Guidelines for Designing New Data Architectures

Rick F. van der Lans
Industry analyst

Rick F. van der Lans

Rick F. van der Lans is a highly-respected independent analyst, consultant, author, and internationally acclaimed lecturer specializing in data warehousing, business intelligence, big data, and database technology. He is managing director of R20/Consultancy BV.

He has presented countless seminars, webinars, and keynotes at industry-leading conferences. Rick helps clients worldwide to design their data warehouse, big data, and business intelligence architectures and solutions and assists them with selecting the right products. He has been influential in introducing the new logical data warehouse architecture worldwide which helps organizations to develop more agile business intelligence systems.

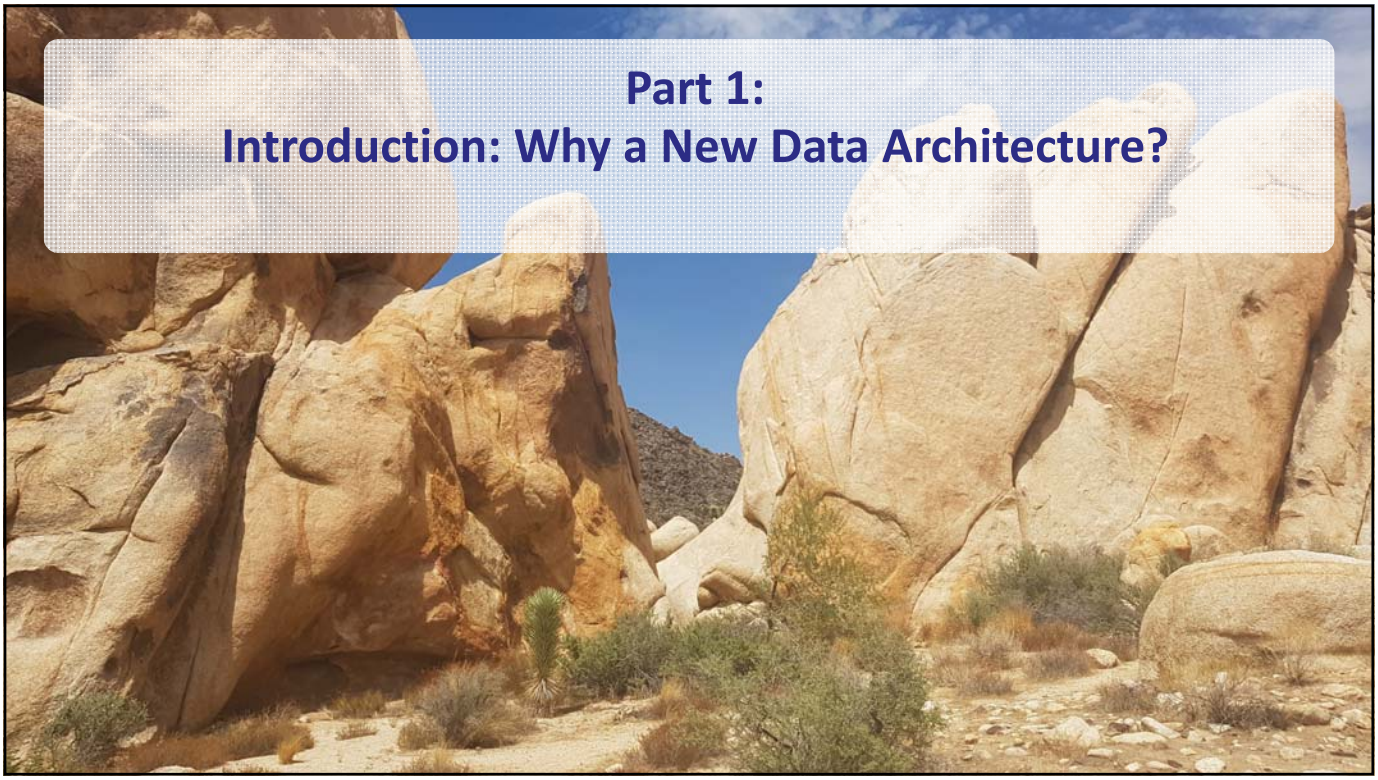
He is the author of several books on computing, including his *Data Virtualization: Selected Writings* and *Data Virtualization for Business Intelligence Systems*. Some of these books are available in different languages. Books such as the popular *Introduction to SQL* is available in English, Dutch, Italian, Chinese, and German and is sold world wide. He also authored numerous whitepapers for vendors.

In 2018 he was selected the sixth most influential BI analyst worldwide by onalytica.com.

You can get in touch with Rick van der Lans via:
Email: rick@r20.nl
Website: www.r20.nl
LinkedIn: http://www.linkedin.com/pub/rick-van-der-lans

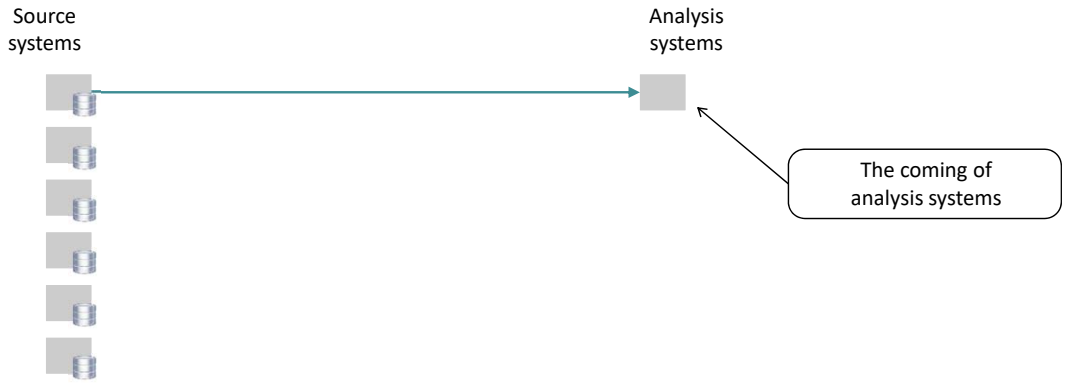
Agenda and Subjects

1. Introduction: Why a New Data Architecture?
2. Terminology and Concepts
3. Introduction to Data Architectures
4. New Technologies for Data Storage, Processing, and Analytics
5. Steps 1-3: Setting the Stage
6. Step 4: Architectural Design Principles
7. Step 5: Reference Data Architectures
8. Step 6-7: Designing New Data Architectures
9. Closing Remarks



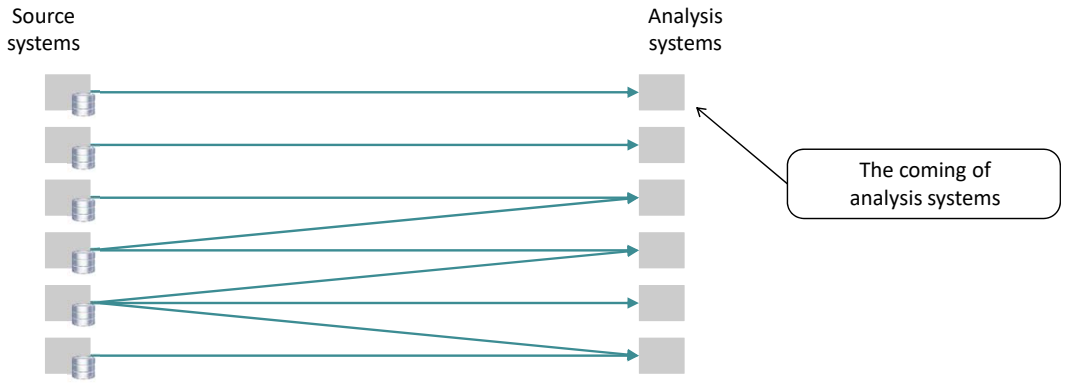
Evolution of the Data Processing Landscape (1)

Evolution of the Data Processing Landscape (2)

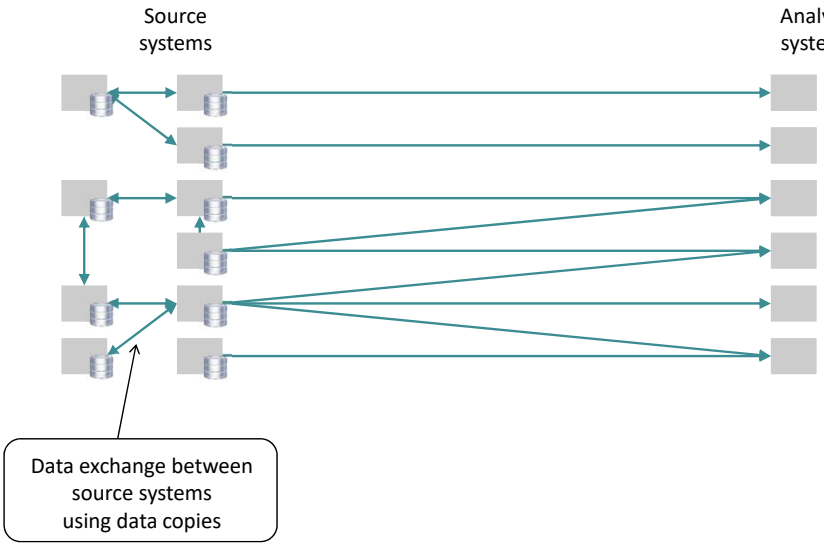


Data for the Happy Few Only

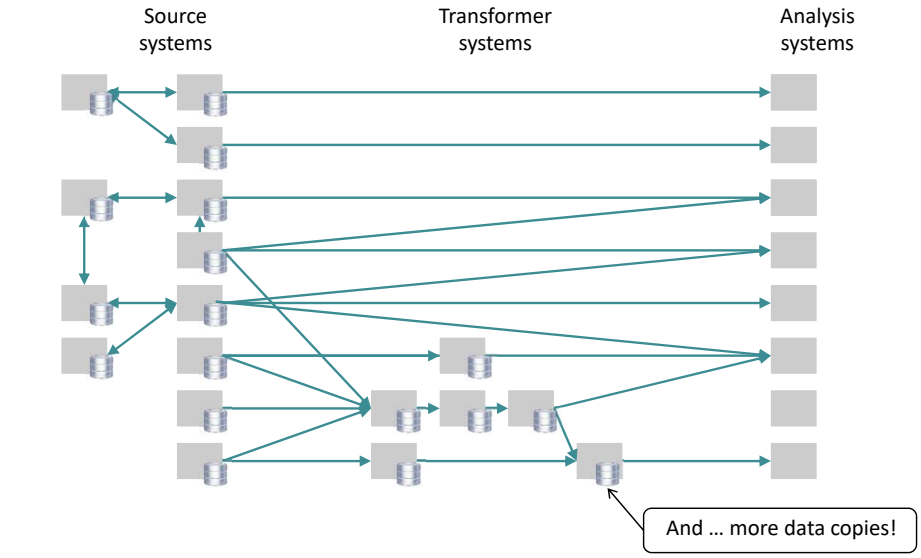
Evolution of the Data Processing Landscape (2)



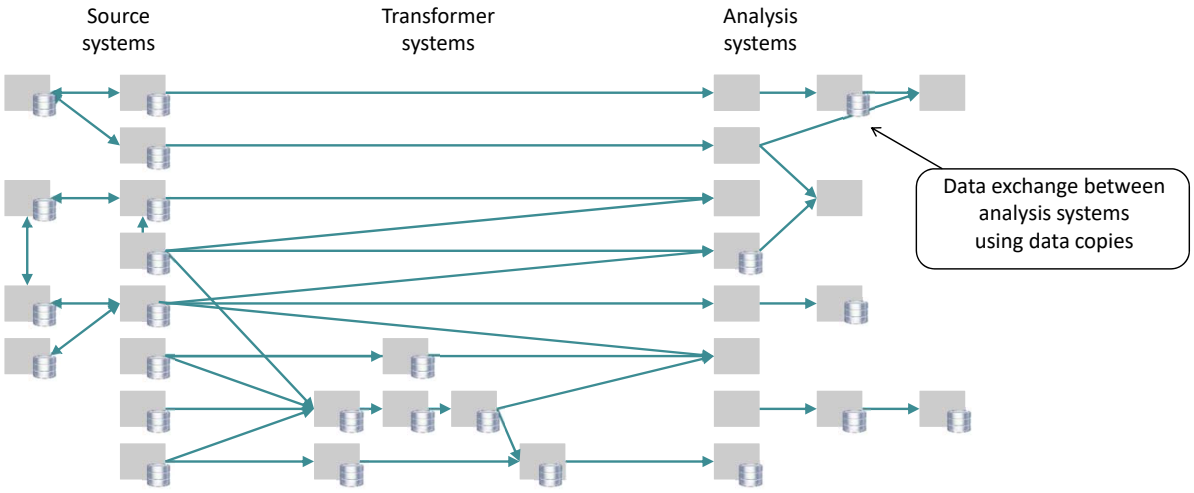
Evolution of the Data Processing Landscape (3)



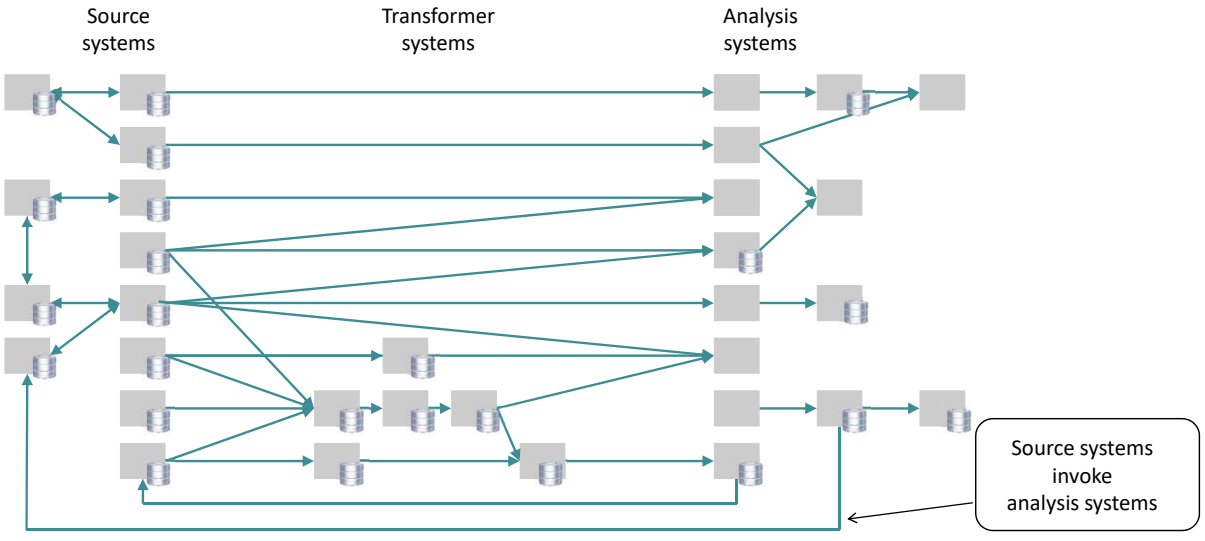
Evolution of the Data Processing Landscape (4)



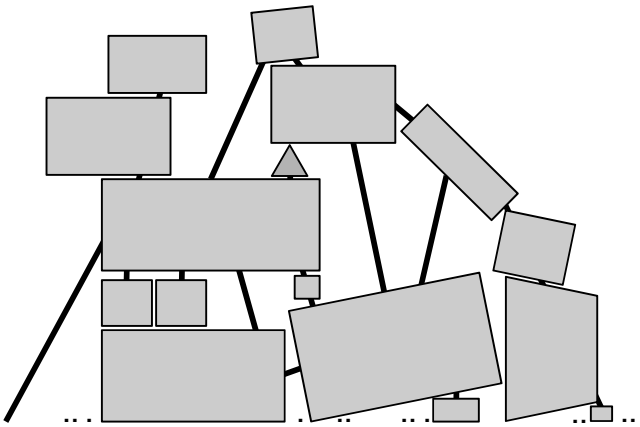
Evolution of the Data Processing Landscape (5)



Evolution of the Data Processing Landscape (6)



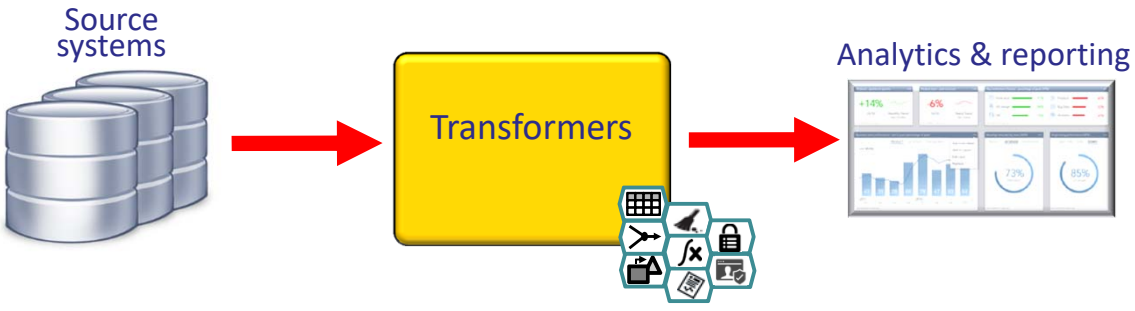
“Wobbly” Foundation of the Data Processing Landscape



Raising the Bar for ICT systems

Part 2:
Terminology and Concepts

The Transformers



- Data structure
- Integration
- Transformation
- Data security

- Data cleansing
- Analytical
- Visualization
- Data privacy

Examples of Transformers

- Data value transformations
- Data structure transformations
- Aggregations
- Filters
- Groupings
- Calculations
- Integrations
- Technical corrections
- Functional corrections
- Anonymizations
- Historizations
- …

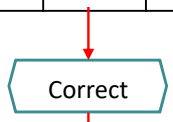
- Summarize
- Keywords extraction
- Translate
- Named Entity Recognition (NER)
- Optical Character Recognition (OCR)
- Text-to-speech
- Speech-to-text
- Data tagging
- Sentiment analysis
- Pattern recognition
- …

- Clustering
- Normalization
- Interpolation
- Extrapolation
- Binning
- Sampling
- Regression
- …

- Vector-to-Raster Conversion
- Containment
- Clipping
- Spatial intersection
- Buffering
- …

Transformer for Correcting Data

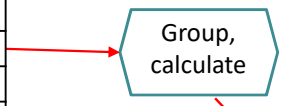
Custid	Lastname	Gender	City	Zipcode
12345	Jansen	Man	Rotterdam, ZH	2651 BJ
23324	Visser	V	Delft, ZH	2289 BG
57657	Dekker	Vrouw	Zwolle, OV	4011 AE
65461	Brouwer	M	Breda, NB	4811 AD



Custid	Lastname	Gender	City	Zipcode
12345	Jansen	M	Rotterdam, ZH	2651 BJ
23324	Visser	V	Delft, ZH	2289 BG
57657	Dekker	V	Zwolle, OV	
65461	Brouwer	M	Breda, NB	4811 AD

Transformer for Grouping Data

Custid	Transactionid	Datetime	Productid	Quantity	Price
12345	13463	202603121145	P34	3	3,50
12345	13463	202603121145	P66	2	2,75
12345	13463	202603121145	P87	1	1,95
12345	29365	202603130927	P66	2	2,75
57657	37721	202603131650	P12	1	4,00
57657	37721	202603131650	P66	2	2,75
65461	43846	202603141012	P73	4	0,75
65461	57290	202603151430	P19	5	2,40
65461	57290	202603151430	P66	3	2,75

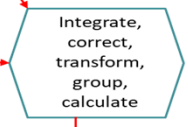


Transactionid	Datetime	Total amount
13463	202603121145	17,95
29365	202603130927	5,50
37721	202603131650	9,50
43846	202603141012	3,00
57290	202603151430	20,25

Transformer for Integrating and Grouping Data

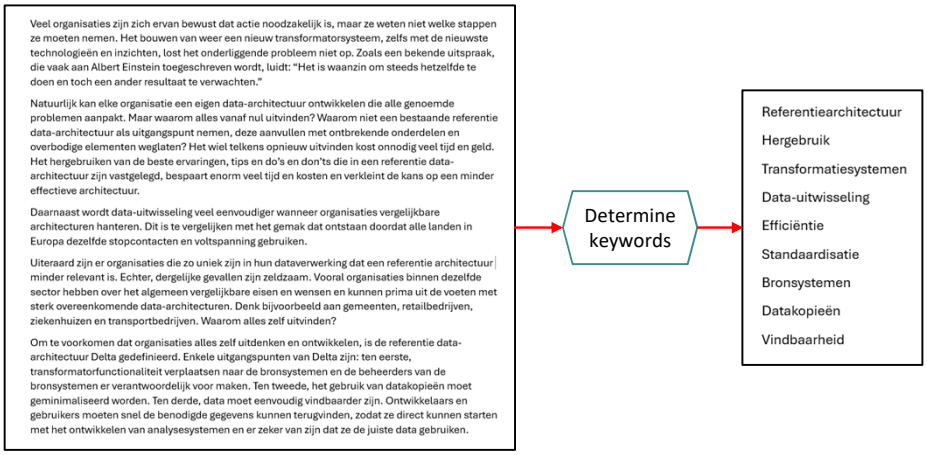
Custid	Lastname	Gender	City	Zipcode
12345	Jansen	Man	Rotterdam, ZH	2651 BJ
23324	Visser	V	Delft, ZH	2289 BG
57657	Dekker	Vrouw	Zwolle, OV	4011 AE
65461	Brouwer	M	Breda, NB	4811 AD

Custid	Transactionid	Datetime	Productid	Quantity	Price
12345	13463	202603121145	P34	3	3,50
12345	13463	202603121145	P66	2	2,75
12345	13463	202603121145	P87	1	1,95
12345	29365	202603130927	P66	2	2,75
57657	37721	202603131650	P12	1	4,00
57657	37721	202603131650	P66	2	2,75
65461	43846	202603141012	P73	4	0,75
65461	57290	202603151430	P19	5	2,40
65461	57290	202603151430	P66	3	2,75

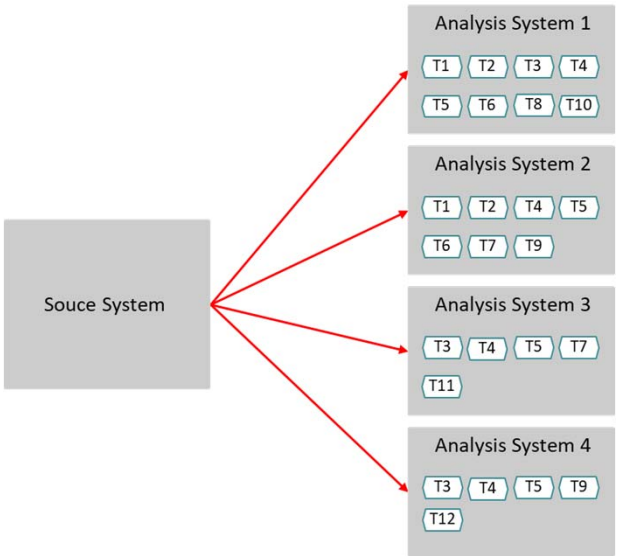


Custid	Lastname	Gender	City	Province	Zipcode	Number of trans	Total
12345	Jansen	M	Rotterdam	Zuid-Holland	2651 BJ	2	23,45
23324	Visser	V	Delft	Zuid-Holland	2289 BG	0	0,00
57657	Dekker	V	Zwolle	Overijssel	4011 AE	1	9,50
65461	Brouwer	M	Breda	Noord-Brabant	4811 AD	2	23,25

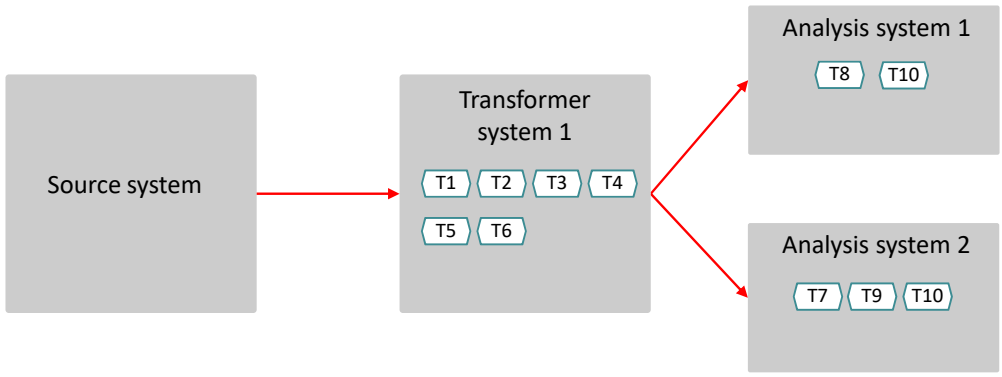
Transformer for Determining Keywords



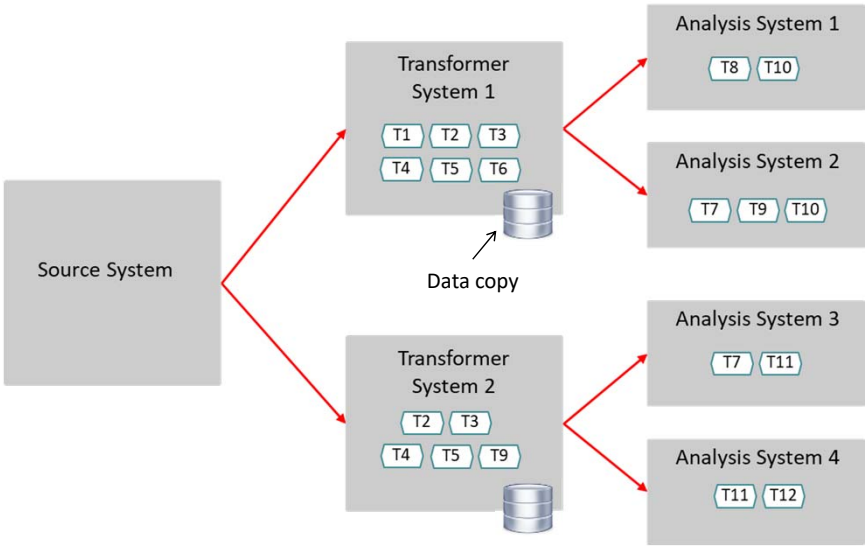
Duplicate Transformers in Analysis Systems



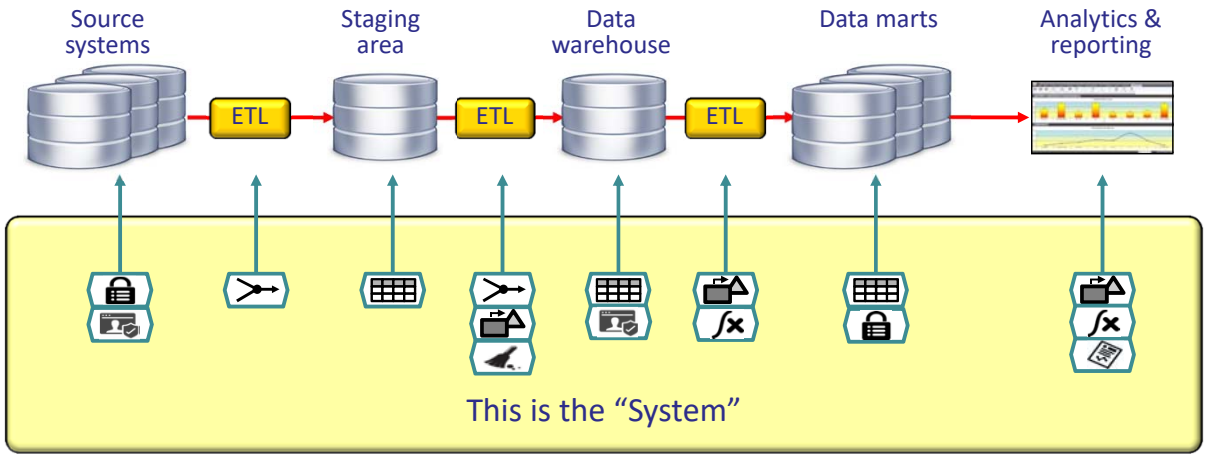
Centralizing Transformers



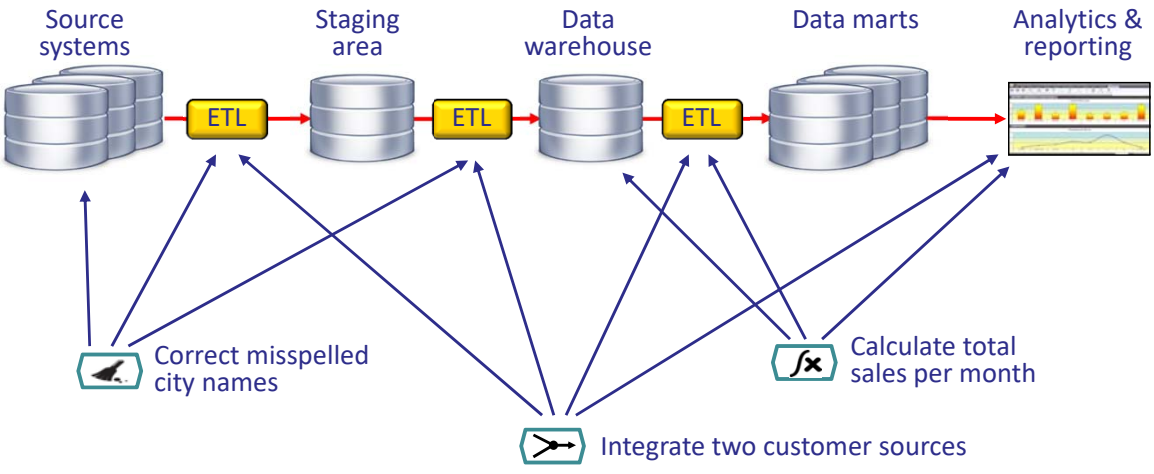
Dupulicate Transformers in Transformer Systems



Transformers in Data Warehouse Environments



Where to Implement Transformers?



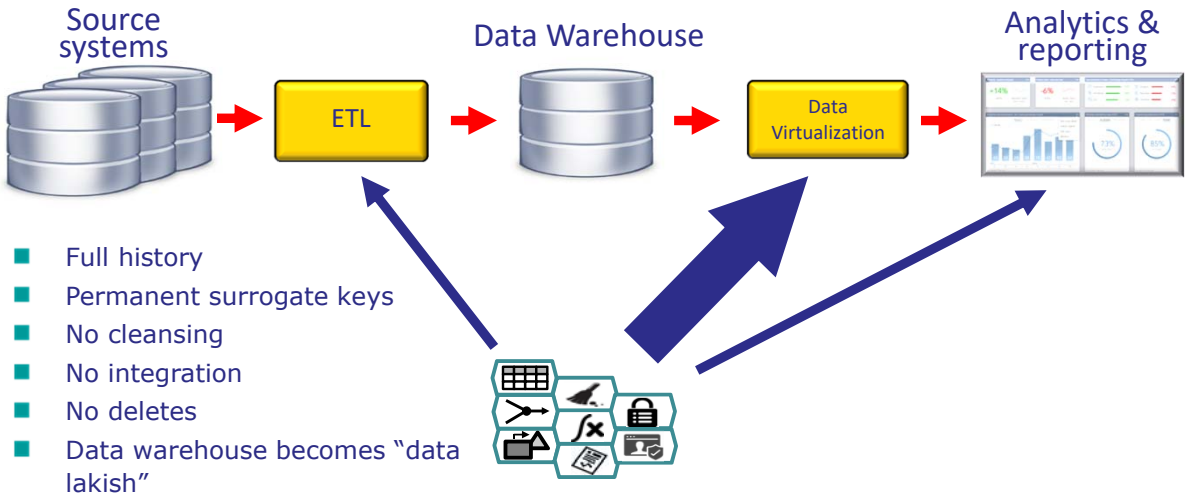
Example: Integration and Aggregation (1)

- Slow processing of integration logic in reports
- Complex queries in reports
- No sharing of integration logic across reports and tools
- Potential errors and inconsistencies in reports
- Fast copying and lower data latency
- Data structure of source database determines all data structures
- Use of original raw data possible
- What about integration errors?

Example: Integration and Aggregation (2)

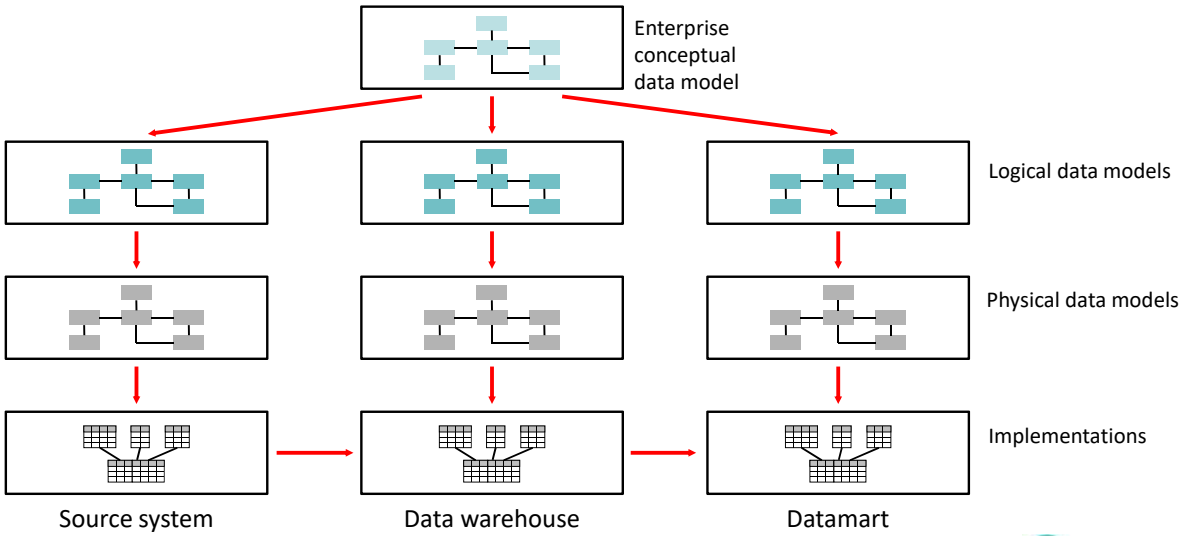
- Fast processing of integration logic in reports
- Simpler queries in reports
- Sharing of integration logic across reports and tools
- Potential errors and inconsistencies in ETL
- Slower copying and higher data latency
- Data structure of source database does not determine all data structures
- Use of original raw data possible
- Integration errors easier to fix

Implementing the Transformers

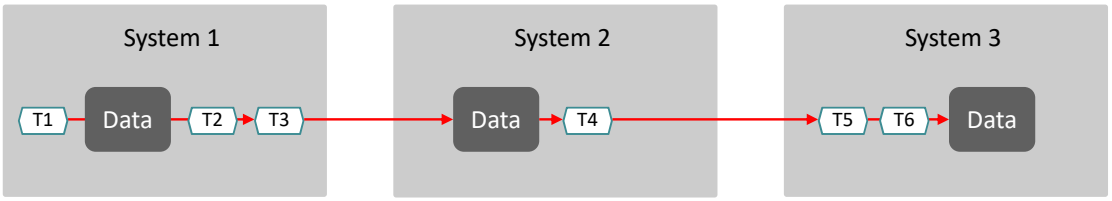


- Full history
- Permanent surrogate keys
- No cleansing
- No integration
- No deletes
- Data warehouse becomes "data lakish"

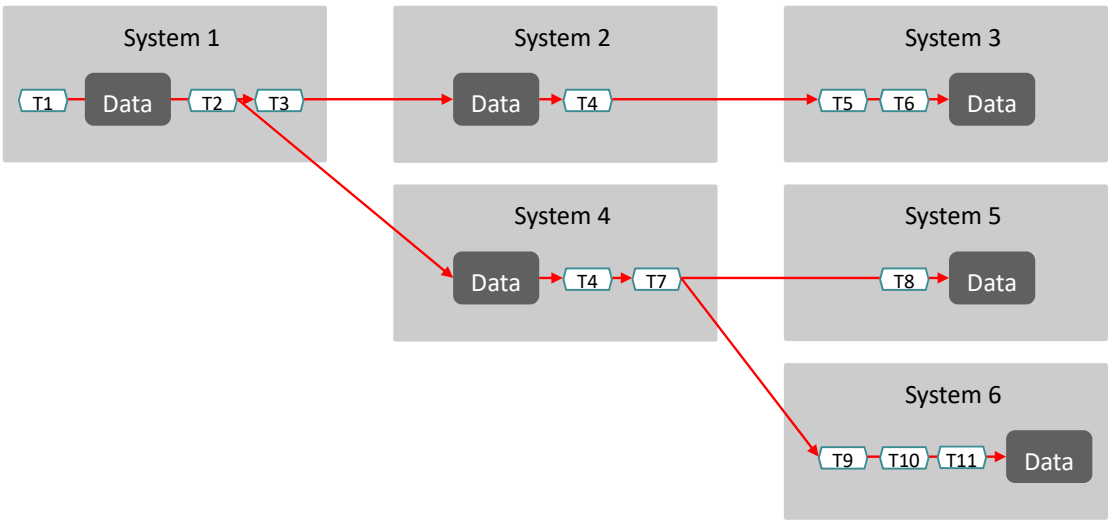
Vertical and Horizontal Lineage



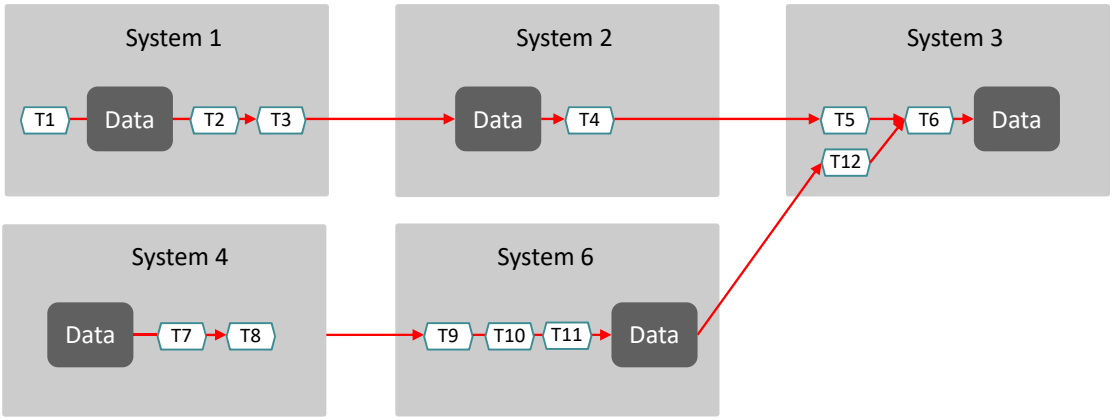
Example of a Data Path with Transformers



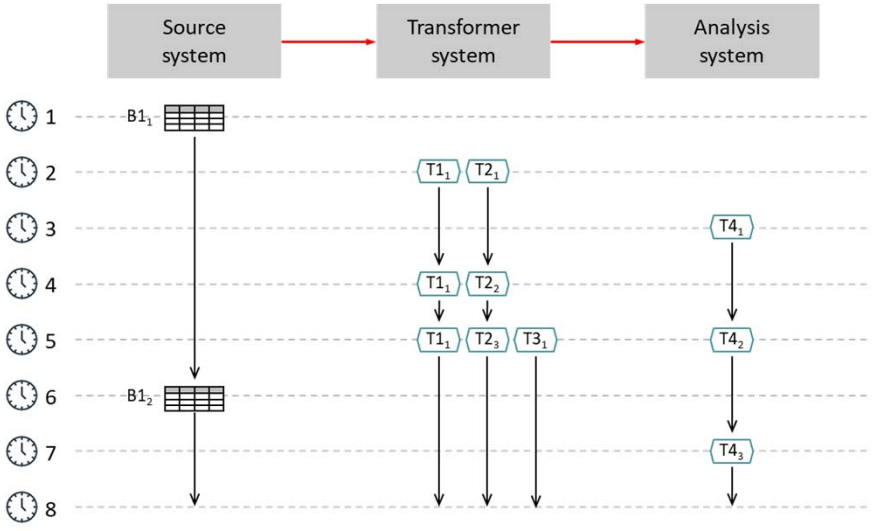
Data Path with Splits



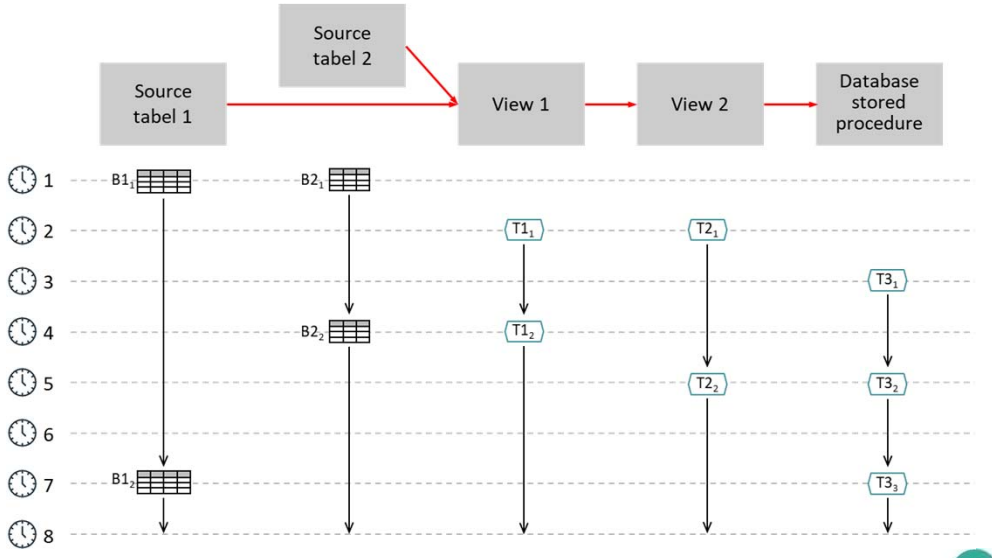
Data Path with Joins



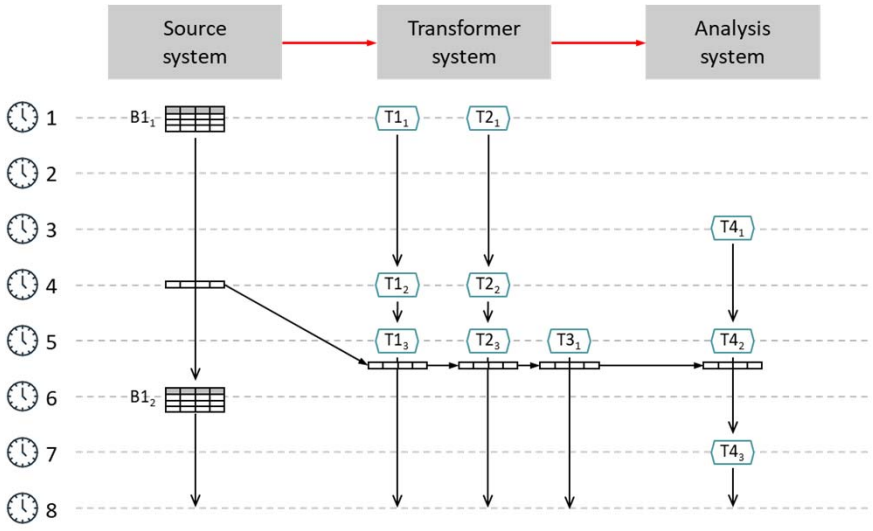
Example Horizontal Lineage (1)



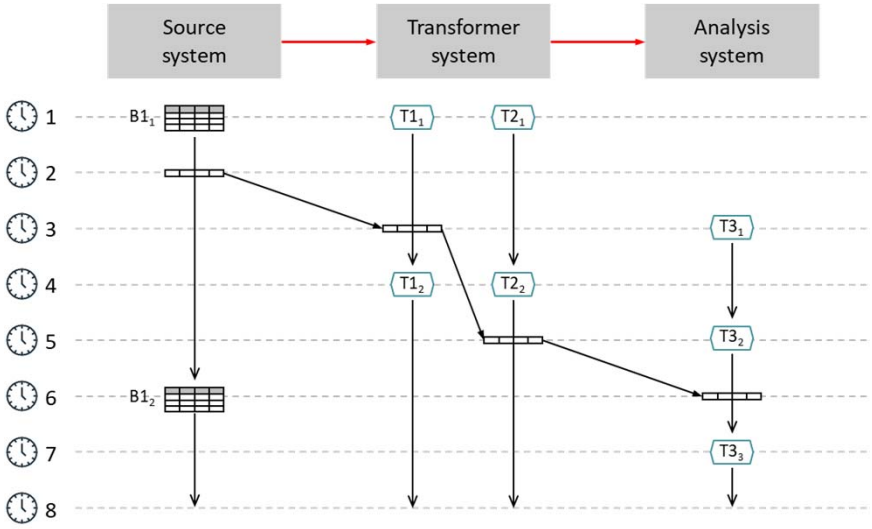
Example Horizontal Lineage (2) Within a Database



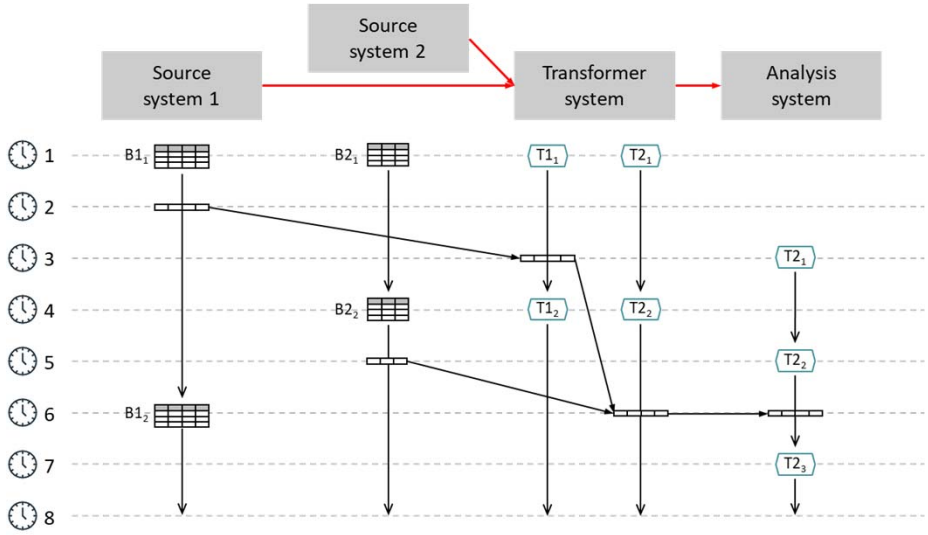
Operational Lineage (1) Simple Realtime Query



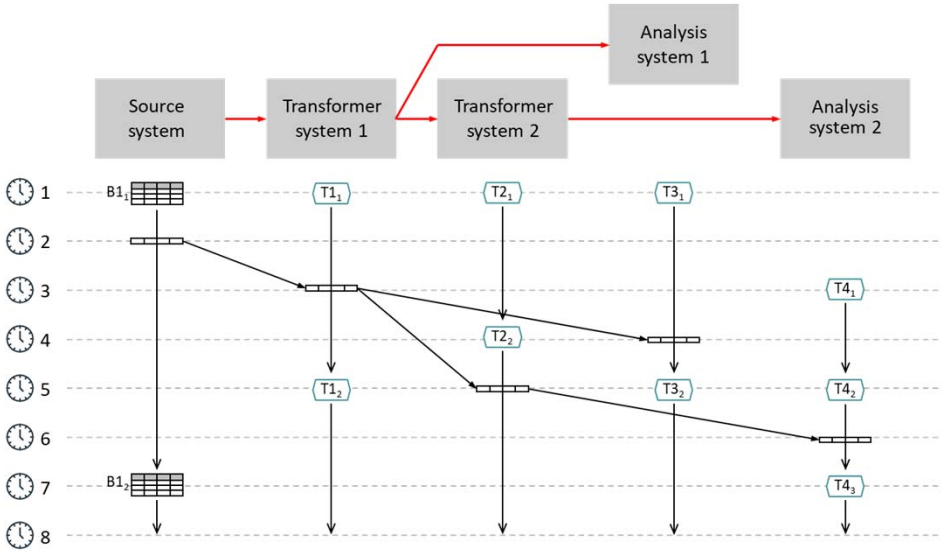
Operational Lineage (2) With Data Copies

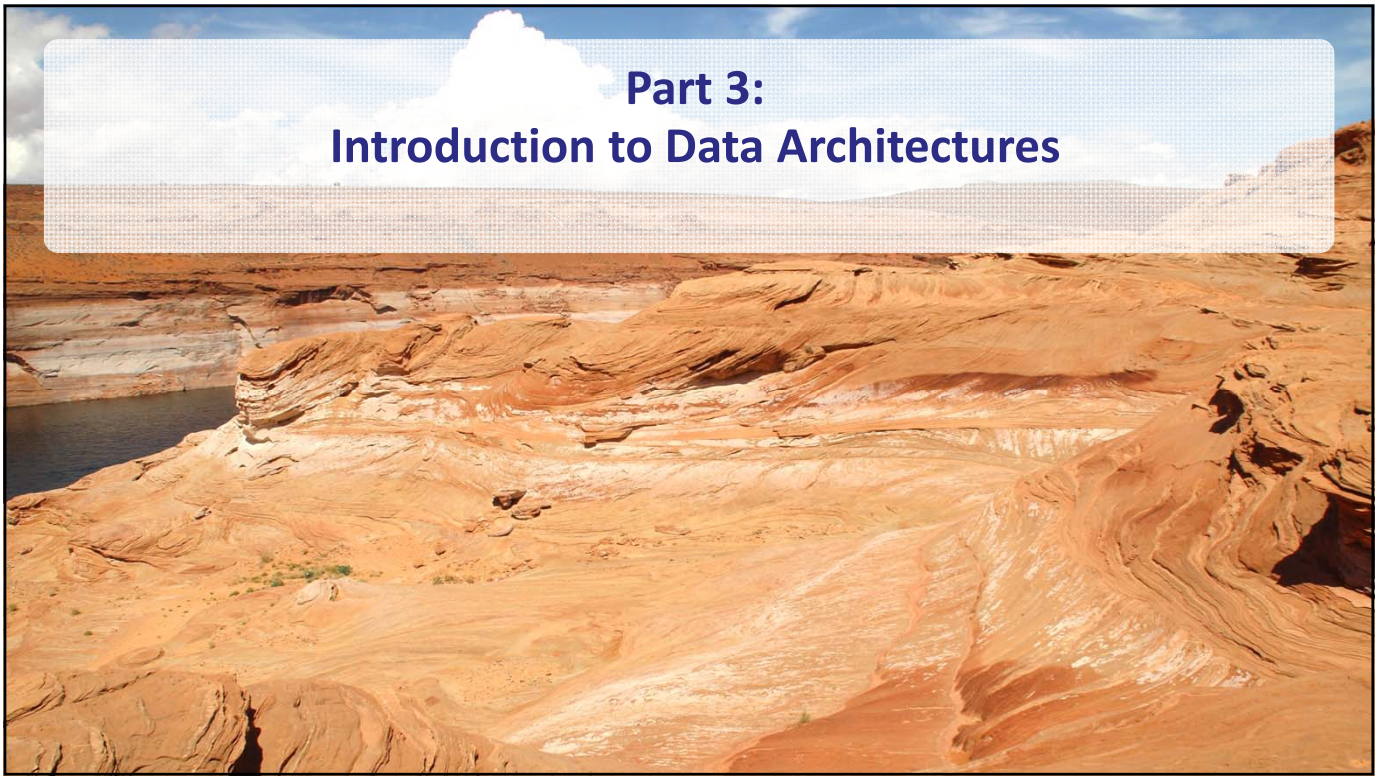


Operational Lineage (3) Integration



Operational Lineage (4) Split

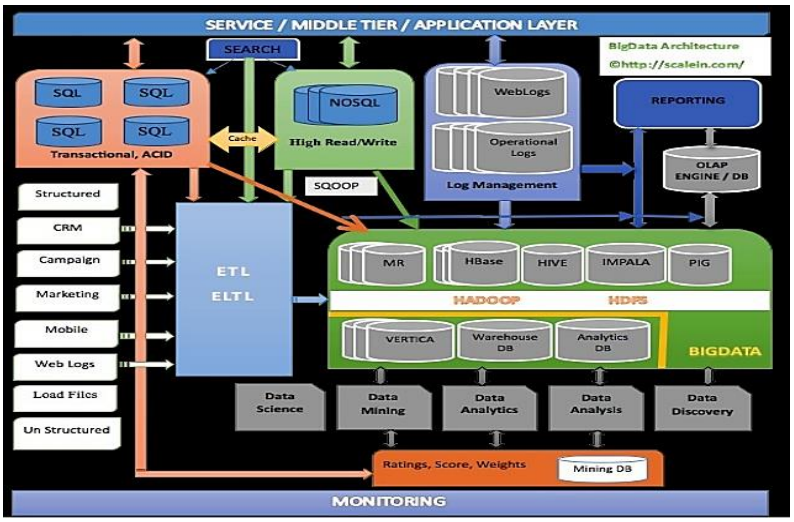




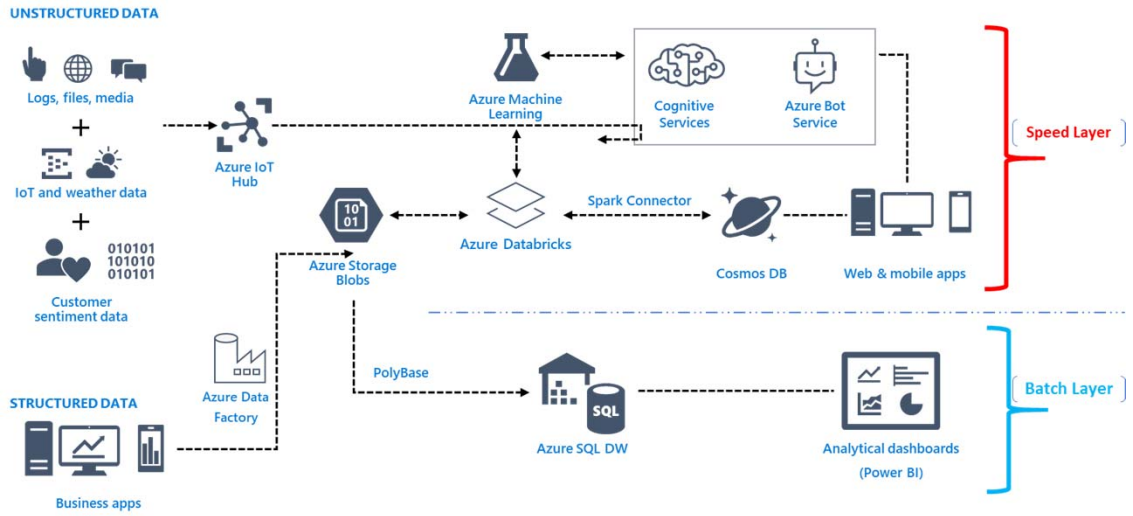
What is a Data Architecture?

- Wikipedia: A *data architecture* is composed of models, policies, rules or standards that govern which data is collected, and how it is stored, arranged, integrated, and put to use in data systems and in organizations.
- Examples of data architectures:
  - Data warehouse architecture
  - Data streaming architecture
  - Transactional system

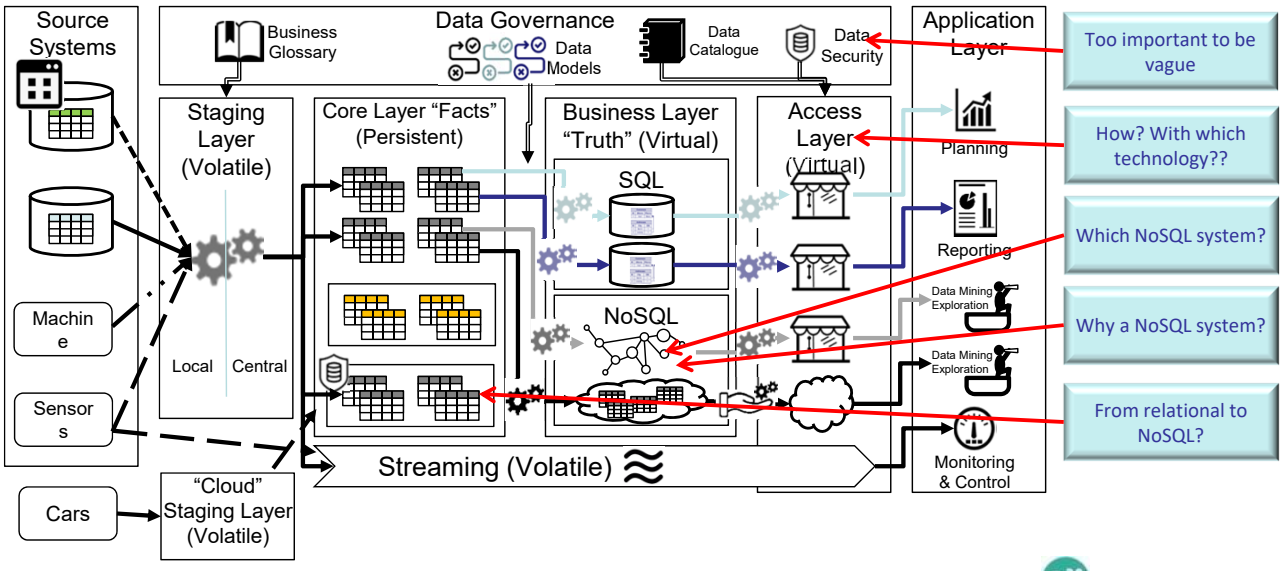
Example Data Architecture (1)



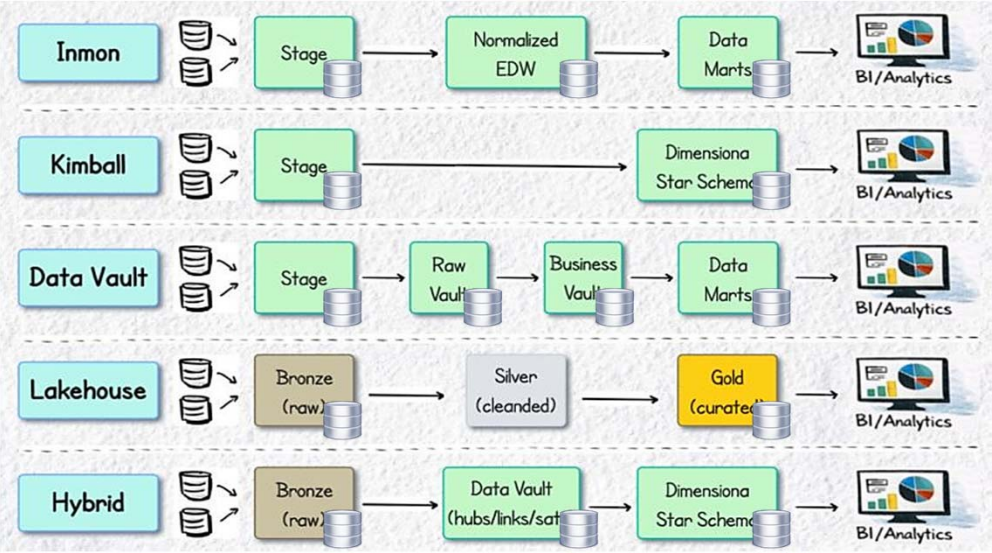
Example Data Architecture (3)



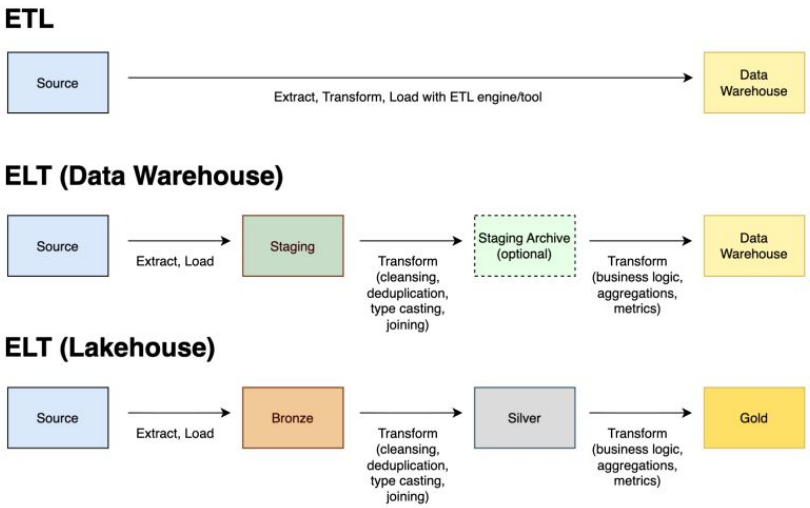
Example Data Architecture (4)



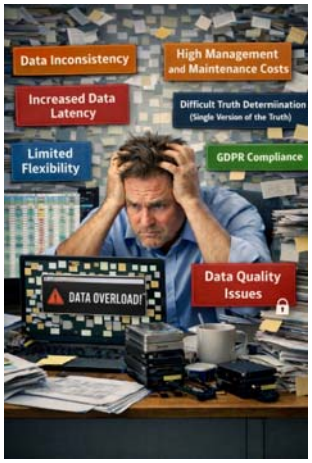
Data-copy Rich, Partial Data Architectures (1)



Data-copy Rich, Partial Data Architectures (2)

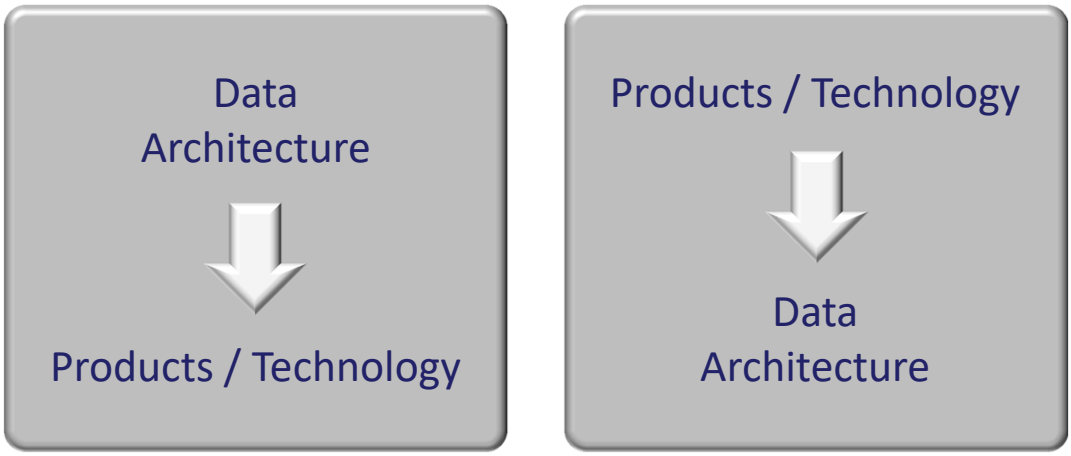


Disadvantages of Data Copying



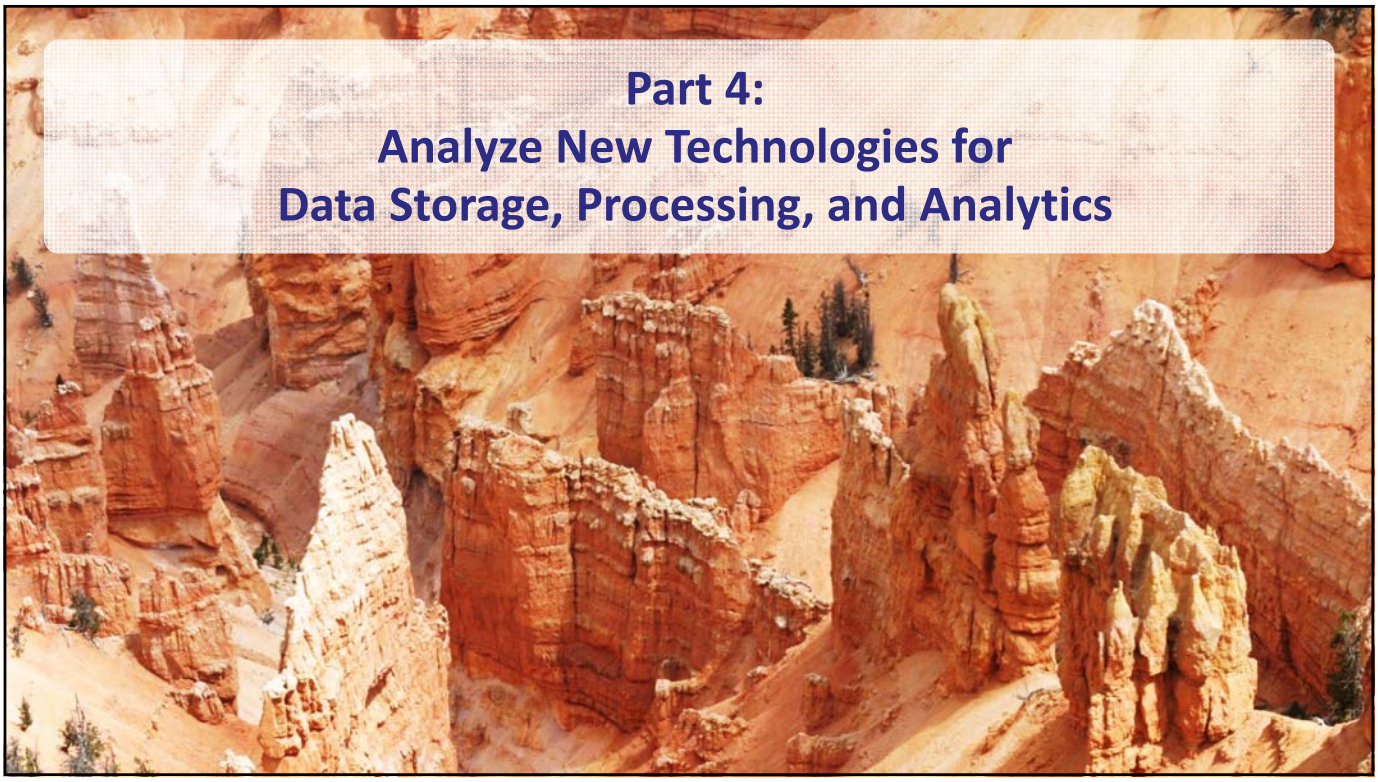
- Data inconsistency
- Difficult determination of the truth ("single version of the truth")
- Limited flexibility and agility
- High management, development, and maintenance costs
- More complex compliance with General Data Protection Regulation (GDPR)
- Complex data synchronization issues
- Data quality challenges
- More complex data security
- Outdated data (increased data latency)
- …

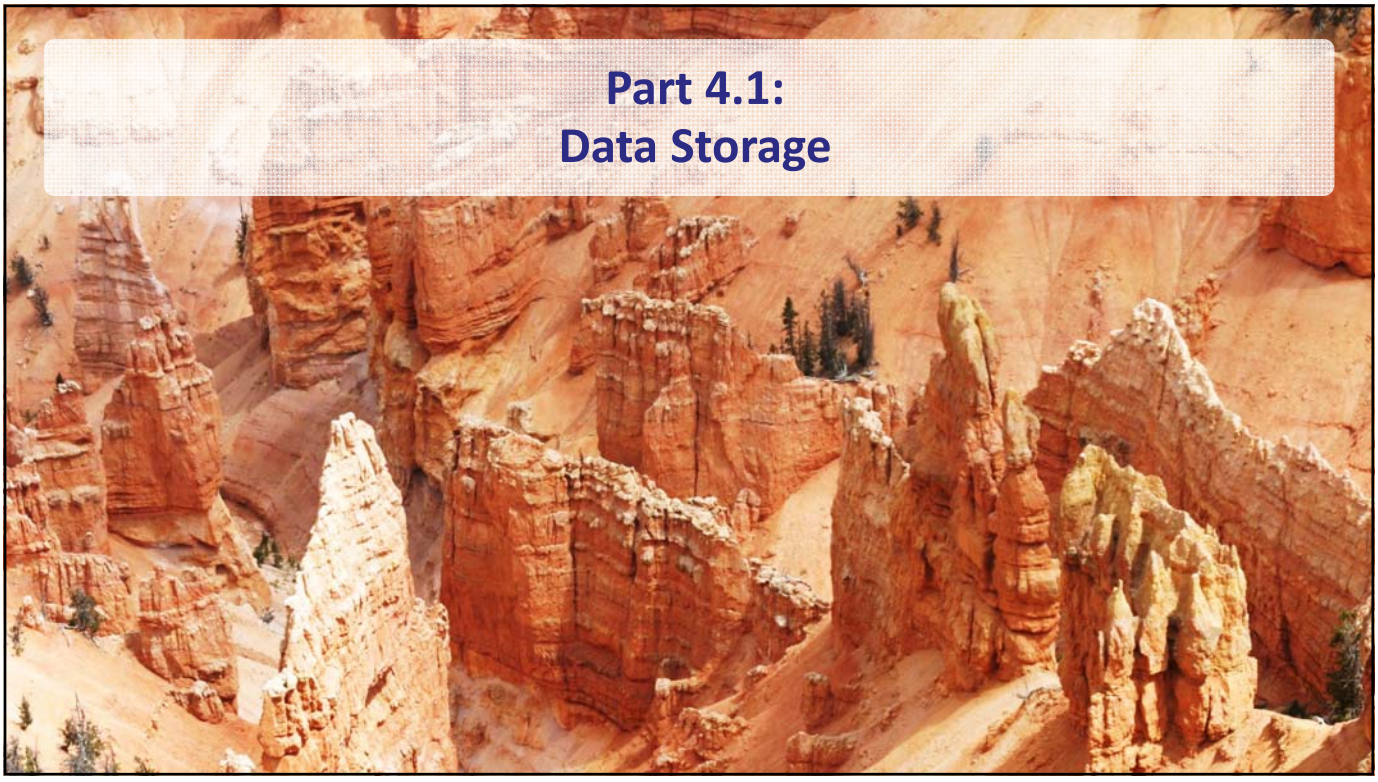
What Comes First?



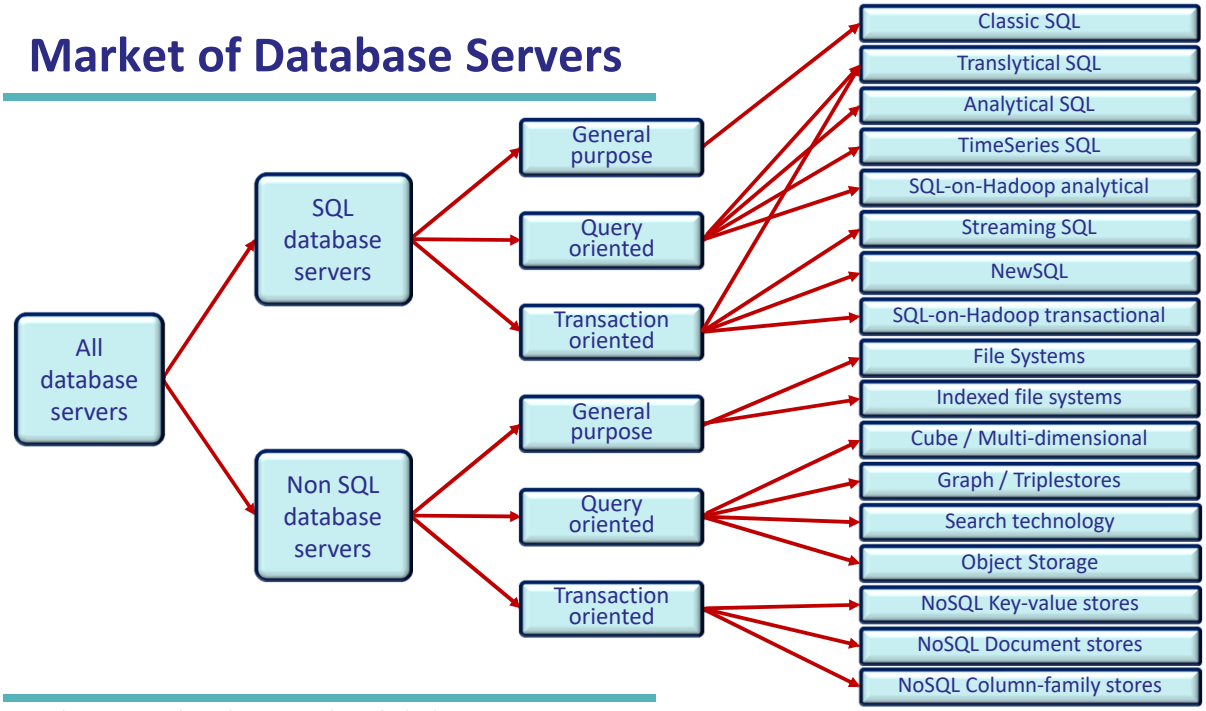
Roadmap for Designing Data Architectures

1. Determine business motivations
2. Determine new requirements
3. Analyze the existing environment
4. Define architectureal design principles
5. Selecta reference data architecture
6. Desogn the new data architecture
7. Determine the implementation approach
8. Select new products and technologies
9. Introduce the data architecture within the organization

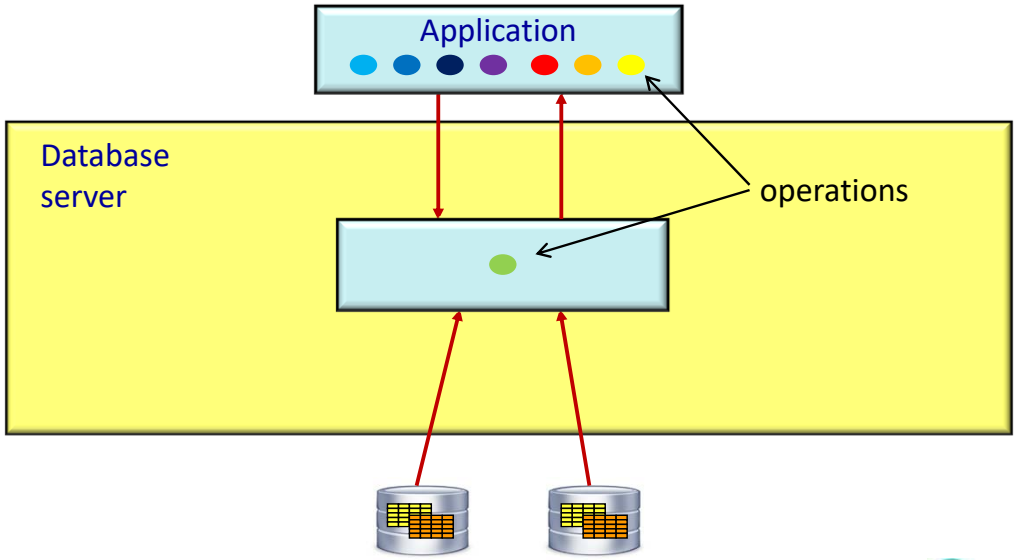

Part 4:
Analyze New Technologies for
Data Storage, Processing, and Analytics


Part 4.1:
Data Storage

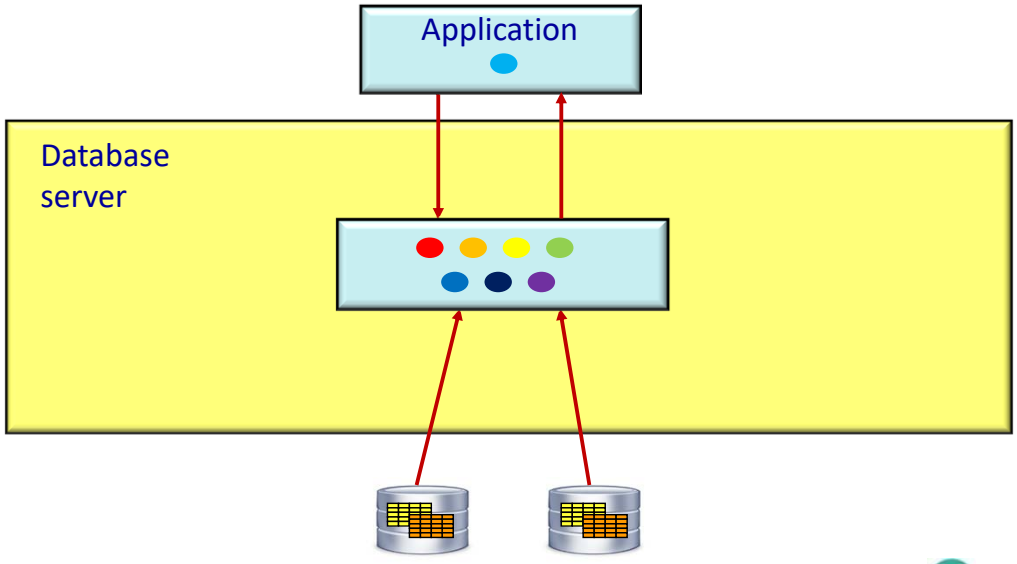
Market of Database Servers



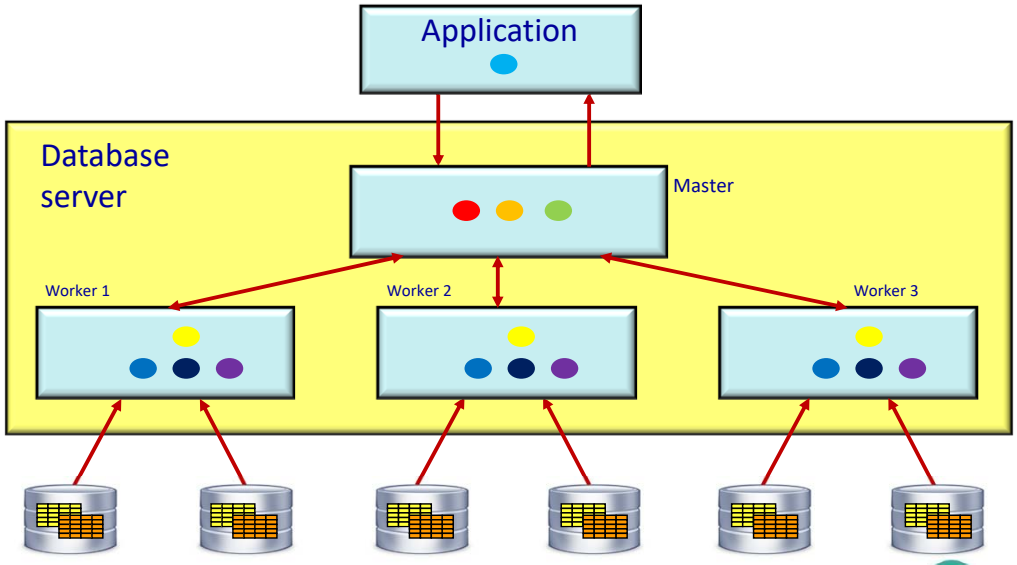
Application-based Analytics



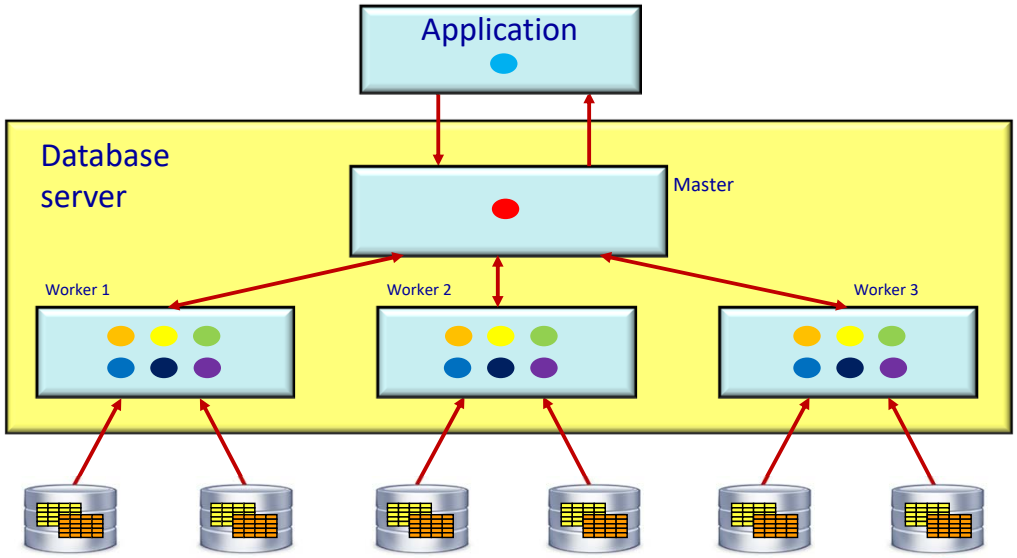
In-Database Analytics



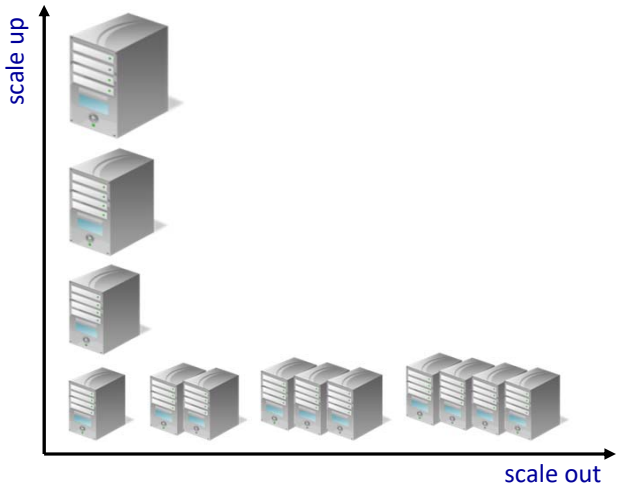
Partial Parallel Analytics



Full Parallel Analytics

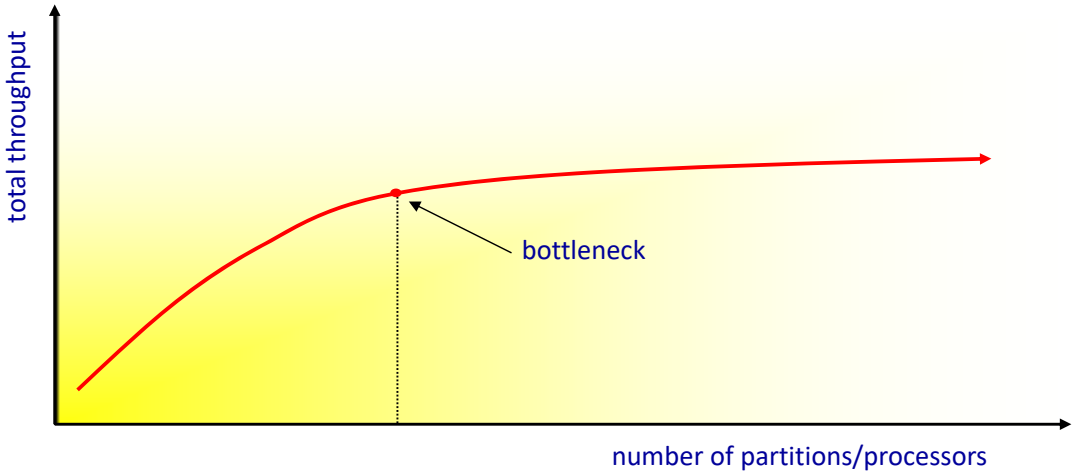


Scale Up versus Scale Out



- Scale up (vertical scaling) means adding more resources to one node in a system
- Scale out (horizontal scaling) means adding more nodes to a system
  - Continuous availability/redundancy
  - Cost/performance flexibility
  - Contiguous upgrades
  - Geographical distribution

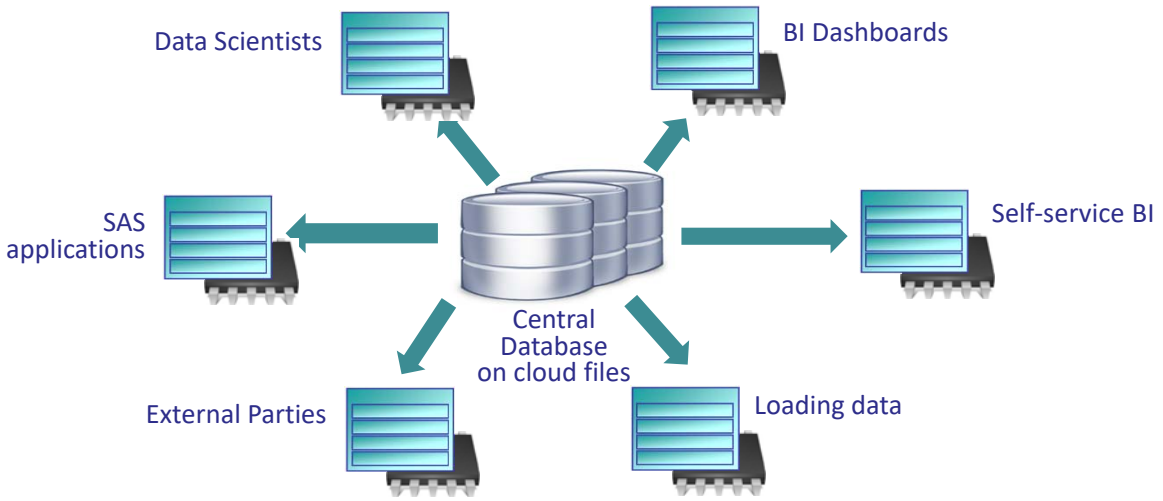
Effect of Partitions on Query Response



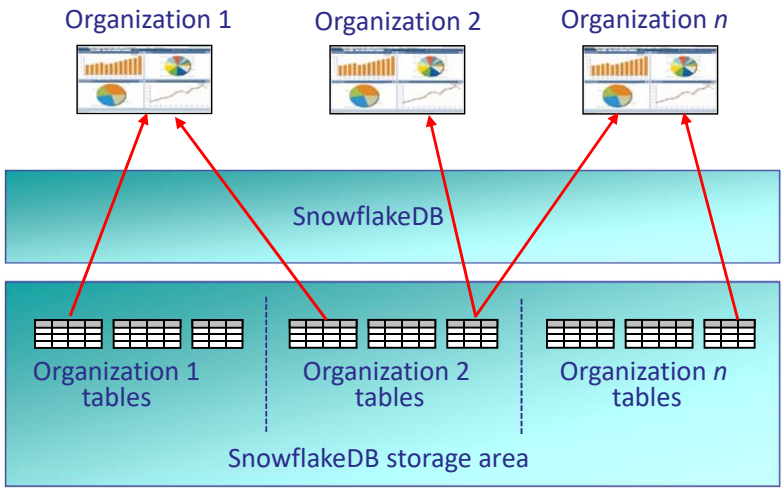
Analytical SQL Database Servers

Examples	
1010Data	Microsoft Azure Synapse
Amazon Redshift and Athena	OmniSci (MapD)
Apache HAWQ	Oracle Database In-Memory
BlazingSQL	SAP HANA
CitusDB (PostgreSQL)	SAP Sybase IQ
ClickHouse SQL	SnowflakeDB
Databricks Delta Lake	Splice Machine
Edge Intelligence	SQream
Exasol	Starburst (Trino formerly Presto)
Google BigQuery	Teradata Vantage
Greenplum	XTremeData dbX
IBM DB2 Warehouse on Cloud	Several SQL-on-Hadoop engines
Ignite InfoBright DB	Vertica
Kinetica	And many others …
Kognitio WX2	

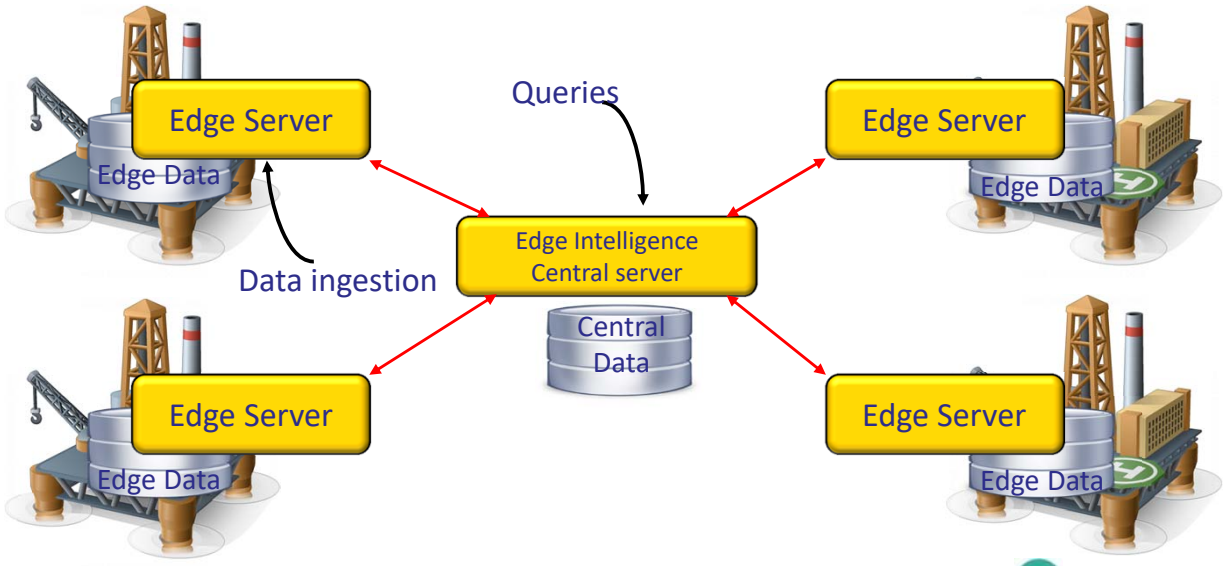
Example 1: Snowflake



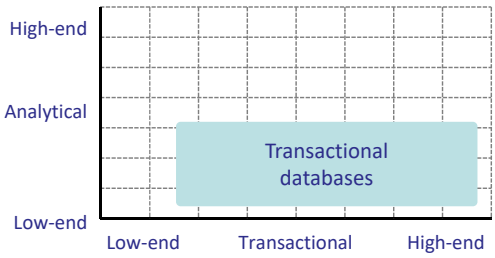
Example 1: Snowflake

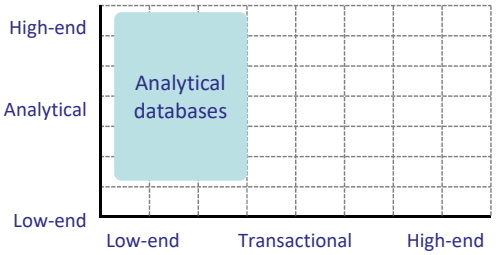


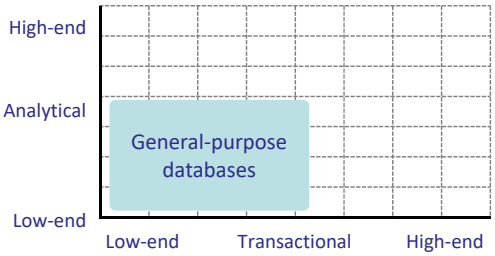
Example 2: Edge Intelligence

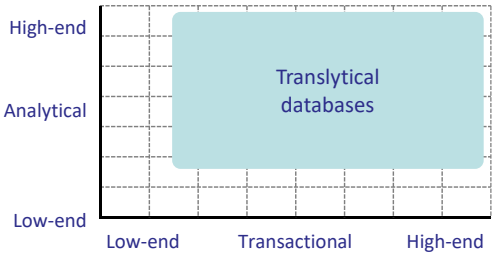


Four Categories of SQL Databases



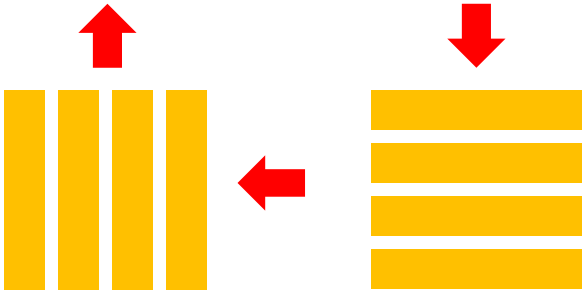


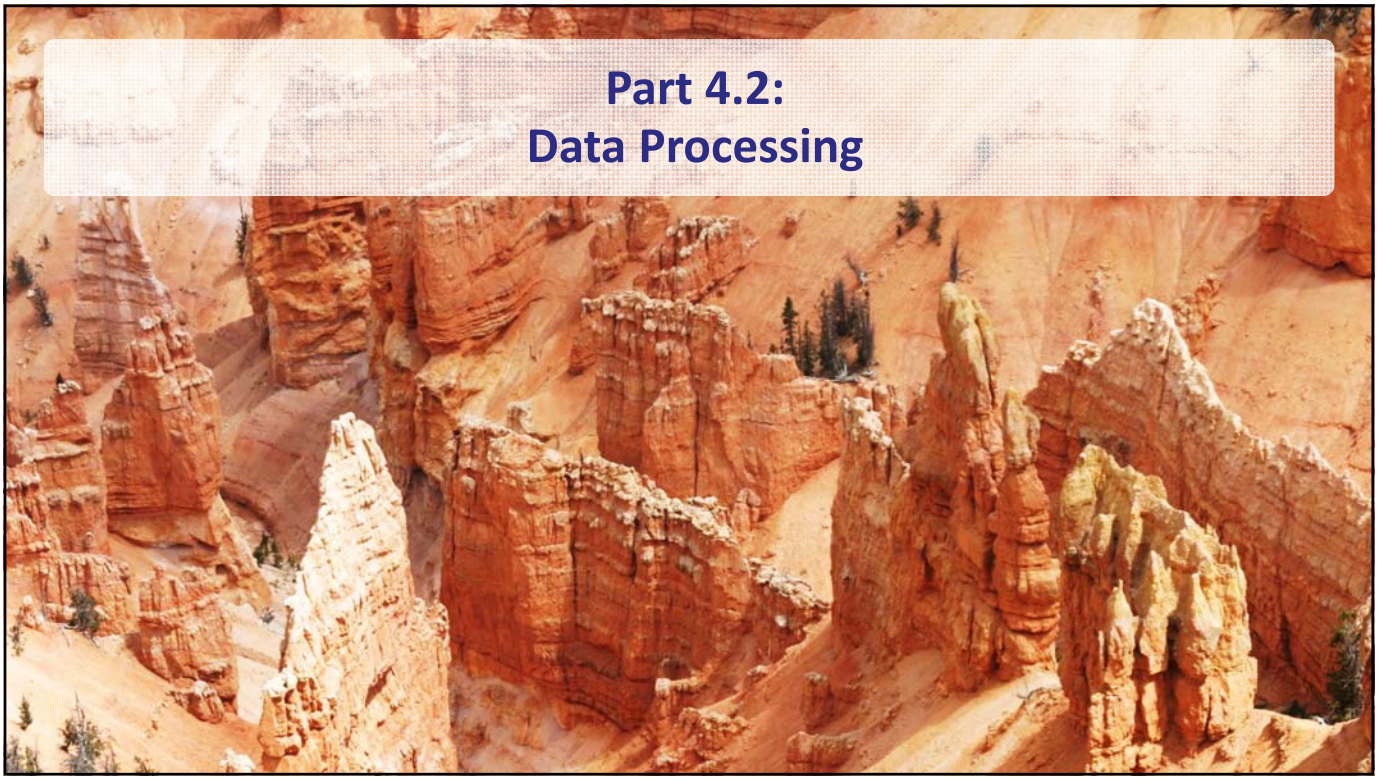




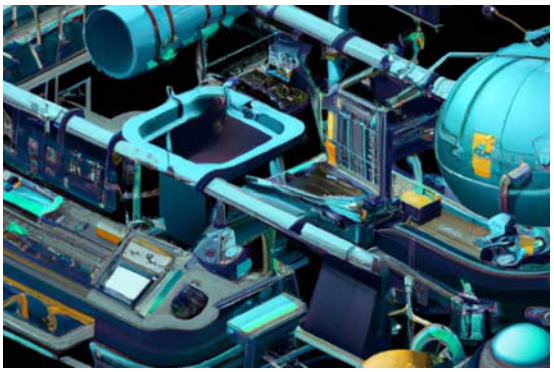
Example: SingleStore (translytical)

ID	Name	Initials	Date Entered	City	State
12345	Young	N	Aug 4, 2008	San Francisco	CA
23324	Stills	S	Sep 10, 2009	New Orleans	LA
57657	Furay	R	Oct 16, 2010	Yellow Springs	OH
65461	Palmer	B	Nov 22, 2011	Boston	MA
…	…	…	…	…	…



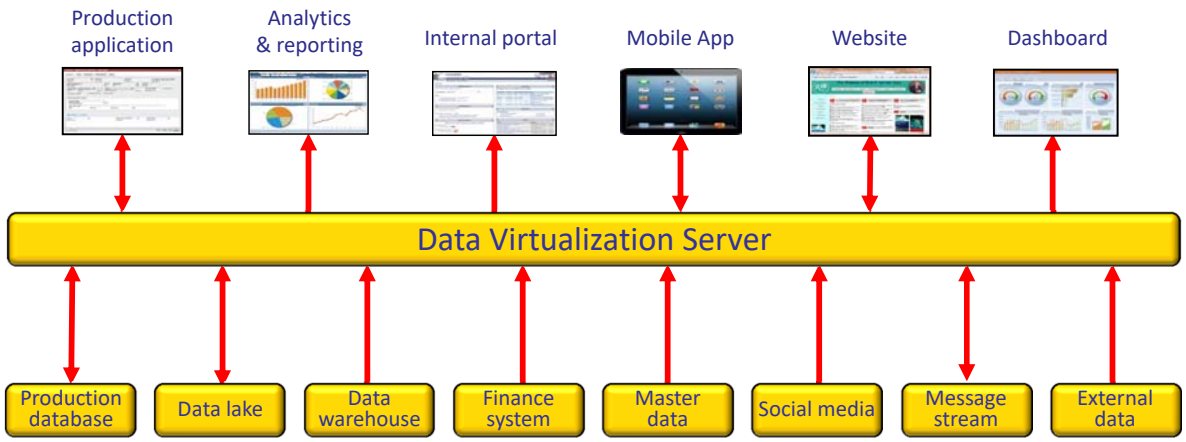


Categories for Data Processing

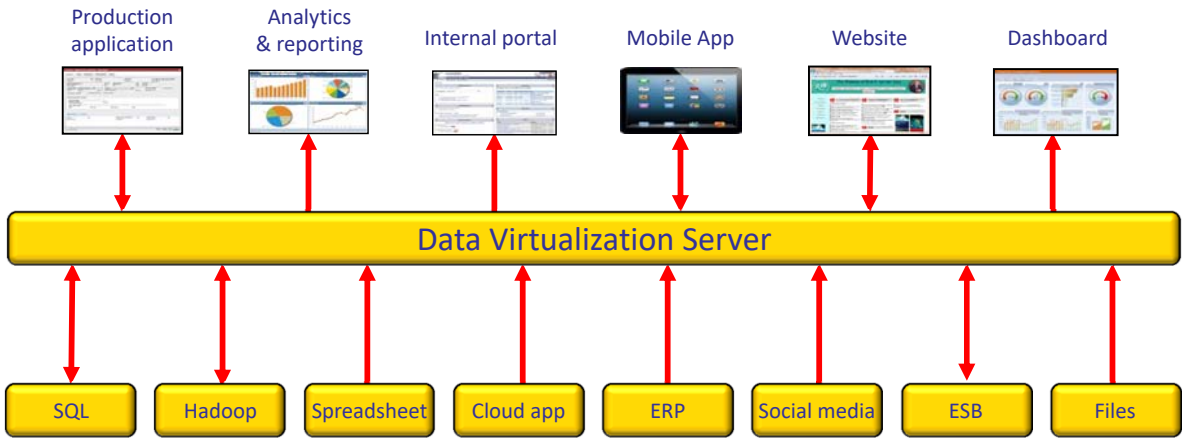


- ETL (Extract Transform Load)
- Data Replication (Change Data Capture)
- ESB (Enterprise Service Bus)
- Data Virtualization

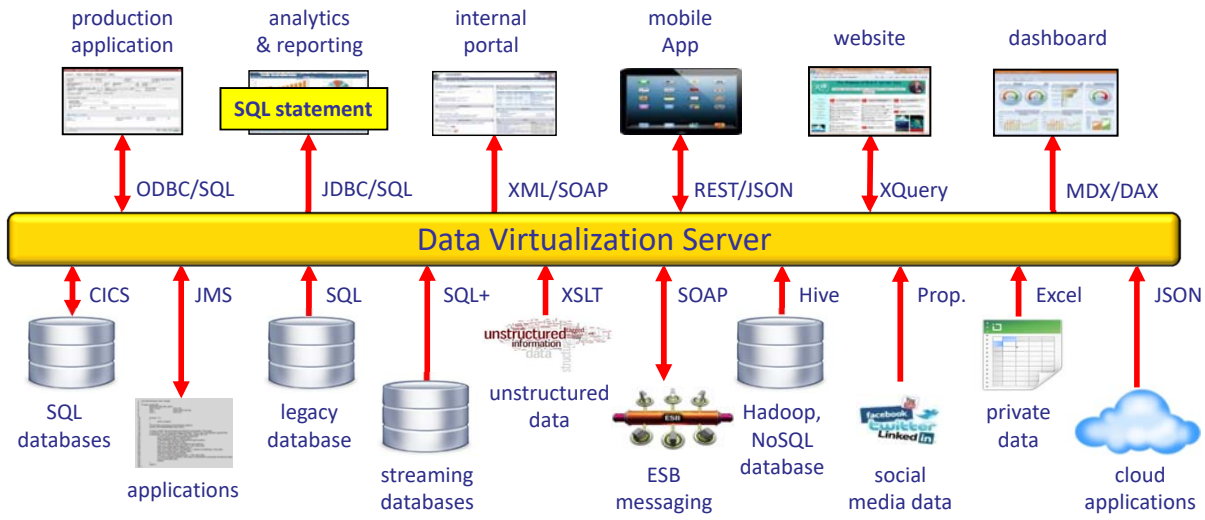
Data Virtualization Overview (1)



Data Virtualization Overview (2)



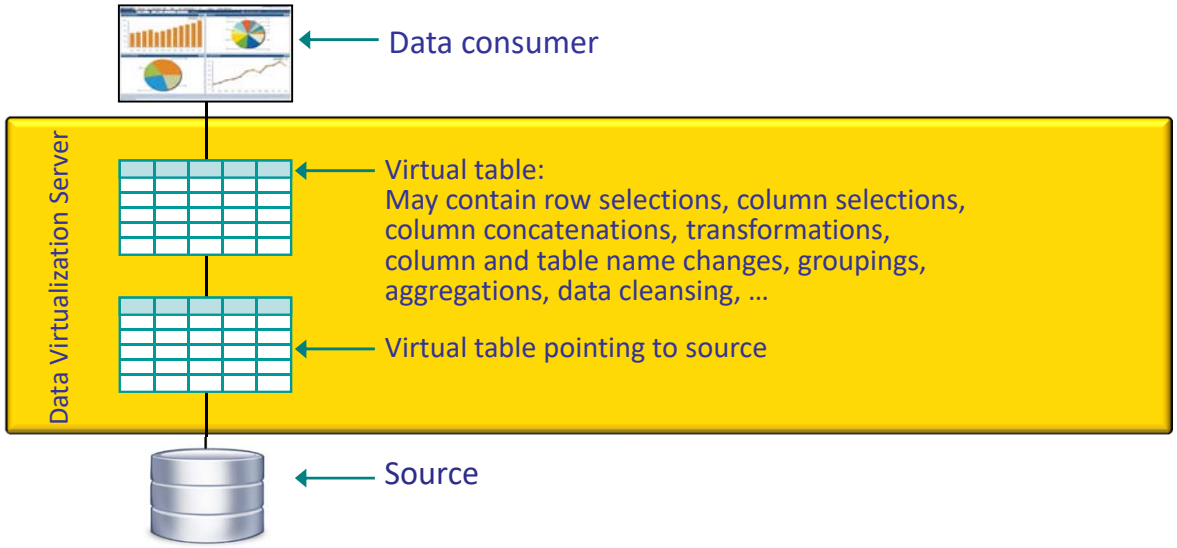
Data Virtualization Overview (3)



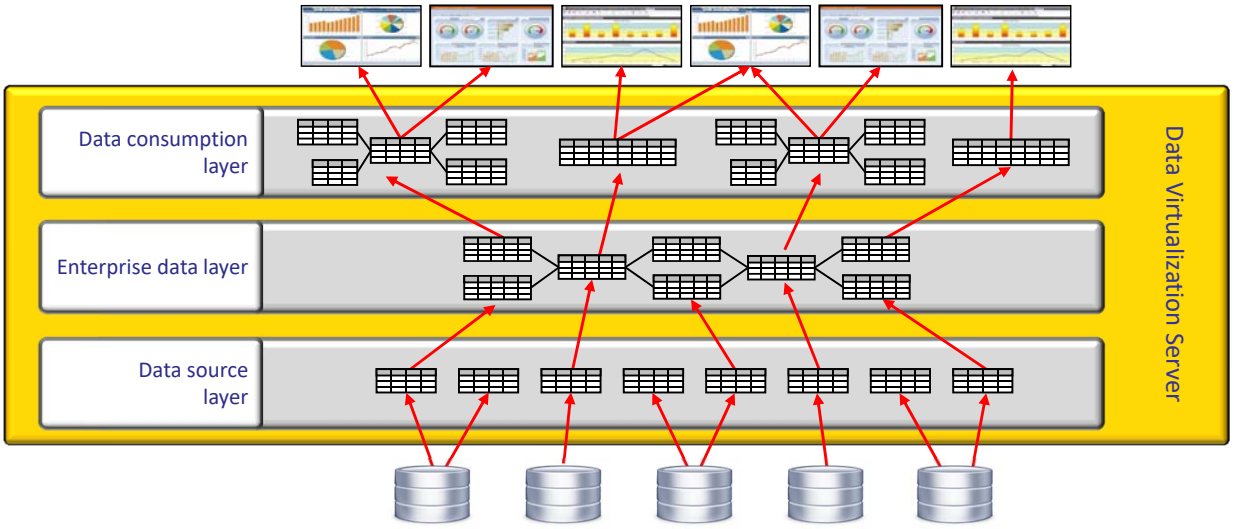
The Market of Data Virtualization Servers

- AtScale
- DataVirtuality (Pipes, Pipes Prof, LDW)
- Denodo Platform
- Dremio
- Fraxses
- IBM InfoSphere Federation Server & IBM Data Virtualization Manager for z/OS (formerly Rocket Data Virtualization)
- Red Hat JBoss Data Virtualization (Teiid) ??
- Stone Bond Enterprise Enabler Virtuoso
- TIBCOData Virtualization (formerly Cisco & Composite)
- Zetaris
- And many more …

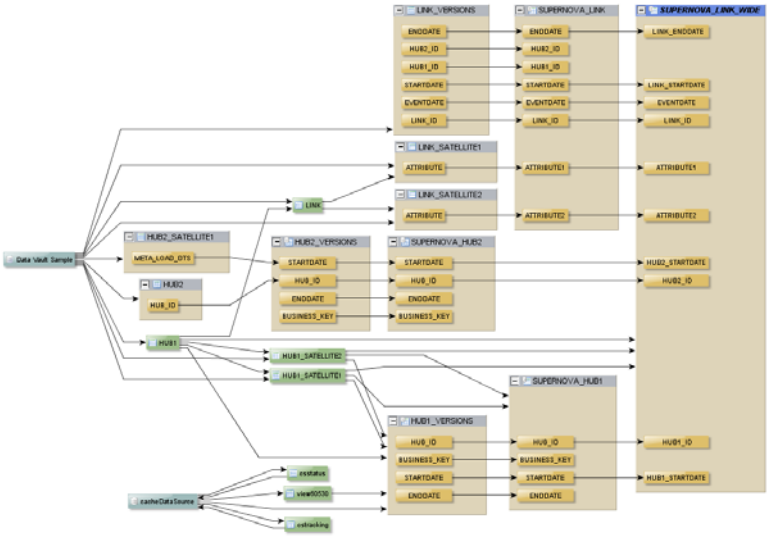
Developing Virtual Tables



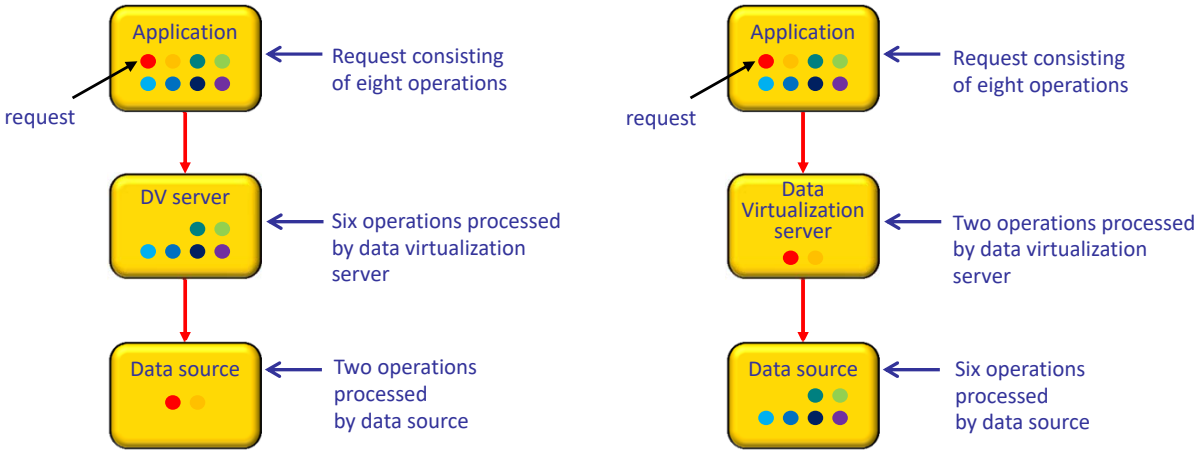
Layers of Virtual Tables



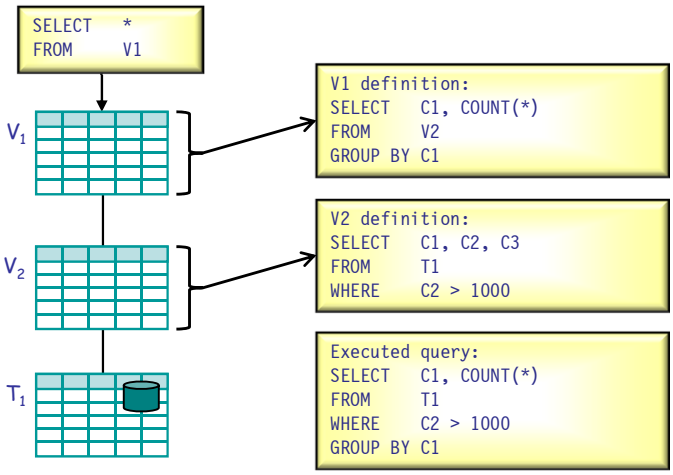
Lineage and Impact Analysis



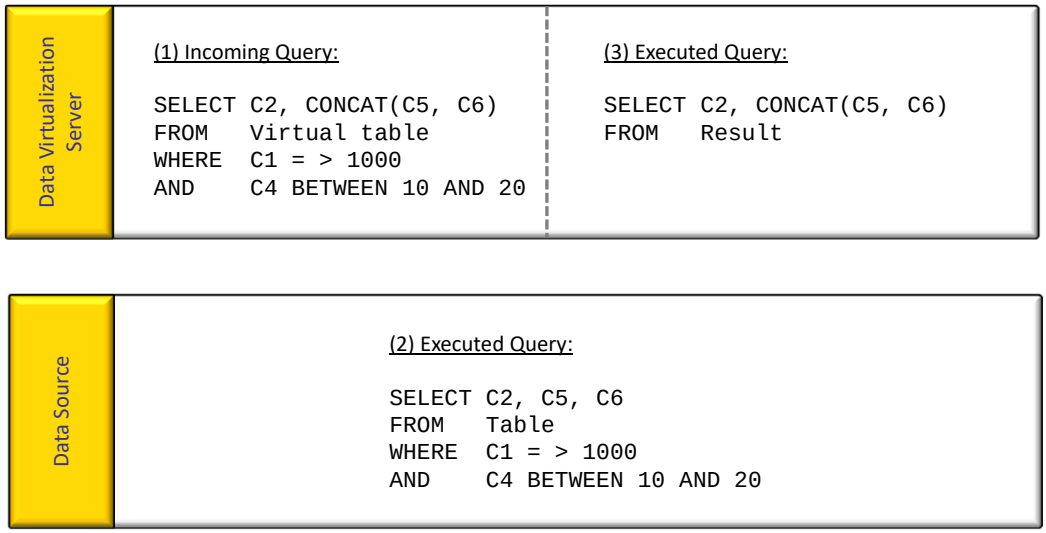
Improved Performance Through Query Pushdown



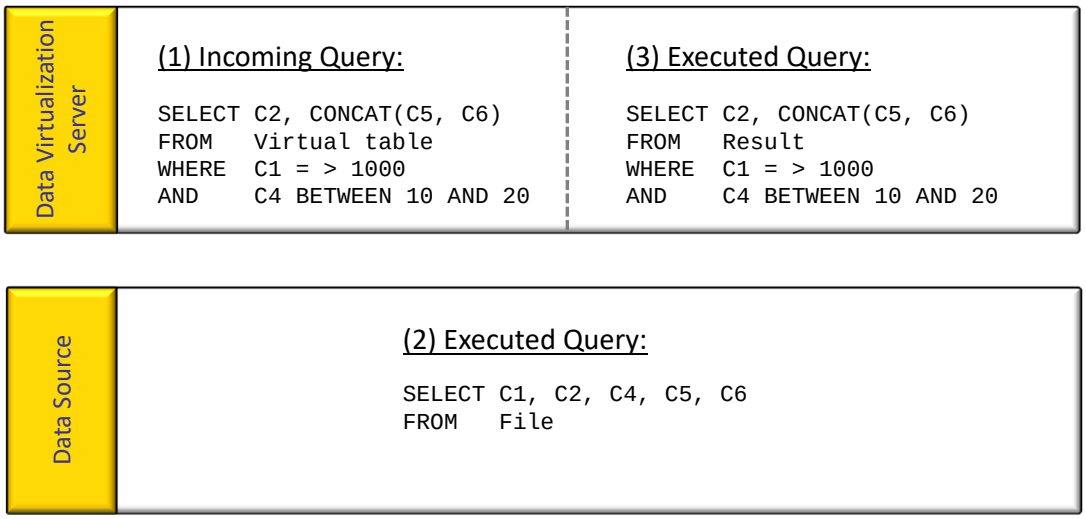
Many Levels and One Query



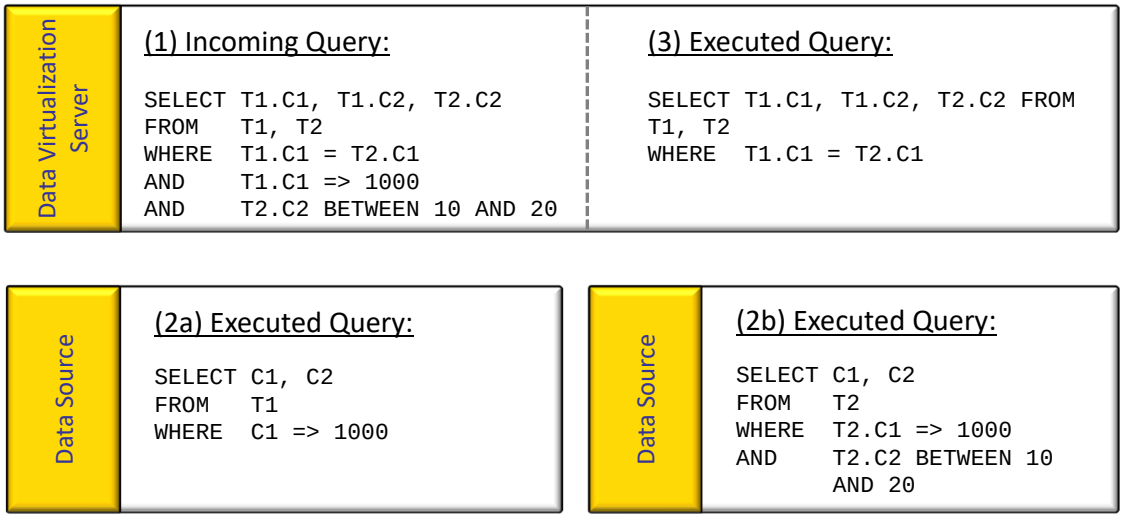
Push Down Query Processing



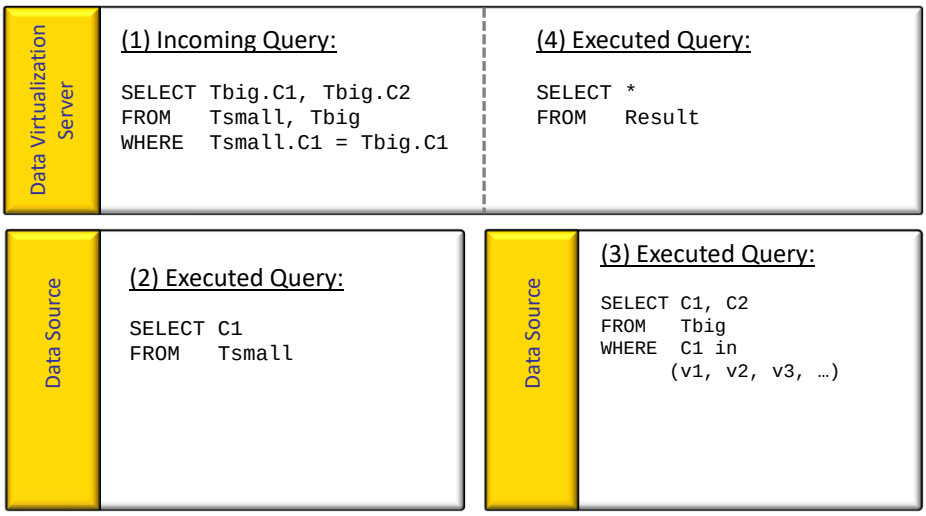
Accessing Files



Inferred Filters



Join of Data Sources with Query Injection



Join of Data Sources with Ship Join

Data Virtualization Server

(1) Incoming Query:

```
SELECT T1.C1, T1.C2, T2. C2
FROM   Tbig1, Tbig2
WHERE  Tbig1.C1 = Tbig2.C1
```

(4) Executed Query:

```
SELECT *
FROM   Result
```

Data Source

(2) Executed Query:

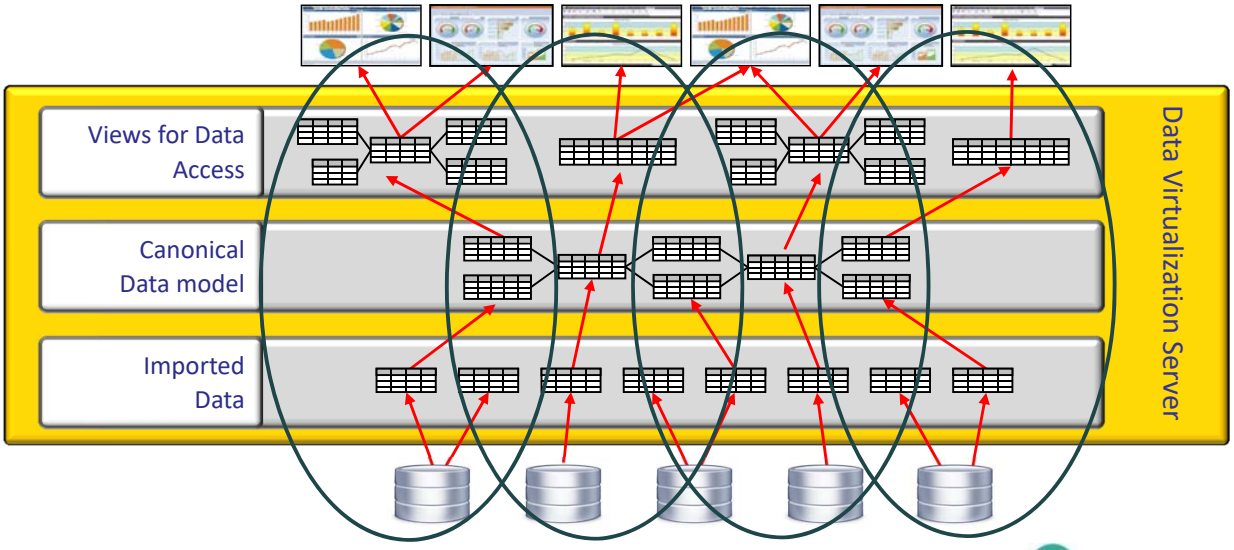
```
SELECT C1, C2
FROM   Tbig1
```

Data Source

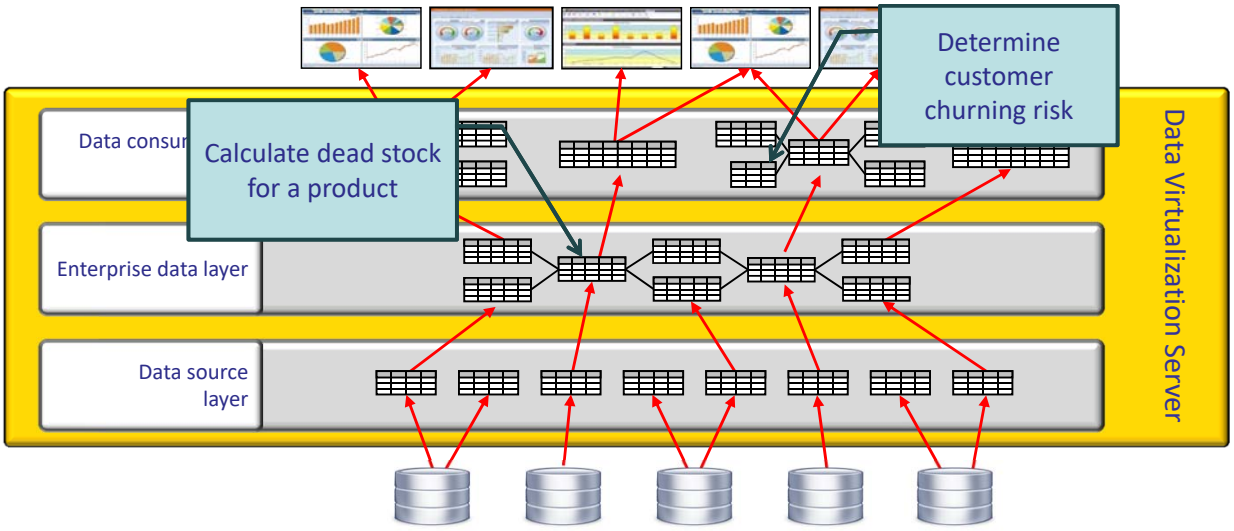
(3) Executed Query:

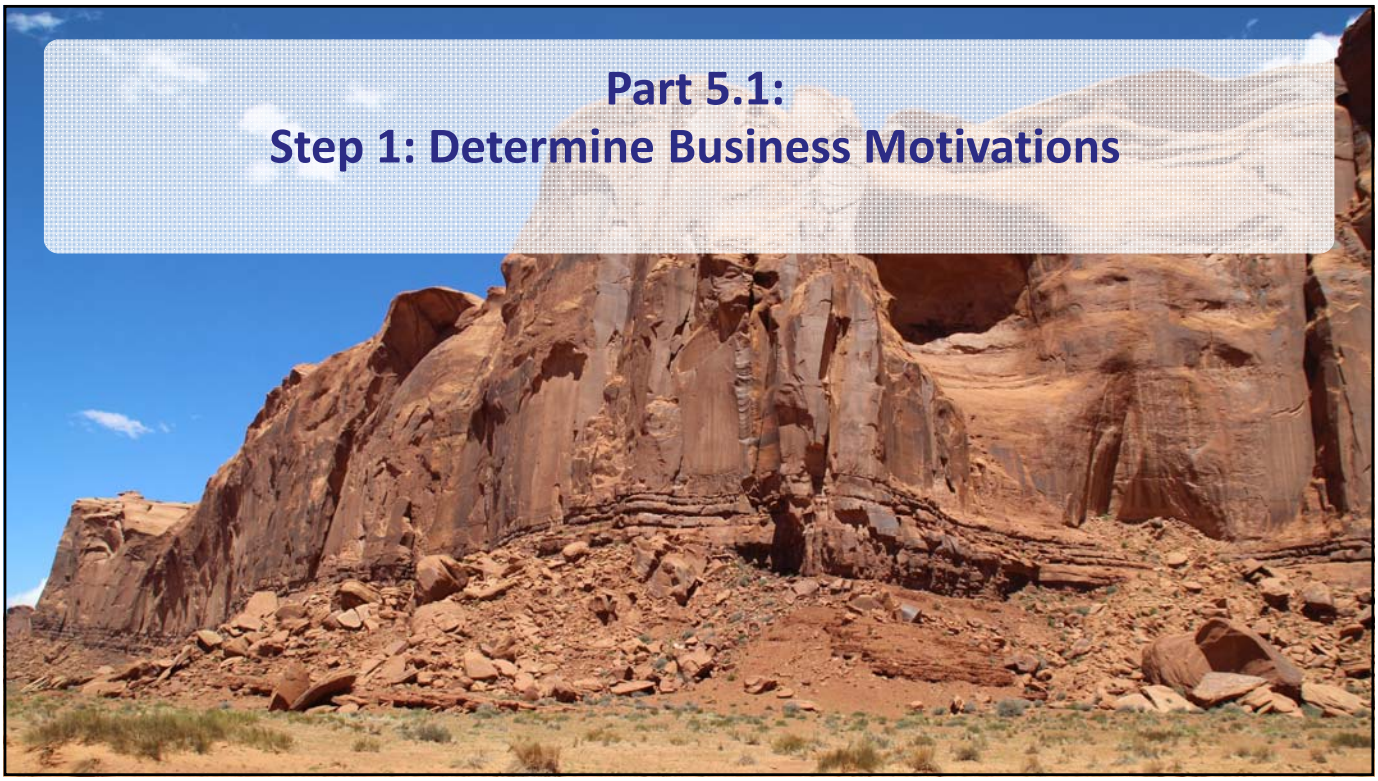
```
CREATE TEMP TABLE TEMP;
INSERT INTO TEMP
SELECT *
FROM   Tbig1;
SELECT TEMP.C1, TEMP.C2,
       Tbig2.C2
FROM   TEMP, Tbig2
WHERE  TEMP.C1 = Tbig2.C1;
DROP TEMPORARY TABLE TEMP;
```


Evolutionary Development Approach



Improved Productity Through Sharing





Part 5.1:
Step 1: Determine Business Motivations

Poor Examples of Business Motivations

- Change insights and requirements
- Deployment of self-service BI
- Optimization of existing data architecture
- The platform on which the current BI system is hosted externally is old and needs to be replaced
- Move to the cloud
- Data science is not very well supported by current data warehouse environment
- We want to do more with the data we have, but it's hard to get to it

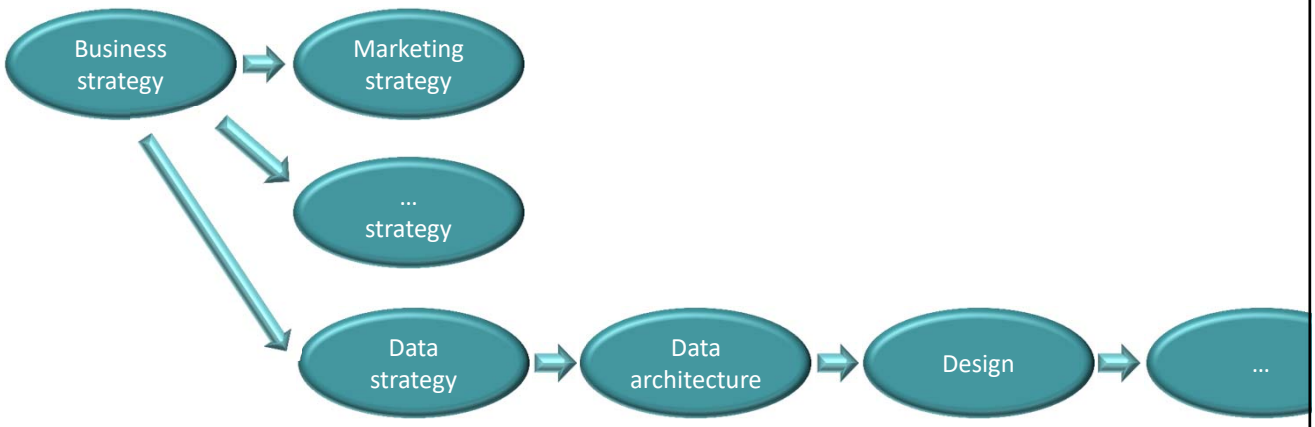
Proper Business Motivations

- Competitive improvement
  - Improving reaction speed to customer requests
- Support for customer journey and valuestream
- New business model
  - Allow customers real-time access to data
- Lower costs of specific business processes to improve margin
- Organization under threat
  - New competitor
- Comply with new laws and regulations
  - E.g. GDPR, CCPA, PSD2

Business Strategy and Data Strategy

- Business Strategy
  - The challenges of top executives
    - New regulations, competitors, …
  - The main concerns for current business processes
  - Future business developments
    - New business domains
- Data Strategy
  - New data architecture has to "fit" the data strategy
  - New demands for data delivery

From Strategy to Data Architecture and Onwards



Determine New Requirements (1)

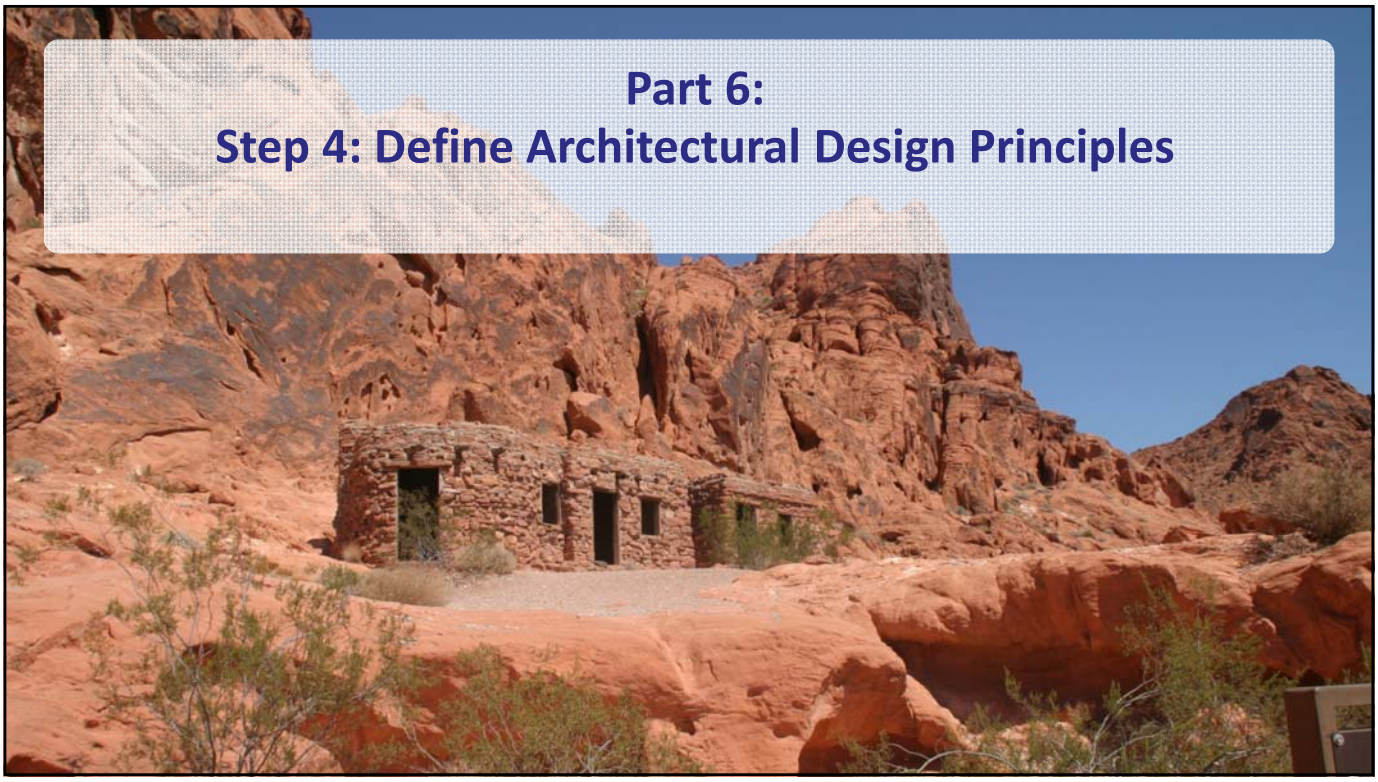

Photo: Markus Spiske

- New analytical functionality
- Lower latency for reports
- More users
- More access to metadata
- Migration to cloud platform
- More data
- More transparency of architecture
- Better security
- Deployment of data science
- …

Determine New Whishes and Requirements (2)


Photo: Markus Spiske

- Reconstruction of processes and decisions
- Data-shop to discover and describe data
- Migrate/redevelop source systems
- Synchronization of source systems
- Minimize data copies
- Sharing data with other organizations
  - Cross-organization analysis systems
- Making data AI-ready
- Strengthening data security and privacy
- Handling unstructured data: video, sound, text, and images
- Simplifying the data processing landscape
- Smarter uses of new technology
- …



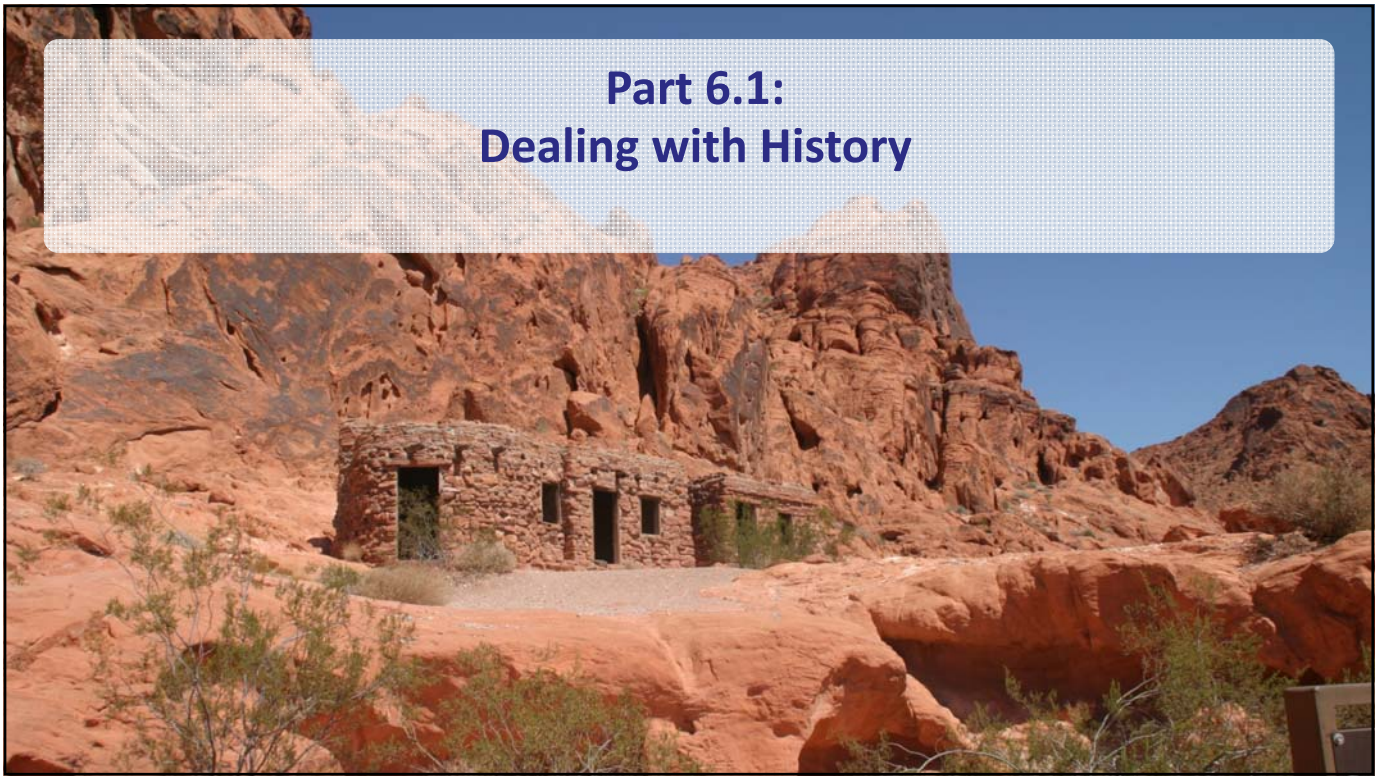
Part 6:
Step 4: Define Architectural Design Principles

Forget Old Architectural Design Principles

- No reporting on the production database
  - Reporting and transaction workloads clash
- *Physical* data marts are needed to improve reporting performance
- Data marts need a star schema design to speed up analytical queries
- ETL is used to transform data
  - Batch oriented
- When SQL databases are used
  - Indexes are required to improve query performance
  - Use locking for concurrency management
  - Not ideal for MPP
  - Need constant tuning by DBA
- ...

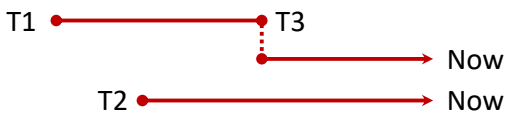
Examples of Architectural Design Principles

- Centralized and active transformers
  - Searchable definitions and descriptions for technical and business users
  - Lineage and impact analysis
- One universal architecture for all forms of data consumption
  - Standard reporting, self-service BI, apps, data science, …
- Data storage and access technology agnostic
  - Hadoop, SQL, cubes, …
  - Abstraction
- Push the processing to the data, not the data to the processing
  - Decentralized data production
  - Edge analytics
  - Hyper-decentralized data production and storage
- Generator-driven
- …



Modeling History: Simple History for Updates

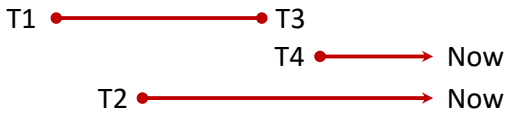
Cid	City	Start	End
C1	London	T1	T3
C1	Leicester	T3	Now
C2	Paris	T2	Now



- No gaps in history
- Only one value for an object on a specific datetime
- Supports following queries:
  - What is the current value – Where End = Now
  - What was the value on a specific datetime - Where date between Start and End

Modeling History: Simple History for Updates with Gaps

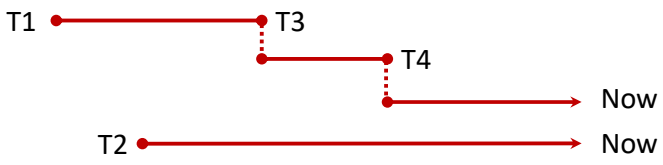
Cid	City	Start	End
C1	London	T1	T3
C1	Leicester	T4	Now
C2	Paris	T2	Now



- No gaps in history
- Only one value for an object on a specific datetime
- Supports following queries:
  - What is the current value – Where End = Now
  - What was the value on a specific datetime - Where date between Start and End - may return no values

Modeling History: Simple History for Updates Without Gaps

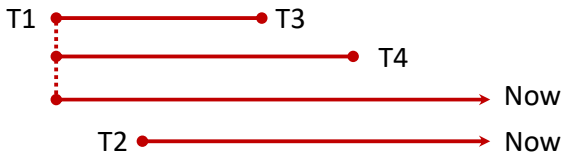
Cid	City	Start	End
C1	London	T1	T3
C1	Gap	T3	T4
C1	Leicester	T4	Now
C2	Paris	T2	Now



- No gaps in history
- Only one value for an object on a specific datetime
- Supports following queries:
  - What is the current value – Where End = Now
  - What was the value on a specific datetime - Where date between Start and End – always returns a value; sometimes nothing

Modeling History: Corrections

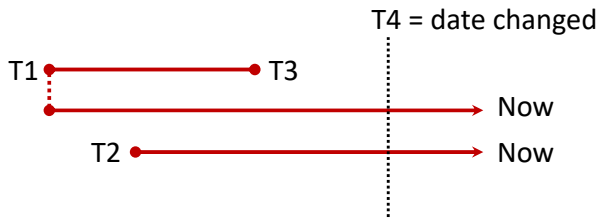
Cid	City	Start	End
C1	Londdon	T1	T3
C1	Londn	T1	T4
C1	London	T1	Now
C2	Paris	T2	Now



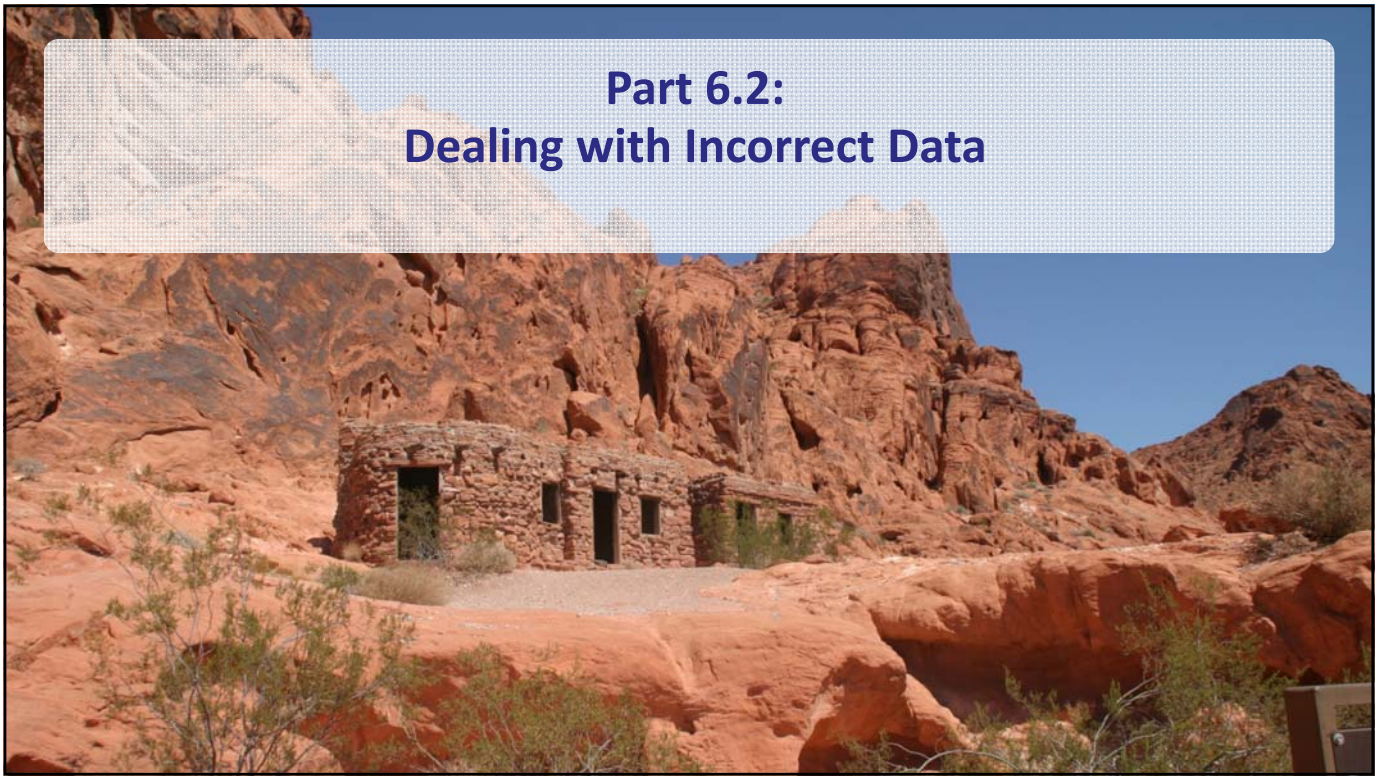
- No gaps in history
- Multiple values for an object on a specific datetime
- Supports following queries:
  - What is the current value – Where End = Now
  - What was the value on a specific datetime – Get the oldest where date between Start and End

Modeling History: Delayed Corrections

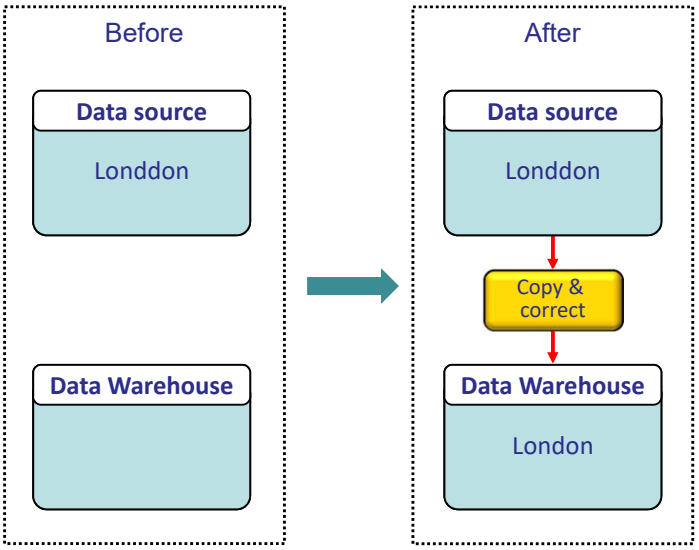
Cid	City	Start	End	Changed
C1	Londdon	T1	T3	T4
C1	London	T1	Now	
C2	Paris	T2	Now	



- Extra column required
- No gaps in history
- Multiple values for an object on a specific datetime
- Supports following queries:
  - What is the current value – Where End = Now
  - What was the value on a specific datetime – Get the oldest where date between Start and End

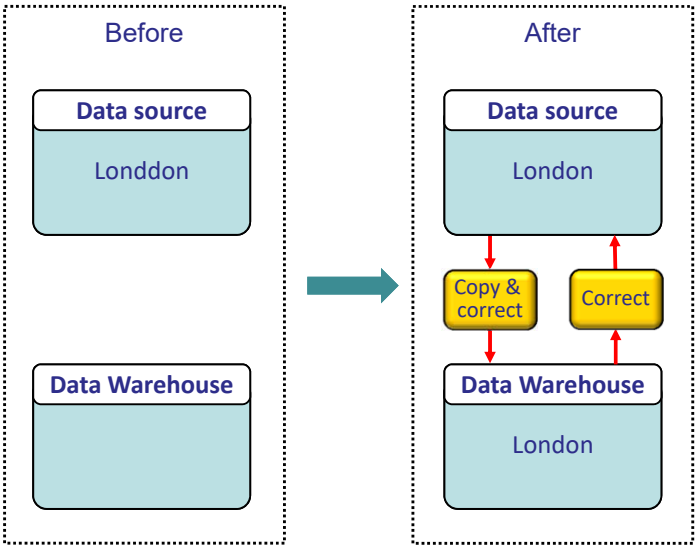


Data Correction Strategies: Simple



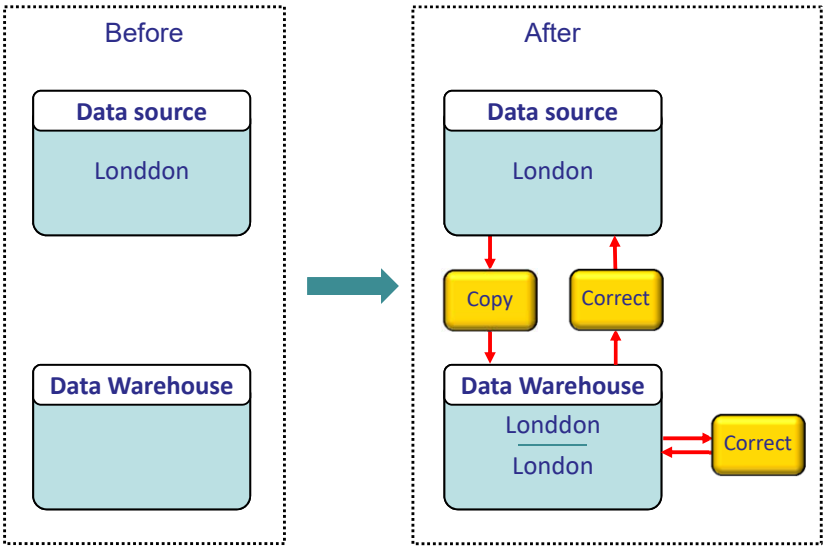
- Restricted to programmable cleansing operations
- Easy to implement
- Source doesn't benefit from cleansing
- Source and data warehouse inconsistent
- No impact on organization
- No time travel supported

Data Correction Strategies: Synchronize



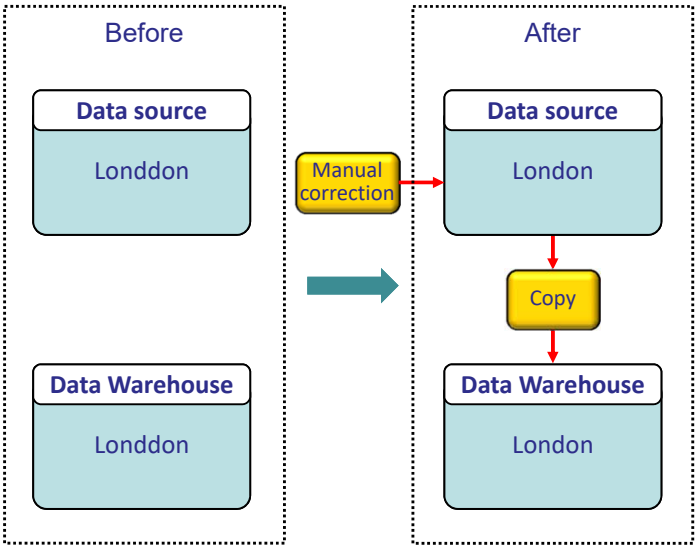
- Manual or automated process?
- Source benefits from cleansing
- Source and data warehouse consistent
- Impact on organization
- No time travel supported

Data Correction Strategies: Time Travel



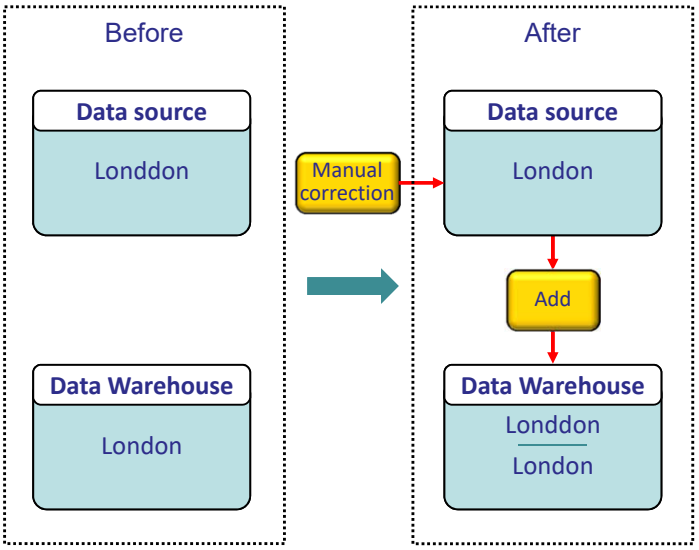
- Restricted to programmable cleansing operations
- Source benefits from cleansing
- Source and data warehouse consistent
- Time travel supported

Data Correction Strategies: Manual Corrections



- User corrections
- Source benefits from cleansing
- Source and data warehouse consistent
- Impact on organization
- No time travel supported

Data Correction Strategies: Manual Corrections + Time Travel



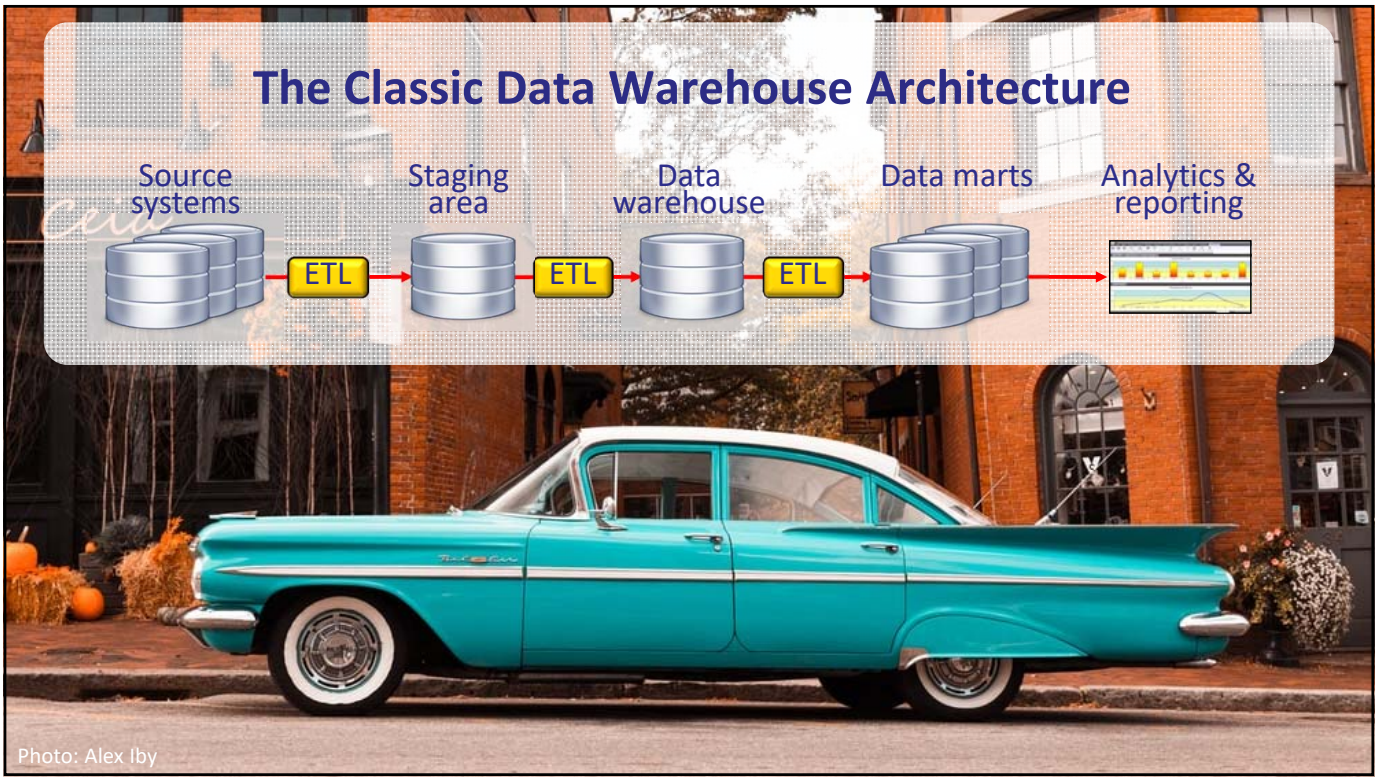
- User corrections
- Source benefits from cleansing
- Source and data warehouse consistent
- Impact on organization
- Time travel supported

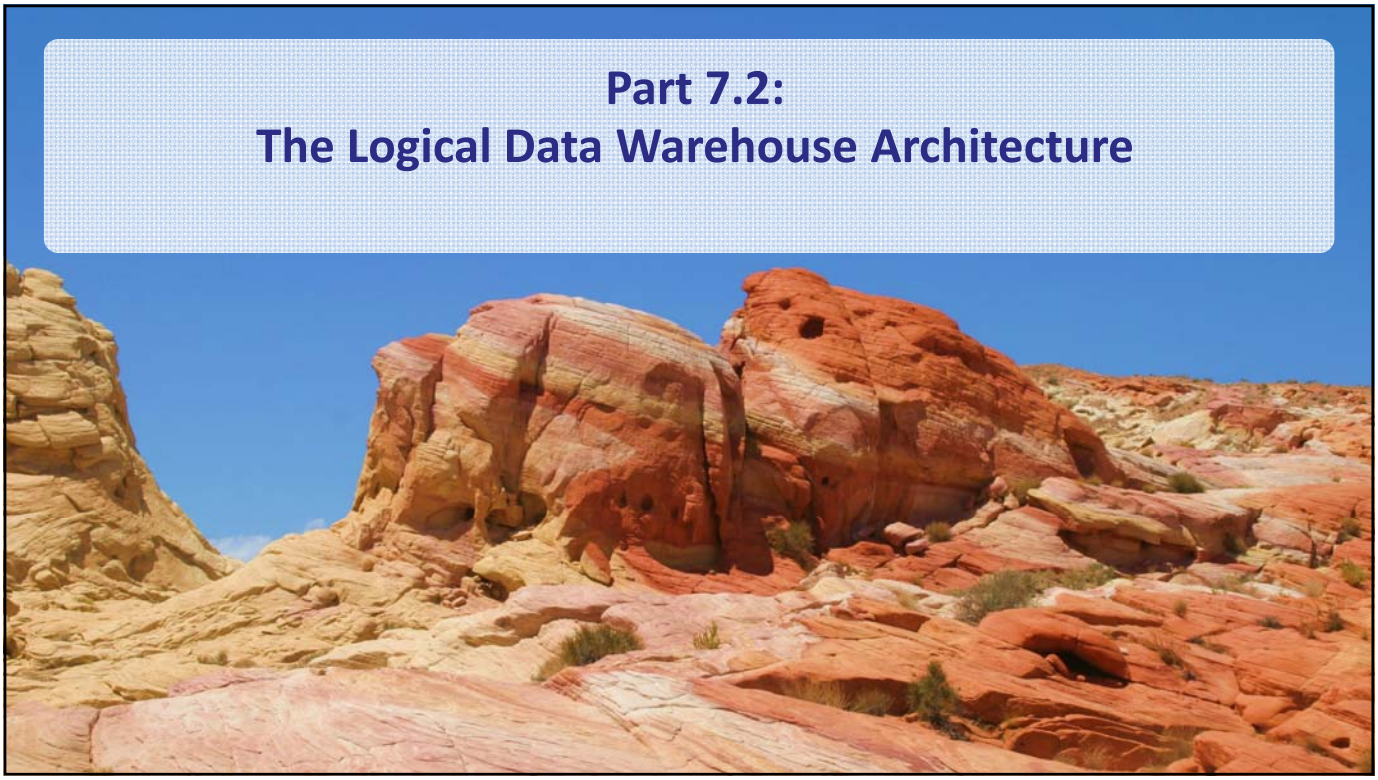
Part 7:
Step 5: Select a Reference Data Architecture

Common Challenges

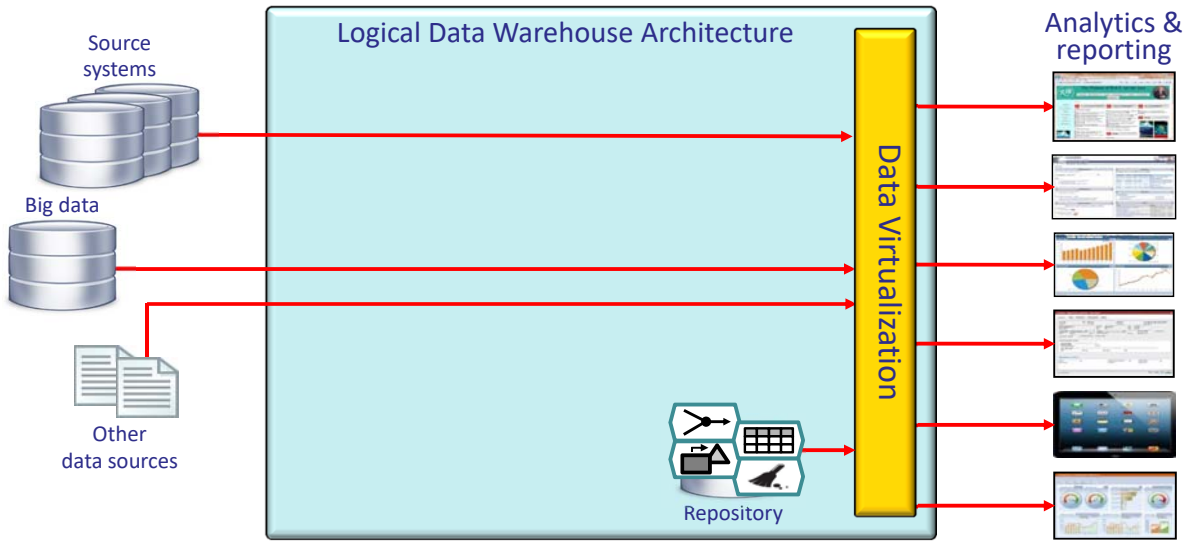
- Source data must be queryable
- Developers and data consumers can't find data easily
- Every insert, update, delete and query should be logged for reconstruction purposes and transparency
- Horizontal and operational lineage
- CRUD interface for real-time synchronization
- Centralized, reusable, versioned business logic
- Proper authorization when integrating data
- …

Part 7.1:
The Classic Data Warehouse Architecture

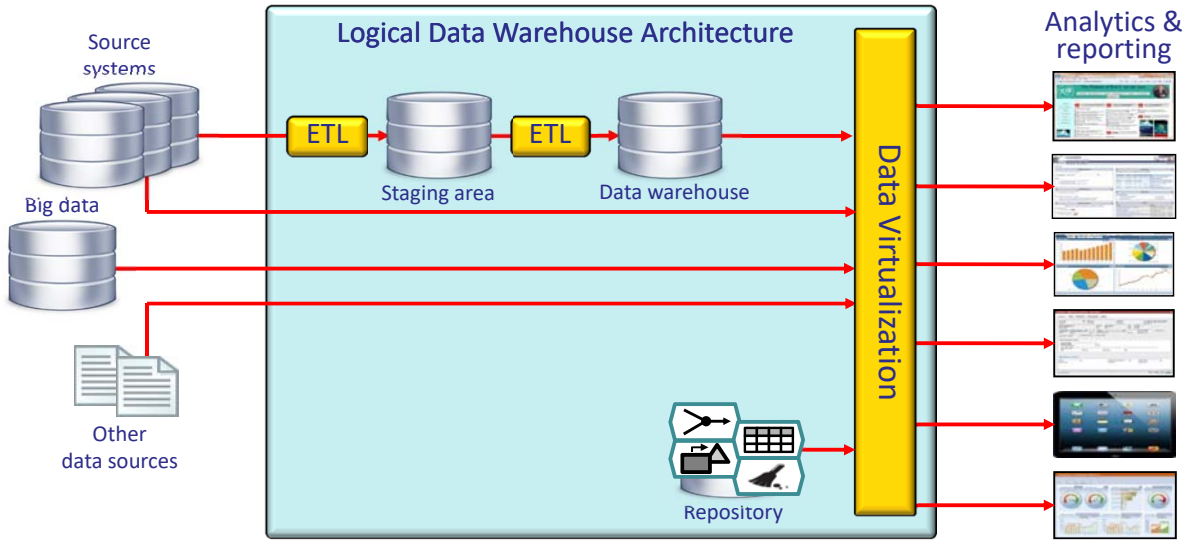

The Classic Data Warehouse Architecture
Source systems
ETL
Staging area
ETL
Data warehouse
ETL
Data marts
Analytics & reporting
Photo: Alex Iby


Part 7.2:
The Logical Data Warehouse Architecture

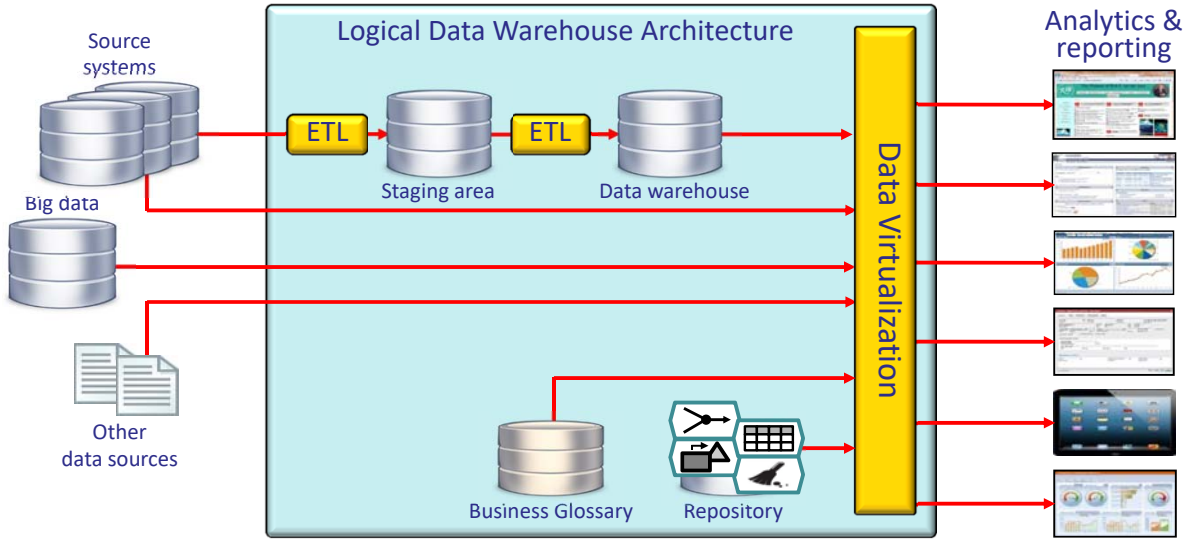
The Logical Data Warehouse Architecture (1)



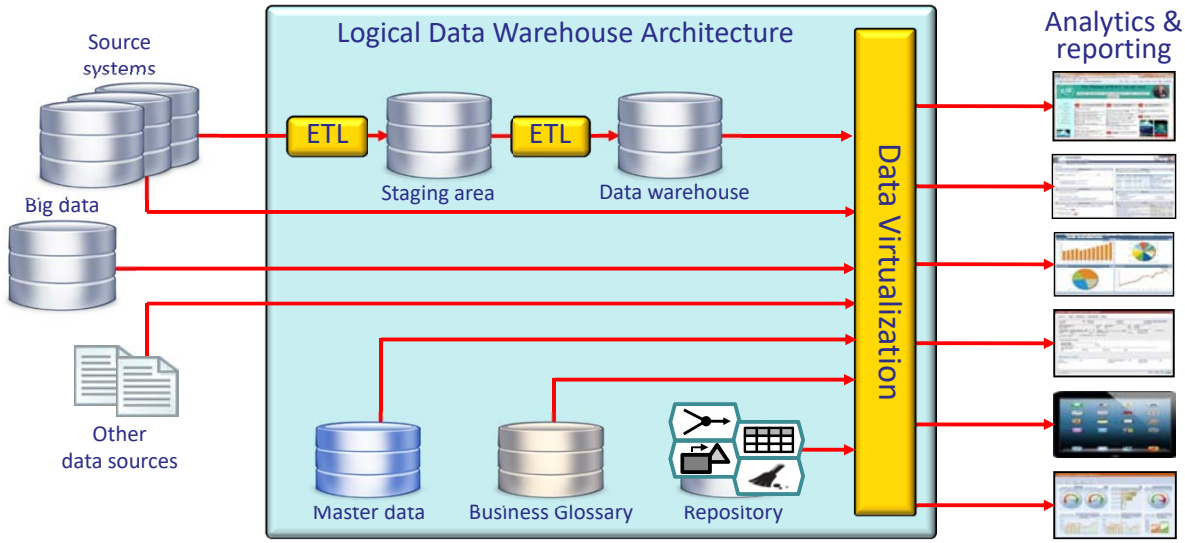
The Logical Data Warehouse Architecture (2)



The Logical Data Warehouse Architecture (3)



The Logical Data Warehouse Architecture (4)



Part 7.3:
The Data Lake

Common Definition of Data Lake

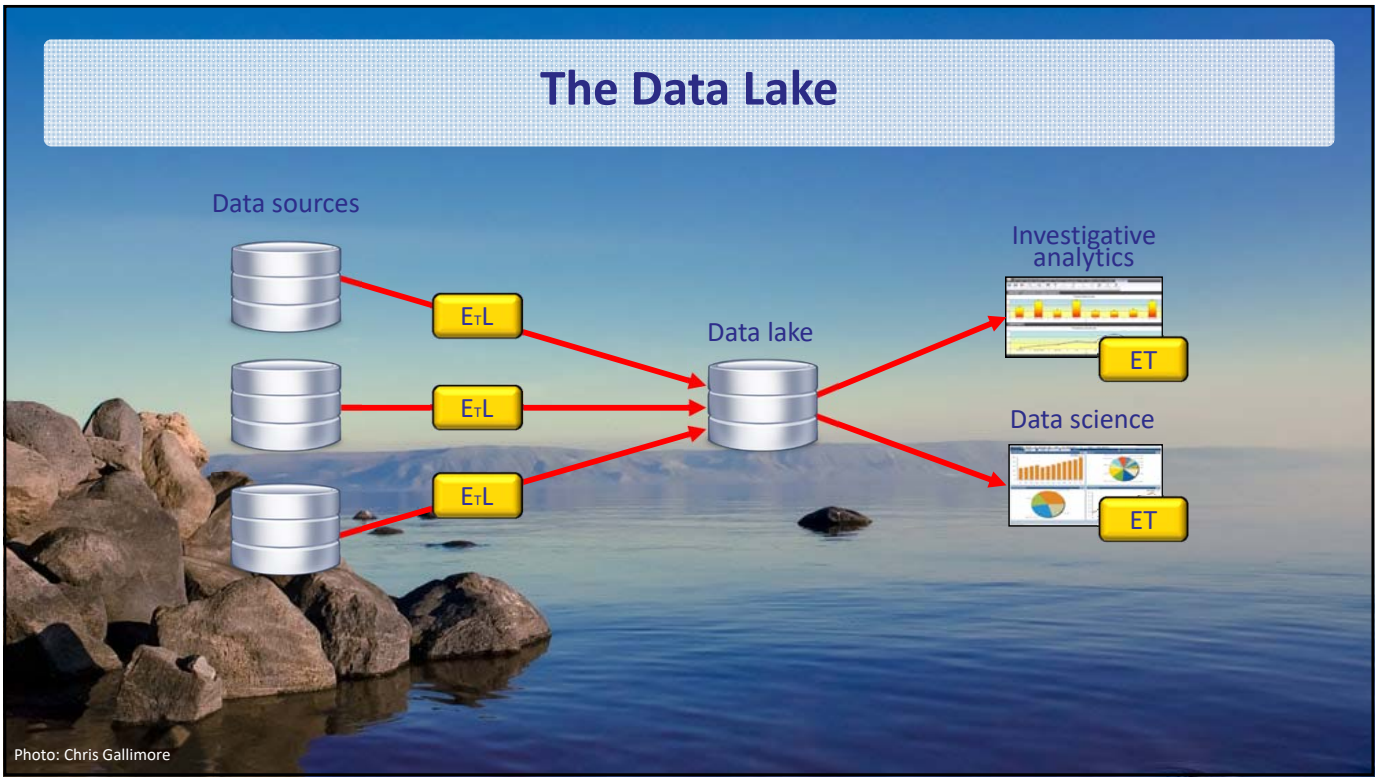
James Serra:

"A "data lake" is a storage repository, usually in Hadoop, that holds a vast amount of raw data in its native format until it is needed. It's a great place for investigating, exploring, experimenting, and refining data, in addition to archiving data."

Photo: Chris Gallimore

Source: http://www.jamesserra.com/archive/2015/04/what-is-a-data-lake/

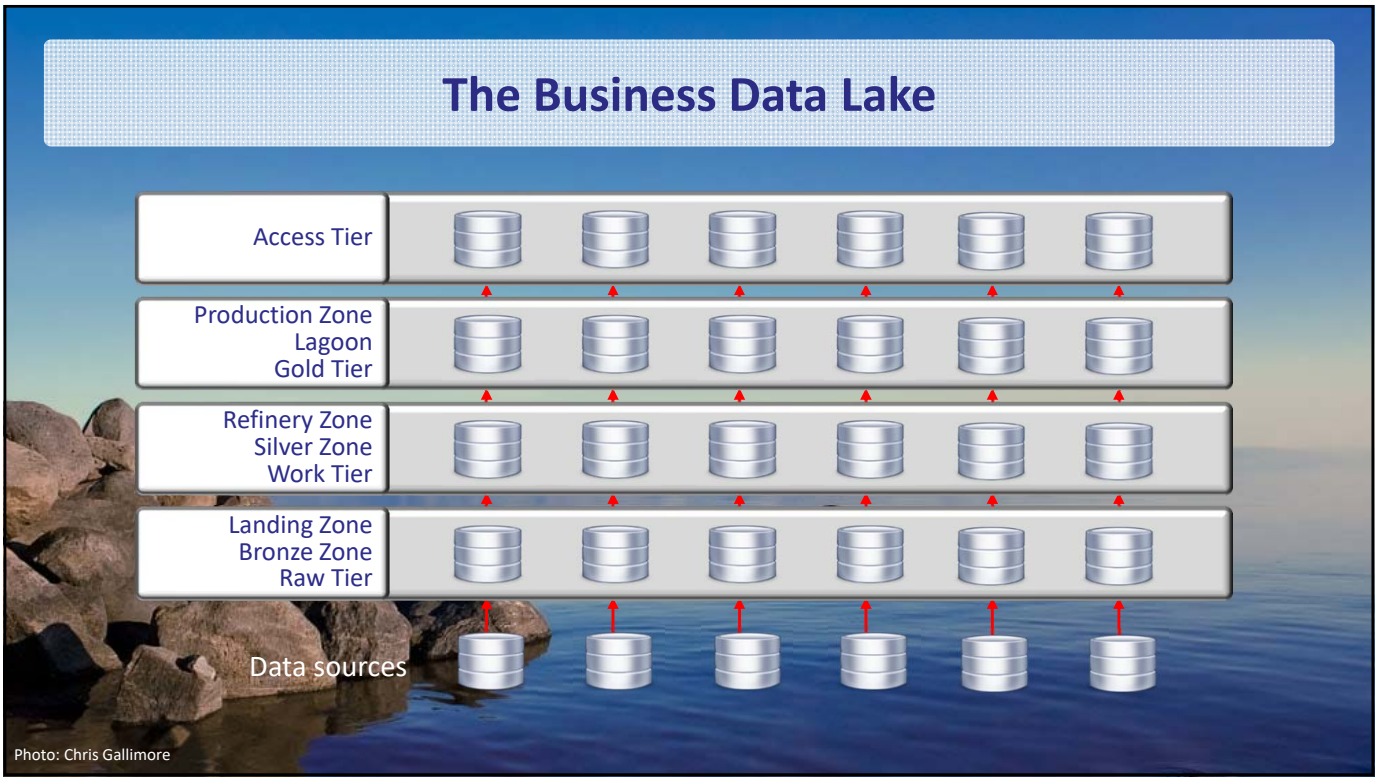
The Data Lake



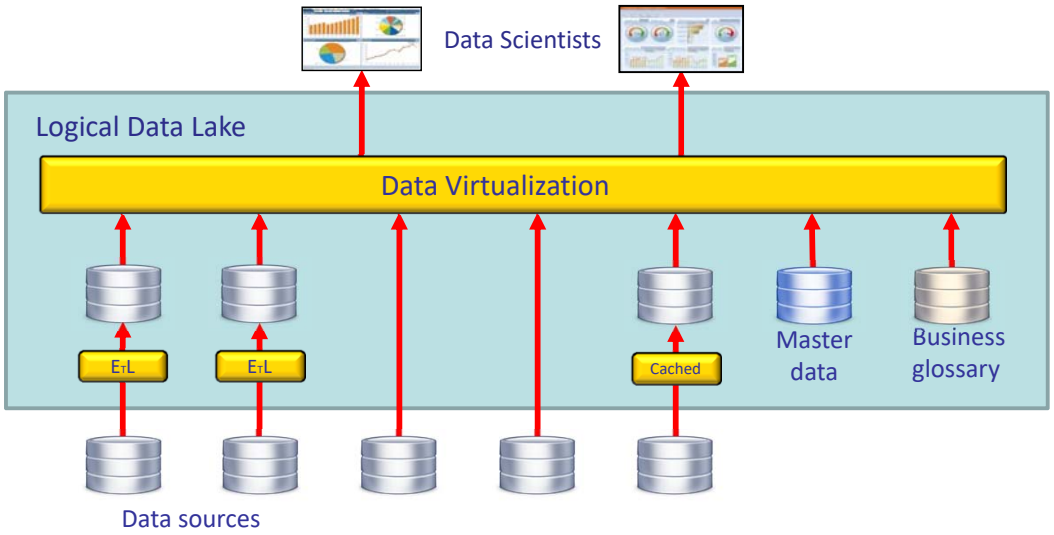
Challenges of a Physical Data Lake

- Big data too big to move
  - Too slow to copy and bandwidth issues
- Complex "T" moved to data consumption
- Company politics
- Data privacy and protection regulations
- Data in data lake is stored outside original security realm
- Metadata to describe data
- Some sources are hard to copy
  - For example, mainframe data
- Refreshing of data lake
- Management of data lake required
- …

The Business Data Lake



The Logical (Virtual) Data Lake

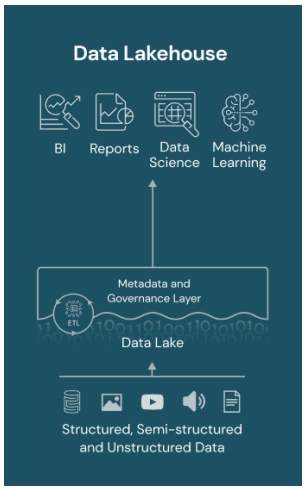


Part 7.4:
The Data Lakehouse

Definitions of Data Lakehouse

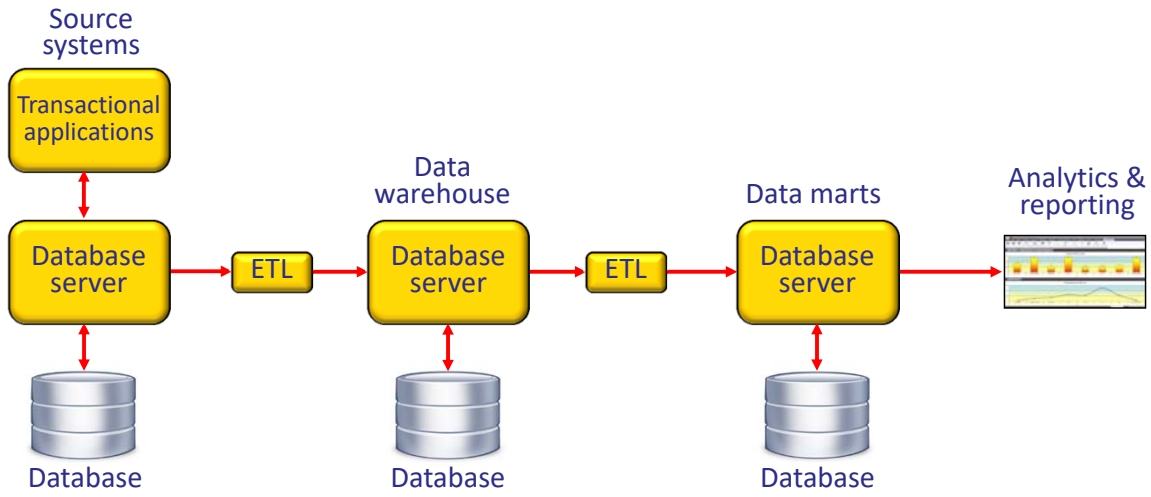
- DataBricks: "A data lakehouse is a […] open data management architecture that combines the flexibility, cost-efficiency, and scale of data lakes with the data management and ACID transactions of data warehouses, enabling business intelligence (BI) and machine learning (ML) on all data."
- Striim, John Kutay: "A data lakehouse is a new, big-data storage architecture that combines the best features of both data warehouses and data lakes. A data lakehouse enables a single repository for all your data (structured, semi-structured, and unstructured) while enabling best-in-class machine learning, business intelligence, and streaming capabilities."
- Dremio, Deepa Sankar: "A [data] lakehouse has the performance and optimization of a data warehouse combined with the flexibility of a data lake."
- Wikipedia: "Databricks develops and sells a cloud data platform using the marketing term lakehouse, a portmanteau based on the terms data warehouse and data lake."

Key Characteristics of a Data Lakehouse

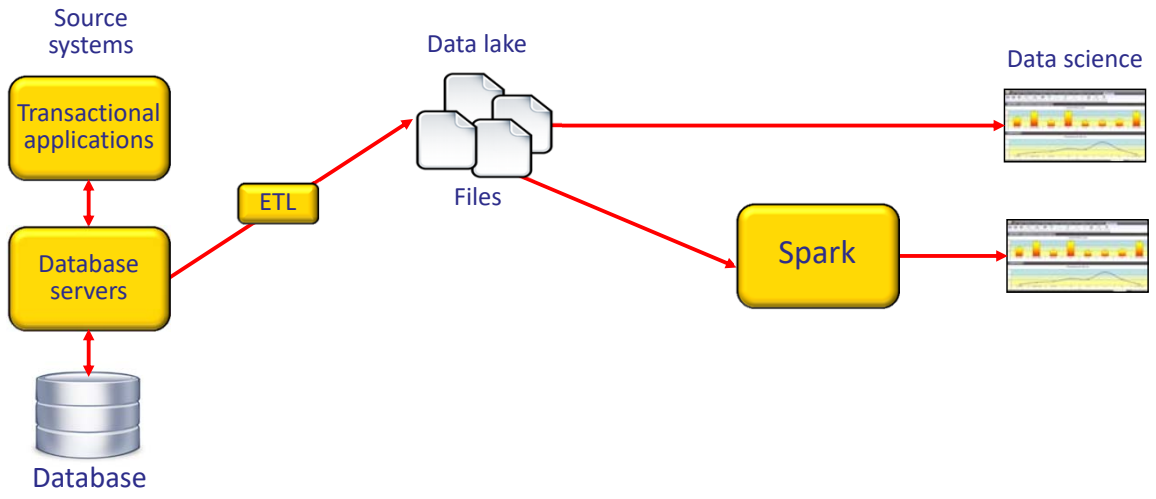


- Two use cases: BI and data science
- Data is stored once
- Supports structured and unstructured data
- Schema enforcement
- Open file formats
- Low-cost data storage
- ACID compliant

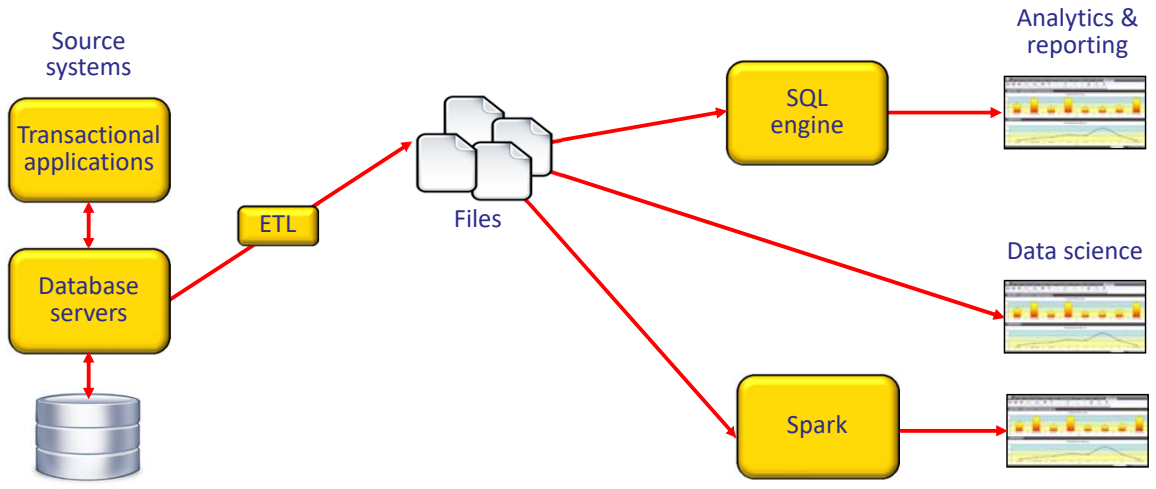
The Data Warehouse Architecture in More Detail



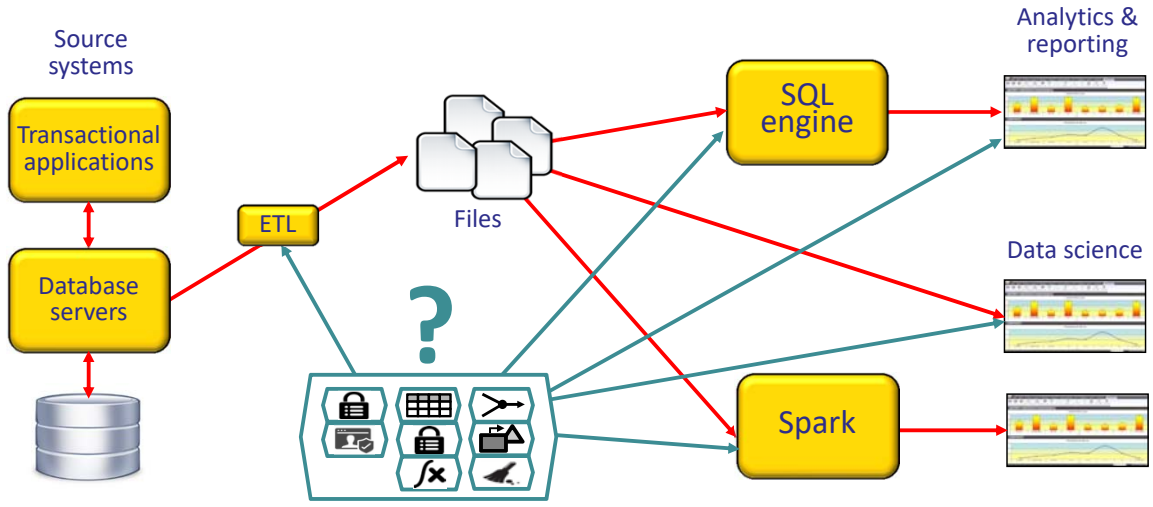
The Data Lake Architecture in More Detail



The Data Lakehouse Architecture in More Detail

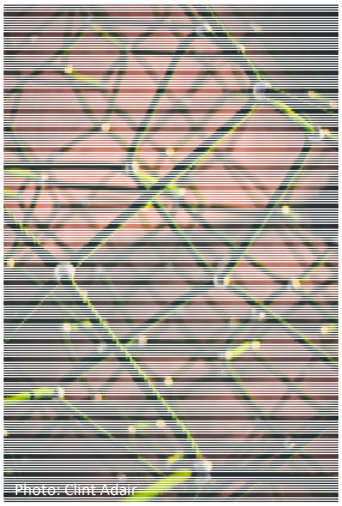


Where to Implement Transformers?



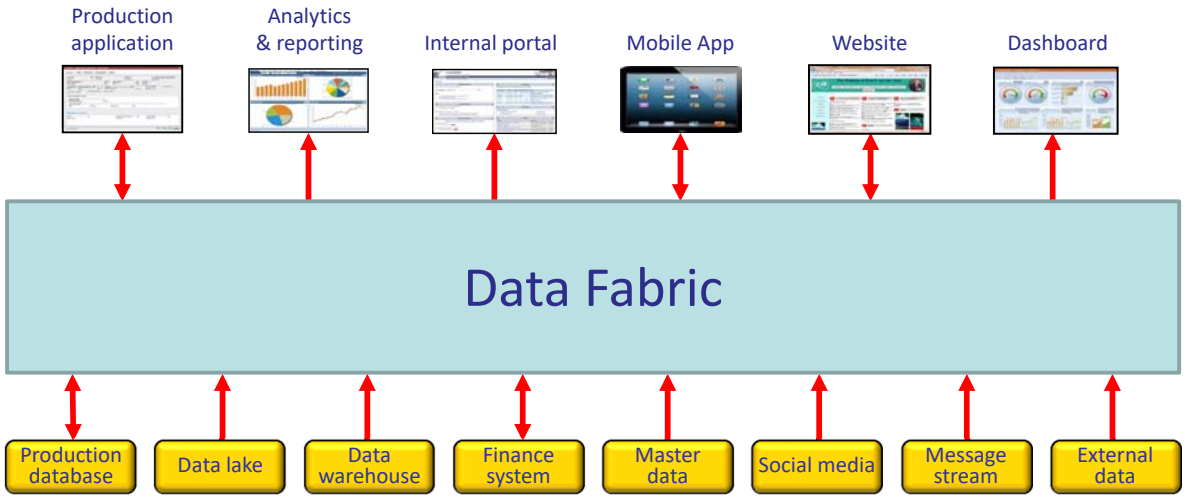
Part 7.5: The Data Fabric

What is the Data Fabric?

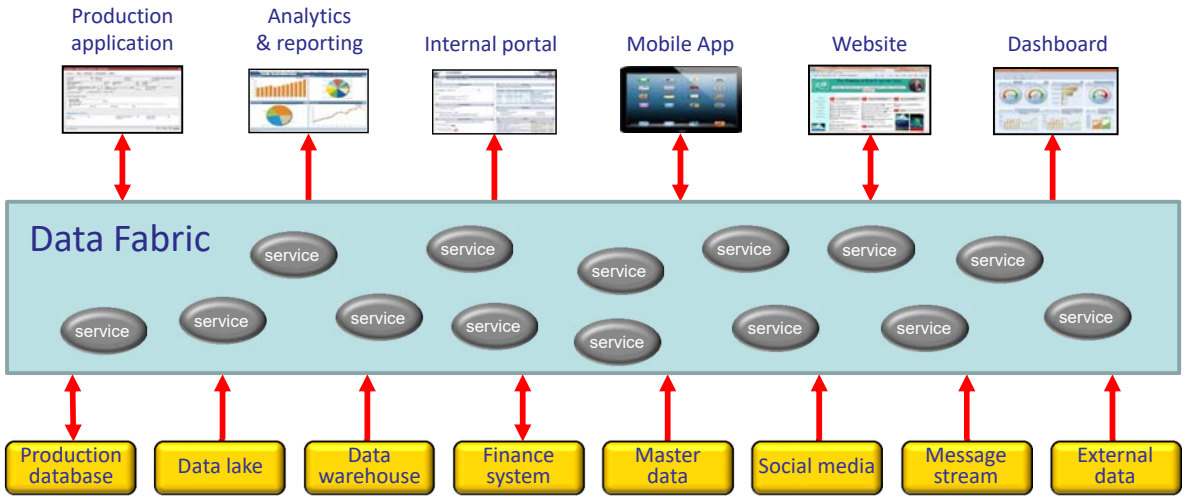


- Gartner: A data fabric is generally a custom-made design that provides reusable data services, pipelines, semantic tiers or APIs via combination of data integration approaches in an orchestrated fashion.
- Gartner: Data fabric enables *frictionless access* and sharing of data in a distributed data environment. It enables a *single* and *consistent* data management framework, which allows seamless data access and processing by design across otherwise siloed storage.

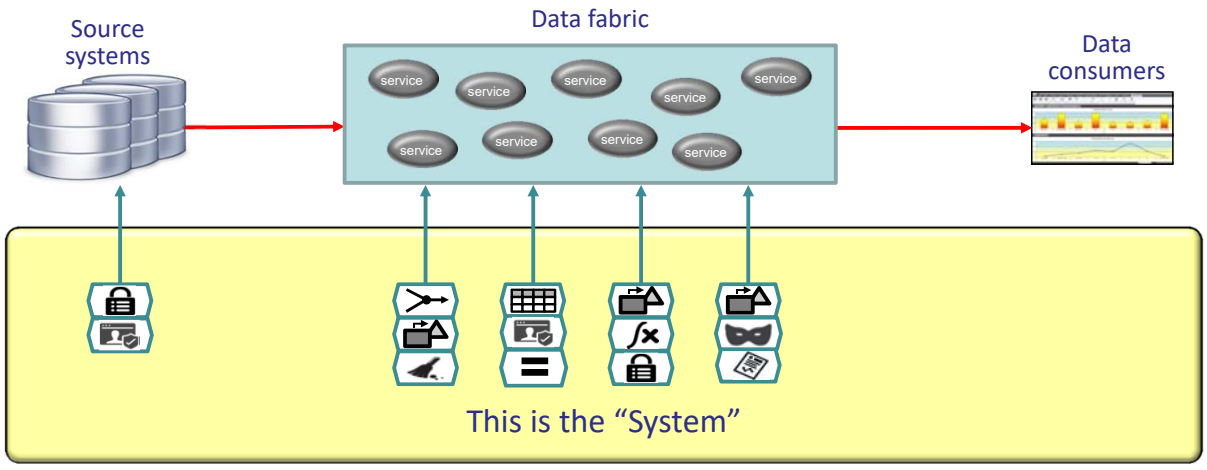
The Data Fabric



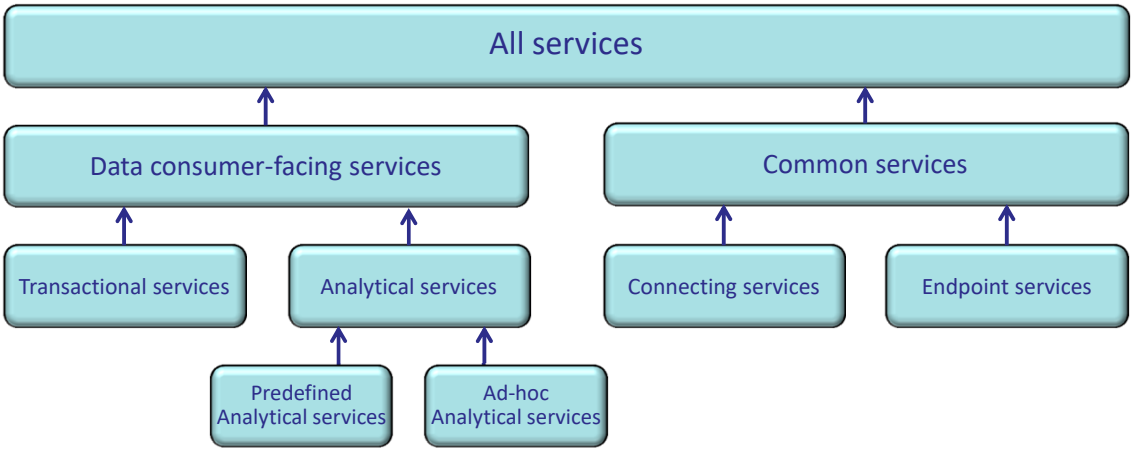
The Services of a Data Fabric



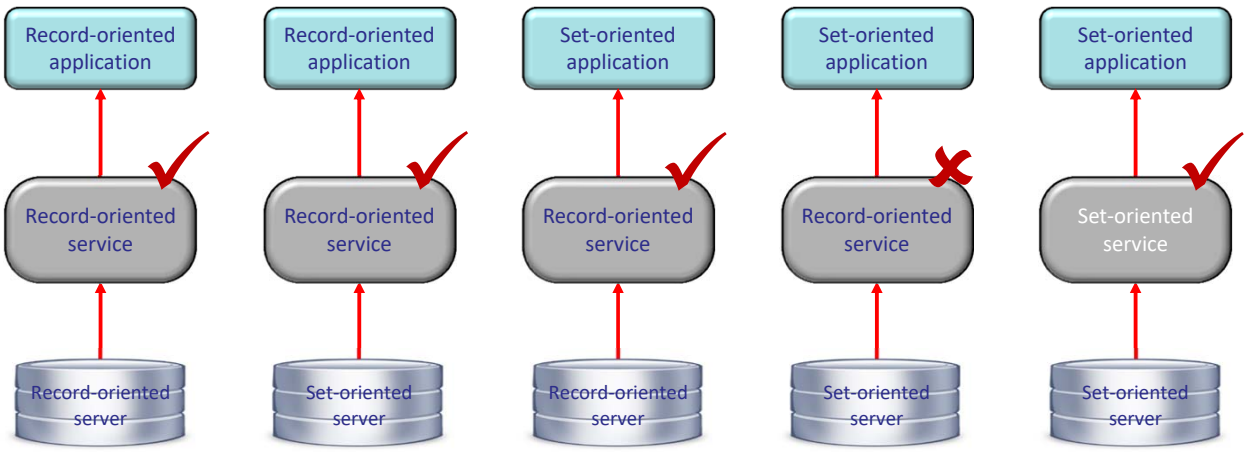
Data Fabrics and Transformers



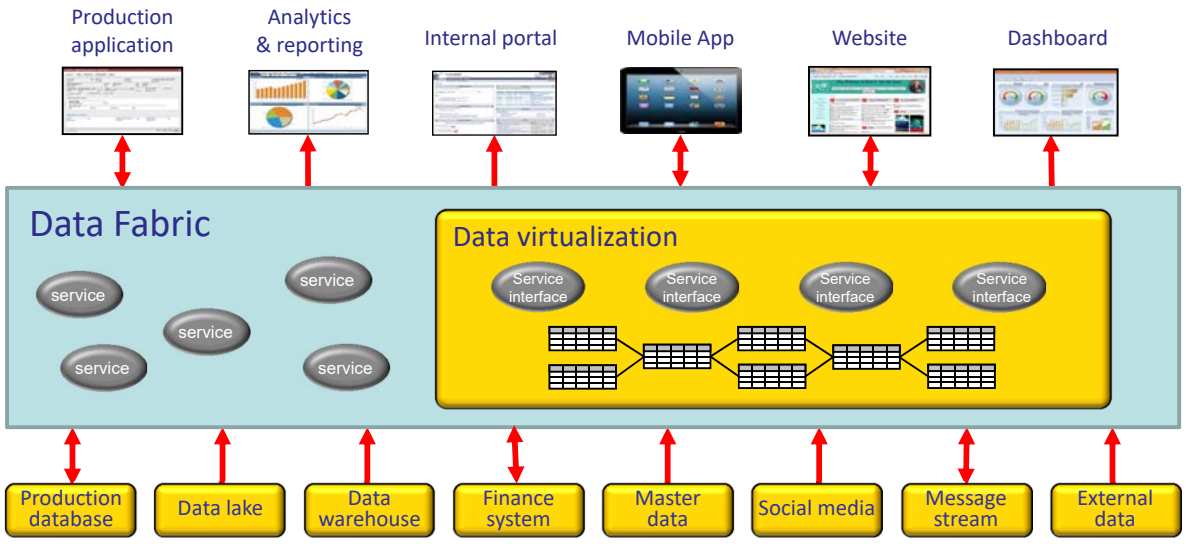
A Data Fabric Consists of Different Types of Services



Record-Oriented or Set-Oriented Interface?



The Data Fabric

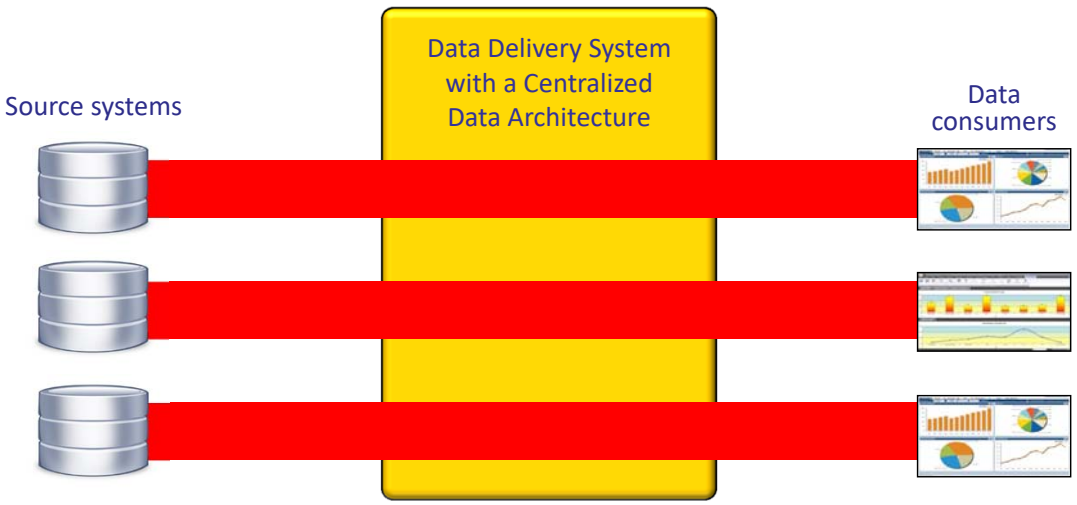


Part 7.6: The Data Mesh

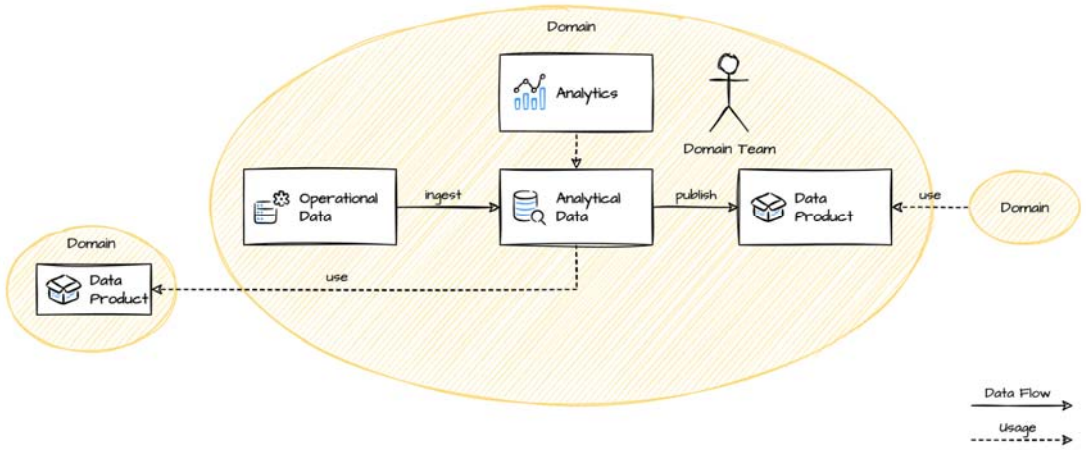
What is a Data Mesh?

- Introduced by Zhamak Dehghani:
- "Data platforms based on the data lake architecture have common failure modes that lead to *unfulfilled promises* at scale.
- To address these failure modes we need to shift from the *centralized paradigm* of a lake, or its predecessor data warehouse.
- We need to shift to a paradigm that draws from *modern distributed architecture*: considering *domains* as the first class concern, applying platform thinking to create self-serve data infrastructure, and *treating data as a product*."

Single-Domain Data Consumers



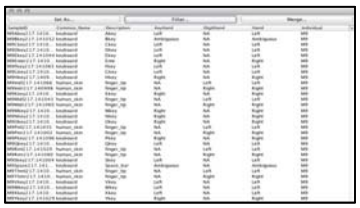
A Domain-Oriented Architecture



datamesh-architecture.com

Potential Data Products

Data as file



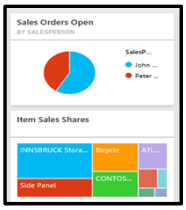
Report

Service

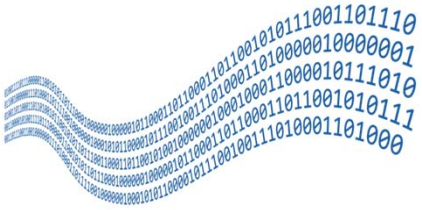
Data via SQL

Apps

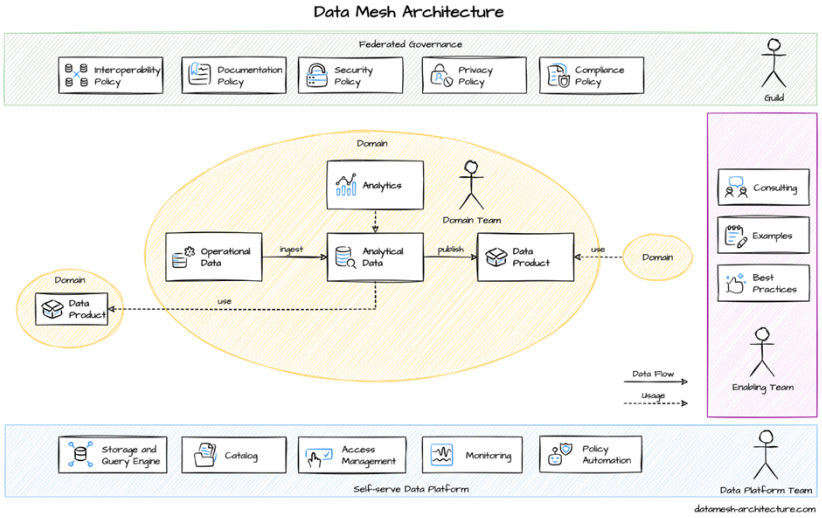
Embeddable KPI



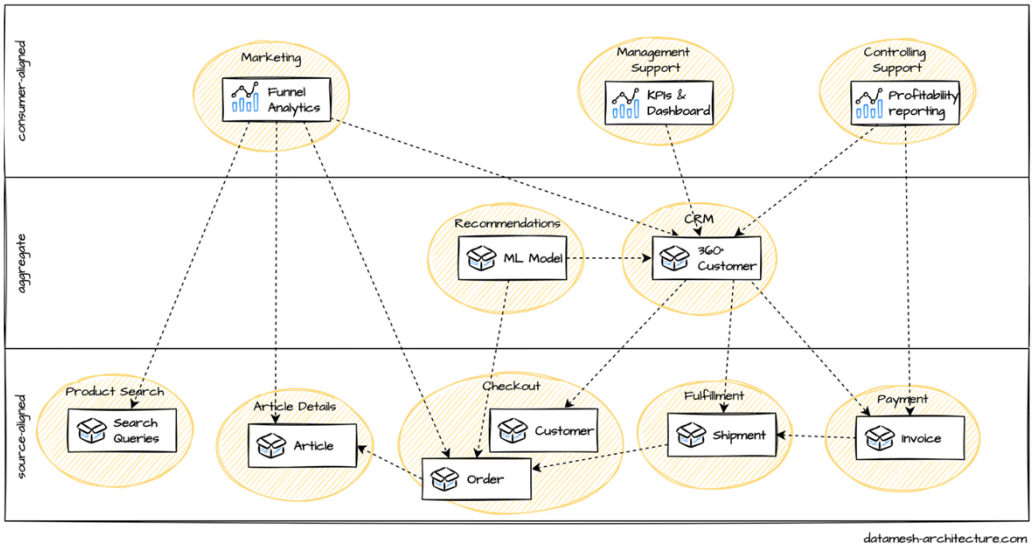
Stream of Data



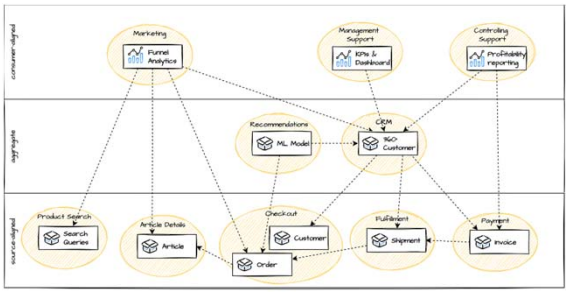
The Periphery of a Domain

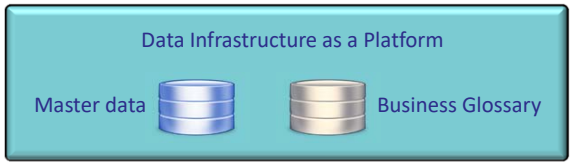


The Mesh Itself



The Foundation: Data Infrastructure as a Platform

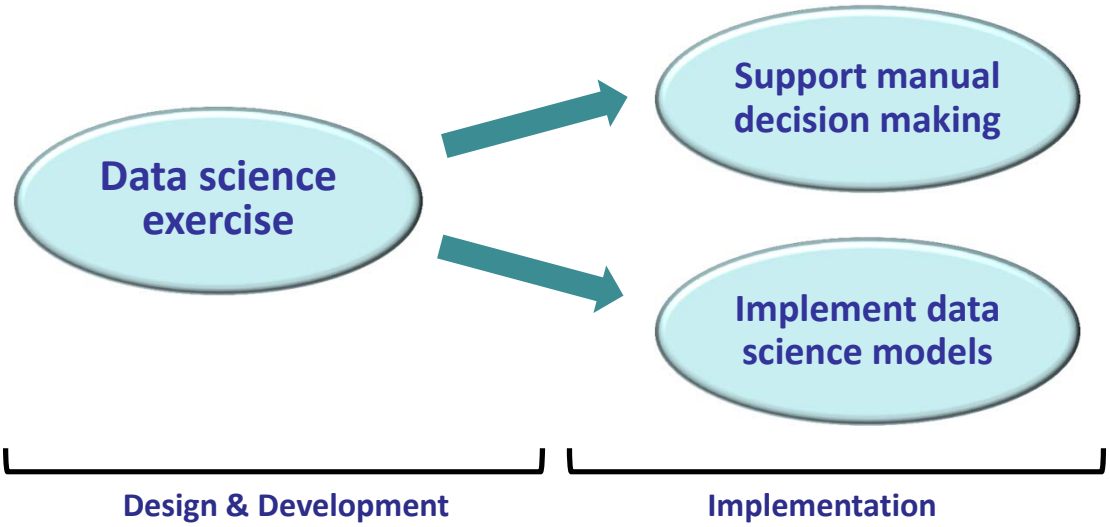




- Scalable polyglot big data storage
- Encryption for data at rest and in motion
- Data product versioning
- Data product schema
- Data product de-identification
- Unified data access control and logging
- Data pipeline implementation and orchestration
- Data product discovery, catalog registration and publishing
- Data governance and standardization
- Data product lineage
- Data product monitoring/alerting/log
- Data product quality metrics (collection and sharing)
- In memory data caching
- Federated identity management
- Compute and data locality
- …

Part 7.7: Embedding Data Science Models in Data Architectures

Operationalization of Data Science Models



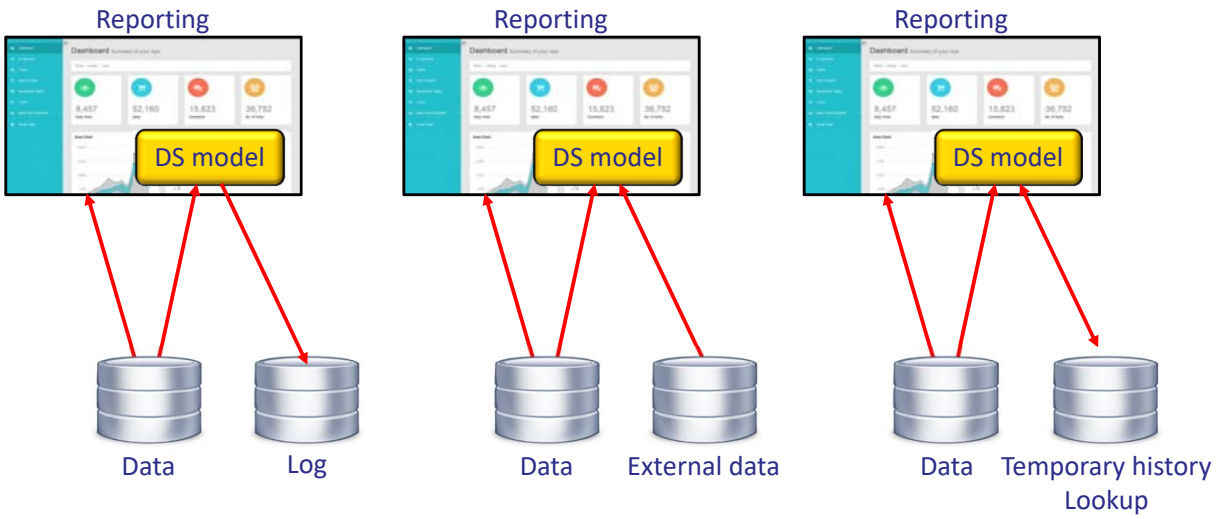
Operationalization of Data Science Models

Type of Decision	Example
Singular manual decision	What will be the impact on total sales of acquiring company X?
Repeatable manual decision	Does a specific location have the right characteristics for opening a new shop?
Partial automated decision	In a call center: What is the churn risk for a customer? What should we offer?
Full automated decision without automated reaction	When credit card payment is dubious, send message to operator
Full automated decision with automated reaction	When sensor indicates the component is heating up too fast, switch off machine
…	…

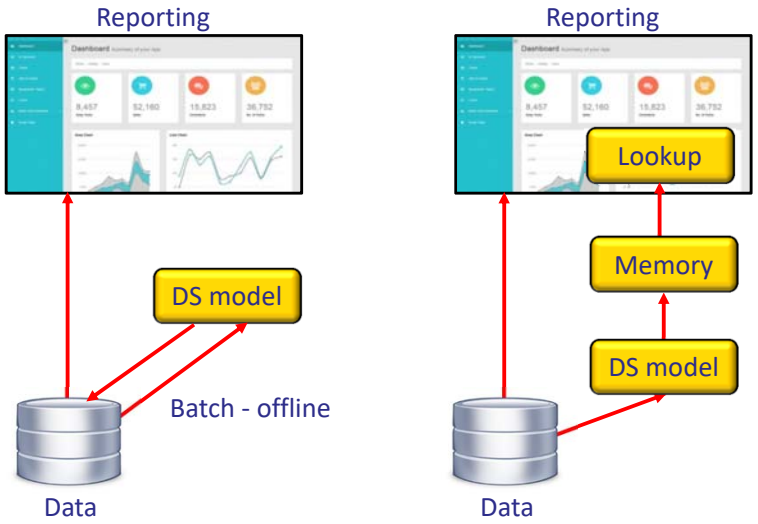
Requirements for a Supporting Data Architecture

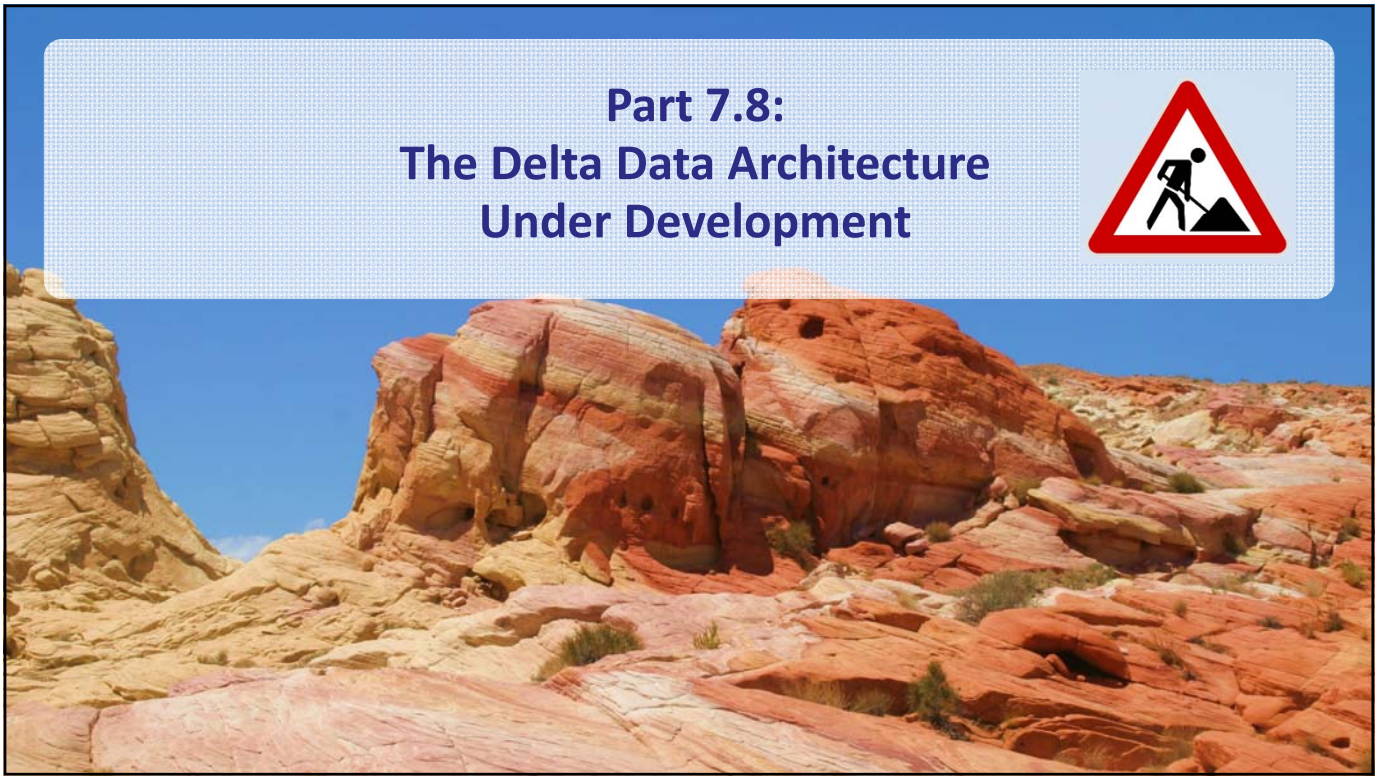
- Versioning of data science models
  - Immutable models
- Auditable data science models
  - Reproducible data for reproducible models
  - Transparency of models
- Different codings must be easy and quick to apply
- Self-learning models or not?
- Delivering metadata
  - Descriptions, definitions, tags, relationships, searchable
- Fast evaluation of models
  - Max time to execute model, SLAs
- And many more …

Architectural Aspects (1)

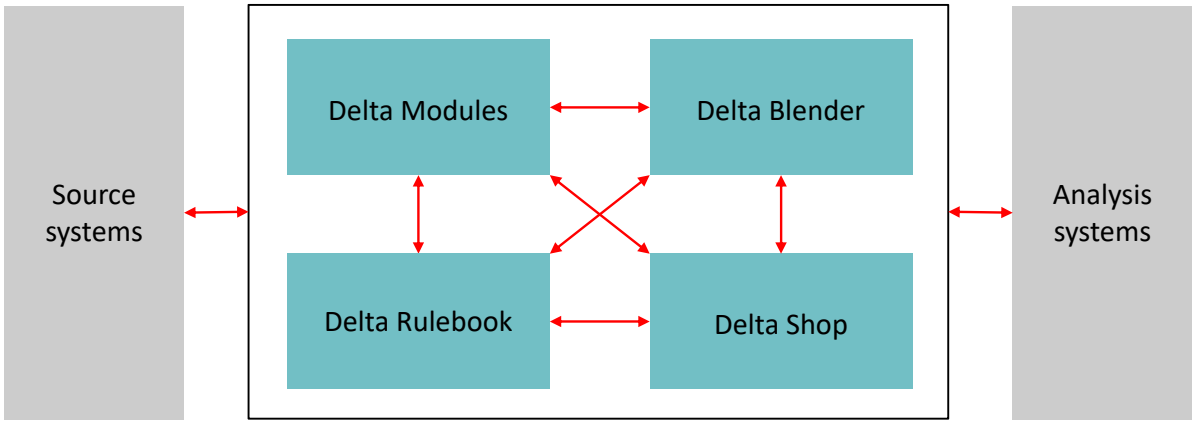


Architectural Aspects (2)





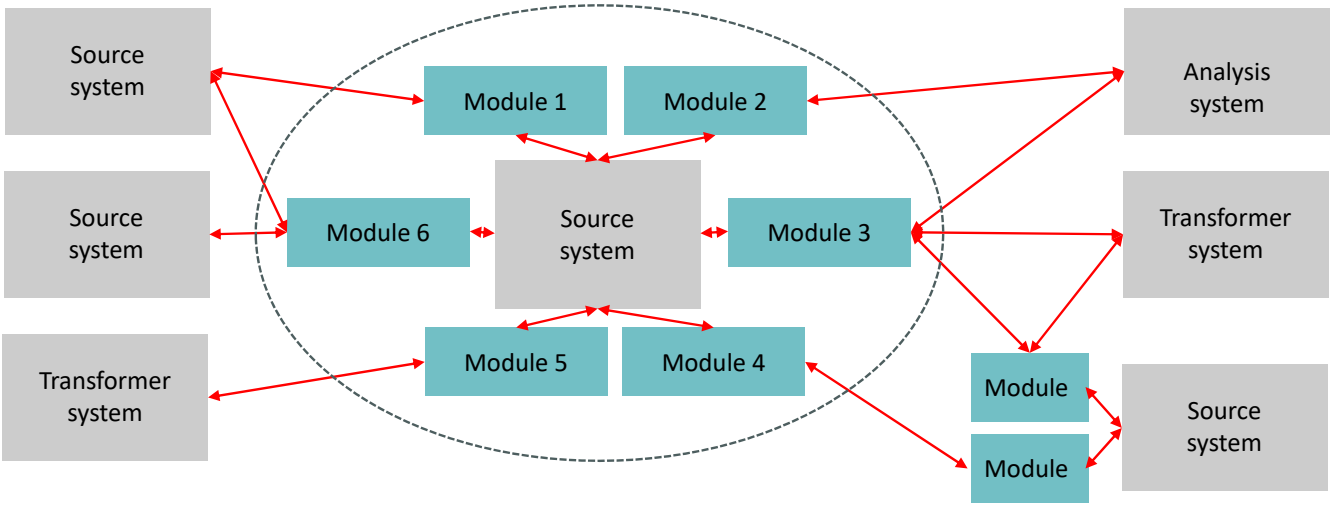
Delta: A Modern, Enterprise Data Architecture



Functions of Source Systems

- Entering, modifying, and deleting data
  - For many business objects
- Retrieving data
- Securing data
- Data quality
- Backup and recovery tasks
- Calculating results
- Handling workflow and processes
- Triggering events
- *Connections with other systems*
- …

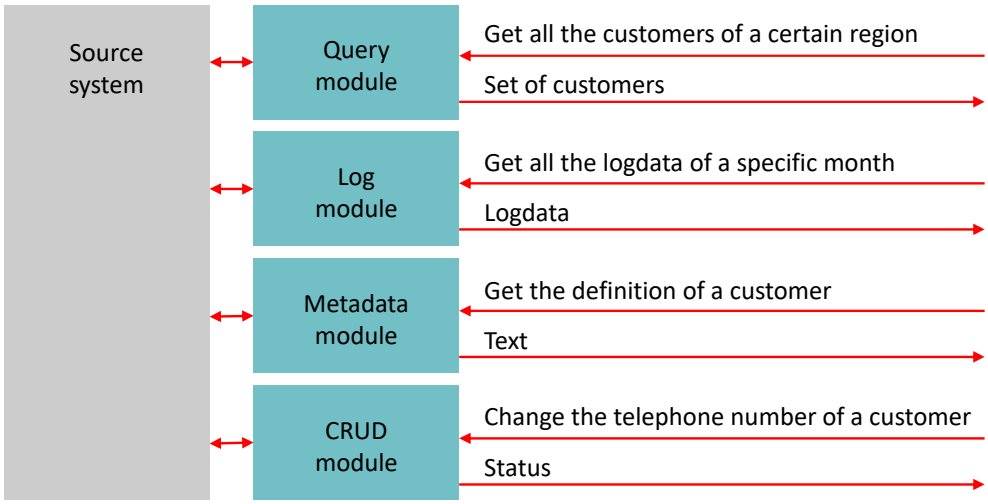
Wrapping the Source Systems - Abstraction



Six Delta Modules to Wrap Source Systems

- CRUD
- Query
- Metadata
- Log
- Messaging
- Data security

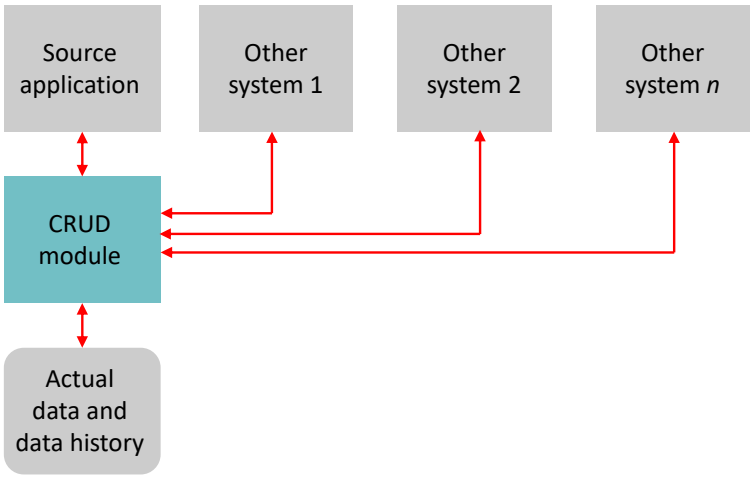
Modules and Functions



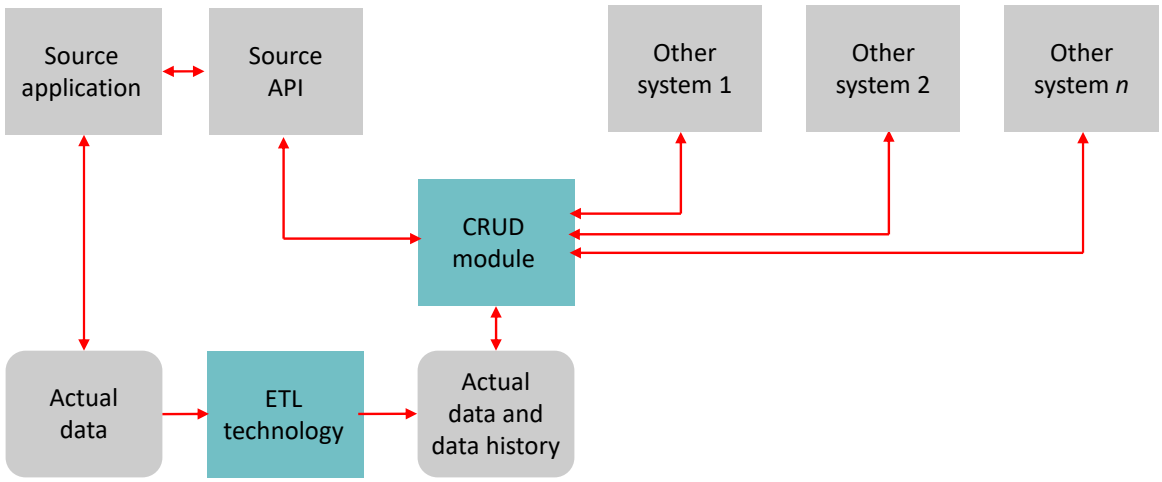
Functions of CRUD modules

- Offer functions to enter, view, modify, and delete data
- Hide internal and external source systems and files
- Data complies with the Enterprise Data Model
- Support time travel
  - Maintains data history if the source system does not
  - Two time dimensions: transaction time and real time
- Work with individual business objects
- Enforce data quality rules
- Involved in source system synchronization
- Replaces data exchange with data sharing
- …

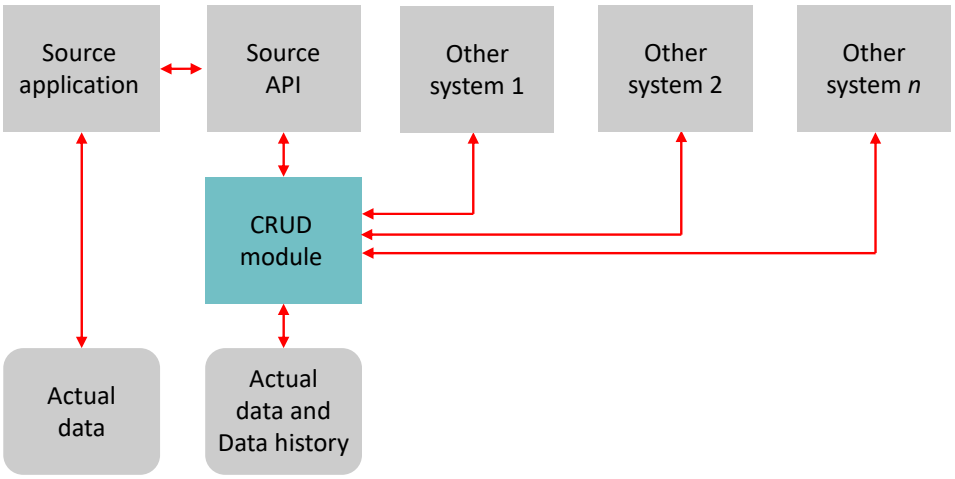
CRUD Module (1) The Dream



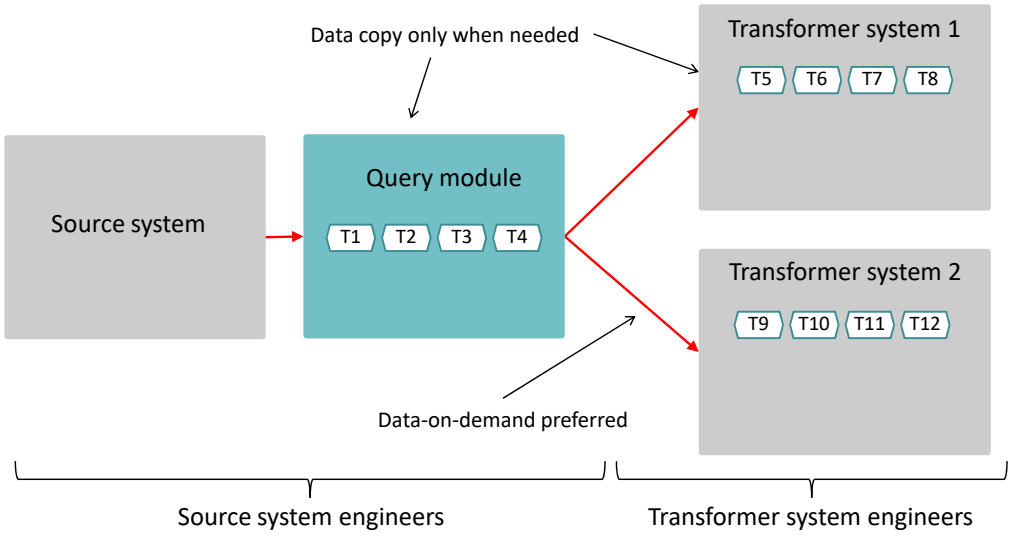
CRUD Module (2) Via API of Source System and ETL



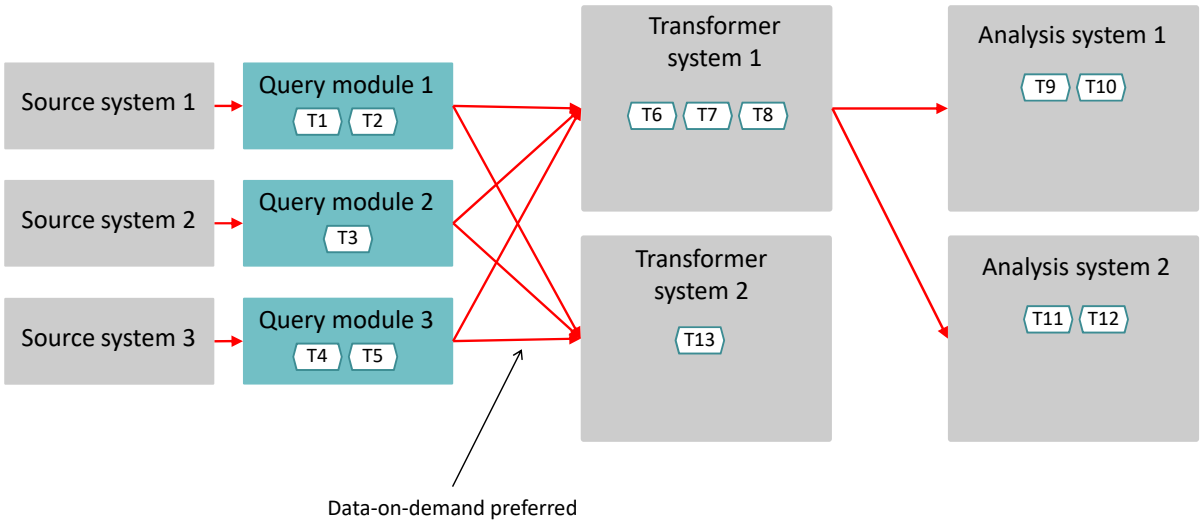
CRUD Module (3) Via API of Source System



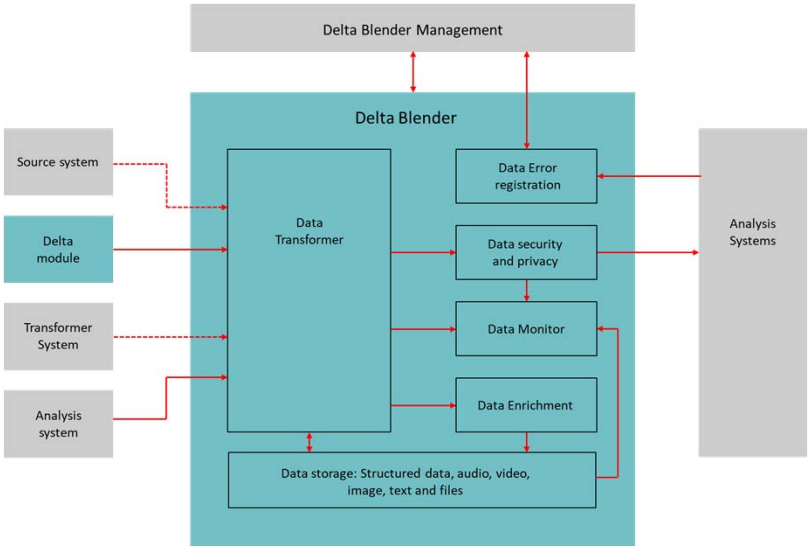
Transformers Closer to the Source Systems



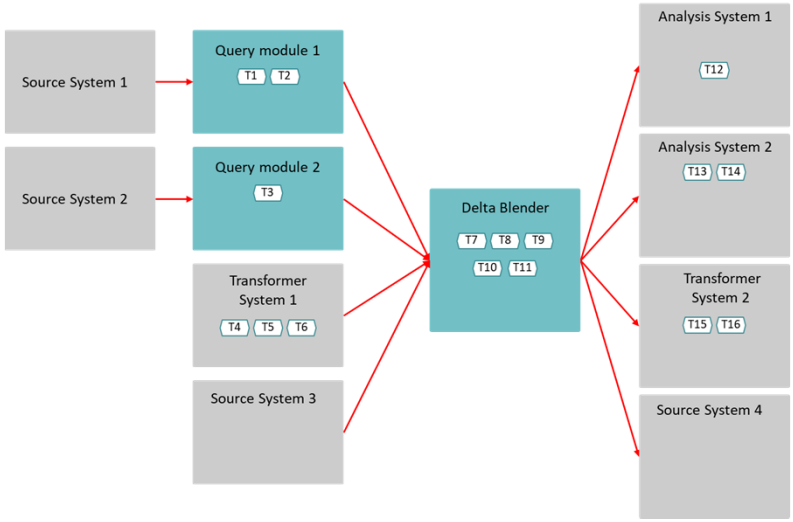
Transforming Data in Query Modules



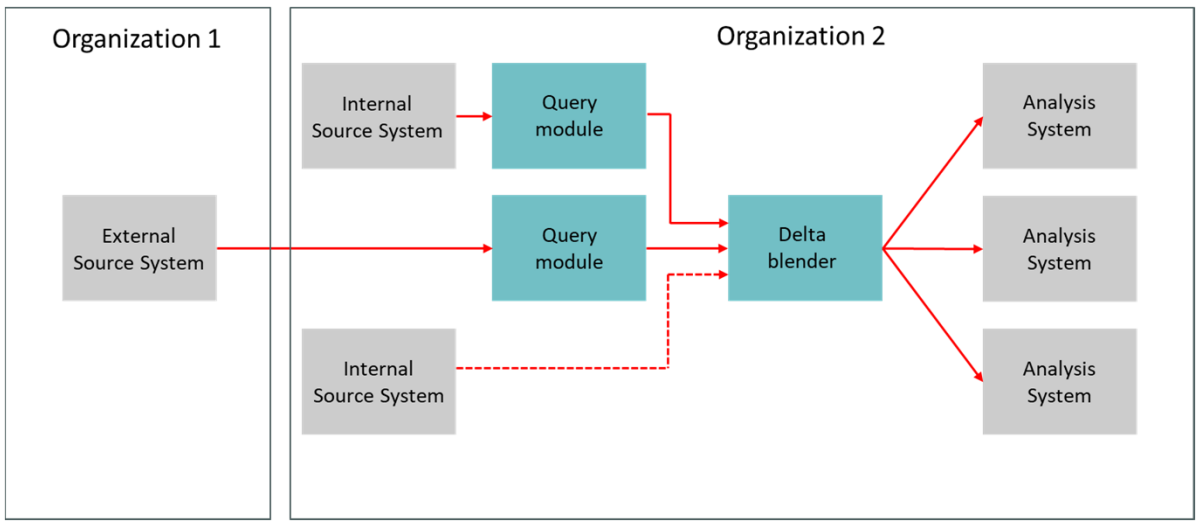
Delta Blender: Produces Ready-to-use Data



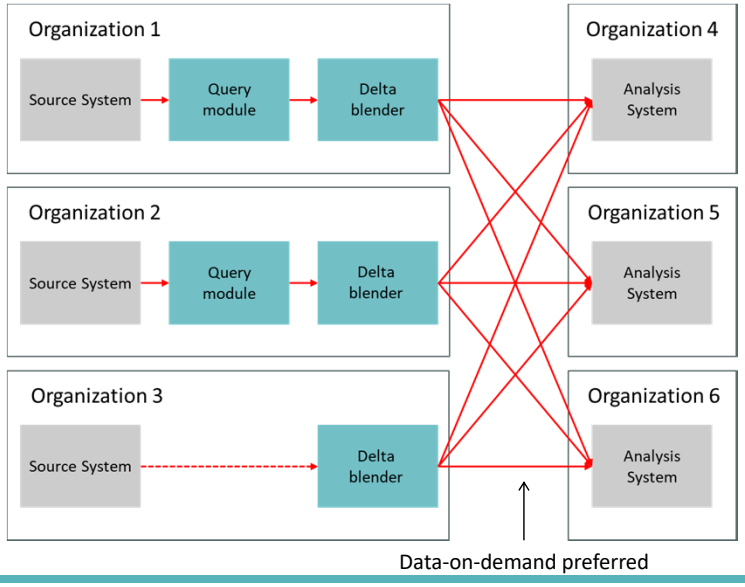
Delta Blender: Transformers



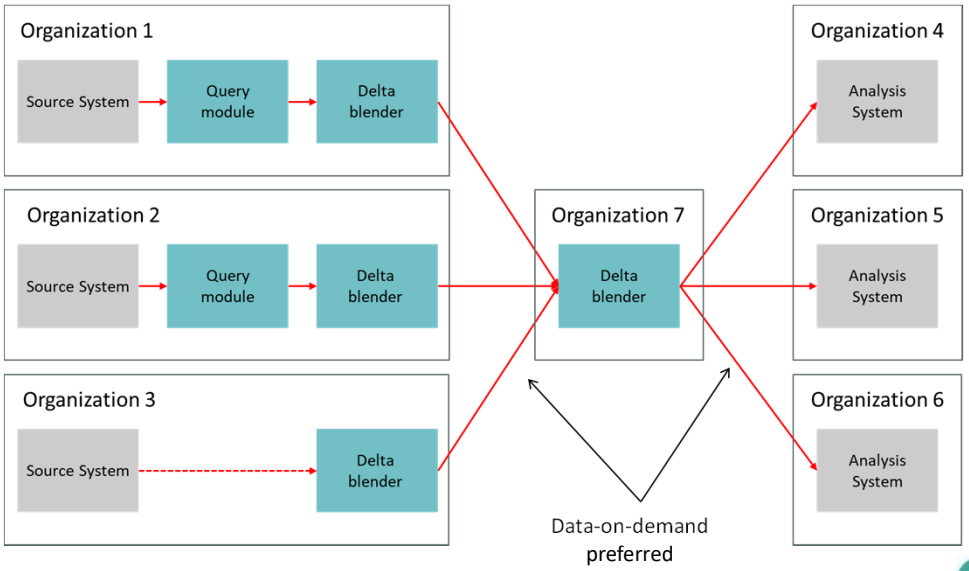
Local Federated Data Architecture



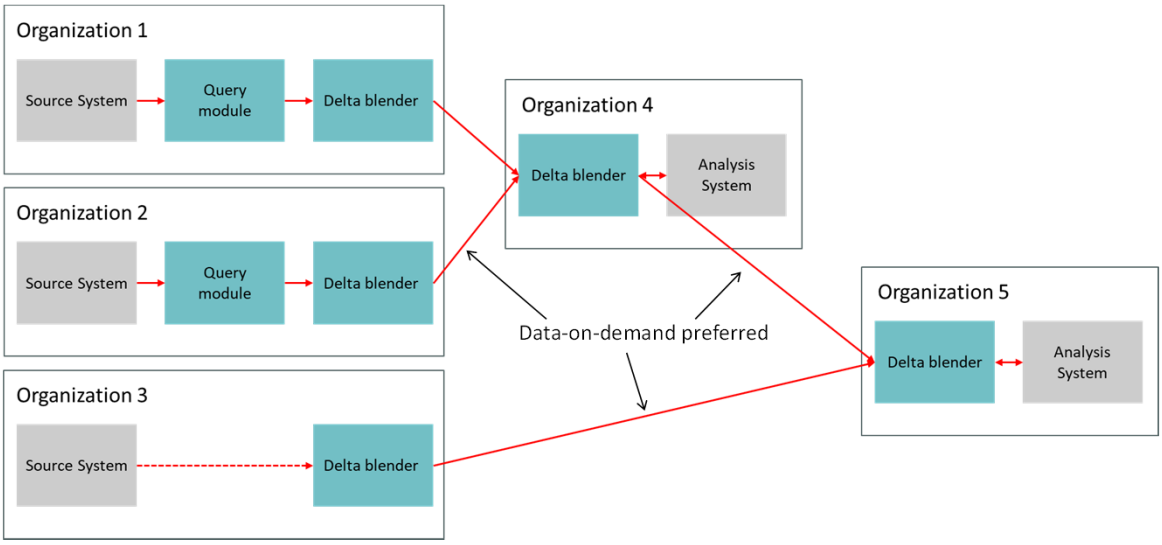
No Federated Data Architecture



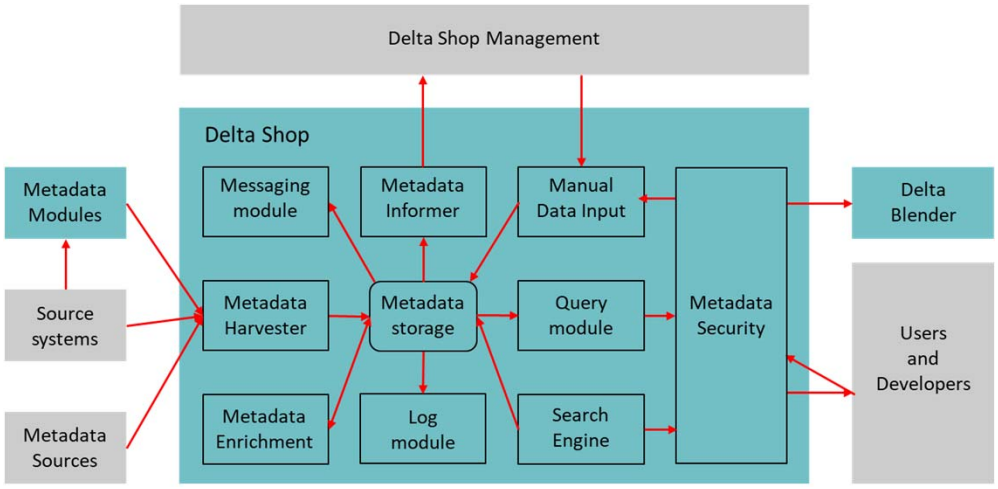
Global Federated Data Architecture



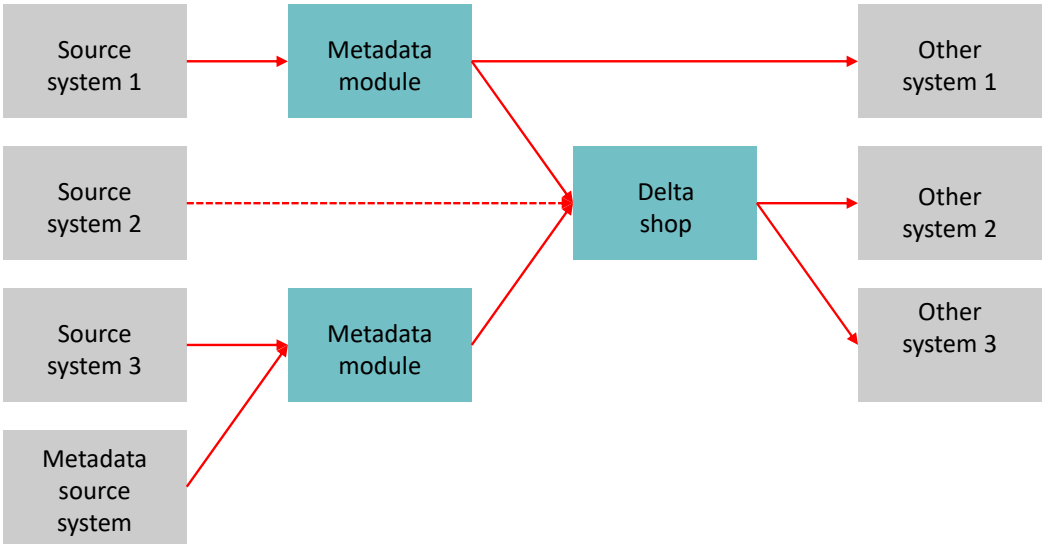
Layered Globale Federated Data Architecture



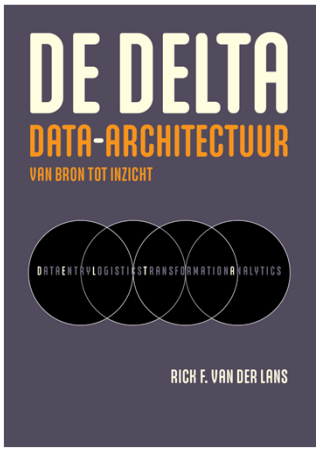
Delta Shop Components - Metadata



Metadata Modules and the Delta Shop



Summary of the Delta Data Architecture

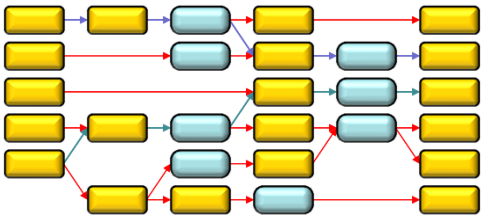


- Enterprise data architecture
  - From source to insight
  - Data must be ready-to-use, discoverable, and insightful
  - It's about the transformers, not the databases
  - Metadata for everyone
- Federated data architecture
  - Unbridled creation of data copies must stop
  - Data-on-demand, not data-by-copy
  - Avoid duplicate transformers
- Transparent data architecture
  - Operational lineage for reconstruction, transparency, and auditability
- Modular data architecture
  - Co-exist with existing solutions

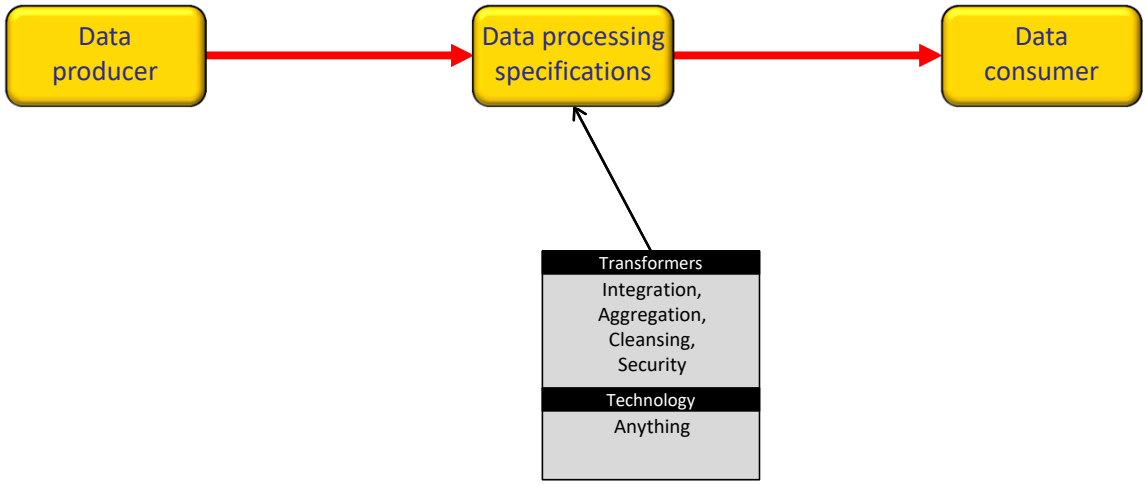
Part 8:
Steps 7-8: Design the New Data Architecture, Determine the Implementation Approach

What is a Data Track?

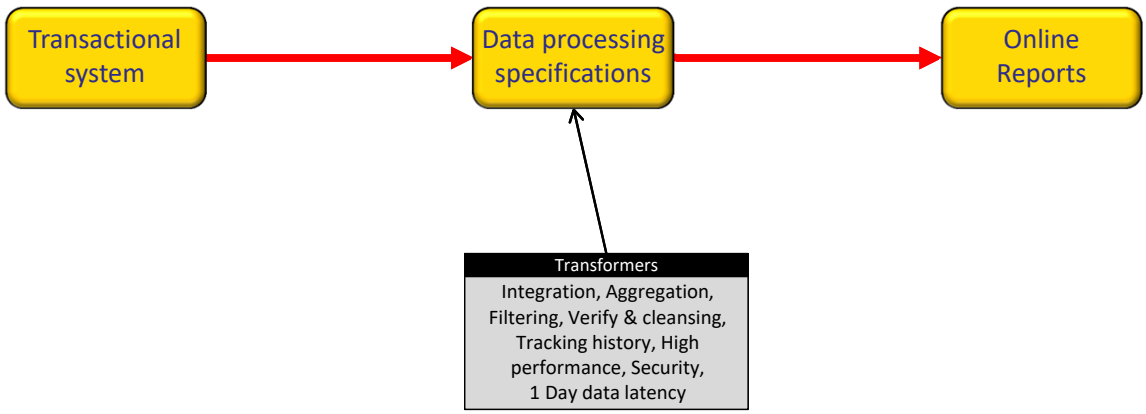
- A *data track* indicates how data "flows" from data producers to data consumers, and specifies the transformers to be applied and by which module.
- Multiple data consumers can share one data track.
- Data tracks may merge and split.



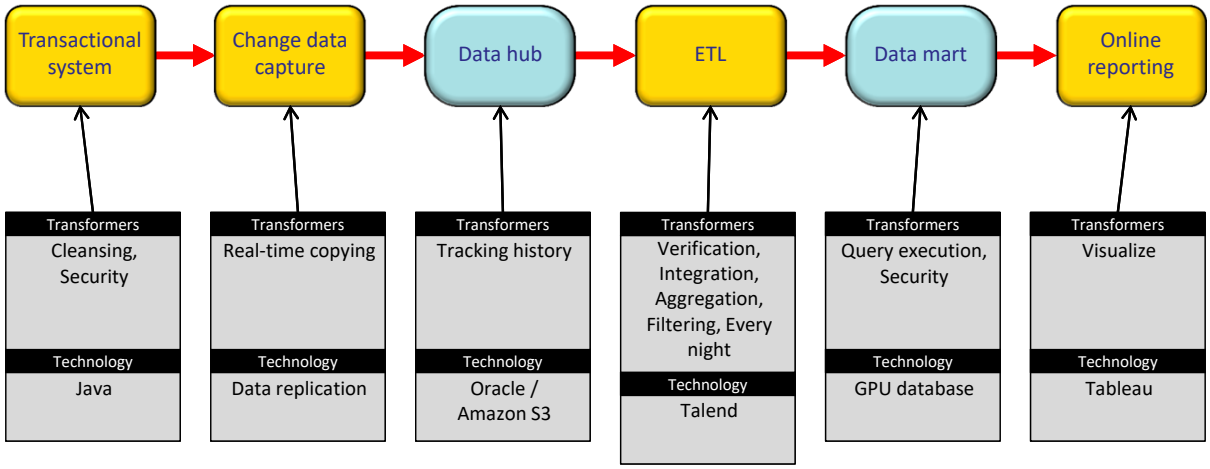
A Data Track Diagram



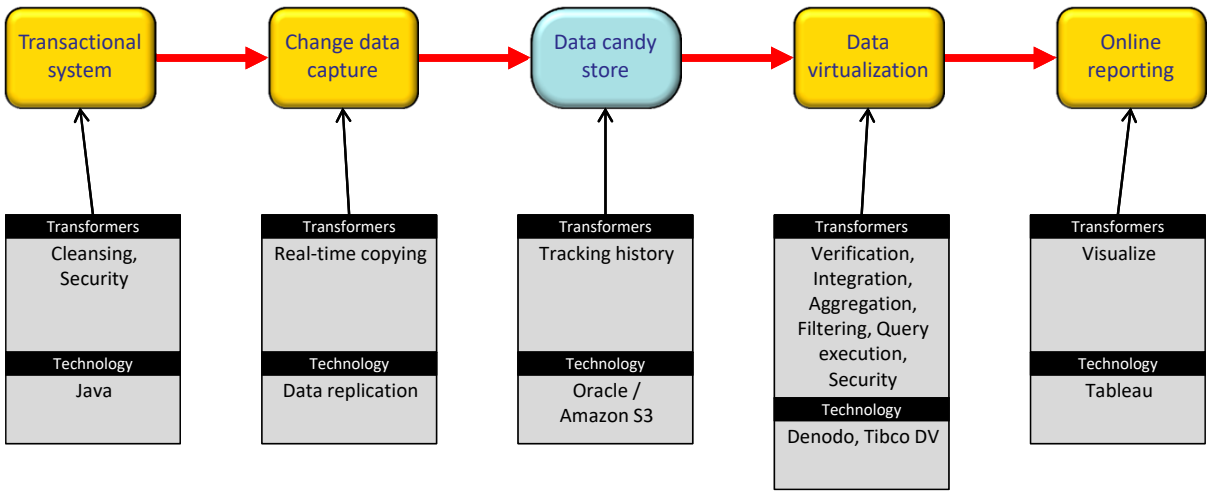
Data Track Example: Standard Online Reporting (1)



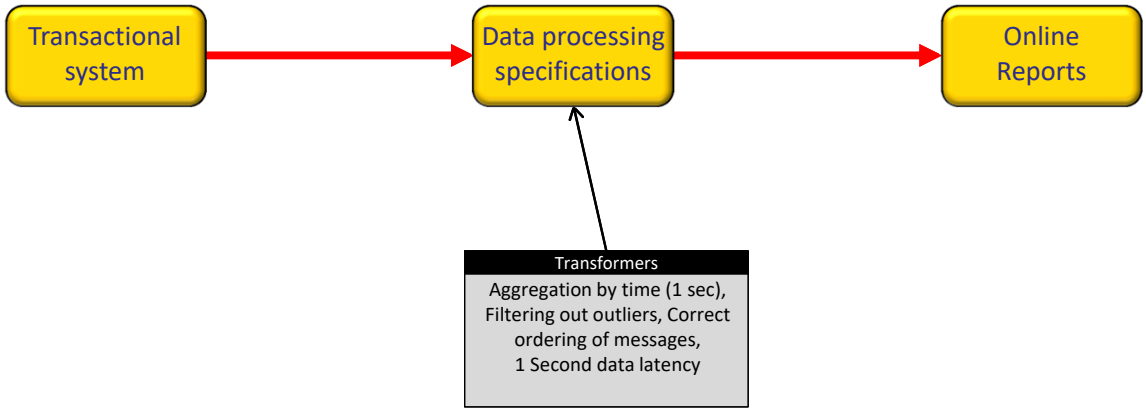
Data Track Example: Standard Online Reporting (2)



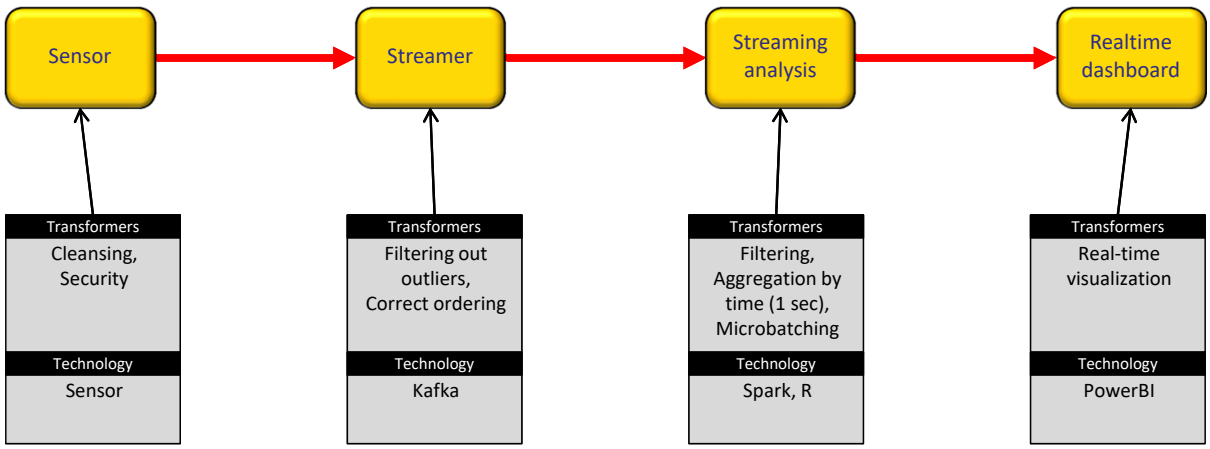
Data Track Example: Standard Online Reporting (3)



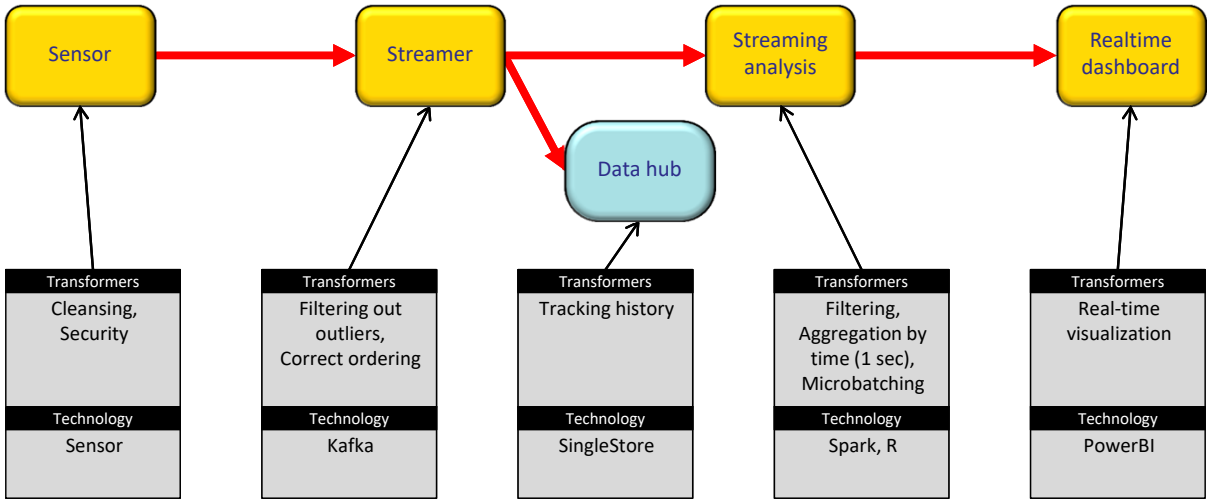
Data Track Example: Streaming Real-time Dashboard (1)



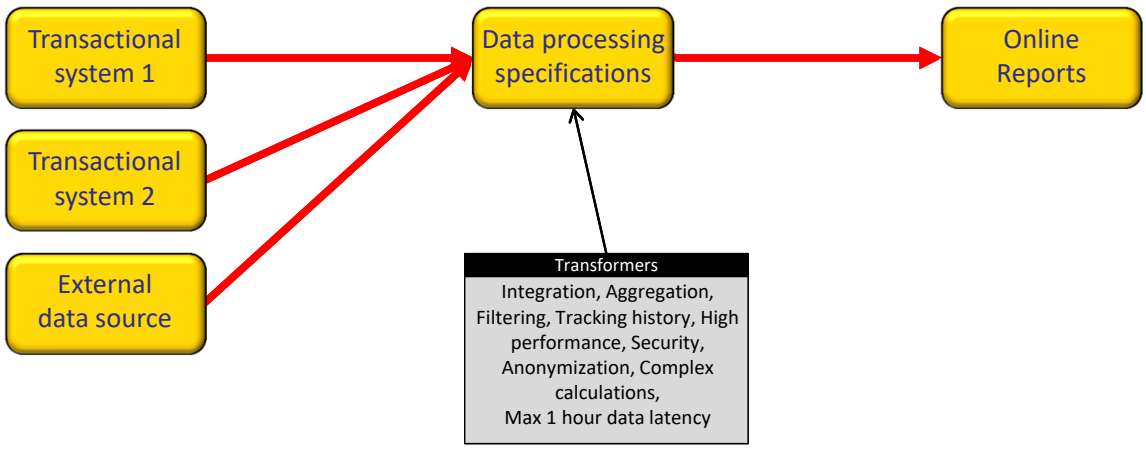
Data Track Example: Streaming Real-time Dashboard (2)



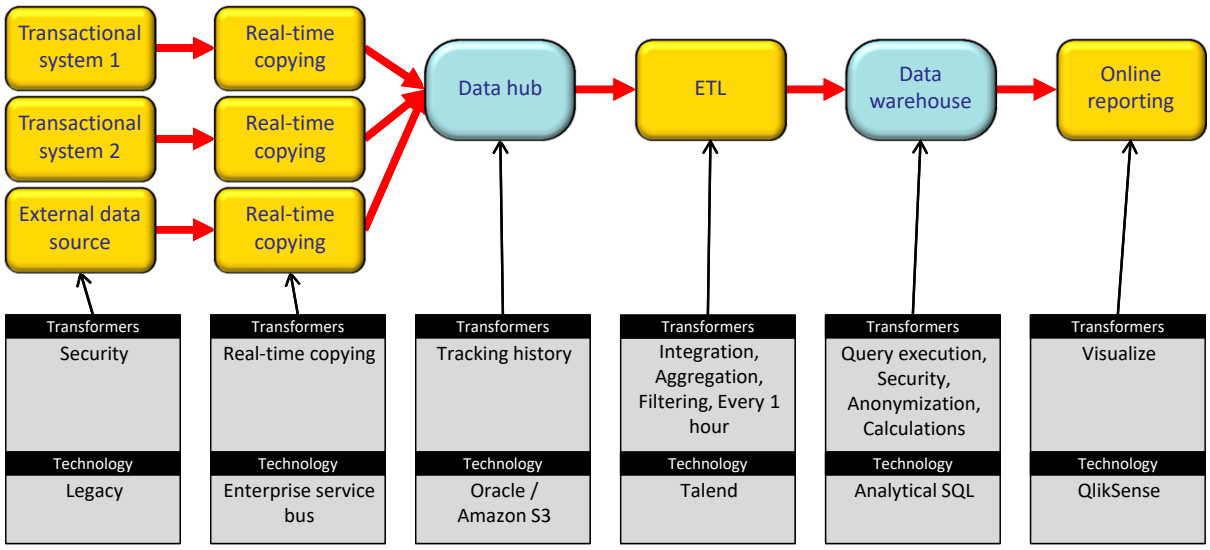
Data Track Example: Streaming Real-time Dashboard (3)



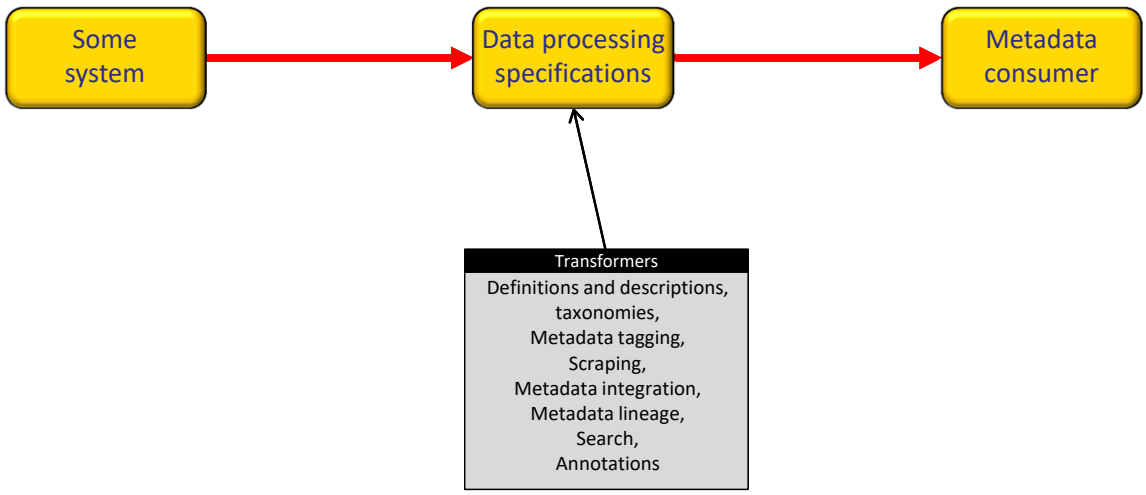
Data Track Example: Integrated Online Reporting (1)



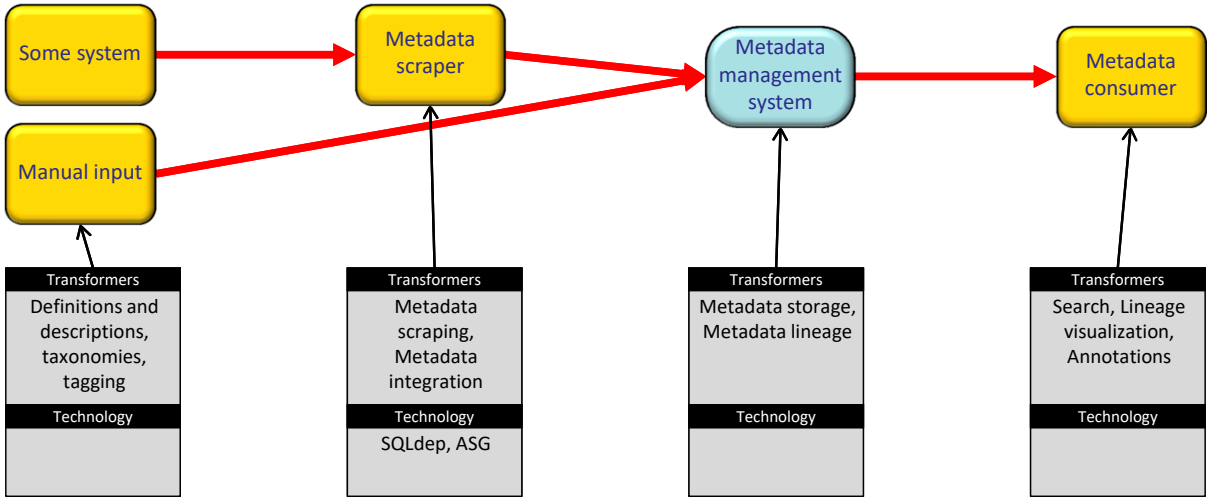
Data Track Example: Integrated Online Reporting (2)



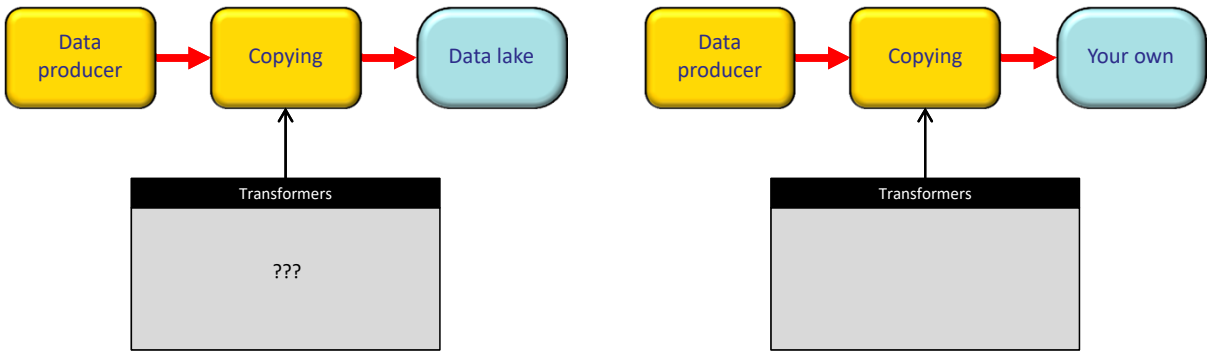
Data Track Example: Metadata Delivery (1)



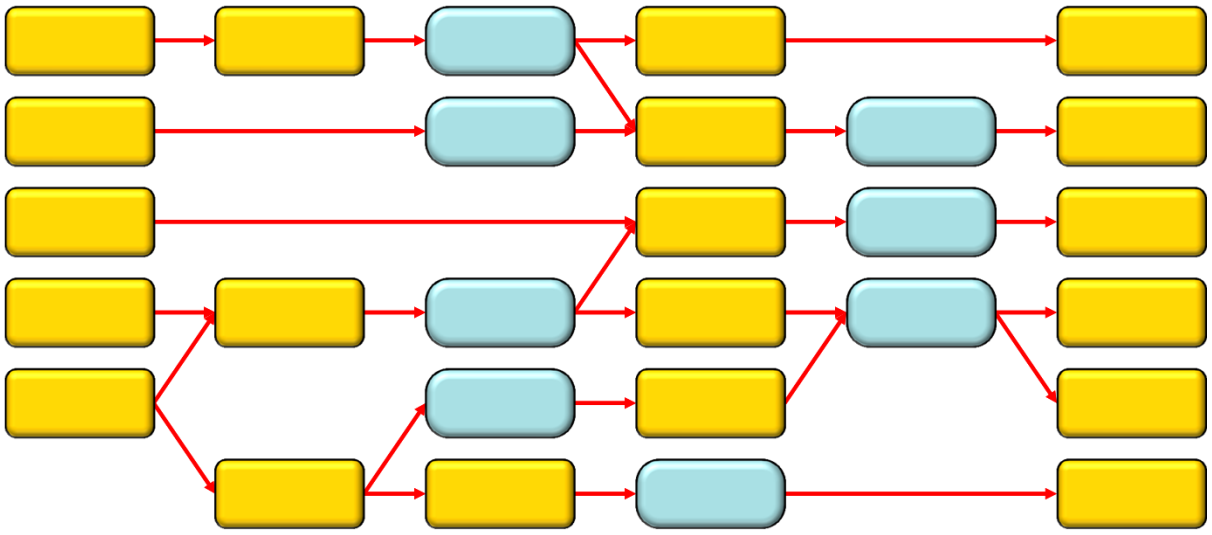
Data Track Example: Metadata Delivery (2)



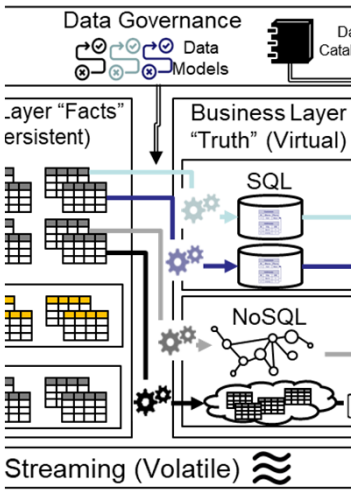
What's in a Name?



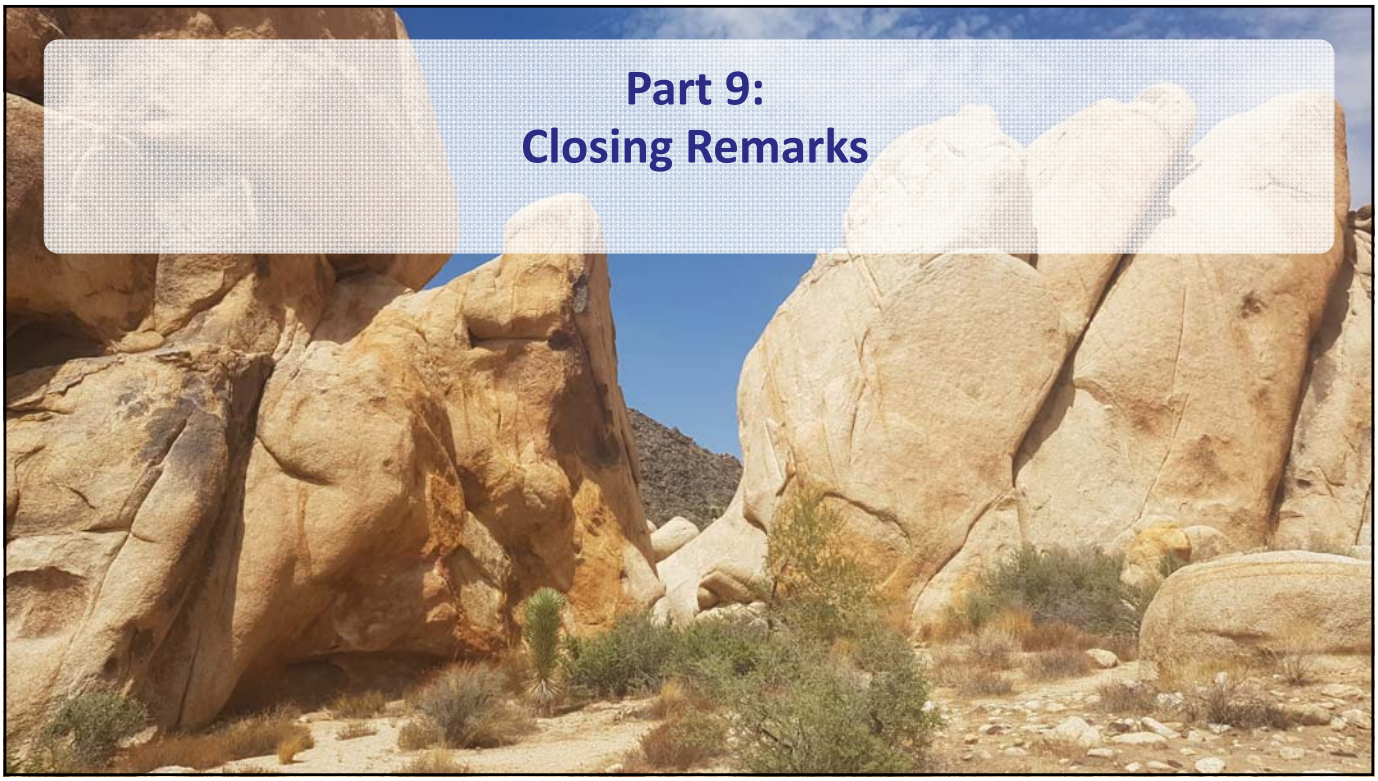
High-Level View of the Tracks



Determine the Intermediate Diagrams



- The current data architecture diagram
- The new data architecture diagram
  - The dream
  - Will never be reached
- The intermediate data architectures
  - The path from current to new
  - Make the steps as small as possible
  - Preferred: Each step leads to business value
- Think big, act small



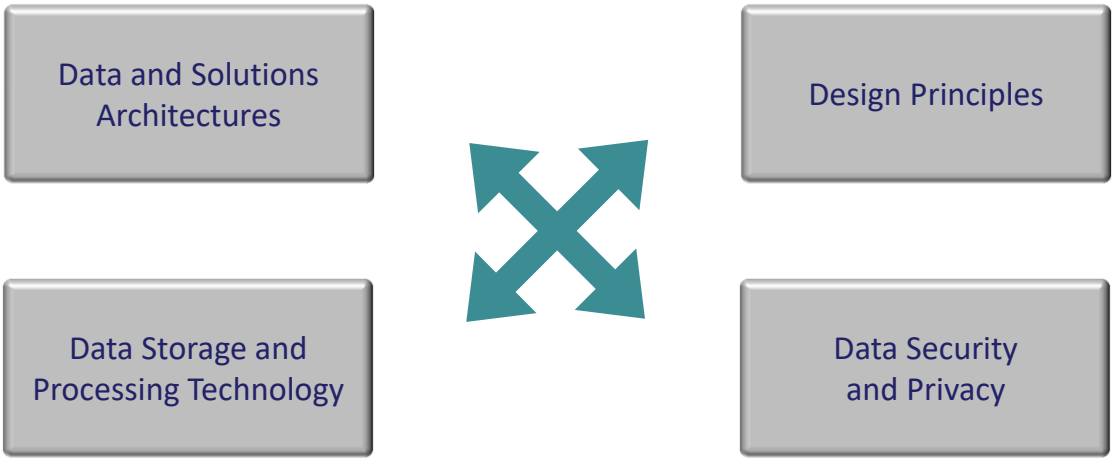
Roadmap for Designing Data Architectures

1. Determine business motivations
2. Determine new requirements
3. Analyze the existing environment
4. Define architectureal design principles
5. Selecta reference data architecture
6. Desogn the new data architecture
7. Determine the implementation approach
8. Select new products and technologies
9. Introduce the data architecture within the organization

Time to Start!

- New data architectures are required
- Focus on transformers, before drawing the storage "boxes"
- Architects must be familiar with the strengths, weaknesses, and use cases of data storage and processing technologies
  - Without this knowledge:
    - Unnecessarily complex architecture
    - Incorrect use of technology
    - Not able to use the full power of a technology
- Design guidelines impact architecture
- Design a data architecture from source to insight

From a Linear to a Holistic Approach




That's all Folks!