Graph Analytics Course

Master Artificial Intelligence & Society
2026-2027

Dauphine UNIVERSITÉ PARIS SCIENCES & LETTRES

General Information

- Graph Processing (Dario Colazzo, 12h) and Graph Databases and Querying (Joyce El Haddad, 12h)

- Joyce El Haddad, Associate Professor in Computer Science, Data Science Team, Mining Software Repositories and Processes, Deputy Director of LAMSADE CNRS UMR 7243

- Contact me: joyce.elhaddad@dauphine.fr

- Course material: https://joyceelhaddad.github.io/

General Information

- Graph Databases and Querying : 3 x lectures with a hands-on labs
  - Lecture 1: *Foundations of Graph Databases*. Introduces graph theory and positions graph databases within the broader landscape of database technologies (relational, document, key-value, column-family). Introduces Neo4j.
    Lab 1: instal Neo4j and explore a first graph dataset.
  - Lecture 2: *Querying Graphs with Cypher*. Introduces Cypher, Neo4j's declarative query language, covering data creation, pattern matching, and data modification.
    Lab2: build a graph and practice the full range of basic Cypher operations.
  - Lecture 3: *Advanced Querying with Cypher*. Covers advanced traversal techniques and query composition.
    Lab3: consolidate skills through applied exercises.

General Information

- Assessment Methods:
  - Exam
  - Project

- Grade Calculation
  - Final grade = 0.7 x Exam + 0.3 x Project

Graph Analytics Course

Graph Databases and Querying

Master Artificial Intelligence & Society
2026-2027

Dauphine UNIVERSITÉ PARIS SCIENCES & LETTRES

Agenda

- Lecture 1: Foundations of Graph Databases
- Lecture 2: Querying Graphs with Cypher
- Lecture 3: Advanced Querying with Cypher

Graph Analytics Course

Graph Databases and Querying

Foundations of Graph Databases

Master Artificial Intelligence & Society
2026-2027

Dauphine UNIVERSITÉ PARIS SCIENCES & LETTRES

Agenda

1. Introduction
2. Types of Databases
3. Graph theory foundations
4. From graph theory to graph databases
5. Introducing Neo4j

Introduction

- Data ?
  - Informal definitions :
    1. Collection of measured and recorded values
    2. Set of values of subjects (entities) with respect to variables (attributes)

- Actors [IMDb Non-Commercial Dataset, 2026]

Nconst (id)	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
nm0000001	Fred Astaire	1899	1987	actor, miscellaneous, producer	tt0050419,tt0072308,tt0027125,tt0025164
nm0000002	Lauren Bacall	1924	2014	actress, soundtrack, archive_footage	tt0037382,tt0075213,tt0038355,tt0117057
nm0000003	Brigitte Bardot	1934	\N	actress, music_department, producer	tt0057345,tt0049189,tt0056404,tt0054452
nm0000004	John Belushi	1949	1982	actor, writer, music_department	tt0072562,tt0077975,tt0080455,tt0078723

Introduction

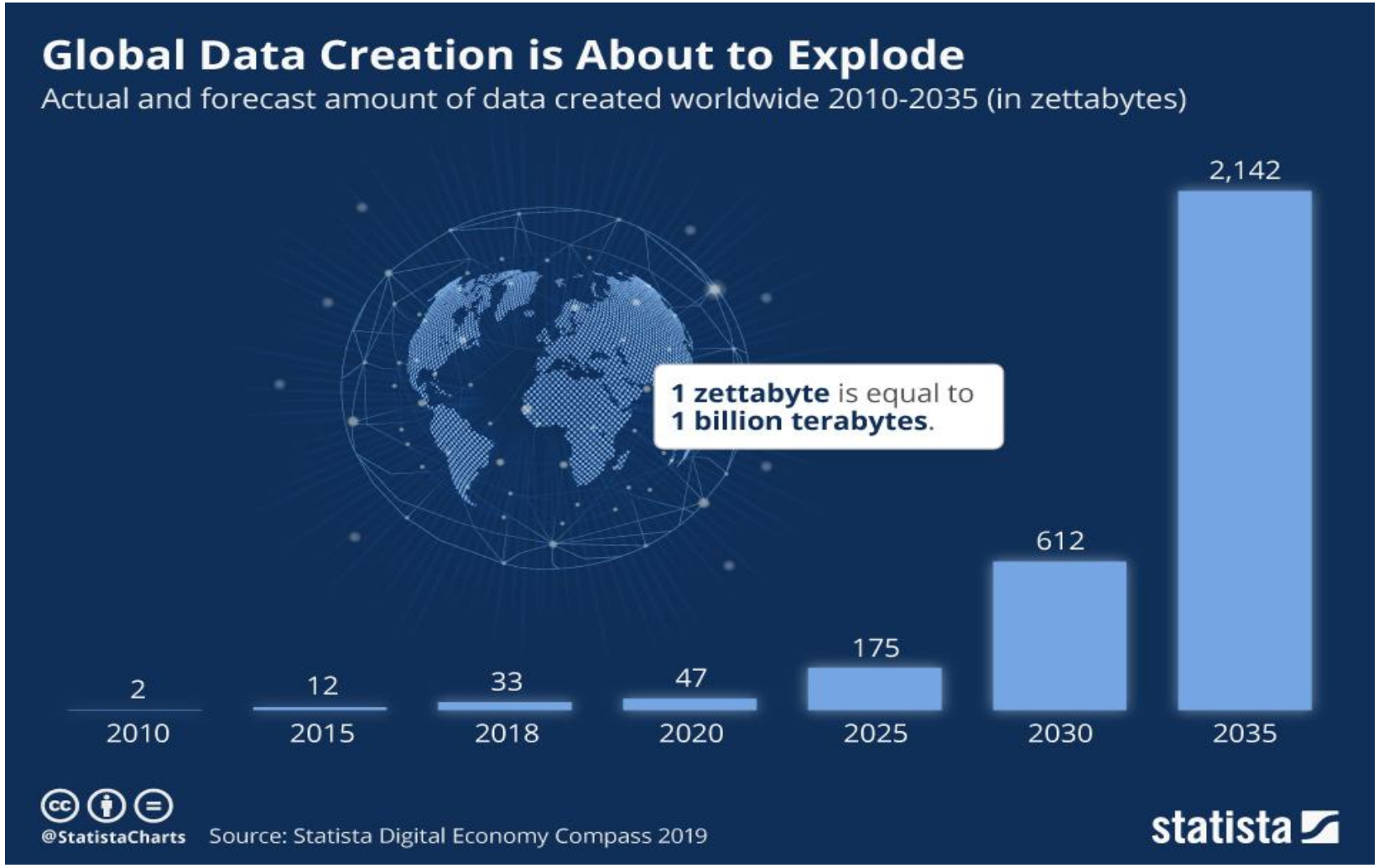
- Actors [IMDb Non-Commercial Dataset, 2026]

Nconst (id)	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
nm0000005	Ingmar Bergman	1918	2007	writer, director, actor	tt0050986,tt0069467,tt0050976,tt0083922
nm0000006	Ingrid Bergman	1915	1982	actress, producer, soundtrack	tt0034583,tt0038109,tt0036855,tt0038787
nm0000007	Humphrey Bogart	1899	1957	actor, producer, miscellaneous	tt0034583,tt0043265,tt0037382,tt0033870
nm0000008	Marlon Brando	1924	2004	actor, director, writer	tt0078788,tt0068646,tt0047296,tt0044081
nm0000009	Richard Burton	1925	1984	actor, producer, director	tt0061184,tt0087803,tt0059749,tt0057877

- Entities and attributes
  - entities: actors
  - attributes: primaryName, birthYear, deathYear, etc.

Introduction

- Trends in Data



Introduction

- Trends in Data
  - IA impact
    - ➢ Trend 1: Introduction of AI into data analytics.

      AI algorithms used to analyze vast amounts of data in real-time, providing valuable insights and predictions for businesses and organizations.

    - ➢ Trend 2: Vector databases for storage

      Optimized for storing and querying vectors (sequences of numbers or embeddings). They can perform similarity searches and help with understanding complex patterns and relationships in data.

Introduction

- Trends in Data
  - Distribution and Performance
    - Trend 3: Edge computing enables federated analytics.

      Processing and analysis of data at or near the data's source (instead of sending it to a centralized location).
    - Trend 4: Data mesh architecture

      An approach that involves decentralizing data ownership and management while still maintaining a unified view of the data across the organization.

Introduction

- Trends in Data
  - Accessibility and Flexibility
    - Trend 5: Data as a Service (DaaS)

      One of the top trends in data for businesses that are looking to outsource their data management and analytics.

    - Trend 6: Data democratization

      It is becoming increasingly important for businesses to ensure that everyone in their organization has access to the right tools and resources to make sense of this data.

Introduction

- Trends in Data
  - Engineering
    - ➢ Trend 7: Analytics engineering

      Optimize data pipelines and ensure efficient data analysis.

    - ➢ Trend 8: Data contracts

      Agreements that define the rules and expectations for how data is collected, managed, and used within an organization.

Introduction

- Database ?
  - A persistent and structured collection of related data.

- Important aspects
  - persistent: data is stored permanently (on a hard drive or server) and does not disappear when the system or application is turned off.
  - structured: data is organized according to specific rules (like tables with rows and columns) so it can be easily searched and managed.
  - related data: information is logically connected (a movie database links actors, movies, directors, and studios together).

Introduction

- Example
  - multiple entities : actors, movies, roles
  - attributes : can be specific to some entities (first_name, title,…)
  - relation between entities

movie_id	title	release_year	genre
1	Inception	2010	Sci_Fic
2	Titanic	1997	Romance
3	Pulp Fiction	1994	Crime

id	first_name	last_name	Birth-date
101	Leonardo	DiCaprio	1974-11-11
102	Christopher	Nolan	1970-07-30
103	Christian	Bale	1974-01-30
104	Heath	Ledger	1979-04-04
105	John	Travolta	1954-02-18

actor_id	movie_id	role
101	2	Jack Dawson
105	3	Vincent Vega

Introduction

- Database management system (DBMS) ?

- Definition : A DBMS is a software system used to manage databases providing
  - access to the data while maintaining integrity, security and efficiency
  - a high level API to build applications on the databases

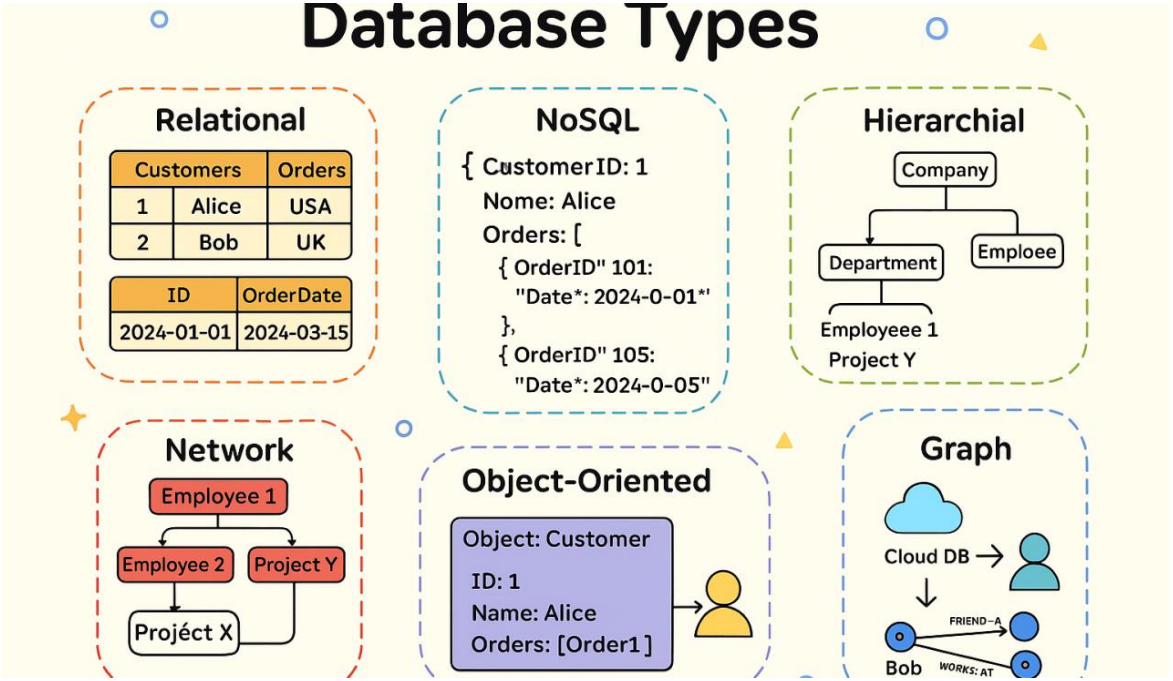
- Architecture
  - DBMS follow the client-server model
  - The DBMS itself the server
  - Clients connect concurrently to the server to access the databases

Agenda

1. Introduction
2. Types of Databases
3. Graph theory foundations
4. From graph theory to graph databases
5. Introducing Neo4j

Introduction

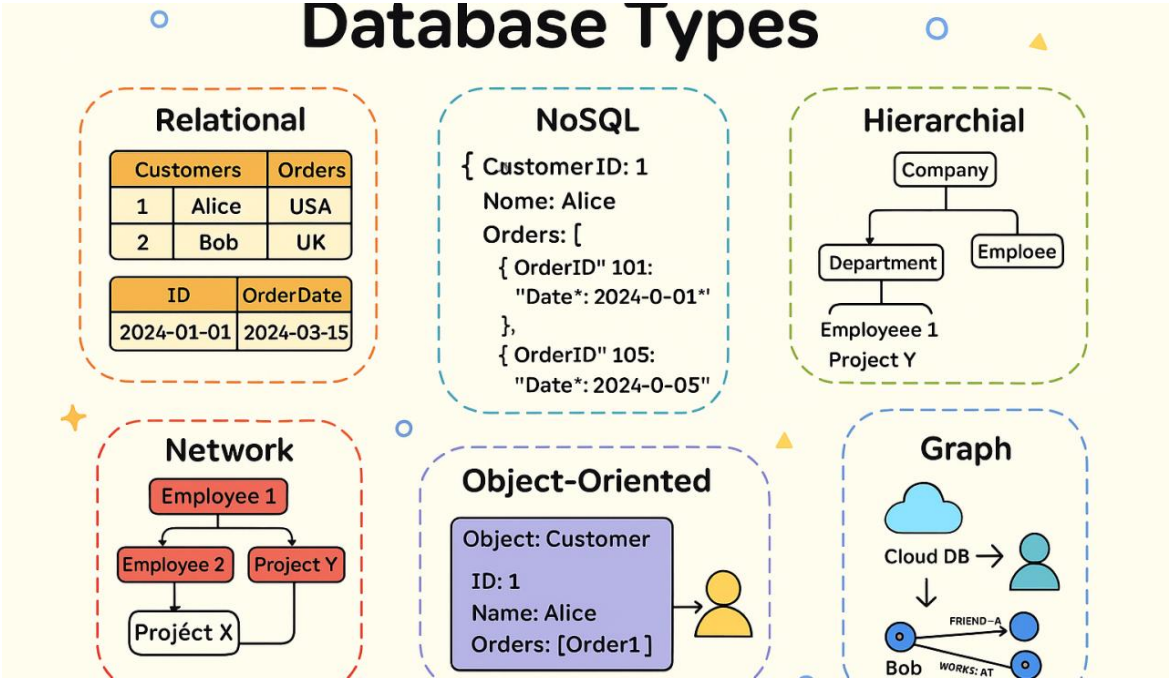
- Types of DBMS
  - old approaches: hierarchical DBMS and network DBMS
  - current standard: relational DBMS
    - based on a relational data model
    - typical based on SQL (Structured Query Language, sequel)
  - niche approach: object-oriented DBMS



https://www.linkedin.com/pulse/types-databases-complete-guide-illustrations-bharath-mashetty-damtf/

Introduction

- Types of DBMS
  - large scale or unstructured data: NoSQL (not-only SQL)
    - ➢ Column databases
    - ➢ Document databases
    - ➢ Graph databases



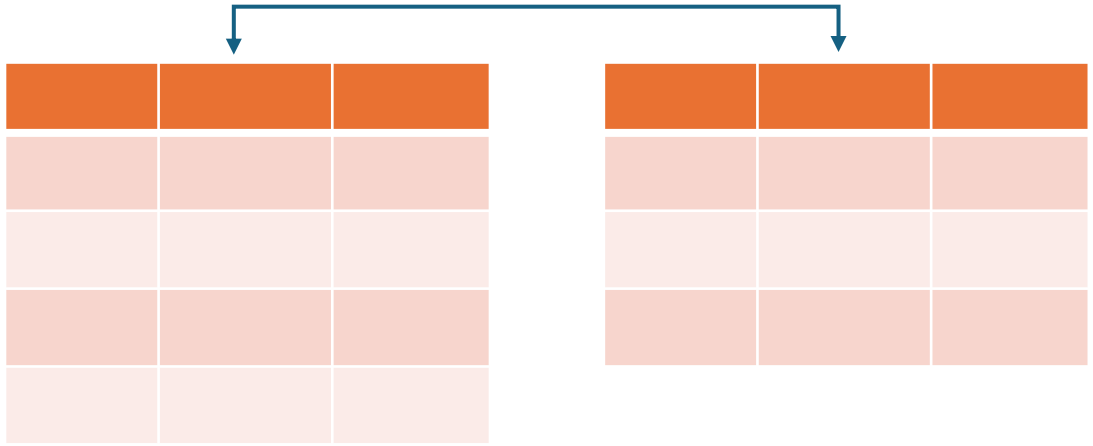
https://www.linkedin.com/pulse/types-databases-complete-guide-illustrations-bharath-mashetty-damtf/

Types of Databases

- Why Database Choice Matters ?
  - Application performance frequently depends on the database you select for a specific task
  - As a developer, you must choose based on your use case and requirements

- What we'll cover for each type ?
  - What it is and how it works
  - When to choose it
  - Use-case examples

Types of Databases



- Relational Database
  - Originally developed by IBM in the 1970s
  - Think of it as a collection of spreadsheet files that organize, manage, and relate data

Customer

Client_id	name	address
1	Alice	2 rue de l'estrapade
2	Bob	24 rue de la paix
3	John	37 rue de l'est
4	Olivia	4 place de la nation
5	David	15 rue necker

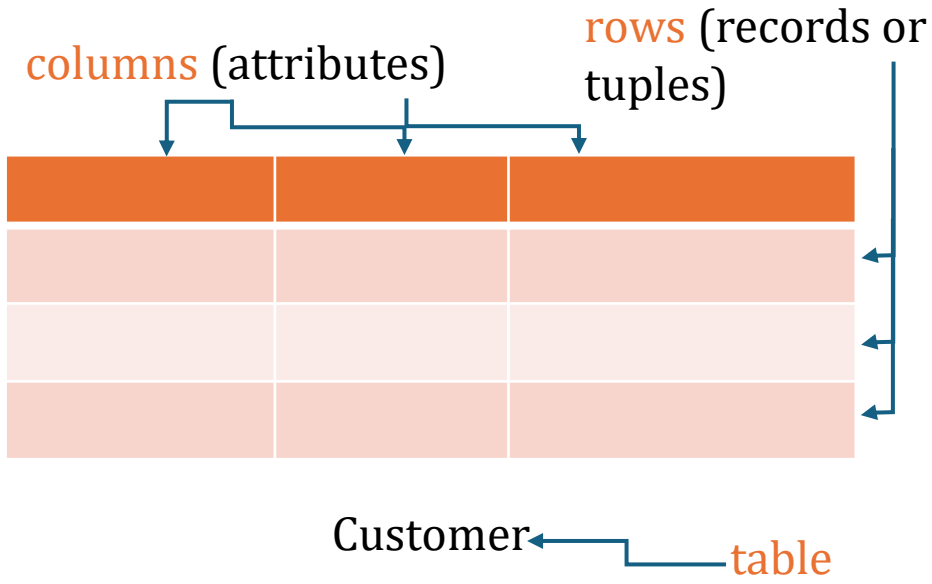
Order

Order_id	Client_id	Order_date
1001	3	09/23/2026
1002	1	10/09/2026
1003	4	12/09/2026

Types of Databases

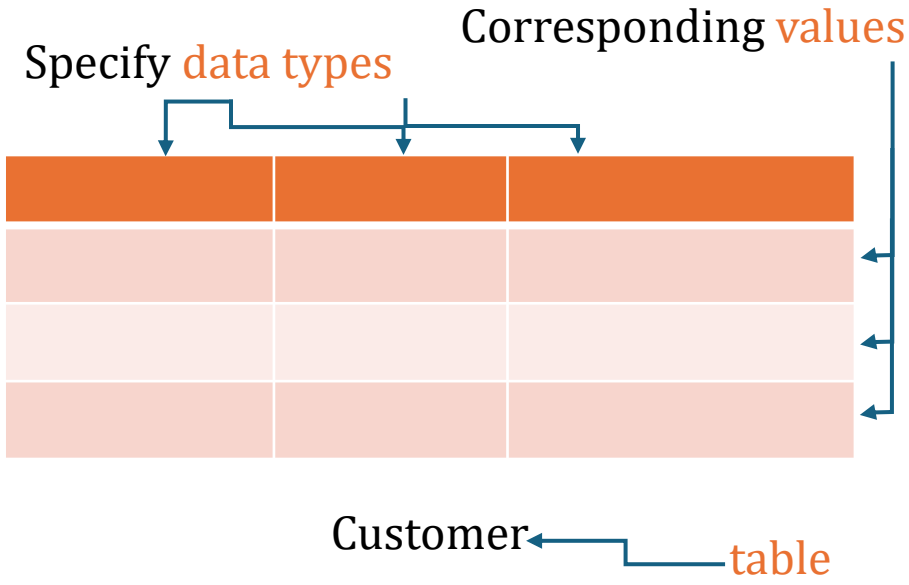
- Relational Database
  - Tables store data as
    - Columns
    - Rows

Client_id	name	address
1	Alice	2 rue de l'estrapade
2	Bob	24 rue de la paix
3	John	37 rue de l'est
4	Olivia	4 place de la nation
5	David	15 rue necker



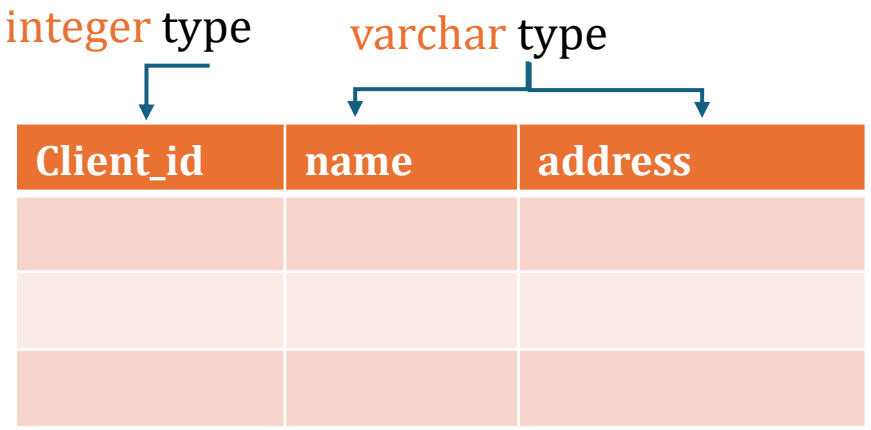
Types of Databases

- Relational Database
  - Tables store data as
    - Columns: attributes with defined data types (e.g. integer ID, varchar name)
    - Rows: records (tuples) with values matching those types



Types of Databases

- Relational Database
  - Tables store data as
    - Columns: attributes with defined data types (e.g. integer ID, varchar name)
    - Rows: records (tuples) with values matching those types



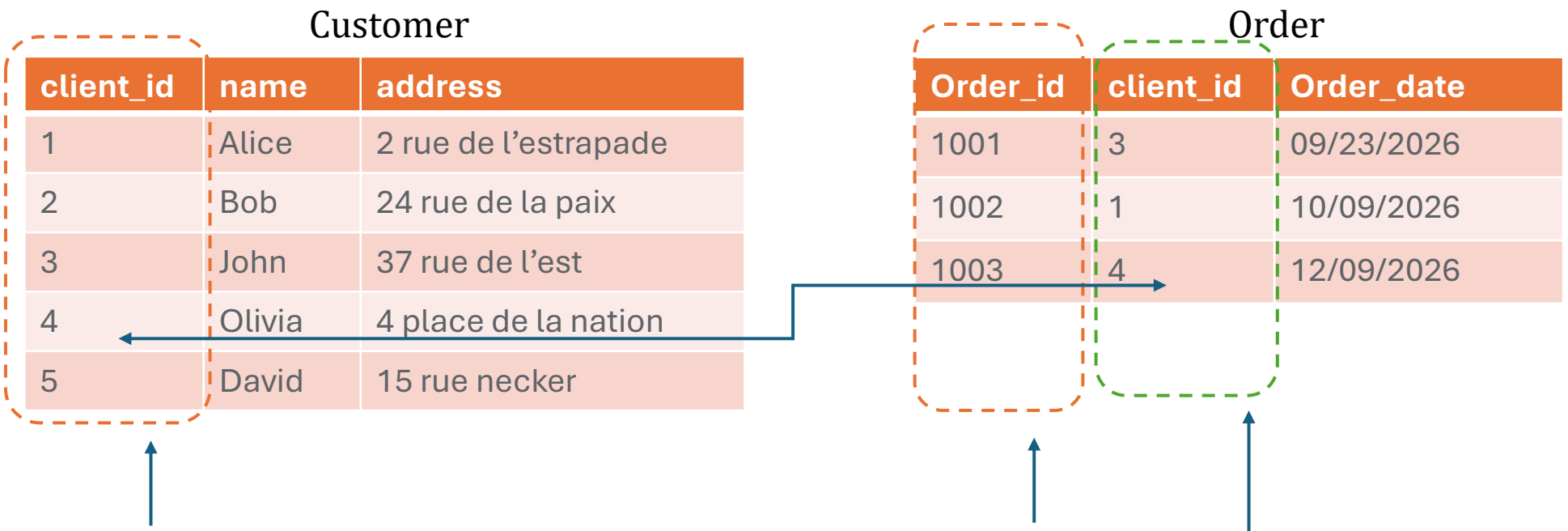
Customer

integer type varchar type

Client_id	name	address
1	Alice	2 rue de l'estrapade
2	Bob	24 rue de la paix
3	John	37 rue de l'est
4	Olivia	4 place de la nation
5	David	15 rue necker

Types of Databases

- Relational Database



Primary key uniquely identifies a record in the table

Foreign key is a reference to a primary key
(links a row to a primary key in another table)

Types of Databases

- How relational database model works ?
  - Use case : Business-to-consumer, track user actions on a website
  - We can use a customer table and an event table, linked by a foreign key each time a customer clicks, adds an item to the shopping cart,….

Customer

client_id	name	address
1	Alice	2 rue de l'estrapade
2	Bob	24 rue de la paix
3	John	37 rue de l'est
4	Olivia	4 place de la nation
5	David	15 rue necker

Event

event_id	client_id	action
101	3	LOGIN
102	3	VIEW PAGE
103	1	LOGIN
104	1	SEARCH

Types of Databases

- How relational database model works ?

Customer

client_id	name	address
1	Alice	2 rue de l'estrapade
2	Bob	24 rue de la paix
3	John	37 rue de l'est
4	Olivia	4 place de la nation
5	David	15 rue necker

Event

event_id	client_id	action
101	3	LOGIN
102	3	VIEW PAGE
103	1	LOGIN
104	1	SEARCH

SQL query

```
SELECT name, address
FROM customer
```

name	address
Alice	2 rue de l'estrapade
Bob	24 rue de la paix
John	37 rue de l'est
Olivia	4 place de la nation
David	15 rue necker

Types of Databases

- How relational database model works ?



Customer

client_id	name	address
1	Alice	2 rue de l'estrapade
2	Bob	24 rue de la paix
3	John	37 rue de l'est
4	Olivia	4 place de la nation
5	David	15 rue necker

Event

event_id	client_id	action
101	3	LOGIN
102	3	VIEW PAGE
103	1	LOGIN
104	1	SEARCH

SQL query : all actives cutomers

```
SELECT *
FROM customer
JOIN event
ON customer.client_id=event.client_id
```

client_id	name	address	event_id	client_id	action
3	John	37 rue de l'est	101	3	LOGIN
3	John	37 rue de l'est	102	3	VIEW PAGE
1	Alice	2 rue de l'estrapade	103	1	LOGIN
1	Alice	2 rue de l'estrapade	104	1	SEARCH

Types of Databases

- ACID Properties
  - All changes are performed as a single operation : if one step fails, everything rolls back
  - Atomicity : All or Nothing
  - Consistency : Only valid data
  - Isolation : No Interference
  - Durability : Data is recoverable
  - Example: Bank transfer. Check funds, debit one account, credit another. Any failure rolls back the whole transaction.

Types of Databases

- When to use ?
  - Structured data
  - ACID compliance

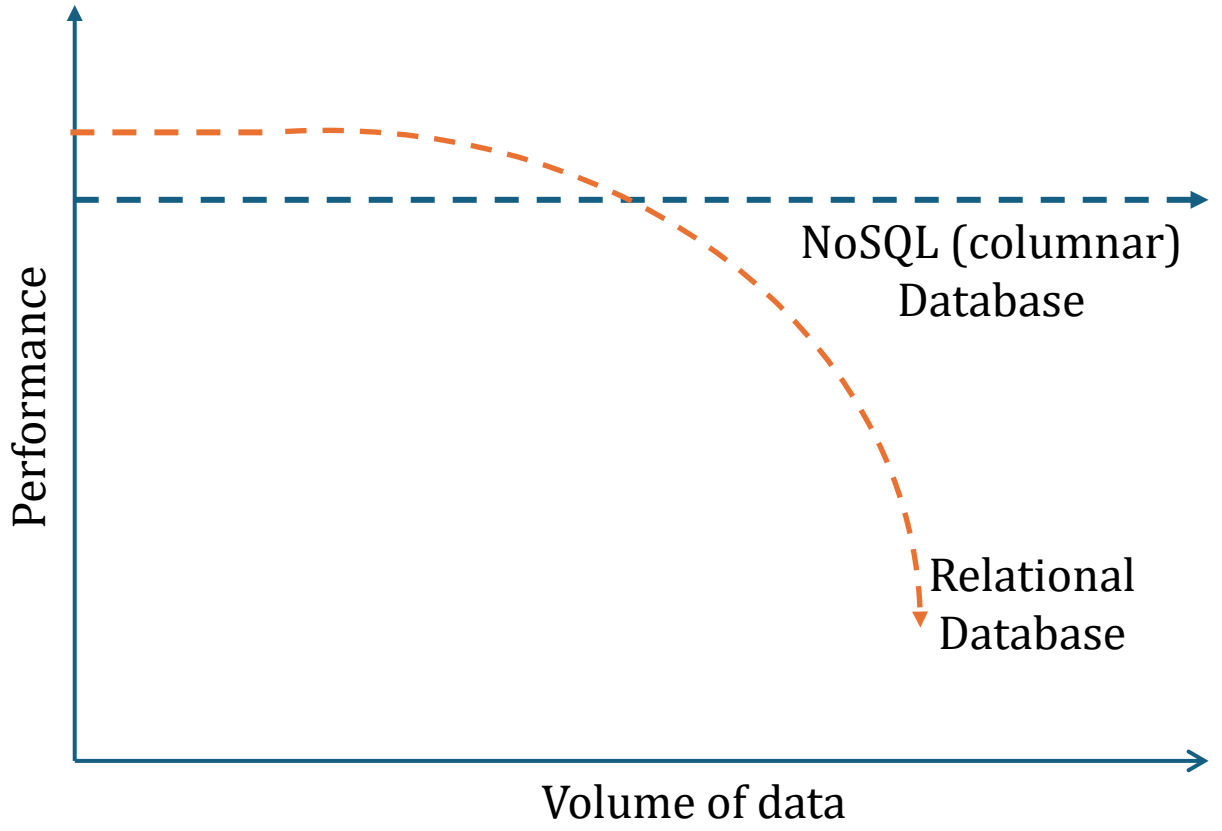
- Examples of relational databases : MySQL, PostgreSQL, MariaDB, Oracle Databases

- Cloud managed databases : Amazon Aurora, Azure SQL

Types of Databases

- Traditional (row-based) databases don't scale well with billions of records
  - queries get slow

- To improve query performance with big data columnar databases have been developed



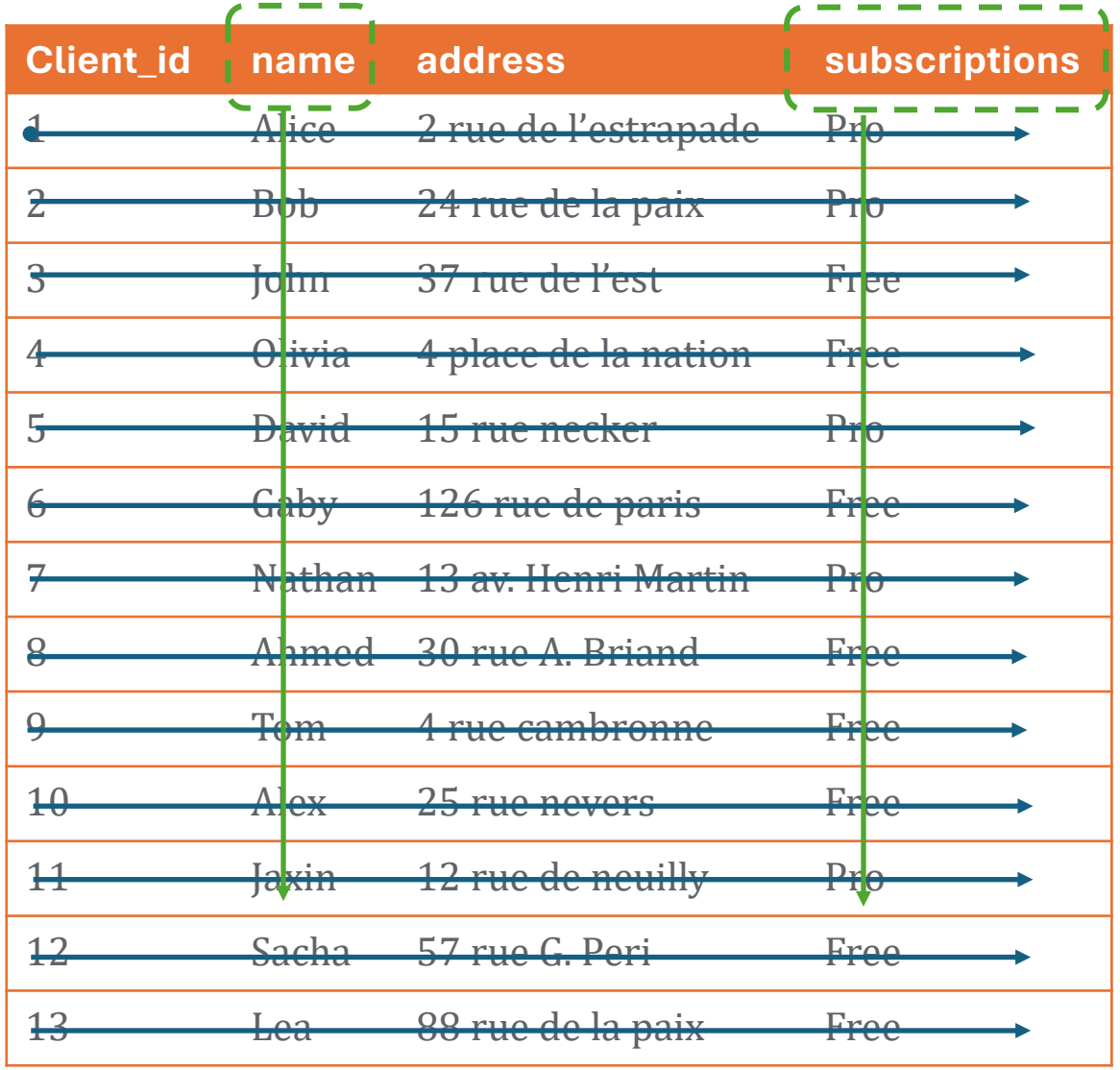
Types of Databases

- Columnar Database
  - Rules in relational databases
    - Read the data from left to right
    - Proceed from row to row
  - Example : return all paying customers
  - Result :

Client_id	name	address	subscriptions
1	Alice	2 rue de l'estrapade	Pro
2	Bob	24 rue de la paix	Pro
3	John	37 rue de l'est	Free
4	Olivia	4 place de la nation	Free
5	David	15 rue necker	Pro
6	Gaby	126 rue de paris	Free
7	Nathan	13 av. Henri Martin	Free
8	Ahmed	30 rue A. Briand	Free
9	Tom	4 rue cambronne	Free
10	Alex	25 rue nevers	Free
11	Jaxin	12 rue de neuilly	Pro
12	Sacha	57 rue G. Peri	Free
13	Lea	88 rue de la paix	Free

Types of Databases

- Columnar Database
  - Rules in relational databases
    - ➢ Read the data from left to right
    - ➢ Proceed from row to row
  - Example : return all paying customers
  - Result : 1, Alice, 2 rue de l'estrapade, Pro, 2, Bob, 24 rue de la paix, Pro, 5, David, 15 rue necker, Pro, 11, Jaxin, 12 rue de neuilly, Pro
- What happens if database contains billion of rows ?



Types of Databases

- Columnar Database
  - Rules
    - Read the data from top to bottom
    - Read only the columns you ask for (skips all unrelated data)
  - Example : return all paying customers
  - Result:

Client_id	name	address	subscriptions
1	Alice	2 rue de l'estrapade	Pro
2	Bob	24 rue de la paix	Pro
3	John	37 rue de l'est	Free
4	Olivia	4 place de la nation	Free
5	David	15 rue necker	Pro
6	Gaby	126 rue de paris	Free
7	Nathan	13 av. Henri Martin	Pro
8	Ahmed	30 rue A. Briand	Free
9	Tom	4 rue cambronne	Free
10	Alex	25 rue nevers	Free
11	Jaxin	12 rue de neuilly	Pro
12	Sacha	57 rue G. Peri	Free
13	Lea	88 rue de la paix	Free

Types of Databases

- Columnar Database
  - Rules
    - ➢ Read the data from top to bottom
    - ➢ Read only the columns you ask for (skips all unrelated data)
  - Example : return all paying customers
  - Result: Alice, Bob, David, Jaxin, Pro, Pro, Pro, Pro

Client_id	name	address	subscriptions
1	Alice	2 rue de l'estrapade	Pro
2	Bob	24 rue de la paix	Pro
3	John	37 rue de l'est	Free
4	Olivia	4 place de la nation	Free
5	David	15 rue necker	Pro
6	Gaby	126 rue de paris	Free
7	Nathan	13 av. Henri Martin	Pro
8	Ahmed	30 rue A. Briand	Free
9	Tom	4 rue cambronne	Free
10	Alex	25 rue nevers	Free
11	Jaxin	12 rue de neuilly	Pro
12	Sacha	57 rue G. Peri	Free
13	Lea	88 rue de la paix	Free

Types of Databases

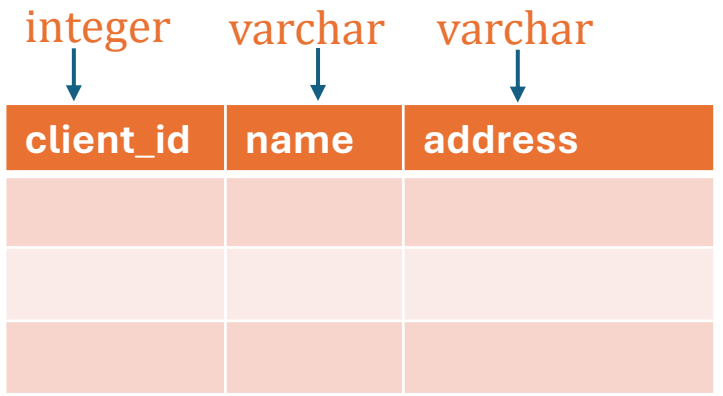
- Columnar Database
  - Row numbering lets the database pair values across columns (e.g. name rows 1–11 match subscriptions rows 1–11)
  - Result: 1, Alice, 2, Bob, 3, David, 4, Jaxin, 1, Pro, 2, Pro, 3, Pro, 4, Pro
  - Columnar database does not read billions of rows data, instead it reads a few long columns
  - Examples of columnar databases: Apache Cassandra, HBase

Client_id	name	address	subscriptions
1	Alice	2 rue de l'estrapade	Pro
2	Bob	24 rue de la paix	Pro
3	John	37 rue de l'est	Free
4	Olivia	4 place de la nation	Free
5	David	15 rue necker	Pro
6	Gaby	126 rue de paris	Free
7	Nathan	13 av. Henri Martin	Pro
8	Ahmed	30 rue A. Briand	Free
9	Tom	4 rue cambronne	Free
10	Alex	25 rue nevers	Free
11	Jaxin	12 rue de neuilly	Pro
12	Sacha	57 rue G. Peri	Free
13	Lea	88 rue de la paix	Free

Types of Databases

- Both relational and columnar databases requires to define a schema before storing anything
  - Example :

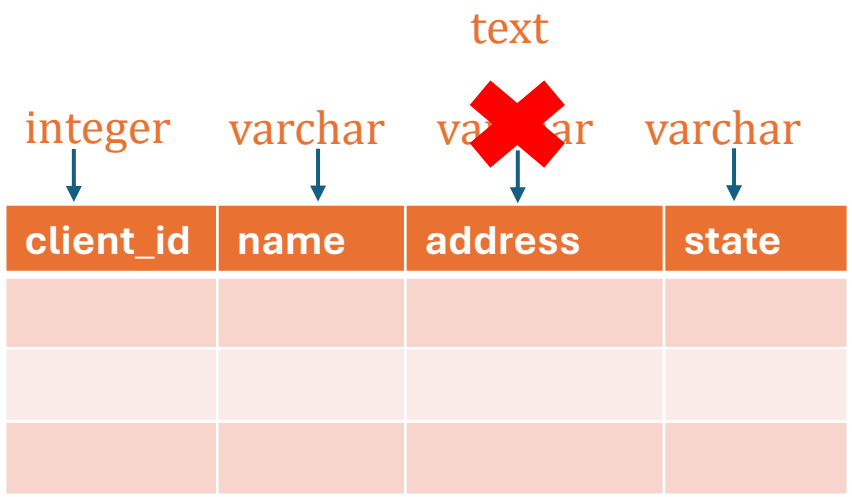
```
CREAT TABLE customer (
client_id integer PRIMARY KEY,
name varchar (256),
address varchar (256),
);
```



Types of Databases

- Problem1 : painful when products have thousands of different attributes
- Problem 2: If your requirement change over time, you might need to add columns or perhaps change the datatype of one of them
  - Example :

```
ALTER TABLE customer
ALTER COLUMN  address TYPE text;
```



Types of Databases

- Suppose you have an e-commerce database with different kind of products, each with a different attributes.
- Managing different attributes in relational database is inefficient and can slow down your database
  - Example : business that sells phones, cameras , and batteries

id	name	os	optics	storage
1	Samsung S26	Android 5.0	-	-
2	Sony Alpha 7V	-	Mirrorless	-
3	Varta	-	-	74 Ah

- Solution : Ideal use case for document database

Types of Databases

- Document database
- Store each product with all its attributes in a single document
  - Simplifies management, improves read speed, and changing one product's attributes doesn't affect others

- Stores and queries data in JSON-like documents : human- and machine-readable, hierarchical, semi-structured.
  - JSON represents data as :
    - Key-value pairs {"year": 2024}
    - Arrays {"fruit": ["apple", "grapes"]}
    - Objects {"address": {"country": "USA", "state": "Texas"}}

Types of Databases

- Document database
  - Example :

id	name	os	optics	storage
1	Samsung S26	Android 5.0	-	-
2	Sony Alpha 7V	-	Mirrorless	-
3	Varta	-	-	74 Ah

```
[
{“id”:1, “name”:“SamsungS26”, “os”:“Android 5.0”}
{“id”:2, “name”:“Sony Alpha 7V”, “optics”:“Mirrorless”}
{“id”:3, “name”:“Varta”, “storage”:“74 Ah”}
]
```

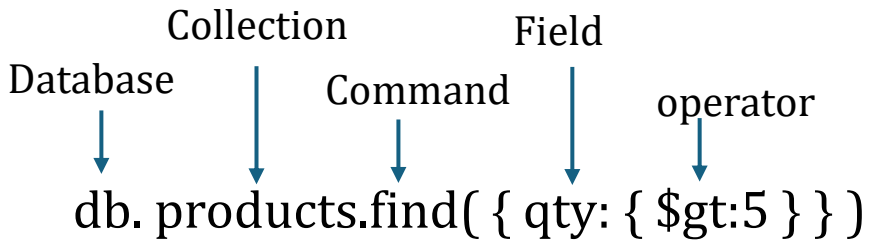
You can change the storage to price

Types of Databases

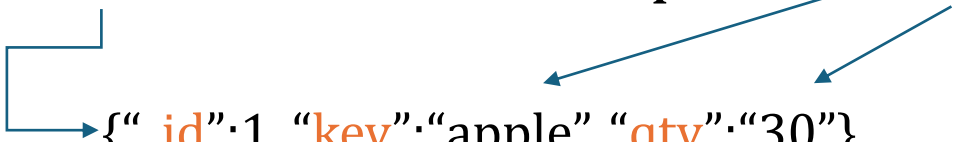
- Document database operations
- You can create, read, update, and delete entire documents
- Create : each document has a unique identifier that serves as a key



- Read



- Update : rewrite the entire documents or update individual values



Types of Databases

- Advantages of document databases
  - Ease of development : JSON documents map to objects; No need to create a data model; development is more rapid and efficient

    JSON >> {“brand”:“BMW”, “color”:“red”} ; Python >> car = {“brand”:“BMW”, “color”:“red”}

  - Flexible schema : documents with different fields

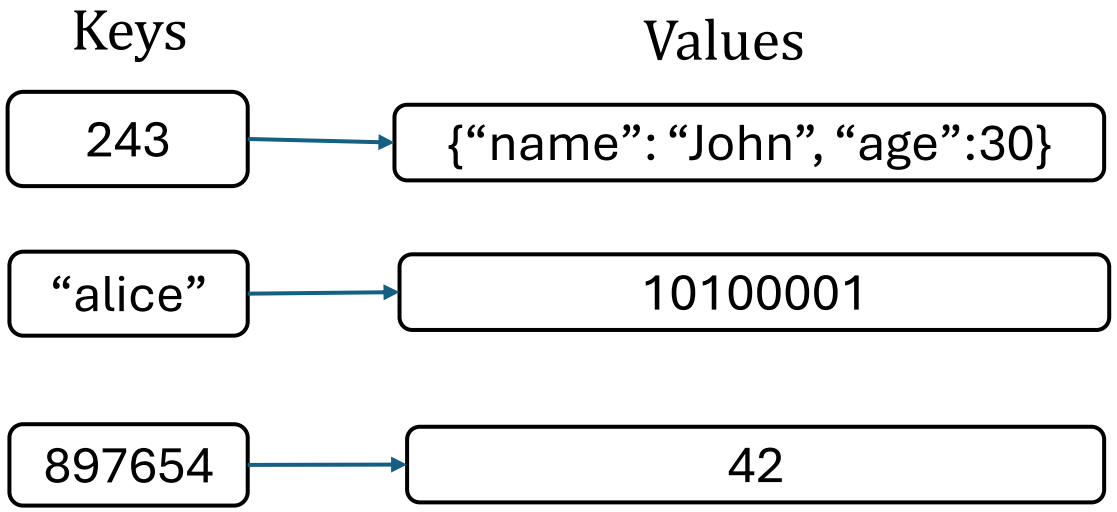
    [{“brand”:“BMW”, “color”:“red”} {“brand”:“Apple”, “os”:“iOS”} ; [

  - Performance at scale : built-in horizontal distribution and replication; fault tolerance and high availability

- Example of document database : MongoDB

Types of Databases

- Key-value database
  - It stores data as a collection of key, value pairs, where
  - keys serves as unique identifier
  - Both keys and values could be anything raging from simple object to complex compound objects



Types of Databases

- Key-value database
  - Document databases are a special case of key-value databases
  - Differences :
    - Keys can only be strings
    - You can read entire value or a part of the value especially if the value is another JSON object
    - Example : {“book”: { “id”: 1,“price”:10} }. => book.price will return 10. key-value databases always return the whole value including both id and price information

Types of Databases

- Key-value database
  - Advantages
    - Scalability : horizontal scaling, and automatic distribution of data across servers
    - Ease of use : object-oriented paradigm maps real-world objects to software objects
    - Performance : no resource-intensive table joins, which makes them much faster

Types of Databases

- Key-value database
  - Use cases :
    - Session management: web app sessions from login to logout or timeout
    - Shopping carts: e-commerce sites handling billions of orders per second during peak season
    - Metadata storage engine: underlying storage layer for higher-level data access
    - Caching: temporary storage for faster retrieval
  - Example: etcd used by Kubernetes to store all cluster data (pods, deployments, services)

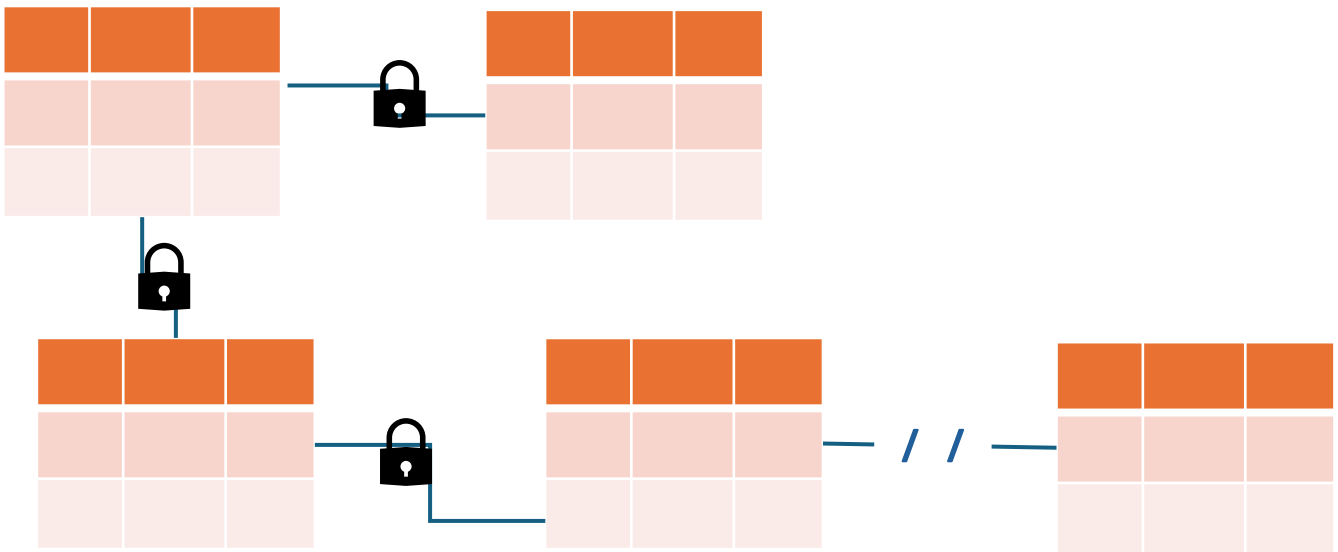
Types of Databases

- Graph databases



Types of Databases

- Graph databases
- To explain graph databases, let's start with relational databases

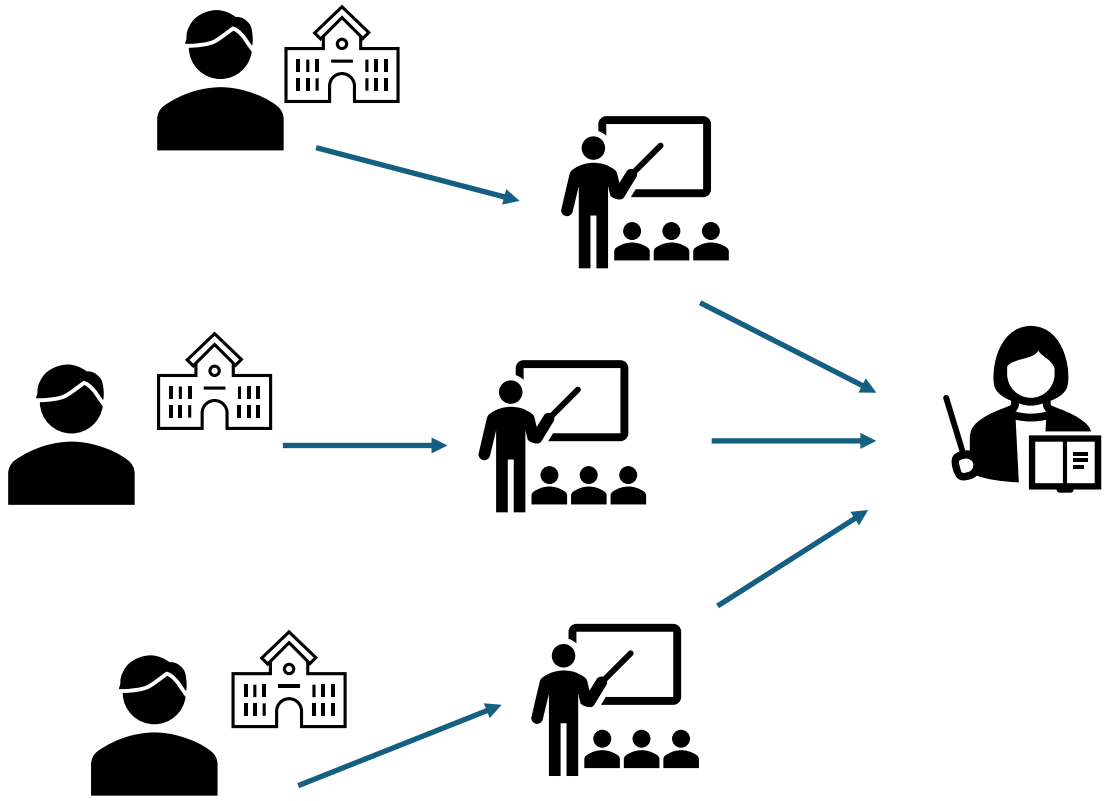


ACID guarantees but constraining nature of relationships

Hard to answer questions abouts distant relationships

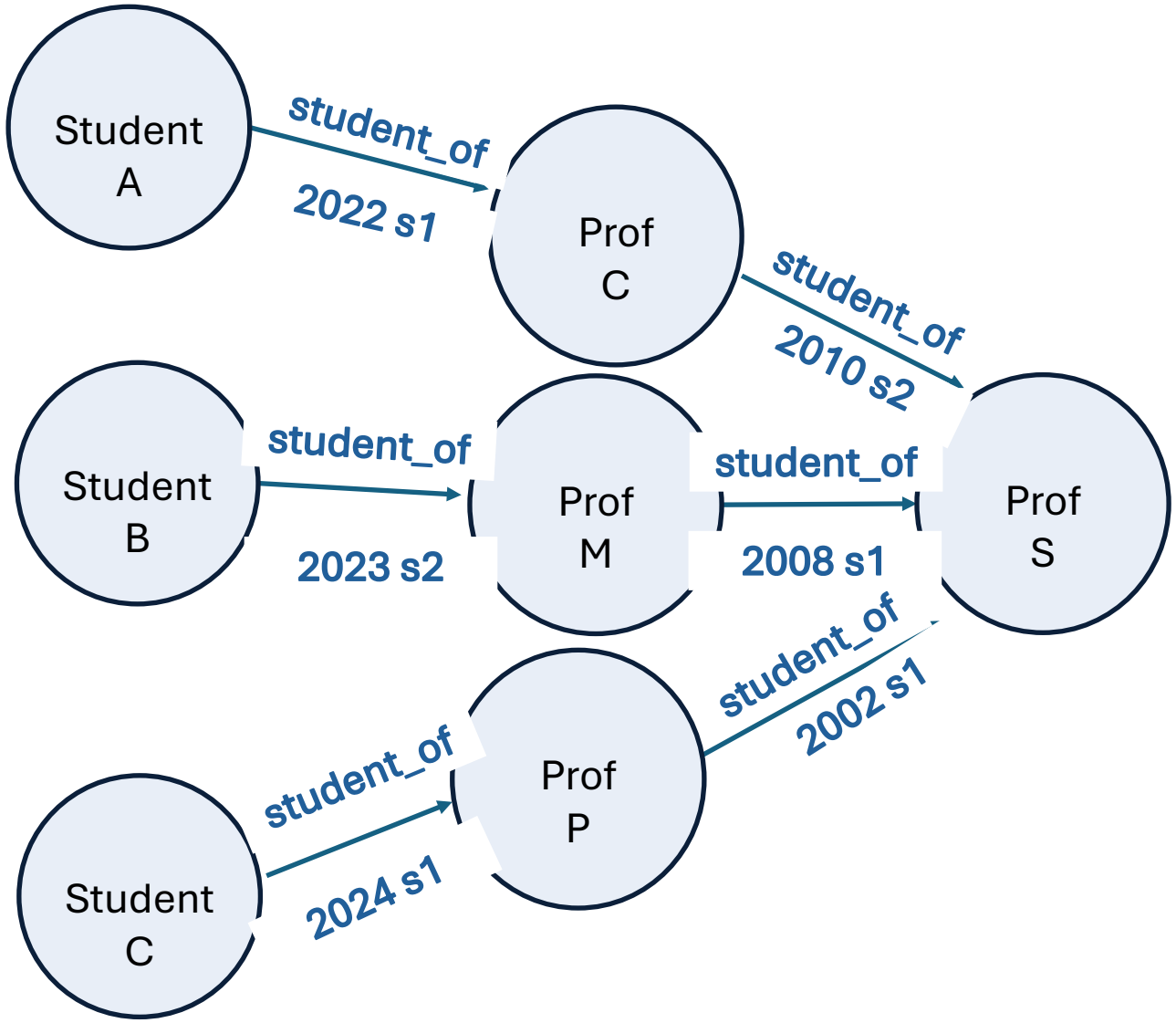
Types of Databases

- Graph databases
- Example :
  - a collection of data for all students and all professors
  - the relationship between a group of 3 students
  - graph analysis reveals they all shared a common professor



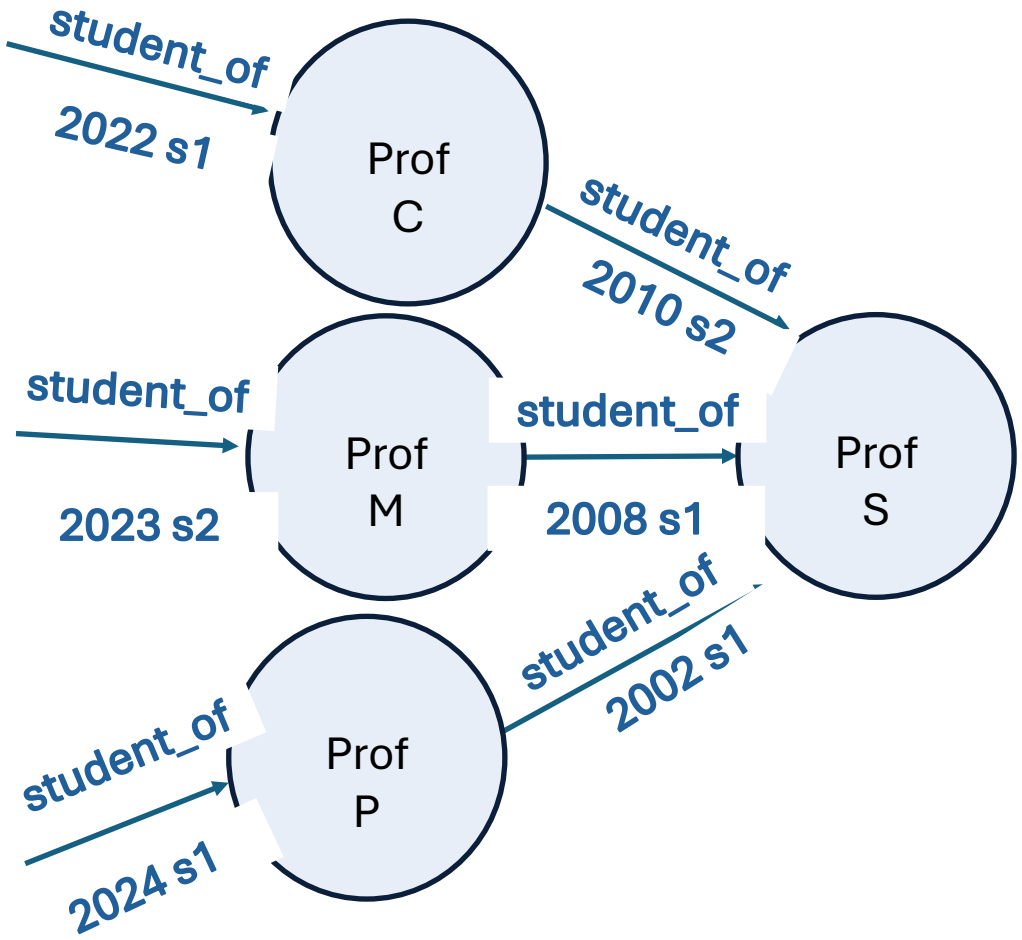
Types of Databases

- Graph databases
- Structure:
  - Nodes — the records
  - Relationships — with a direction, a type (e.g. “student_of”), and properties (e.g. year and semester)
- Querying :
  - Vendors have their own query languages : Gremlin, SparQL, Cypher



Types of Databases

- Graph databases
- Graph databases can infer connections that do not actually mean anything.
  - Suppose that all our 3 students A, B and C ended up of leaving school
  - Does that mean that professor S had some kind of meaningful impact ?
- Graph databases are usually a mechanism for starting questions but not necessarily for answering them
- Example: Neo4j, ArangoDB



Types of Databases

- Summary : NoSQL vs. NewSQL vs. Graph
  - NoSQL: umbrella term for non-relational stores (document, key-value, column, graph) usually trading consistency/joins for scale and flexibility.
  - NewSQL: relational semantics (SQL, ACID) with distributed-system scalability.
  - Graph databases are a NoSQL category, but many (including Neo4j) are fully ACID-compliant (best of both worlds for connected data).

Types of Databases

- Summary — Which Database for Which Job?

Database type	Best for
Relational	Structured data, ACID compliance
Columnar	Huge write volume, Analytics on big data
Document	Unstructured data flexibility
Key-value	Simple, fast data access
Graph	Connected data, Complex relationships
Time-series	Time-stamped data
Vector	AI and ML scenarios

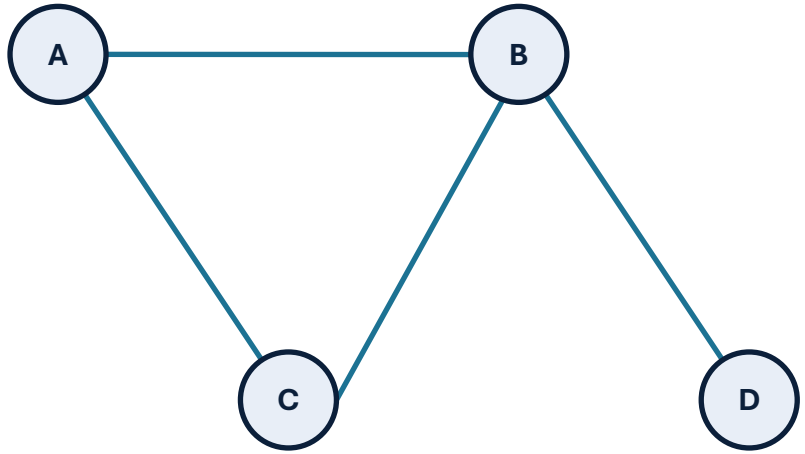
The right database is a game-changer for performance (the wrong one slows your queries and gets expensive)

Agenda

1. Introduction
2. Types of Databases
3. Graph theory foundations
4. From graph theory to graph databases
5. Introducing Neo4j

Graph theory foundations

- What Is a Graph?
  - A mathematical structure used to model pairwise relationships between objects.
  - Vertices (nodes) represents the objects.
  - Edges (links / relationships) represent the connections between objects.



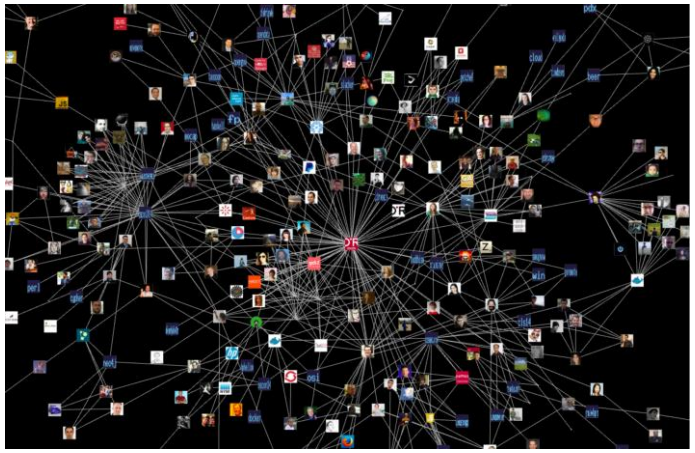
- Formally

  A graph G is a tuple (V, E) where, V is a set of vertices and $E \subseteq V \times V$ is a set of edges.

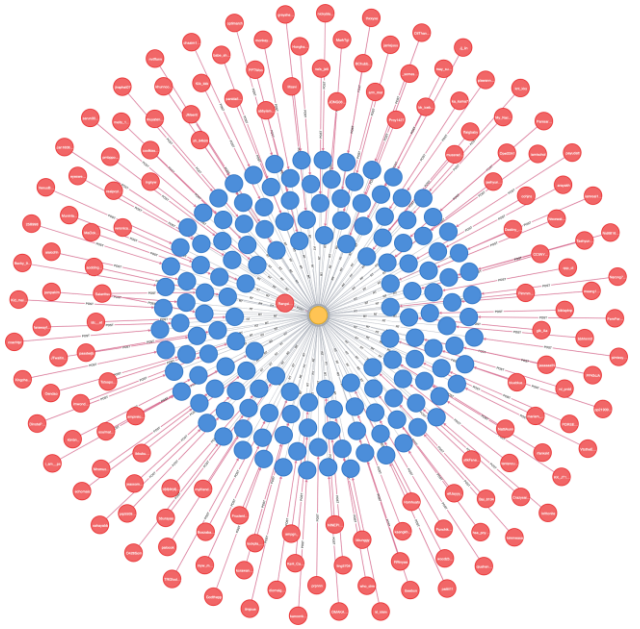
- Graphs let us reason about connections, not just isolated data points.

Graph theory foundations

- Graphs are everywhere
  - Social networks (people, friendships)
  - The Web (pages, hyperlinks)
  - Molecules (atoms, bonds)
  - Computer networks (devices, links)
  - Supply chains (suppliers, shipments)
  - Knowledge (concepts, relations)
  - ….



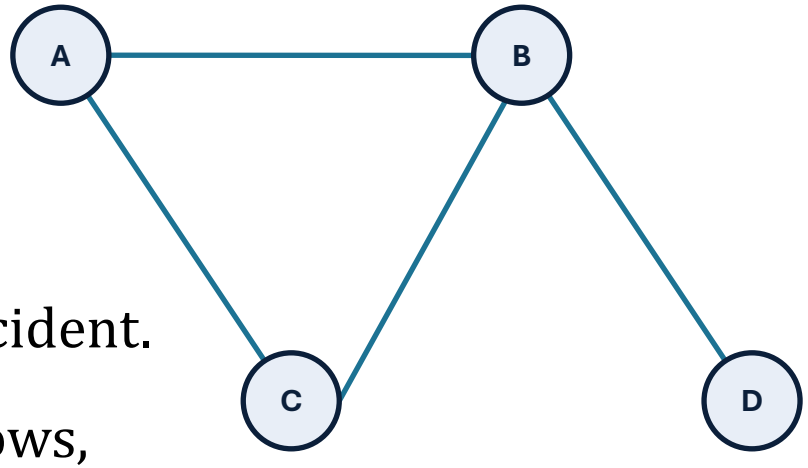
O’Reilly’s Graph Databases ebook (Neo4j)



Twitter data was collected on February 19, 2021 (Neo4j)

Graph theory foundations

- A vertex represents an entity: a person, a city, a product...
- An edge represents a relationship between two vertices: knows, is_connected_to, bought, ...
- An edge always has two endpoints.
- Two vertices joined by an edge are adjacent, or neighbours.
- A vertex and an edge that touch each other are said to be incident.
- Most real-world relationships have a natural direction: Follows, purchased, manages, ...





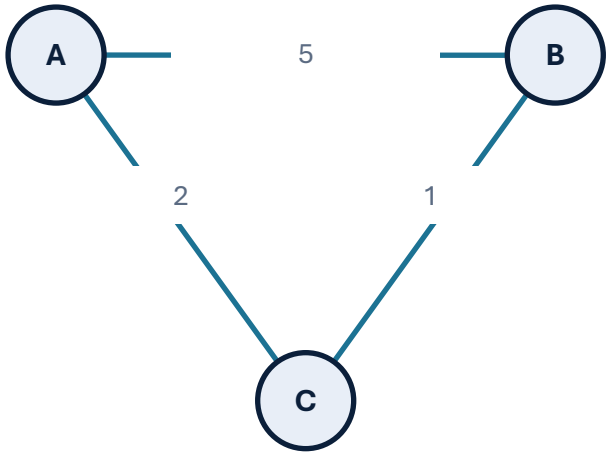
A knows B and B knows A



A follows B

Graph theory foundations

- Weighted Graphs
  - Edges can carry a weight that represents a cost, a distance, a strength, a probability



- The degree of a node is the number of edges connected to it.

- In directed graphs we distinguish:
  - in-degree: number of incoming edges.
  - out-degree: number of outgoing edges.

- High-degree nodes are often the most influential/central (a hub airport, a celebrity account).

Graph theory foundations

- Cycles
  - A cycle is a path that starts and ends at the same node.
  - A graph with no cycles is called acyclic.
  - A DAG (Directed Acyclic Graph) is a directed graph with no cycles (common for dependency graphs, workflows, version histories).
- Connected Graphs
  - An undirected graph is connected if there is a path between every pair of nodes.
  - A connected component is a maximal set of nodes that are mutually reachable.
  - Real-world graphs (social networks, the Web) often have one giant connected component plus many tiny isolated ones.

Graph theory foundations

- Density
  - Sparse graph: few edges relative to the number of possible edges (most real-world graphs).
  - Dense graph: close to the maximum number of edges, $n(n-1)/2$.
  - Social networks, the Web, and enterprise data are typically sparse but highly connected (this matters a lot for how we store them).

Graph theory foundations

- Density
  - Sparse graph: few edges relative to the number of possible edges (most real-world graphs).
  - Dense graph: close to the maximum number of edges, $n(n-1)/2$.
  - Social networks, the Web, and enterprise data are typically sparse but highly connected (this matters a lot for how we store them).
- Paths and Walks
  - A walk is a sequence of nodes connected by edges: $v1 \rightarrow v2 \rightarrow ... \rightarrow vn$.
  - A path is a walk that does not repeat any node. The length of a path is its number of edges.
  - The shortest path between two nodes has the minimum number of edges (or minimum total weight).

Graph theory foundations

- Representing Graphs
  - Adjacency Matrix
  - Adjacency List : each node stores a list of its neighbours

	A	B	C	D
A	0	1	1	0
B	1	0	0	1
C	1	0	0	1
D	0	1	1	0

```
A -> [B, C]
B -> [A, D]
C -> [A, D]
D -> [B, C]
```

- Limits of These Representations at Scale
  - Adjacency matrices become unusable memory-wise for millions of nodes.
  - Adjacency lists in a general-purpose language don't give ACID guarantees, indexing, querying, or persistence.
  - We need a system that stores billions of nodes/edges efficiently, traverse relationships in constant time, and supports transactions, security, and a query language.
  - =>> that system is a graph database

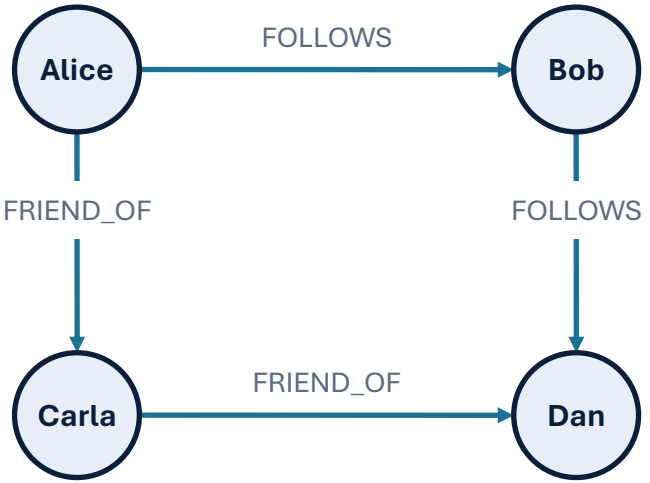
Agenda

1. Introduction
2. Types of Databases
3. Graph theory foundations
4. From graph theory to graph databases
5. Introducing Neo4j

From graph theory to graph databases

- Why to store Data as a Graph?
  - Many real datasets are naturally relationship-centric: connections carry as much meaning as the data itself.
  - Traditional databases store data in tables/documents and compute relationships at query time.
  - A graph database stores relationships as first-class citizens, physically, at write time.
  - Result: traversing connected data becomes fast and natural, even for deep, complex queries.

- Example 1: Social Network
  - “Who are Alice's friends-of-friends?”
    is one hop-by-hop traversal, not a chain of table joins.



From graph theory to graph databases

- Example 2: Recommendation Engine
  - Nodes: User, Product, Category.
  - Relationships: BOUGHT, VIEWED, BELONGS_TO.
  - Query: “Customers who bought this also bought...” => traverse BOUGHT two hops out.

- Example 3: Supply Chain
  - Nodes: Supplier, Factory, Warehouse, Shipment.
  - Relationships: SUPPLIES, SHIPS_TO, STORED_IN.
  - Question: “If Supplier X has an issue, which final products are impacted?” → multi-hop traversal across the whole network.

From graph theory to graph databases

- The Common Pattern
  - All these use cases share one trait: highly connected data where the value is in the relationships.
  - Common question shapes:
    - “What is connected to X, up to N hops away?”
    - “What is the shortest / strongest path between X and Y?”
    - “What patterns of connections repeat across the dataset?”
  - These questions are hard and slow in tabular systems, natural and fast in graph databases.

From graph theory to graph databases

- What Is a Property Graph?
  - The data model used by Neo4j and most modern graph databases.
  - Extends plain graph theory with labels and key-value properties on both nodes and relationships.
  - Four building blocks: Nodes, Labels, Relationships, Properties.

- Nodes
  - Represent entities: a person, a product, an event.
  - Can have zero, one, or several labels.
  - Can hold any number of properties (key-value pairs).
  - In Cypher, drawn as (n).

From graph theory to graph databases

- Labels
  - A label tags a node with a category/type, e.g. :Person, :Product, :City.
  - A node can have multiple labels (:Person:Employee).
  - Labels are used to group nodes and to speed up lookups (via indexes).

- Relationships
  - Connect exactly two nodes (a start node and an end node).
  - Always have exactly one type, e.g. :FRIEND_OF, :PURCHASED.
  - Always have a direction in storage (though queries can ignore it).
  - Can hold properties too, e.g. since: 2021 on a :FRIEND_OF relationship.

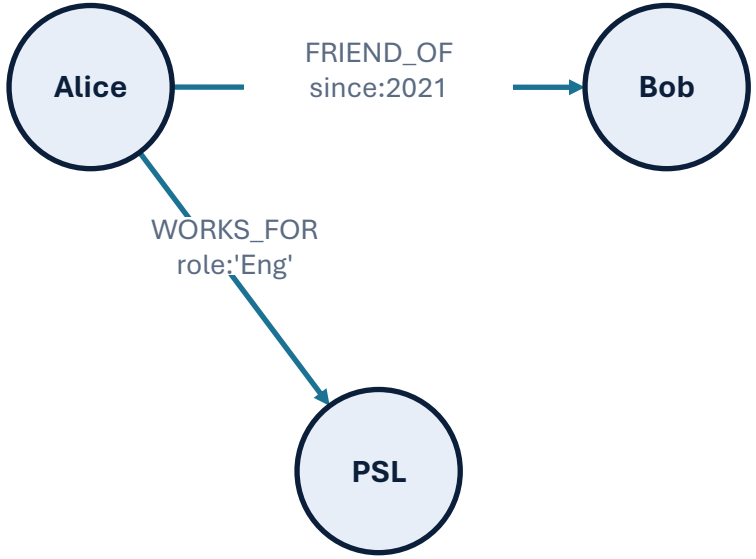
From graph theory to graph databases

- Relationship direction
  - Every relationship is stored with a direction: (a)-[:TYPE]->(b)
  - Direction gives meaning: (Alice)-[:FOLLOWS]->(Bob) ≠ (Bob)-[:FOLLOWS]->(Alice).
  - Queries can still traverse relationships in either direction, or ignore direction entirely when it doesn't matter.

- Relationship types
  - Similar to labels, but for relationships (exactly one type per relationship).
  - Common convention: UPPER_SNAKE_CASE verbs, e.g. :OWNS, :LIVES_IN, :WORKS_FOR.
  - A rich, expressive relationship type is often more valuable than a generic one.

From graph theory to graph databases

- Properties
  - Key-value pairs attached to nodes or relationships.
  - Values are typed: string, integer, float, boolean, date, list, etc.
  - Example node: (:Person {name:'Alice', age:30}).
  - Example relationship: [:FRIEND_OF {since:2021}].
  - No fixed schema is required (different nodes with the same label can have different properties)



(:Person {name:'Alice', age:30}) and (:Company {name:'PSL'})

Agenda

1. Introduction
2. Types of Databases
3. Graph theory foundations
4. From graph theory to graph databases
5. Introducing Neo4j

Introducing Neo4j

- What Is Neo4j?
  - The world's leading native graph database, first released in 2007.

☐ include secondary database models

44 systems in ranking, October 2026

Rank			DBMS	Database Model	Score		
Oct 2026	Sep 2026	Oct 2025			Oct 2026	Sep 2026	Oct 2025
1.	1.	1.	Neo4j	Graph	48.88	+1.72	-3.63
2.	2.	2.	Microsoft Azure Cosmos DB	Multi-model	21.38	+1.03	-1.33
3.	3.	3.	Aerospike	Multi-model	4.30	+0.18	-0.23
4.	4.	4.	Arango	Multi-model	3.29	-0.10	+0.37
5.	5.	↑ 9.	Amazon Neptune	Multi-model	2.83	+0.09	+0.57
6.	6.	6.	OrientDB	Multi-model	2.46	-0.04	-0.16
7.	7.	↑ 8.	GraphDB	Multi-model	2.44	+0.06	+0.14
8.	8.	↓ 5.	Virtuoso	Multi-model	2.35	-0.01	-0.39
9.	9.	↑ 13.	Stardog	Multi-model	2.32	+0.18	+0.59
10.	↑ 11.	↑ 12.	TigerGraph	Graph	1.95	0.00	+0.18
11.	↓ 10.	11.	JanusGraph	Graph	1.92	-0.03	+0.11

https://db-engines.com/en/ranking/graph+dbms

Introducing Neo4j

- What Is Neo4j?
  - The world's leading native graph database, first released in 2007.
  - Stores data as a property graph and provides Cypher, a dedicated declarative query language.
  - Cypher introduced as an intuitive, ASCII-art-style query language.
  - Cypher has since become the foundation of GQL, the new ISO standard graph query language (2024).
  - Fully ACID-compliant transactional database.
  - Used across finance, retail, telecom, life sciences, and increasingly GenAI/RAG applications.

Introducing Neo4j

- Native Graph Storage
  - Native means the storage engine was designed for graphs from the ground up (not a relational or document store with a graph layer on top).
- Index-Free Adjacency in Neo4j
  - Each node record physically references its first relationship; each relationship references the next one in a linked list, per node.
  - Traversing from a node to all its neighbours is a pointer-chase (constant time per hop, independent of database size).
  - This is why deep traversals remain fast even on graphs with billions of nodes.

Introducing Neo4j

- Neo4j Architecture Overview
  - DBMS: manages one or more databases.
  - Storage engine: native graph store on disk (nodes, relationships, properties, labels stores).
  - Query engine: parses and executes Cypher, with a cost-based optimiser.
  - Transaction manager: guarantees ACID semantics.
  - Driver layer: Bolt protocol for client communication (Java, Python, JS, .NET, Go...).

Introducing Neo4j

- Neo4j Ecosystem
  - Neo4j Browser: web-based query and visualisation tool,
  - Neo4j Desktop: local database management,
  - Neo4j Bloom: no-code graph visual exploration,
  - Neo4j Data Importer: CSV-to-graph import wizard,
  - APOC: extended standard library of procedures/functions,
  - Graph Data Science (GDS): graph algorithms library (PageRank, community detection...).

Introducing Neo4j

- Cypher: A Sneak Preview

```
MATCH (a:Person)-[:FRIEND_OF]->(b:Person)
WHERE a.name = 'Alice'
RETURN b.name;
```

- (a:Person) is a node pattern, -[:FRIEND_OF]-> is a relationship pattern.
- Full details in the next deck!

Summary

- Graphs model entities and their connections: nodes, edges, direction, weight.
- Graph databases store relationships physically ⇒ index-free adjacency ⇒ fast traversals.
- The property graph model (nodes, labels, relationships, properties) is Neo4j's data model.
- Neo4j is a native, ACID-compliant graph database with a rich ecosystem.

- Next: learn to write graph queries with Cypher.

Bibliography

- Neo4j, Neo4j Documentation and Cypher Manual, Neo4j, Inc. https://neo4j.com/docs/cypher-manual/current/introduction/
- Neo4j, Graph Data Science Library Manual, Neo4j, Inc. https://neo4j.com/docs/graph-data-science/current/
- Bratanic, T. (2024). Graph algorithms for data science: With examples in Neo4j. Simon and Schuster.
- Scifo, E. (2023). Graph Data Science with Neo4j: Learn how to use Neo4j 5 with Graph Data Science library 2.0 and its Python driver for your project. Packt Publishing
- Rossi F. (2020). An introduction to database management system.
- https://www.splunk.com/en_us/blog/learn/data-trends.html
- Putra A (2024). Types of Databases.