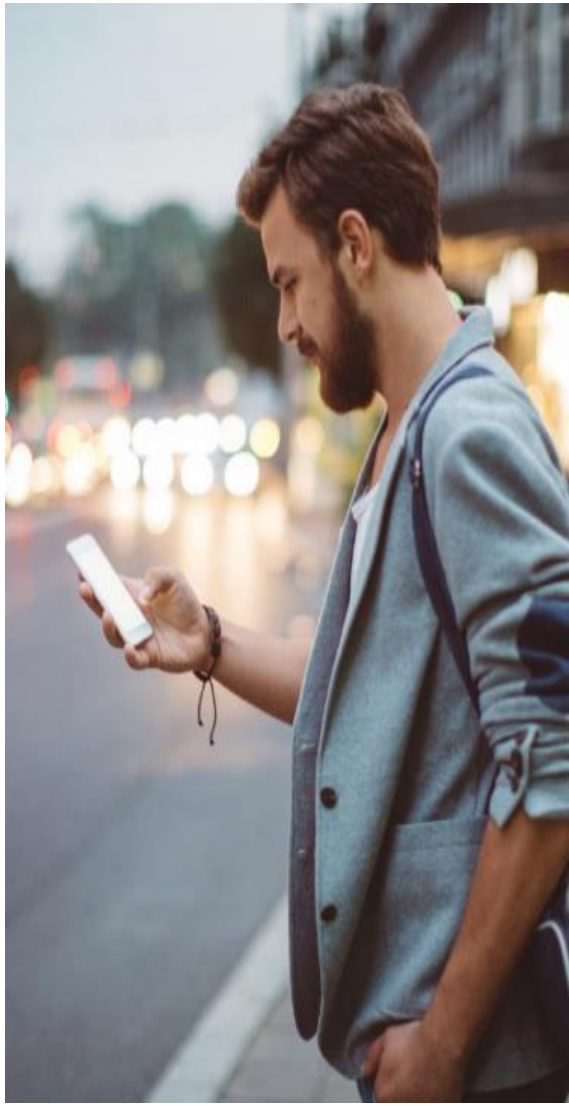
Unit 1: The Database Models, Languages and Architecture

CS352-ADVANCED DATABASES

DR. JOHN CONKLIN

Agenda

1. Overview of the Presentation
    - This presentation will provide an introduction to the fundamental concepts of database models, languages, and architecture.
2. Key Topics
    - Introduction to Databases
    - Database Environment
    - The Relational Model
3. Learning Objectives
    - Understand the purpose and evolution of database management systems (DBMS)
    - Explore the components and roles within a database environment
    - Gain knowledge of the relational data model and its characteristics
    - Recognize the architecture of a relational database system
4. Importance of Database Concepts
    - Databases are the backbone of modern information systems
    - Understanding database models, languages, and architecture is essential for effective database design, development, and management
    - This foundational knowledge will prepare you for more advanced database topics and applications
5. Outline of the Presentation
    - The presentation will cover the three main topics listed above, providing a comprehensive introduction to the key concepts and principles.



Introduction To Databases

A **database** is an organized collection of data that allows for efficient storage, retrieval, and management of information. Databases are commonly used in applications ranging from websites and business systems to scientific research and healthcare. They can be managed manually or through **Database Management Systems (DBMS)** like MySQL, PostgreSQL, or Microsoft SQL Server. Databases often follow a **relational model**, using tables, keys, and relationships to maintain data integrity and reduce redundancy.

Databases

Definition of a Database

- A database is an organized collection of data that is stored and managed in a way that allows for efficient retrieval, manipulation, and management of information.

Purpose of Databases

- Databases serve as a central repository for data, allowing multiple users and applications to access and interact with the same data.
- They provide a structured and organized way to store, manage, and maintain data, ensuring data integrity, security, and accessibility.

Evolution of Database Management Systems (DBMS)

- Early database systems were file-based, with data stored in separate files, leading to data redundancy and integrity issues.
- The development of database management systems (DBMS) addressed these limitations by providing a centralized, integrated, and managed approach to data storage and retrieval.
- DBMS software, such as MySQL, Oracle, and SQL Server, have evolved over time to offer advanced features, improved performance, and support for complex data structures and applications.

Databases

Benefits of Using a DBMS

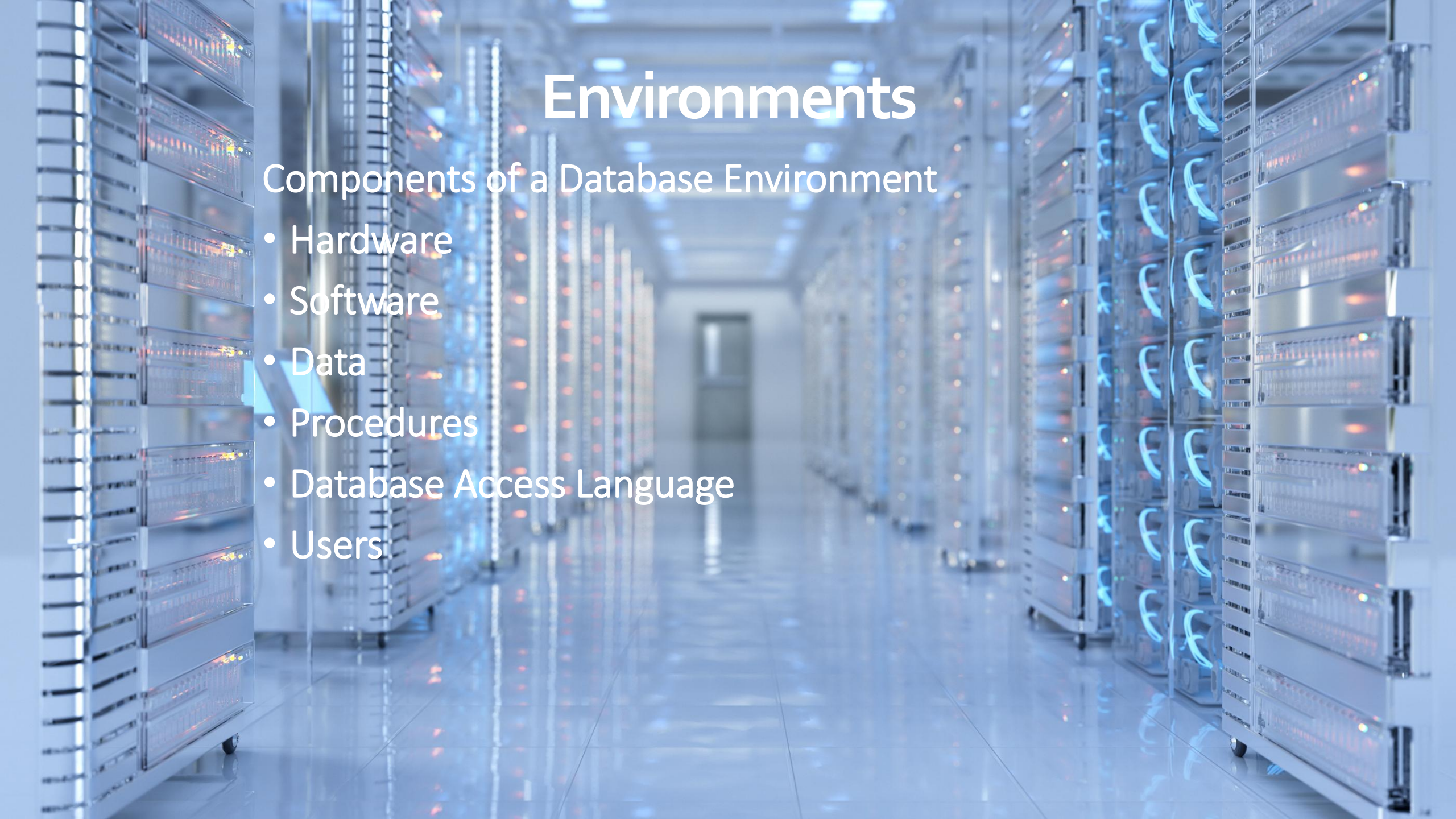
- Data independence: Separation of application logic from physical data storage
- Reduced data redundancy: Elimination of duplicate data storage
- Improved data integrity: Enforcement of data consistency and validity rules
- Efficient data access: Optimized data retrieval and manipulation through indexing and query optimization
- Centralized data management: Allowing multiple users and applications to access and interact with the same data

Key Characteristics of a DBMS

- Data definition: Providing a mechanism to define the structure and schema of the database
- Data manipulation: Offering tools and languages to insert, update, delete, and query data
- Data control: Implementing security, concurrency control, and backup/recovery mechanisms
- Data dictionary: Maintaining metadata about the database and its objects

Database Environment

A **database environment** refers to the complete system setup that supports the storage, processing, and management of data within a database. It encompasses the hardware, software (such as the database management system), **data**, **procedures**, and **users** involved in database operations. This environment ensures that data is accessible, secure, and consistent. It also includes tools for backup, recovery, and user access management to maintain data integrity and availability.



Environments

Components of a Database Environment

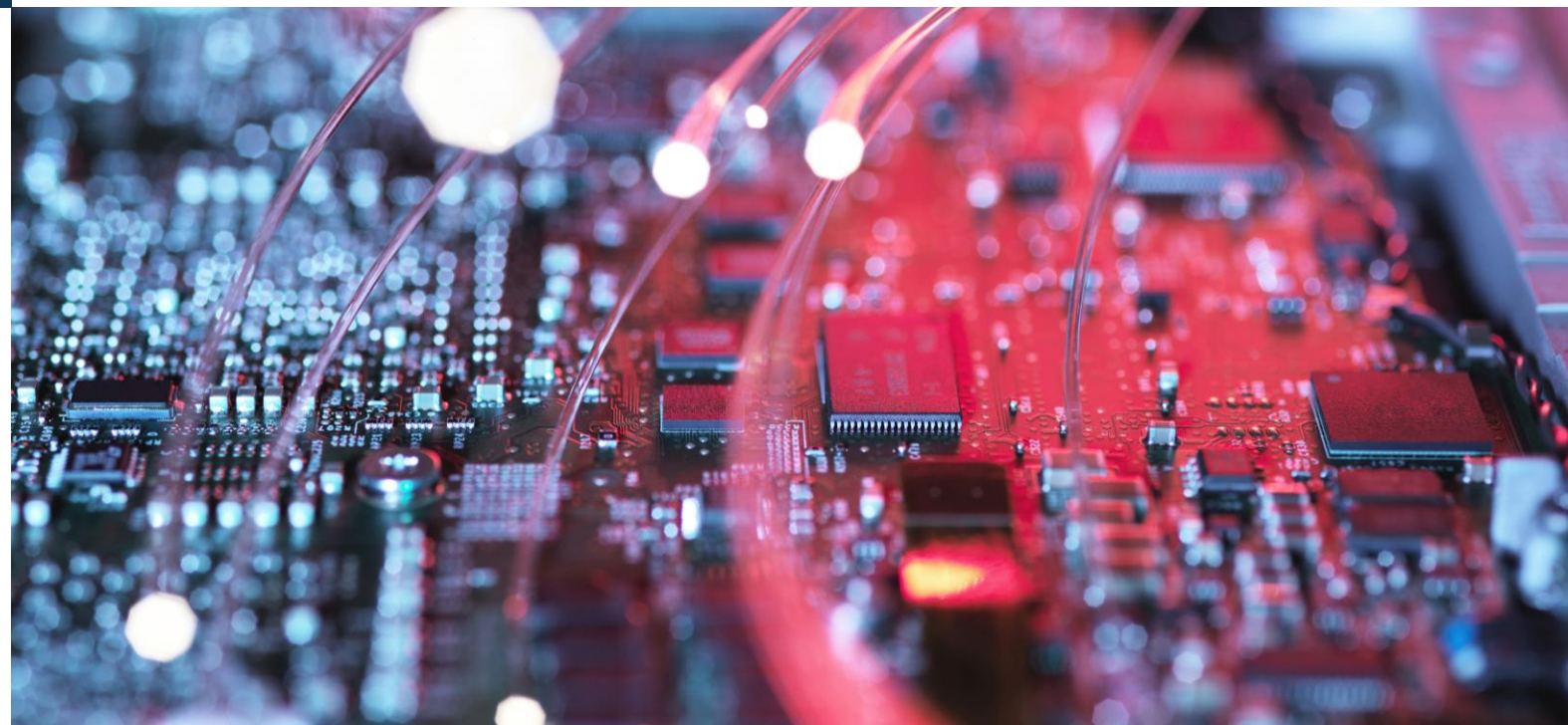
- Hardware
- Software
- Data
- Procedures
- Database Access Language
- Users

Hardware

This refers to the **physical devices** on which the database system runs. It includes:

- **Servers** (to host the DBMS and data)
- **Client machines** (user devices accessing the database)
- **Storage devices** (for data persistence)

The hardware components of a database environment are crucial for the efficient and reliable operation of the database system. The primary hardware elements include powerful servers that host the database management system (DBMS) and store the actual data. These servers are typically equipped with high-performance processors, large amounts of memory, and high-capacity storage devices, such as hard disk drives (HDDs) or solid-state drives (SSDs), to handle the storage and retrieval of large volumes of data.

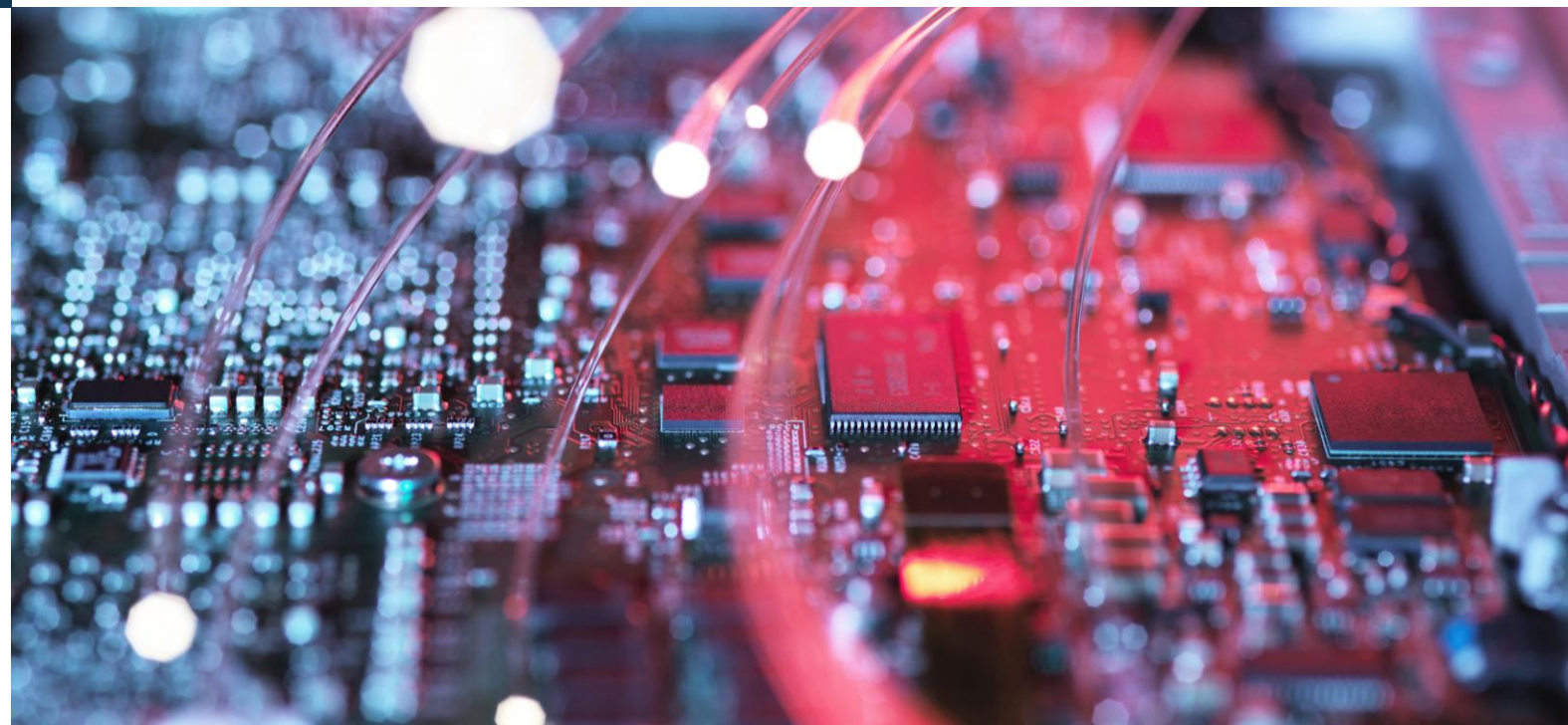


Software

Software includes:

- The **Database Management System (DBMS)** — the core system that manages database structure and access.
- **Application programs** — custom or commercial software that interfaces with the DBMS.
- **Operating systems** — that support the execution of the DBMS and applications.

The software components of a database environment are crucial for managing the storage, organization, and retrieval of data. At the core of the database environment is the database management system (DBMS), which is the software responsible for controlling and facilitating all interactions with the database. Popular DBMS solutions include MySQL, Oracle Database, Microsoft SQL Server, PostgreSQL, and SQLite, each with its own set of features, capabilities, and target use cases.

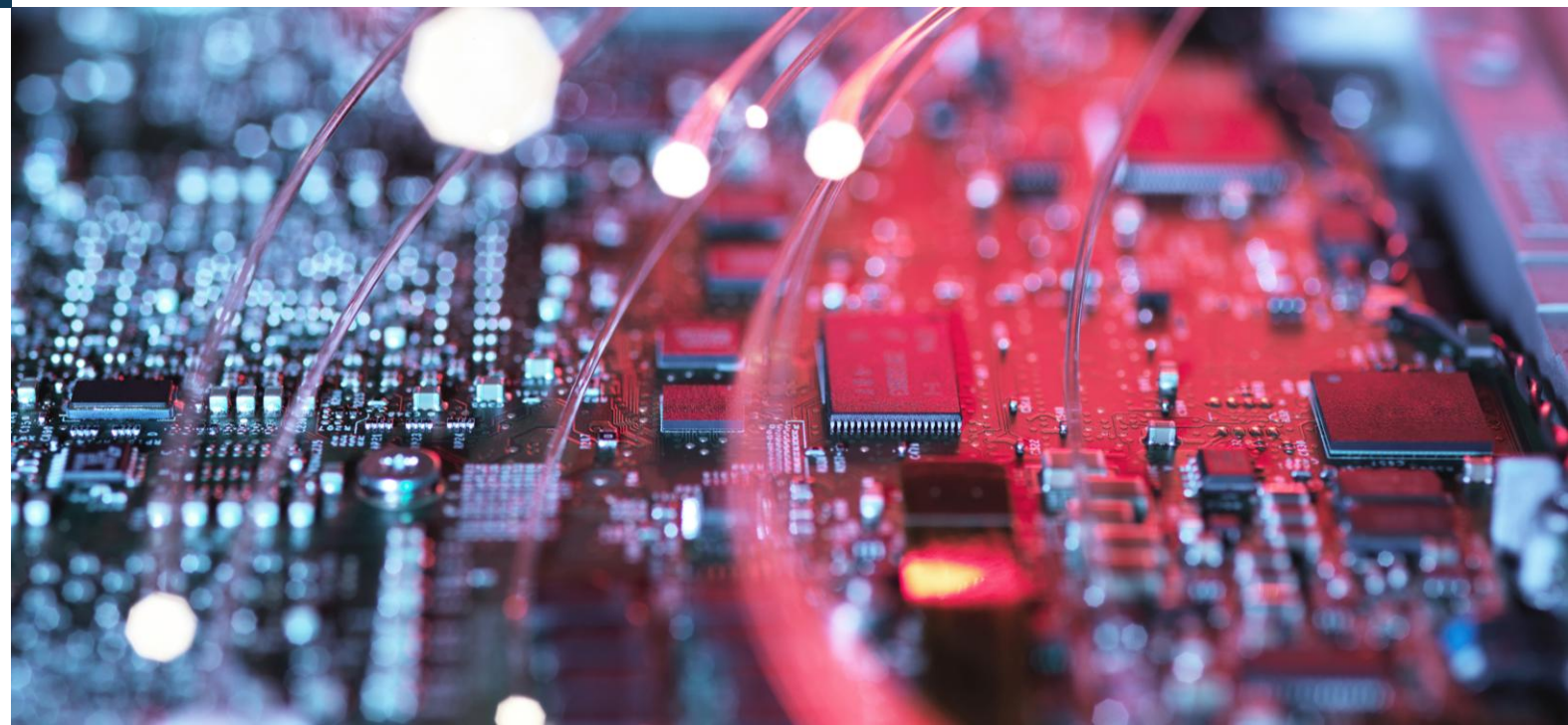


Data

Data is the **central component** of the database environment. It includes:

- **Operational data** used by the organization
- **Metadata** (data about data), which describes the structure, rules, and constraints of the database

The data stored in a database is typically organized into tables, which consist of rows (records or tuples) and columns (attributes or fields). These tables and their relationships form the core of the database's schema, defining the structure and rules governing the data. The efficient storage, indexing, and querying of this data is a primary responsibility of the database management system.

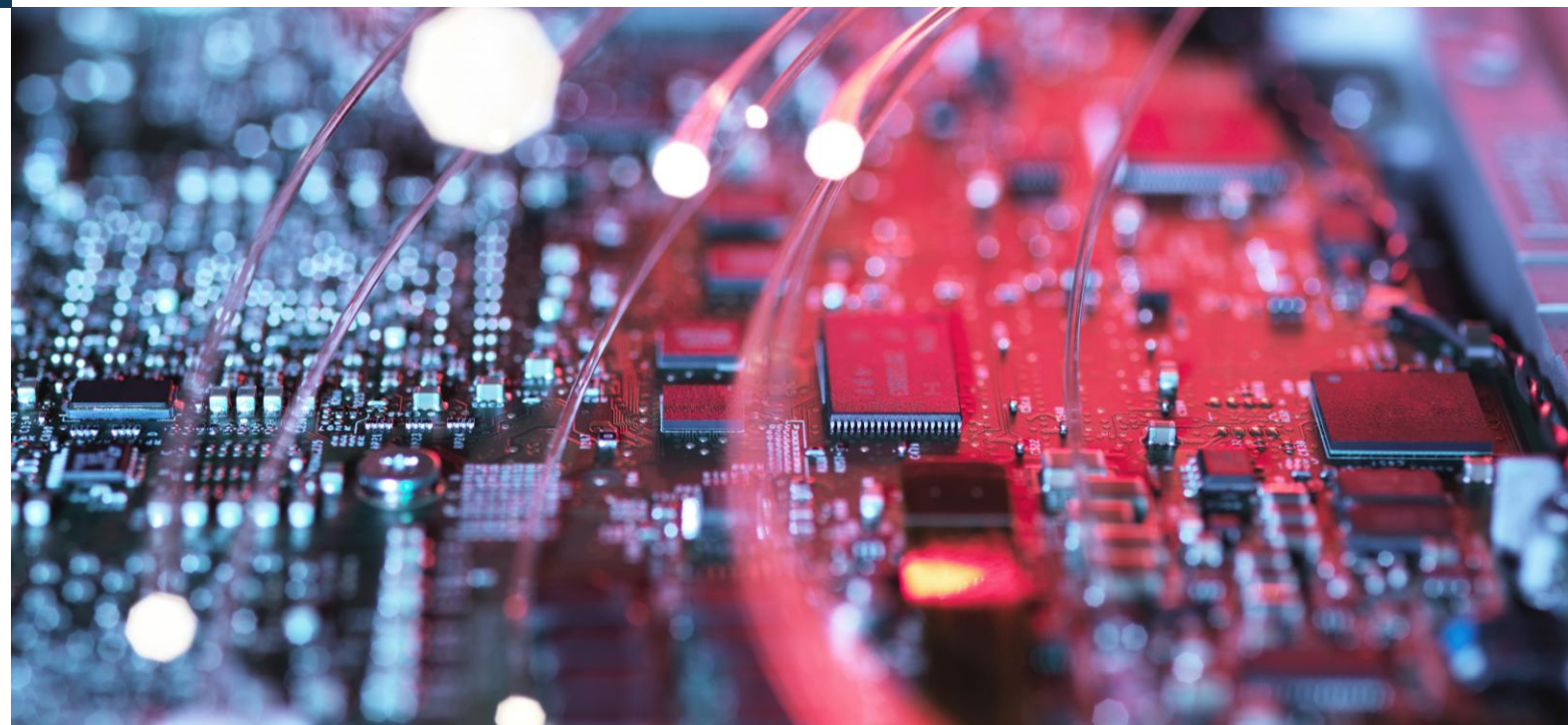


Procedures

Procedures are the **instructions and rules** that govern:

- How the database is designed and maintained
- How users interact with the database
- Backup and recovery processes
- Data entry and quality control methods

Procedures in a database environment refer to the documented methods and rules that guide how data is entered, managed, maintained, and retrieved. They ensure consistent operations, help enforce data integrity, and support tasks such as backup, recovery, and security. Well-defined procedures also help users and administrators interact with the database system efficiently and reduce the risk of errors or data loss.



Procedures

Change Management:

Procedures for managing changes to the database schema, application code, or other components of the database environment, ensuring a controlled and coordinated approach to updates and modifications.

Disaster Recovery and Business Continuity:

Documented plans and procedures to ensure the database can be restored and made available in the event of a major system failure, natural disaster, or other disruptive events.

Monitoring and Alerting:

Processes for continuously monitoring the database, detecting and responding to issues or anomalies, and triggering appropriate alerts or notifications.

Database Backup and Recovery:

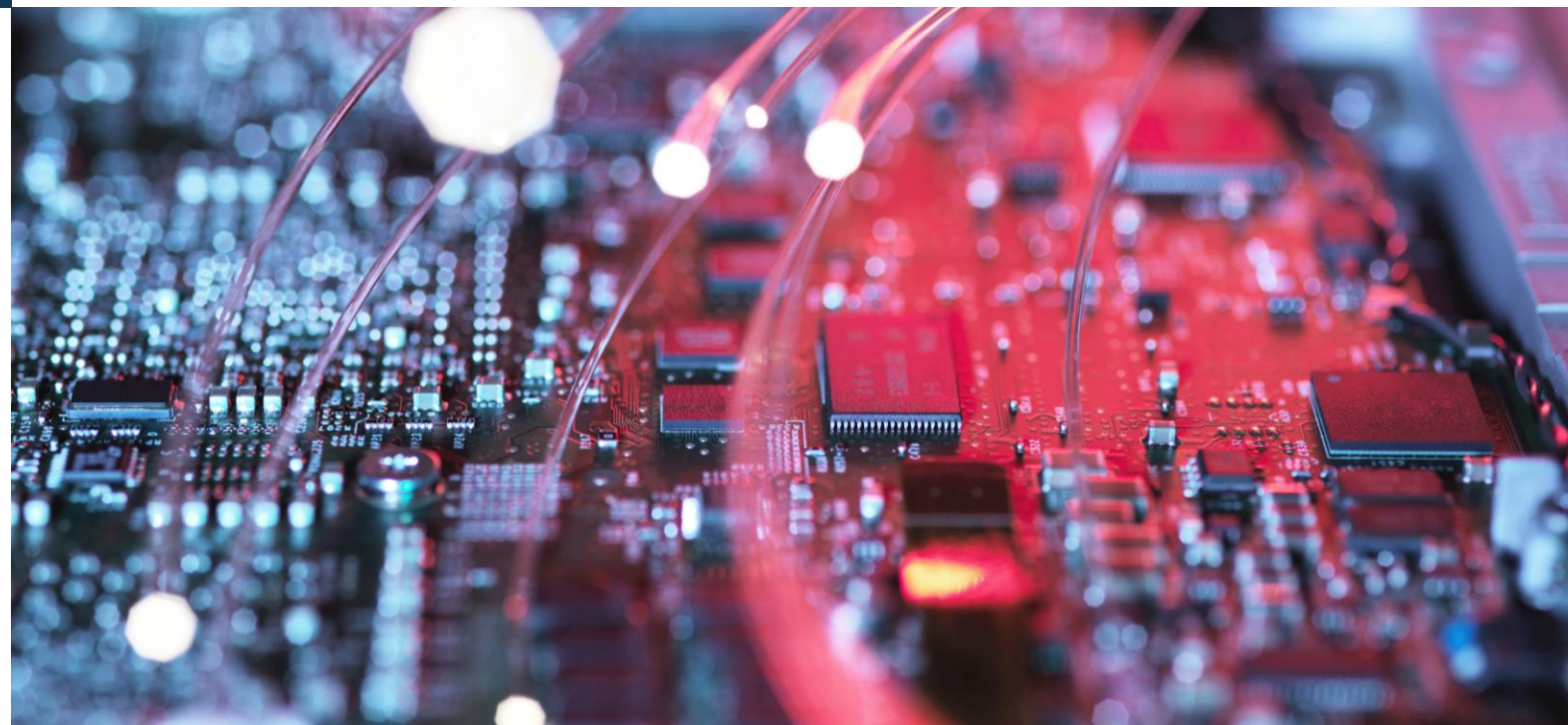
Procedures for regularly backing up the database, as well as the steps to restore the database in the event of data loss or system failure.

Security and Access Control:

Policies and procedures for managing user accounts, setting up appropriate permissions, and implementing security measures to protect the database from unauthorized access or tampering.

Performance Optimization:

Processes for monitoring database performance, identifying bottlenecks, and implementing optimization techniques, such as indexing, query tuning, and resource allocation.

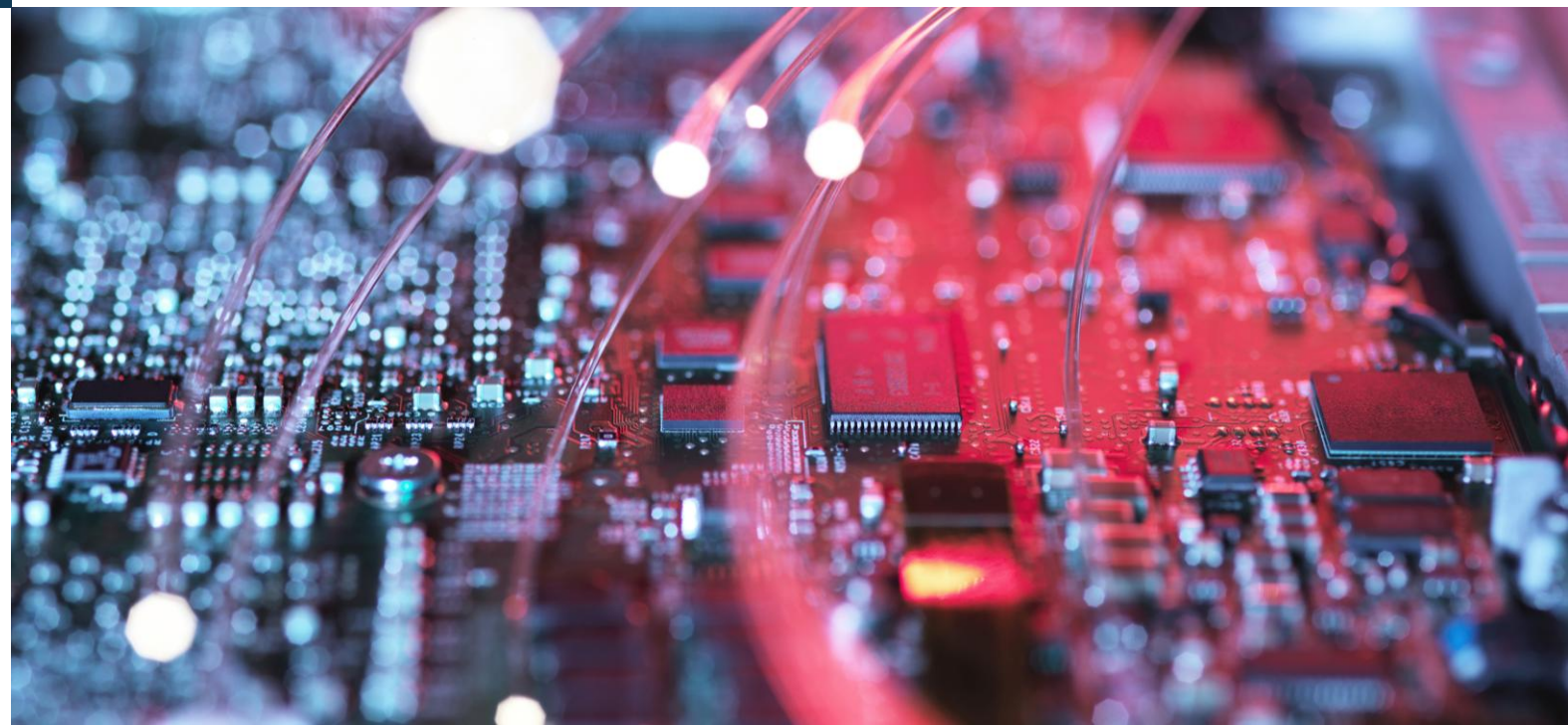


Database Access Language

SQL provides a standard set of commands and syntax for performing various database operations, such as:

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)
- Transaction Control Language (TCL)

The data stored in a database is typically organized into tables, which consist of rows (records or tuples) and columns (attributes or fields). These tables and their relationships form the core of the database's schema, defining the structure and rules governing the data. The efficient storage, indexing, and querying of this data is a primary responsibility of the database management system.



Commands

DDL: Commands like CREATE, ALTER, and DROP to define the structure of the database, including tables, views, and indexes.

DML: Commands like SELECT, INSERT, UPDATE, and DELETE to retrieve, add, modify, and remove data from the database.

DCL: Commands like GRANT and REVOKE to manage user permissions and access control in the database.

TCL: Commands like COMMIT and ROLLBACK to manage database transactions and ensure data integrity.

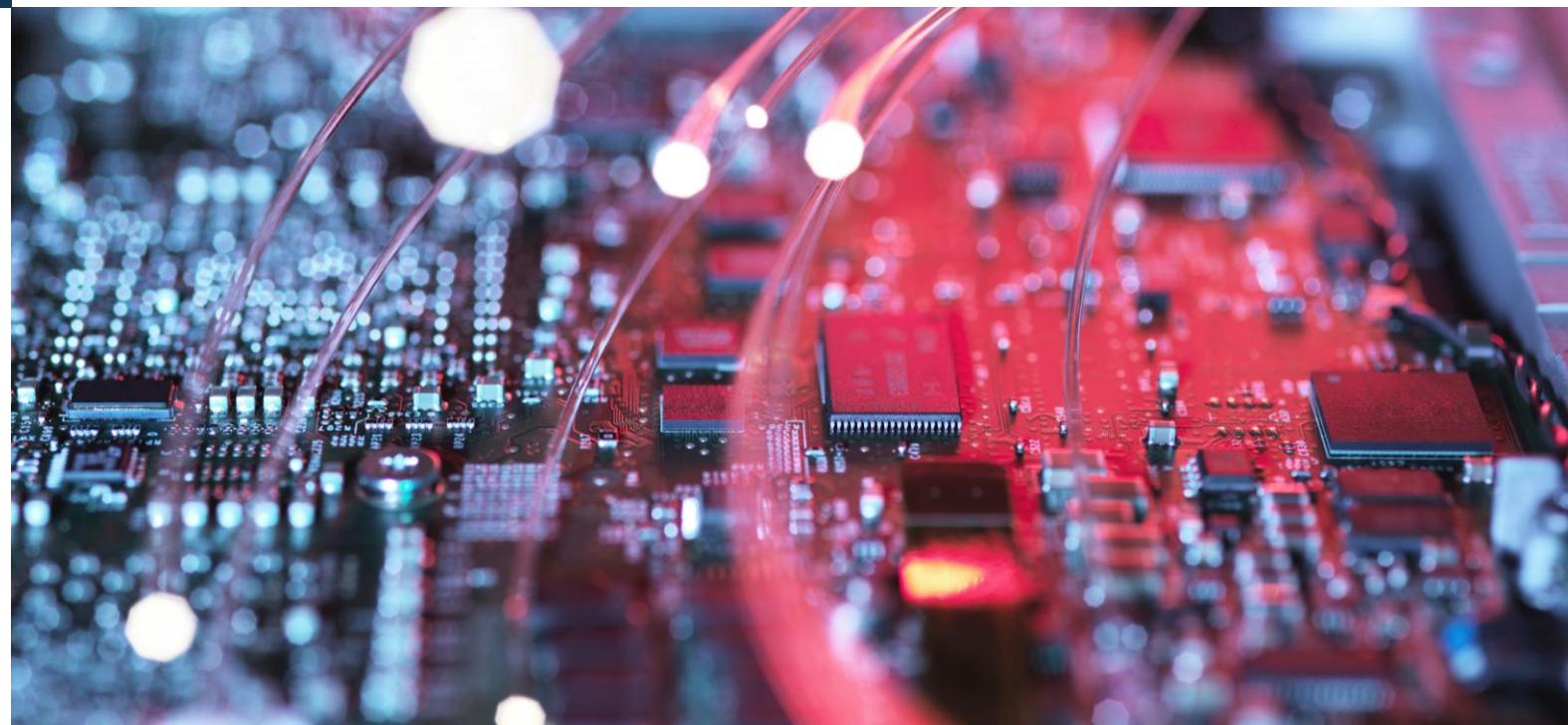
SQL is supported by all major relational database management systems (RDBMS), such as MySQL, Oracle, Microsoft SQL Server, PostgreSQL, and SQLite. While the specific syntax and implementation details may vary slightly between different RDBMS, the core SQL language remains consistent, allowing for portability and interoperability between database platforms.

Users

Users can be classified into several types:

- Database administrators (DBAs)
- Application programmers
- End-users

The database environment involves various user roles, each with distinct responsibilities and levels of interaction with the system. The most prominent user roles include database administrators, application developers, and end-users.



Roles and Responsibilities in the Database Environment

The database environment involves a diverse set of stakeholders, each with specific roles and responsibilities in ensuring the effective management and utilization of the database resources. The primary user roles include database administrators, application developers, and end-users.

Database Administrator (DBA)

- Responsible for the overall management, configuration, and maintenance of the database
- Ensures data integrity, security, performance, and availability

Application Developer

- Designs and implements applications that interact with the database
- Writes SQL queries and stored procedures to access and manipulate data

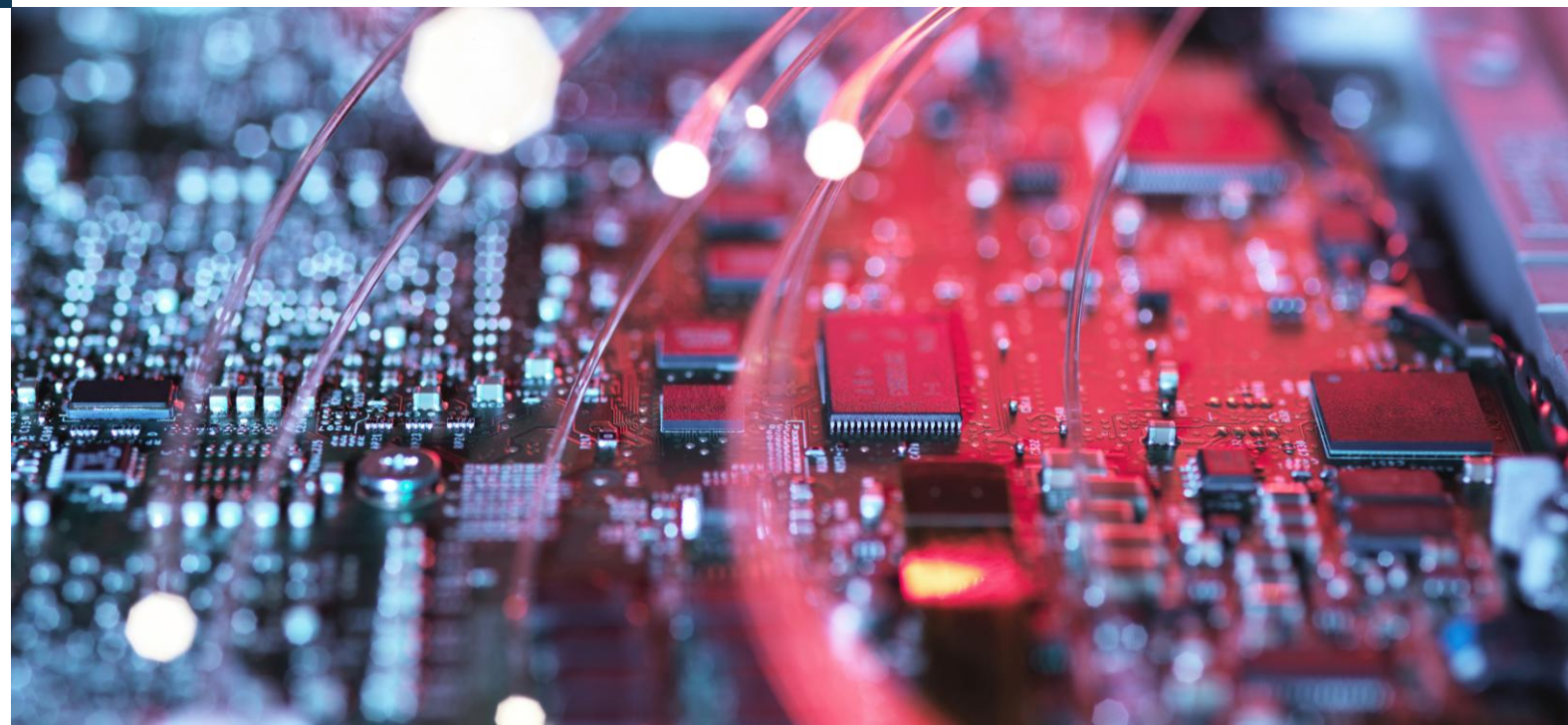
End-User

- Utilizes the applications and interfaces provided to access and retrieve data
- May have limited or specific permissions to interact with the database

Database

The database itself is the **collection of related data** organized to serve one or more purposes. It is stored on a disk and managed by the DBMS.

Together, these components create a structured environment that supports data storage, processing, retrieval, and management efficiently and securely.



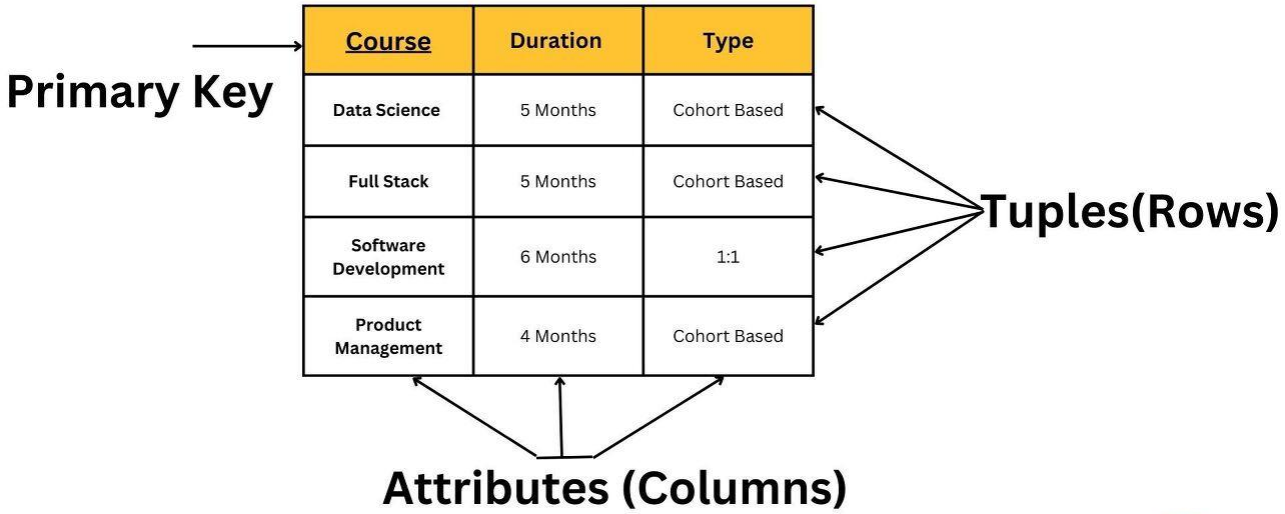
The Relational Model

The **relational model** is a way of organizing data into one or more tables, called **relations**, where each table consists of rows (**tuples**) and columns (**attributes**). It was introduced by E.F. Codd in 1970 and forms the foundation of most modern databases. Each table has a **primary key** that uniquely identifies each row, and relationships between tables are established using **foreign keys**. The relational model promotes data integrity, consistency, and eliminates redundancy through normalization.

The Relational Model

- Introduction to the Relational Model
- Key Concepts of the Relational Model
- Characteristics of the Relational Model
- Advantages of the Relational Model

Relational Model in DBMS

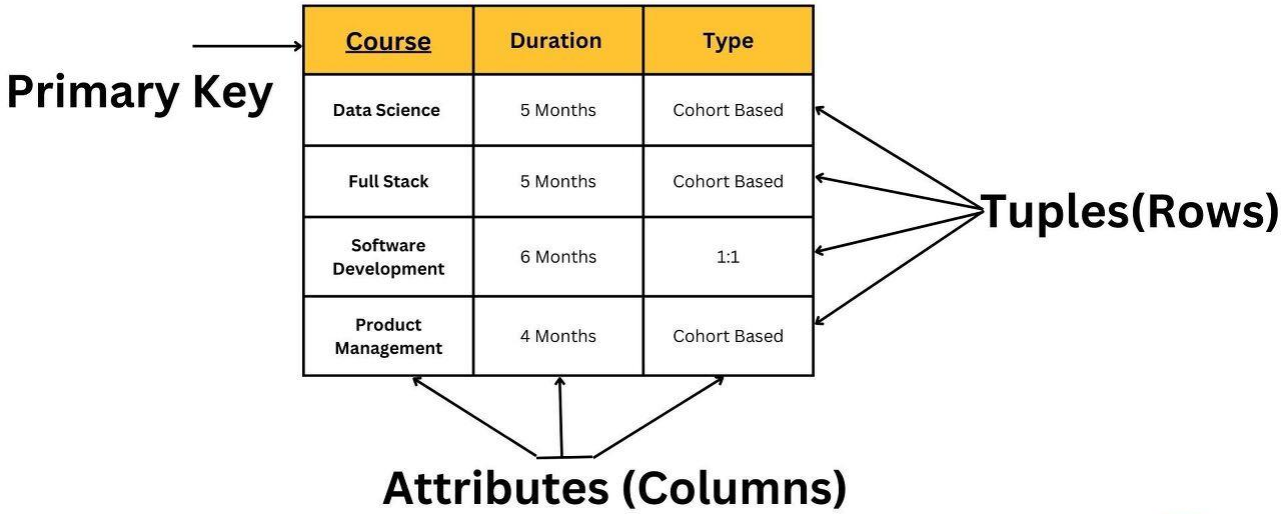


BOARD

Introduction

- An **introduction to the relational model** in database design begins with understanding how data is logically structured into **tables**, also known as **relations**. Each table consists of **rows (tuples)** and **columns (attributes)**, where each row represents a single record, and each column represents a data field.
- The relational model, introduced by **E.F. Codd in 1970**, uses **keys** to establish and enforce relationships between tables—**primary keys** uniquely identify records within a table, while **foreign keys** link related records across tables.
- This model promotes data integrity, consistency, and reduces redundancy by organizing data in a way that supports normalization. It forms the foundation of Structured Query Language (SQL), which is used to query and manage relational databases.
- The relational model is widely used in business, healthcare, education, and many other sectors due to its logical structure and ease of use in handling complex data relationships.

Relational Model in DBMS



BOARD

Key Concepts

Relation (Table)

- A **relation** is a table composed of rows and columns, where each table represents a specific entity such as customers, products, or orders.

Tuple (Row)

- Each **tuple** is a single record in the table, representing one instance of the entity (e.g., one customer or one product).

Attribute (Column)

- An **attribute** is a named column in a table that holds a specific type of data, such as a customer's name, ID, or email address.

Domain

- A **domain** is the set of valid values for a given attribute. For example, a "birthdate" attribute may have a domain limited to valid date values.

Primary Key

- The **primary key** uniquely identifies each tuple in a relation, ensuring no duplicate rows exist.

Foreign Key

- A **foreign key** links one table to another by referencing the primary key in a related table, establishing a relationship between the two.

Integrity Constraints

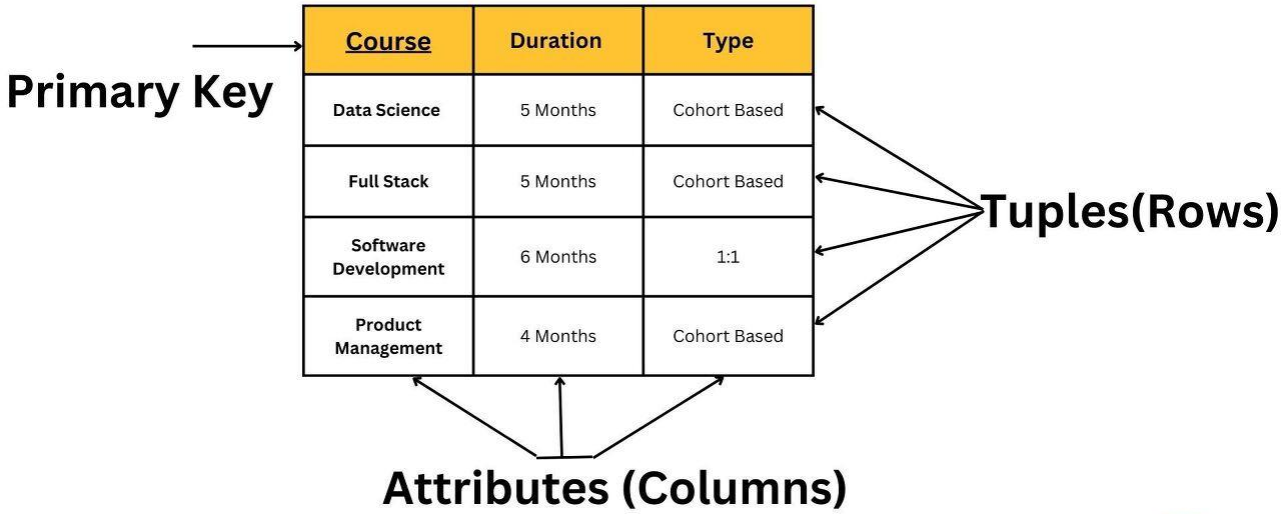
These are rules that ensure the accuracy and consistency of the data:

- **Entity integrity** requires that primary keys are unique and not null.
- **Referential integrity** ensures that foreign keys correctly reference existing primary keys.

Relational Schema

- The **schema** is the formal description of a database's structure, including its tables, attributes, and relationships.

Relational Model in DBMS



BOARD

Characteristics

Data is Stored in Tables

- All data is represented as relations (tables), consisting of rows (tuples) and columns (attributes). Each table stores data about one entity type.

Each Table Has a Primary Key

- Every relation includes a primary key, which uniquely identifies each row. This ensures that no two rows are exactly the same.

Use of Foreign Keys

- Foreign keys establish relationships between tables by referencing primary keys in other tables. This supports referential integrity.

Data Independence

- The relational model supports logical and physical data independence, meaning changes in storage structure or logical schema do not necessarily affect how users interact with the data.

Integrity Constraints

- The model enforces data integrity through:
  - Entity integrity (primary keys must be unique and not null)Referential integrity (foreign keys must match primary keys or be null)
  - Structured Query Language (SQL)The relational model uses SQL to define, query, update, and manage data in the database.

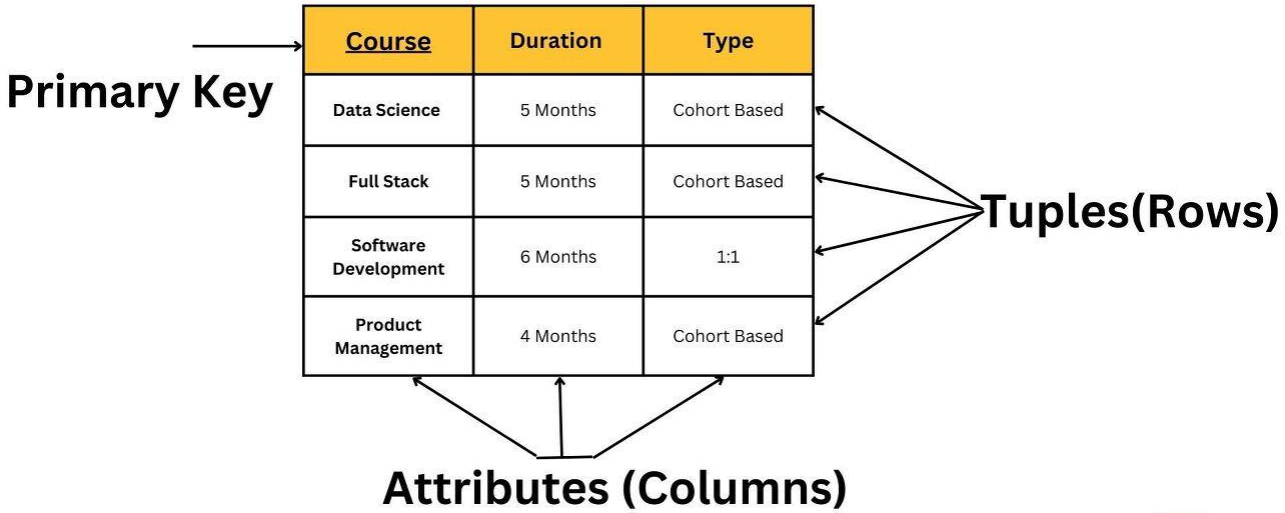
Normalization

- The model encourages normalization to minimize data redundancy and maintain consistency across the database.

Ordering is Not Important

- In a pure relational model, the order of rows and columns does not affect the data's meaning. Relations are treated as sets (unordered collections).

Relational Model in DBMS



BOARD

Advantages

- Simplicity
  - Data is organized into tables (relations), which are intuitive and easy to understand.
  - Users can query data without needing to understand complex data structures.
- Structural Independence
  - Changes to the database structure (such as adding a column) do not affect how applications access the data.
- Data Integrity and Accuracy
  - Relational constraints (e.g., primary keys, foreign keys, unique constraints) enforce data accuracy and consistency.
- Flexibility
  - Tables can be easily extended with new attributes or records.
  - SQL makes it easy to retrieve and manipulate data using powerful queries.
- Security
  - Fine-grained access control can be applied at the table, row, or column level.
- Data Redundancy Control
  - Through normalization, redundancy is minimized, which reduces storage requirements and the potential for update anomalies.
- Data Independence
  - Logical and physical data independence allows for changes in data storage or structure without impacting user interactions.
- Support for Multiple Users and Transactions
  - Concurrency control and transaction management support multi-user environments while maintaining data consistency.
- Strong Theoretical Foundation
  - Based on set theory and first-order predicate logic, which provides a solid mathematical basis for consistency and optimization.

Individual Project

Description

The current database that your company is using keeps track of all of the customers and suppliers as well as products offered for sale, but the current environment has all of these data spread out across multiple different databases, and it is difficult to see a complete picture of the environment. You have been tasked to consolidate the environment into a single database environment with the end goal of adding a data warehouse for the company.

Your manager is excited with the project description, is anxious to have a new database built for the company, and more excited that at the end of the project he will also have a data warehouse to help make strategic decisions. To start, your manager would like to understand the benefits of following a formal design methodology, especially with respect to database design. Describe how and why the company can benefit from spending time planning at the beginning of the project, instead of just jumping in and developing the applications to meet the perceived needs. To ensure that you give the best response, you choose to describe the 3-level ANSI-SPARC architecture and want to make sure you discuss how the use of this will promote data independence to save time in the long run.

You also want to take this opportunity to answer any potential questions about personnel needs with this new database environment. Describe the roles of a data administrator (DA) and a database administrator (DBA); describe the job functions of each and how the tasks they each perform differ. Would you recommend that your company have one person to perform both tasks, or should it hire two people?

Individual Project

Your paper should contain the following information:

- Describe the 3-level architecture.
- Describe data independence.
- Talk about the differences between a DA and a DBA.
- Discuss the pros and cons of having a separate DA and DBA or the need for 1 person doing it all.

Individual Project

You also should take this opportunity to create the template for your entire project, and create a Word document that you will add to for each remaining assignment. The document should follow this format:

Advanced Database Systems Project document shell:

- Use MS Word
- Title Page
    - Course number and name
    - Project name
    - Student name
    - Date
- Table of Contents
    - Use autogenerated TOC
    - Separate page
    - Maximum of 3 levels deep
    - Update fields of the TOC so it is up-to-date before submitting project
- Section headings (create each heading on a new page with TBD as the content, except for the sections listed under "New content" in each week's assignment)
    - Project Outline
    - The Database Models, Languages, and Architecture
    - Database System Development Life Cycle
    - Database Management Systems
    - Advanced SQL
    - Web and Data Warehousing and Mining in the Business World

Each week, you will add to this document and submit it for grading. As a preview, each section will contain the following:

Individual Project

Each week, you will add to this document and submit it for grading. As a preview, each section will contain the following:

•The Database Models, Languages, and Architecture (Week 1: IP)

- **A description of the 3-level ANSI architecture model**
- **A description of data independence**
- **The difference in responsibility between:**
  - **Data administrator**
  - **Database administrator**

•Database System Development Life Cycle (Week 2: IP)

- A completed enhanced entity-relationship diagram
- A description about the relationship setup and multiplicity

•Database Management Systems (Week 3: IP)

- The normalization of a given logical data model to Boyce-Codd Normal Form
- A logical data model for the Enhanced ERD from IP2

•Advanced SQL (Week 4: IP)

- A physical data model for your Enhanced ERD in a DBMS of your choice, including:
  - The DDL to create the tables
  - The DDL to create the primary and foreign keys
  - DML to manage data for the tables
  - 3 SELECT statements (one will be a JOIN)

•Web and Data Warehousing and Mining in the Business World (Week 5: IP)

- The design and DDL to create a star schema for a data warehouse for the database previously designed
- A description of the ETL process

Individual Project

Add the discussion about the 3-layer ANSI architecture, data independence, and the different administrators to the section titled "The Database Models, Languages, and Architecture."

Name the document CS352_<First and Last Name>_IP1.docx

Ensure your paper format meets all expectations of APA 7, including the title page, page numbers, reference citations, and being entirely double-spaced.

Resources

Relational Databases

A **relational database** is a type of database that stores data in structured tables with rows and columns, using relationships between tables to organize and connect data. This model is widely used due to its simplicity, flexibility, and ability to maintain data integrity through primary and foreign keys. Common relational database systems include **MySQL**, **PostgreSQL**, **Oracle**, and **Microsoft SQL Server**.

Here are some helpful online resources for students:

W3Schools SQL Tutorial:
https://www.w3schools.com/sql/

A beginner-friendly guide with interactive examples.

Khan Academy – Intro to SQL:
https://www.khanacademy.org/computing/computer-programming/sql

Great for understanding SQL basics through practice.

SQLZoo: https://sqlzoo.net/

Offers interactive SQL exercises for various topics.

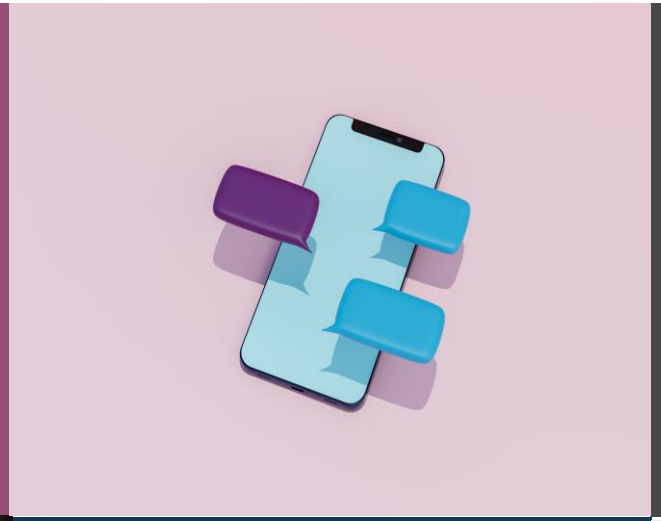
Mode SQL Tutorial: https://mode.com/sql-tutorial/

Real-world SQL training using sample data sets.

These resources are great starting points for learning how to interact with relational databases using SQL.

INSTRUTOR

DR. JOHN CONKLIN



PHONE

602.796.5972

EMAIL

jconklin@coloradotech.edu

WEBSITE

drjconklin.com