Advanced Data Management (CSCI 640/490)

Graph Databases

Dr. David Koop

Time Series Data

- A row of data that consists of a timestamp, a value, optional tags

timestamp tags value

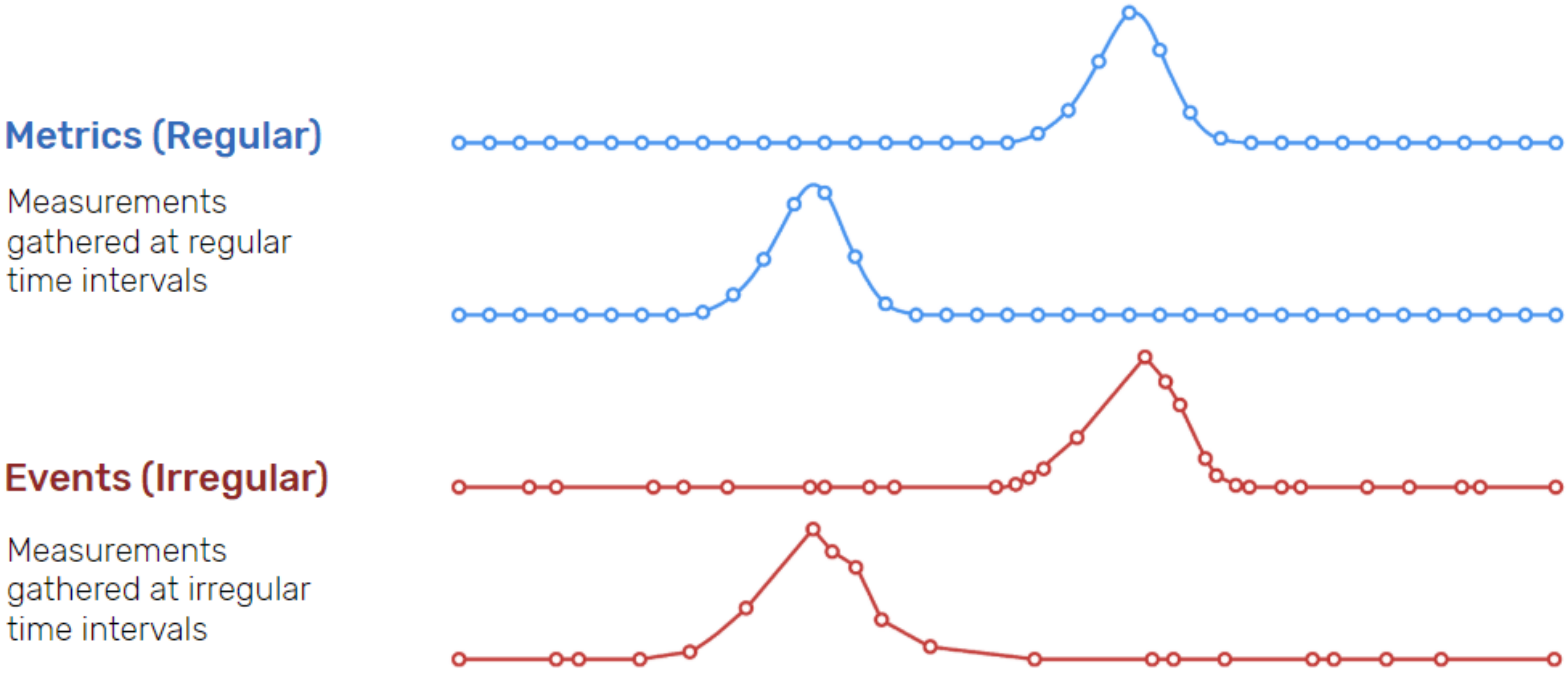
ul1

time	generated	message_subtype	scaler	short_id	tenant	value
2016-07-12T11:51:45Z	"true"	"34"	"4"	"3"	"saarlouis"	465110000
2016-07-12T11:51:45Z	"true"	"34"	"-6"	"2"	"saarlouis"	0.06196699999999994
2016-07-12T12:10:00Z	"true"	"34"	"7"	"5"	"saarlouis"	49370000000
2016-07-12T12:10:00Z	"true"	"34"	"6"	"2"	"saarlouis"	18573000000
2016-07-12T12:10:00Z	"true"	"34"	"5"	"7"	"saarlouis"	5902300000

[A. Bader, 2017]

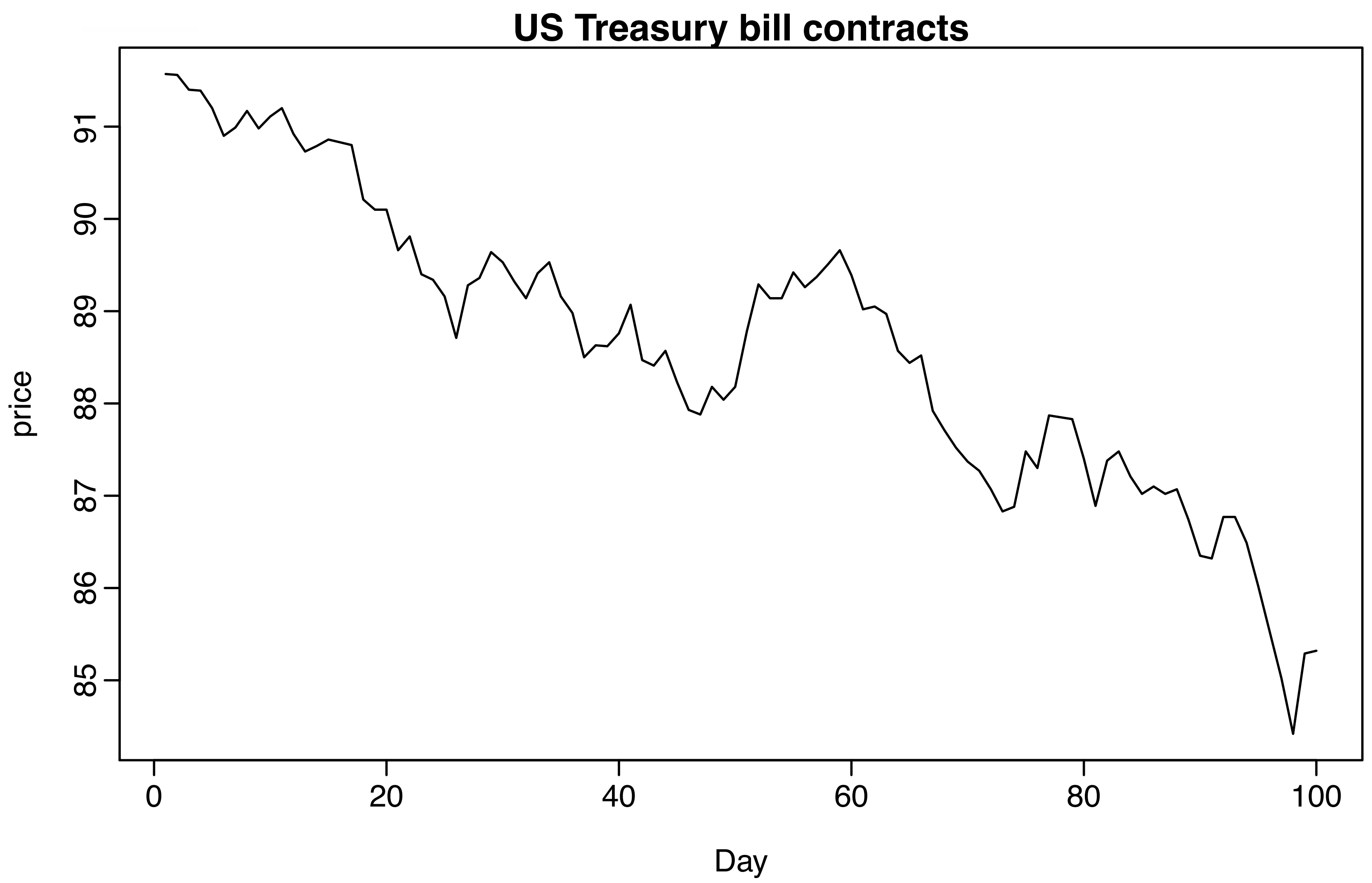
Time Series Data

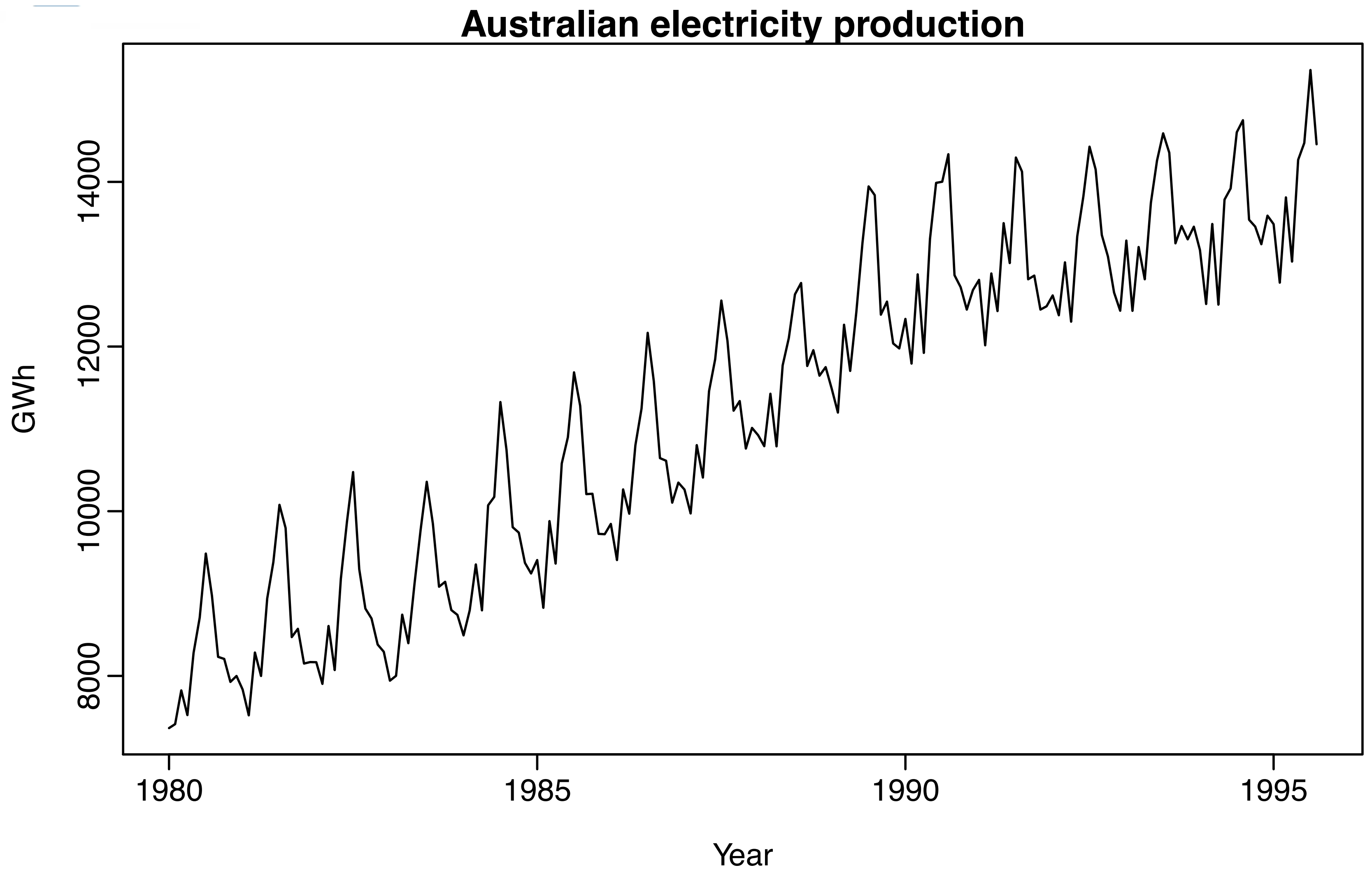
- Metrics: measurements at regular intervals
- Events: measurements that are not gathered at regular intervals

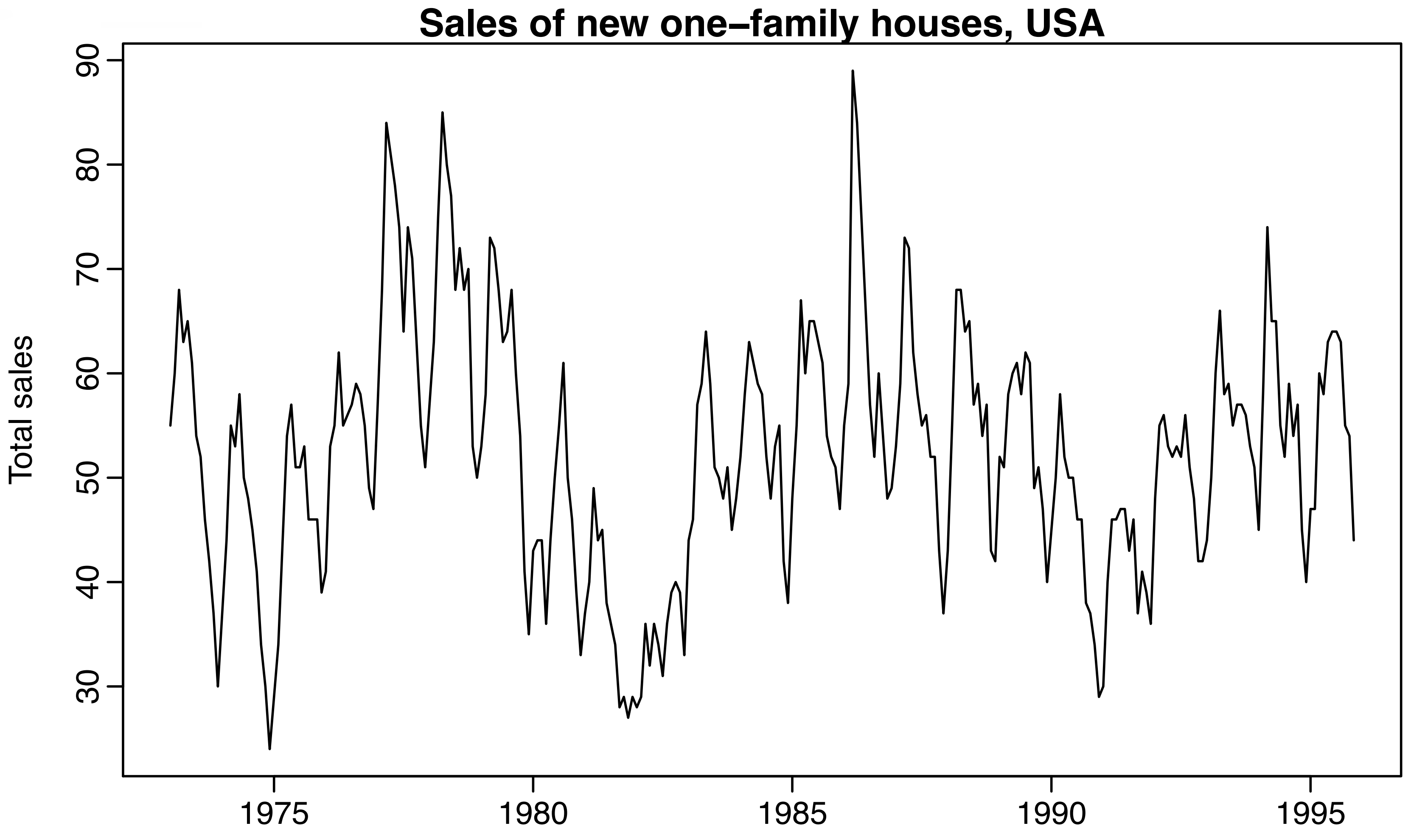


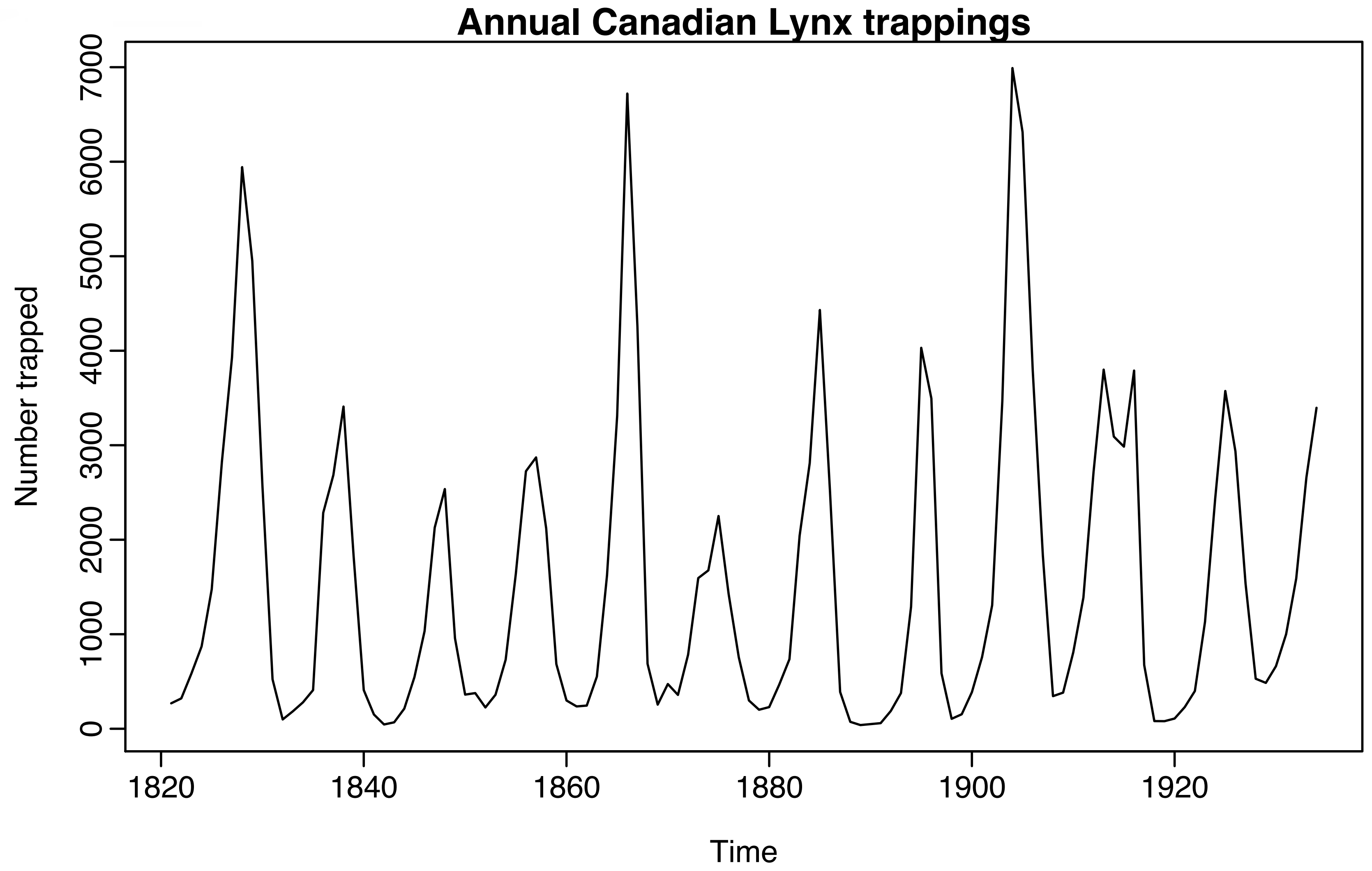
[InfluxDB]

Examples





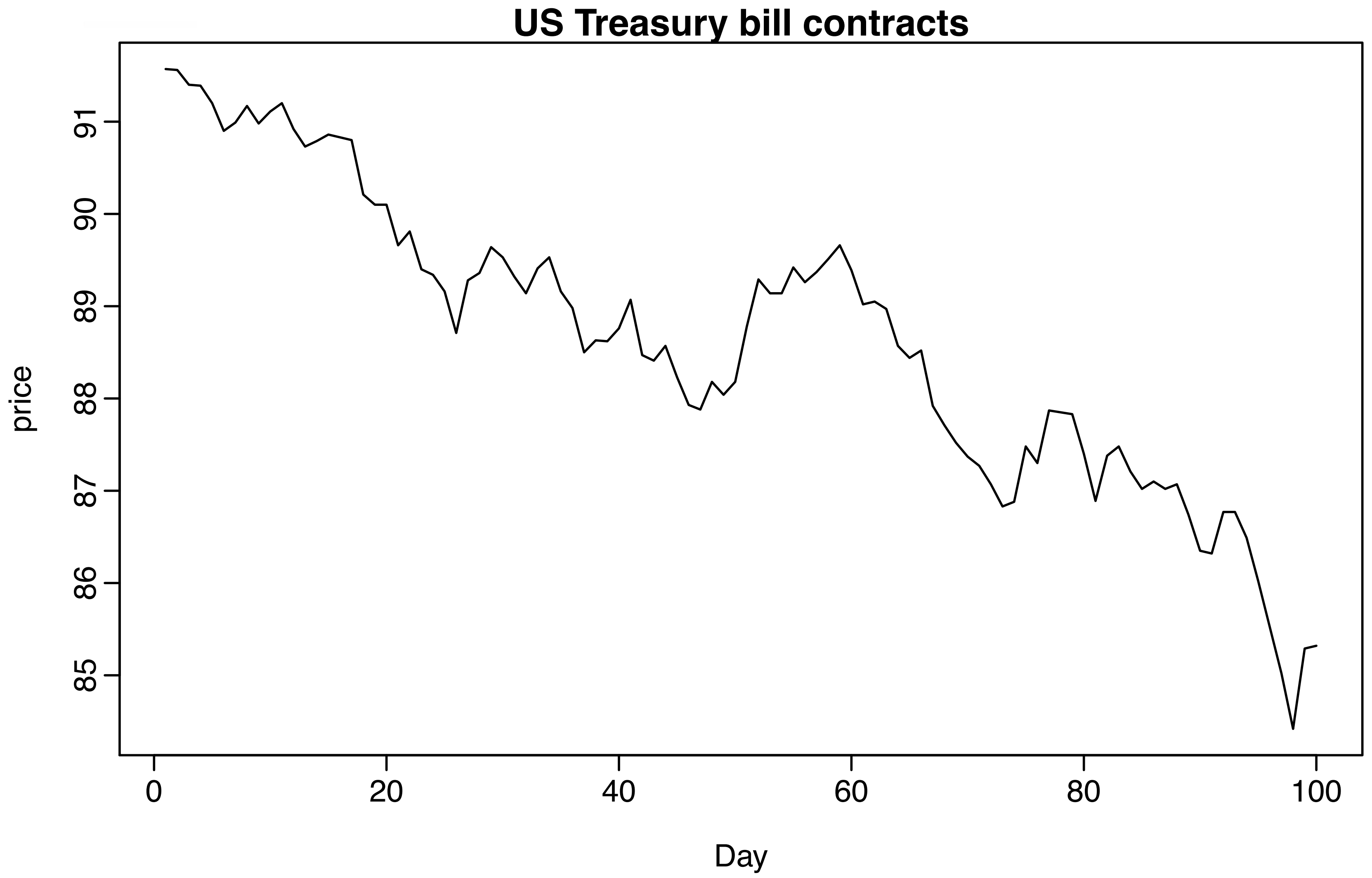


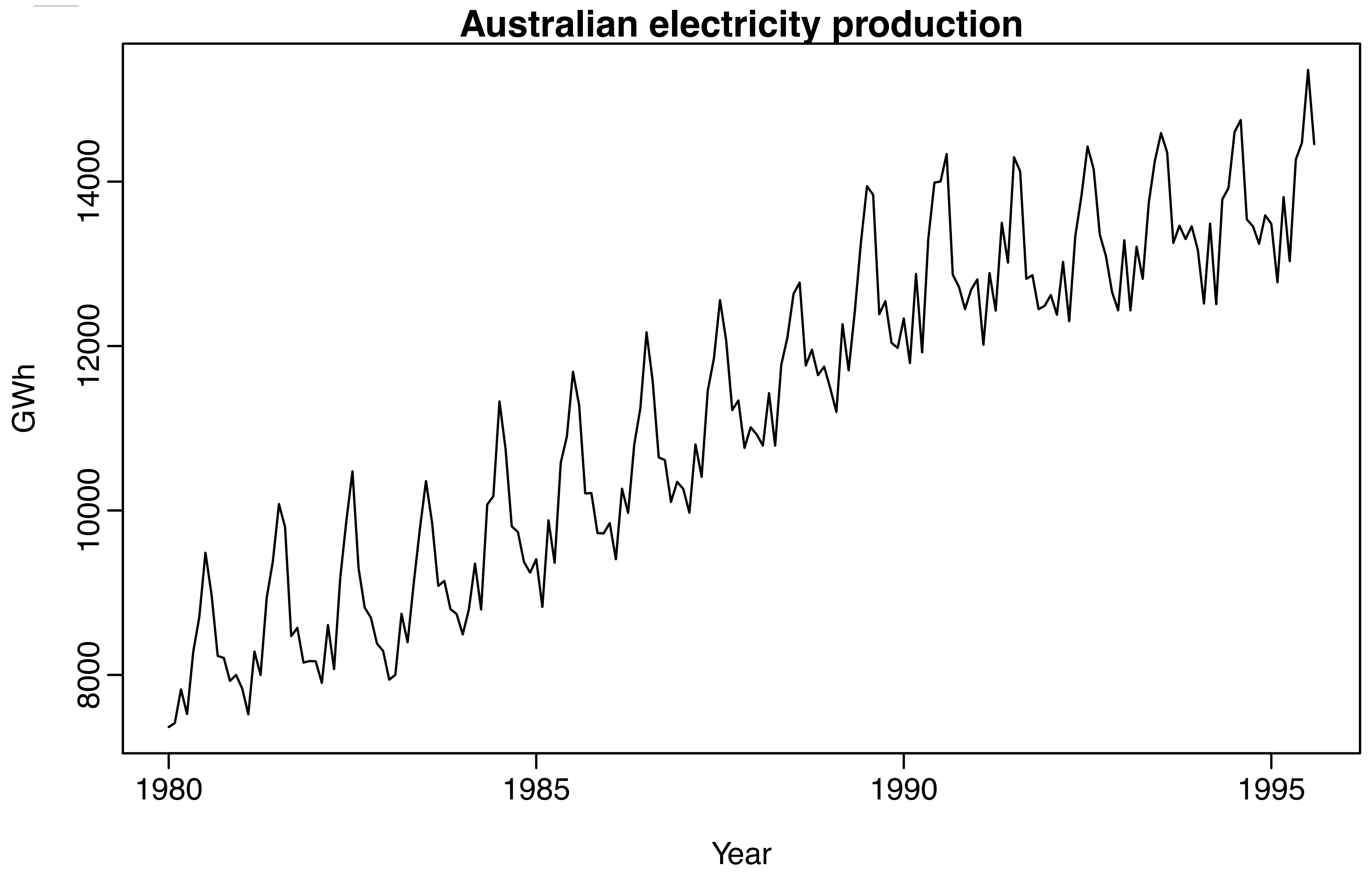


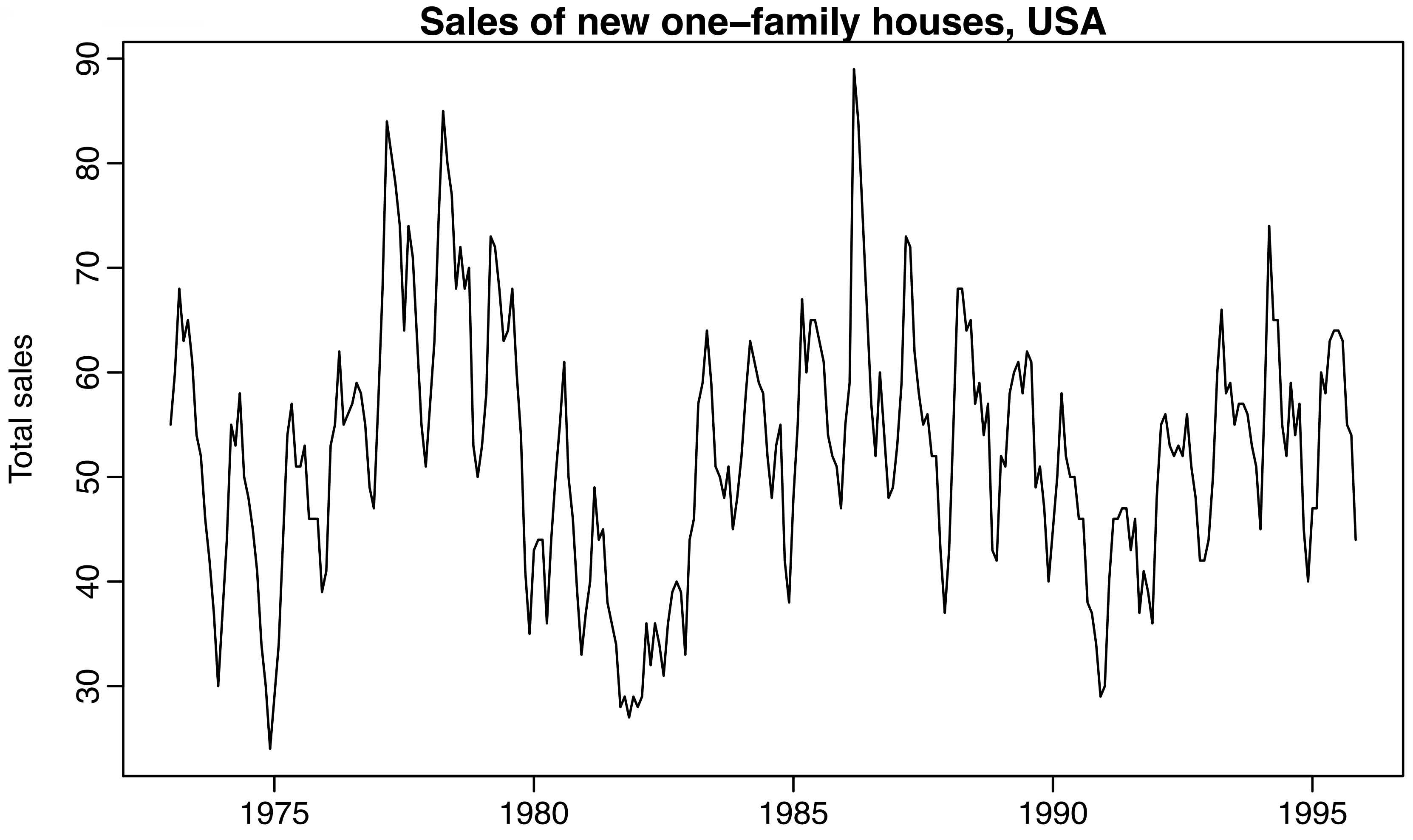
[R. J. Hyndman]

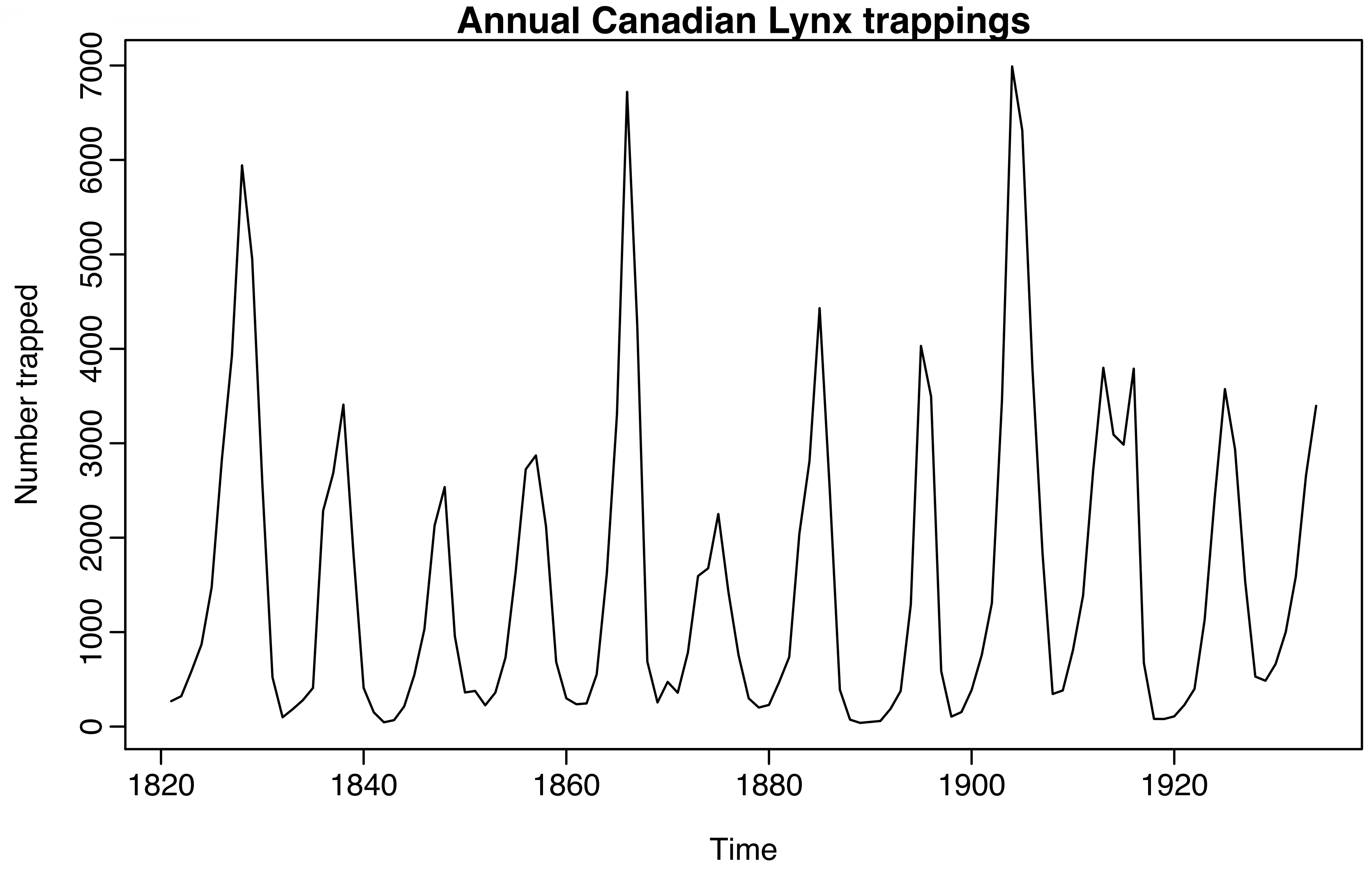
Examples

Trend





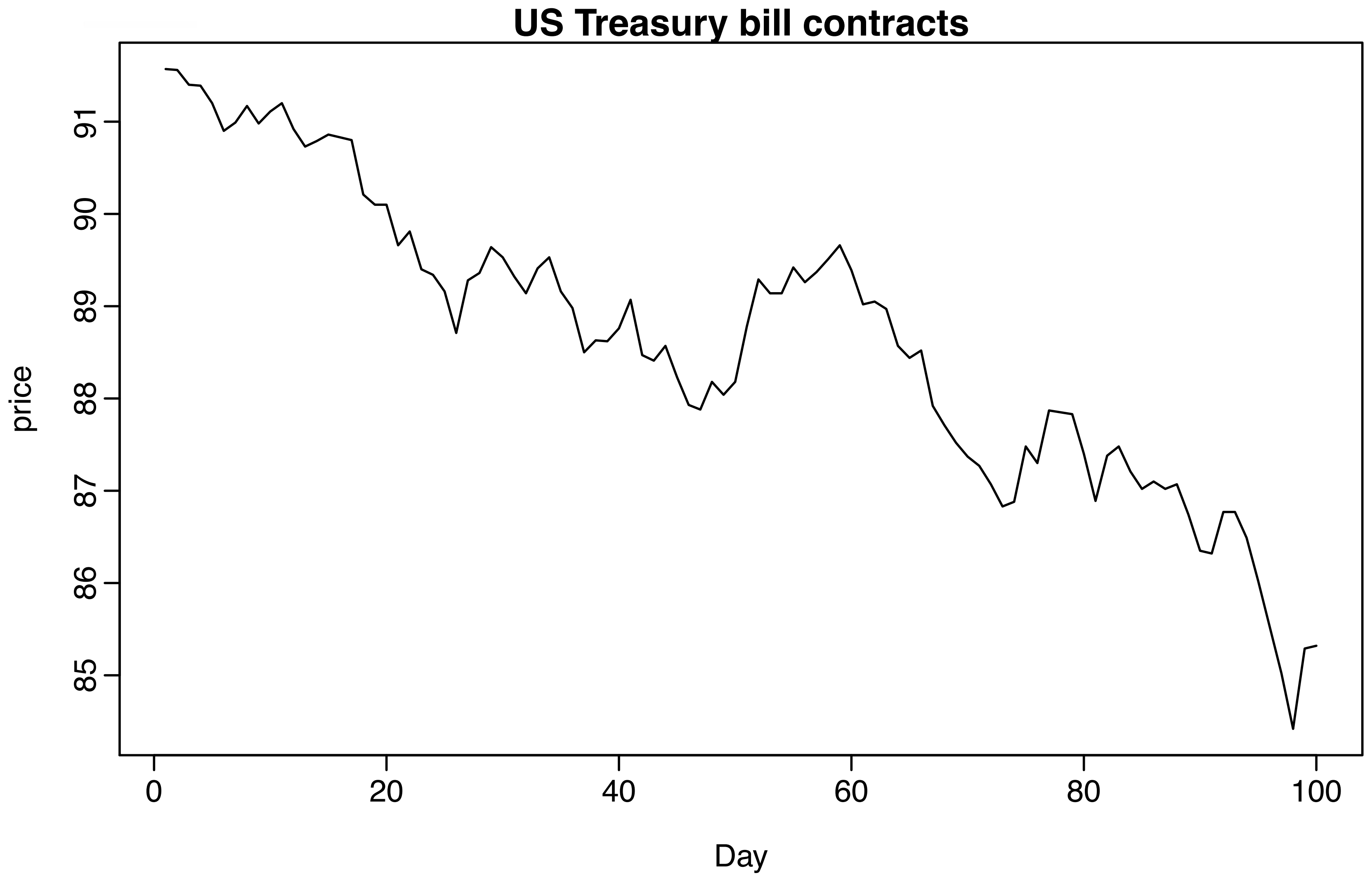


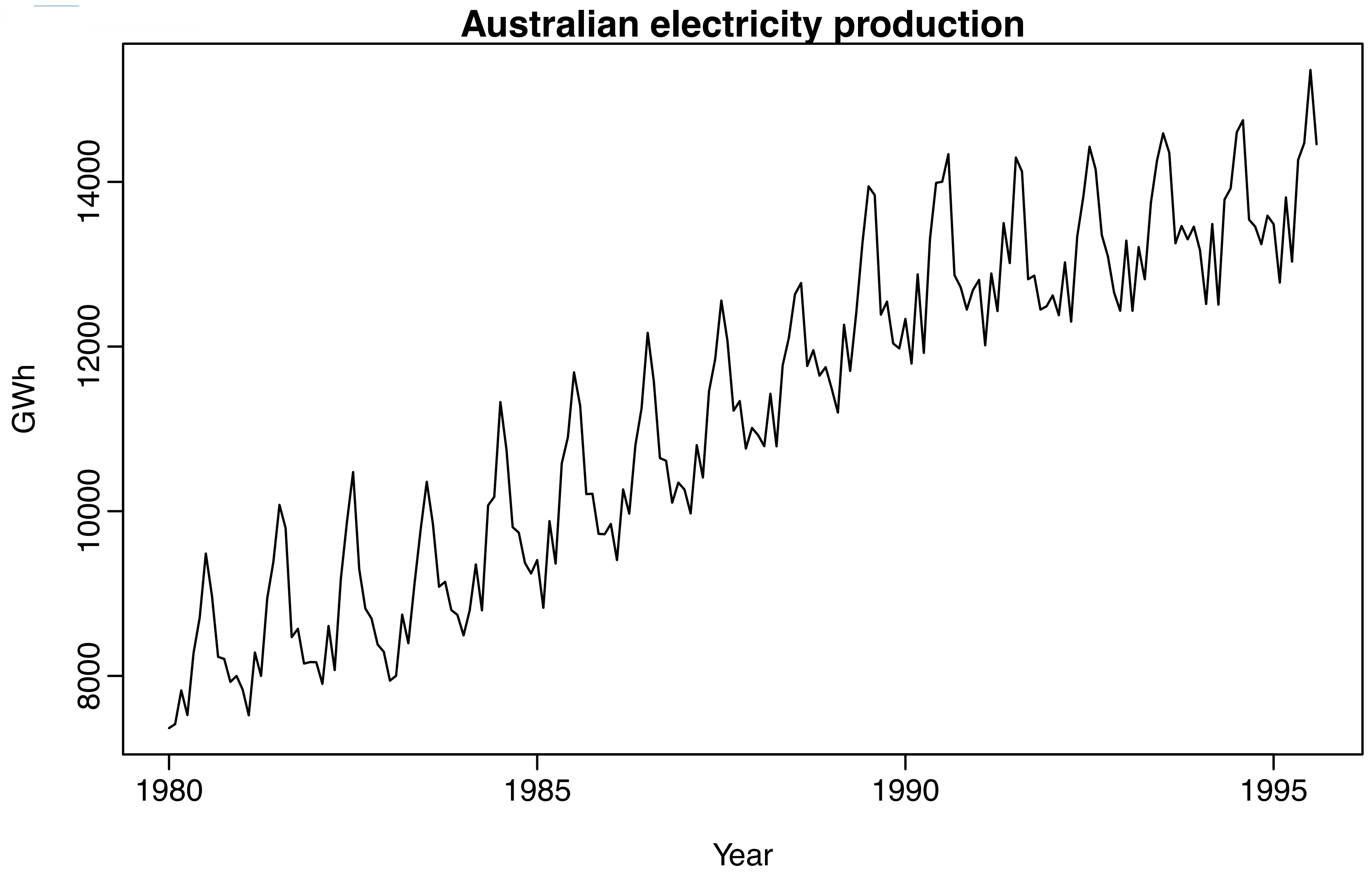


[R. J. Hyndman]

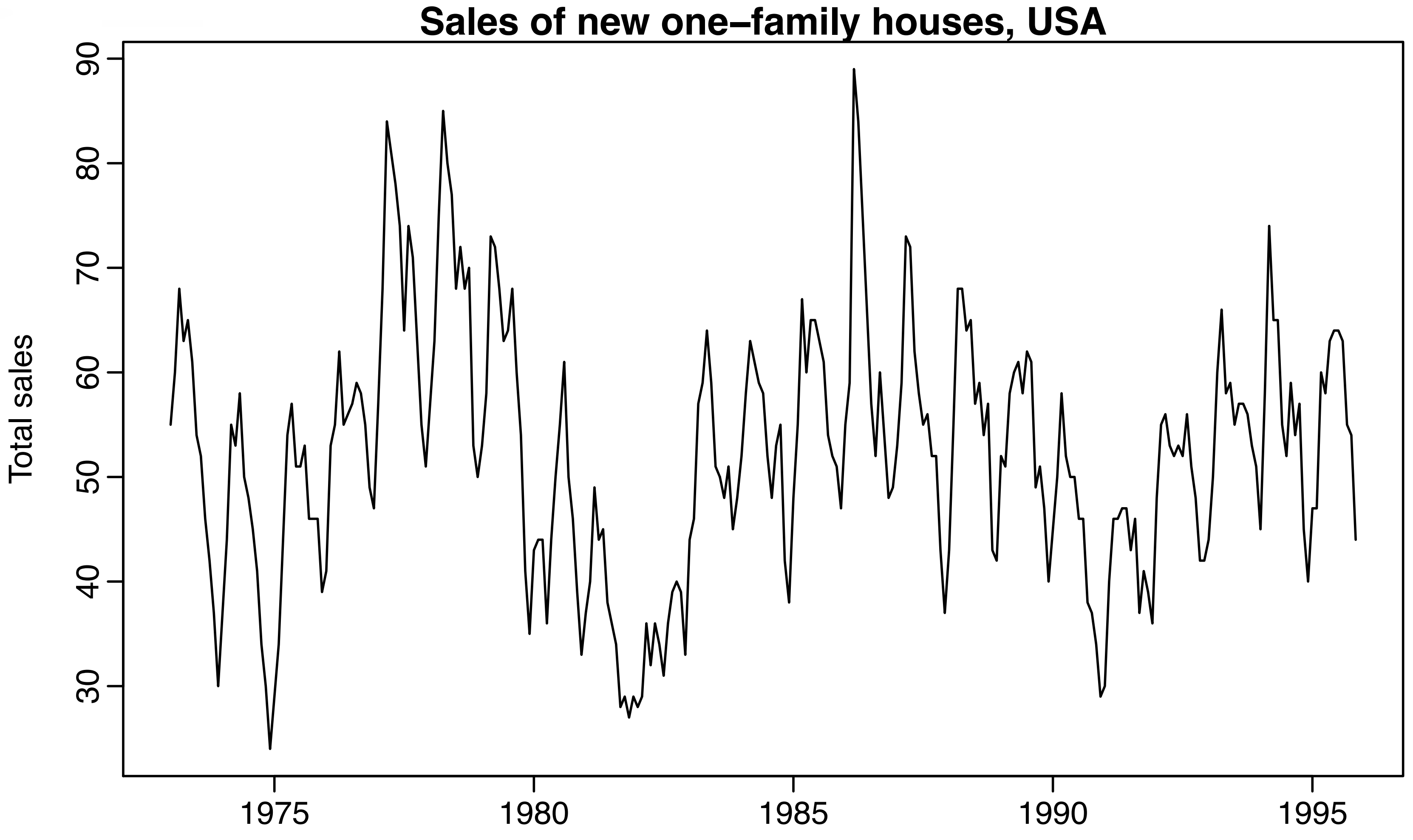
Examples

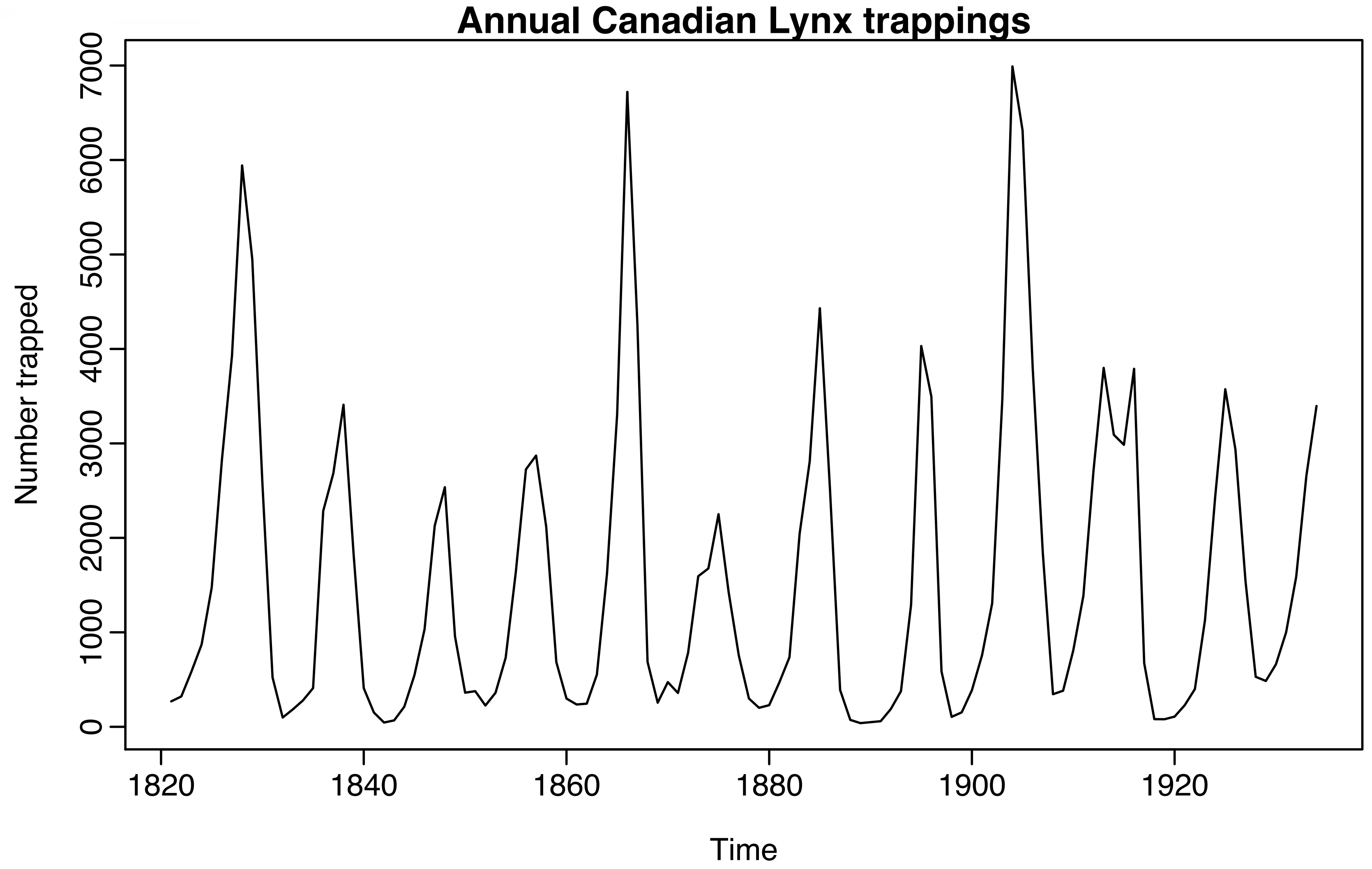
Trend





Trend + Seasonality





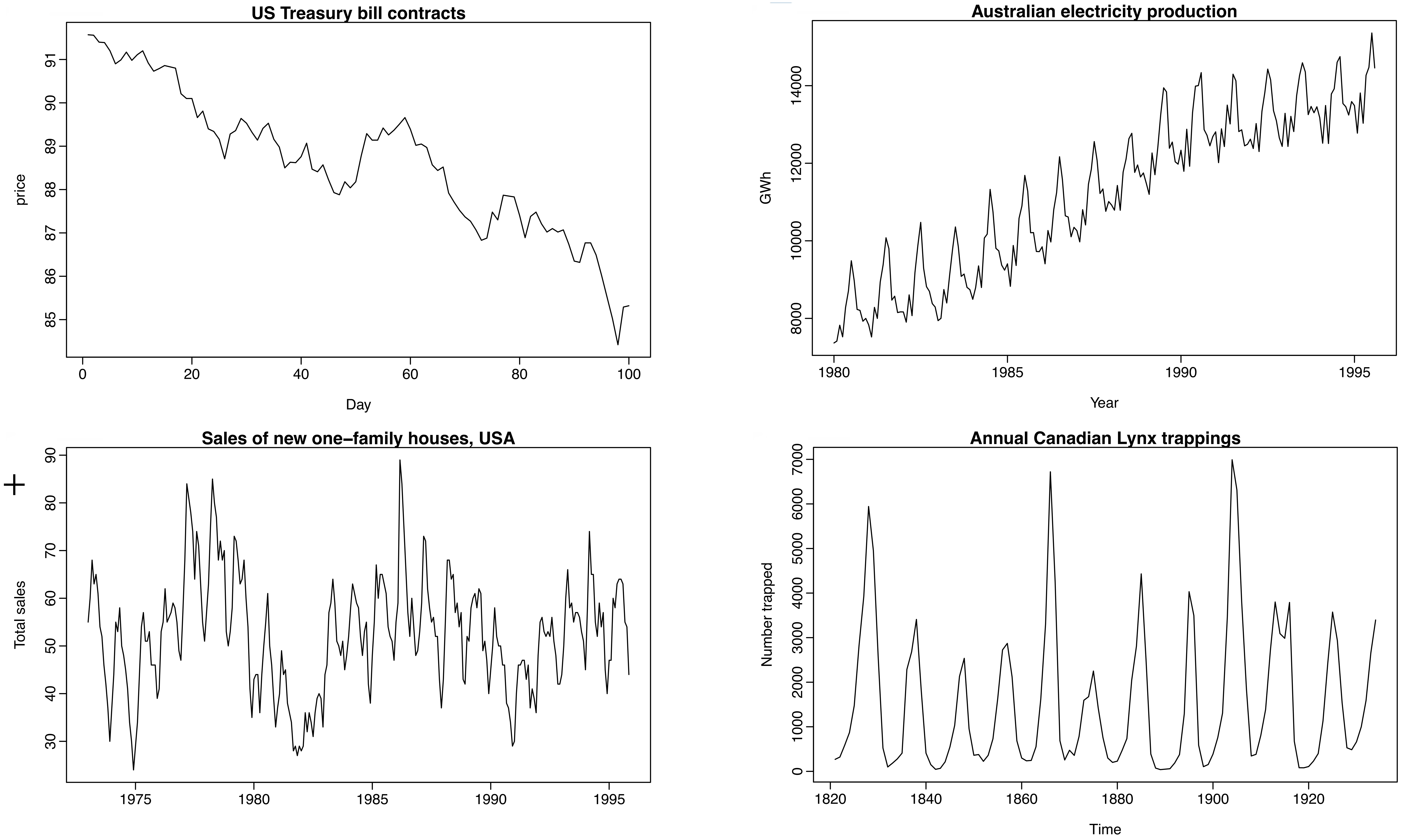
[R. J. Hyndman]

Examples

Trend

Trend + Seasonality

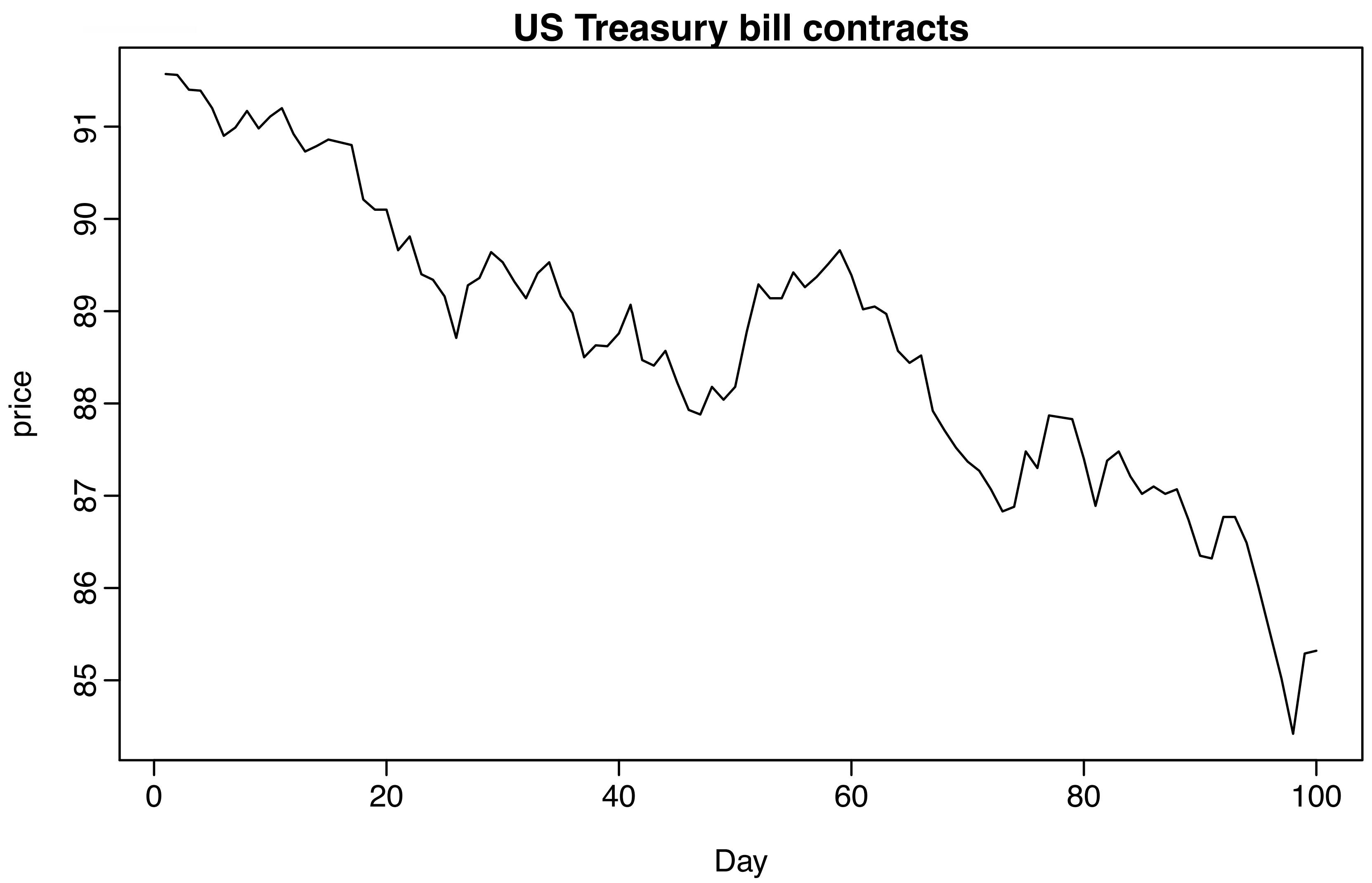
Seasonality + Cyclic



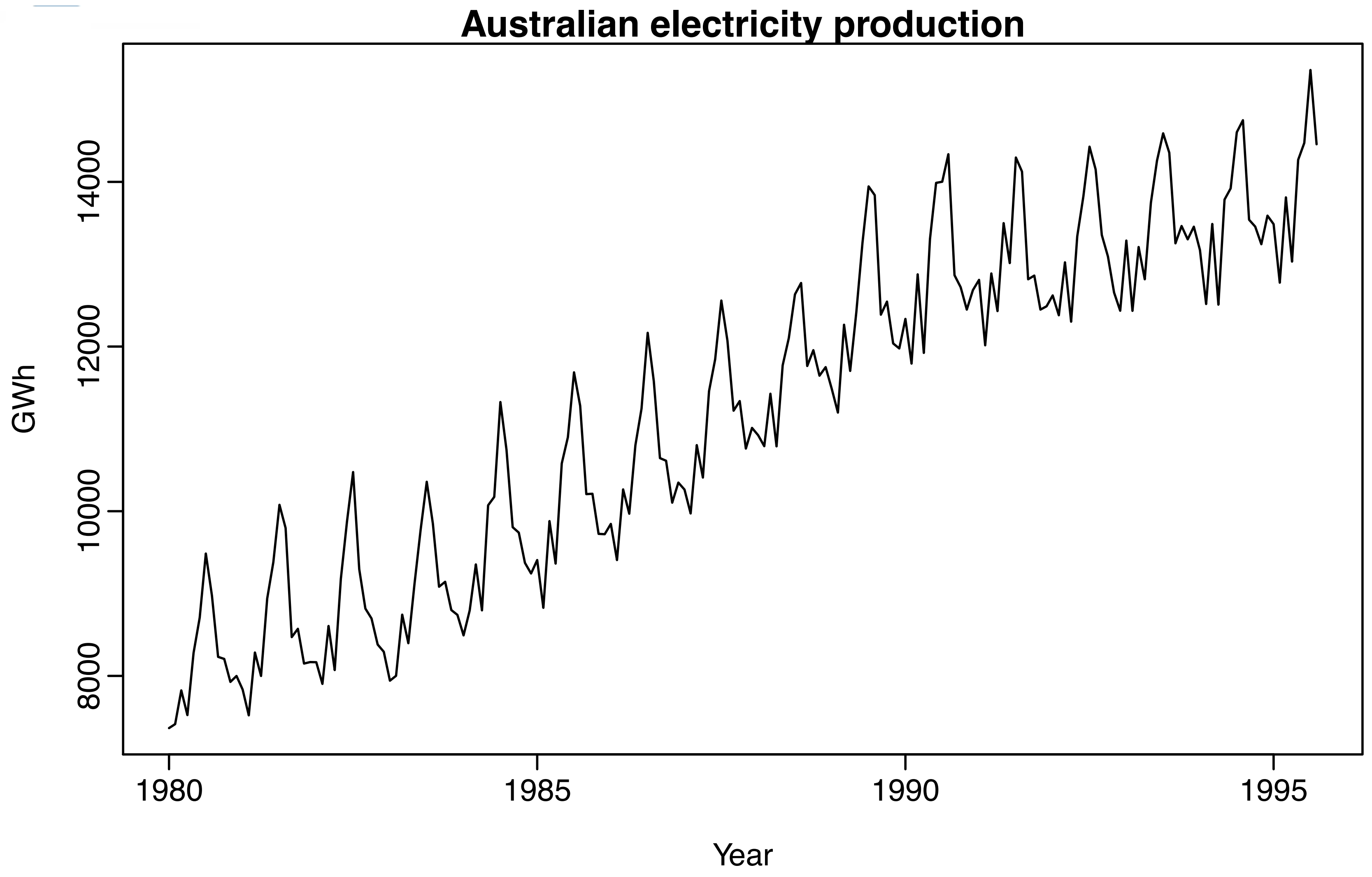
[R. J. Hyndman]

Examples

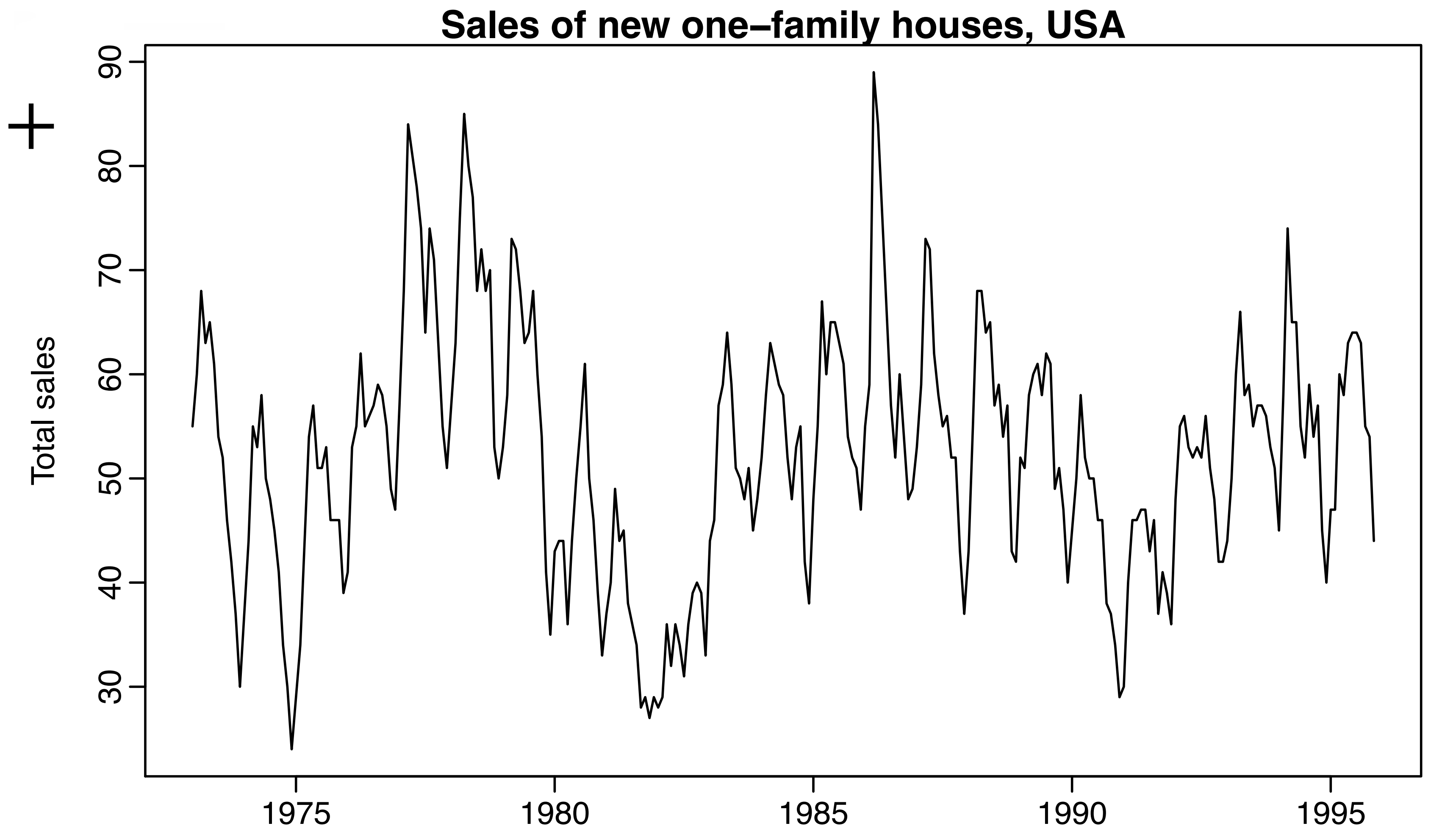
Trend



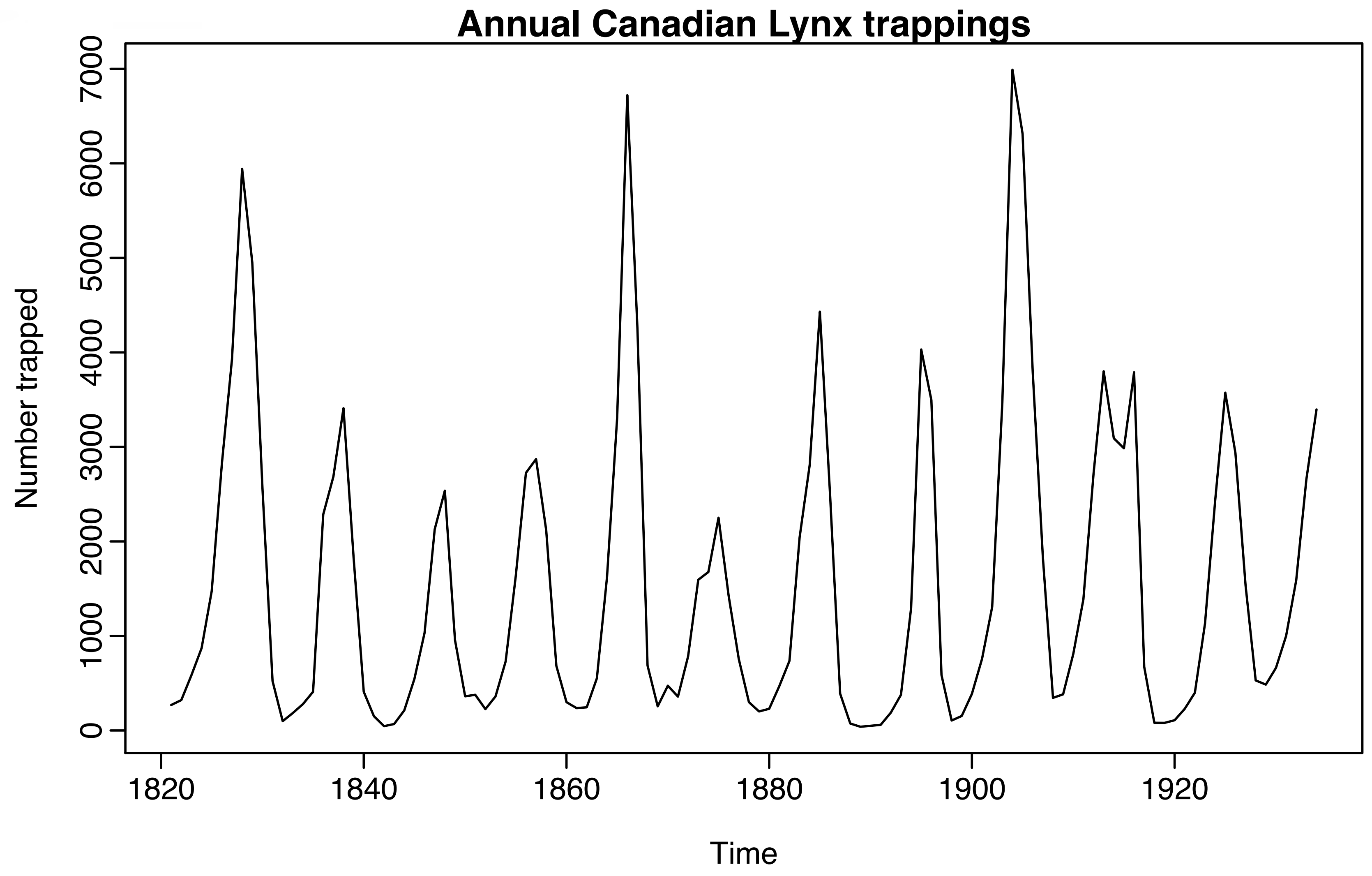
Trend + Seasonality



Seasonality + Cyclic



Stationary



[R. J. Hyndman]

Pandas Support for Datetime

- `pd.to_datetime`:
  - convenience method
  - can convert an entire column to datetime
- Has a `NaT` to indicate a missing time value
- Stores in a `numpy.datetime64` format
- `pd.Timestamp`: a wrapper for the `datetime64` objects

Resampling

- Could be
  - downsample: higher frequency to lower frequency
  - upsample: lower frequency to higher frequency
  - neither: e.g. Wednesdays to Fridays
- resample method: e.g. `ts.resample('M').mean()`

Argument	Description
`freq`	String or DateOffset indicating desired resampled frequency (e.g., 'M', '5min', or `Second(15)`)
`axis`	Axis to resample on; default axis=0
`fill_method`	How to interpolate when upsampling, as in `'ffill'` or `'bfill'`; by default does no interpolation
`closed`	In downsampling, which end of each interval is closed (inclusive), `'right'` or `'left'`
`label`	In downsampling, how to label the aggregated result, with the `'right'` or `'left'` bin edge (e.g., the 9:30 to 9:35 five-minute interval could be labeled `9:30` or `9:35`)
`loffset`	Time adjustment to the bin labels, such as `'-1s'` / `Second(-1)` to shift the aggregate labels one second earlier
`limit`	When forward or backward filling, the maximum number of periods to fill
`kind`	Aggregate to periods (`'period'`) or timestamps (`'timestamp'`); defaults to the type of index the time series has
`convention`	When resampling periods, the convention (`'start'` or `'end'`) for converting the low-frequency period to high frequency; defaults to `'end'`

[W. McKinney, Python for Data Analysis]

Time Series Databases

- Most time series data is heavy **inserts**, few updates
- Also analysis tends to be on ordered data with trends, prediction, etc.
- Can also consider **stream** processing
- Focus on time series allows databases to specialize
- Examples:
  - InfluxDB (noSQL)
  - TimescaleDB (SQL-based)

What is a Time Series Database?

- A DBMS is called TSDB if it can
  - store a row of data that consists of timestamp, value, and optional tags
  - store multiple rows of time series data grouped together
  - can query for rows of data
  - can contain a timestamp or a time range in a query

ul1

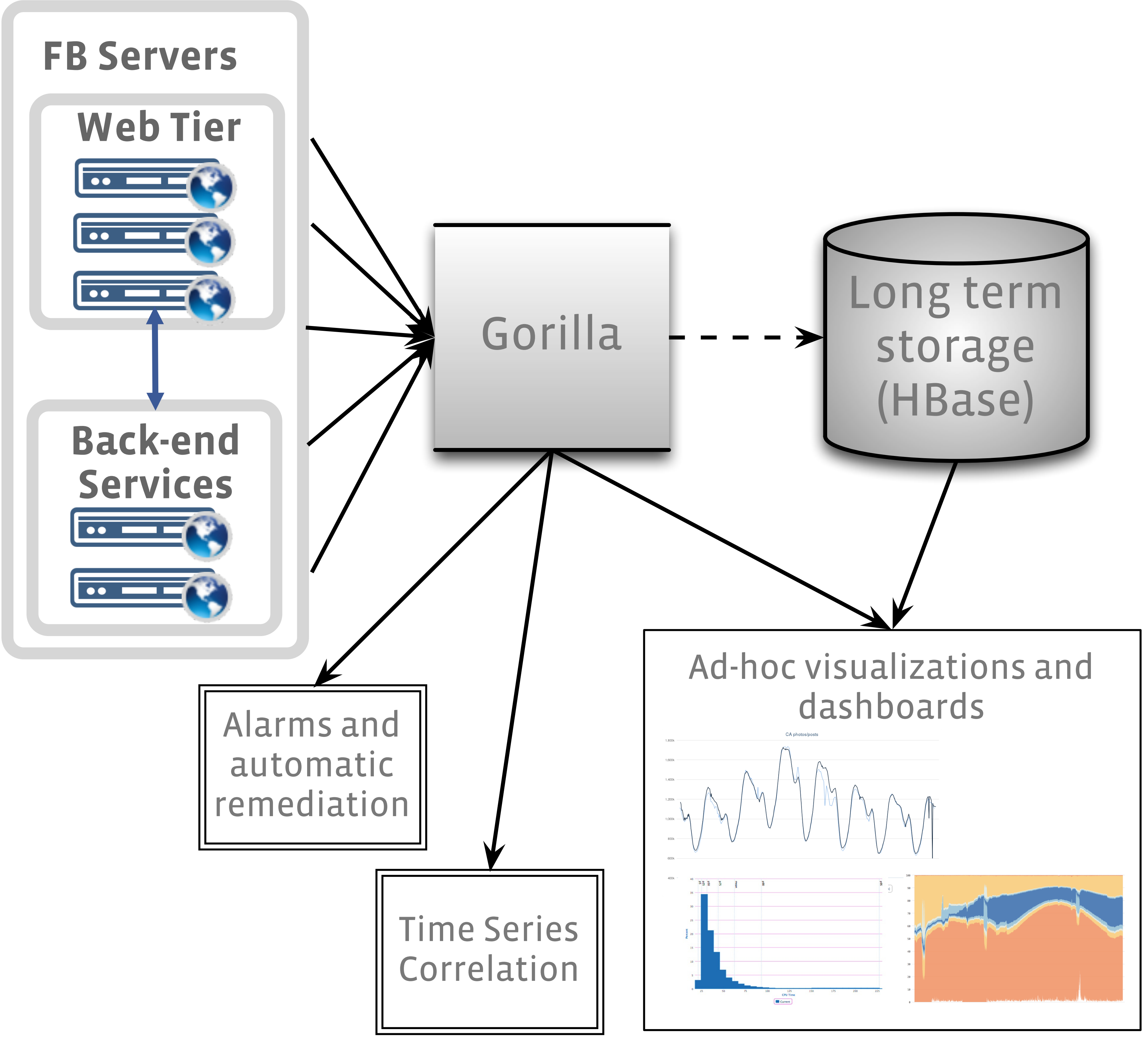
“SELECT * FROM ul1 WHERE time >= '2016-07-12T12:10:00Z‘”

time	generated	message_subtype	scaler	short_id	tenant	value
2016-07-12T11:51:45Z	"true"	"34"	"4"	"3"	"saarlouis"	465110000
2016-07-12T11:51:45Z	"true"	"34"	"-6"	"2"	"saarlouis"	0.061966999999999994
2016-07-12T12:10:00Z	"true"	"34"	"7"	"5"	"saarlouis"	49370000000
2016-07-12T12:10:00Z	"true"	"34"	"6"	"2"	"saarlouis"	18573000000
2016-07-12T12:10:00Z	"true"	"34"	"5"	"7"	"saarlouis"	5902300000

[A. Bader, 2017]

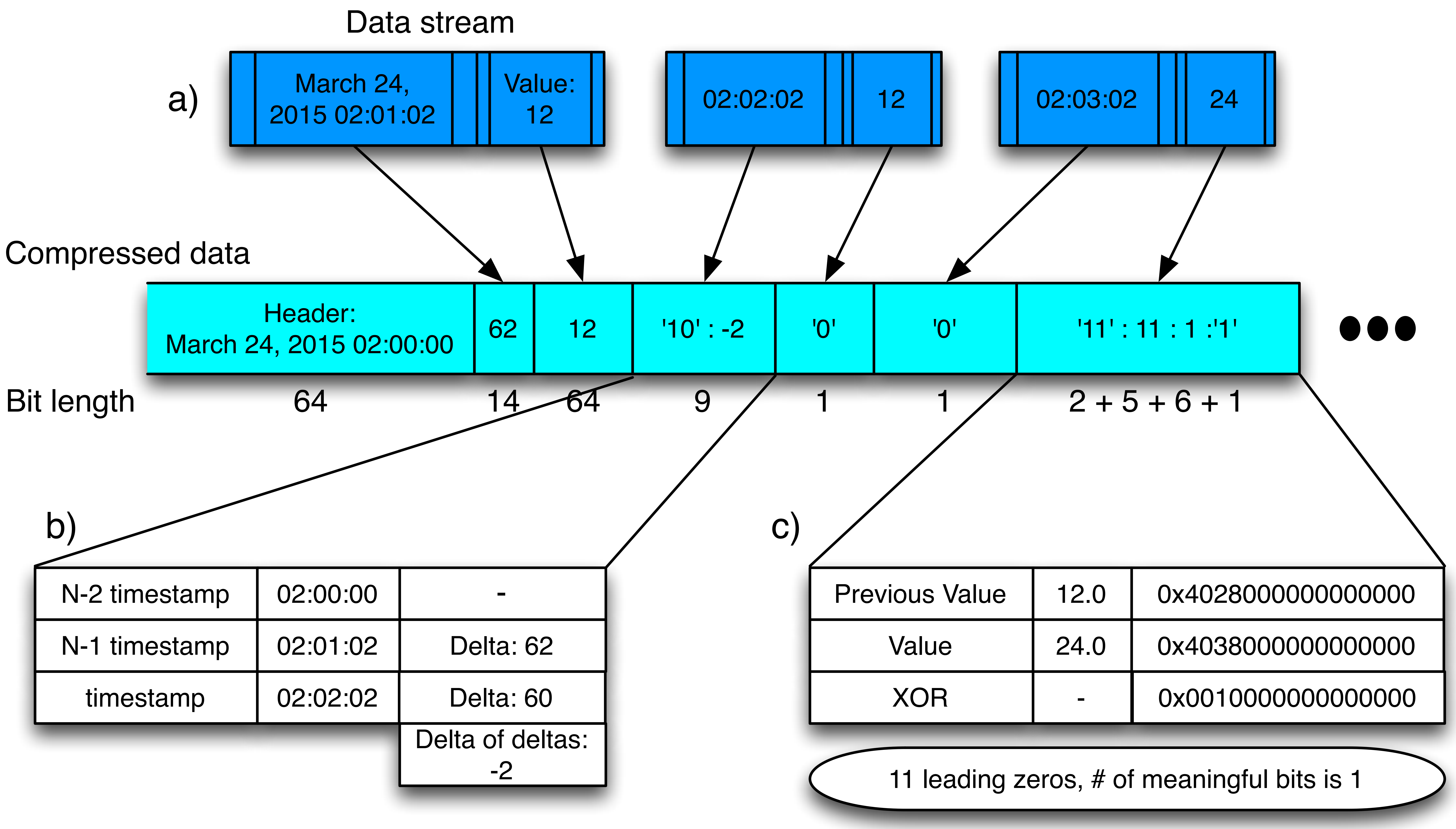
Gorilla Motivation

- Large-scale internet services rely on lots of services and machines
- Want to monitor the health of the systems
- Writes dominate
- Want to detect state transitions
- Must be highly available and fault tolerant



[Pelkonen et al., 2015]

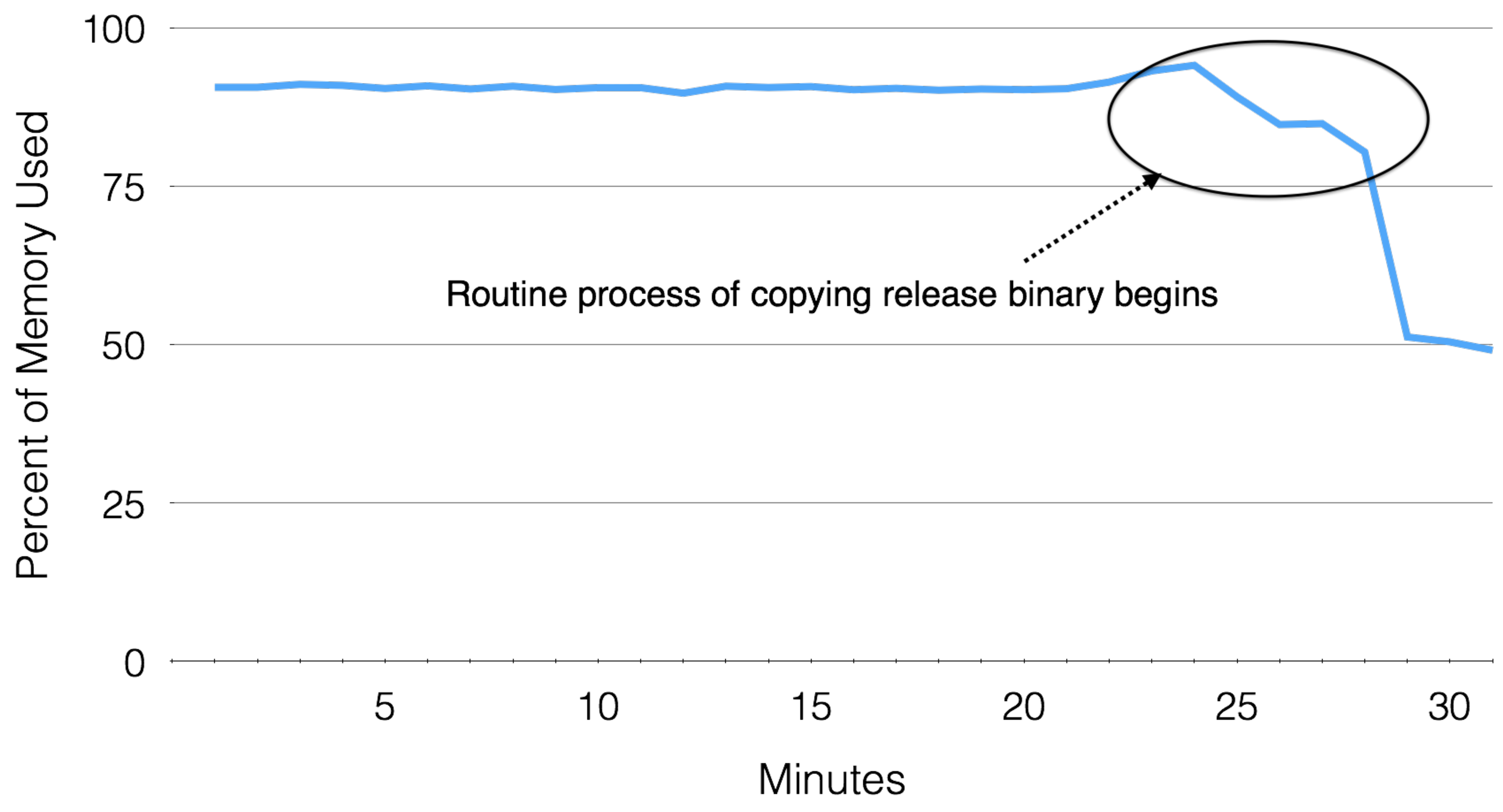
Gorilla Compression



[Pelkonen et al., 2015]

Enabling Gorilla Features

- Correlation Engine: "What happened around the time my service broke?"
- Charting: Horizon charts to see outliers and anomalies
- Aggregations: Rollups locally in Gorilla every couple of hours



[Pelkonen et al., 2015]

Gorilla Lessons Learned

- Prioritize recent data over historical data
- Read latency matters
- High availability trumps resource efficiency
  - Withstand single-node failures and "disaster events" that affect region
  - "[B]uilding a reliable, fault tolerant system was the most time consuming part of the project"
  - "[K]eep two redundant copies of data in memory"

[Pelkonen et al., 2015]

Assignment 4

- Work on Data Integration and Data Fusion
- Integrate travel datasets from different institutions (UN World Tourism Office, World Bank, OECD)
    - Integrate information with population
- Record Matching:
    - Which countries are the same?
- Data Fusion:
    - The receipts/expenditures
    - Country names

Test 2

- Upcoming… this coming Monday
- Similar format, but more emphasis on topics we have covered including the research papers

Specific Types of Data

Graphs: Social Networks

[P. Butler, 2010]

What is a Graph?

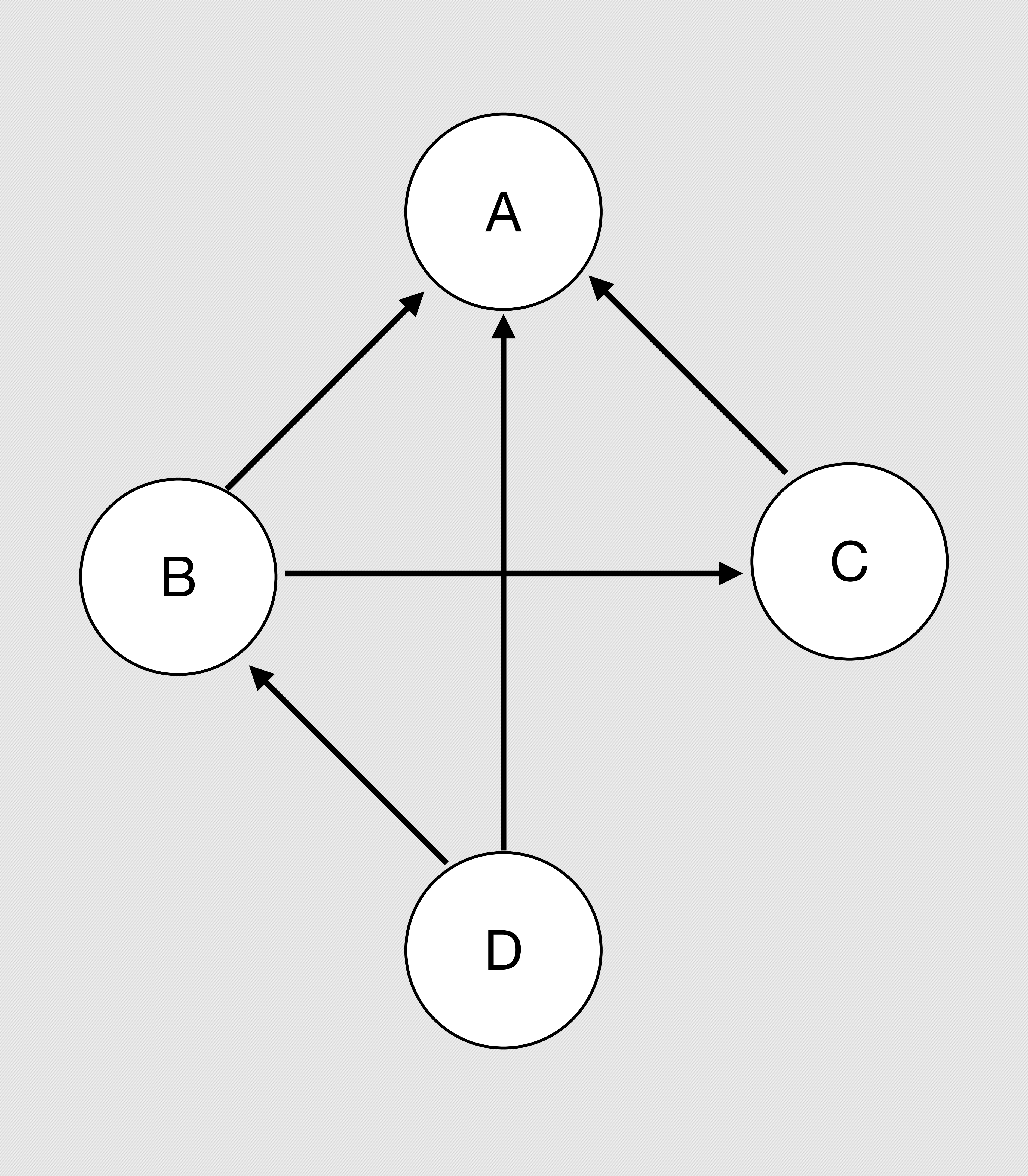
- An abstract representation of a set of objects where some pairs are connected by links.

Object (Vertex, Node)

Link (Edge, Arc, Relationship)

[M. De Marzi, 2012]

What is a Graph?

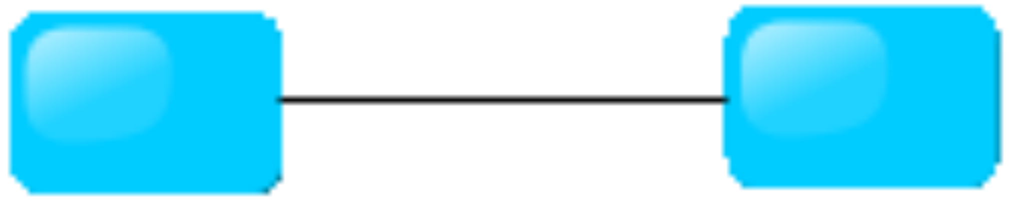


- In computing, a **graph** is an abstract **data structure** that represents set objects and their relationships as **vertices** and **edges/links**, and supports a number of graph-related **operations**
- Objects (nodes): `{A,B,C,D}`
- Relationships (edges): `{(D,B),(D,A),(B,C),(B,A),(C,A)}`
- Operation: shortest path from `D` to `A`

[K. Salama, 2016]

Different Kinds of Graphs

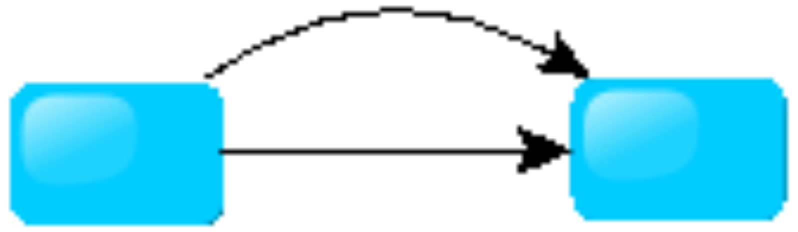
- Undirected Graph



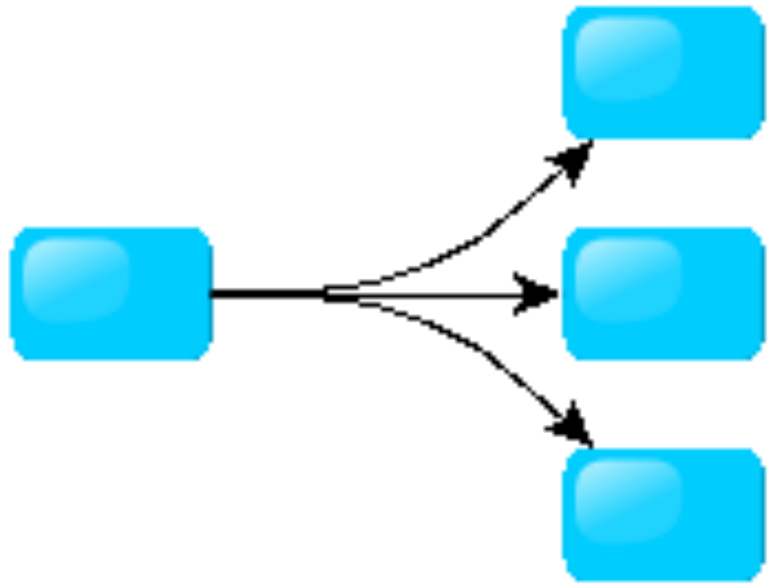
- Directed Graph

- Pseudo Graph

- Multi Graph



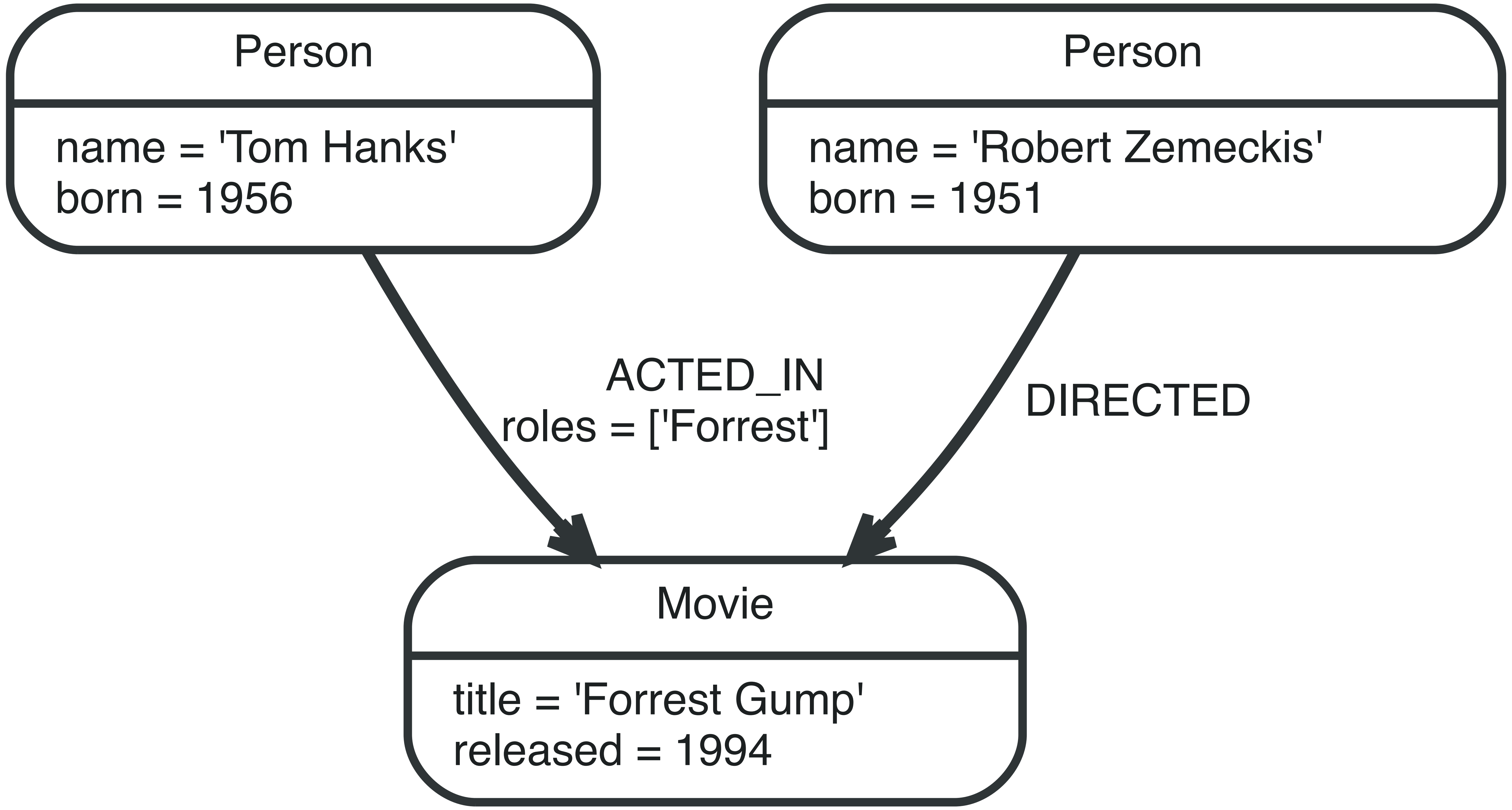
- Hyper Graph



[M. De Marzi, 2012]

Graphs with Properties

- Each vertex or edge may have properties associated with it
- May include identifiers or classes



[neo4j]

Types of Graph Operations

- Connectivity Operations:
  - number of vertices/edges, in- and out-degrees of vertices
  - histogram of degrees can be useful in comparing graphs
- Path Operations: cycles, reachability, shortest path, minimum spanning tree
- Community Operations: clusters (cohesion and separation)
- Centrality Operations: degree, vulnerability, PageRank
- Pattern Matching: subgraph isomorphism
  - can use properties
  - useful in fraud/threat detection, social network suggestions

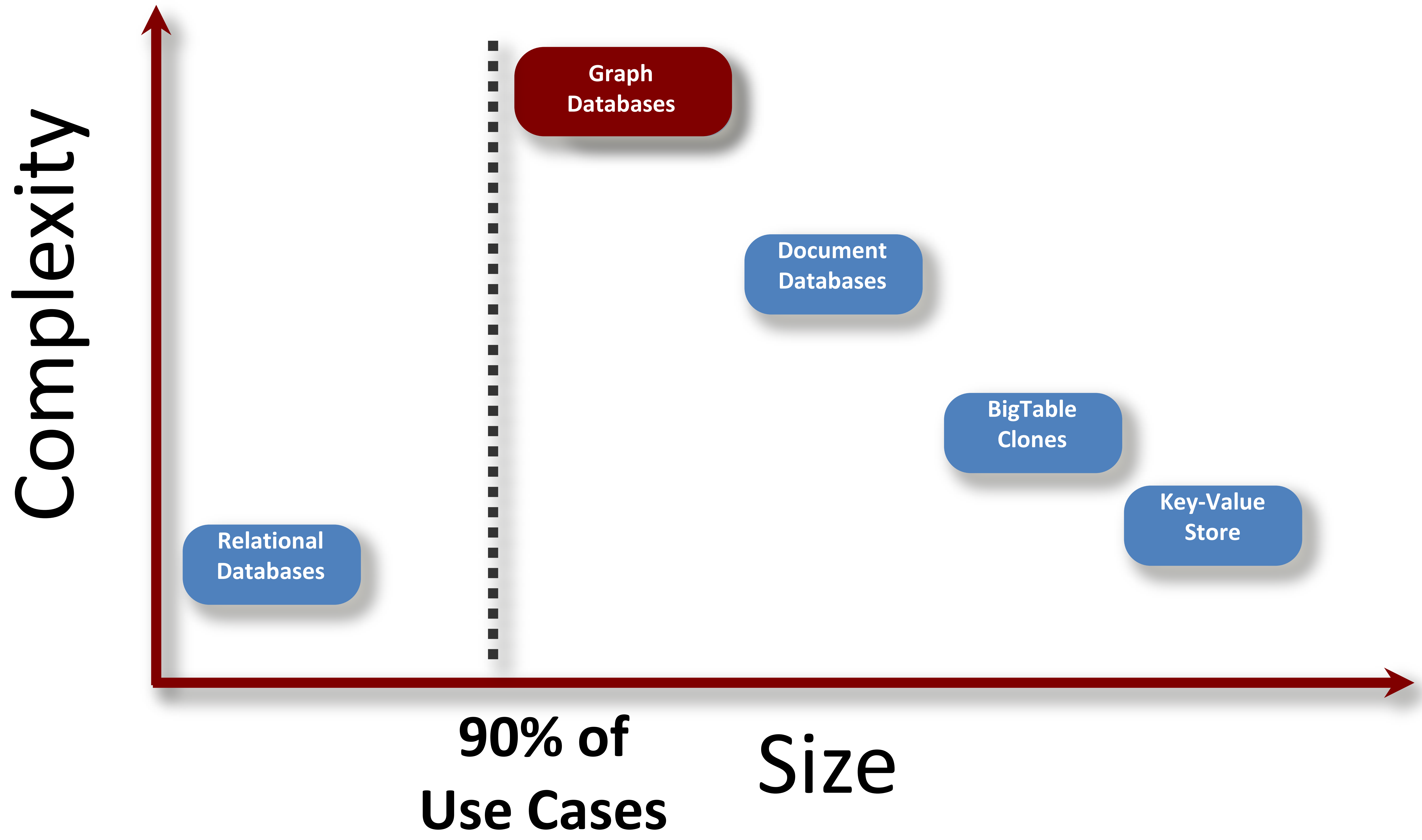
[K. Salama, 2016]

What is a Graph Database?

- A database with an explicit graph structure
- Each node knows its adjacent nodes
- As the number of nodes increases, the cost of a local step (or hop) remains the same
- Plus an Index for lookups

[M. De Marzi, 2012]

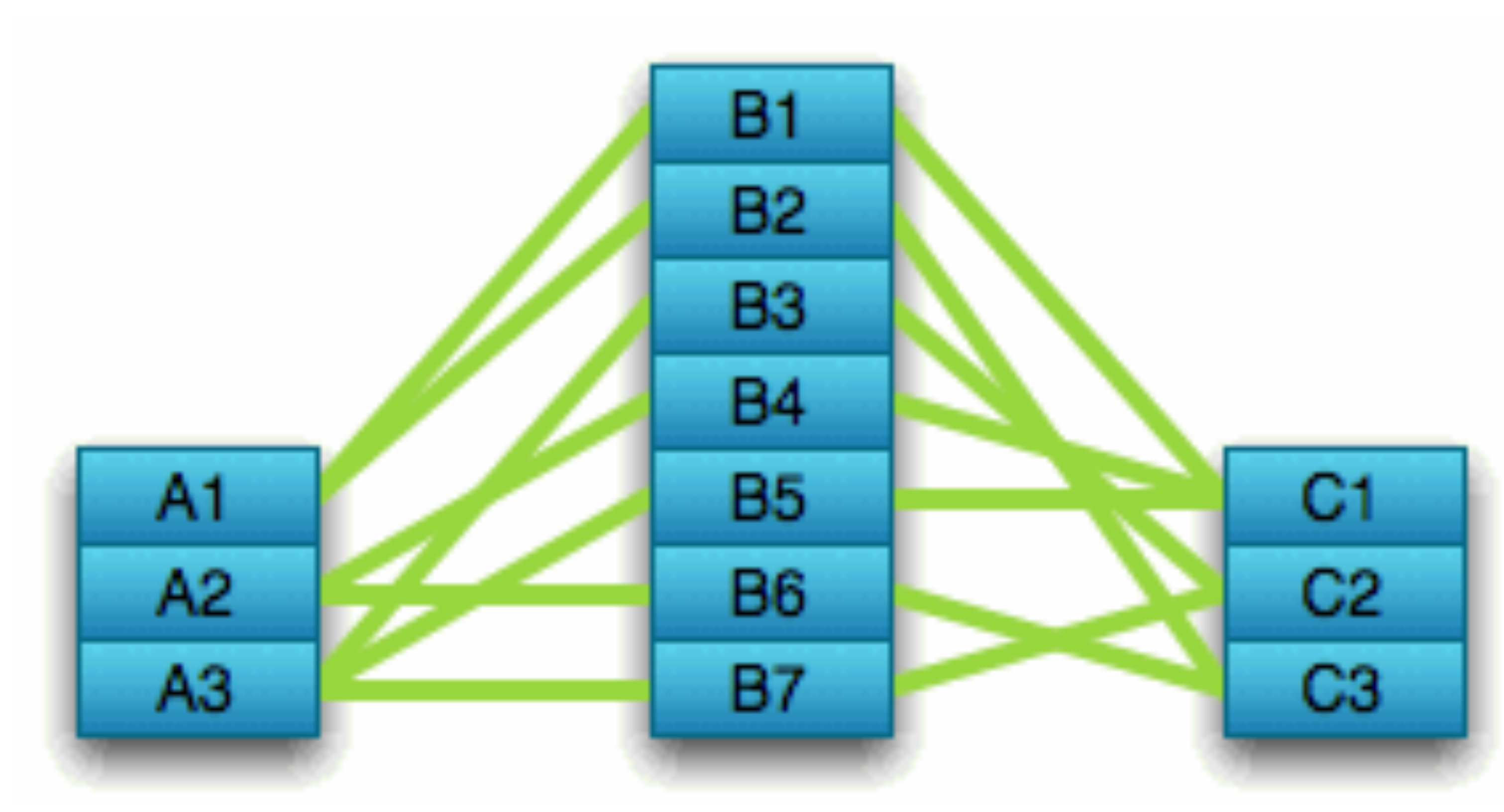
How do Graph Databases Compare?



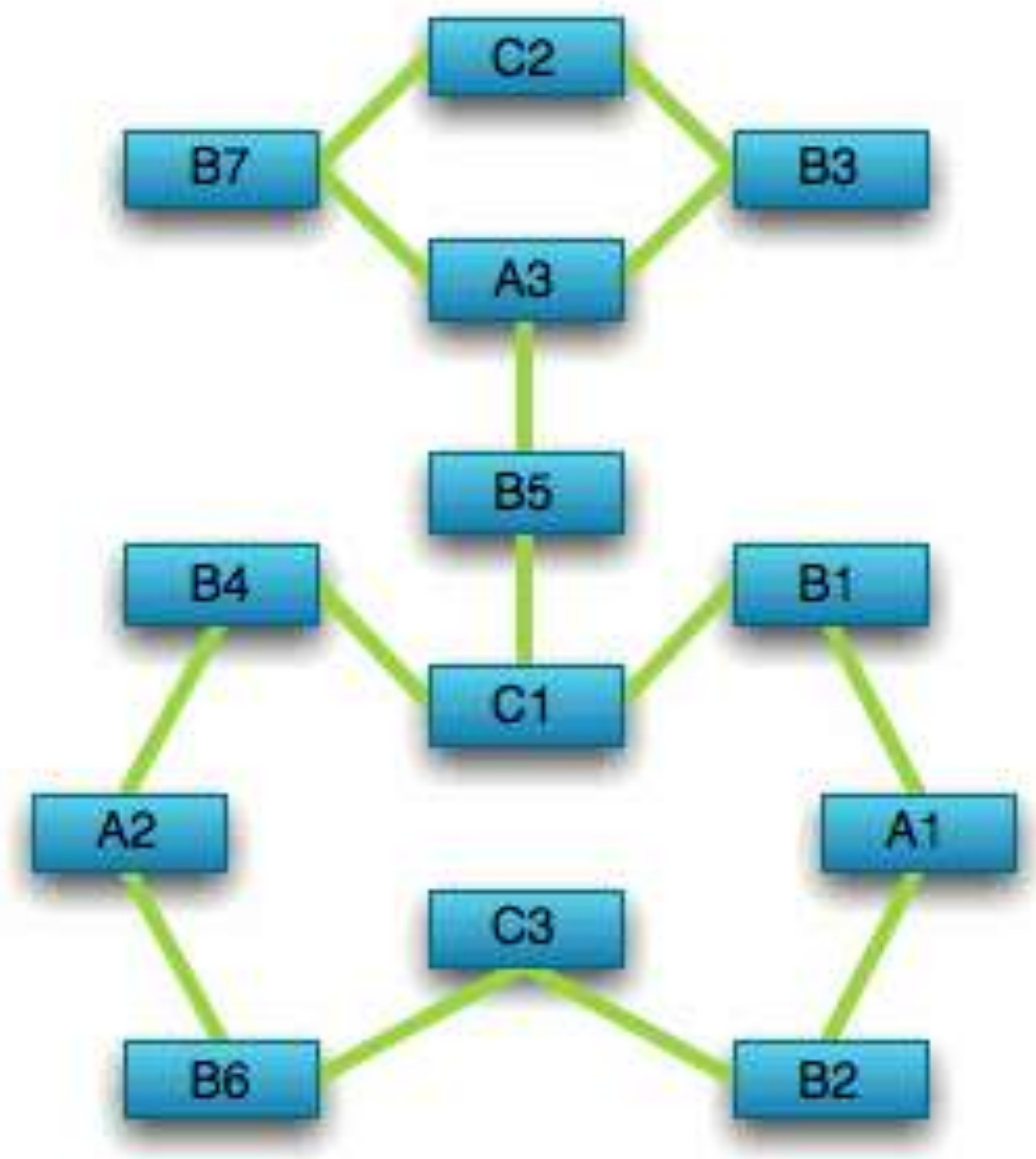
[M. De Marzi, 2012]

Graph Databases Compared to Relational Databases

Optimized for aggregation



Optimized for connections

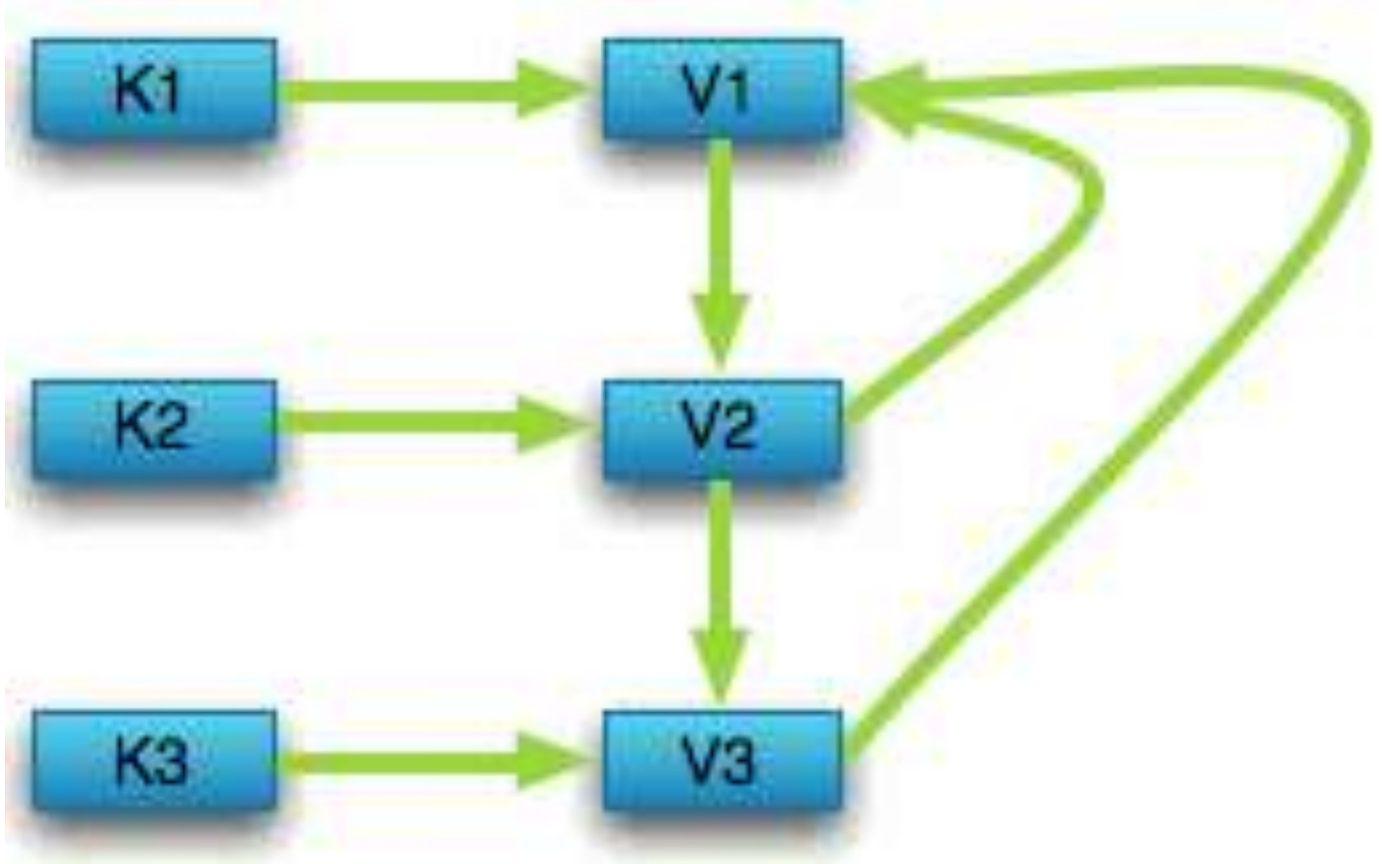


[M. De Marzi, 2012]

Graph Databases Compared to Key-Value Stores

Optimized for simple look-ups

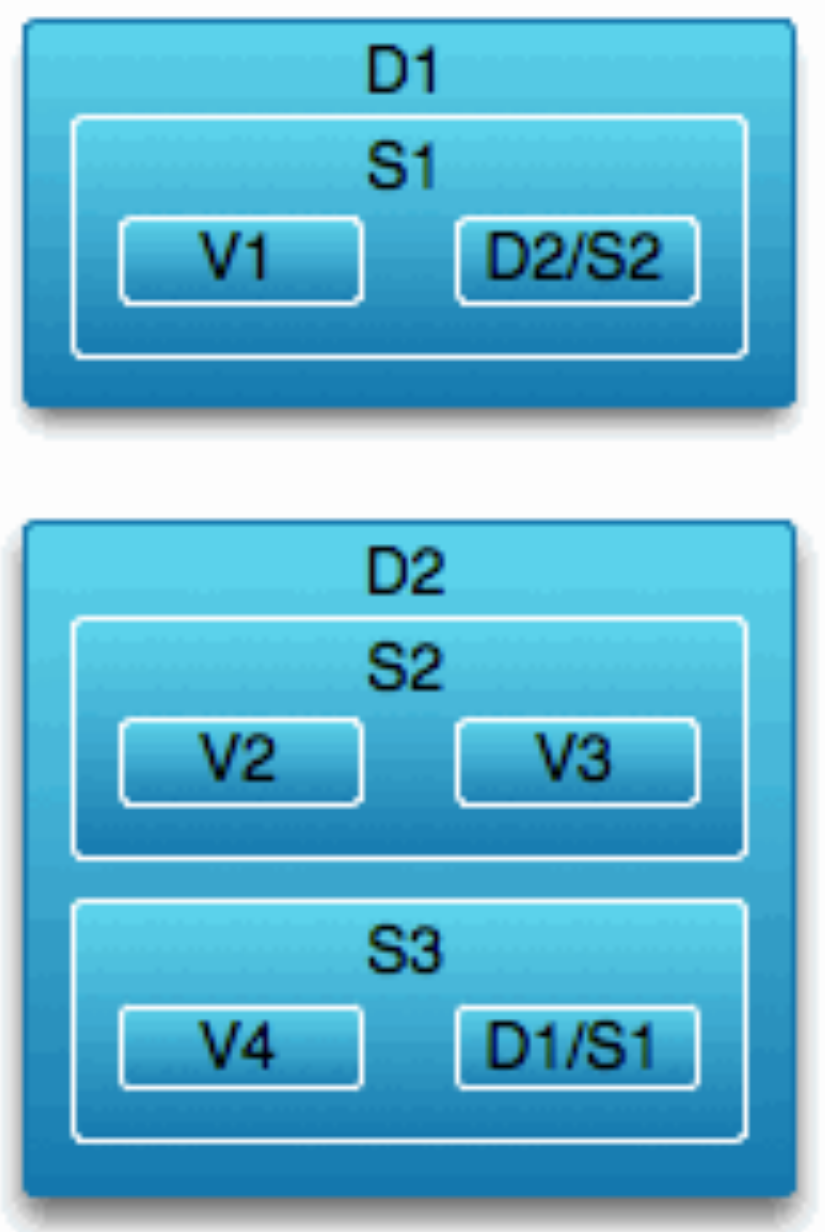
Optimized for traversing connected data



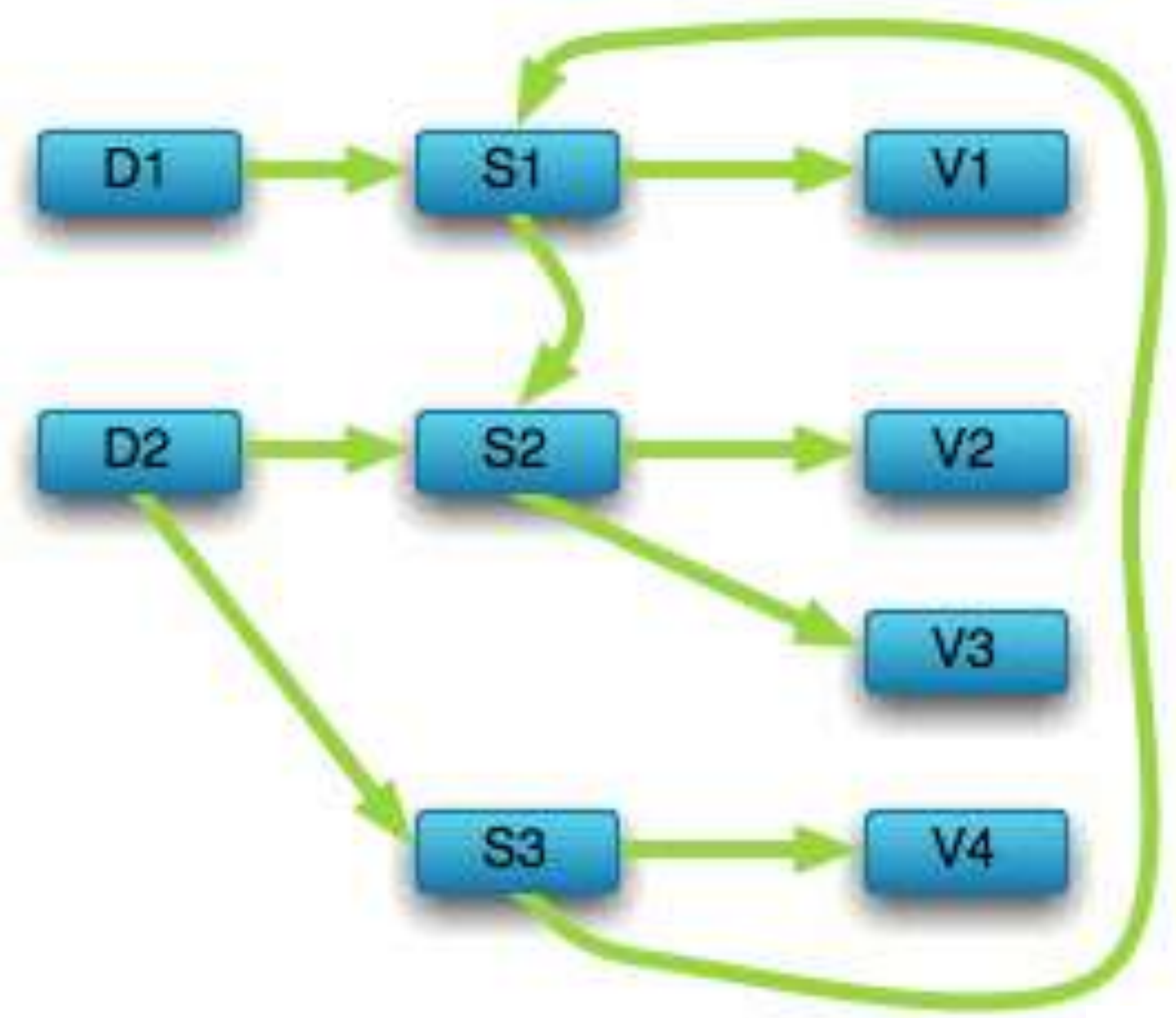
[M. De Marzi, 2012]

Graph Databases Compared to Document Stores

Optimized for “trees” of data



Optimized for seeing the forest and the trees, and the branches, and the trunks



[M. De Marzi, 2012]

The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing

S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, and M. T. Özsu

The Future is Big Graphs

S. Sakr et al

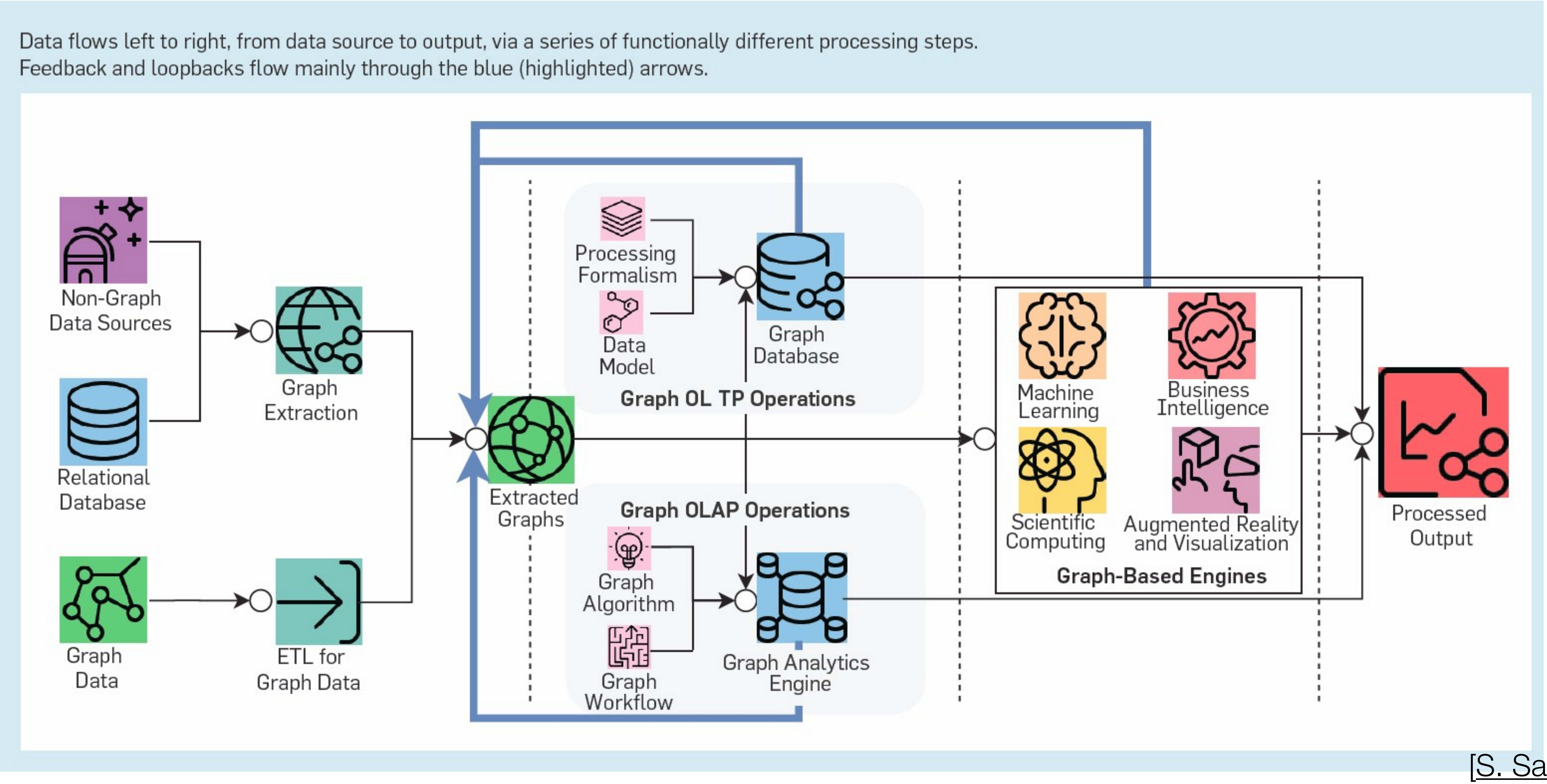
CACM

Insights for the Future of Graph Processing

- Graphs are ubiquitous abstractions enabling reusable computing tools for graph processing with applications in every domain.
- Diverse workloads, standard models and languages, algebraic frameworks, and suitable and reproducible performance metrics will be at the core of graph processing ecosystems in the next decade.

[S. Sakr et al.]

Pipeline for Graph Processing



[S. Sakr et al.]

Graph Databases

D. Lembo and R. Rosati

Why Graph Database Models?

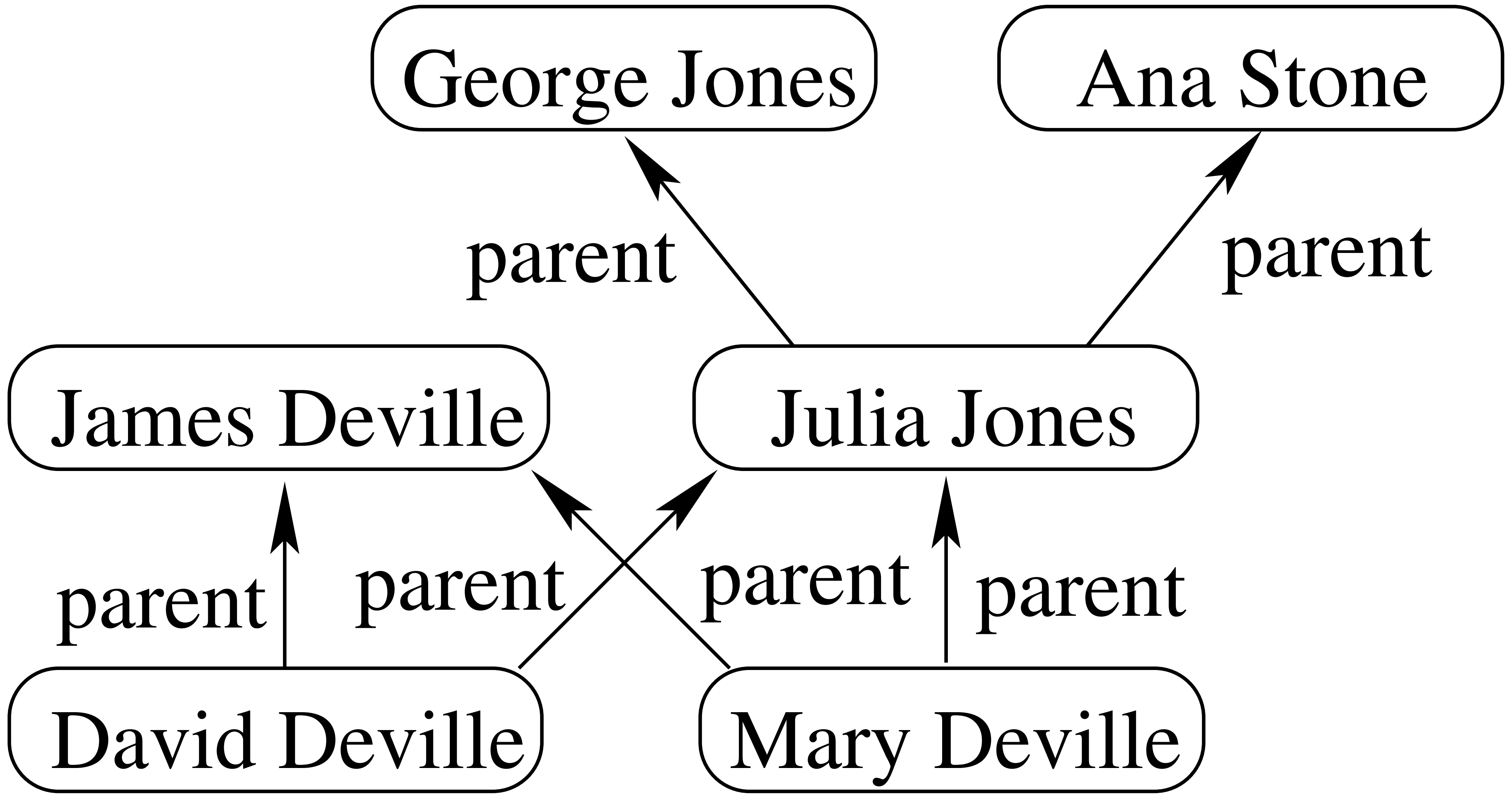
- Graphs has been long ago recognized as one of the most simple, natural and intuitive knowledge representation systems
- Graph data structures allow for a natural modeling when data has graph structure
- Queries can address direct and explicitly this graph structure
- Implementation-wise, graph databases may provide special graph storage structures, and take advantage of efficient graph algorithms available for implementing specific graph operations over the data

[R. Angles and C. Gutierrez, 2017]

Relational Model

NAME	LASTNAME
George	Jones
Ana	Stone
Julia	Jones
James	Deville
David	Deville
Mary	Deville

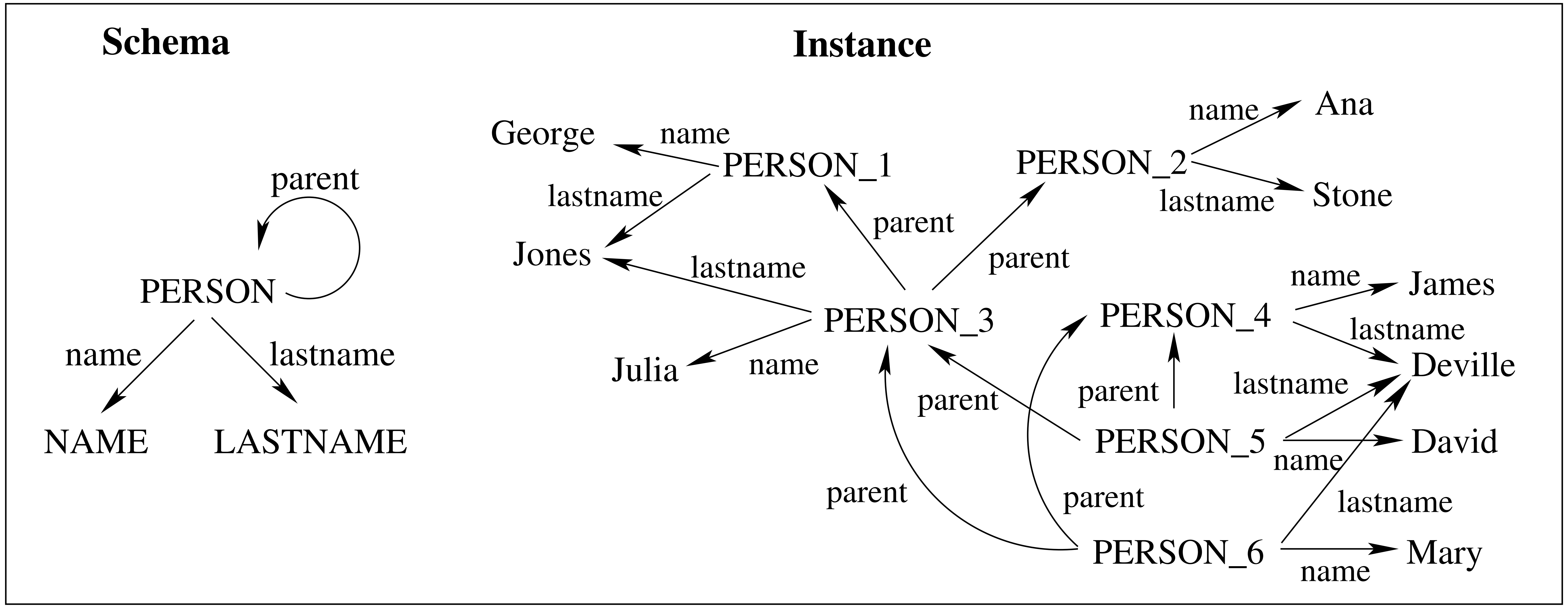
PERSON	PARENT
Julia	George
Julia	Ana
David	James
David	Julia
Mary	James
Mary	Julia



[R. Angles and C. Gutierrez, 2017]

Basic Labeled Model (Gram)

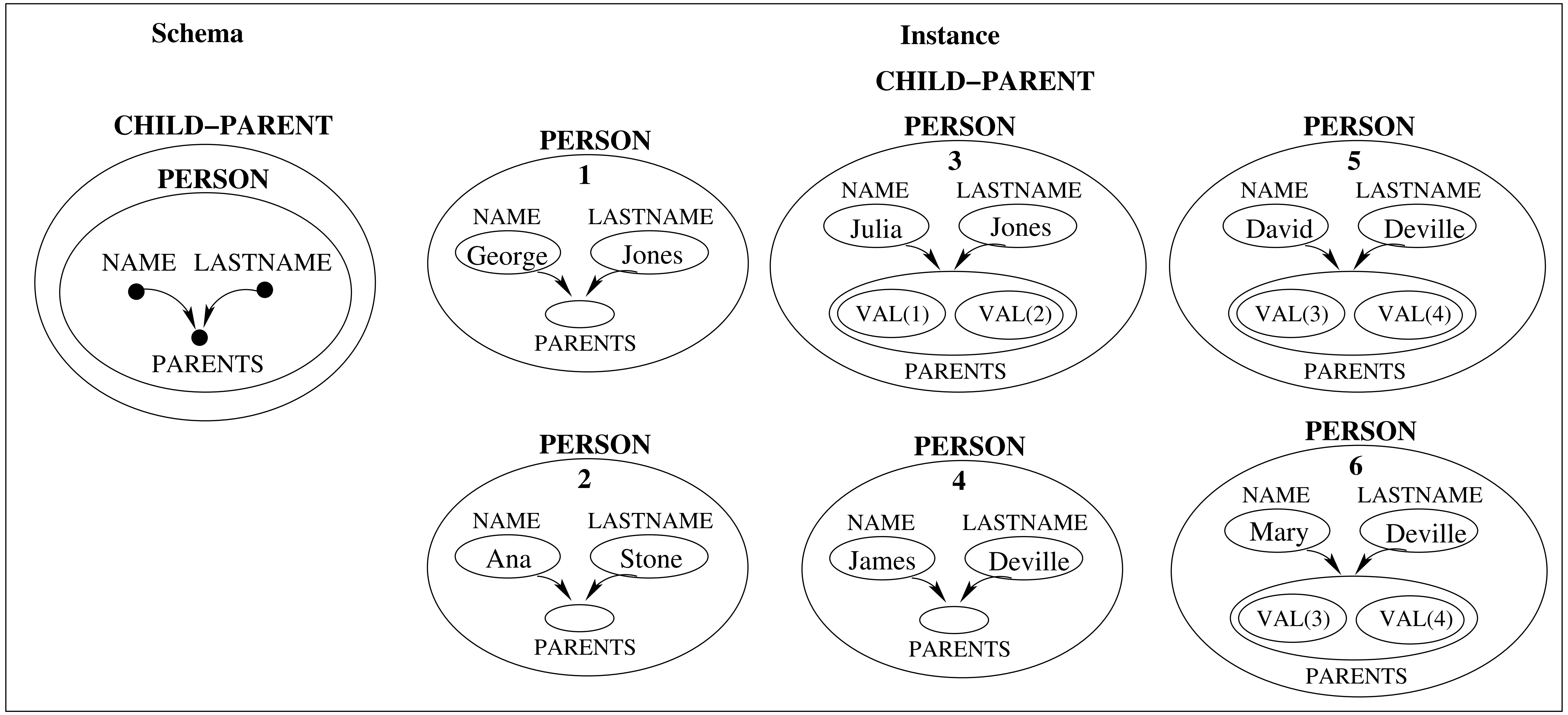
- Directed graph with nodes and edges labeled by some vocabulary
- Gram is a directed labeled multigraph
  - Each node is labeled with a symbol called a **type**
  - Each edge has assigned a label representing a **relation** between types



[R. Angles and C. Gutierrez, 2017]

Hypergraph Model (Groovy)

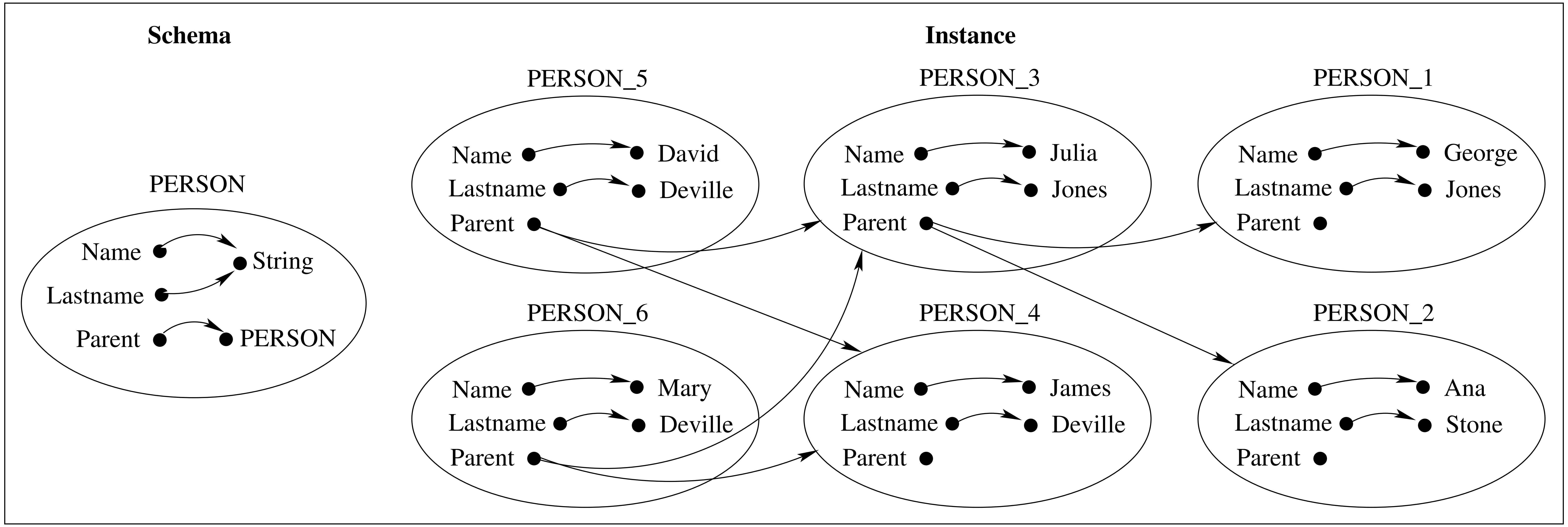
- Notion of edge is extended to **hyperedge**, which relates an arbitrary set of nodes
- Hypergraphs allow the definition of complex objects (undirected), functional dependencies (directed), object-ID and (multiple) structural inheritance



[R. Angles and C. Gutierrez, 2017]

Hypernode Model

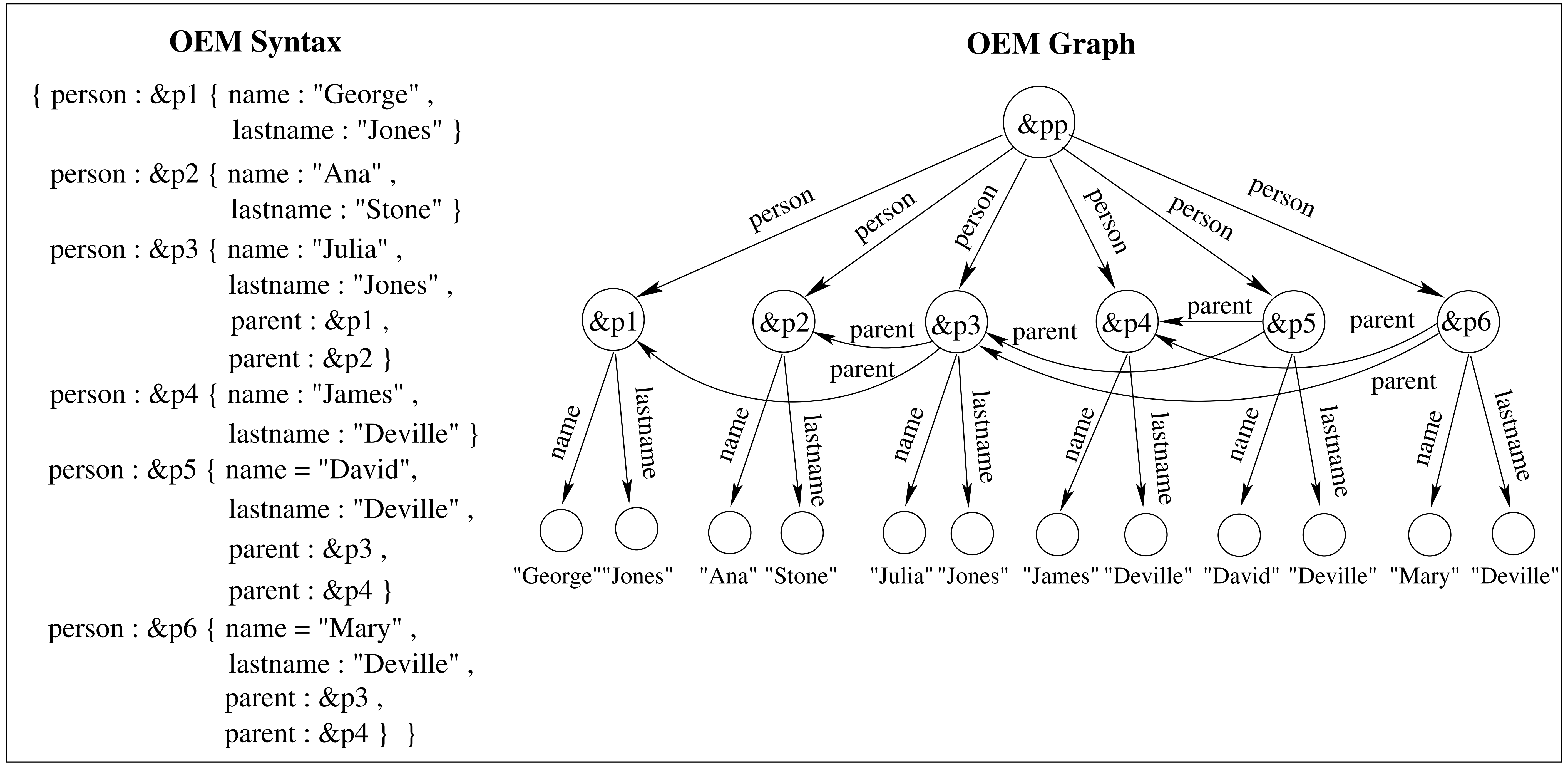
- Hypernode is a directed graph whose nodes can themselves be graphs (or hypernodes), allowing **nesting** of graphs
- **Encapsulates** information



[R. Angles and C. Gutierrez, 2017]

Semistructured (Tree) Model: (OEM Graph)

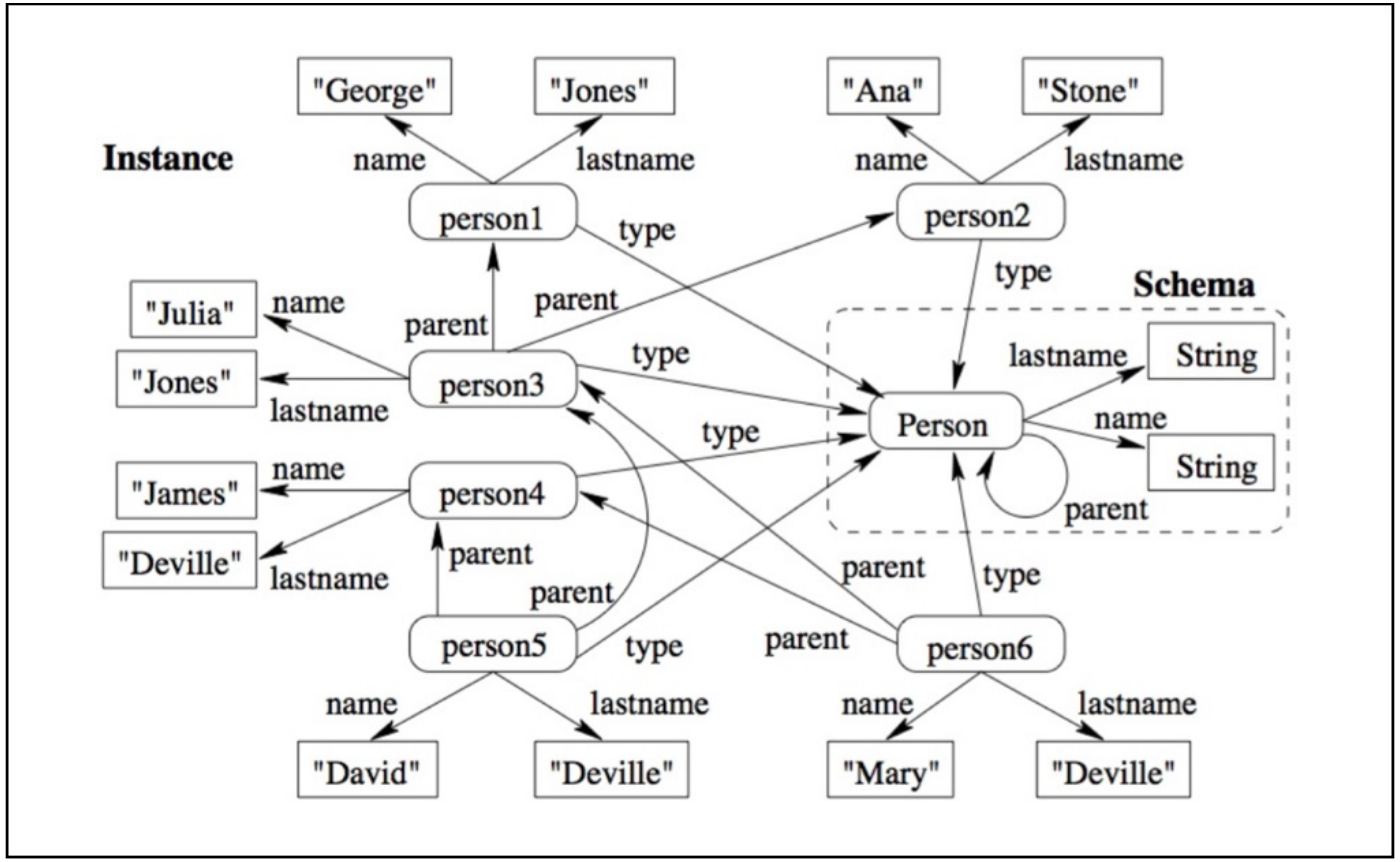
- "Self-describing" data like JSON and XML
- OEM uses pointers to data in the tree



[R. Angles and C. Gutierrez, 2017]

RDF (Triple) Model

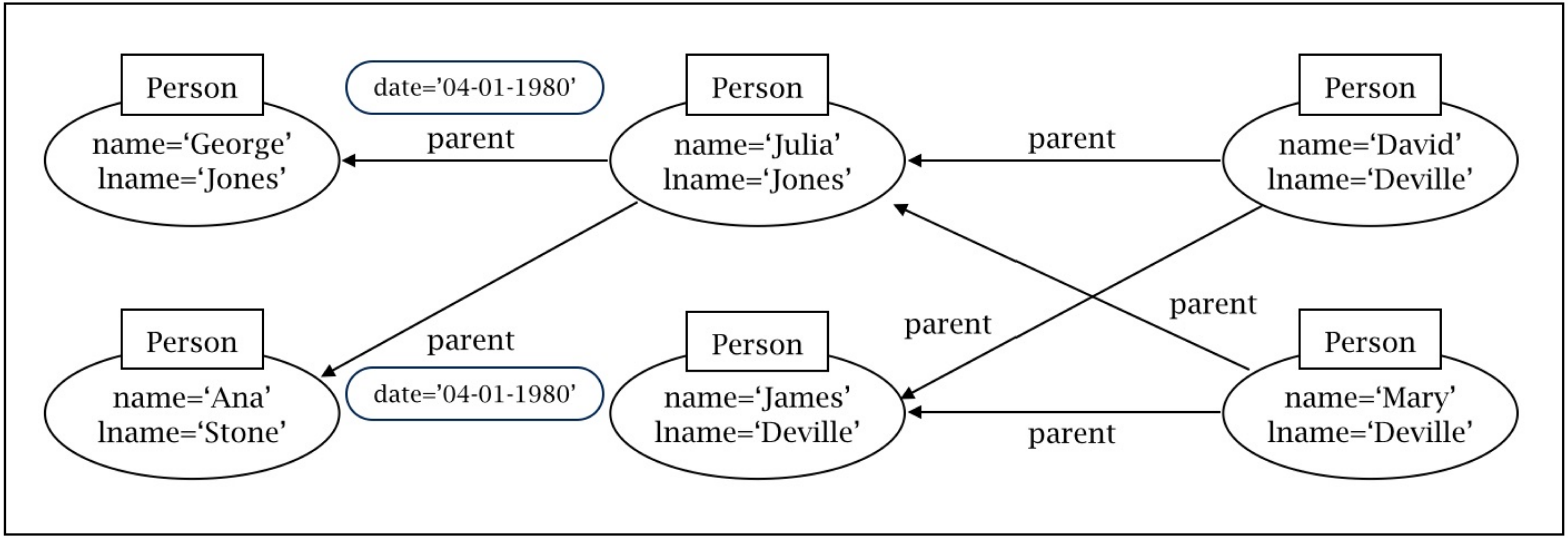
- Interconnect resources in an extensible way using graph-like structure for data
- Schema and instance are **mixed** together
- SPARQL to query
- Semantic web



[R. Angles and C. Gutierrez, 2017]

Property Graph Model (Cypher in neo4j)

- Directed, labelled, attributed multigraph
- Properties are **key/value pairs** that represent metadata for nodes and edges



[R. Angles and C. Gutierrez, 2017]

Types of Graph Queries

- Adjacency queries (neighbors or neighborhoods)
- Pattern matching queries (related to graph mining)
  - Graph patterns with structural extension or restrictions
  - Complex graph patterns
  - Semantic matching
  - Inexact matching
  - Approximate matching
- Reachability queries (connectivity)

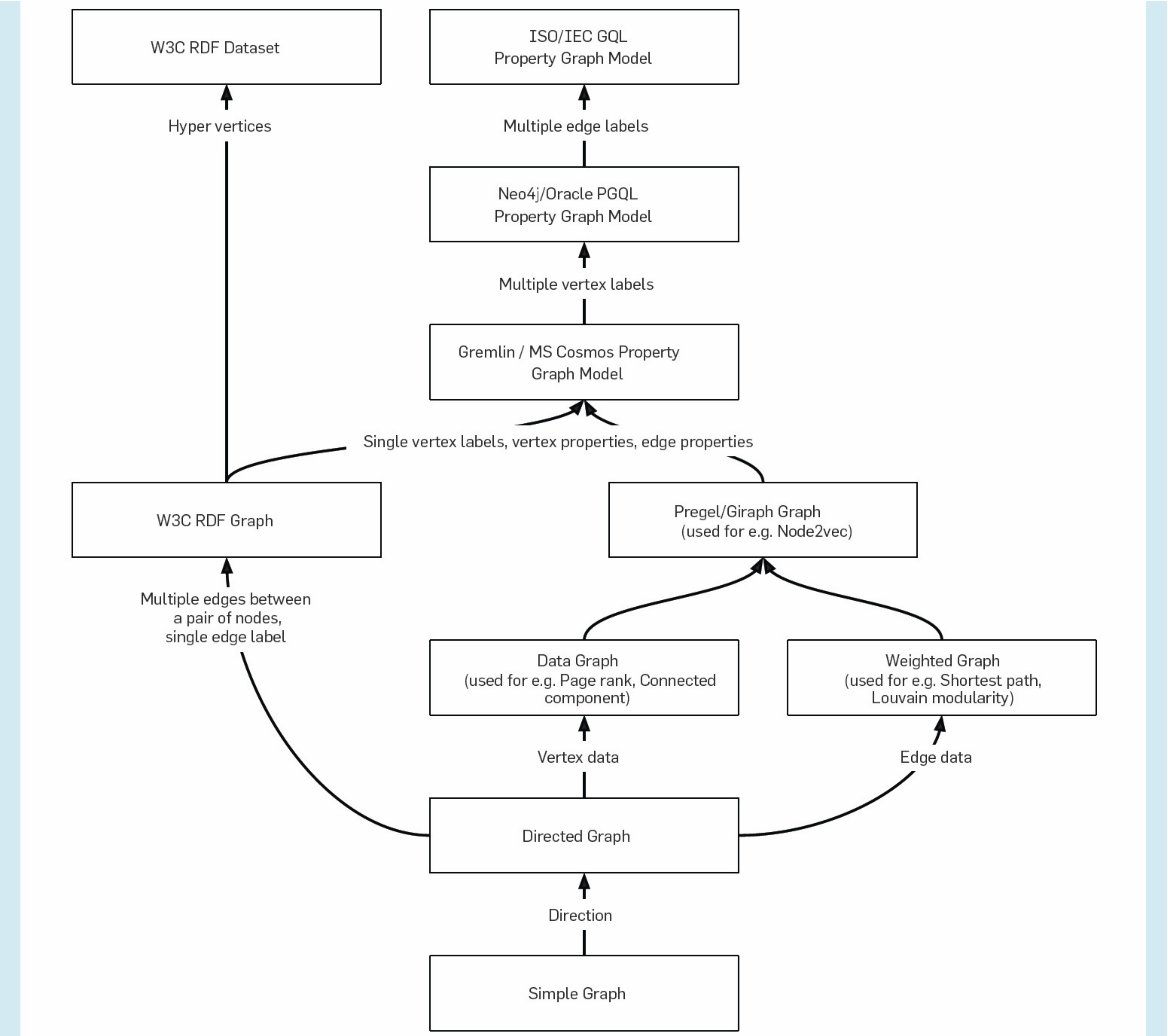
[R. Angles and C. Gutierrez, 2017]

Types of Graph Queries (continued)

- Analytical queries
  - Summarization queries
  - Complex analytical queries (PageRank, characteristic path length, connected components, community detection, clustering coefficient)

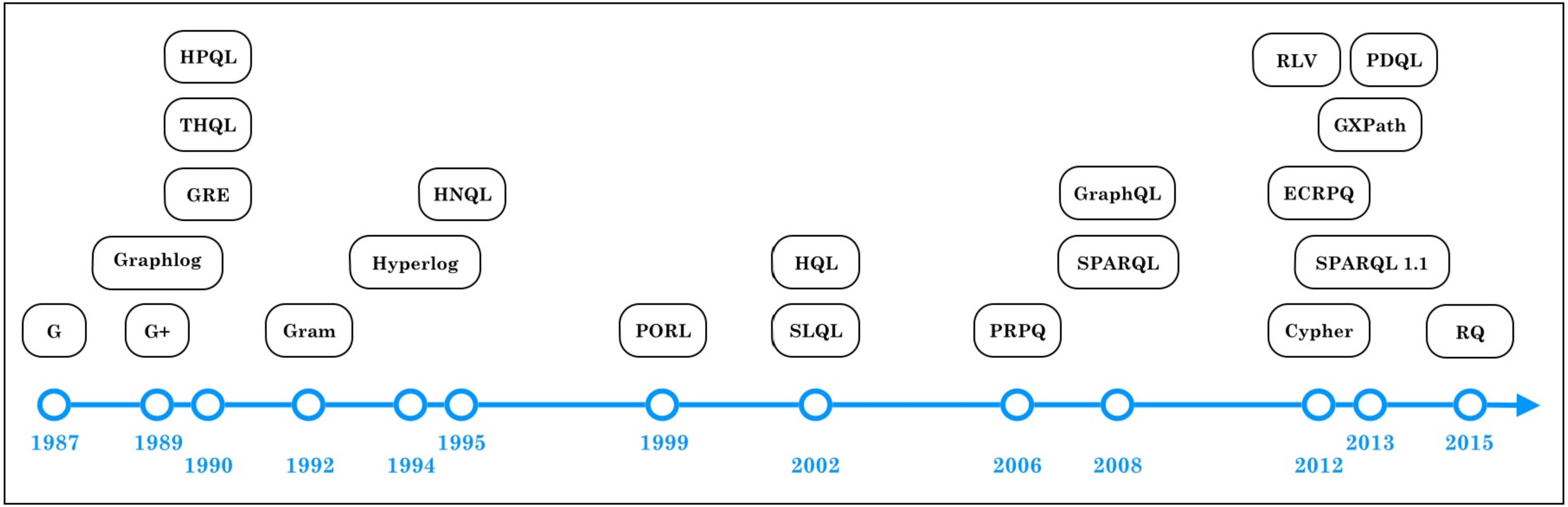
[R. Angles and C. Gutierrez, 2017]

Graph Structures



[S. Sakr et al.]

Graph Query Languages



[R. Angles and C. Gutierrez, 2017]

Cypher

- Implemented by neo4j system
- Expresses reachability queries via path expressions
  - `p = (a)-[:knows*]->(b)`: nodes from `a` to `b` following `knows` edges
- ```
  START x=node:person(name="John")
  MATCH (x)-[:friend]->(y)
  RETURN y.name
  ```

[R. Angles and C. Gutierrez, 2017]
  ```

SPARQL (RDF)

- Uses SELECT-FROM-WHERE pattern like SQL
- ```
  SELECT ?N
  FROM <http://example.org/data.rdf>
  WHERE { ?X rdf:type voc:Person . ?X voc:name ?N }
  ```

[R. Angles and C. Gutierrez, 2017]
  ```

Comparing Graph Database Systems: Features

Data Storage

Graph Database	Main memory	External memory	Backend Storage	Indexes
AllegroGraph	●	●		●
DEX	●	●		●
Filament	●		●	
G-Store		●		
HyperGraphDB	●	●	●	●
InfiniteGraph		●		●
Neo4j	●	●		●
Sones	●			●
vertexDB		●	●	

Operations/Manipulation

Graph Database	Data Definition Language	Data Manipulat. Language	Query Language	API	GUI
AllegroGraph	●	●	●	●	●
DEX				●	
Filament				●	
G-Store	●		●	●	
HyperGraphDB				●	
InfiniteGraph				●	
Neo4j				●	
Sones	●	●	●	●	●
vertexDB				●	

[R. Angles, 2012]

Comparing Graph Database Systems: Representation

Graph Data Structures

	Graphs				Nodes		Edges		
Graph Database	Simple graphs	Hypergraphs	Nested graphs	Attributed graphs	Node labeled	Node attribution	Directed	Edge labeled	Edge attribution
AllegroGraph	•				•		•	•	
DEX				•	•	•	•	•	•
Filament	•				•		•	•	
G-Store	•				•		•	•	
HyperGraphDB		•			•		•	•	
InfiniteGraph				•	•	•	•	•	•
Neo4j				•	•	•	•	•	•
Sones		•		•	•	•	•	•	•
vertexDB	•				•		•	•	

Entites & Relations

	Schema			Instance					
Graph Database	Node types	Property types	Relation types	Object nodes	Value nodes	Complex nodes	Object relations	Simple relations	Complex relations
AllegroGraph					•			•	
DEX	•		•	•	•		•	•	
Filament					•			•	
G-Store					•			•	
HyperGraphDB	•		•		•			•	•
InfiniteGraph	•		•	•	•		•	•	
Neo4j				•	•		•	•	
Sones					•			•	•
vertexDB					•			•	

[R. Angles, 2012]

Comparing Graph Database Systems: Queries

Query Support

	Type			Use		
Graph Database	Query Lang.	API	Graphical Q. L.	Retrieval	Reasoning	Analysis
AllegroGraph	○	●	●	●	●	●
DEX		●		●		●
Filament		●		●		
G-Store	●			●		
HyperGraphDB		●		●		
InfiniteGraph		●		●		
Neo4j	○	●		●		
Sones	●		●	●		●
vertexDB		●		●		

Types of Queries

	Adjacency		Reachability				
Graph Database	Node/edge adjacency	k-neighborhood	Fixed-length paths	Regular simple paths	Shortest path	Pattern matching	Summarization
Allegro	●		●			●	
DEX	●		●	●	●	●	
Filament	●		●			●	
G-Store	●		●	●	●	●	
HyperGraph	●					●	
Infinite	●		●	●	●	●	
Neo4j	●		●	●	●	●	
Sones	●					●	
vertexDB	●		●	●		●	

[R. Angles, 2012]

The (sorry) State of Graph Database Systems

Peter Boncz

Keynote, EDBT-ICDT 2022