Database Analysis and Design

Lesson 2: Introduction to Database

Ms. SEAK Leng

Have you ever wondered how websites like Amazon manage millions of transactions every day without losing data?

What is a database?

- A database is an organized collection of data, typically stored and accessed electronically.
- It allows for efficient data management, retrieval and updating.
- Databases are typically managed by Database Management Systems (DBMS) that provide an interface for users and applications to interact with the data.

Key features of a database

1. Structured data: data typically stored in a structured manner, often in tables, but modern databases can also support unstructured data.
2. Data integrity: database enforces rules to maintain the accuracy and consistency of the stored data.
3. Data retrieval: query languages are used to fetch specific information based on user needs.
4. Concurrent access: multiple users or applications can access and manipulate data concurrently without causing conflicts.

Database Schema

- A database schema refers to the overall logical structure or blueprint of the database.
- It describes *how data in the database is structured.*
- The schema does not store actual data; it defines the structure of the data.
- The names of tables, columns of each table, datatype, functions, and other objects are included in the schema. Example:
  – Teacher(TID, Tname, Dob, Degree, Field)
  – Department(Dname, Description)

Database Schema

- Example:

The schema diagram for the university database.

Student Table

ID	First_Name	Last_Name	Class	Major

Course Table

Course_ID	Course_Name	Course_credits

Department Table

Department_Code	Department_Name

Instructor Table

Instructor_ID	Instructor_fname	Department_Code

Grade Table

ID	Course_ID	Grade

Database Instance

- An instance is the information collected in a database at some specific moment in time, also known as the **database state.**
- It is a snapshot of the current state or occurrence of a database. Each time data is inserted into or deleted from the database, it changes the state of the database.
- In simple terms: The schema is the design, or blueprint, of the database. The instance is the actual data present in the database at any moment.

Database Instance

- Example at a particular moment in time

ID	First_Name	Last_Name	Class	Major
1001	Bob	Dylan	Junior	Maths
1002	Ceaser	Zappelli	Freshman	Economics
1003	Antony	Rodgers	Senior	Psychology
1004	George	Miller	Sophomore	Computer science
...	...	...	...	...

- This is just one instance of the STUDENT table. If we add, remove or update records into this table than we will enter a new database state.

DBMS

- A Database Management System (DBMS) is software that enables users to define, create, manage, and control access to a database.
- It provides a layer of abstraction between the raw data stored in the database and the applications or users accessing it.

Functions of a DBMS

1. Data definition: defines the structure of the data
2. Data manipulation: provides ways to retrieve and update data using languages like SQL.
3. Data security: controls who can access or manipulate the data
4. Backup and Recovery: ensures that data can be recovered in case of failure

Functions of a DBMS

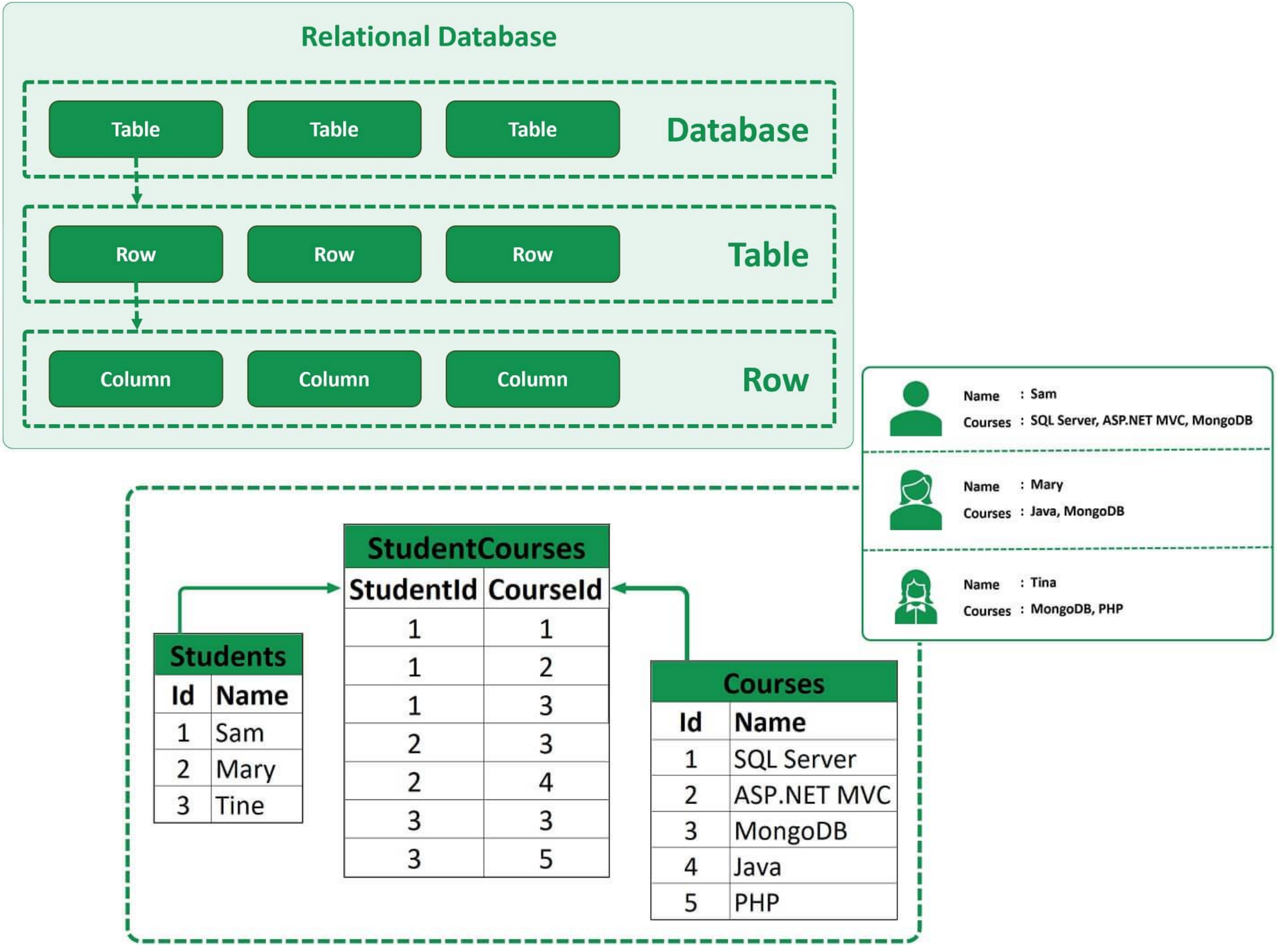
4. Concurrency control: manages multiple users accessing the data at the same time.
5. Data integrity: enforces rules such as constraints to maintain the accuracy and consistency of data.
6. Performance optimization: indexing, query optimization, and caching to improve performance.

Different Types of Databases

1. Relational Databases (RDBMS)
2. NoSQL Databases
3. Hierarchical Databases
4. Network Databases
5. Object-Oriented Databases (OODBMS)
6. Time-series Databases

Different Types of Databases

1. Relational Databases (RDBMS)
   - Stores data in tables with rows and columns.
   - Supports SQL for querying and managing data
   - Examples: MySQL, PostgreSQL, Microsoft SQL Server, Oracle.
   - Strong support for data integrity and ACID properties (Atomicity, Consistency, Isolation, Durability).
   - Use cases: E-Commerce platforms, Content Management Systems (CMS), Educational Management Systems, Financial institutions, Inventory Management, HRMS, Reservation system



Relational Database
Table
Table
Table
Database
Row
Row
Row
Table
Column
Column
Column
Row
Name : Sam
Courses : SQL Server, ASP.NET MVC, MongoDB
Name : Mary
Courses : Java, MongoDB
Name : Tina
Courses : MongoDB, PHP
StudentCourses
StudentId CourseId
1 1
1 2
1 3
2 3
2 4
3 3
3 5
Students
Id Name
1 Sam
2 Mary
3 Tine
Courses
Id Name
1 SQL Server
2 ASP.NET MVC
3 MongoDB
4 Java
5 PHP

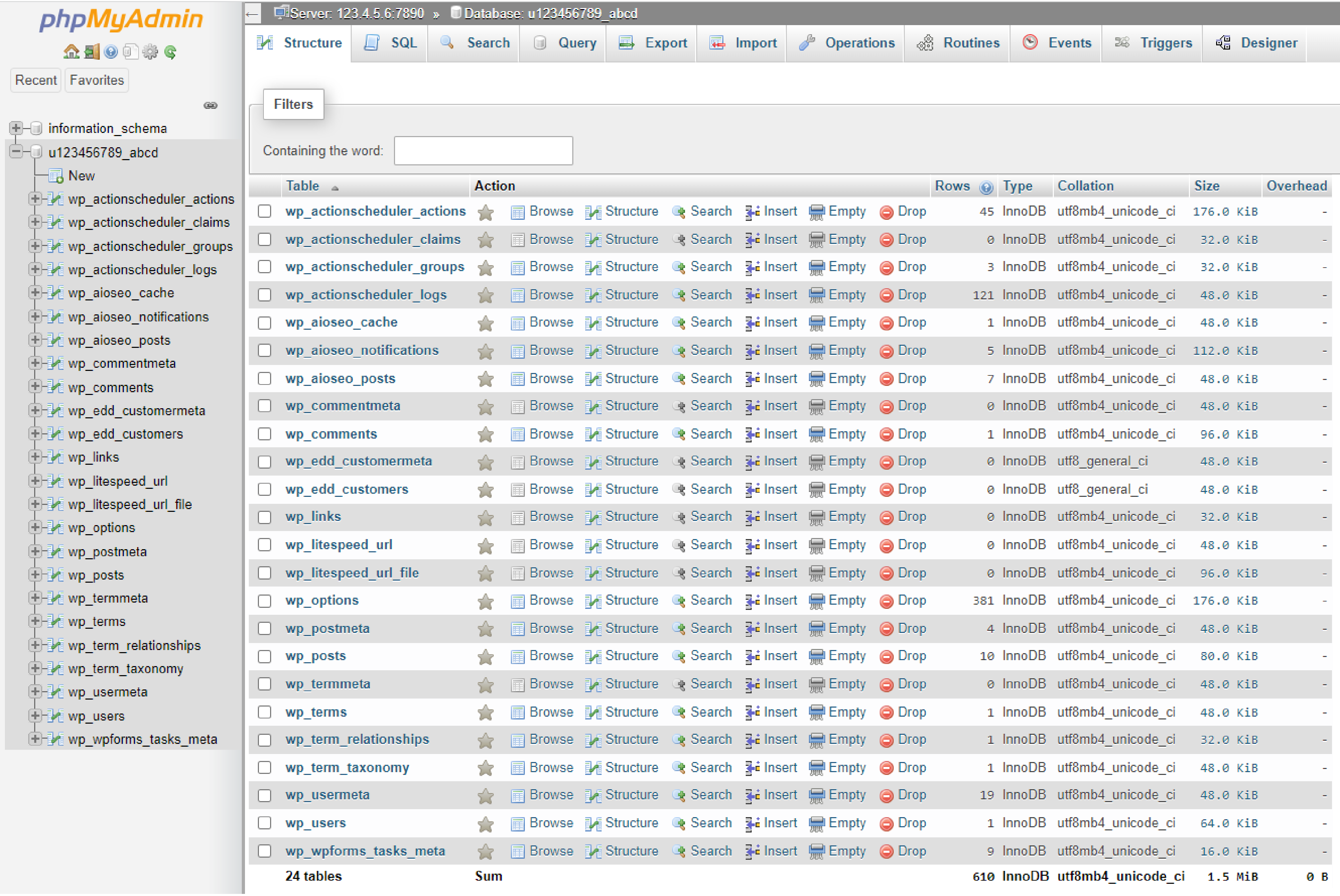
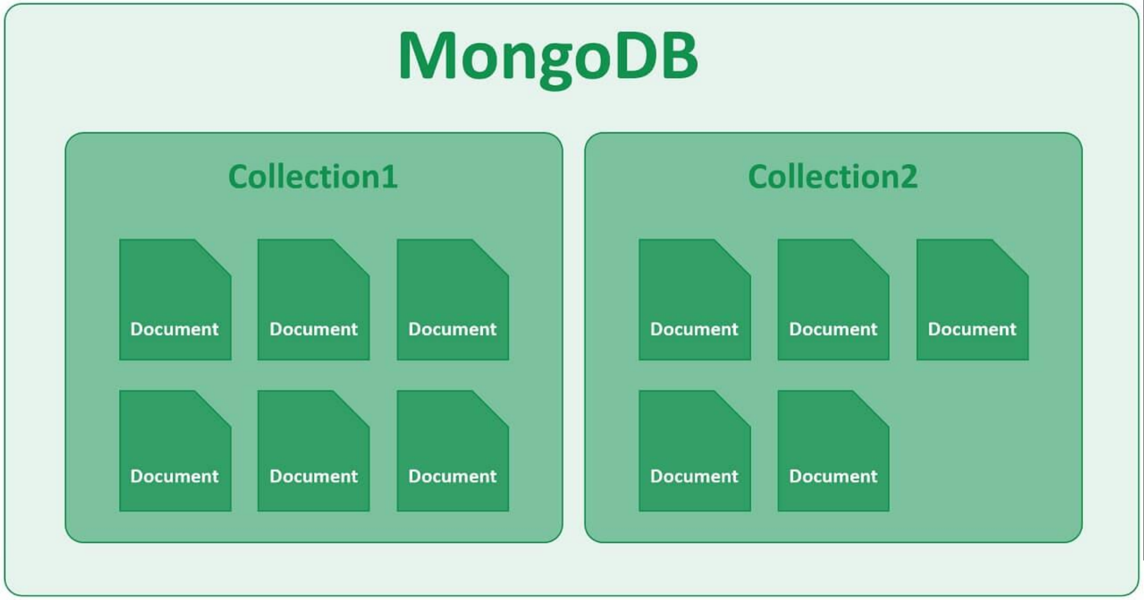


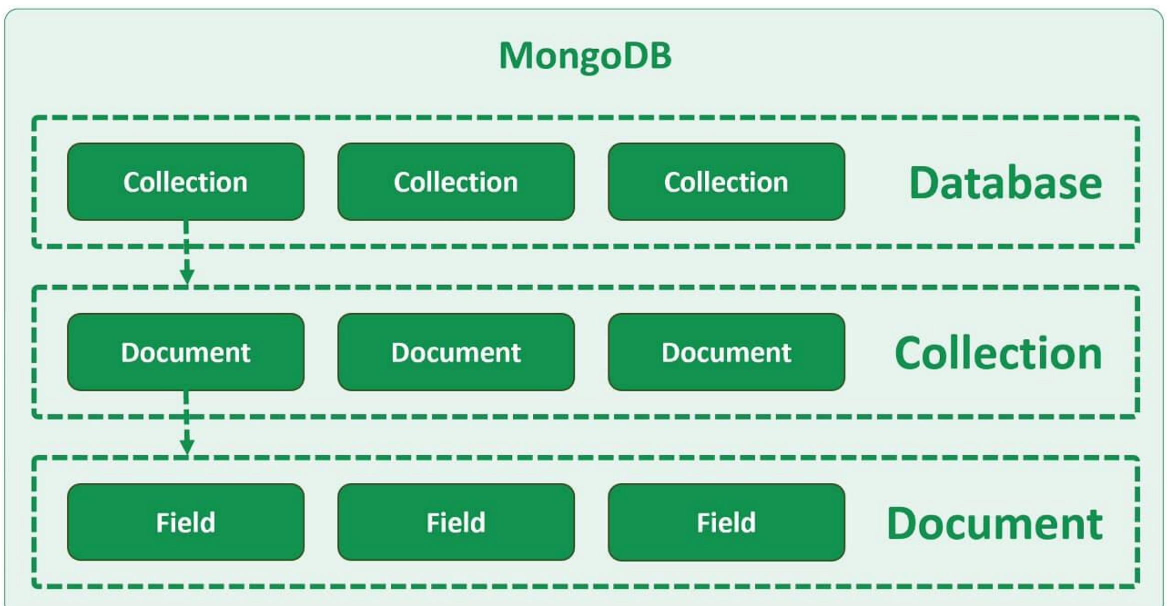
Table	Action	Rows	Type	Collation	Size	Overhead
wp_actionscheduler_actions	Browse Structure Search Insert Empty Drop	45	InnoDB	utf8mb4_unicode_ci	176.0 KiB	-
wp_actionscheduler_claims	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
wp_actionscheduler_groups	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
wp_actionscheduler_logs	Browse Structure Search Insert Empty Drop	121	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
wp_aioseo_cache	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
wp_aioseo_notifications	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_unicode_ci	112.0 KiB	-
wp_aioseo_posts	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
wp_commentmeta	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
wp_comments	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_unicode_ci	96.0 KiB	-
wp_edd_customermeta	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	48.0 KiB	-
wp_edd_customers	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	48.0 KiB	-
wp_links	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
wp_litespeed_url	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
wp_litespeed_url_file	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	96.0 KiB	-
wp_options	Browse Structure Search Insert Empty Drop	381	InnoDB	utf8mb4_unicode_ci	176.0 KiB	-
wp_postmeta	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
wp_posts	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_unicode_ci	80.0 KiB	-
wp_termmeta	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
wp_terms	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
wp_term_relationships	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
wp_term_taxonomy	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
wp_usermeta	Browse Structure Search Insert Empty Drop	19	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
wp_users	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_unicode_ci	64.0 KiB	-
wp_wpforms_tasks_meta	Browse Structure Search Insert Empty Drop	9	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
24 tables	Sum	610	InnoDB	utf8mb4_unicode_ci	1.5 MiB	0 B

Different Types of Databases

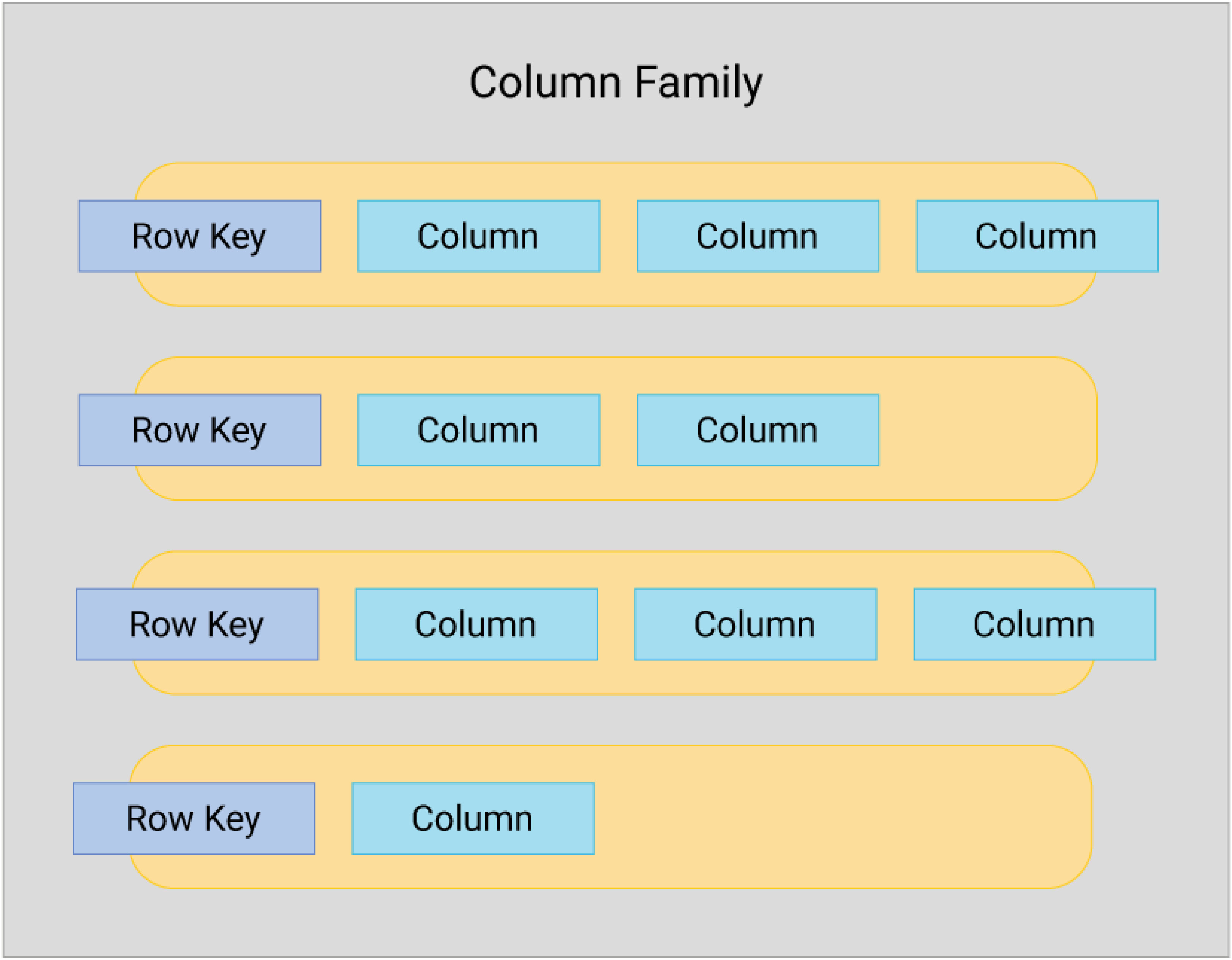
2. NoSQL Databases (Not Only SQL)
   - Non-relational, often used for handling unstructured or semi-structured data
   - Designed for scalability, flexibility, and speed.
   - Types of NoSQL databases:
     - Document-oriented databases: store data as JSON-like documents. Ex: MongoDB, CouchDB
     - Key-value stores: store data as key-value pairs. Ex: Redis
     - Column family stores: store data in columns rather than rows , efficient for analytical queries on large datasets. Ex: Cassandra
     - Graph databases: store data in graph structures to model relationships between entities. Ex: Neo4j, Dgraph, OrientDB
   - Used in large-scale, distributed systems, real-time applications, social media data, IoT, E-commerce ,etc.

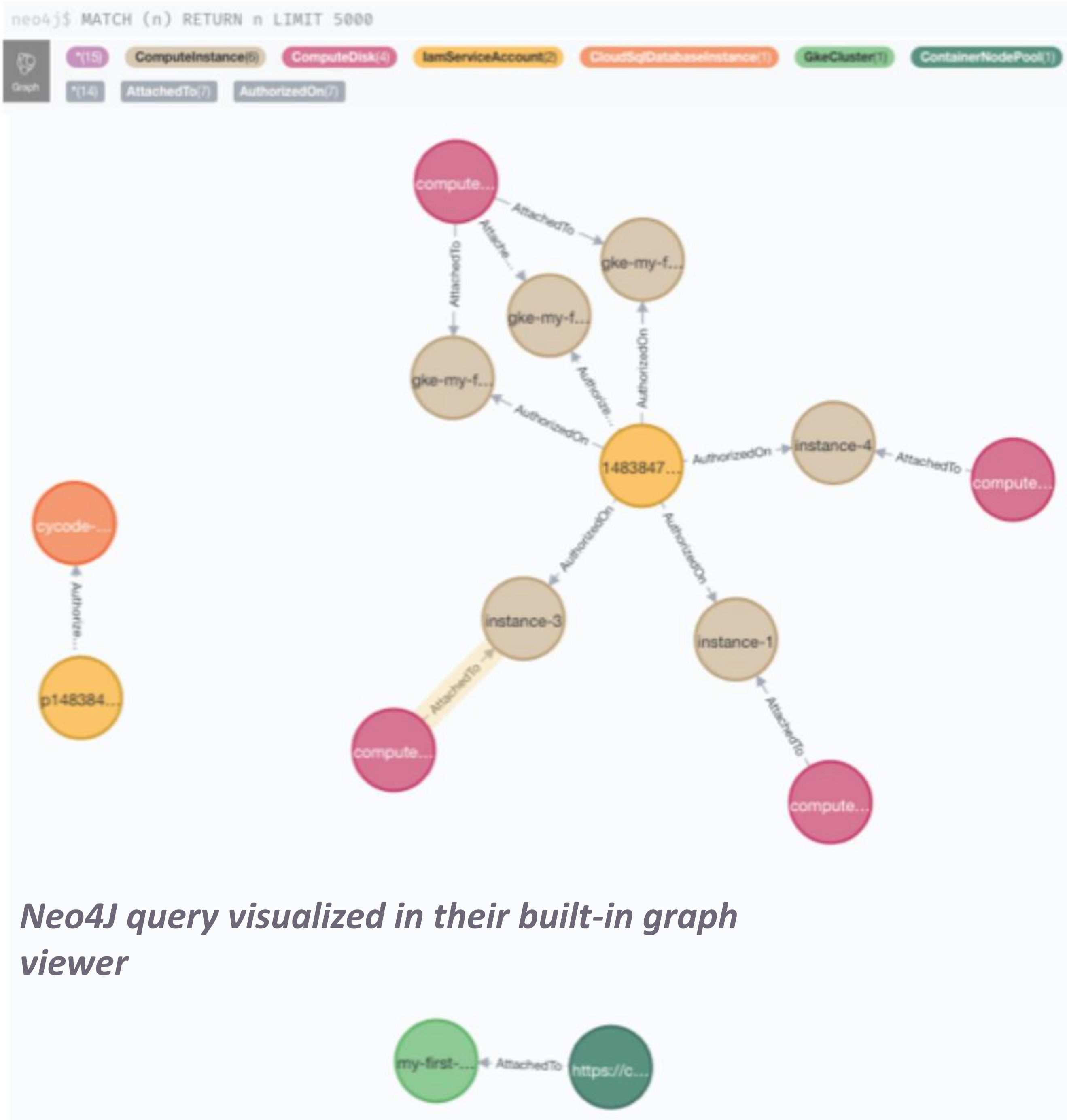

MongoDB
Collection1
Document
Document
Document
Document
Document
Document
Collection2
Document
Document
Document
Document
Document


Collection
{
"_id": "5",
"isActive": false,
"age": 24,
"eyeColor": "brown",
"name": "Ada Compton"
}
{
"_id": "6",
"isActive": false,
"age": 43,
"eyeColor": "blue",
"name": "Andray Conway"
}


MongoDB
Collection
Collection
Collection
Database
Document
Document
Document
Collection
Field
Field
Field
Document


JSON Document 1
{
"_id": 1,
"name": "John",
"graduated": true
}
JSON Document 2
{
"_id": 1,
"name": "John",
"gender": "Male"
}

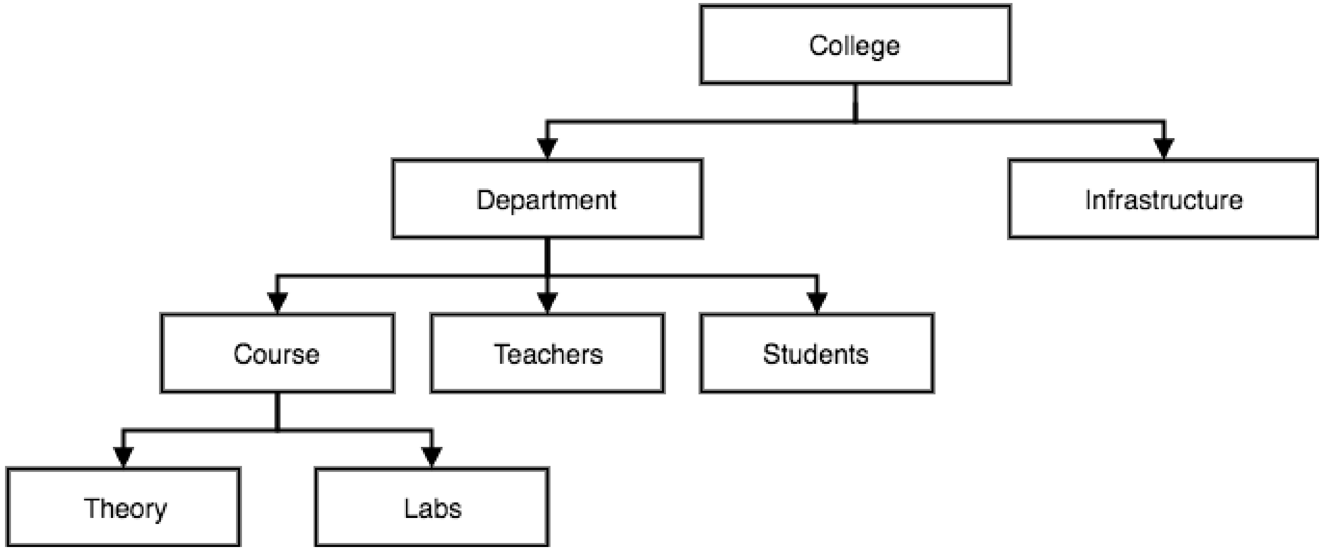

Column Family
Row Key
Column
Column
Column
Row Key
Column
Column
Row Key
Column
Column
Column
Row Key
Column

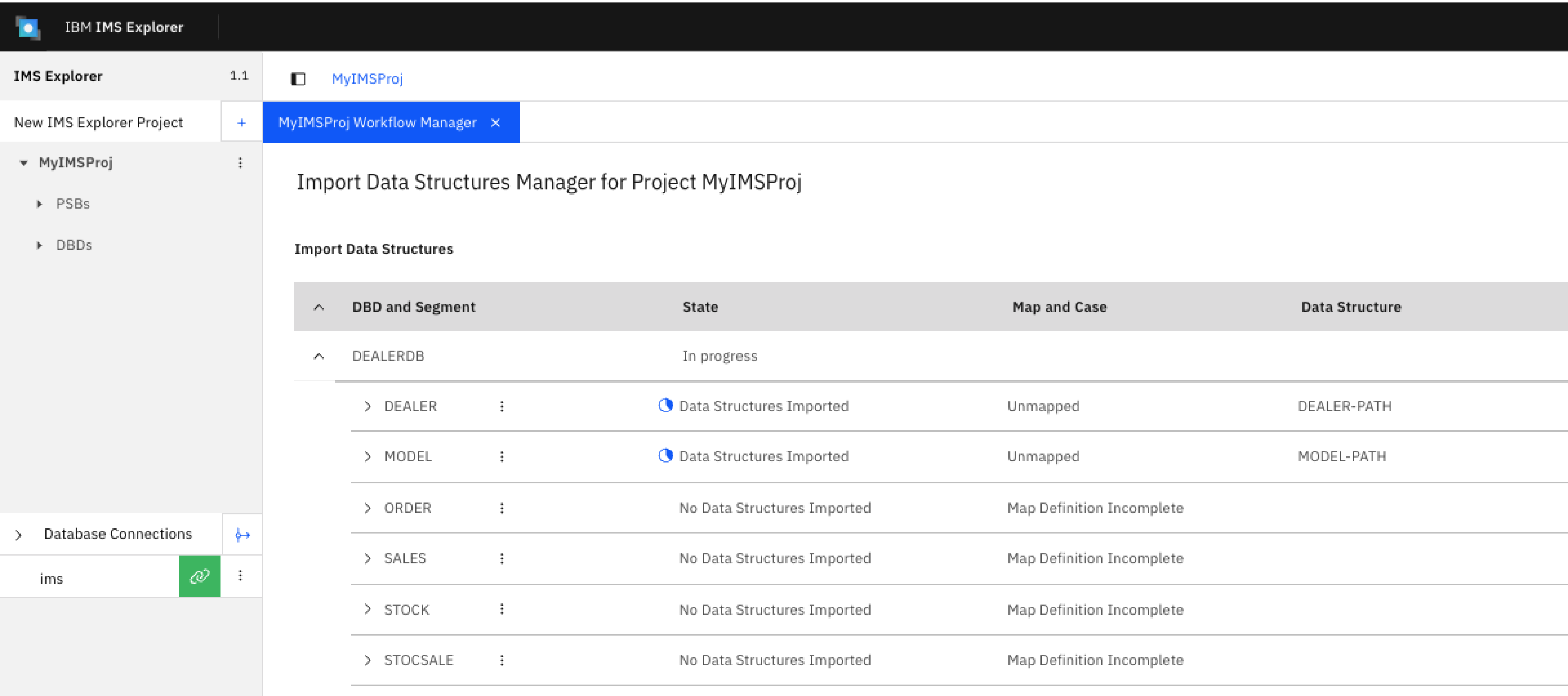


Neo4J query visualized in their built-in graph viewer

Different Types of Databases

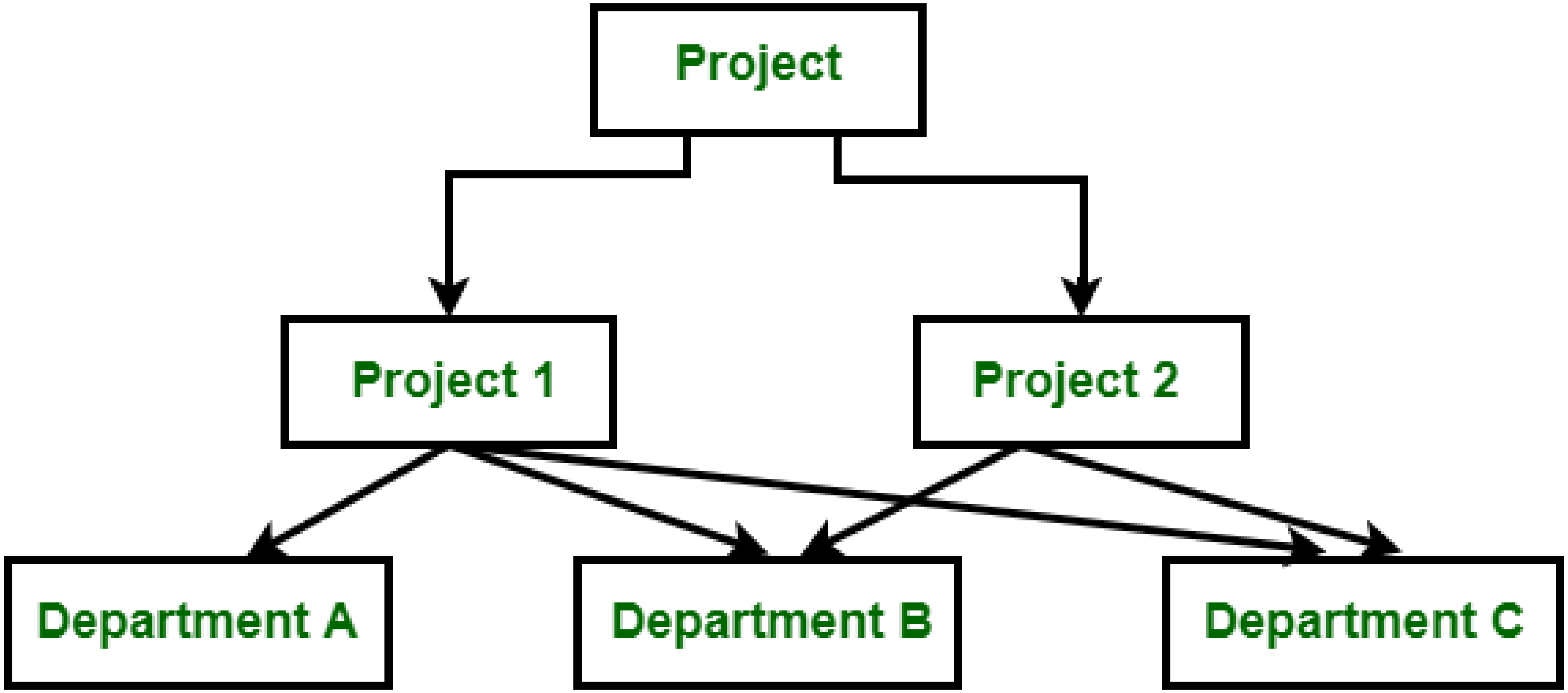
3. Hierarchical Databases
   - Data is organized in a tree-like structure (parent-child relationships).
   - Data is stored in a hierarchical model where each parent can have multiple children but each child has only one parent.
   - Example: IBM’s information management system (IMS)
   - Used in banking and telecommunications systems.


College
Department
Infrastructure
Course
Teachers
Students
Theory
Labs


IBM IMS Explorer
IMS Explorer
1.1
MyIMSProj
New IMS Explorer Project
MyIMSProj Workflow Manager
MyIMSProj
PSBs
DBDs
Database Connections
ims
Import Data Structures Manager for Project MyIMSProj
Import Data Structures
DBD and Segment
State
Map and Case
Data Structure
DEALERDB
In progress
DEALER
Data Structures Imported
Unmapped
DEALER-PATH
MODEL
Data Structures Imported
Unmapped
MODEL-PATH
ORDER
No Data Structures Imported
Map Definition Incomplete
SALES
No Data Structures Imported
Map Definition Incomplete
STOCK
No Data Structures Imported
Map Definition Incomplete
STOCSALE
No Data Structures Imported
Map Definition Incomplete

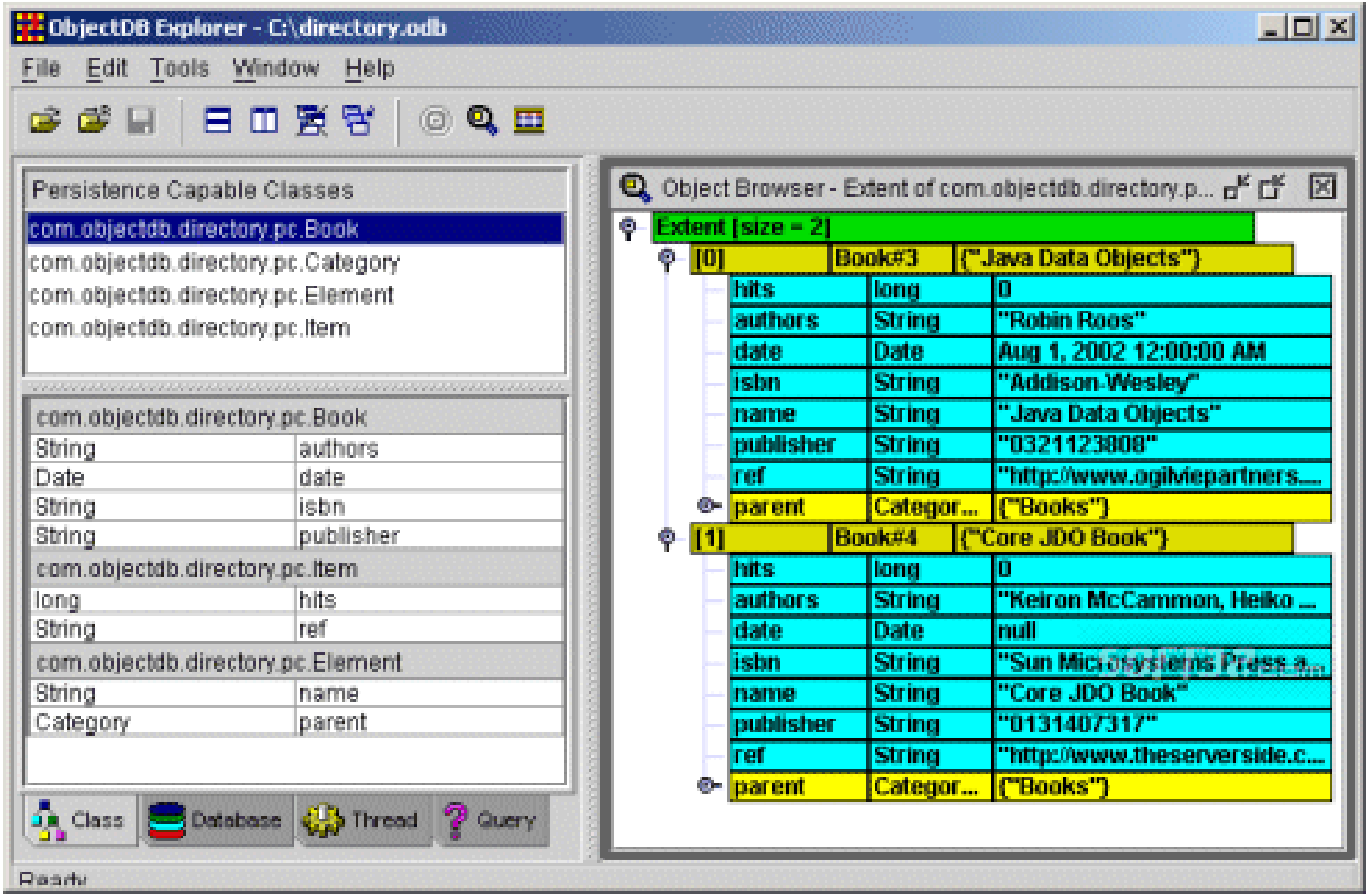
Different Types of Databases

4. Network Databases
   - More flexible than hierarchical databases, where each record can have multiple parent and child relationships.
   - Example: Integrated Data Store (IDS), Integrated Database Management System (IDMS)
   - Used in older mainframe systems or legacy applications where complex many-to-many relationships exist.


Project
Project 1
Project 2
Department A
Department B
Department C

Different Types of Databases

5. Object-Oriented Databases (OODBMS)
   - Stores data in the form of objects, similar to object-oriented programming
   - Objects contain both data (attributes) and methods (functions)
   - Example: db4o, ObjectDB, ZODB
   - Use cases: applications that require complex data representations, like computer-aided design (CAD) and computer-aided manufacturing (CAM) or multimedia system, simulation system


ObjectDB Explorer - C:\directory.odb
File Edit Tools Window Help
Persistence Capable Classes
com.objectdb.directory.pc.Book
com.objectdb.directory.pc.Category
com.objectdb.directory.pc.Element
com.objectdb.directory.pc.Item
com.objectdb.directory.pc.Book
String authors
Date date
String isbn
String publisher
com.objectdb.directory.pc.Item
long hits
String ref
com.objectdb.directory.pc.Element
String name
Category parent
Class
Database
Thread
Query
Ready
Object Browser - Extent of com.objectdb.directory.p...
Extent [size = 2]
[0] Book#3 {"Java Data Objects"}
hits long 0
authors String "Robin Roos"
date Date Aug 1, 2002 12:00:00 AM
isbn String "Addison-Wesley"
name String "Java Data Objects"
publisher String "0321123808"
ref String "http://www.ogilviepartners....
parent Categor... {"Books"}
[1] Book#4 {"Core JDO Book"}
hits long 0
authors String "Keiron McCammon, Heiko ...
date Date null
isbn String "Sun Microsystems Press a...
name String "Core JDO Book"
publisher String "0131407317"
ref String "http://www.theserverside.c...
parent Categor... {"Books"}

Different Types of Databases

6. Time-Series Databases
   - Are optimized for storing and querying time-stamped data, making them ideal for applications that need to track changes over time.
   - Examples: InfluxDB, Prometheus, TimescaleDB
   - Use cases: IoT Data Monitoring, Financial Market Data, Application Performance Monitoring (APM), Energy Grid Monitoring

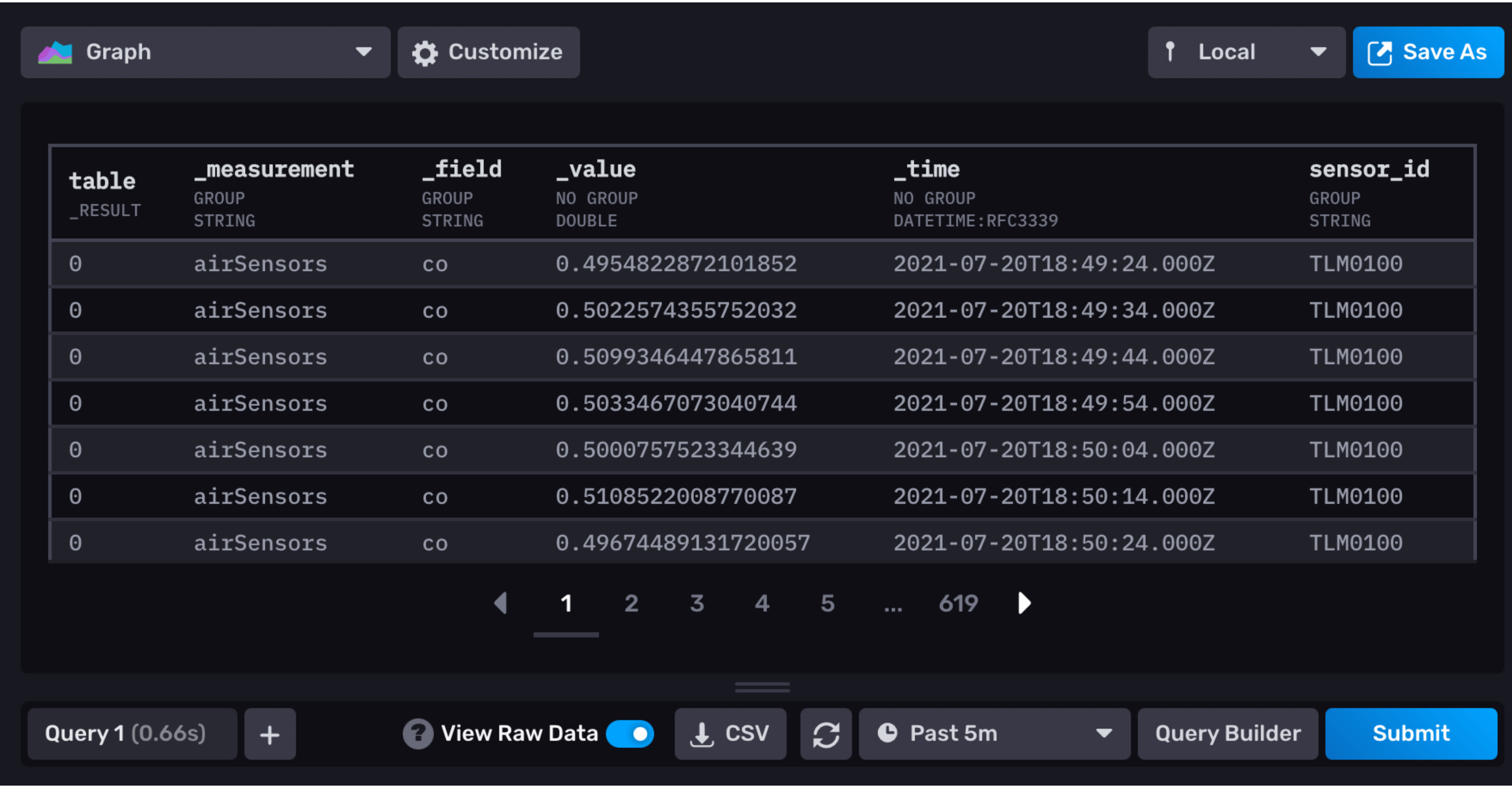


table _RESULT	_measurement GROUP STRING	_field GROUP STRING	_value NO GROUP DOUBLE	_time NO GROUP DATETIME:RFC3339	sensor_id GROUP STRING
0	airSensors	co	0.4954822872101852	2021-07-20T18:49:24.000Z	TLM0100
0	airSensors	co	0.5022574355752032	2021-07-20T18:49:34.000Z	TLM0100
0	airSensors	co	0.5099346447865811	2021-07-20T18:49:44.000Z	TLM0100
0	airSensors	co	0.5033467073040744	2021-07-20T18:49:54.000Z	TLM0100
0	airSensors	co	0.5000757523344639	2021-07-20T18:50:04.000Z	TLM0100
0	airSensors	co	0.5108522008770087	2021-07-20T18:50:14.000Z	TLM0100
0	airSensors	co	0.49674489131720057	2021-07-20T18:50:24.000Z	TLM0100

Database Design

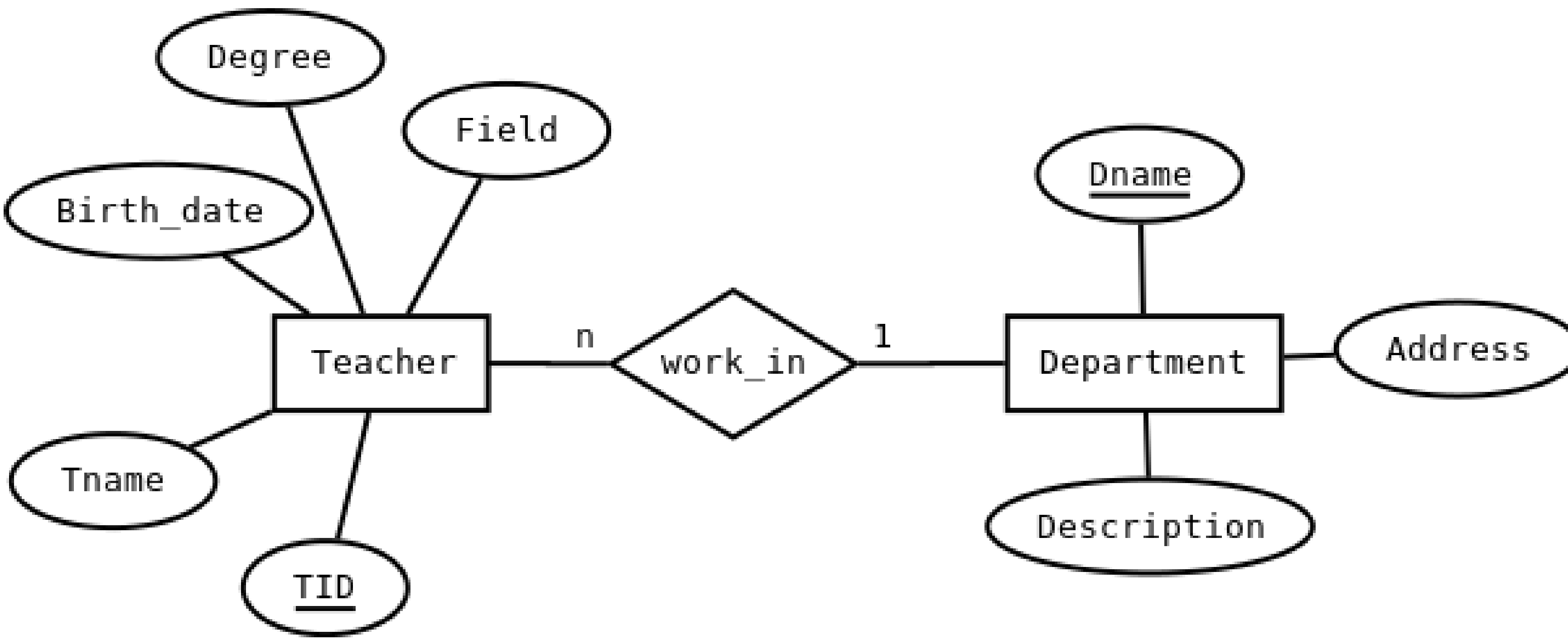
- Database design is a critical aspect of developing effective database systems.
- It involves a set of techniques and principles that guide the organization of data to ensure efficiency, integrity, and scalability.
- Some key concepts and techniques related to database design:

Data Modeling

- Data Modeling is representing and structuring the data requirements systematically.
- Entity-Relationship (ER) modeling is a visual representation of the data entities in a system and relationships between them.
- An ER diagram helps in understanding the structure of the database.

Data Modeling

- Example of ER diagram



Normalization

- Normalization is the process of organizing data to reduce redundancy and improve data integrity.
- It involves dividing a database into tables and defining relationships between them to eliminate duplicate data.
- Normalization structures the database according to a series of rules called normal form.
- The most common normal forms are 1NF, 2NF, 3NF and BCNF.

Normalization

- Example

EID	Name	DID	Dname	Skills
1	Kevin	201	R&D	C, Perl, Java
2	Barbara	224	IT	Linux, Mac
3	Jake	201	R&D	DB2, Oracle, Java

DID	Dname
201	R&D
224	IT

EID	Name	DID
1	Kevin	201
2	Barbara	224
3	Jake	201

EID	Skill
1	C
1	Perl
1	Java
2	Linux
2	Mac
3	DB2
3	Oracle
3	Java

Denormalization

- Denormalization is the process of intentionally introducing redundancy into a database design for performance optimization.
- This technique can improve read performance by reducing the number of joins required in queries.
- Denormalization is often used in data warehousing and reporting systems where read performance is prioritized over write performance.
  - Example: combining the two tables back into a single table to improve query performance.

Denormalization

- Example

EID	Name	DID
1	Kevin	201
2	Barbara	224
3	Jake	201

EID	Skill
1	C
1	Perl
1	Java
2	Linux
2	Mac
3	DB2
3	Oracle
3	Java

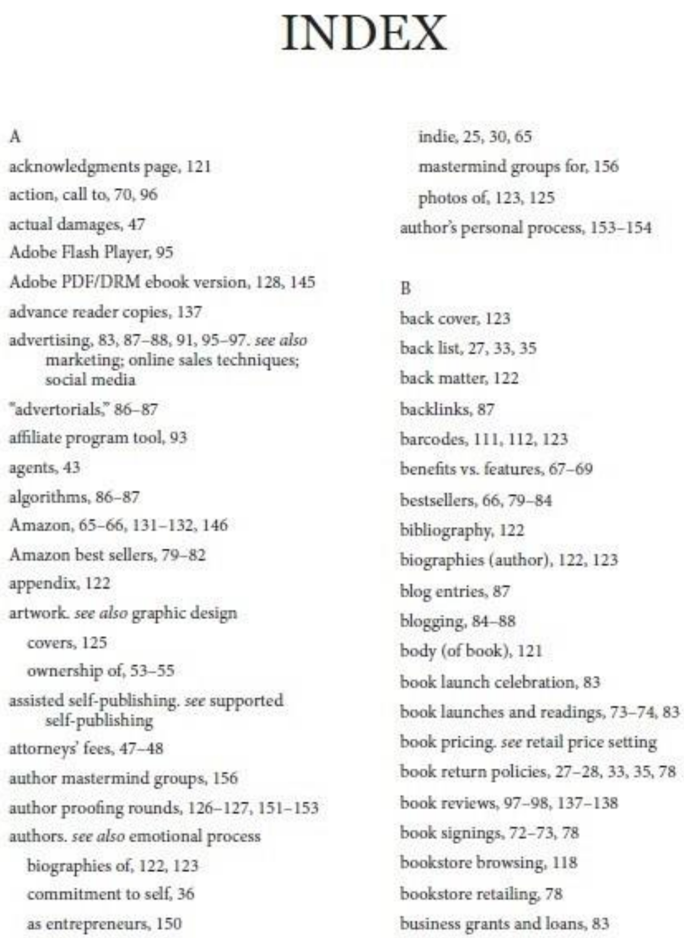
DID	Dname
201	R&D
224	IT

EID	Name	Gender	DID	Dname	Skill
1	Kevin	M	201	R&D	C
1	Kevin	M	201	R&D	Perl
1	Kevin	M	201	R&D	Java
2	Barbara	F	224	IT	Linux
2	Barbara	F	224	IT	Mac
3	Jake	M	201	R&D	DB2
3	Jake	M	201	R&D	Oracle
3	Jake	M	201	R&D	Java

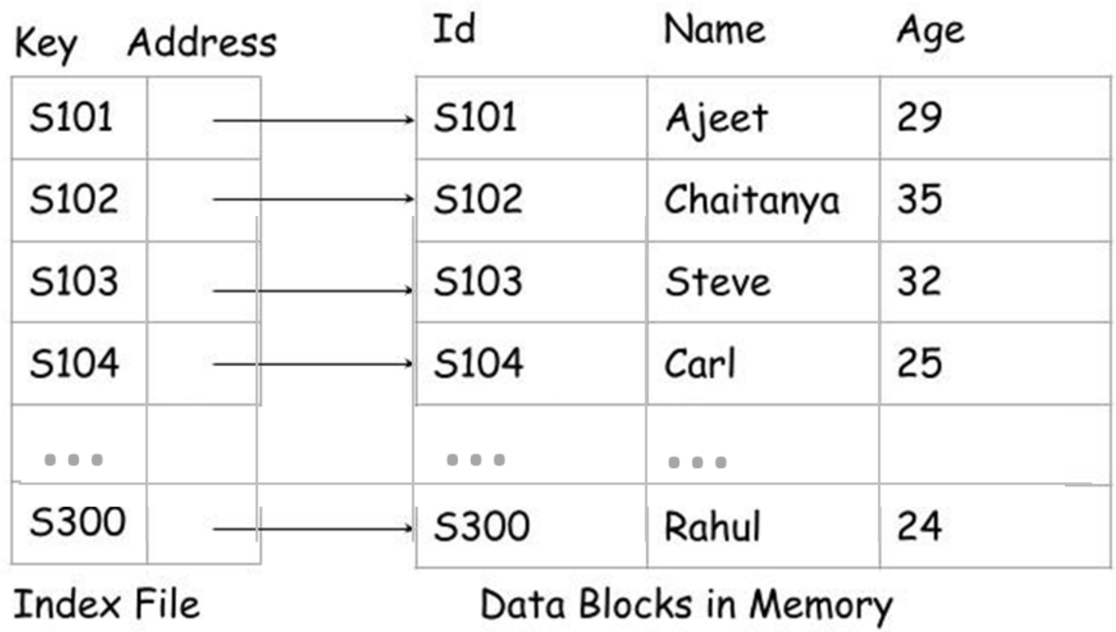
Indexing

- Indexing is a technique that improves the speed of data retrieval operations on a database table.
- An index is a data structure that allows for quick lookup of rows based on the values of one or more columns.
- Primary index is automatically created when a primary key is defined. It enforces uniqueness.

Indexing



INDEX



Dense Indexing

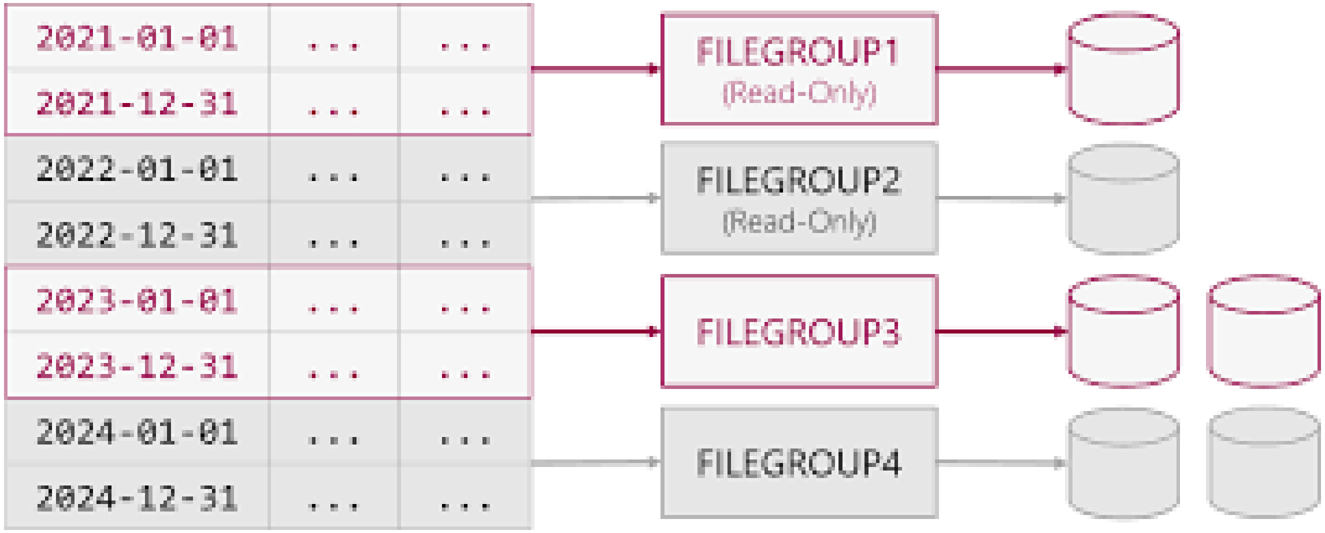
Partitioning in database

- Partitioning divides large tables into smaller, more manageable segments (partitions), which can be processed independently to improve performance.
  - Horizontal Partitioning (Sharding): splits the rows of a table across multiple partitions.
    - Example: orders could be partitioned by date, with orders from each year stored in separate partitions,
  - Vertical Partitioning: splits the columns of a table into different tables or partitions, often done for performances or security reasons.
    - Example: sensitive columns like passwords could be stored in one partition, and general customer information in another.

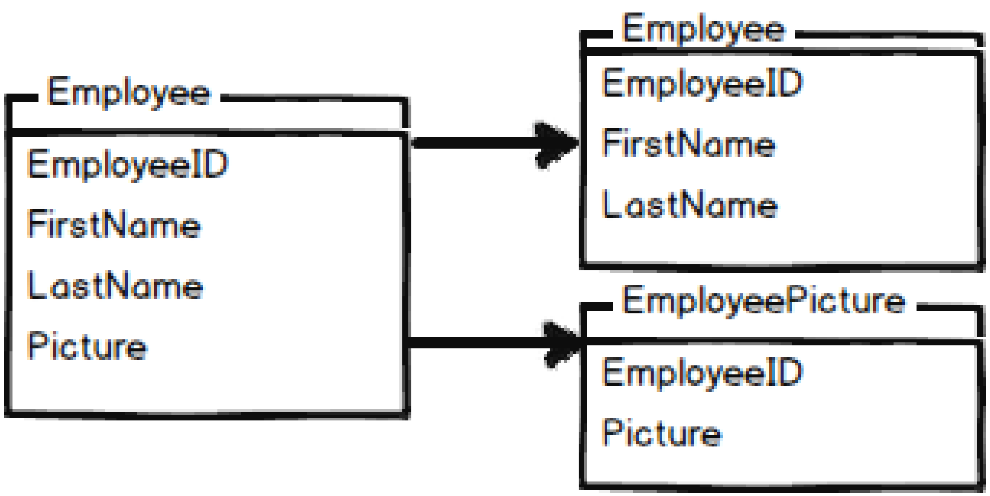
Partitioning in database

- Example

Horizontal Partitioning



Vertical Partitioning



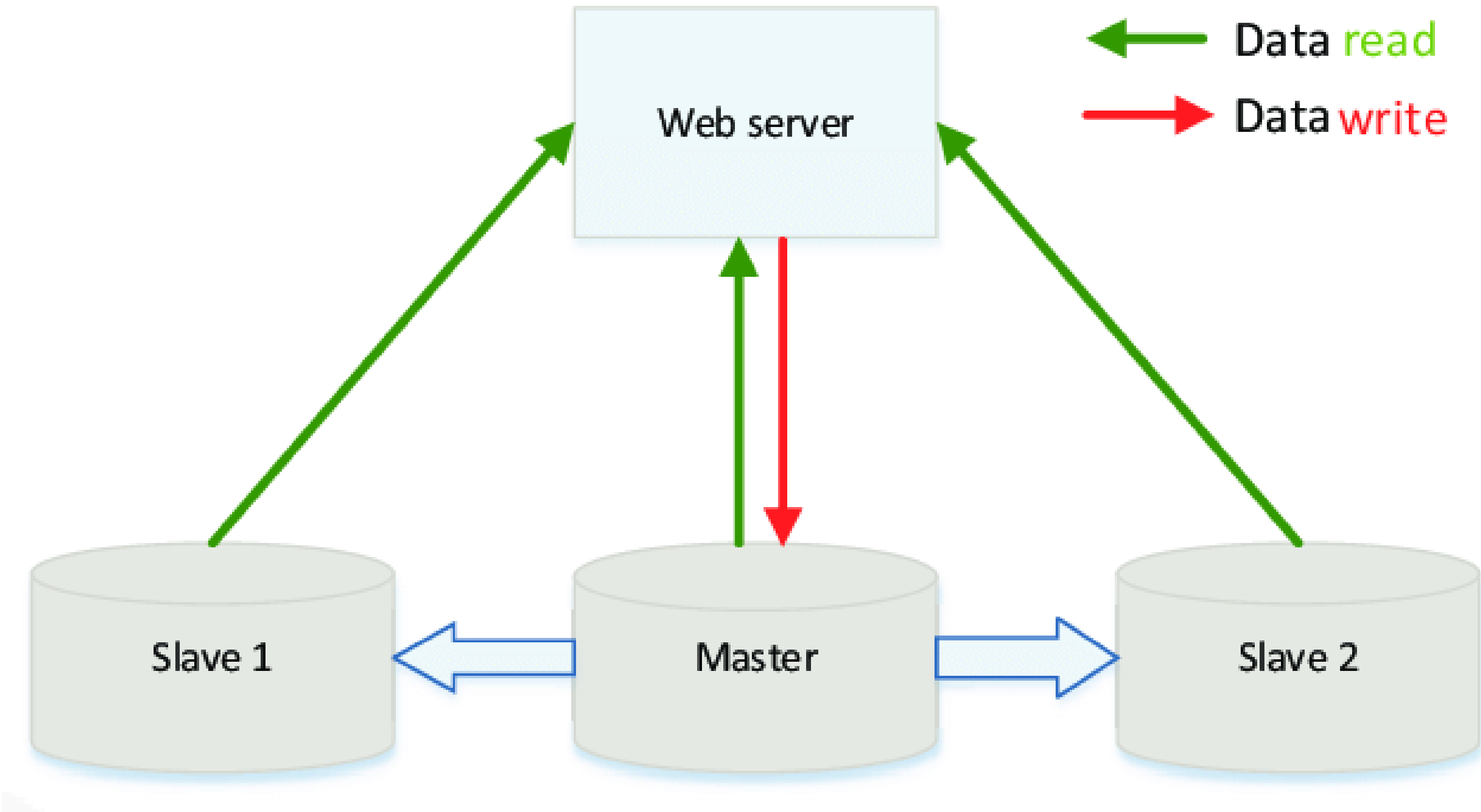
Replication in databases

- Replication in databases is the process of copying and maintaining database objects, such as tables, across multiple database servers.
- It involves creating copies of data across multiple locations to ensure data availability, fault tolerance, and sometimes to distribute read loads.

Replication in databases

a. Master-slave replication: A master database handles writes, while read operations are offloaded to one or more replicas (slaves).
   - Suitable for read-heavy applications where read operations can be distributed across multiple slave servers.
   - Example: in an e-commerce system, the master handles all customer orders, while slave databases serve read requests for product details and user profiles.

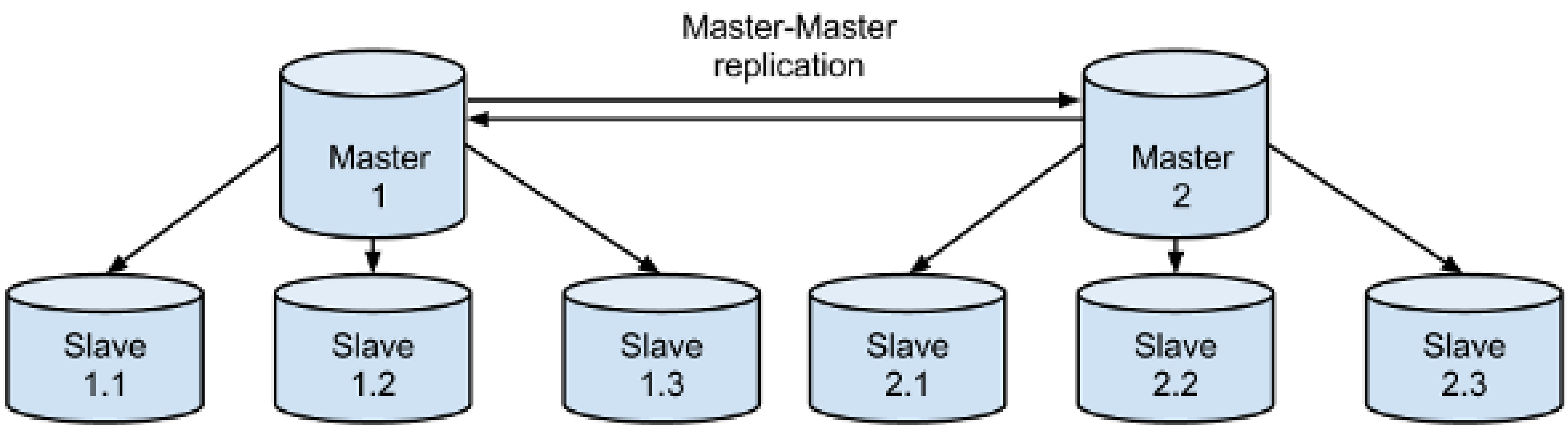
Replication in databases



Replication in databases

b. Master-Master Replication: all nodes are capable of handling both reads and writes.
   - Changes made on one master are replicated to the other master.
   - This improves availability but can lead to complexity in maintaining data consistency.
   - Useful in high-availability scenarios where both servers need to accept writes and provide redundancy.
   - Example: In a global application where users can write data from different regions, both servers can accept updates and replicate changes.

Replication in databases



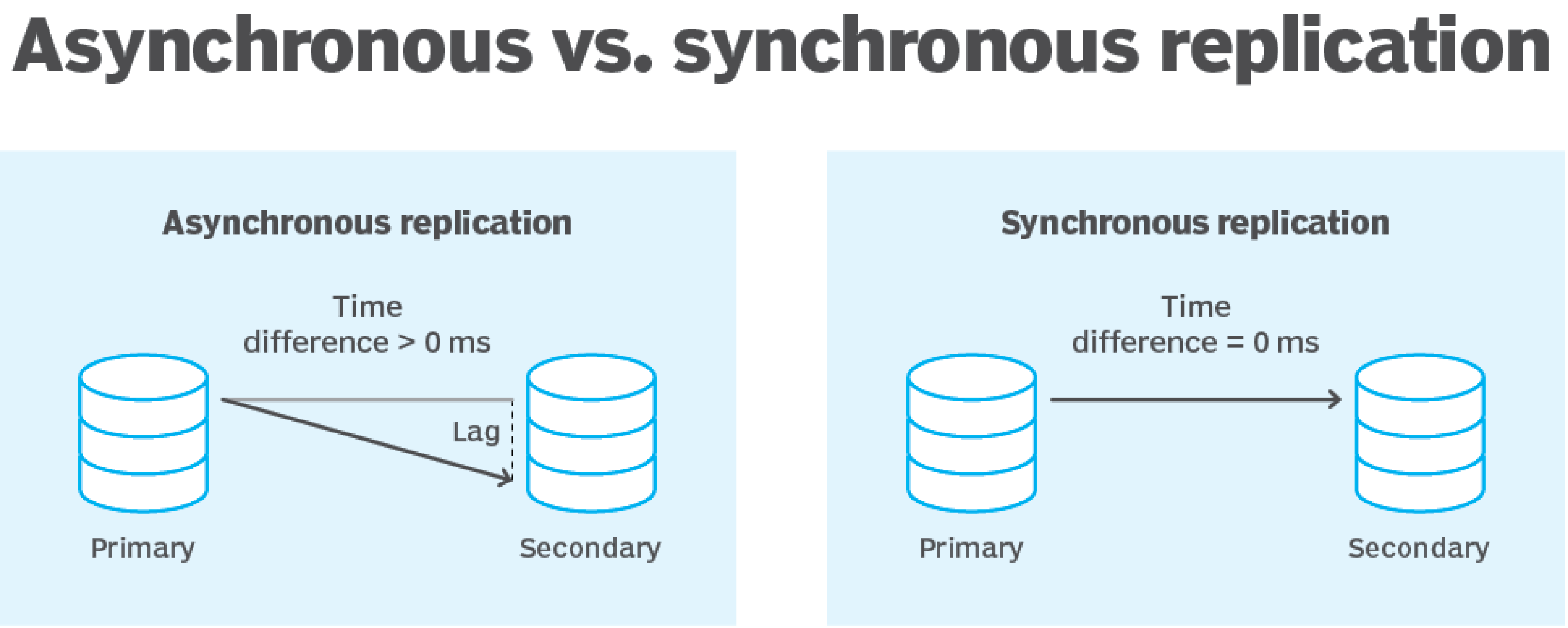
Replication in databases

c. Asynchronous Replication: changes made on the master are not immediately replicated to the slaves. The slaves can lag behind the master, which may lead to stale data in the replicas.
   - Suitable for scenarios where immediate consistency is not critical, allowing for better performance during high write loads.
   - Example: a social media app where user feeds can tolerate slight delays in updates.

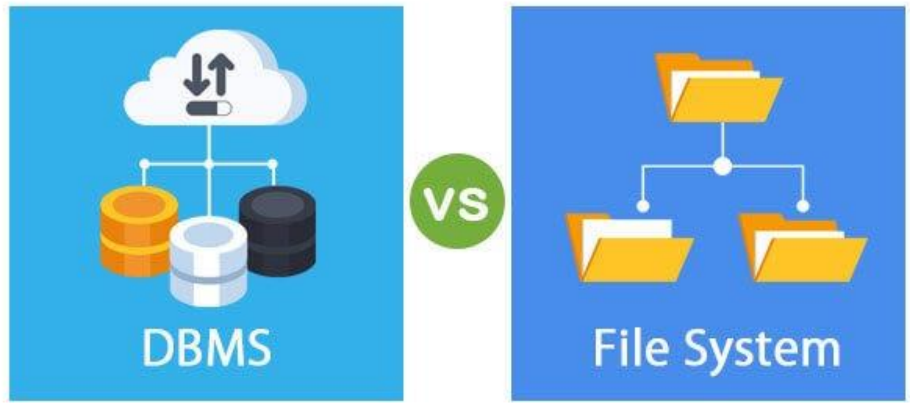
Replication in databases

d. Synchronous Replication: changes made on the master are replicated to the slaves before the transaction is considered complete.
   - All replicas are up-to-date before confirming the write operation
   - Essential for applications requiring high data consistency and reliability
   - Example: financial applications where transaction integrity is critical, and all replicas must reflect the same state before proceeding.

Replication in databases



Database vs. File



- It is possible to store data in simple files or in a database.
- The choice between using a database and a file for data storage depends on various factors, including the complexity of the data, access patterns, scalability requirements, and the specific use case.

Database vs file

1. Data complexity:
   a. Database: if data has complex relationships (multiple entities with foreign key), a database is ideal for managing these relationships and ensuring data integrity.
   b. File: for simpler data structures that do not require relationships, such as configuration files or logs.

Database vs file

2. Data Volume:
   a. Database: for applications expected to handle large volumes of data or data that grows over time
   b. File: if the expected data volume is small and unlikely to grow significantly
3. Data Access Patterns:
   a. Database: if the application requires complex queries, aggregations, and reporting
   b. File: for simple read operations where data does not change often, especially if the data can be cached

Database vs file

4. Concurrency:
   a. Database: when multiple users or processes need to access or modify the data simultaneously, databases manage concurrency and provide transaction support to maintain data integrity
   b. File: if only one user or process will access the data at a time, or if the application can handle potential conflicts through manual checks

Database vs file

5. Data Integrity and Security:
    a. Database: databases enforce data integrity through constraints like primary keys and foreign keys and offer robust security features, including user authentication and authorization.
    b. File: if data integrity and security are less critical, or if the application can enforce these through other means

Database vs file

6. Backup and Recovery Needs:
    a. Database: most databases have built-in backup and recovery mechanisms to ensure data resilience and quick restoration in case of failure.
    b. File: backup and recovery for files must be managed manually, which can be more complex and error-prone, especially as the number of files grows.

Database vs file

7. Development Speed and Flexibility:
   a. Database: if you anticipate frequent changes to the data model or require a flexible schema, a database can handle these changes more gracefully, especially with migration tools.
   b. File: for rapid prototyping or simple applications where the data structure is unlikely to change significantly, files can enable faster development without the overhead of a database.

Database vs file

8. Cost considerations:
    a. Database: while databases offer many features, they may also come with licensing costs, operational costs for hosting and maintenance
    b. File: using files can be less expensive initially, but can incur higher maintenance costs as complexity grows.
9. Long-Term vs Temporary Data:
    a. Database: for long-term data storage and management, when data needs to be retained for regulatory or operational reasons
    b. File: if the data is temporary or ephemeral (session data)

Aspect	File System	Database
Data Structure	Simple data structures, lightweight applications	Complex data relationships, large volumes
Data Integrity	Does not enforce data integrity or relationships	Enforces rules and constraints (like primary keys, foreign keys)
Redundancy Control	High data redundancy due to lack of central control	Data is normalized to avoid redundancy and inconsistency
Data Manipulation	Access through file read/write operations	Access through query languages like SQL
Concurrency Control	Limited concurrent access without conflicts	Advanced concurrency management (ACID properties)
Security	Basic security like file permissions	Fine-grained control over who can access what data
Backup/Recovery	Manual or limited automatic backup	Automated and systematic backup and recovery features
Complex Queries	Difficult to execute complex search queries efficiently	Optimized query execution using indexing and optimization algorithms

Database vs file

When to Use a Database

- Structured data
- Multi-user application
- Transactional System
- Complex queries
- Scalability

When to Use a file

- Simple data storage
- Configuration files
- Lightweight applications
- Rapid prototyping
- Temporary data

Reflection

- Which database do you think powers social Facebook, Twitter, Netflix?
- Can you think of an application where a file system would work better than a database?

Summary

No	Topic	Summary
1	Notion of Database	Database is a collection of persistent data that is used by informatio n system/applications.
2	Database vs File system	Database is so popular because: •Multiple different user access, huge volume of data, tool for data access, backup, restore and recovery •Reduce dependency between data management and programming •Reduce development time of program
3	Database Management System	A software which: •Allow the storage of large amount of data and allow efficient access to data •Manage database •Allow data manipulation from third party by using query language •Enable durability and security of data •Control concurrent access
4	Different Type of Database	Type of database refers to the type of data model which is used for designing the database.