: best way. Avoid collecting unless necessary. 不需要的历史数据要删除.
Least Privilege: grant human/machine only the necessary access.视图可以达成

维度	静态加密（At Rest）	传输加密（In Transit / TLS）
数据状态	静止（存在磁盘中）	流动（在网络中传输）
主要威胁	硬盘被窃、数据库被攻破、管理员越权访问	嗅探、中间人攻击（MITM）、非法劫持
核心技术	AES-256、RSA、KMS、TDE	TLS 1.3、SSH、IPsec
性能损耗	主要发生在数据写入 / 读取时的磁盘 I/O	主要发生在连接建立（握手）时的 CPU 开销
主要挑战	密钥的安全保管与定期轮转	证书颁发机构（CA）信任链的安全性

3 – 2 – 1 rule: At least 3 copies On 2 different media At least 1 off-premise
RPO: how much data you can afford to lose; RTO: how fast you need to recover.
-An RPO of 1 hour means data backups must be made at least every hour
-An RTO of 2 hours means the system must be fully restored within that time.
On-Premise Backup: Local storage devices like NAS.
Cloud backups: may take longer to restore than local backups.
Best Practice: Combining local replication with remote cloud backups ensures both high availability and disaster recovery readiness.

Crash Recovery：Using log；Disaster Recovery：using log and backup.
Choosing Technologies across the Data Engineering Lifecycle by using MOBILIST: Monolith/Modular, Optimizing cost, Build vs buy, Interoperability 互操作性 API, Location 本地/云, Immutability, Speed to market, Team size

Monolith	Modular
自包含系统	解耦系统，采用最佳技术，通过 API 通信，各组件各司其职
简单，一切在同一处（更少移动部件，减少上下文切换）	组件可替换，技术更新/升级灵活（可随技术变化更换工具）
脆弱，迁移困难	系统数量众多，维护复杂，更新/发布耗时且可能麻烦
UI、业务层、数据接口全部在同一系统	各层可独立拆分，模块化实现

	Tool	Purpose	Execution	Scope
Distributed Storage & Processing	Apache Hadoop	Distributed storage & batch processing	Runs MapReduce jobs across a cluster	Large-scale storage (HDFS) & processing
	Apache Spark	Unified analytics engine for large-scale data	In-memory processing for both batch and streaming data	Fast data processing, ML, and analytics
Streaming & Real-Time Processing	Apache Flink	Stream and batch data processing	Low-latency processing for real-time data streams	Stream-first architecture with batch mode
	Apache Kafka	Distributed streaming platform	Real-time event streaming & message brokering	High-throughput distributed event streaming
Data Pipelines	Apache Beam	Unified model for batch and streaming pipelines	Executes on multiple runners (e.g., Flink, Spark, Dataflow)	Flexible pipelines for stream/ batch data
Orchestration & Workflow	Apache Airflow	Workflow orchestration and scheduling	Executes DAGs across workers to manage task dependencies	Scheduling & monitoring of data workflows
Dataflow Automation	Apache Nifi	Dataflow automation and management	Automates data movement between systems	Low-latency data integration
NoSQL Databases	Apache HBase	Distributed NoSQL database (columnar storage)	Provides real-time reads/writes for big data	Random access to large datasets