# Databases and SQL for Data  Science with Python

**Session 1**

30/06/2024

# Course Overview

- If there is a shortcut to becoming a Data Scientist, then learning to think and work like a In this course you will learn SQL inside out- from the very basics of Select statements to
- advanced concepts like JOINs.
- You will:
- write foundational SQL statements like: SELECT, INSERT, UPDATE, and DELETE
- filter result sets, use WHERE, COUNT, DISTINCT, and LIMIT clauses.
- differentiate between DML & DDL
- CREATE, ALTER, DROP and load tables.

# Benefits of Enrolling in a Course:

- Analyze data within a database using SQL and Python.
- Create a relational database and work with multiple tables using DDL commands.
- Construct basic to intermediate level SQL queries using DML commands.
- Compose more powerful queries with advanced SQL techniques like views, transactions, stored procedures, and joins.

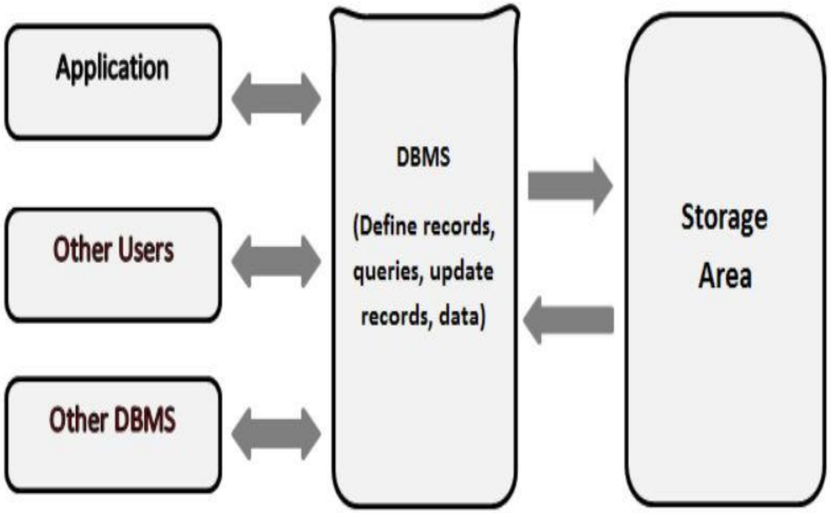
# Sessions Content

- What is a database?
- History and evolution of databases
- Types of databases
- Database Design
- Database Management Systems (DBMS)
- Database Life cycle
- Be introduced to SQL and its basic syntax.
- Write simple SQL queries using SELECT, FROM, and WHERE.

# What are Databases?

- A database is an organized collection of data, generally stored and accessed electronically from a computer system. It supports the storage and manipulation of data.

- In other words, databases are used by an organization as a method of storing, managing and retrieving information.



# What are Databases?

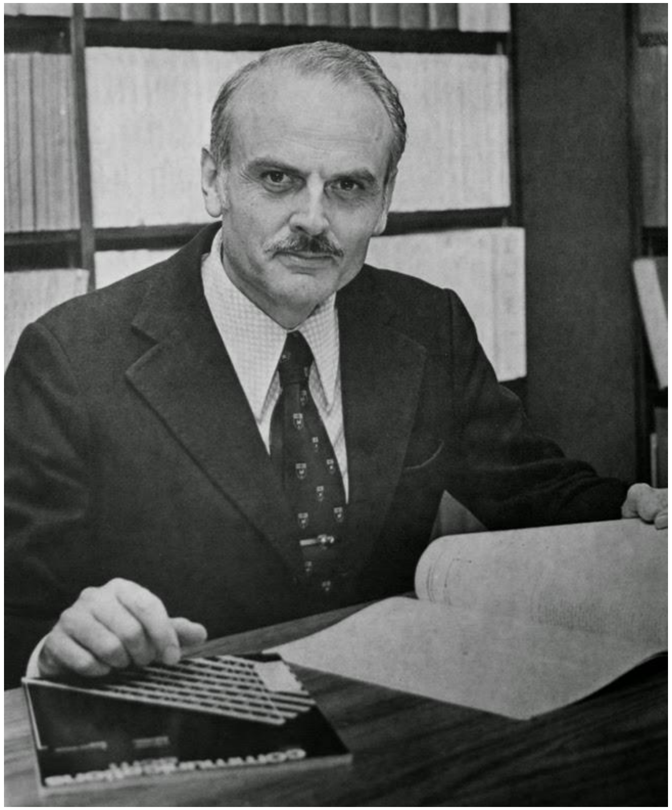
- Key Characteristics:
  - Persistence: Data is stored over time.
  - Organized: Structured in a way to facilitate easy access.
  - Accessible: Can be queried using specific languages (e.g., SQL).
  - Scalable: Designed to grow with increasing data.

# History and evolution of databases

- 1950s–1960s: File-Based Systems
  - Data stored in flat files.
  - Manual processes were required to retrieve and manage data.
  - Lacked a structured way to organize or relate data.
  - Redundancy: Repeated data across files.
  - Inconsistency: Updates in one file might not reflect in others.
  - Scalability: Difficult to manage as data size increased.

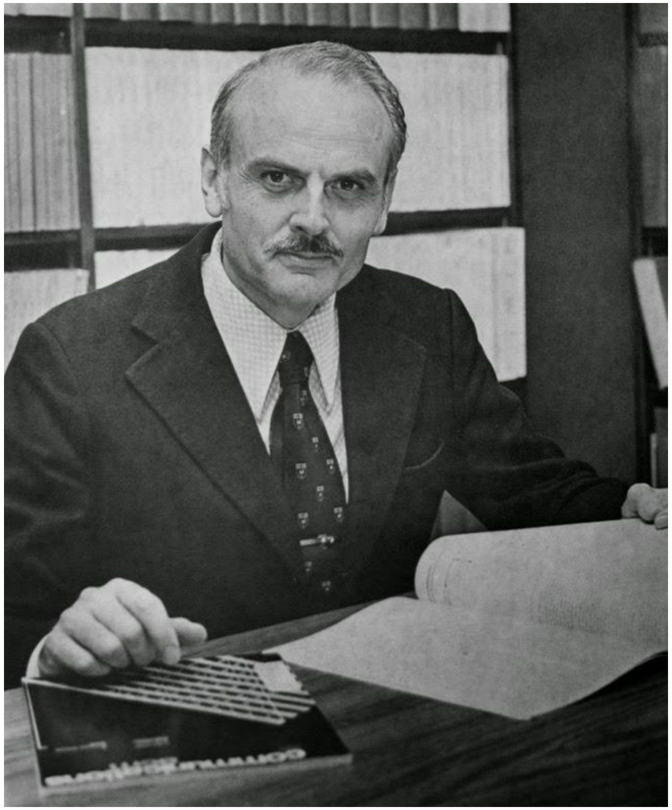
# History and evolution of databases

- 1970s: Birth of Relational Databases
  - In 1970, Edgar F. Codd, a computer scientist at IBM, proposed the relational model in his paper, "A Relational Model of Data for Large Shared Data Banks."



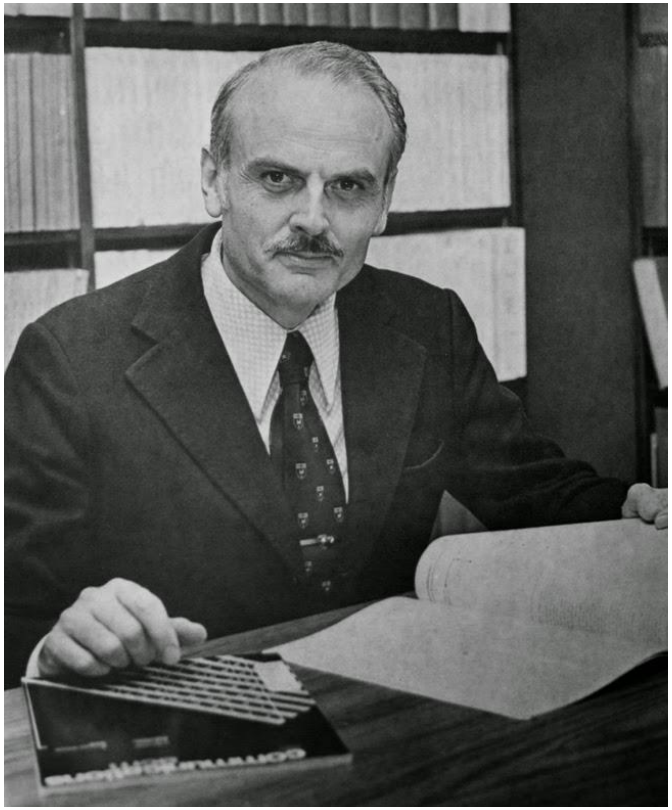
# History and evolution of databases

- 1970s: Birth of Relational Databases
  - Organized data into tables (relations) with rows and columns.
  - Tables were connected through keys (Primary and Foreign).



# History and evolution of databases

- 1970s: Birth of Relational Databases
  - Reduced redundancy using normalization.
  - Provided data consistency and integrity.
  - Enabled powerful querying with SQL.



# History and evolution of databases

- 1980s: Emergence of Popular Systems
  - Companies like Oracle, IBM, and Microsoft began releasing commercial RDBMS

  - Products:
    - Oracle Database (1979)
    - IBM DB2 (1983)
    - Microsoft SQL Server (1989).

# History and evolution of databases

- 1980s: Emergence of Popular Systems
  - Relational databases became the backbone of business applications like inventory management, accounting, and customer records.

# History and evolution of databases

- 2000s: The NoSQL Movement
  - Why NoSQL?
  - The rise of the internet and large-scale web applications demanded:
    - Scalability for handling millions of users.
    - Flexibility to store semi-structured and unstructured data.

# History and evolution of databases

- 2000s: The NoSQL Movement
  - Types of NoSQL Databases:
    - Document: MongoDB, Couchbase.
    - Key-Value: Redis, DynamoDB.
    - Wide-Column: Cassandra, HBase.
    - Graph: Neo4j, Amazon Neptune.

# History and evolution of databases

- 2010s–Present: Cloud Databases and Big Data
  - Cloud Databases:
    - Hosted on cloud platforms like AWS, Google Cloud, and Azure.
  - Benefits:
    - On-demand scalability.
    - High availability and fault tolerance.
    - Pay-as-you-go pricing models.

# History and evolution of databases

- 2010s–Present: Cloud Databases and Big Data
  - Big Data Era:
    - Tools like Hadoop and Spark revolutionized data processing for massive datasets.
    - Data lakes emerged for handling diverse data types.

# Advantages of using Databases

- There are many advantages of databases
- Reduced data redundancy
- Reduced updating errors and increased consistency
- Greater data integrity and independence from application programs
- Improved data access to users through the use of host and query languages
- Improved data security
- Reduced data entry, storage, and retrieval costs

# Disadvantages of using Databases

- There are many disadvantages of databases
- Although databases allow businesses to store and access data efficiently, they also have certain disadvantages
- Complexity
- Cost
- Security
- Compatibility

# Relational Databases

- Definition:
  - Store data in tables (rows and columns) with predefined schemas.
  - Tables are related through primary and foreign keys.

# Relational Databases

**STORE**

| Store_key | City | Region |
|---|---|---|
| 1 | New York | **East** |
| 2 | Chicago | Central |
| 3 | Atlanta | **East** |
| 4 | Los Angeles | **West** |
| 5 | San Francisco | **West** |
| 6 | Philadelphia | **East** |
| . | . | . |
| . | . | . |

**PRODUCT**

| Product_key | Description | Brand |
|---|---|---|
| 1 | Beautiful Girls | MKF Studios |
| 2 | Toy Story | **Wolf** |
| 3 | Sense and Sensibility | Parabuster Inc. |
| 4 | Holiday of the Year | **Wolf** |
| 5 | Pulp Fiction | MKF Studios |
| 6 | The Juror | MKF Studios |
| 7 | From Dusk Till Dawn | Parabuster Inc. |
| 8 | Hellraiser: Bloodline | Big Studios |
| . | . | . |
| . | . | . |

**SALES_FACT**

| Store_key | Product_key | Sales | **Cost** | Profit |
|---|---|---|---|---|
| 1 | 6 | 2.39 | 1.15 | 1.24 |
| 1 | 2 | 16.7 | 6.91 | 9.79 |
| 2 | 7 | 7.16 | 2.75 | 4.40 |
| 3 | 2 | 4.77 | 1.84 | 2.93 |
| 5 | 3 | 11.93 | 4.59 | 7.34 |
| 5 | 1 | 14.31 | 5.51 | 8.80 |
| . | . | . | . | . |
| . | . | . | . | . |

# Relational Databases

- Advantages:
  - Easy to organize and retrieve structured data.
  - Ensures consistency across related data..

# NoSQL Databases

- Store data as JSON-like documents.
- Examples:
  - MongoDB, DynamoDB

# Database Design

- Definition:
  - Database design is the process of organizing data into a structured format to meet the needs of an application or organization.

# Database Design

What is an Entity-Relationship Diagrams (ERD)?

- A visual representation of entities, attributes, and relationships in a database.
- Helps in designing structured, efficient databases before implementation.

# Database Design

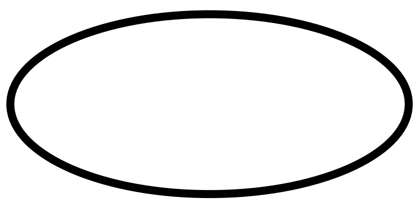
- **Entities** – Objects or concepts (e.g., Student, Course, Employee).
- **Attributes** – Properties of entities (e.g., Student_ID, Name, Age).
- **Relationships** – How entities interact (e.g., Student enrolls in Course).
- **Primary Key** – Unique identifier for an entity.
- **Foreign Key** – Ensures referential integrity between tables.

# Database Design

- Entity:
  - person, place, object, event, concept (often corresponds to a real time object that is distinguishable from any other object)

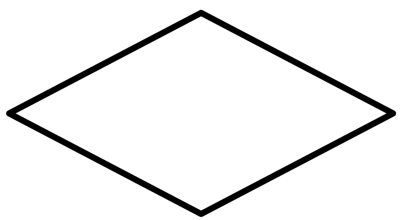
# Database Design

- Attribute:
  - property or characteristic of an entity type (often corresponds to a field in a table)

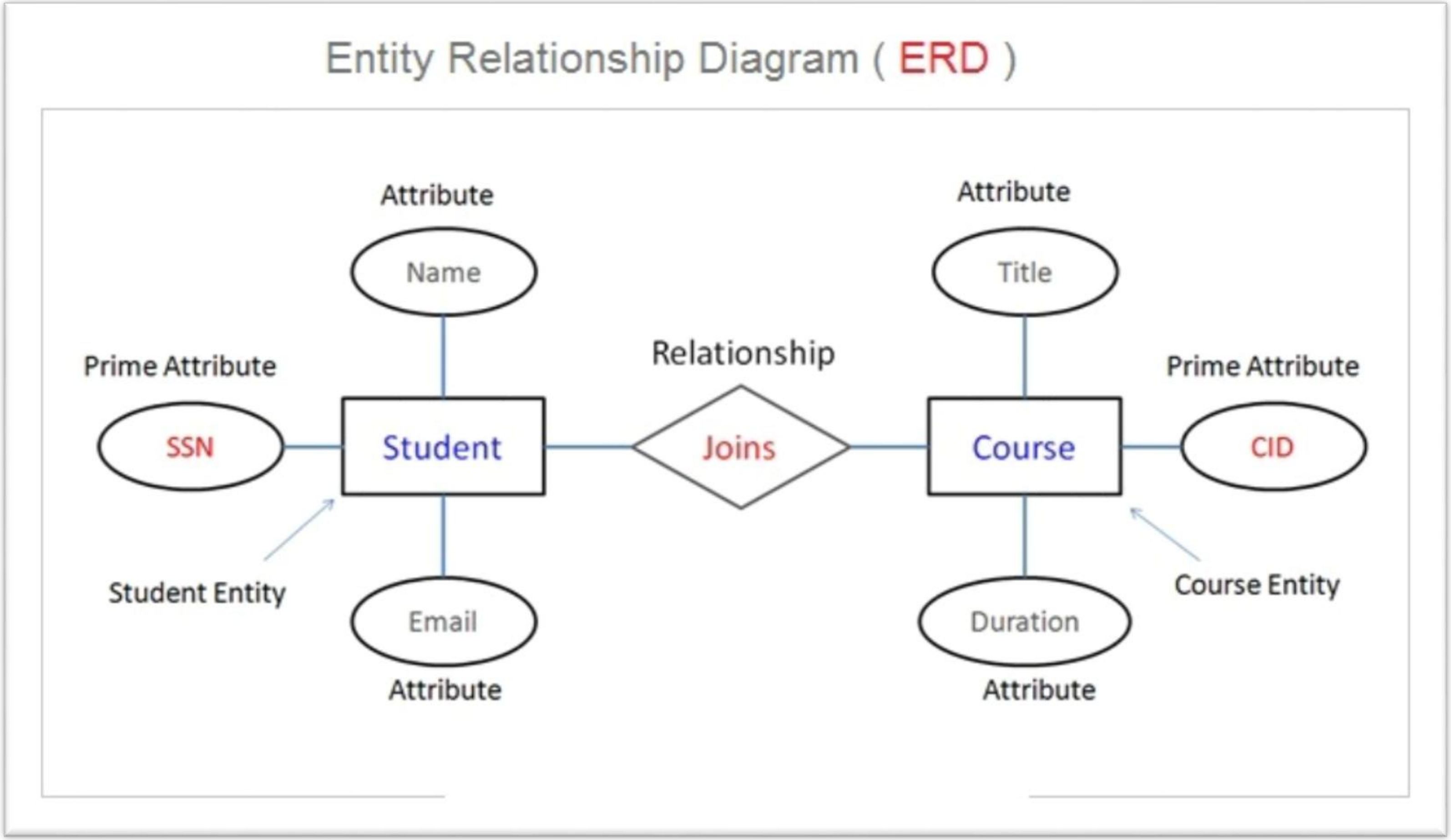


# Database Design

- Relationship:
  - link between entities (corresponds to primary key-foreign key equivalencies in related tables).



# Database Design

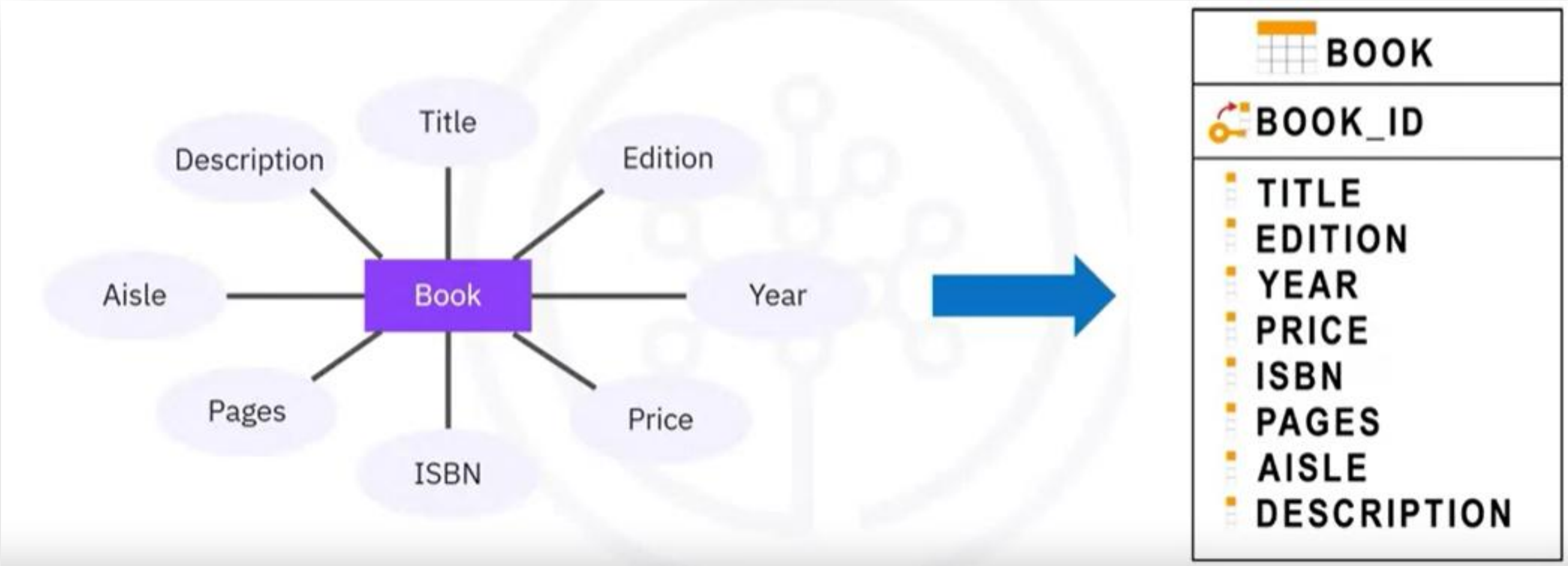


# Database Constraints

- Primary Key  ( Not Null + Unique)
- Not Null
- Unique Key
- Referential Integrity ( FK )
- Check

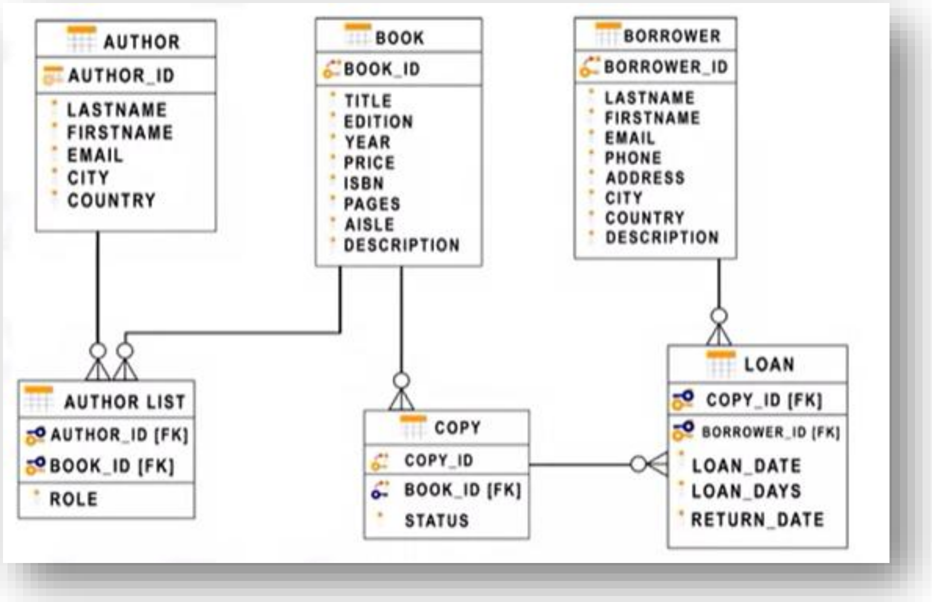
# Entity-Relationship Model

- Used as a tool to design relational databases



# Relational Model

- Most used Data Model
- Allows for data independence
- Data is stored in tables



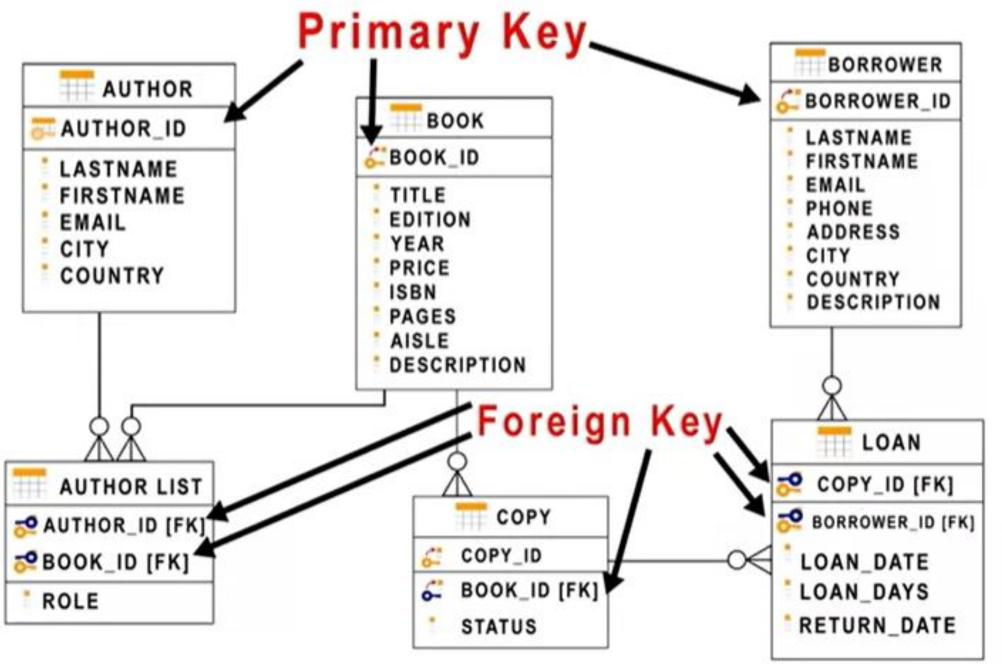
# Mapping Entity Diagrams To Tables

- Entities become tables
- Attributes get translated into columns

Table: Book

| Title | Edition | Year | Price | ISBN | Pages | Aisle | Description |
|---|---|---|---|---|---|---|---|
| Database Fundamentals | 1 | 2010 | 24.99 | 978-0-98662 83-1-1 | 300 | DB-A02 | Teaches you the fundamentals of databases |
| Getting started with DB2 Express-C | 1 | 2010 | 24.99 | 978-0-9866 283-5-1 | 280 | DB-A01 | Teaches you the essentials of DB2 using DB2 Express-C, the free version of DB2 |

# Primary keys and Foreign keys



# Database Transaction

- A transaction is an executing program that forms a logical unit of database actions.
- It includes one or more database access operations such as  insert, delete and update.
- The database operations that form a transaction can either be embedded within an application program or they can be specified interactively via a high-level query language such as SQL.

# Database Transaction Properties

Transactions should possess several properties, often called the ACID properties:

1. Atomicity
2. Consistency
3. Isolation
4. Durability

# Database Schema

- A schema is a group of related objects in a database.
- There is one owner of a schema who has access to manipulate the structure of any object in the schema.
- A schema does not represent a person, although the schema is associated with a user that resides in the database.

# Data types

- A data type determines the type of data that can be stored in a database column.

- The most commonly used data types are:
  1. Alphanumeric: data types used to store characters, numbers, special characters, or nearly any combination.
  2. Numeric
  3. Date and Time

# Database Management Systems (DBMS)

- Oracle Database – Oracle Corporation – 1979
- Microsoft SQL Server – Microsoft – 1989
- IBM Db2 – IBM – 1983
- MySQL – Oracle Corporation (originally developed by MySQL AB) – 1995
- PostgreSQL – PostgreSQL Global Development Group – 1996

# Database Management Systems (DBMS)

- SQLite – SQLite Consortium (originally developed by D. Richard Hipp) – 2000
- MariaDB – MariaDB Corporation – 2009
- (Forked from MySQL)Amazon Aurora – Amazon Web Services (AWS) – 2014

# Row-based vs. Column-based Databases

## Row-based Databases (Traditional Relational Databases)

- Store data **row-by-row** (each row represents a complete record).
- Optimized for **transactional workloads (OLTP)**.
- Common in **relational databases (RDBMS)**.
- Examples: **MySQL, PostgreSQL, SQL Server, Oracle DB**.
- **Use Case:** Best for applications that require **frequent inserts, updates, and deletes**, such as banking or e-commerce.

# Row-based vs. Column-based Databases

## Column-based Databases (Analytical Databases)

- Store data **column-by-column** instead of rows.
- Optimized for **analytical processing (OLAP)**.
- Improves query performance for **aggregations and analytics**.
- Examples: **Apache Cassandra, Amazon Redshift, Google BigQuery**.
- **Use Case:** Best for **big data analytics, reporting, and data warehousing**, where queries scan large datasets.

Questions?