# Course 2 - Introduction to Relational Databases (RDBMS)

## *Week 1*

## *Relational Database Concepts*

## *Module 1: - Fundamental Relational Database Concepts*

### *1] Course Introduction*

Every good relational database solution begins with a solid design and implementation strategy. A well-designed relational database ensures that the users and applications that depend on the data will know that it is:

- **Accurate.** Can you rely on the accuracy of the data as new information is added or it is modified?
- **Easy to access.** Is the data organized in a way that makes it fast, easy, and predictable to query and maintain?
- **Reliable.** Can your database design ensure data integrity and maintain consistent and reliable data?
- **Flexible.** Can you update or expand on the design to meet future data requirements?

### *2] Review of Data Fundamentals*

**What is data?**

- Data is unorganized information, that is processed to make it meaningful.
- It can consist of Facts, observations, perceptions, Numbers, characters, symbols, Images Or a mix of any of these.
- Data can be categorized by the level and rigidity of its structure. Data can be structured, semi-structured or unstructured.
- **Structured Data -** Structured data can be represented in rows and columns, just like a table. It has a well-defined schema, and a rigid structure. These characteristics make relational databases, which store data in tables, ideal for structured data.
- **Semi-Structured Data -** Semi-structured data has some organizational properties, but not enough to be easily stored in the rows and columns required by a rigid, tabular schema. Instead, semi-structured data is organized into a hierarchy using tags and metadata.
- **Unstructured Data -** Unstructured data doesn't have an identifiable structure; it doesn't follow any specific format, sequence, semantics, or rules. It cannot be

organized into tabular format for storage in a relational database. Unstructured data is often stored in NoSQL databases.

**Data Sources:-**

1. Relational Databases
2. Flat files and XML Datasets
3. APIs and Web Service
4. Web Scraping
5. Data Streams and Feeds
6. Social Platforms
7. Sensor Devices

- All of this data can be stored, processed, and made available for analysis, providing businesses with insights into their performance.

**File Formats:-**

1. Delimited text files
    - CSV, TSV
    - Store data in rows, items separated by a delimiter
2. Spreadsheets
    - XLSX
    - Store data in rows and columns
3. Language
    - XML, JSON
    - Transfer data
    - Platform independent

**Data Repositories:-**

**Where should data be stored?**

- Structured and semi-structured data is often stored in databases; either relational databases like DB2 or non-relational databases like MongoDB.
- Each type of database is optimized for different types of operations.
- The type of data you need and the processes you want to apply to it will determine the type storage you choose.

**OLTP (Online Transaction Processing) systems:-**

- Designed to store high volume day-to-day operational data
- Typically, relational but can also be non-relational

**OLAP (Online Analytical Processing) systems:-**

- Optimized for conducting complex data analytics.
- Include relational and non-relational databases, data warehouses, data lakes and big data stores.

**Relational Databases:-**

- Consist of structured data stored in related tables.
- The links between the tables are defined in a way that minimizes the duplication of data while still maintaining all the complex relationships required.
- Relational databases and their supporting systems are called Relational Database Management Systems, or RDBMS.
- Examples - IBM DB2, Microsoft SQL Server, Oracle, My SQL.
- Relational databases are primarily OLTP systems, used to support day-to-day business activities such as customer transactions, human resource activities, and workflows.
- They can also be used to perform data analysis, for example, data from a customer relationship management system can be used to make sales projections.

**Summary: -**

- Data is information like facts, observations, perceptions, numbers, characters, and images that can be processed to become meaningful.
- Data can be structured, semi-structured, or unstructured. Different data sources offer different types of data, for example data from social media can be unstructured or semi-structured.
- Data can be stored in repositories like relational databases and non-relational databases, amongst many others.
- Data can be transferred in CSV, XML, and JSON files.

## *3] Information and Data Models*

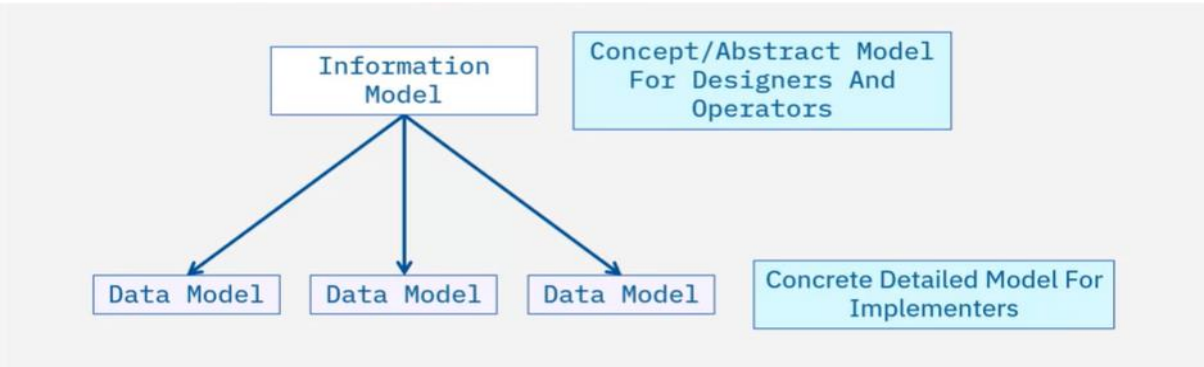


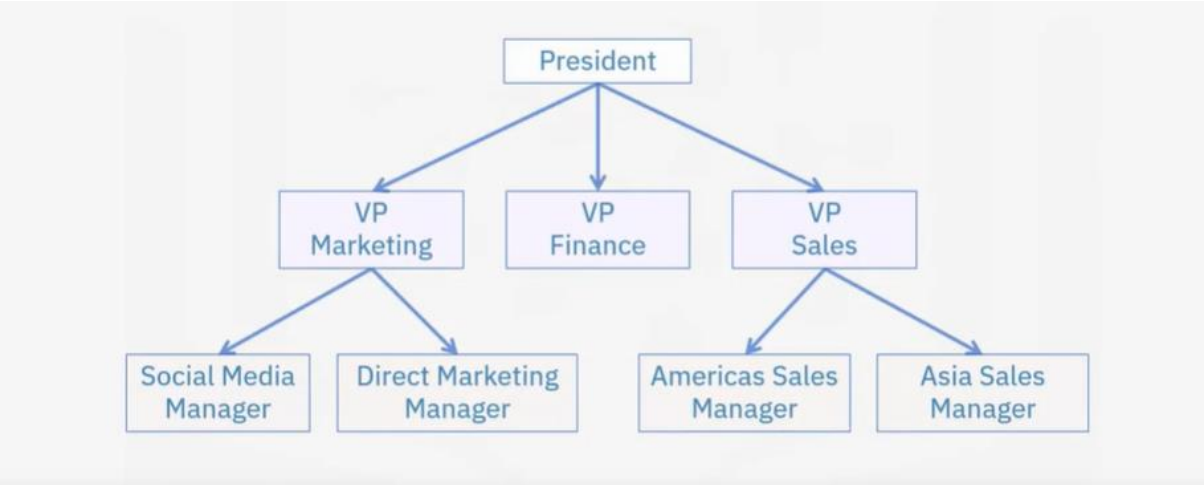

***Fig. Relationship between Information Model and Data Model***

- An Information Model is an abstract, formal representation of entities that includes their properties, relationships and the operations that can be performed on them.

- The entities being modelled can be from the real world, such as a library. Information Models and Data Models are different and serve different purposes.
- An Information Model is at the conceptual level and defines relationships between objects.
- Data Models are defined at a more concrete level, are specific and include details. A data model is the blueprint of any database system.

**Types of Information Model:-**

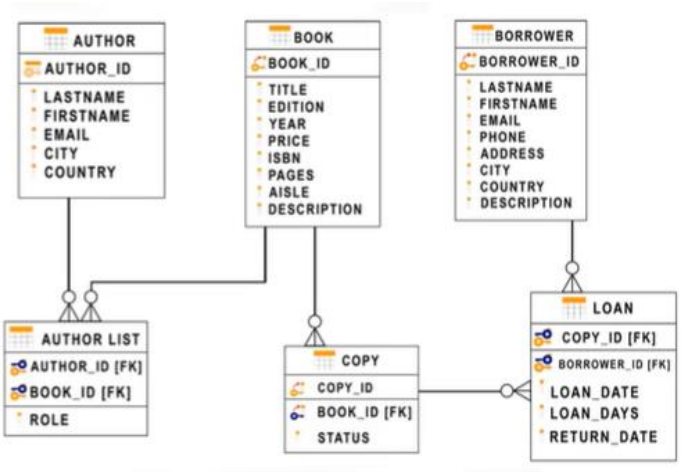
1. **Hierarchical Information Model:-**




- Organizes its data using a tree structure.
- The root of the tree is the parent node followed by child nodes.
- A child node cannot have more than one parent; however, a parent can have many child nodes.
- The first hierarchical database management system was the Information Management System released by IBM in 1968 and was originally built as the database for the Apollo space program.

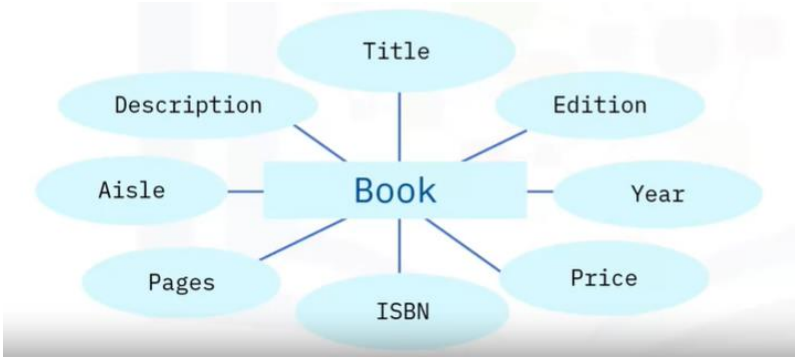
2. **Relational Model:-**

- Most used data model
- Allows for data independence
- Data is stored in tables
- Provides –
  - Logical data independence
  - Physical data independence
  - Physical storage independence

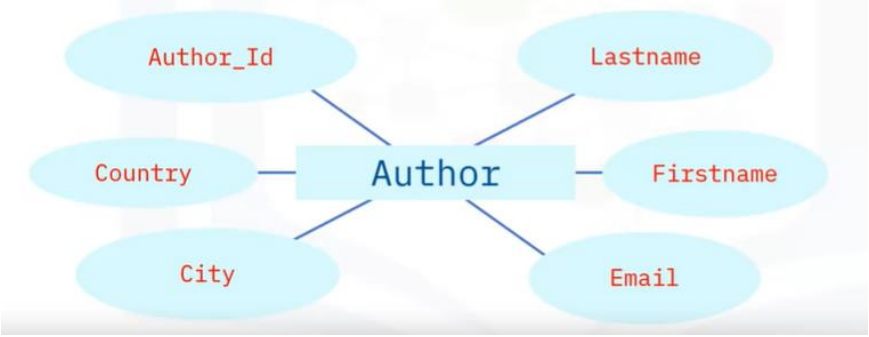


## 3. Entity-Relationship Model:-

- Collection of entities.
- Used as a tool to design relational databases.
- Entities are objects that exist independently of any other entities in the database.
- It is simple to convert an ER Diagram into a collection of tables.
- The building blocks of an ER Diagram are entities and attributes.
- Entities have attributes, which are the data elements that characterize the entity.
- Attributes tell us more about the entity. Attributes are certain properties or characteristics of an entity and tell us more about the entity.
- Example – An entity is drawn as a rectangle, and attributes are drawn as ovals.

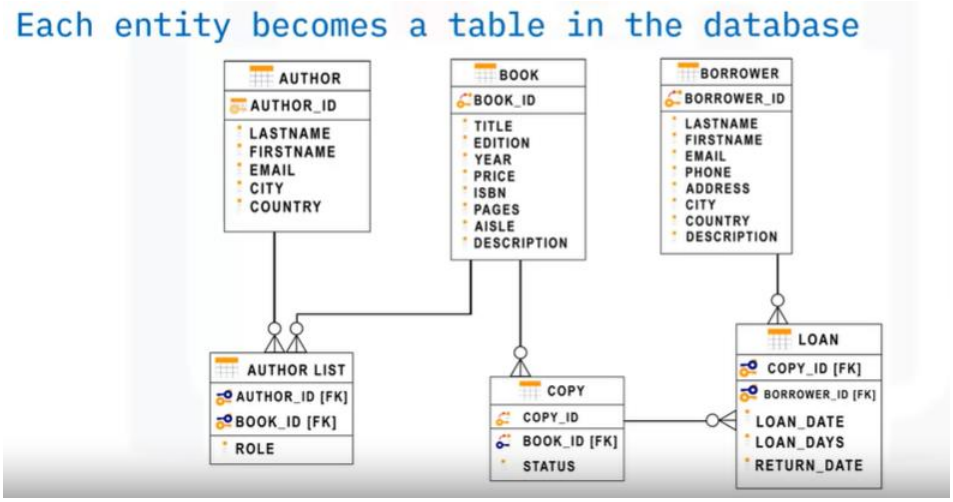



- A book is an example of an entity. The entity Book has attributes such as book title, the edition of the book, the year the book was written, etc. Attributes are connected to exactly one entity. The entity Book becomes a table in the database, and the attributes become the columns in a table.




- Books are written by authors. Book is an entity, and Author is an entity. For the entity Author, the ER Diagram would look like this. The entity Author has attributes such as the author's last name, first name, email, city, country and an author ID (to uniquely identify the author). The entity Author becomes a table in the database, and the attributes become the columns in the table.

***Simplified Library ER Diagram:-***



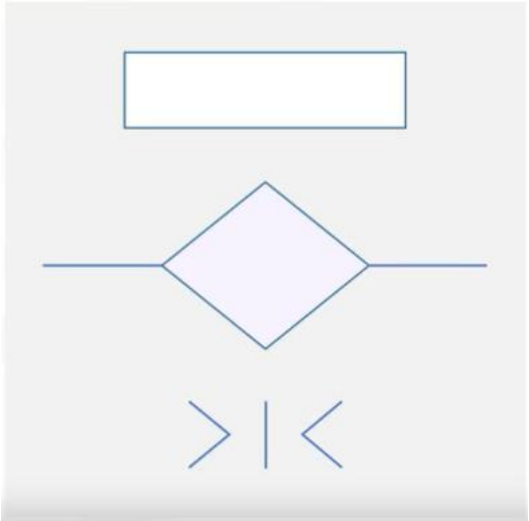

**Summary:-**

- Information Models are abstract, formal representations of entities that include their properties, relationships and the operations that can be performed on them.
- Data Models are defined at a more concrete level, are specific and include details.
- The Relational Model is the most used data model for databases because this model allows for logical data independence, physical data independence, and physical storage independence.
- Entities are objects that exist independently of any other entities in the database. Entities can be a noun (person, place, or thing), such as a book or an author.
- Attributes are the data elements that characterize the entity.
- For example, the book entity will have attributes like author and title.

## *4] ERDs & Types of Relationship*

***Building blocks:-***

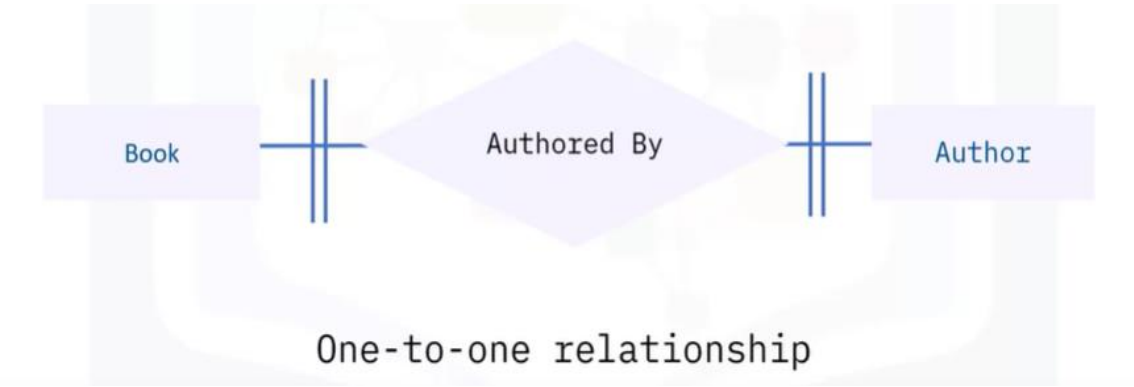
- Building blocks of a relationships are:
  1. Entities
  2. Relationship sets
  3. Crows foot notations
- Entity sets are represented by a rectangle. Relationship sets are represented by a diamond, with lines connecting associated entities. Different techniques are used in representing relationships. For ease of understanding, this video uses the crows



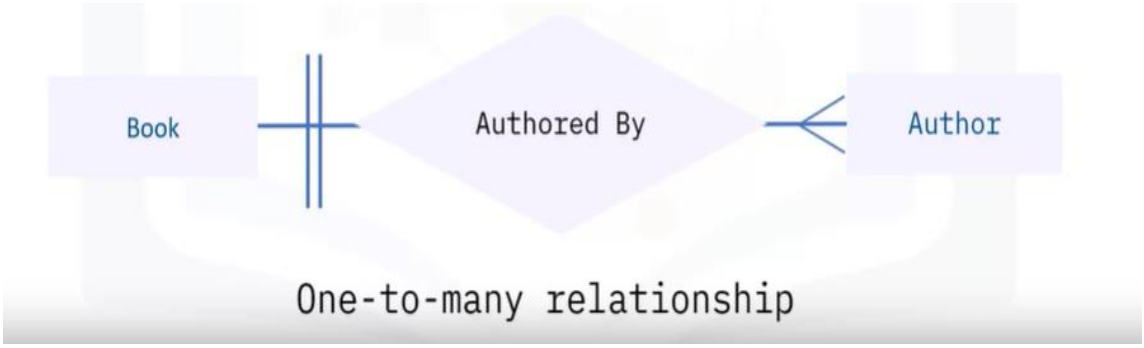
foot notations. Some of these are the greater-than symbol, the less-than symbol and a vertical line.

***Defining the relationships between entities:-***

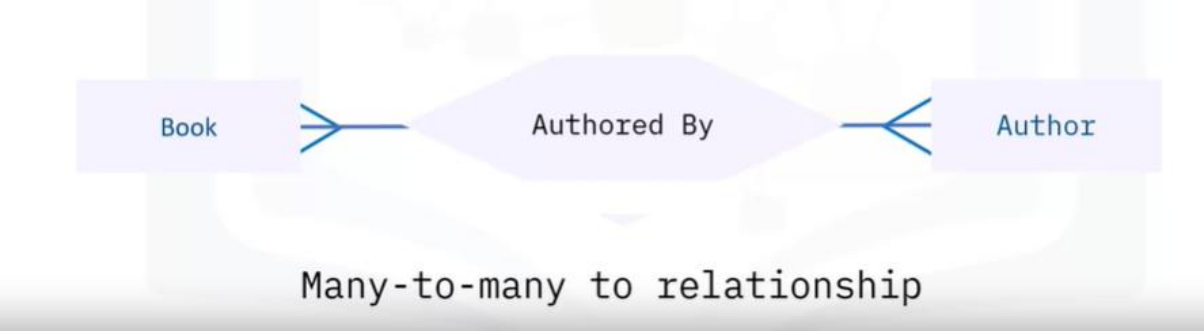
1. **One-to-One Relationship:-**
   - One book needs to be written by at least 1 author
   - One author can write many books
   - One book can be written by many authors.




2. **One-to-Many Relationship:-**
   - A book written by many authors
   - Crows foot notation is the less-than symbol
   - Many-to-one relationship for many authors writing a single book



3. **Many-to-Many Relationship:-**
    - Crows foot notation uses greater-than and less-than symbols.
    - Many books written by many authors.
    - Many authors writing many books.




**Summary:-**

- The building blocks of a relationship are entities, relationship sets, and crows foot notations.
- In a one-to-one relationship, one entity is associated with one and only one instance of another entity. For example, when one book has only one author.
- In a one-to-many relationship, one entity is associated with one or more instances of another entity. For example, when one book has many authors.
- In a many to many relationships, many instances of an entity are associated with many instances of another entity. For example, when many authors write many different books.

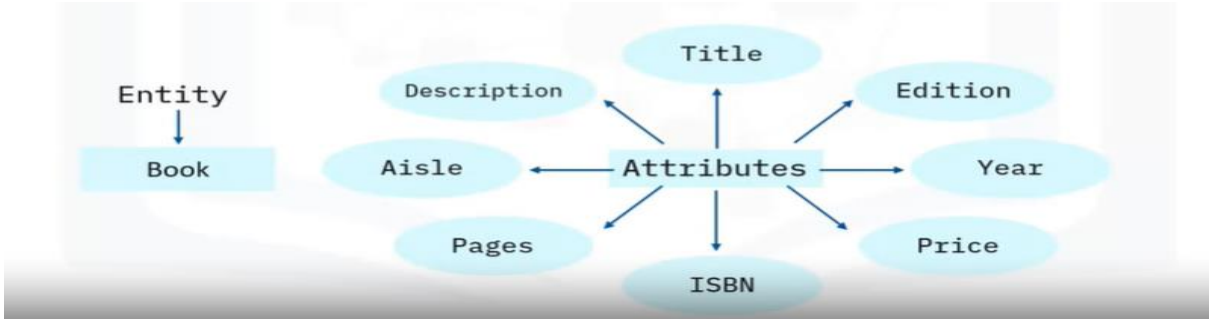
## *5] Mapping Entities to Tables*

**E-R Diagrams:-**

- Foundation for designing a database
- Begin with ERD, and map the ERD to the table.

**Example: Mapping an ERD to a table**

- Example: Entity Book
- Entity book has several attributes.
- Separate the entity from the attributes.




**Example: Entity Book**

- Book = Table
- Attributes = Columns

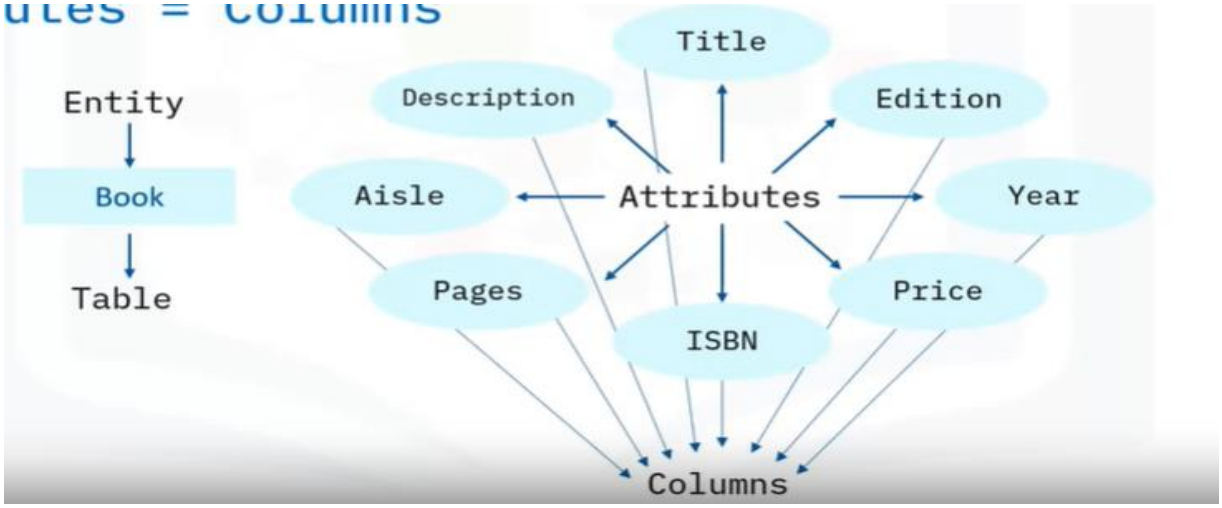


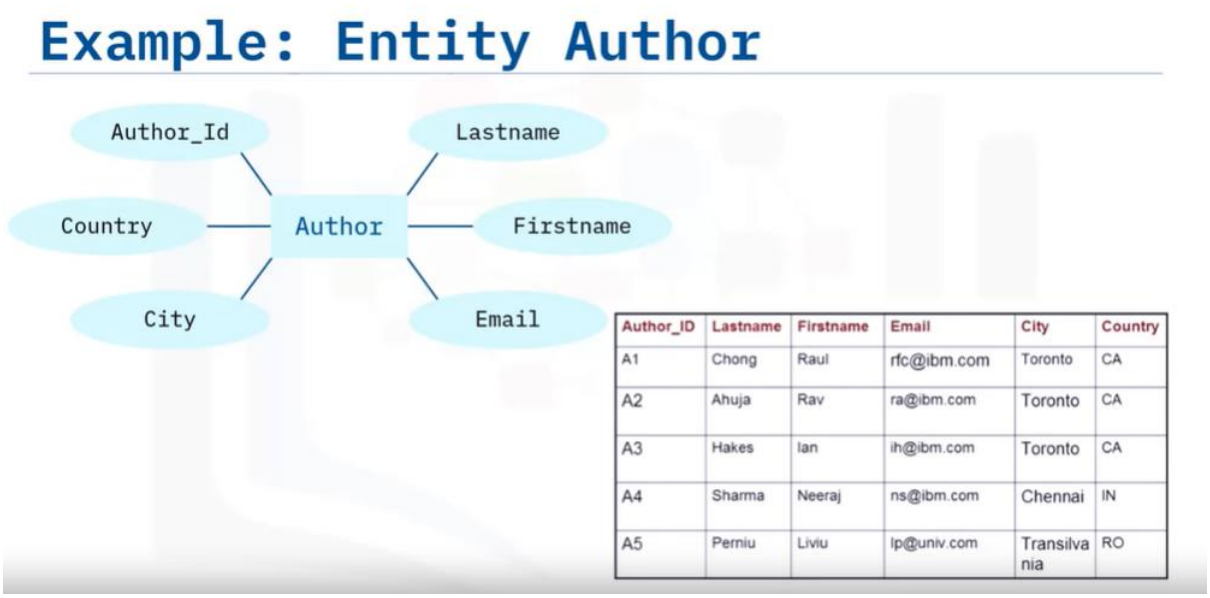

**Table: Book**

- 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 |

**Example: Entity Author**




| Author_ID | Lastname | Firstname | Email | City | Country |
|---|---|---|---|---|---|
| A1 | Chong | Raul | rfc@ibm.com | Toronto | CA |
| A2 | Ahuja | Rav | ra@ibm.com | Toronto | CA |
| A3 | Hakes | Ian | ih@ibm.com | Toronto | CA |
| A4 | Sharma | Neeraj | ns@ibm.com | Chennai | IN |
| A5 | Perniu | Liviu | lp@univ.com | Transilvania | RO |

**Summary:-**

- Entity-Relationship Diagrams (ERD) are the basic foundation for designing a database
- To translate an ERD into a relational database table:
  - The entity becomes the table
  - The attributes become columns in the table

## *6] Data Types*

**What are data types?**

- The information entered into each column should always be of the same sort, or type, of data.
- The data type that you assign to a column controls the data that the column can store.
- Different database management systems may use different names for similar data types, but they generally support a standard set.

**Data types: -**

1. **Character string: -**
   - Character string data types include fixed length data types and variable length.
   - **Fixed Length -** The length of a fixed length character string is often denoted in brackets after the type name, such as CHAR(10). This type uses the same amount of space in the database irrespective of the length of the actual data

stored in it. For example, storing a city code such as NY in a CHAR(10) column will still take up 10 characters of space.

- **Variable Length -** Variable length character strings, often named VARCHAR, can specify a maximum length for the string.

2. **Numeric: -**
   - **Integer Type –**
     - Only hold whole numbers, with no decimal parts.
     - **Types – INT, SMALLINT, BIGINT**
     - Typically use 2 or 4 bytes of storage to hold at numbers from negative 2 million or 32 thousand to positive 2 million or 32 thousand.
     - Smallints enable you to use less space for smaller numbers.
     - Bigints increase the size of the number that the data type can hold.

   - **Decimal Type –**
     - Can store whole numbers and decimal numbers.
     - **Types – DECIMAL, NUMERIC, FLOAT, SINGLE, DOUBLE, DEC, REAL, DECFLOAT.**

3. **Date/Time: -**
   - **Date –**
     - Consist of three-part values for the year, month, and day.
   - **Time –**
     - Consist of a three-part value for the hours, minutes, and seconds.
   - **Timestamp –**
     - Combination of both date & time and consists of seven parts: year, month, day, hour, minute, second, and microsecond.

4. **Boolean: -**
   - Only holds 1 bit of information: a 0 or a 1.
   - Can use these for true/false or yes/no type data.

5. **Binary String: -**
   - Holds a sequence of bytes that represent image, voice, or other media data.

6. **LOB (Large Object): -**
   - A very large object such as a file.
   - Often this type of data is stored outside of the main database table and a pointer to it is held in the table.

7. **XML: -**
   - Can store platform agnostic unstructured data in a hierarchical form.

8. **User defined types: -**
    - Many relational databases also allow you to create your own custom or "user defined" data types (UDTs) that are derived or extended from the built in types.

**Advantages of using data types: -**

- Data Integrity
- Data sorting
- Range Selection
- Data Calculation
- Use of standard functions

**Summary: -**

- Data types define the type of data that can be stored in a column.
- There are many different data types for all kinds of data.
- Using the correct data type for a column has many advantages.

## *7] Relational Model Concepts*

**Relational Model Concepts: -**

- First proposed in 1970, based on mathematical model and terms.
- Building Blocks:
    - Relation
    - Sets
- Set:
    - Unordered collection of distinct elements
    - Items of same type
    - No order and no duplicates

**Relational Database: -**

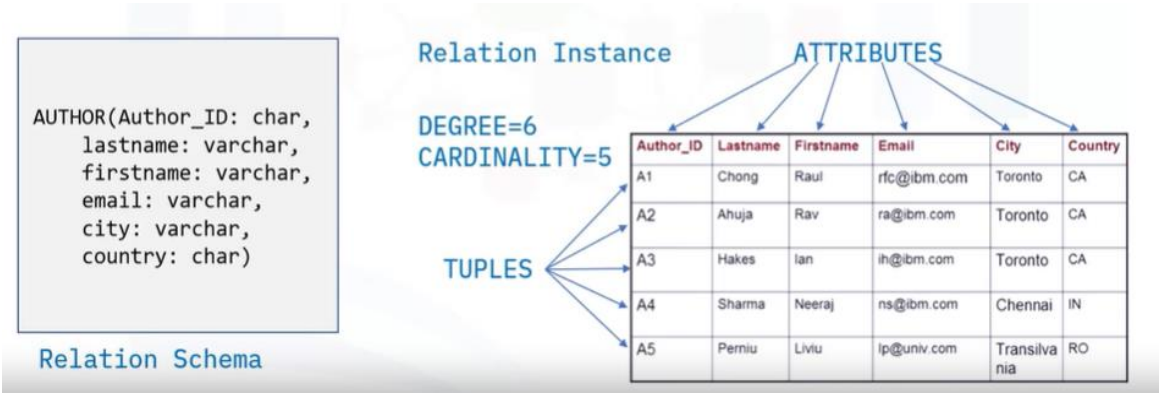
- A set of relations
- Relation = mathematical term for table
- 2 parts:
    - Relation Schema
    - Relation Instance

**Relation: -**

- Specifies: name of relation. Name and type of each column (attribute)

```
AUTHOR(Author_ID: char,
       lastname: varchar,
       firstname: varchar,
       email: varchar,
       city: varchar,
       country: char)
```

- A Relation Instance is a table made up of rows and columns.
- Columns = attributes = fields.
- Rows = tuples.
- Degree = number of attributes, or columns, in a relation.
- Cardinality = number of tuples, or rows.




**Summary: -**

- Relational Model of data is based on the concept of relation
- Relation:
  - Mathematical term for table.
  - A Relation Schema specifies the relation's name & attributes.
  - Relation Instance is a table made up of the attributes (columns) and tuples (rows)
- Degree refers to the number of attributes.
- Cardinality refers to the number of tuples.

## ***Summary & Highlights:-***

- The relational model is the most used data model for databases because this model allows for logical data independence, physical data independence, and physical storage independence.

- Entities are objects that exist independently of any other entities in the database, while attributes are the data elements that characterize the entity.

- The building blocks of a relationship are entities, relationship sets, and crows foot notations.

- Relationships can be one-to-one, one-to-many, or many-to-many.

- When translating an Entity-Relationship Diagram to a relational database table, the entity becomes the table and the attributes become columns in the table.

- Data types define the type of data that can be stored in a column and can include character strings, numeric values, dates/times, Boolean values and more.

- The advantages of using the correct data type for a column are data integrity, data sorting, range selection, data calculations, and the of standard functions.

- In a relational model, a relation is made up of two parts: A relation schema specifying the name of a relation and the attributes and a relation instance, which is a table made up of the attributes, or columns, and the tuples, or rows.

- Degree refers to the number of attributes, or columns, in a relation.

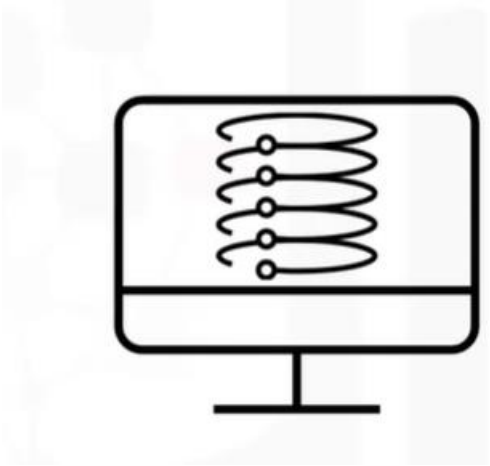
- Cardinality refers to the number of tuples, or rows in a relation.

# *Module 2: - Introducing Relational Database Products*

## *1] Database Architecture*

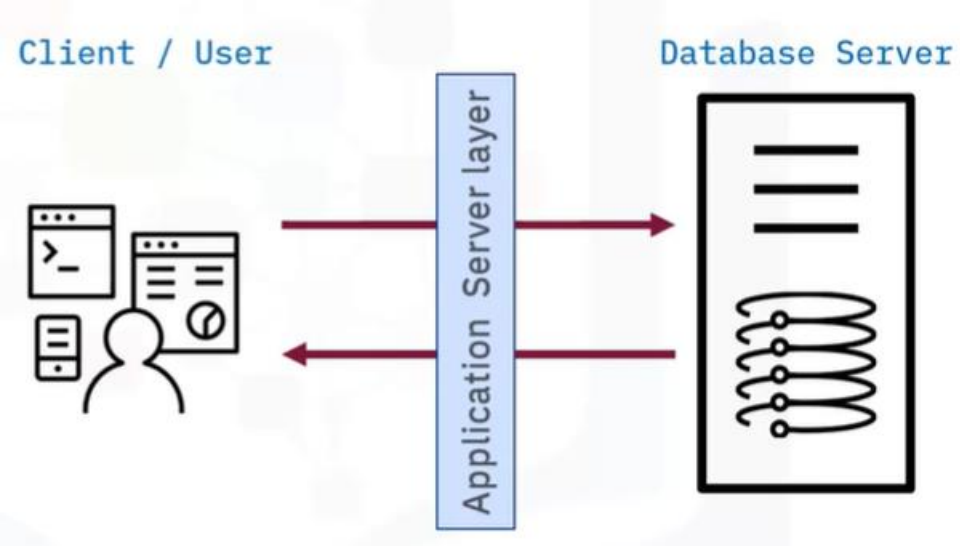
**Deployment Topologies: -**

- The deployment topology you use for your database is determined by how it will be used and accessed.
- **Local deployment** –
  - You can deploy a small database which requires limited user access on a local desktop.
  - The database resides on the user's system and access is often limited to a single user.
  - Single-tier-architecture.
  - Usage scenarios:
    - Development/Test
    - When database embedded in a local application.

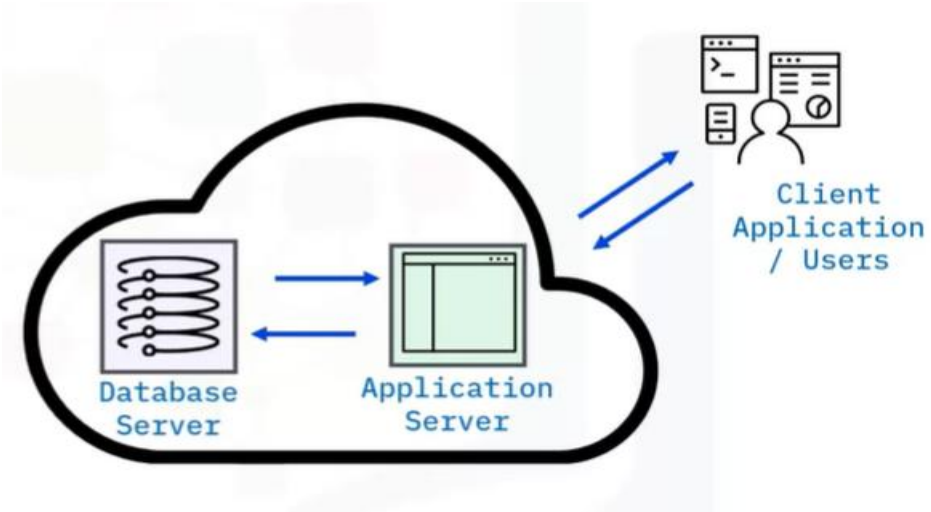


- **Client / Server deployment –**
  - You can deploy a larger database that many users must access in a client-server architecture.
  - The database resides on a remote server and users access it from client systems, often through a web page or local application.
  - Some scenarios employ a middle-tier (or an application server layer) between the application client and the remote database server.

- Usage scenarios:
  - Multi-user scenarios
  - Production environments

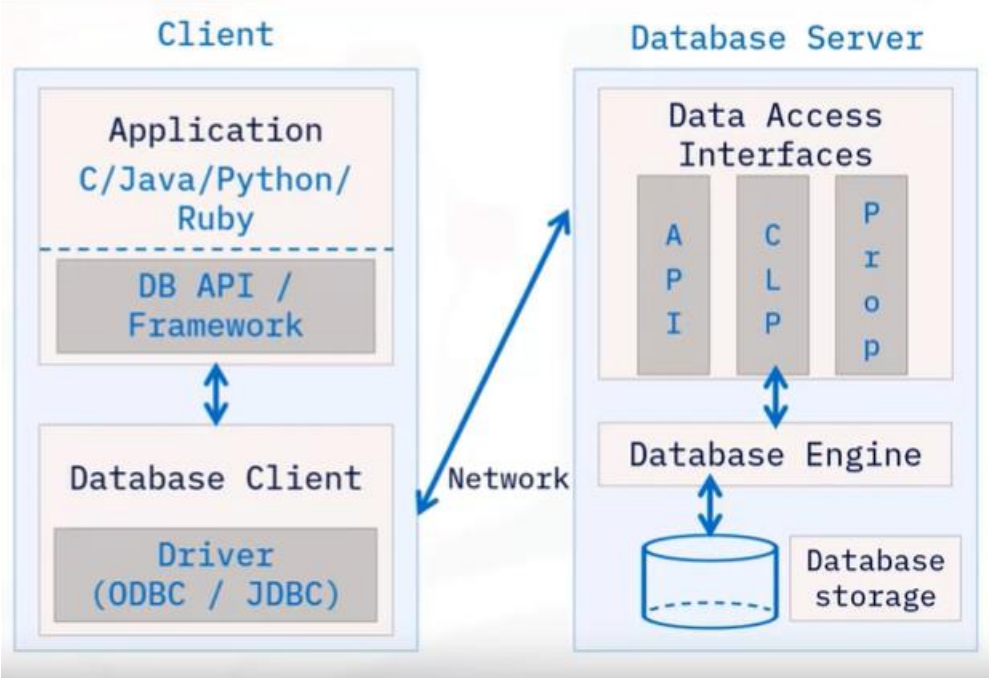



- **Cloud deployment –**
  - Deploying a database in the Cloud is an increasingly popular option.
  - The database resides in a Cloud environment and has all the advantages of a cloud-based service.
  - 2-tier architecture.
  - No need to download or install software.
  - No need to maintain the supporting infrastructure.
  - Users can access the cloud easily.
  - Clients access the database through an application server layer or interface in the cloud.
  - Cloud deployments are very flexible
  - Usage scenarios: For development, testing, and full production environments.



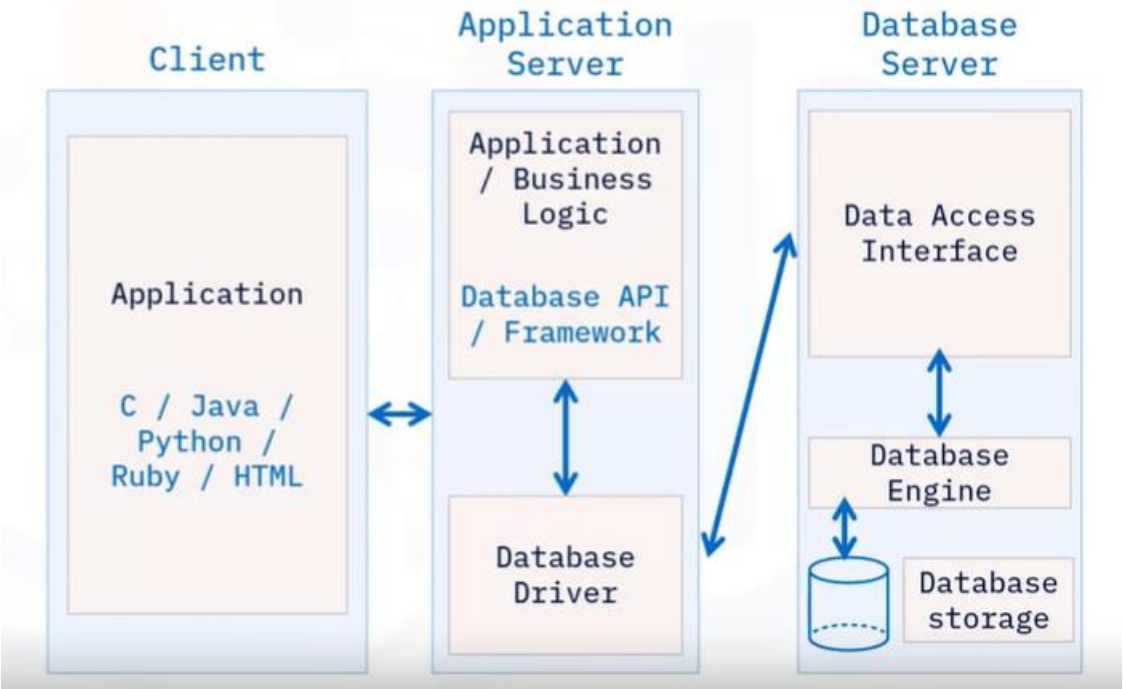
**2-Tier Database Architecture: -**

- Client application and database server run in separate tiers.
- Application connects to the database through an API or framework.
- Database interface communicates through a database client.
- **Database management Layers:**
  - **Data Access Layer -** includes interfaces for different types of clients which can include data industry standard APIs such as JDBC and ODBC, Command Line Processor (CLP) interfaces as well vendor specific or proprietary interfaces.
  - **Database Engine Layer -** contains an Engine which compiles queries, and retrieves and processes the data and returns the result set
  - **Database Storage or Persistence Layer -** is where the data is stored, which may be on local storage on the same device, or reside physically on network storage or specialized storage appliances




**3-Tier Database Architecture: -**

- Application presentation layer and business logic layer reside in different tiers.
- Users interact with a client presentation layer such as a mobile application.
- Client application communicates with application server
- Application server communicates with database server.




**Summary: -**

- Databases are deployed in different topologies, depending on which best suits the processing and access requirements.
- A single-tier topology is one where the database is installed on a user's local desktop.
- It is useful for small databases that only require single user access.
- In 2-tier database topologies the database resides on a remote server and users access it from client systems.
- In 3-tier database topologies the database resides on a remote server and users access it through an application server or a middle-tier.
- In Cloud deployments the database resides in the cloud, and users access it through an application server layer or another interface that also resides in the cloud.

## *2] Distributed Architecture and Clustered Databases*

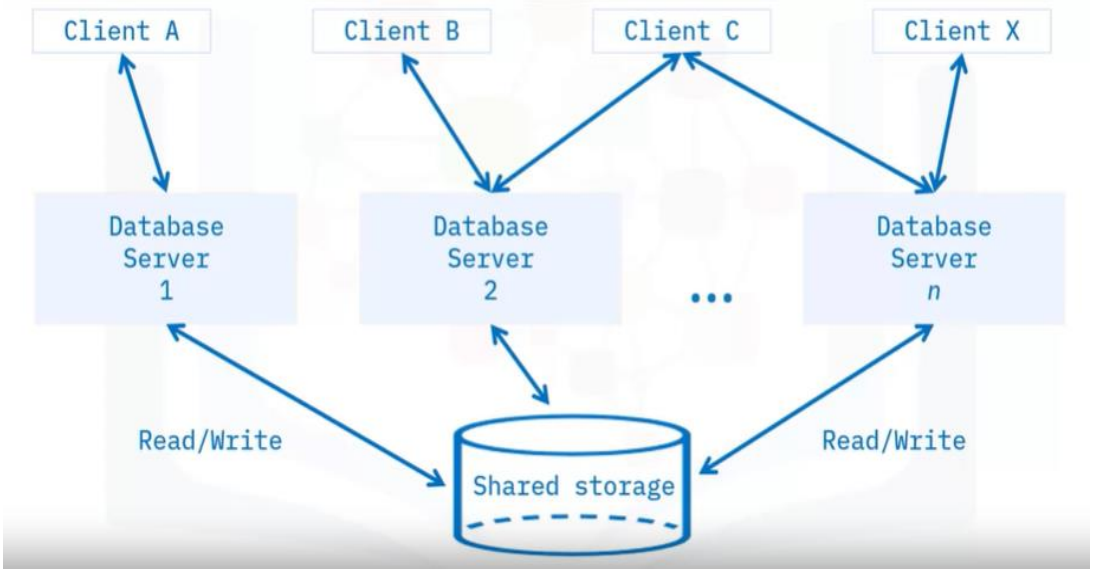
**Distributed Architectures: -**

- Mission Critical / Large scale workloads
- High Availability / High Scalability requirements
- Databases distributed on a cluster of servers
- Shared disk architecture
  - Share common storage
- Shared Nothing architecture
  - Replication

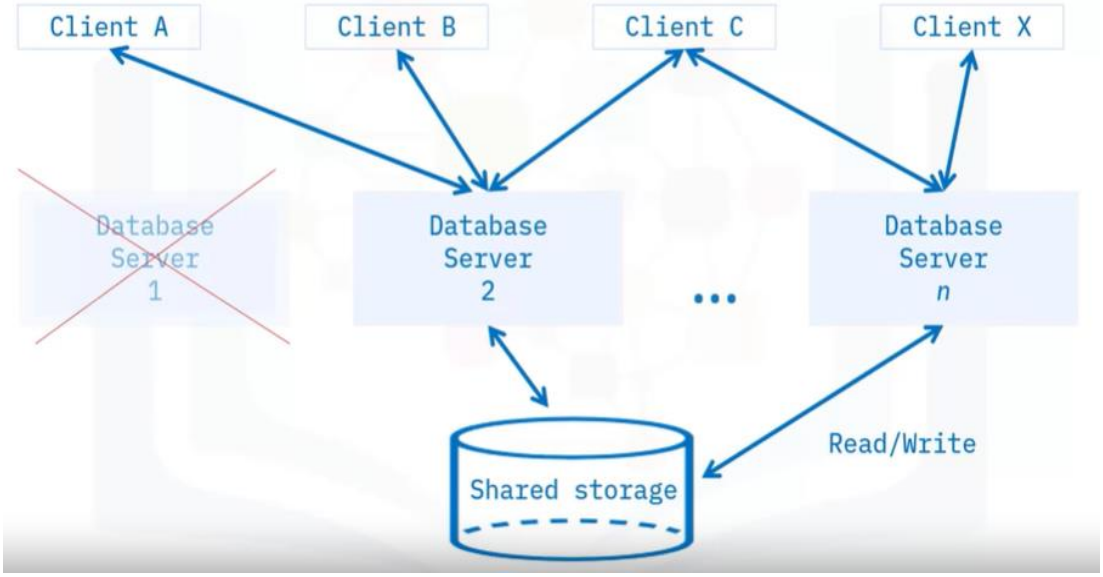
- Partitioning

**Shared disk architecture: -**

- Multiple database servers process the workload in parallel, allowing the workload to be processed faster.
- Each database server is connected to shared storage infrastructure and to each other using high speed interconnects. This allows the client workloads to be distributed amongst different database servers allowing for scalability.

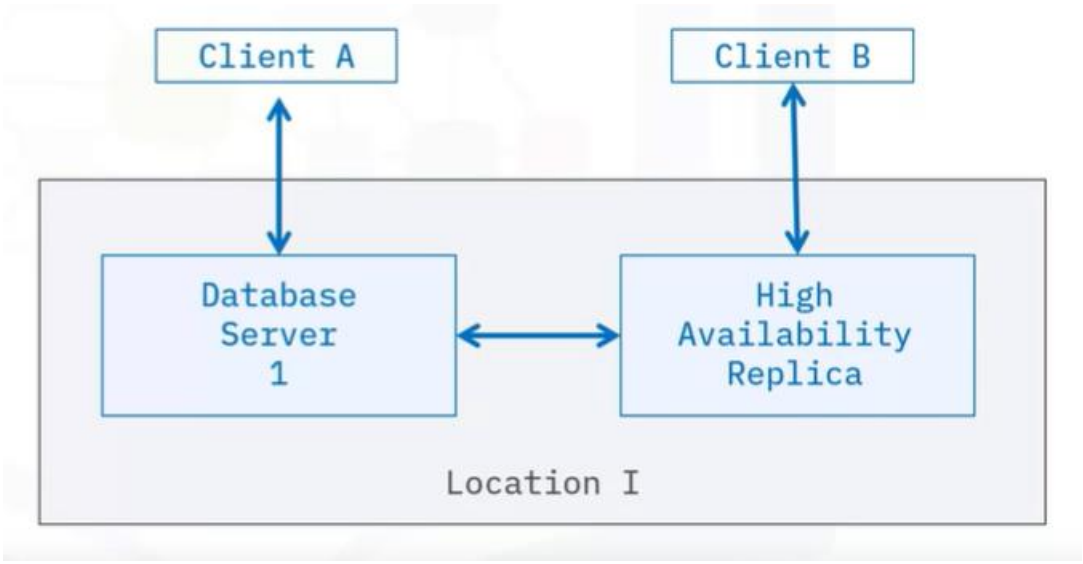



- In case of failure at one of the servers, the clients connected to it can be rerouted to other servers in the database cluster achieving for high availability.

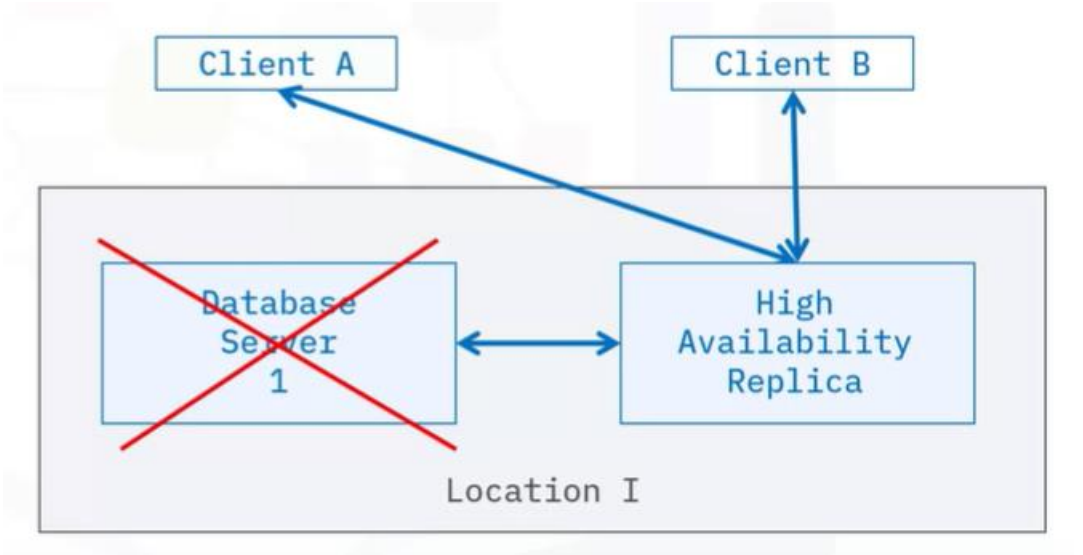


**Replication: -**

- Technique whereby changes taking place on a database server are replicated to one or more database replicas.
- Client workload is distributed across servers for improved performance. When the replica is within the same location it is known as a High Availability replica.



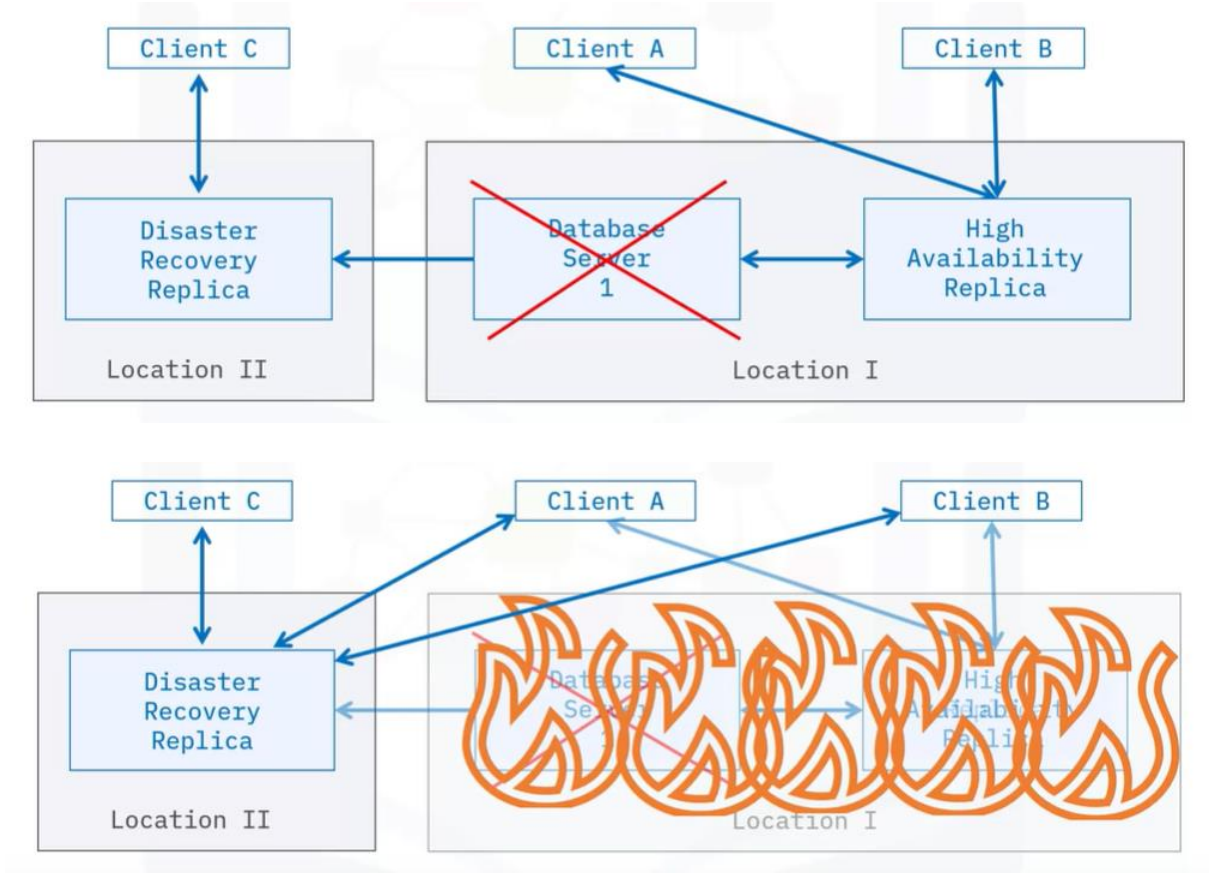

- In case of a failure on the Primary database server, such as a software or hardware failure, the Clients connected to it can be re-rerouted to the HA Replica.




**What happens if there is a site wide disaster?**

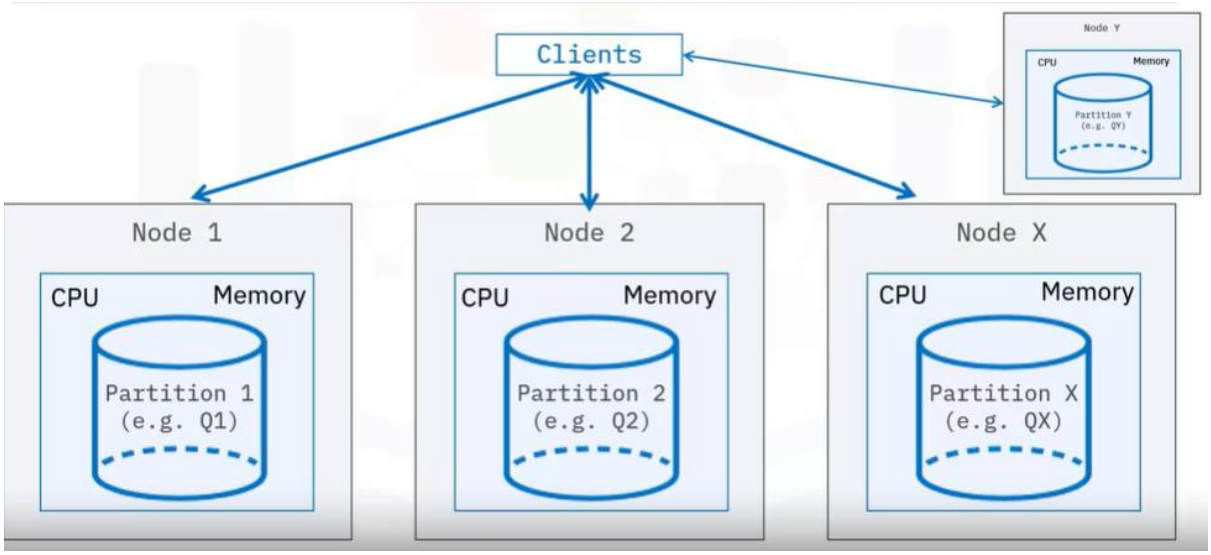
- To mitigate against such scenarios Replicas can be setup in geographically distributed locations.

- So, in case the entire data Center or location has an outage, such as a power outage, fire, earthquake or flood, the clients can be routed to Disaster Recovery replicas.




**Partitioning and Sharding: -**

- Need to contain very large quantities of data into multiple logical partitions, with each partition containing a subset of the overall data.
- When these partitions are placed on separate nodes in a cluster, it is called Sharding. Each shard contains its compute resources - Processing, Memory and Storage to act on its subset or partition of data.
- When a client issues a query, it is processed in parallel on multiple nodes or shards of the database and the query result from different nodes is synthesized and returned to the client. As the data or query workloads increases, additional shards and nodes can be added to the database cluster for increased parallel processing and improved performance.



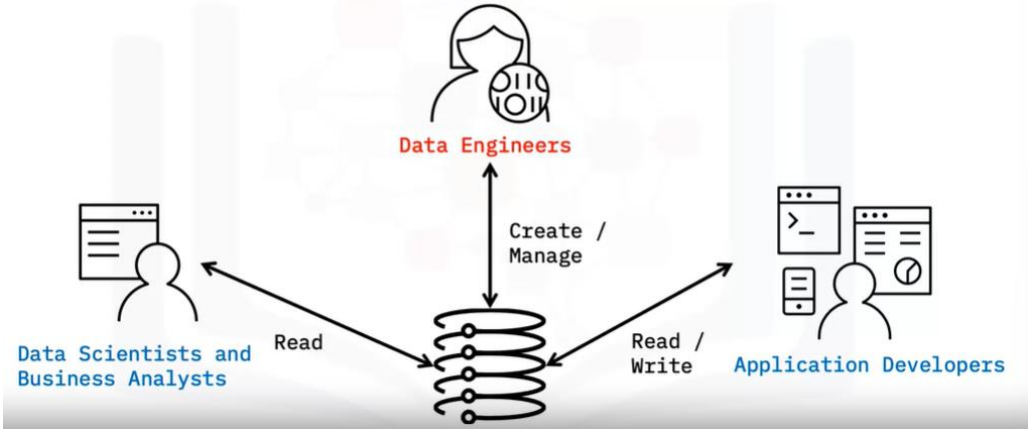

- Database Partitioning and Sharding are more common for data warehousing and business intelligence workloads that involve very large volumes of data.

**Summary: -**

- In shared disk database architectures multiple database servers process the workload in parallel, allowing the workload to be processed faster.
- In database replication changes taking place on a database server are replicated to one or more database replicas.
- In a single location database replication provides high availability. When a database replica is stored in a separate location, it provides a copy of the data for disaster recovery.
- In partitioning, very large tables are split across multiple logical partitions.
- In sharding, each partition has its own compute resources.

## *3] Database Usage Patterns*

**Database Access – Key user roles**



- Three main classes of Database users are: Data Engineers, Data Scientists and Business Analysts.

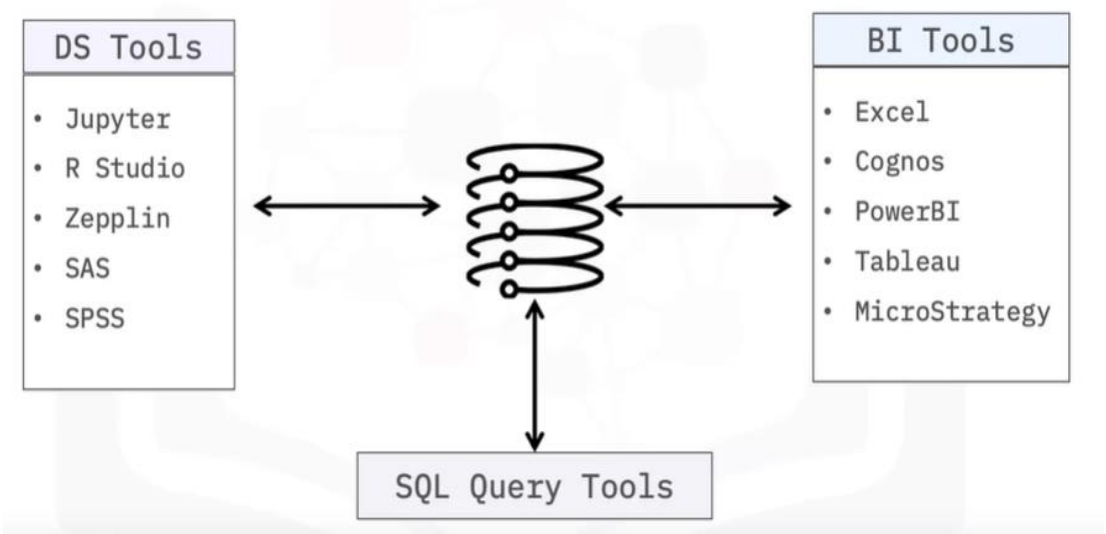
**Data Engineers and Database Administrators: -**

- Application Developers, Data Engineers and Database Administrators (DBAs) generally access the database for administrative tasks such as creating and managing database objects, setting access controls, monitoring and performance tuning.
- They typically use one or more of the following mechanisms for these tasks:
  - **GUI or web-based management tools** –
    - Most databases come with graphical tools or web-based tools (especially for cloud databases) or even mobile apps.
    - In some cases, the tools provided by the database vendors maybe limited in functionality or usability so third-party tools or specialized tools may also be available.
  - **Command line interfaces –**
    - Command line interfaces and utilities – With more fully functional GUI tools, command line usage has declined but a data engineer still requires some knowledge Command line interfaces can be simple database commands you issue from the terminal.
      eg: > db2 create database sample > mysqldump sakila > sakila.sql
    - Interactive command line shells such as sqlplus for oracle or db2
      clp > db2
      db2> connect to sample
    - SQL scripts and batch files that you execute from the shell.
  - **API –**
    - Many databases also include programmatic Interfaces or APIs for administrative tasks that can be invoked from applications and tools that the Data Engineers or third parties create Data analysts, data scientists, business analysts, and business intelligence analysts access databases for analyzing data in databases, deriving insight from this data, and making data driven predictions.

**Data Scientists and Business Analysts: -**

- Access databases for analyzing data in databases, deriving insight from this data, and making data driven predictions.
- Therefore, their data access patterns involve accessing existing data sources and is typically read-only.
- However, at times they may need to create database objects and populate them with data, especially in their own sandbox environments.

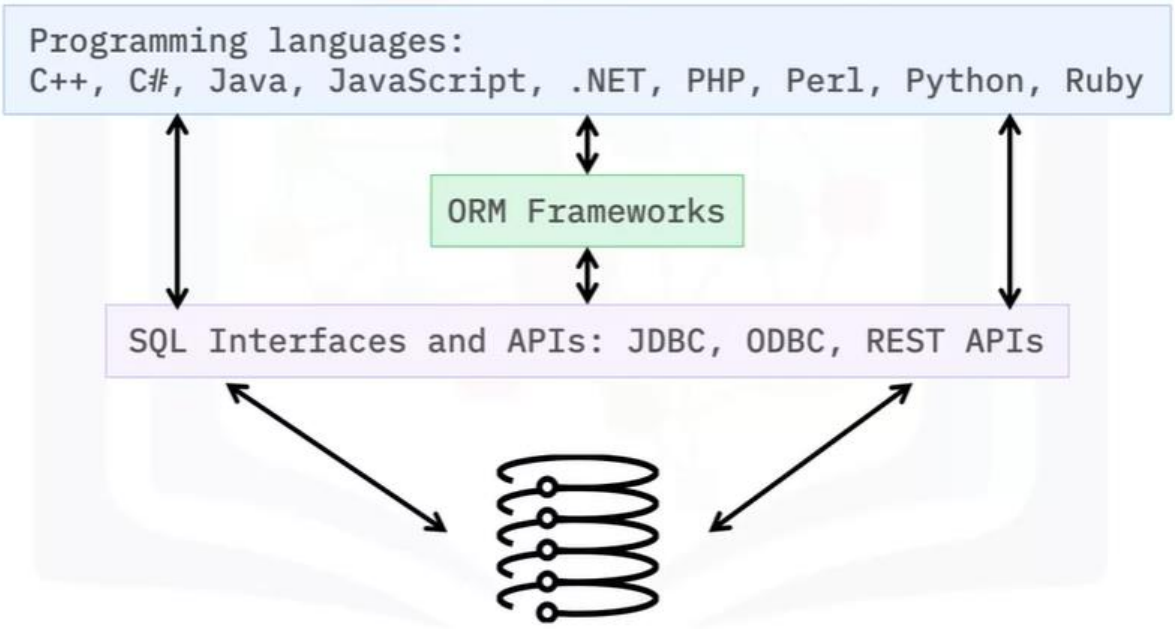
- Tools for data science & machine learning –
  - Jupyter
  - R Studio
  - Zeppelin
  - SAS
  - SPSS
- Tools for Reporting, Dashboarding, and Business Intelligence include spreadsheet software –
  - Microsoft Excel
  - IBM Congos
  - Microsoft Power BI
  - Tableau
  - MicroStrategy
- Both the DS and BI tools interface with relational databases using SQL interfaces and APIs.
- Some also need to use SQL Query tools for adhoc querying.
- Most databases come with their own visual SQL Query tools but there are also 3rd party tools that work with a variety of databases.




**Application Developers and Programmers: -**

- Application developers seldom access databases directly.
- They create applications which can require both read and write access to databases.
- Applications are written using programming languages such as C++, C#, Java, Javascript, .Net, PHP, Perl, Python, and Ruby.
- These languages talk to the database using SQL interfaces and APIs such as ODBC and JDBC. Some Databases, especially cloud-based ones, also include

REST APIs for accessing data. And while it is possible to program your applications using these lower-level APIs or REST APIs, and this is how applications in the past were developed, most programmers these days use object relational mapping (ORM) frameworks for working with databases, as they are easier to use and mask the complexity of the underlying relational database and SQL.




- ORM Framework include:
  - ActiveRecord in Ruby applications
  - Django in Python
  - Entity Framework in .NET
  - Hibernate in Java
  - Sequelize in JavaScript

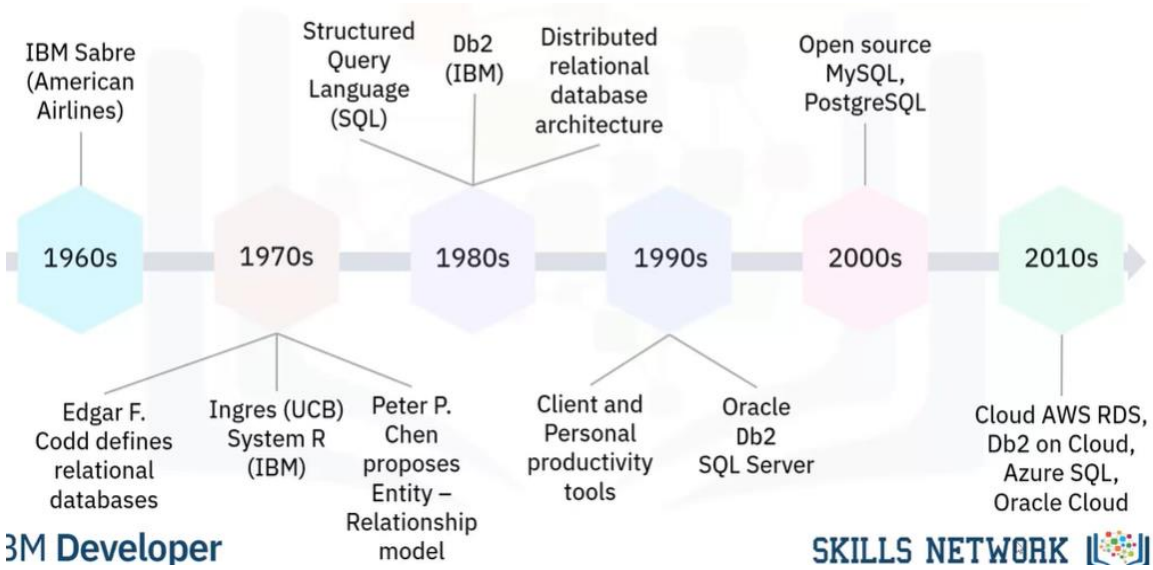
**Summary: -**

- Three main classes of users are:
  - Data Engineers
  - Data Scientists and Business Analysts
  - Application Developers
- Databases can be accessed through:
  - **Graphical and Web Interfaces** which make it easy to interact with the database visually.
  - **Command line tools and scripts** can be cumbersome to use but are powerful in the hands of an experienced Data Engineer and help with automating repetitive tasks.

- **APIs and ORMs** which help application developers create applications that access a database on behalf of a user or client application.
- Major categories of database applications include:
  - **Database Management tools** like phpMyAdmin or pgAdmin.
  - **Data Science and BI tools** like Microsoft Excel, IBM Congos, and MicroStrategy, which enable Data Scientists and Data Analysts to analyze data and produce targeted reports.
  - **Business & Industry applications** for tasks such as e-commerce, supply chain, etc.

## *4] Introduction to Relational Database Offerings*

**Relational database timeline:**




**Commercial relational databases:**

- **Oracle Database –**
  **Company:** Oracle Corporation
  **License:** Commercial

- **Microsoft SQL Server –**
  **Company:** Microsoft Corporation
  **License:** Commercial

- **IBM Db2 –**
  **Company:** IBM Corporation
  **License:** Commercial

**Open-source relational databases:**

- **MySQL –**
  Oracle Corporation
  General Public License (GPL) - version 2

- **PostgreSQL –**
  PostgreSQL Global Development Group
  PostgreSQL License (free and open-source, permissive)

- **SQLite –**
  Dwayne Richard Hipp
  Public Domain

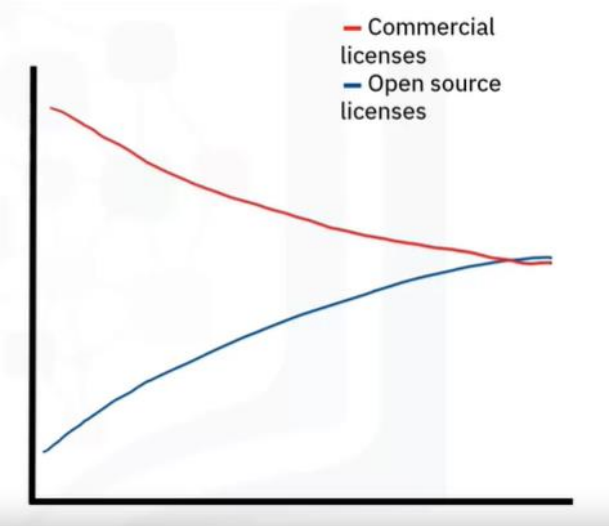
**Most Popular databases:**

DB Engine top ten as at February 2021:

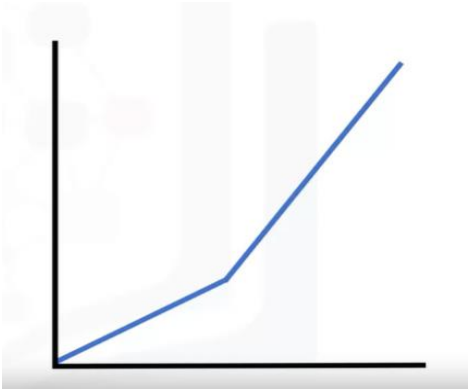
1. Oracle
2. MySQL
3. Microsoft SQL Server
4. PostgreSQL
5. IBM Db2
6. SQLite
7. Microsoft Access
8. MariaDB
9. Hive
10. Microsoft Azure SQL Database

**Popularity of license type:**

- Open-source licenses rising
- Commercial licences falling
- Each is now approx. 50% of the market



**Cloud databases:**



- Cloud database popularity increasing steadily
- Driven by move to SaaS model
- Highly scalable for data analytics
- By 2022, 75% of all databases will be cloud-based (Gartner)

**Leading cloud databases include:**

- Amazon DynamoDB
- Microsoft Azure Cosmos DB
- Microsoft Azure SQL DB
- Google BigQuery
- Amazon Redshift

**Summary: -**

- Relational databases are available with commercial licenses.
- Open-source relational databases are offered with a variety of licences that make them free to the user.
- The popularity of open-source relational databases has risen to approx. 50% in the past decade
- The popularity of Cloud databases has more than doubled.

## *5] Db2*

**History of Db2: -**

- **Database 2 (DB2) :**
  - Released in 1983
  - Originally for mainframes
  - Later for OS/2, UNIX, Linux, and Windows

- **Today DB2 is a whole suite of database management products including:**
  - Db2 Database
  - Db2 Warehouse
  - Db2 on Cloud
  - Db2 Warehouse on Cloud
  - Db2 Big SQL
  - Db2 Event Store
  - Db2 for z/OS

**Db2 Features: -**

- AI-powered functionality:
  - Machine learning algorithms
  - Column store
  - Data skipping
- Common SQL Engine
- Support for all data types
- High availability and disaster recovery
- Scalability
- Table partiotioning

**Db2 Products: -**

1. **Db2 Database**
   - On premises
   - Supported on Linux, UNIX, and Windows (LUW)
   - Provides:
     - Performance
     - High availability
     - Scalability
     - Resilience

2. **Db2 Warehouse**
   - On premises
   - Optimized for OLAP
   - Provides:
     - Advanced data analytics
     - Massively parallel processing (MPP)
     - Machine learning

3. **Db2 on Cloud**
   - Fully Managed
   - Cloud-based
   - Provides:
     - Performance
     - High availability
     - Scalability
     - Resilience

4. **Db2 Warehouse on Cloud**
    - Fully Managed
    - Cloud-based
    - Provides:
        - Advanced data analytics
        - Massively parallel processing (MPP)
        - Machine learning
5. **Db2 Big SQL**
    - SQL-on-Hadoop:
        - MPP
        - Advanced querying
    - Works with:
        - Hadoop HDFS and WebHDFS
        - RDBMS
        - NoSQL
    - Platforms:
        - Cloudera Data Platform
        - IBM Cloud Pak for Data

6. **Db2 Event Store**
    - Memory-optimized
    - Analyze streamed data for event-driven applications
    - Includes IBM Watson

7. **Db2 for z/OS**
    - Enterprise data server
    - Runs on IBM Z providing:
        - Availability
        - Scalability
        - Security
    - Mission critical
    - Millions of users

**Cloud Pak: -**

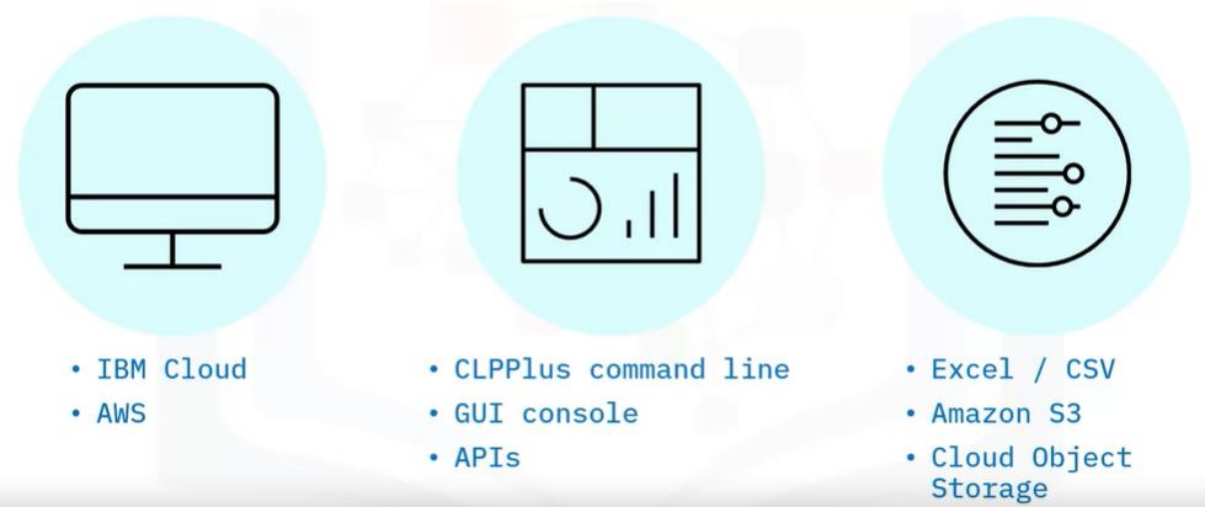
- Fully integrated data and AI platform
- Runs on Red Hat OpenShift:
    - Private cloud
    - Public cloud
    - Hybrid cloud
- Enables you to:

- Connect to data – on premises, private cloud, public cloud
- Organise data
- Analyze data
- Infuse AI

**Db2 on Cloud Plans: -**

| Lite | Standard | Enterprise |
|---|---|---|
| Free<br>Time unlimited<br>200 MB storage<br>15 connections | Flexible scaling:<br>• Compute capability<br>• Storage<br>HA clustering | Dedicated instance<br>Flexible scaling:<br>• Compute capability<br>• Storage<br>HA clustering |

**Working with Db2 on Cloud: -**




**Db2 high availability: -**

- Db2 provides high availability disaster recovery, or HADR, functionality to support high availability systems.
- HADR replicates changes made at a primary database to up to three standby servers.
- If the primary database fails for any reason - hardware, software, or network issues - you can automatically promote one of the standby databases to be the primary database,

redirect client applications to this new primary database, and continue to replicate to the other standby servers in the group.

- When the original primary database comes back online, it can either take the place of a standby server or be promoted back to the primary position.

**Scalability with partiotioning: -**

- Db2 Warehouse offers massively parallel processing and data analytics for BI workloads.
- At times, you may need to scale the storage capabilities of your system to meet peak demand or to reduce costs when demand is low.
- Data in Db2 Warehouse is stored in data nodes. To scale up your storage capacity, you just add a node to your deployment. The partitions and their workloads are then automatically rebalanced across the new node setup. Similarly, to scale down, you just remove a node to return to your original state.

**Summary: -**

- Db2 is a family of products that you can use to work with and manage your data in whatever way you need.
- You can deploy Db2 across many platforms, both on premises and in the cloud.
- Cloud Pak for Data includes Db2 and many IBM tools for data.
- Db2 on Cloud is a fully managed, cloud-based SQL database that can run on IBM Cloud or AWS.
- Db2 provides high availability, disaster recovery, and scalability functionality.

## *6] MySQL*

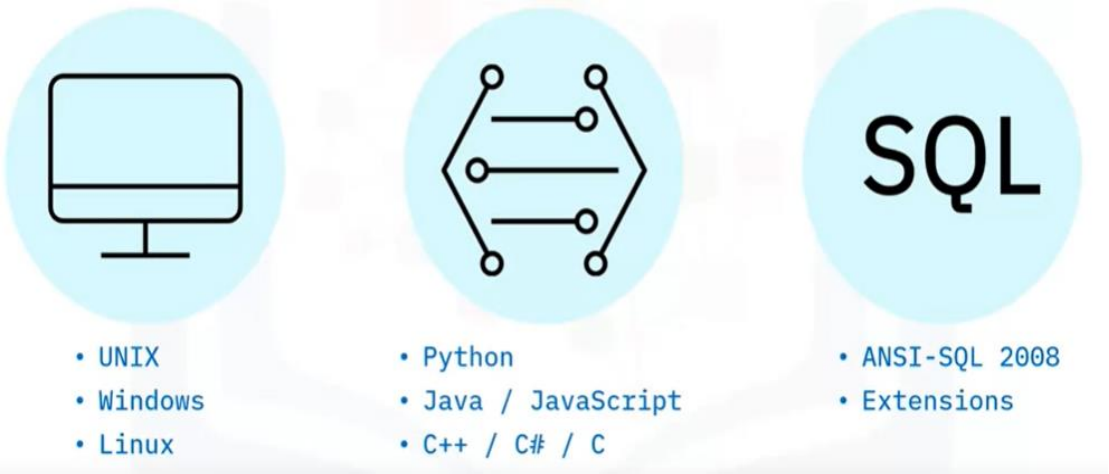
**History of MySQL: -**

- Developed by MySQL AB
- MySQL name after the Monty Widenius's daughter, My
- Sakila the dolphin
- LAMP stack
- Dual license:
  - GNU General Public License (GPL)
  - Commercial license
- MariaDB

**What is MySQL?**

- Object-relational database management system
- Low Maintenance database
- Is available in various flavors & editions including a clustered version for demanding workloads.

**Working with MySQL: -**




- MySQL works primarily with relational data, but also supports JSON.

**MySQL storage engines: -**

- Handle the SQL operations on a table.

| InnoDB | MyISAM | NDB |
|---|---|---|
| Transactions<br>Row-level locking<br>Clustered indexes<br>Foreign keys | For data warehouse and web applications<br>Table-level locking | Cluster<br>High availability<br>High redundancy |

- **InnoDB –**
  - Default Storage Engine
  - Supports transactions to ensure the consistency of data
  - Row-level locking to improve a balance of high performance and reliability.

- **MyISAM –**
  - Good for workloads with mainly read operations & few updates, such as data warehouse or web applications.
  - Uses table-level locking which inhibits performance in a read/write environment.

- **NDB –**
  - Supports multiple instances of MySQL servers running in a cluster.
  - Primarily used for applications that require high levels of availability & redundancy.

**High Availability and Scalability: -**

- You can use replication to create a copy of data on one to more replicas.
- Data changes at the source database are then also performed at the replicas.
- Multiple copies of the same data means that you can share the read load for data across the replica set, improving scalability.
- And replication also increases availability because if the source database fails, you can failover to use one of the replicas instead. MySQL provides two clustering options.
- The first uses the InnoDB storage engine with group replication and enables you to work with one read-write primary server and multiple secondary servers.
- You can then use MySQL Router to load balance client applications across the multiple server instances and, in the event of unplanned downtime of any of the servers, MySQL Router will reconnect client applications to an available server.
- Alternatively, the MySQL Cluster edition uses the NDB storage engine to provide a highly available and scalable solution. Multiple MySQL server nodes access a set of data nodes (usually stored in memory). Running multiple data nodes provides redundancy and hence increased availability in the event of failure, and running multiple server nodes provides for scalability.

**Summary: -**

- MySQL is an object-relational database, available in various editions MySQL supports many operating systems.
- MySQL supports a supports a range of languages for client application development.
- MySQL supports relational and JSON data.
- MySQL provides multiple storage engines for differing workloads.
- MySQL provides high availability and scalability options.

## *7] PostgreSQL*

**History of PostgreSQL: -**

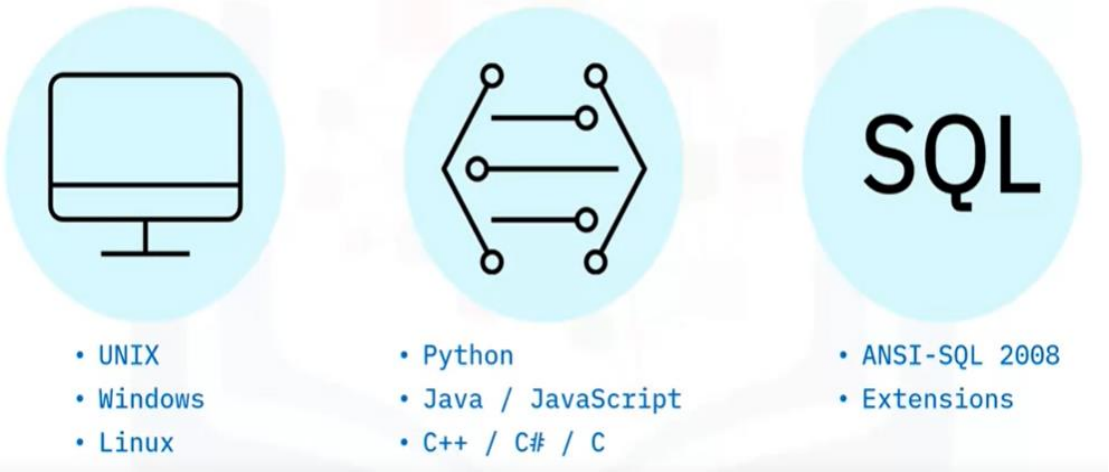
- Originates from POSTGRES project:
  - Research and production applications
  - Across finance, aviation, and medicine

- Postres95:
  - Open Source
  - Included SQL language interpreter
- PostgreSQL:
  - LAPP stack
  - Extensions such as PostGIS

**What is PostgreSQL?**

- Free open-source object-relational database management system.
- The open-source part means that you can use, modify, and distribute the Postgres source code to meet your business requirements.
- The object part describes that, like object-oriented programming languages, it supports inheritance and overloading.
- You can use these techniques to simplify your design and reuse database objects.

**Working with PostgreSQL: -**

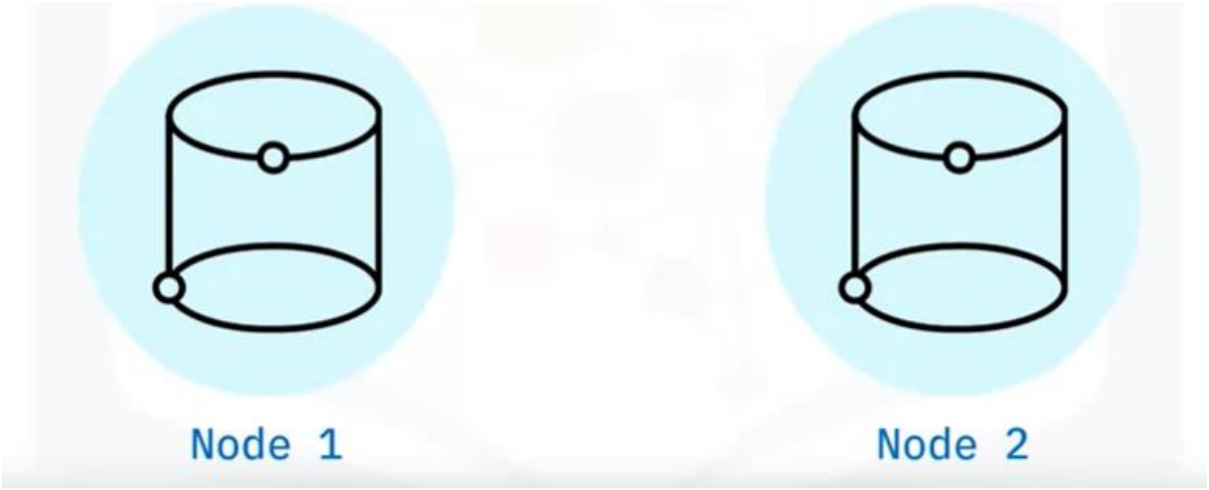



- When using Postgres, you can use all the standard relational database constructs, such as keys, transactions, views, functions, and stored procedures.

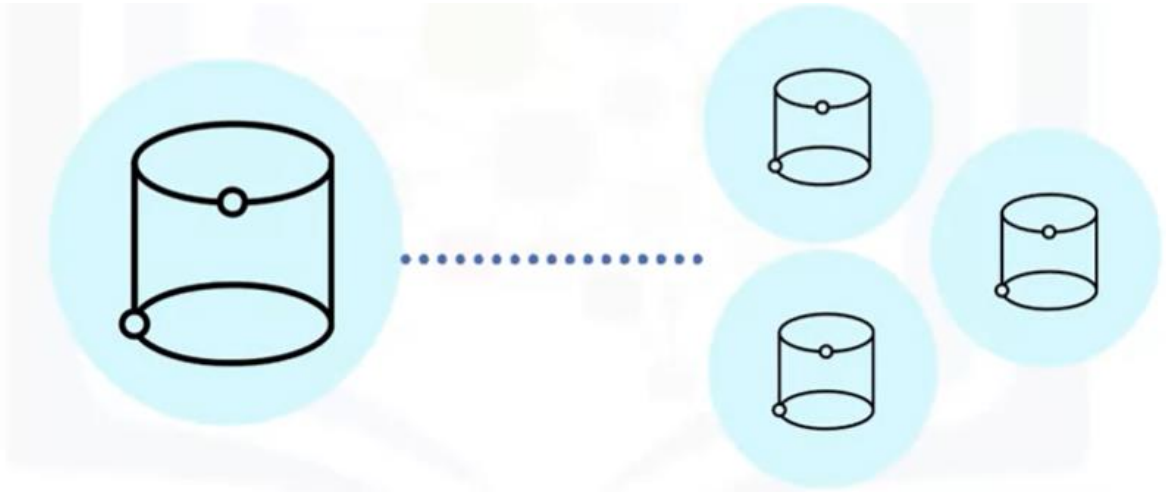
- And you can also use some of the NoSQL functionality, such as JSON for structured data and HSTORE for non-hierarchical data.

**High Availability and Scalability: -**

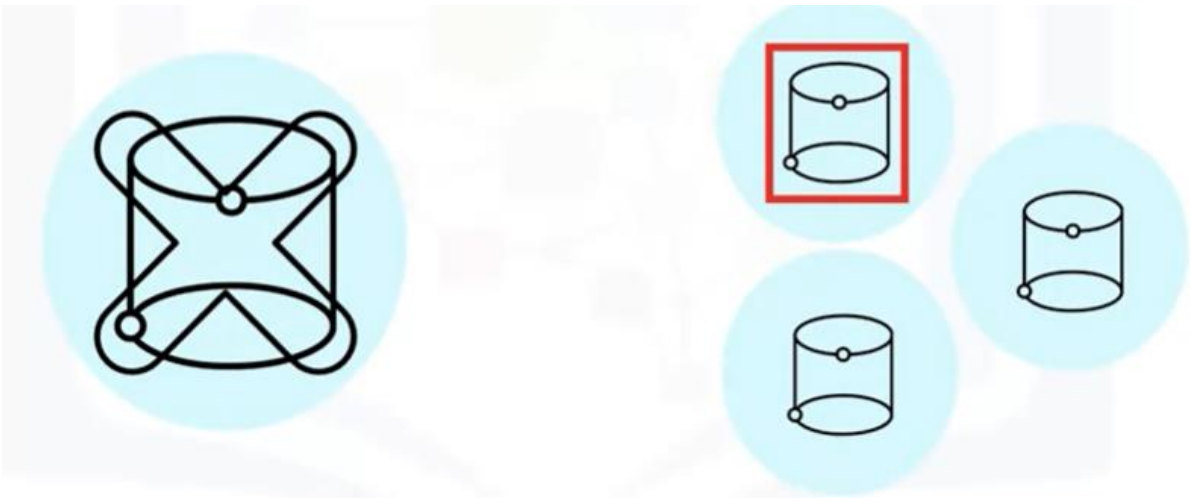
- Postgres supports replication for high availability.
- It supports two-node synchronous replication.
- This stores a copy of your data on a second server and applies each change you make to Node 1 at Node 2.




- You can then share read loads across the two servers and if Node 1 fails, you can enable Node 2 to service all clients until Node 1 is running again.
- It also supports multi-node asynchronous replication for high availability and scalability.
- Here, one master node distributes its changes to multiple read-only replicas for scalability purposes.



- And again, if the read/write node fails, you can quickly replace it with one of the read-only replicas.



- For even greater flexibility for scaling applications, you can use commercial editions such as EDB PostgreSQL Replication Server which provide multi-master read/write replication. This enables you to run multiple read/write databases that all replicate changes with each other. If one fails, users can easily be redirected to another instance until it is available again.
- Other technologies that have been added to PostgreSQL in recent releases to enhance scalability and working with larger data sets include:
  - Partitioning which enables you to split a large table into multiple smaller sections, or partitions, to improve query performance and
  - Sharding which enables you to store horizontal partitions across multiple remote servers.

**Summary: -**

- PostgreSQL:
  - Is an open source, object-relational database.
  - Supports a range of languages for client application development.
  - Supports relational, structured, and non-structured data.
  - Supports replication and partitioning for high availability and scalability.

## ***Summary & Highlights:-***

There are four types of database topology:

- **Single tier -** The database is installed on a user's local desktop.
- **2-tier -** The database resides on a remote server and users access it from client systems.
- **3-tier -** The database resides on a remote server and users access it through an application server or a middle tier.
- **Cloud deployments -** The database resides in the cloud, and users access it through an application server layer or another interface that also resides in the cloud.

In shared disk distributed database architectures, multiple database servers process the workload in parallel, allowing the workload to be processed faster. There are three shared nothing distributed database architectures:

- **Replication -** Changes taking place on a database server are replicated to one or more database replicas. In a single location, database replication provides high availability. When database replica is stored in a separate location, it provides a copy of the data for disaster recovery.
- **Partitioning -** Very large tables are split across multiple logical partitions.
- **Sharding -** Each partition has its own compute resources.

There are different classes of database users, who use databases in different ways:

- Three main classes of users are Data Engineers, Data Scientists and Business Analysts, and Application Developers.
- Database users can access databases through Graphical and Web interfaces, command line tools and scripts, and APIs and ORMs.
- Major categories of database applications include Database Management tools, Data Science and BI tools, and purpose built or off the shelf business applications.
- Relational databases are available with commercial licenses or open source.
- MySQL is an object-relational database that supports many operating systems, a range of languages for client application development, relational and JSON data, multiple storage engines, and high availability and scalability options.
- PostgreSQL is an open source, object-relational database that supports a range of languages for client application development, relational, structured, and non-structured data, and replication and partitioning for high availability and scalability.

# Week 2

# Using Relational Databases

## Module 1: - Creating Tables and Loading Data

### 1] Types of SQL statements (DDL vs. DML)

- SQL Statements are used for interacting with Entities (that is, tables), Attributes (that is, columns) and their tuples (or rows with data values) in relational databases.
- SQL statements fall into two different categories:
    1. Data Definition Language statements and
    2. Data Manipulation Language statements

**DDL (Data Definition Language): -**

- Define, change or drop data
- Common DDL:
    - **CREATE -** which is used for creating tables and defining its columns.
    - **ALTER -** is used for altering tables including adding and dropping columns and modifying their datatypes.
    - **TRUNCATE -** is used for deleting data in a table but not the table itself.
    - **DROP -** is used for deleting tables.

**DML (Data Manipulation Language): -**

- Used to read and modify data in tables.
- CRUD operations (Create, Read, Update & Delete rows)
- Common DML:
    - **INSERT –** is used for inserting a row or several rows of data into a table.
    - **SELECT –** reads or selects row or rows from a table.
    - **UPDATE –** edits row or rows in a table.
    - **DELETE –** removes a row or rows of data from a table.

**Summary: -**

- DDL used for defining objects (tables)
- DML used for manipulating data in tables.

## *2] Creating Tables*

**Creating Tables: -**

**Considerations for creating tables -**

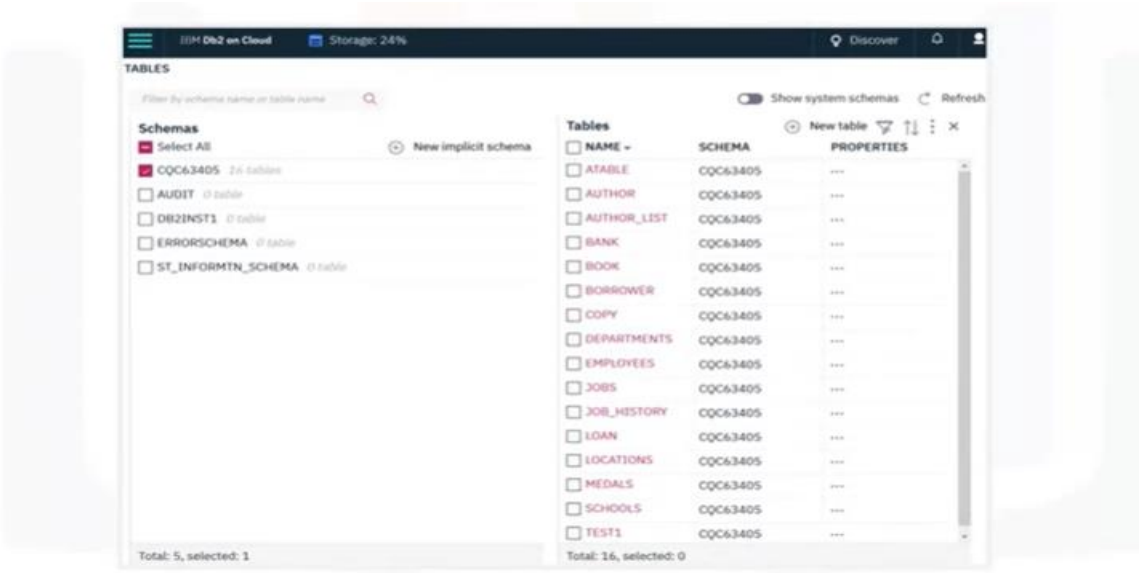
- **Schema –** way of organizing database objects into logical groups.
- **Table information –** Table name, Column names, Data Types.
- **Duplicates –** Whether a column can contain duplicate values
- **Null Value –** If a column will allow null values

**Methods for creating tables: -**

- **Visual or UI tools –** smaller & occasional tasks
  - Db2 on Cloud console
  - MySQL phpMyAdmin
  - PostgreSQL PGAdmin
- **CREATE TABLE** SQL statement – or a script file to help to automate the creation process when creating multiple tables.
- **Administrative APIs –** creating & managing databases programmatically.

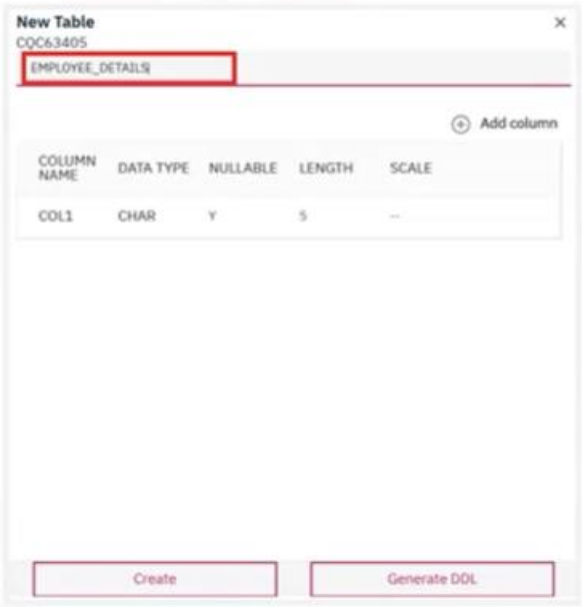
***Illustrative Example –***

- The first step in creating a table using the Db2 on Cloud console is to choose a schema to hold the table. In this case, the user schema, CQC63405 is selected. In Db2 the default schema is the username. In this example, you can see that the username is CQC63405. Each user will have a different username, so you will see a different username in your own Db2 database. You can create new Schemas to hold your tables, views, and other database objects. In this case, the user schema, CQC63405 is selected. Select New table to create a new table. Give your new table a name. In this example, the table is named Employee details.



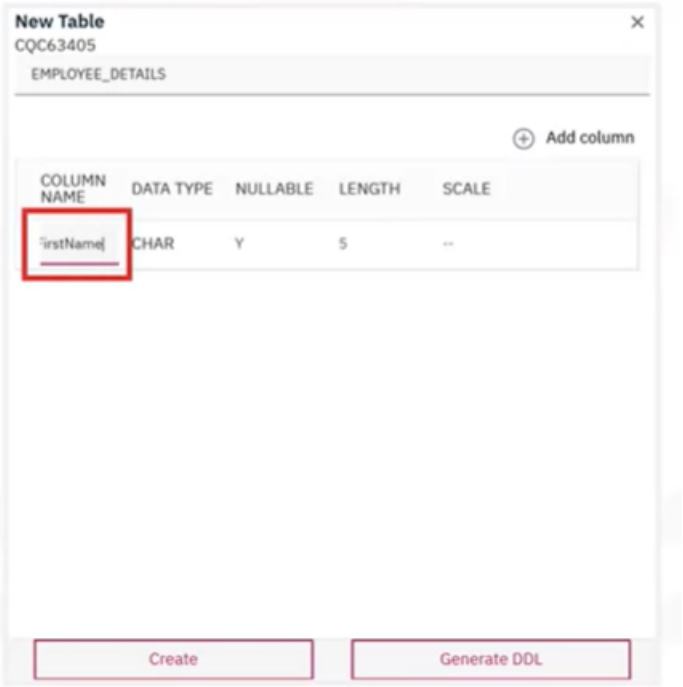

**Naming the new table: -**

It exists within the CQC63405 schema, the fully qualified name for this table is CQC63405.Employee Details.



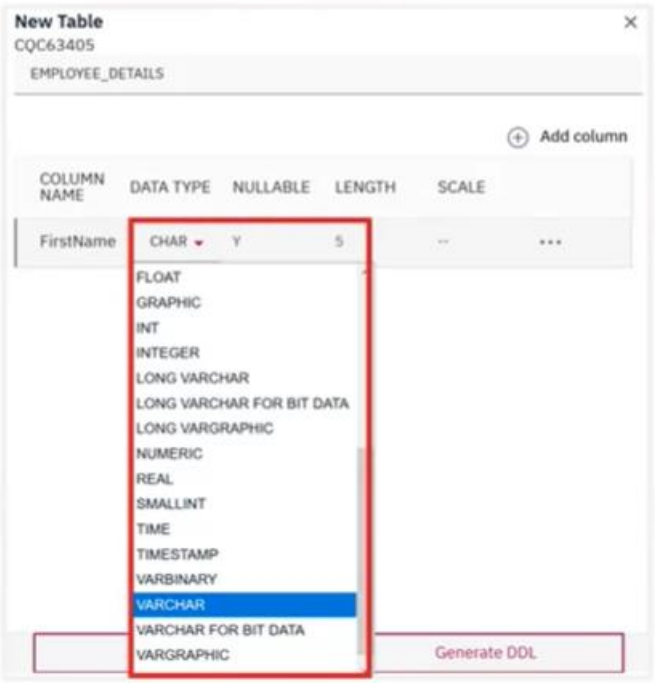

**Adding column and data types: -**

- The new table has a single column by default. You can rename this column to fit your needs.



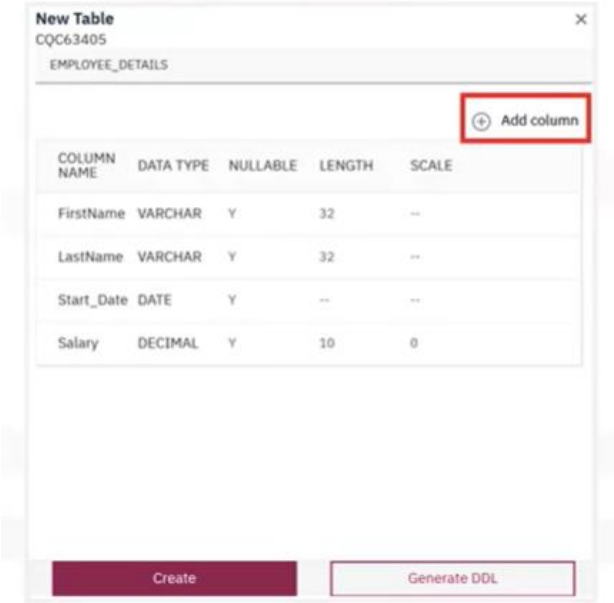

**<u>Specifying data types</u>: -**

- Set a data type for your column by selecting one from the list.



**Adding Columns: -**

- Use Add column to continue to add columns until you have constructed the entire table.




- Remember to specify the data type for each column. You can also specify whether the column accepts null values, and specify length and scale depending on the data type.

**Post-creation changes: -**

- Many actions take once you have created the table.
- Can Drop, or delete, the table.
- Can Generate SQL code to perform actions such as SELECT, INSERT, UPDATE, and DELETE.
- Can ALTER the table to add a new column, set constraints, or change the structure of the table in some other way.
- Can see the database objects that the table depends on.

**Summary: -**

- Many Relational Database Management Systems (RDBMS) have schemas which contain tables, views, functions, and other database objects.
- Most RDBMS provide a GUI which you can create tables through.
- You can also use SQL statements to create tables.
- You can alter the structure of the table after it's created, should you need to add a column, change a data type, or add a primary or foreign key.

## 3] CREATE TABLE Statement

**CREATE table: -**

**Syntax –**

```
CREATE TABLE table_name
   (
    column_name_1 datatype optional_parameters,
    column_name_2 datatype,
    ...
    column_name_n datatype
   )
```

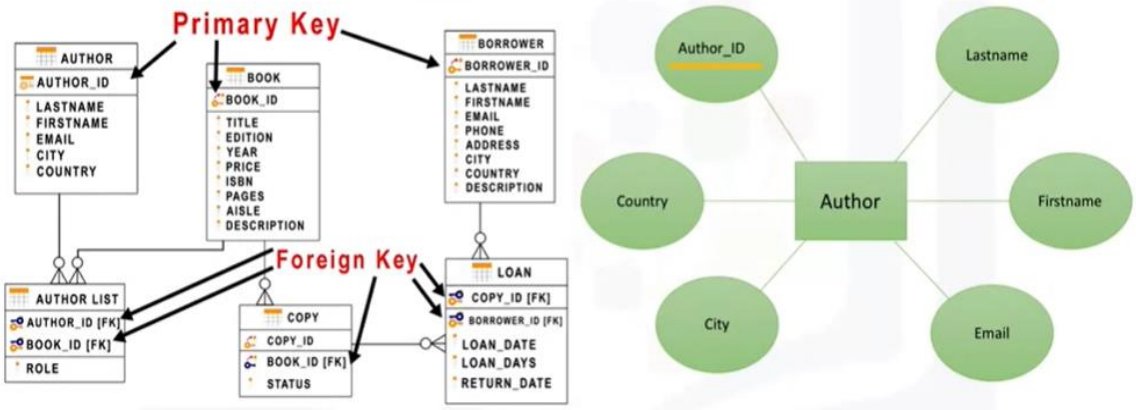
**Example –**

Create a table for Canadian provinces

```
CREATE TABLE provinces(
    id char(2) PRIMARY KEY NOT NULL,
    name varchar(24)
   )
```

| id<br>*char(2)* | name<br>*varchar(24)* |
|---|---|
| AB | ALBERTA |
| BC | BRITISH COLUMBIA |
| ... | ... |

Create a table



- **Primary Key:** Uniquely Identifies each Row in a Table.

To create the Author table, use the following columns and datatypes:

AUTHOR(Author_ID:char, Lastname:varchar, Firstname:varchar, Email:varchar, City:varchar, Country:char)

```
CREATE TABLE author (
    author_id CHAR(2) PRIMARY KEY NOT NULL,
    lastname VARCHAR(15) NOT NULL,
    firstname VARCHAR(15) NOT NULL,
    email VARCHAR(40),
    city VARCHAR(15),
    country CHAR(2)
  )
```

***Note:***

- Author_ID is the Primary Key. This constraint prevents duplicate values in the table.
- Also, Last Name and First Name have the constraint NOT NULL. This ensures that these fields cannot contain a NULL value, since an author must have a name.

**Summary: -**

- CREATE is a DDL statement for creating Entities or tables in a database.
- The CREATE TABLE statement includes definition of attributes of columns in the table, including:
  - Names of columns
  - Datatypes of columns and
  - Constraint (e.g., Primary Key)

## *4] ALTER, DROP, and TRUNCATE*

### 1. ALTER statement: -

**ALTER TABLE…….ADD COLUMN: -**

- Add or remove columns
- Modify the data type of columns
- Add or remove keys
- Add or remove constraints

**Syntax –**

```
ALTER TABLE <table_name>
  ADD COLUMN <column_name_1> datatype
  . . .
  ADD COLUMN <column_name_n> datatype;
```

**Example –**

- To add a telephone number column to the AUTHOR table in the library database to store the author's telephone number, use the following statement:

```
ALTER TABLE author
  ADD COLUMN telephone_number BIGINT;
```

| author_id | lastname | firstname | email | city | country | telephone_number |
|---|---|---|---|---|---|---|
| 1001 | Thomas | John | johnt@... | New York | USA | 5551111 |
| 1002 | James | Alice | alicej@... | Seattle | USA | 5551112 |
| 1003 | Wells | Steve | stevew:@... | Montreal | Canada | 5552222 |
| 1004 | Kumar | Santosh | kumars@... | London | UK | 5553333 |

**ALTER TABLE…….ALTER COLUMN: -**

**Syntax –**

```
ALTER TABLE <table_name>
  ALTER COLUMN <column_name> SET DATA TYPE
<datatype>;
```

**Example –**

- Using a numeric data type for telephone number means that you cannot include parentheses, plus signs, or dashes as part of the number. You can change the column to use the CHAR data type to overcome this.

```
ALTER TABLE author
  ALTER COLUMN telephone_number SET DATA TYPE
CHAR(20);
```

| author_id | lastname | firstname | email | city | country | telephone_number |
|---|---|---|---|---|---|---|
| 1001 | Thomas | John | johnt@... | New York | USA | 555-1111 |
| 1002 | James | Alice | alicej@... | Seattle | USA | 555-1112 |
| 1003 | Wells | Steve | stevew@... | Montreal | Canada | 555-2222 |
| 1004 | Kumar | Santosh | kumars@... | London | UK | 555-3333 |

**ALTER TABLE…….DROP COLUMN: -**

- To delete a column from a table.

**Syntax with example –**

- Changing a column from the CHAR data type to a numeric data type will not work if the column already contains non-numeric data. If you try to do this, you will see an error message in the notification log and the statement will not run. If your spec changes and you no longer need this extra column, you can again use the ALTER TABLE statement, this time with the DROP COLUMN clause, to remove the column as shown:

```
ALTER TABLE author
  DROP COLUMN telephone_number;
```

| author_id | lastname | firstname | email | city | country | telephone_number |
|---|---|---|---|---|---|---|
| 1001 | Thomas | John | johnt@... | New York | USA | 555-1111 |
| 1002 | James | Alice | alicej@... | Seattle | USA | 555-1112 |
| 1003 | Wells | Steve | stevew:@... | Montreal | Canada | 555-2222 |
| 1004 | Kumar | Santosh | kumars@... | London | UK | 555-3333 |

## 2. DROP statement: -

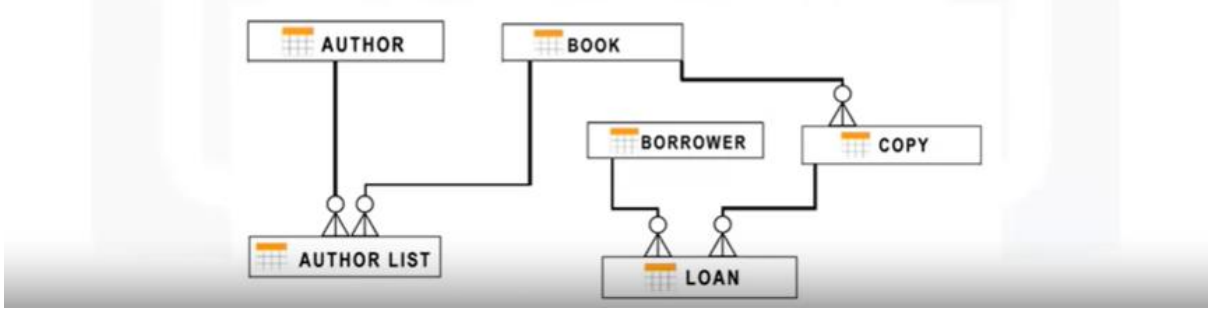
- To delete a table from a database.
- If you delete a table that contains data, by default the data will be deleted alongside the table.

**Syntax –**

```
DROP TABLE <table_name>;
```

**Example –**

```
DROP TABLE author;
```




## 3. TRUNCATE statement: -

- To delete all of the rows in a table.

**Syntax –**

```
TRUNCATE TABLE <table_name>
    IMMEDIATE;
```

- IMMEDIATE specifies to process the statement immediately and that it cannot be undone.

**Example –**

```
TRUNCATE TABLE author
  IMMEDIATE;
```

| author_id | lastname | firstname | email | city | country |
|---|---|---|---|---|---|
| 1001 | Thomas | John | johnt@... | New York | USA |
| 1002 | James | Alice | alicej@... | Seattle | USA |
| 1003 | Wells | Steve | stevew:@... | Montreal | Canada |
| 1004 | Kumar | Santosh | kumars@... | London | UK |

**Summary: -**

- The ALTER TABLE statement changes the structure of an existing table, for example, to add, modify, or drop columns.
- The DROP TABLE statement deletes an existing table.
- The TRUNCATE TABLE statement deletes all rows of data in a table.

## *5] Data Movement Utilities*

**Scenarios for Data Movement: -**

- Data engineers and database administrators will often need to move data into an out of an existing database. This could be for several reasons such as:
  - Initially populate the entire database.
  - Create a development and testing copy
  - Create a snapshot for disaster recovery
  - Create new table using data from external source/file
  - Add or append data in existing table

**Data Movement Tools & Utilities: -**

- Multiple methods and files types supported.
- 3 main categories of Data Movement Tools:
  - Backup & Restore
  - Import & Export
  - Load

**Backup & Restore: -**

- Backup creates a file for the entire database.
- Restore creates exact copy of the database from file.
- Both preserves all database objects and their data:
  - Schemas, Tables, Views
  - User Defined Types, Functions, Stored Procedures
  - Constraints, Triggers, Security
- Backups performed periodically for disaster recovery
- Create copies of database for development and test.

**Import & Export: -**

- Import inserts data from a file into a table.
- Export saves table data into a file.
- Operations available using different interfaces:
  - Command Line, API, GUI, Third Party Tools

Import/Export file formats –

- **DEL –** Delimited ASCII, for data exchange among a wide variety of database managers and file managers, include CSV.
- **ASC –** Non-delimited ASCII, for importing or loading data from other applications that create flat text files with aligned column data.
- **PC/IXF –** PC version of the Integration Exchange Format (IXF), a structured description of a database table that contains an external representation of the internal table.
- **JSON –** Lately, some databases have also started to support JSON as a format for importing and exporting data.

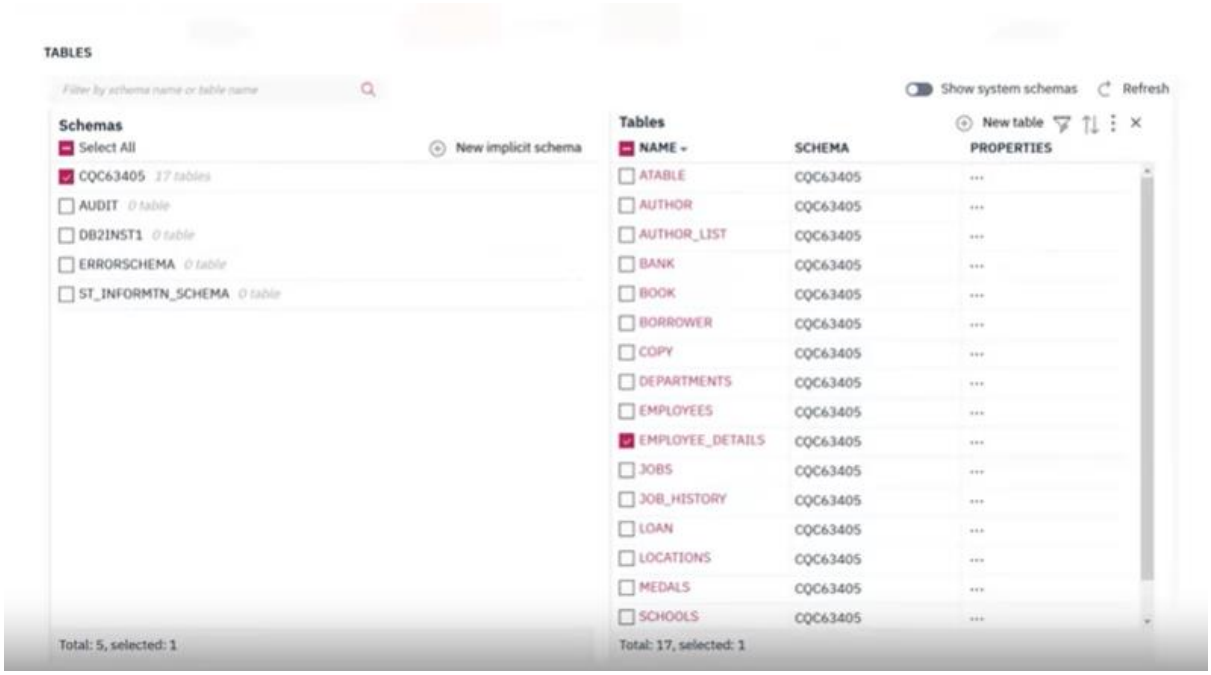
Import/Export Examples –

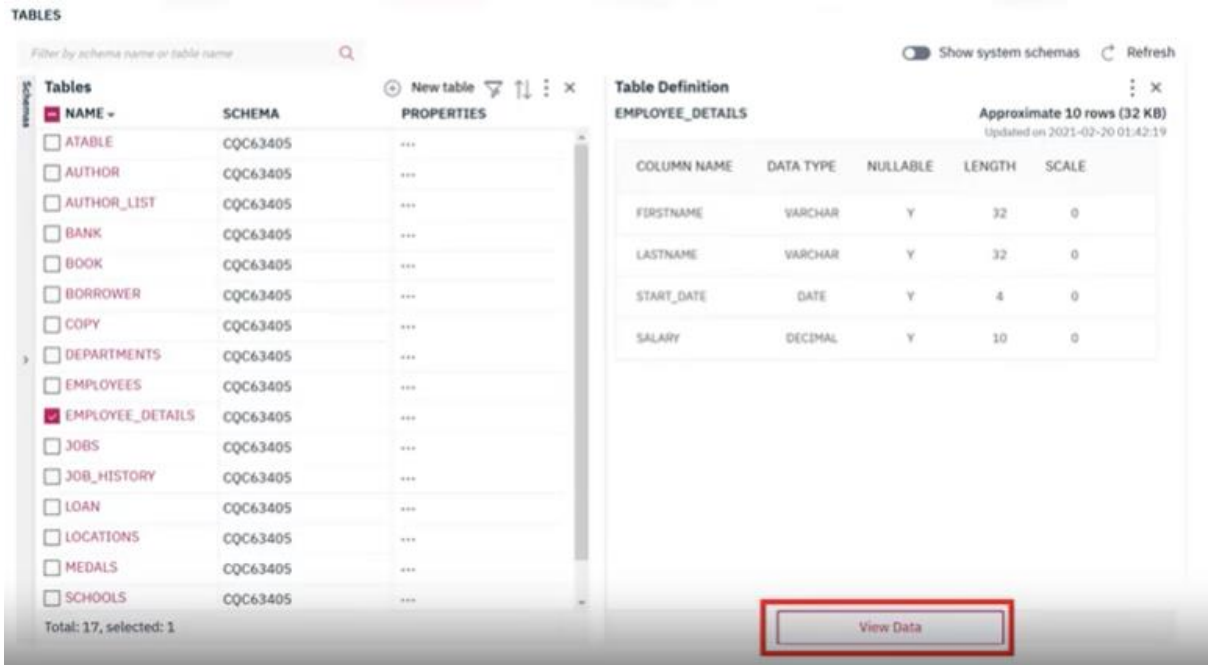
- **Db2 Command line:**

```
db2 import from filename of fileformat messages messagesfile into table
```

```
db2 export to filename of fileformat messages messagesfile select * from table
```

- **Export Example – Visual Tool:**

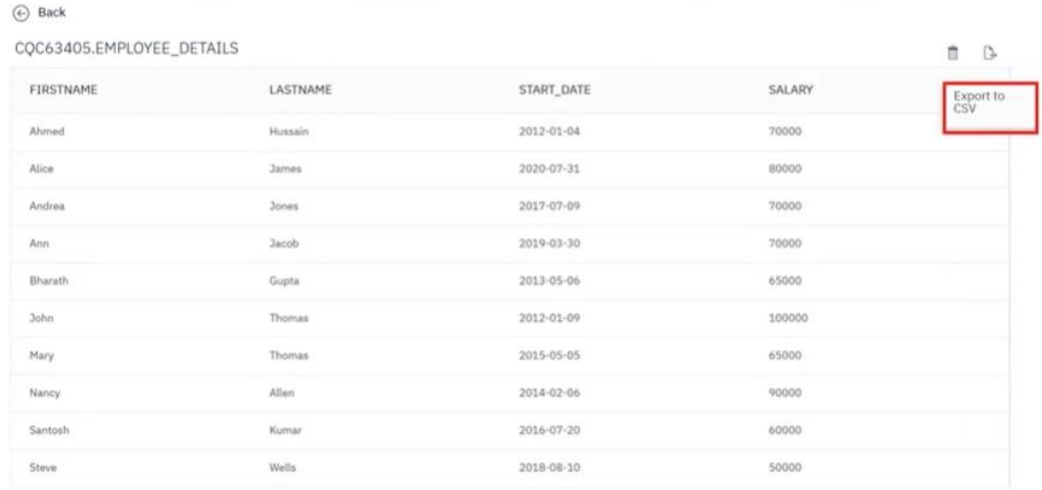


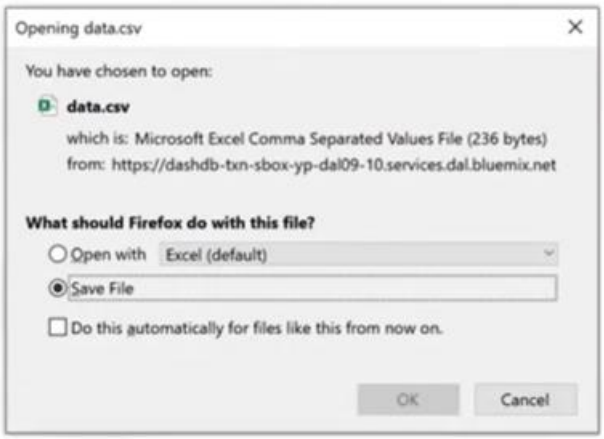



Back

CQC63405.EMPLOYEE_DETAILS

| FIRSTNAME | LASTNAME | START_DATE | SALARY |
|---|---|---|---|
| Ahmed | Hussain | 2012-01-04 | 70000 |
| Alice | James | 2020-07-31 | 80000 |
| Andrea | Jones | 2017-07-09 | 70000 |
| Ann | Jacob | 2019-03-30 | 70000 |
| Bharath | Gupta | 2013-05-06 | 65000 |
| John | Thomas | 2012-01-09 | 100000 |
| Mary | Thomas | 2015-05-05 | 65000 |
| Nancy | Allen | 2014-02-06 | 90000 |
| Santosh | Kumar | 2016-07-20 | 60000 |
| Steve | Wells | 2018-08-10 | 50000 |



**Load utilities: -**

- Supports XML, large objects, and user-defined types
- Faster than the import utility
- Doesn't perform as many checks
- Preferred option for loading very large datasets
- Initiate from command line / API / Visual Tool
- **Command line syntax:**

```
db2 load from filename of fileformat messages messagesfile import_mode
into table copy yes|no use tsm data buffer pages
```

**Summary: -**

- Data movement is required for initially populating databases and tables, adding or appending data, and making copies for development test or disaster recovery.
- BACKUP & RESTORE utilities are used to create and recover copies of entire databases including all objects like tables, views, constraints and their data.
- IMPORT utility enables inserting data into a specific table from different formats such as DEL/CSV, ASC, IXF.
- EXPORT utility enables saving data from a specific table into various formats like CSV.
- LOAD utilities enable high performance insertion of data into specified tables and useful for large volumes.

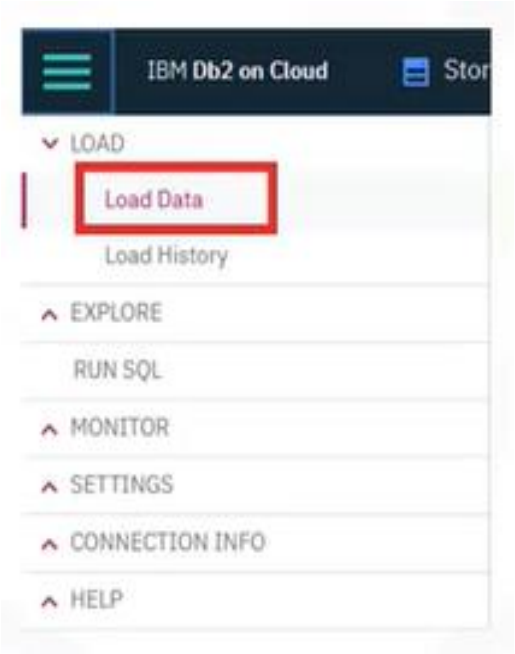
## *6] Loading Data*

- Most relational database management systems, or RDBMS, provide a method of directly loading large amounts of data into tables in a quick, efficient, and scalable manner.

**Sources of data: -**

- You can load data:
  - From different data sources
  - In different formats
- Db2 Web Console Load utility supports:
  - Delimited text files on local computer
  - S3 object storage
  - Cloud object storage

**Using the Load Data utility: -**

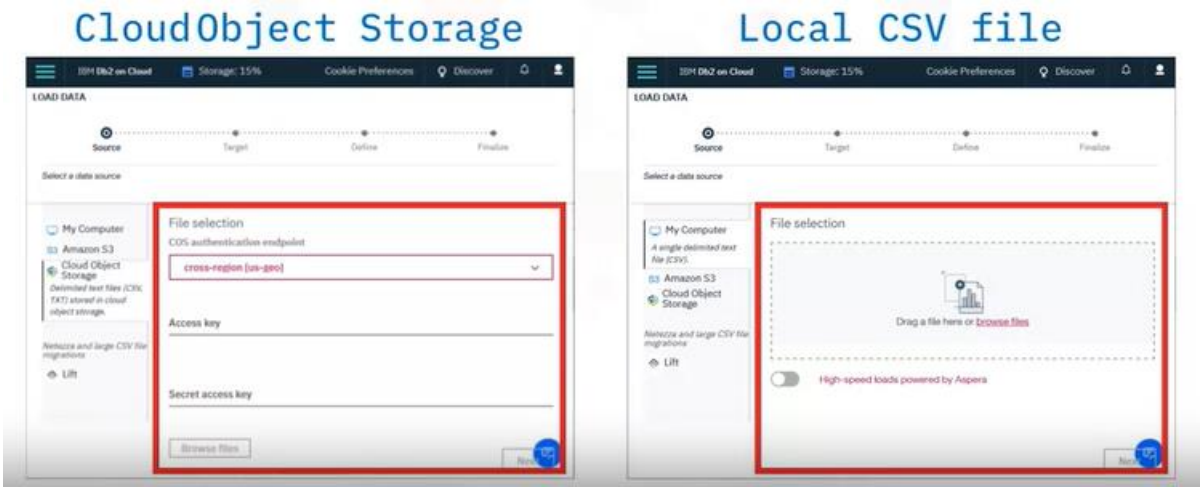



Four steps to loading data:

1. Source
2. Target
3. Define
4. Finalize

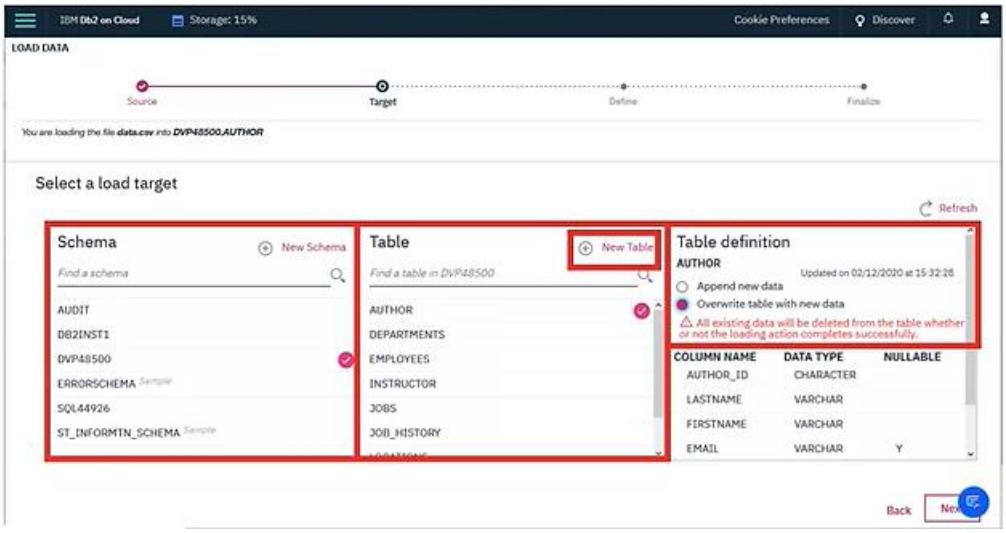
**Source –**

- On the Source page, select the location of the existing data and enter any authentication information which that storage type requires.
- For example, for data in IBM Cloud Object Storage, you provide the COS authentication endpoint, access key, and secret access key. Whereas for a locally stored CSV file, you simply identify the file to upload.



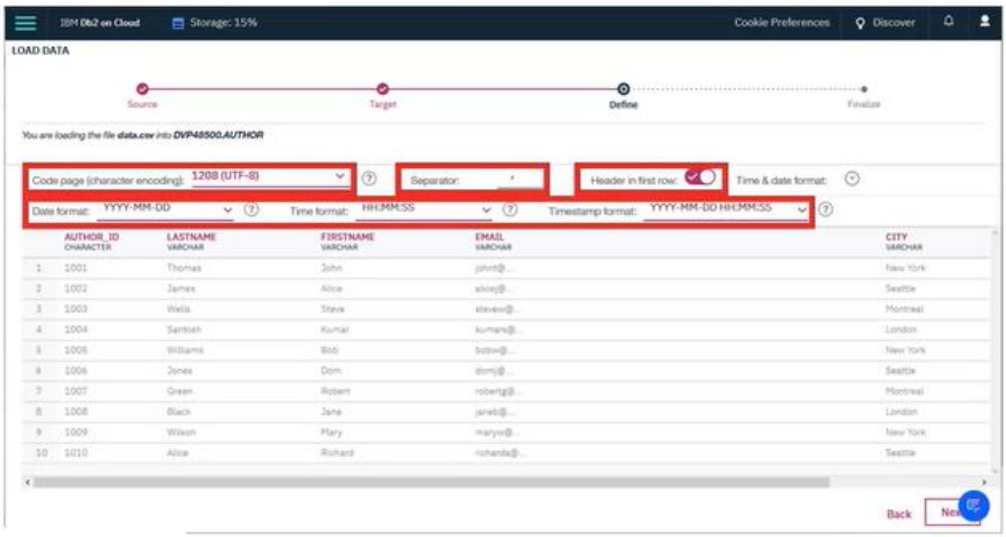

**Target –**

- On the Target page, select the target schema and table for the data location. You can also specify here whether to append the new data to the end of the table or overwrite the existing data with the new data. Note that if you use the overwrite option, even if your load fails, you will still lose all of the existing data in that table. Alternatively, you can click New Table to load the data into an entirely new table, using the format of the new data to define the data types for the columns.



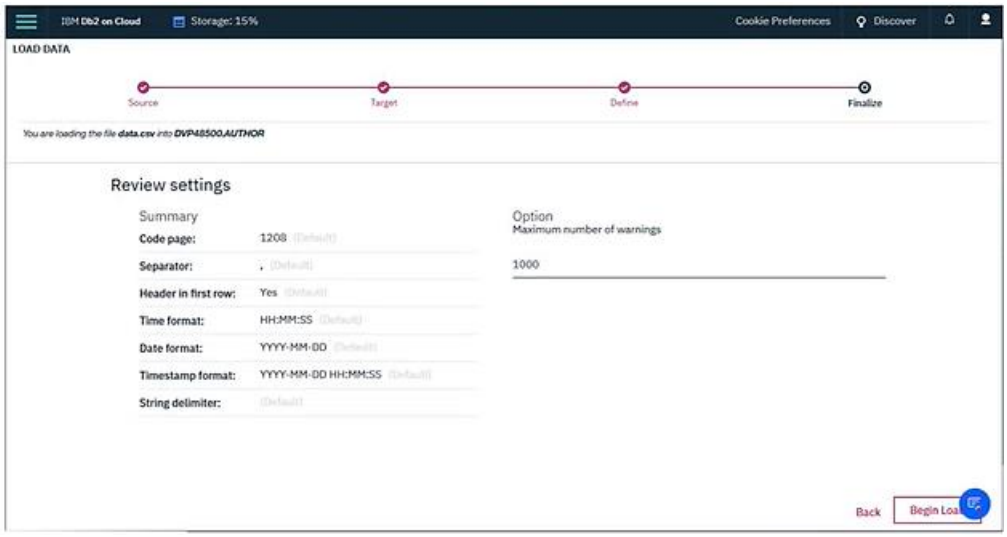

**Define –**

- On the Define page, you define the character encoding and delimiter for the text file, whether there are column headings in the first row, and what time and date formats to use.



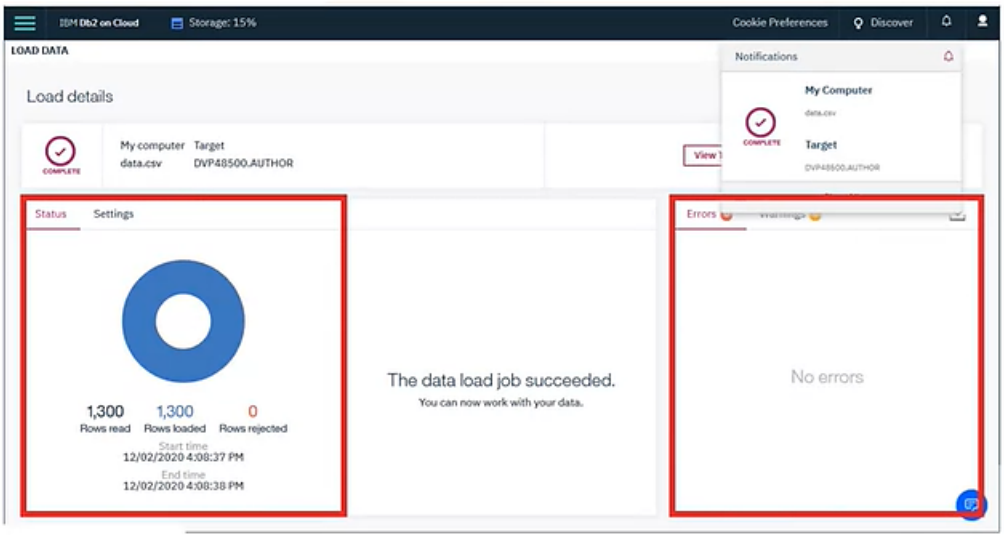

**Finalize –**

- And on the Finalize page, you can review all the settings before beginning to load the data.




**Completion –**

- When the load completes, the console displays the status of the process and also shows any errors or warnings that the load process has raised.




**<u>Summary: -</u>**

- Loading data is quicker, more efficient, and more scalable that using multiple INSERT statements
- You can load data from a variety of data sources, including delimited text files and Cloud Object Storage
- The Load Data utility is a simple to use interface in the Db2 Web Console

## ***Summary & Highlights:-***

- DDL statements, including CREATE, ALTER, TRUNCATE, and DROP, are used for defining objects like tables in a database.
- DML statements, including INSERT, SELECT, UPDATE, and DELETE, are used for manipulating data in tables.
- Many Relational Database Management Systems (RDBMS) have schemas that contain tables, views, functions, and other database objects.
- Most RDBMS provide a GUI through which you can create and alter the structure of tables.

- You can also create and alter tables by using DDL SQL statements:
  - **CREATE TABLE –** Creates entities (tables) in a relational database and sets the attributes (columns) in a table, including the names of columns, the data types of columns, and constraints (for example, the Primary Key.)
  - **ALTER TABLE –** Changes the structure of a table by adding or removing columns, modifying the data type of columns, and adding or removing keys and constraints.
  - **DROP TABLE –** Deletes a table from a database.
  - **TRUNCATE TABLE -** Removes all rows in a table.

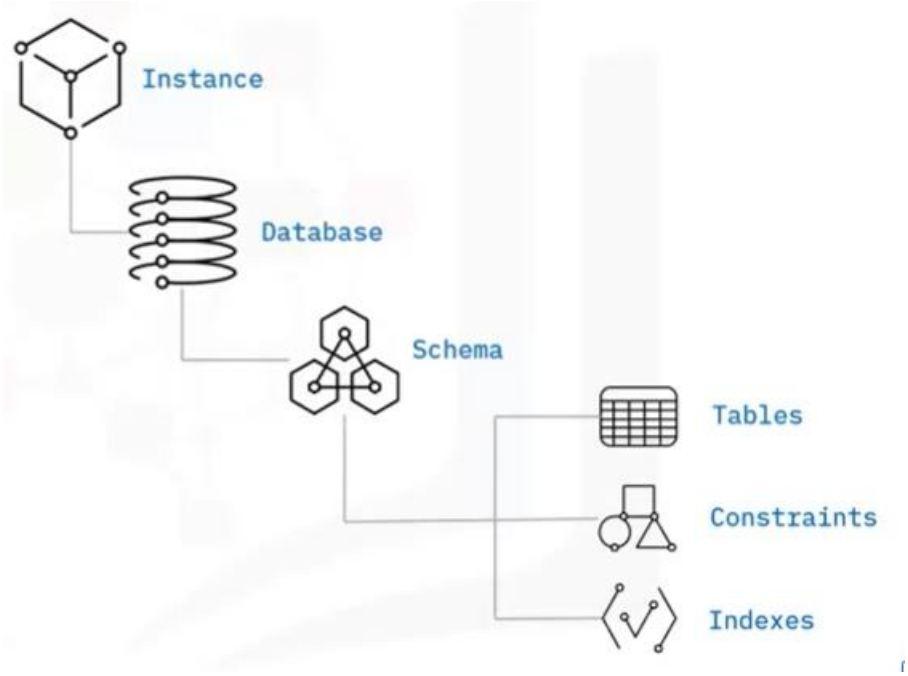
- There are utilities that help you to manage the movement of data:
  - You use the **BACKUP and RESTORE** utilities to create and recover copies of entire databases, including all objects like tables, views, constraints, and data.
  - You use the **IMPORT** utility to insert data into a specific table from different formats, such as DEL/CSV, ASC and IXF, and the **EXPORT** utility to save data from a specific table into various formats, such as CSV.
  - You can use the **LOAD** utilities, instead of INSERT statements, to quickly insert large amounts of data a variety of different data sources into tables.
  - The Load Data utility is a simple to use interface in the Db2 Web Console.

# Module 2: - Designing Keys, Indexes, and Constraints

## 1] Database Objects & Hierarchy (Including Schemas)

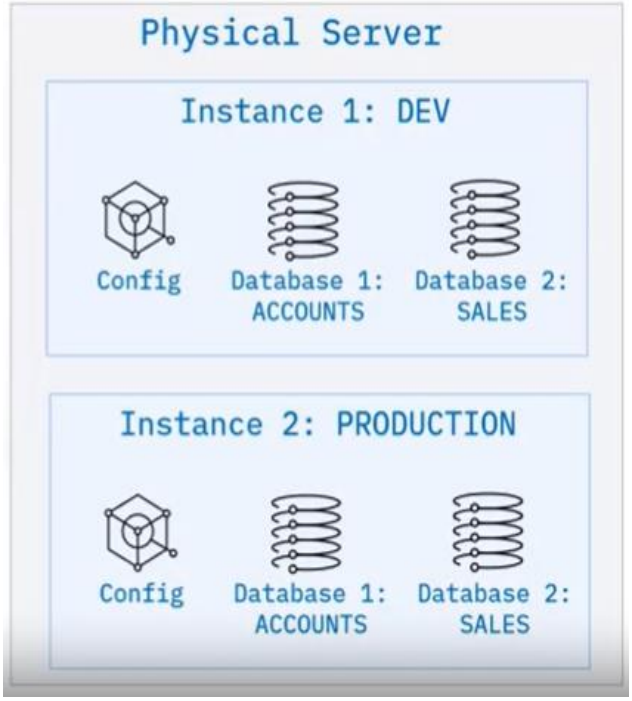
**Database Hierarchy: -**

- Instance
- Database
- Schema
- Database objects
  - Tables
  - Constraints
  - Indexes
- Other database objects in a hierarchical structure allows database administrators to manage security, maintenance, and accessibility.




- Most RDBMSes begin with an instance, a single way of organizing the database and everything it contains. Many RDBMSes permit more than one database within a single instance. You will generally find at least one schema at some level in the hierarchy. A schema is a logical grouping of objects within a database. Schemas define how database objects are named and prevent ambiguous references. Some RDBMSs consider the schema a parent object of a database, and others consider it a database object. Within a schema are the database objects, including tables, constraints, and indexes.

**Instances –**




- A logical boundary for a database or set of databases where you organize database objects and set configuration parameters.
- Every database within an instance is assigned a unique name, has its own set of system catalog tables (which keep track of the objects within the database) and has its own configuration files.
- You can create more than one instance on the same physical server
- provides a unique database server environment for each instance.
- The databases and other objects within one instance are isolated from those in any other instance.
- You can use multiple instances when you want to use one instance for a development environment and another instance for a production environment, restrict access to sensitive information, or control high-level administrative access.
- Not all RDBMSes use the concept of instances, often managing database configuration information in a special database instead.
- In Cloud-based RDBMSes, the term instance means a specific running copy of a service.

**Relational databases –**

- Set of objects used to store, manage, and access data.
- Includes tables, views, indexes, functions, triggers, and packages.
- Database objects can be either defined by the system (built-in objects) or defined by the user (user-defined objects).
- Database Engineers establish relationships between tables to reduce redundant data and improve data integrity.
- A distributed relational database shares tables and other objects across different but interconnected computer systems.

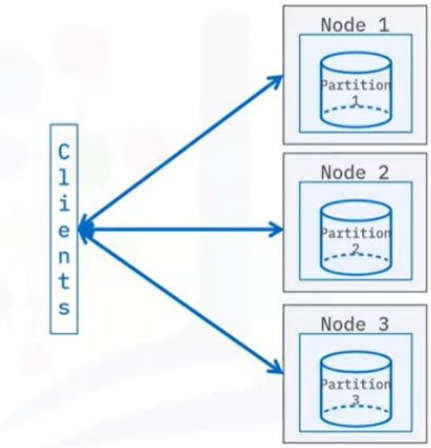
**Schemas –**

- Specialized database object that provides a way to group other database objects logically.
- Can contain tables, views, nicknames, triggers, functions, packages, and other objects.
- When you create a database object, you can assign it to a schema.

- If you want to assign the object to a specific schema, you can explicitly include the schema name.
- If you don't include the schema name, the object is implicitly assigned to the current schema.
- Default schema is the user schema for the currently logged-on user.
- Provides a naming context. Using the schema name as a name qualifier enables you to distinguish between objects with the same name in different schemas.
- For example, the schema names Internal and External make it easy to distinguish two different SALES tables: INTERNAL.SALES in the internal schema and EXTERNAL.SALES in the External schema. Thus, schemas enable multiple applications to store data in a single database without encountering namespace conflicts.
- Many RDBMSes use a specialized schema to hold configuration information and metadata about a particular database. For example, tables in a system schema can store lists of database users and their access permissions, information about the indexes on tables, details of any database partitions that exist, and user-defined data types.

**Database partitions –**

- Data is managed across multiple partitions.
- Split tables that contain very large quantities of data.
- Hold a subset of the data
- Common in data warehousing & data analysis for business intelligence.




**Database objects –**

- Physical database design consists of defining database objects.
- Common database objects:
  - Tables
  - Constraints
  - Indexes
  - Views
  - Aliases
- Create and manage using Graphical tools, scripting, APIs, SQL.

**Summary: -**

- An instance is a logical boundary for a database or set of databases where you organize database objects and set configuration parameters.
- A relational database is a set of objects used to store, manage, and access data. Relationships between tables to reduce redundant data and improve data integrity.
- A schema is a specialized database object that provides a way to logically group tables, views, nicknames, triggers, functions, packages, and other database objects. Schemas provide naming contexts so that you can distinguish between objects with the same name.
- User schemas contain database objects like tables, views, functions.
- System schemas contain configuration information and metadata for the database.
- You can split very large tables across multiple partitions to improve performance.
- Database objects are the items that exist within the database, such as tables, constraints, indexes, views, and aliases.

## *2] Primary Keys and Foreign Keys*

**What is Primary Key?**

- In some tables, the choice of primary key is easy because it is a naturally occurring unique attribute of entity.
  For example, the book id of a book or the employee ID number of a staff member.
- If your table doesn't have an existing unique attribute, you can add a column to the table to serve as the primary key. Or if a combination of two attributes uniquely identifies each row, you can create a primary key across the two columns.
  For example, where employees have a unique identifier within their work site, you can use the combination of their site id and employee id.
- Each table can only have one primary key.
- Note to use a comma to separate the column names when using multiple columns for a primary key.

**Syntax with Example –**

- You can create a primary key when you create the table by using the PRIMARY KEY clause of the CREATE TABLE statement. In the parenthesis for the PRIMARY KEY clause, state the name of the column or columns that will be the primary key.

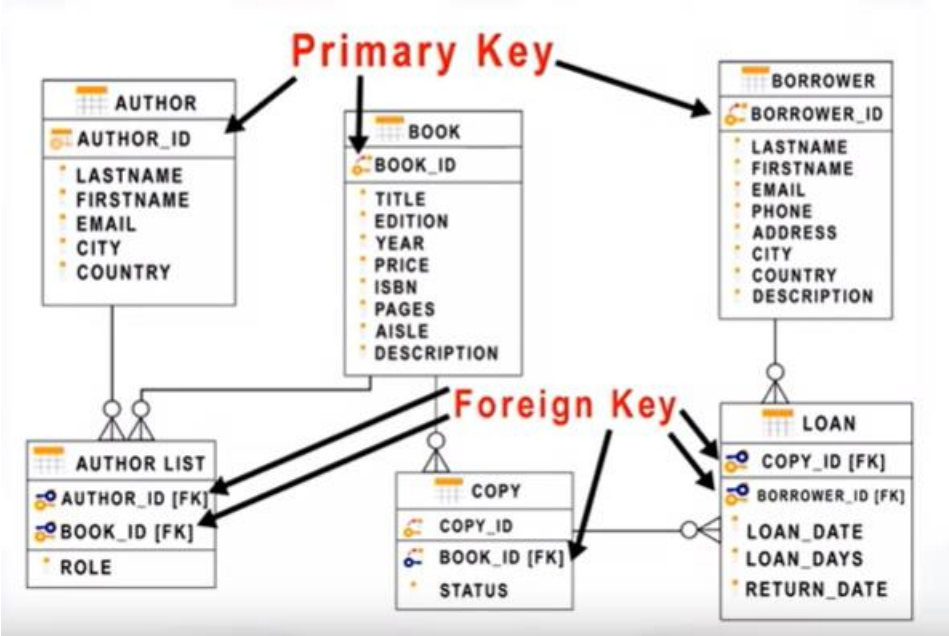
```
CREATE TABLE book(
   book_id INT NOT NULL,
   . . .
   pub_id INT NULL,
   PRIMARY KEY(book_id));
```

- Alternatively, you can create a primary key on an existing table by using the ADD PRIMARY KEY clause of the ALTER TABLE statement. And again in the parenthesis, state the name of the column or columns that will be the primary key.

```
ALTER TABLE book
   ADD PRIMARY KEY(book_id, ISBN);
```

**What is Foreign Key?**

- You use primary and foreign keys to define the relationships between your tables.
- A foreign key is a column in a table which contains the same information as the primary key in another table.




- For example, the Copy table might list all books that the library owns. Therefore, the book_id of a copy of an individual book must exist in the Book table as a valid book.

Where the library owns multiple copies of a popular book, the book_id of that particular book will appear multiple times in the Copy table. You can also specify that whenever you add a row to the Copy table, the book_id you use must already exist in the Book table.

**Syntax with Example –**

```
CREATE TABLE copy(
   copy_id INT NOT NULL,
   book_id INT NULL,
   . . .
   CONSTRAINT fk_copy_book FOREIGN KEY(book_id)
     REFERENCES book(book_id)
     );
```

- You can also use the rule clause to define what action to take if a row in the parent table, that is the table with the primary key, is updated or deleted. For updates and deletes, you can specify to take no action in which case the update or delete operation on the parent table may fail.

```
CREATE TABLE copy(
   copy_id INT NOT NULL,
   book_id INT NULL,
   . . .
   CONSTRAINT fk_copy_book FOREIGN KEY(book_id)
     REFERENCES book(book_id)
     ON UPDATE NO ACTION);
```

- For deletions, you can also specify to cascade the delete, that is to delete the related child rows in the dependent table, or to set the values in the foreign key column of the child table to null.

```
CREATE TABLE copy(
  copy_id INT NOT NULL,
  book_id INT NULL,
  . . .
  CONSTRAINT fk_copy_book FOREIGN KEY(book_id)
    REFERENCES book(book_id)
    ON DELETE CASCADE);
```

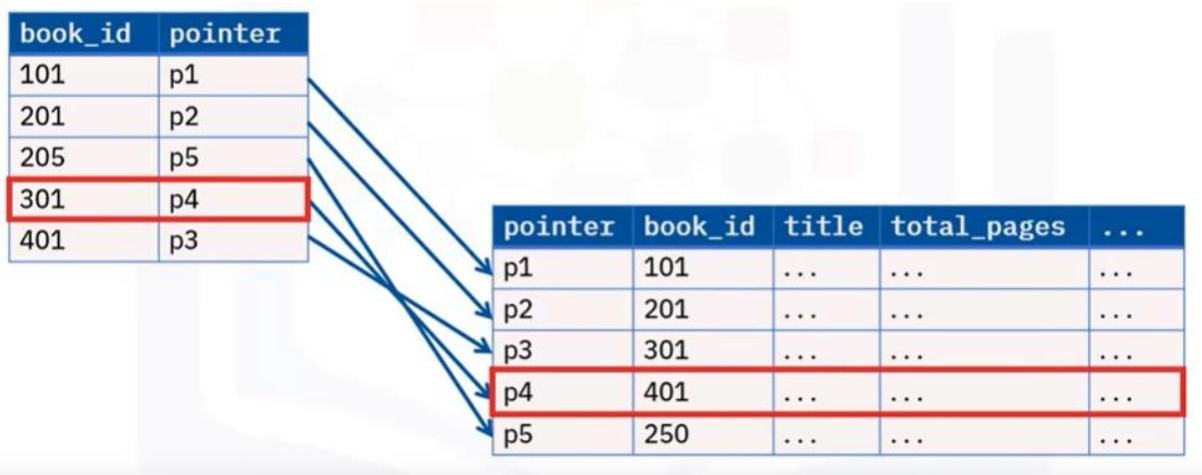
**Summary: -**

- You can use primary keys to enforce uniqueness of rows in a table.
- Foreign keys are column in a table which contain the same information as the primary key in another table.
- You can use primary and foreign keys to create relationships between tables.

## *3] Overview of Indexes*

**What is an Index?**

- Usually, when you add data to a table it is appended to the end of the table, however, there is no guarantee of this and there is no inherent order to the data. So, when you select a particular row from that table, the processor must check each row in turn until it finds the one that you want. On a large table, this can become a very slow way of locating a row. Also, when you select multiple rows, unless you specify a sort order in your SELECT statement they may be returned in an unordered state. Because you often want to return the rows in a particular order or select a subset of sequential rows, you can create an index on a table to easily locate the specific row or set of rows you require.
- An index works by storing pointers to each row in the table so when you request a particular row, the SQL processor can use the index to quickly locate the row. This is similar to how you use the index in a book to quickly find a particular section of the book. The index is ordered by values within the unique key upon which it is based.
- If you create many indexes on a table, you can actually negate the performance benefits in the same way that indexing every word in a book would result in an unhelpful index.




**Syntax with Example –**

```
CREATE TABLE book(
  . . .
  PRIMARY KEY(book_id));
```

```
CREATE UNIQUE INDEX unique_book_id
  ON book(book_id);
```

**Advantages:**

- Improved performance of SELECT queries.
- Reduced need to sort data.
- Guaranteed uniqueness of rows.

**Disadvantages:**

- Use disk space
- Decreased performance of INSERT, UPDATE, and DELETE queries.

**Summary: -**

- Indexes provide ordered pointers to rows in tables.
- Indexes can improve performance of SELECT queries.

- Indexes can decrease performance of INSERT, UPDATE, and DELETE queries.

## *4] Normalization*

**What is Normalization?**

- Data duplication leads to inconsistencies.
- Normalization reduces data duplication
- Increases speed of transactions
- Improves the integrity of data
- Normalize each table
- Most used:
  - First Normal Form
  - Second Normal Form
  - Third Normal Form

**First Normal Form: -**

- Each row must be unique
- Each cell must contain only a single value
- Also called 1NF

**Example –**

- In this example, the Book table contains some basic information about books, including title, formats, and authors. To meet first normal form requirements, each cell must contain a single value, not a list.

| Book_id | Title | Format | Author_name |
|---|---|---|---|
| 101 | Lean Software Development | Paperback | Mary Poppendieck |
| 201 | Facing the Intelligence Explosion | Paperback | David Robson |
| 301 | Scala in Action | Hardback | Yehuda Katz |
| 401 | Patterns of Software | Hardback, Paperback | Mary Poppendieck |
| 501 | Anatomy of LISP | Paperback | Eric Redmond |

- In this example, you can see that all formats of a book are listed in the same cell.

| Book_id | Title | Format | Author_name |
|---|---|---|---|
| 101 | Lean Software Development | Paperback | Mary Poppendieck |
| 201 | Facing the Intelligence Explosion | Paperback | David Robson |
| 301 | Scala in Action | Hardback | Yehuda Katz |
| 401 | Patterns of Software | Hardback, Paperback | Mary Poppendieck |
| 501 | Anatomy of LISP | Paperback | Eric Redmond |

- To normalize this table, you can add an extra row, and split the two formats of Patterns of Software into their own row. So now you have a row for the paperback version, and a row for the hardback version. Each cell in the table now has only one entry, and .so the table is in first normal form.

| Book_id | Title | Format | Author_name |
|---|---|---|---|
| 101 | Lean Software Development | Paperback | Mary Poppendieck |
| 201 | Facing the Intelligence Explosion | Paperback | David Robson |
| 301 | Scala in Action | Hardback | Yehuda Katz |
| 401 | Patterns of Software | Paperback | Mary Poppendieck |
| 401 | Patterns of Software | Hardback | Mary Poppendieck |
| 501 | Anatomy of LISP | Paperback | Eric Redmond |

**Second Normal Form: -**

- Database must be in first normal form.
- Create separate tables for sets of values that apply to multiple rows.
- Also called as 2NF.

**Example –**

- For clarity, this example shows just a subset of the data in the book table.

| Book_id | Title | Format | Author_name |
|---|---|---|---|
| 101 | Lean Software Development | Paperback | Mary Poppendieck |
| 201 | Facing the Intelligence Explosion | Paperback | David Robson |
| 301 | Scala in Action | Hardback | Yehuda Katz |
| 401 | Patterns of Software | Paperback | Mary Poppendieck |
| 401 | Patterns of Software | Hardback | Mary Poppendieck |
| 501 | Anatomy of LISP | Paperback | Eric Redmond |

- Book 401 comes in both paperback and hardback format, so in the current form it must be listed twice, once for each format.

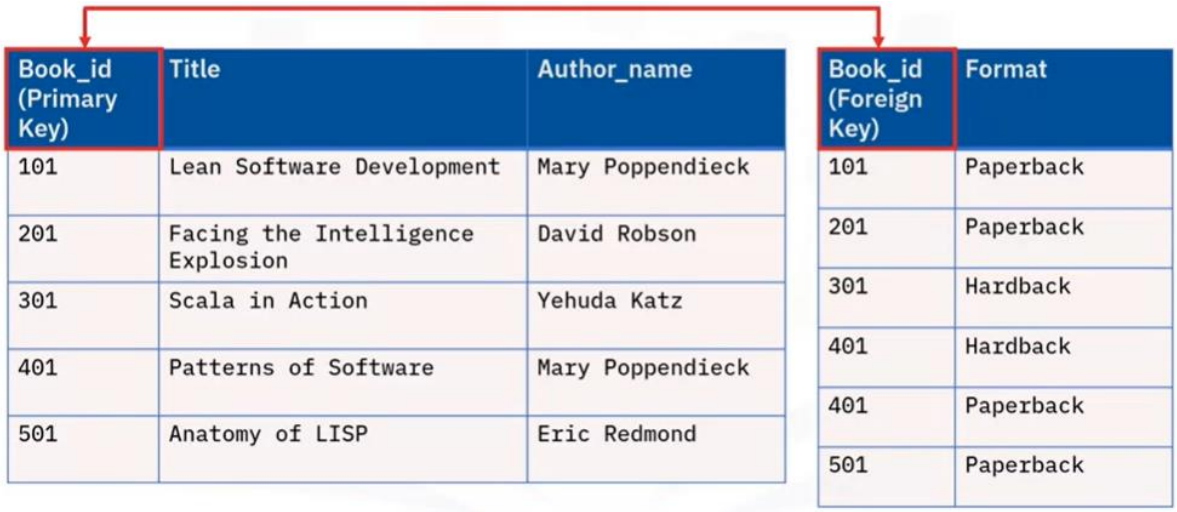
| Book_id | Title | Format | Author_name |
|---|---|---|---|
| 101 | Lean Software Development | Paperback | Mary Poppendieck |
| 201 | Facing the Intelligence Explosion | Paperback | David Robson |
| 301 | Scala in Action | Hardback | Yehuda Katz |
| 401 | Patterns of Software | Paperback | Mary Poppendieck |
| 401 | Patterns of Software | Hardback | Mary Poppendieck |
| 501 | Anatomy of LISP | Paperback | Eric Redmond |

- In this case the Format column contains values that apply to both rows that reference Book 401, and so there is some data duplication. To meet the requirements for second normal form, and achieve just one row for Book 401, you can split the Book table, so that the format information for the book is separated from unrelated information such as title and author. Each resulting table is in 1NF. To maintain a relationship between the two tables, identify a Primary Key for one table that will be used a foreign key in the other.

| Book_id (Primary Key) | Title | Author_name |
|---|---|---|
| 101 | Lean Software Development | Mary Poppendieck |
| 201 | Facing the Intelligence Explosion | David Robson |
| 301 | Scala in Action | Yehuda Katz |
| 401 | Patterns of Software | Mary Poppendieck |
| 501 | Anatomy of LISP | Eric Redmond |

| Book_id (Foreign Key) | Format |
|---|---|
| 101 | Paperback |
| 201 | Paperback |
| 301 | Hardback |
| 401 | Hardback |
| 401 | Paperback |
| 501 | Paperback |

- In this example, the Book_id is unique to each book, so you can make that the Primary Key in the book table and include it as a Foreign Key in the Format table. Now you can use it to link between the two tables to find the different formats of each of the unique books.



| Book_id (Primary Key) | Title | Author_name |
|---|---|---|
| 101 | Lean Software Development | Mary Poppendieck |
| 201 | Facing the Intelligence Explosion | David Robson |
| 301 | Scala in Action | Yehuda Katz |
| 401 | Patterns of Software | Mary Poppendieck |
| 501 | Anatomy of LISP | Eric Redmond |

| Book_id (Foreign Key) | Format |
|---|---|
| 101 | Paperback |
| 201 | Paperback |
| 301 | Hardback |
| 401 | Hardback |
| 401 | Paperback |
| 501 | Paperback |

**Third Normal Form: -**

- Database must be in first and second normal form.
- Eliminate columns that do not depend on the key.
- Also called 3NF.

**Example –**

- Let's consider some additional data about books – the publisher and where the book ships from. Each publisher ships books from warehouses in their own location, so where the book ships from depends on the publisher, not the Book_id.
- Therefore, the Book table is not in 3NF, because the Ships from data does not depend on the Primary Key.

| Book_id (Primary Key) | Title | Author_name | Publisher | Ships from |
|---|---|---|---|---|
| 101 | Lean Software Development | Mary Poppendieck | Tech Books | UK |
| 201 | Facing the Intelligence Explosion | David Robson | Amazing Books | US |
| 301 | Scala in Action | Yehuda Katz | Better Tech Books | India |
| 401 | Patterns of Software | Mary Poppendieck | Publisher 101 | US |
| 501 | Anatomy of LISP | Eric Redmond | Best Tech Books | Canada |

- To meet the requirements of 3NF, you must separate the Publisher and Ships from information into a Publishers table. Both tables are now in third normal form, which is as far as most relational databases go.

| Book_id (Primary Key) | Title | Author_name | Pub_id |
|---|---|---|---|
| 101 | Lean Software Development | Mary Poppendieck | 1 |
| 201 | Facing the Intelligence Explosion | David Robson | 2 |
| 301 | Scala in Action | Yehuda Katz | 3 |
| 401 | Patterns of Software | Mary Poppendieck | 4 |
| 501 | Anatomy of LISP | Eric Redmond | 5 |

| Pub_id (Primary Key) | Publisher | Ships from |
|---|---|---|
| 1 | Tech Books | UK |
| 2 | Amazing Books | US |
| 3 | Better Tech Books | India |
| 4 | Publisher 101 | US |
| 5 | Best Tech Books | Canada |

- There are also higher normal forms, such as Boyce Codd Normal form or BCNF, which is an extension to the third normal form, as well as fourth and fifth normal forms which may be needed for specific scenarios.

**Normalization in OLTP and OLAP: -**

- **Online Transactional Processing (OLTP) –**
  - Data is read and written frequently.
  - Data is normalized to 3NF or BCNF.
- **Online Analytical Processing (OLAP) –**
  - Data is mostly read only.
  - Data is de-normalized to 2NF or 1NF.

**Summary: -**

- Normalization reduces redundancy and increases consistency of data.
- First normal form (1NF)
  - Each row must be unique
  - Each cell must contain only a single item.
- Second normal form (2NF)
  - Separate tables for sets of values that apply to multiple records.
- Third normal form (3NF)
  - Eliminate any columns that do not depend on the key.

## *5] Relational Model Constraints – Advanced*

**Constraints: -**

- Within any business, data must often adhere to certain restrictions or rules. Constraints help implement the business rules. In a relational data model, data integrity can be achieved using integrity rules or constraints.

**Six Constraints –**

1. **Entity Integrity Constraint**

   - To identify each tuple in a relation, the relation must have a primary key.
   - The primary key is a unique value that identifies each tuple (or row) in a table. This is the Entity Integrity Constraint.
   - The terms Primary Key Constraint or Unique Constraint are also used.
   - This constraint prevents duplicate values in a table.
   - To implement these constraints, indexes are used.
   - The Entity Integrity Constraint states that no attribute participating in the primary key of a relation is allowed to accept NULL values. The value NULL indicates that the value is unknown.
   - In the Entity Integrity Constraint, the primary key cannot have an unknown value.
   - With the Entity Integrity Constraint, no attribute participating in the primary key is allowed to accept NULL values.

**Example –**

- For example, in the relation Author, Author_ID is the primary key. The primary key identifies each tuple in the relation. The Author_ID A1 points to author Raul Chong from Toronto. If you replace the value A1 with NULL, you can still identify the author as Raul Chong. However, if you also replace Author_ID A4 with NULL, now you do not know which NULL value identifies which tuple.

AUTHOR

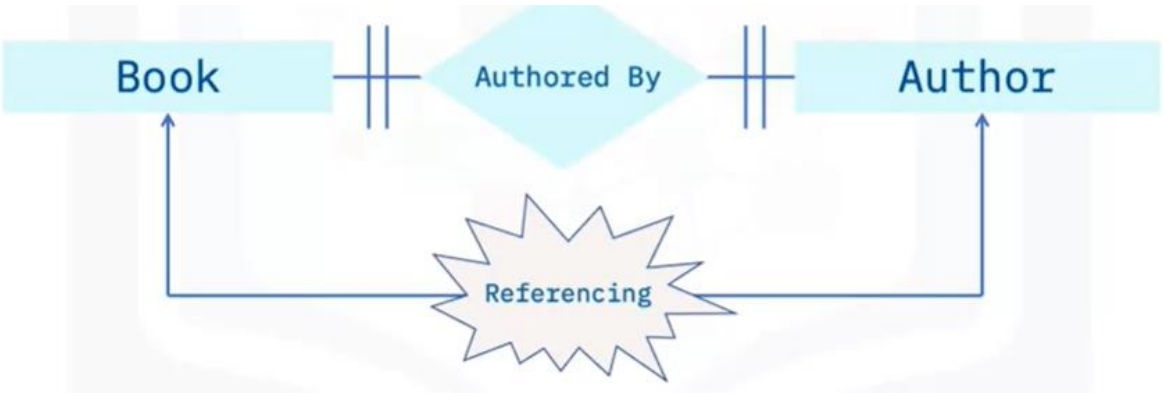
| Author_ID [PK] | Lastname | Firstname | Email | City | Country |
|---|---|---|---|---|---|
| A1 | Chong | Raul | rfc@ibm.com | Toronto | CA |
| A2 | Ahuja | Rav | ra@ibm.com | Toronto | CA |
| A3 | Hakes | Ian | ih@ibm.com | Toronto | CA |
| A4 | Sharma | Neeraj | ns@ibm.com | Chennai | IN |
| A5 | Perniu | Liviu | lp@univ.com | Transilvania | RO |

## 2. Referential Integrity Constraint

- Referential integrity constraint defines relationships between tables and ensures that these relationships remain valid. The validity of the data is enforced using a combination of Primary Keys and Foreign Keys.

**Example –**

- As mentioned previously, for a book to exist it has to be written by at least one author.



### 3. Semantic Integrity Constraint

- The Semantic Integrity Constraint refers to the correctness of the meaning of the data.

**Example –**

- For example, in the relation Author, if the attribute (or column) City contains a garbage value instead of Toronto, the garbage value does not have any meaning.

AUTHOR

| Author_ID [PK] | Lastname | Firstname | Email | City | Country |
|---|---|---|---|---|---|
| Null | Chong | Raul | rfc@ibm.com | 12(*)&^23 | CA |
| A2 | Ahuja | Rav | ra@ibm.com | Toronto | CA |
| A3 | Hakes | Ian | ih@ibm.com | Toronto | CA |
| Null | Sharma | Neeraj | ns@ibm.com | Chennai | IN |
| A5 | Perniu | Liviu | lp@univ.com | Transilvania | RO |

### 4. Domain Constraint

- A Domain Constraint specifies the permissible values for a given attribute.

**Example –**

- For example, in the relation Author, the attribute Country must contain a two-letter country code, such as CA for Canada, or IN for India. If a number value of 34 is entered for the Country attribute instead of a two-letter country code, the value 34 does not having any meaning.

AUTHOR

| Author_ID [PK] | Lastname | Firstname | Email | City | Country |
|---|---|---|---|---|---|
| A1 | Chong | Raul | rfc@ibm.com | Toronto | CA |
| A2 | Ahuja | Rav | ra@ibm.com | Toronto | 34 |
| A3 | Hakes | Ian | ih@ibm.com | Toronto | CA |
| A4 | Sharma | Neeraj | ns@ibm.com | Chennai | IN |
| A5 | Perniu | Liviu | lp@univ.com | Transilvania | RO |

## 5. Null Constraint

- The NULL constraint specifies that attribute values cannot be null.

**Example –**

- For example, in the relation Author, if either LastName or FirstName contains a NULL value, it could be difficult to identify the correct author. In this example, FirstName and LastName attribute values cannot be NULL. An author must have a name.

AUTHOR

| Author_ID [PK] | Lastname | Firstname | Email | City | Country |
|---|---|---|---|---|---|
| A1 | Chong | NULL | rfc@ibm.com | Toronto | CA |
| A2 | Ahuja | Rav | ra@ibm.com | Toronto | CA |
| A3 | NULL | Ian | ih@ibm.com | Toronto | CA |
| A4 | Sharma | Neeraj | ns@ibm.com | Chennai | IN |
| A5 | Perniu | Liviu | lp@univ.com | Transilvania | RO |

## 6. Check Constraint

- The CHECK constraint enforces domain integrity by limiting the values that are accepted by an attribute.

**Example –**

- In the relation Book, the attribute Year is the year in which a particular book is published. If this was still the year 2010, it would not be meaningful to have a year greater than the current year.
- The CHECK constraint would enforce the domain integrity by limiting the values that are accepted by the attribute Year.

## Check Constraint

| Title | Edition | Year | Price | ISBN | Pages | Aisle | Description |
|---|---|---|---|---|---|---|---|
| Database Fundamentals | 1 | 2010 | 24.99 | 978-0-9866283-1-1 | 300 | DB-A02 | Teaches you the fundamentals of databases |
| Getting started with DB2 Express-C | 1 | 2010 | 24.99 | 978-0-9866283-5-1 | 280 | DB-A01 | Teaches you the essentials of DB2 using DB2 Express-C, the free version of DB2 |

**Summary: -**

- Entity Integrity Constraint ensures primary key is a unique value that identifies each tuple (or row).
- Referential integrity constraint defines relationships between tables.
- Semantic Integrity Constraint refers to the correctness of the meaning of the data.
- Domain Constraint specifies the permissible values for a given attribute.
- Null Constraint specifies that attribute values cannot be null.
- Check Constraint limits the values that are accepted by an attribute.

## *Summary & Highlights:-*

The objects in a Relational Database Management System (RDBMS) object hierarchy include:

- **Instances -** This is a logical boundary for a database or set of databases where you organize and isolate database objects and set configuration parameters.
- **Relational databases -** This is a set of objects used to store, manage, and access data.
- **Schemas -** A user or system schema is a logical grouping of tables, views, nicknames, triggers, functions, packages, and other database objects. Schemas provide naming contexts so that you can distinguish between objects with the same name.
- **Database partitions -** You can split very large tables across multiple partitions to improve performance.
- **Database objects -** Database objects are the items that exist within the database, such as tables, constraints, indexes, views, and aliases.

Primary key and Foreign Keys have several uses:

- Primary keys enforce uniqueness of rows in a table, whereas Foreign keys are columns in a table that contain the same information as the primary key in another table.
- You can use primary and foreign keys to create relationships between tables. Relationships between tables reduce redundant data and improve data integrity.
- Indexes provide ordered pointers to rows in tables and can improve the performance of SELECT queries, but can decrease the performance of INSERT, UPDATE, and DELETE queries.

Normalization reduces redundancy and increases consistency of data. There are two forms of normalization:

- **First normal form (1NF) -** In this form, the table contains only single values and has no repeating groups.
- **Second normal form (2NF) -** This form splits data into multiple tables to reduce redundancy.

You can define six relational model constraints:

- **Entity integrity constraint -** Ensures that the primary key is a unique value that identifies each tuple (or row.)
- **Referential integrity constraint -** Defines relationships between tables.
- **Semantic integrity constraint -** Refers to the correctness of the meaning of the data.
- **Domain constraint -** Specifies the permissible values for a given attribute.
- **Null constraint -** Specifies that attribute values cannot be null.
- **Check constraint -** Limits the values that are accepted by an attribute.

# Week 3

## MySQL and PostgreSQL

## Module 1: - MySQL

## 1] Getting Started with MySQL

**MySQL RDBMS: -**

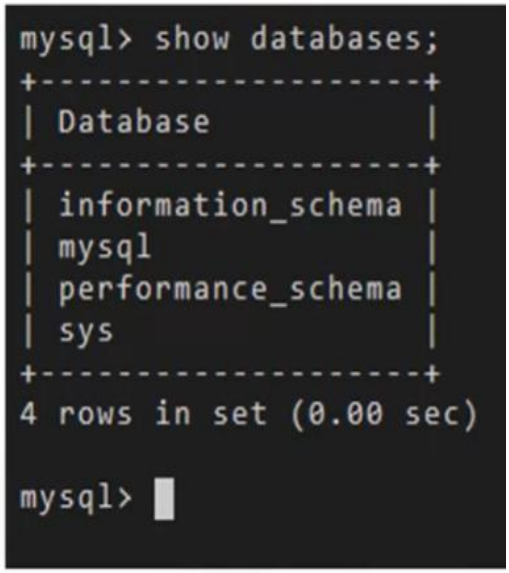
- MySQL is a popular open-source relational database management system, or RDBMS.
- MariaDB is a fork of the MySQL database by some of the original developers of MySQL.

**MySQL Options: -**

- Download and install using:
  - Community Edition
  - Commercial Editions
- Cloud:
  - VM images and containers.
  - Managed service – such as IBM Cloud, Amazon RDS for MySQL, Azure Database for MySQL or Google Cloud SQL

**Popular MySQL tools: -**

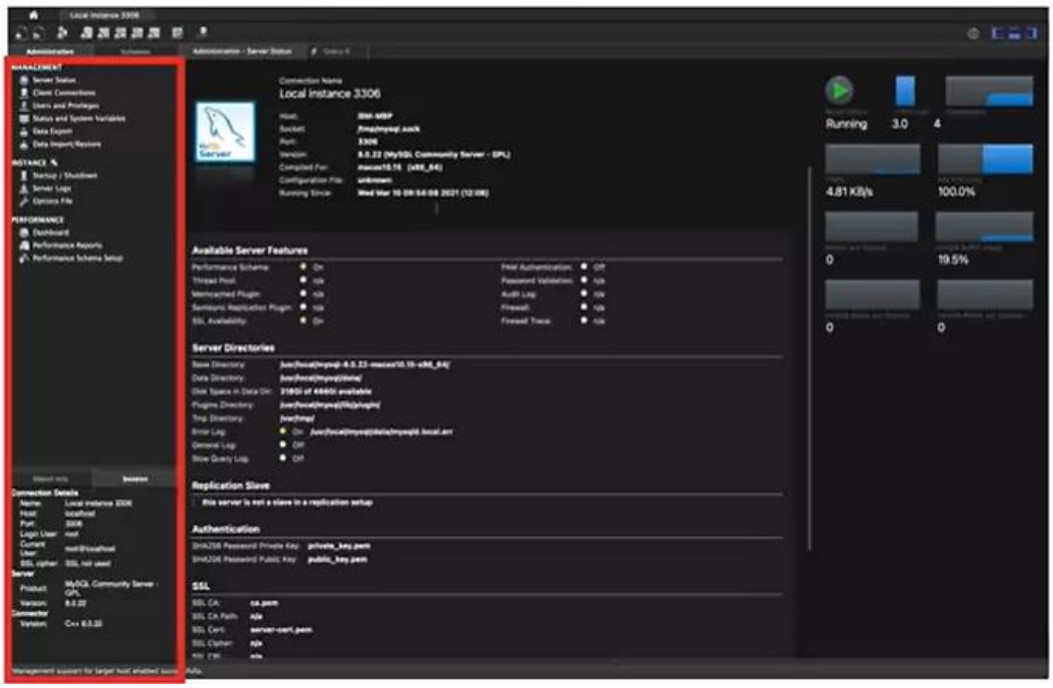
- **MySQL Command Line**
  - Enables you to issue commands to interact with your MySQL server and data.
  - These commands can be typed directly at the prompt for interactive use or run from a text file that you invoke from the command prompt.
  - This screenshot shows running the show databases command interactively to list the databases currently available.
  - When running in batch mode, you can specify a file to store any output messages in for later use.



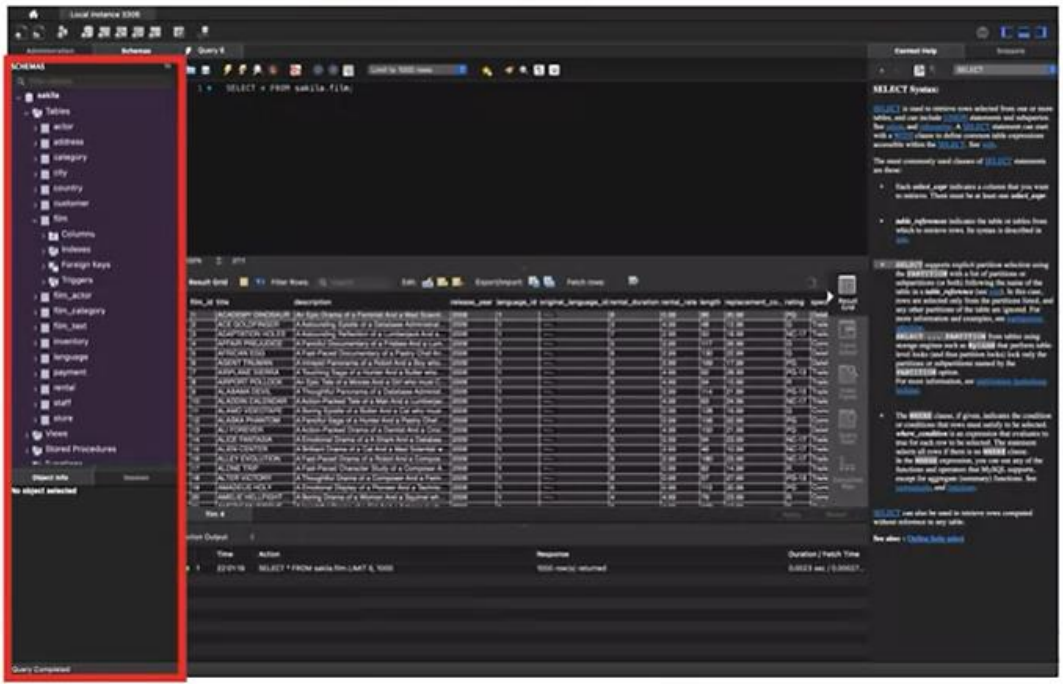

- **mysqladmin command line – for administering RDBMS**

- **MySQL Workbench desktop application for windows, Linux, Mac OS versions.**
  - MySQL Workbench is a visual database design tool that integrates SQL development, administration, database design, creation, and maintenance into a single development environment for the MySQL database system.

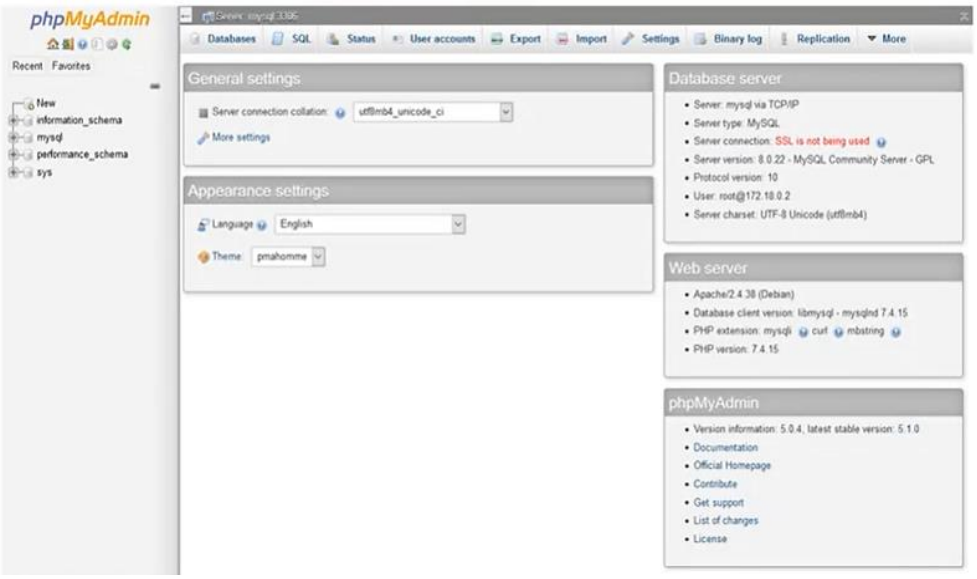

  - From the Administration page, you can view connection details and server features as well as performing administration tasks such as importing and exporting data and reviewing server logs and performance reports. phpMyAdmin – popular third-party web interface.



- The Schemas page enables you to access the objects in your database, work with your data, and it provides direct access to the Help documentation.



- **phpMyAdmin – Third-party web interface**
  - phpMyAdmin is a graphical user interface, or GUI, that you can use to interact with your MySQL databases.
  - When you first connect to the server, you see the server information and the system databases. You can then create your own user databases and use the different tabs to interact with them. You can create databases and tables, load and query data, and import and export data using phpMyAdmin.



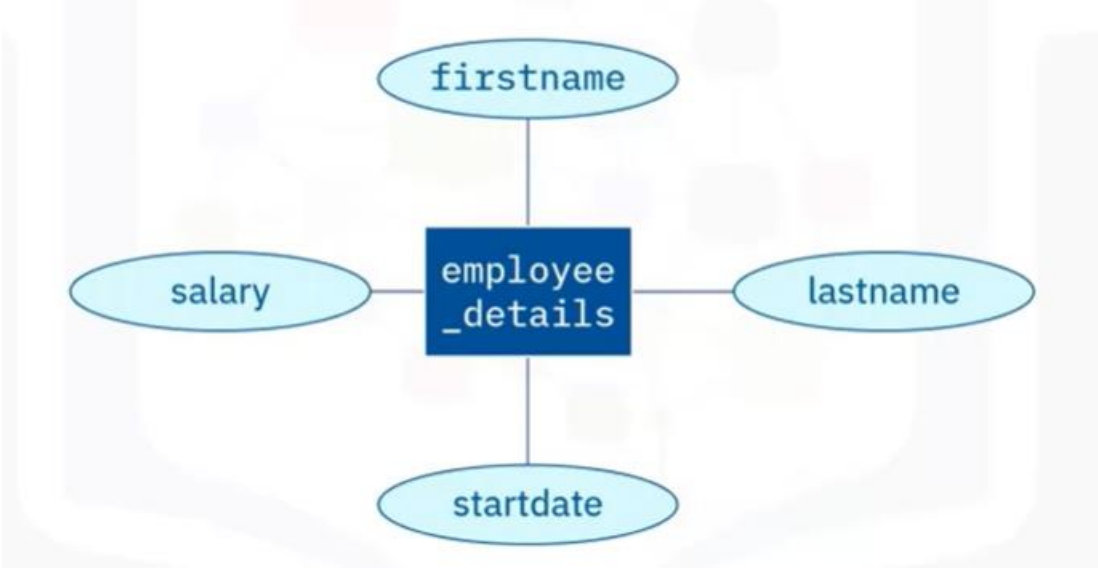
**Summary: -**

- You can download and install MySQL on your own desktop servers,
- You can self-manage or use managed services for MySQL in the cloud.
- mysql and mysqladmin are command line interfaces for database management.
- MySQL Workbench is a desktop application for designing, developing and administering MySQL databases.
- phpMyAdmin is a web interface for working with MySQL databases.

## *2] Creating Databases and Tables in MySQL*

**Creating databases and tables : -**

- As with many RDBMS, you can create databases and tables in MySQL by using a command line interface, a graphical user interface, or API calls.




1. **Using the Command line**

   - You can issue commands at the MySQL command line to create databases objects.
   - You can use the DESCRIBE command to show the structure of the newly created table.

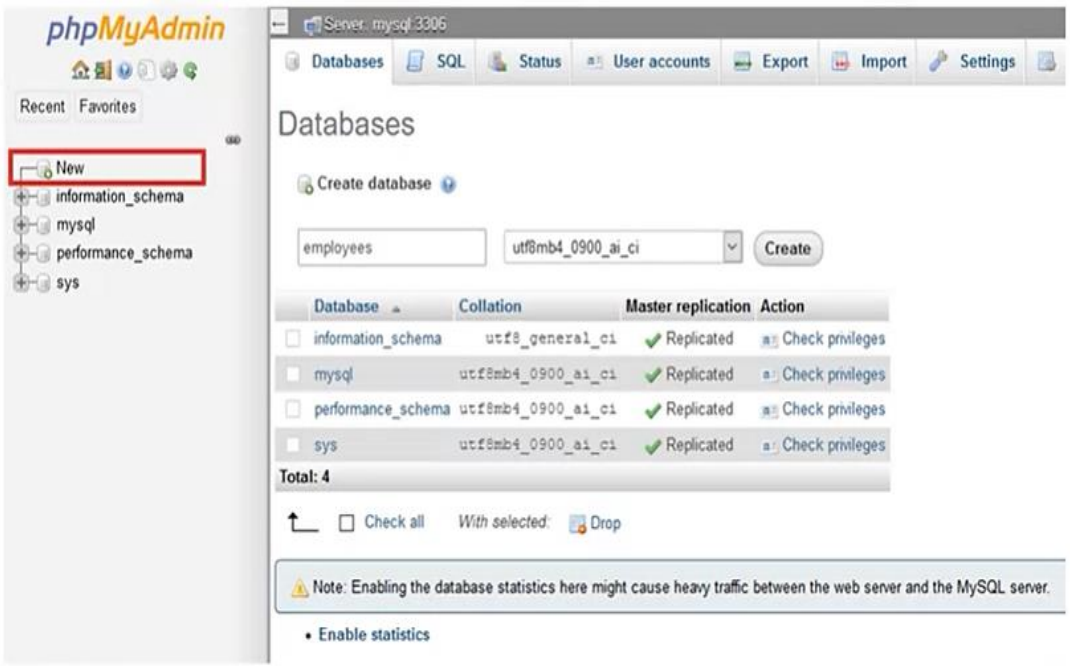
```
CREATE DATABASE employees;
USE employees;
CREATE TABLE employee_details (firstname VARCHAR(20),
lastname VARCHAR(20), startdate DATE, salary DECIMAL);
```

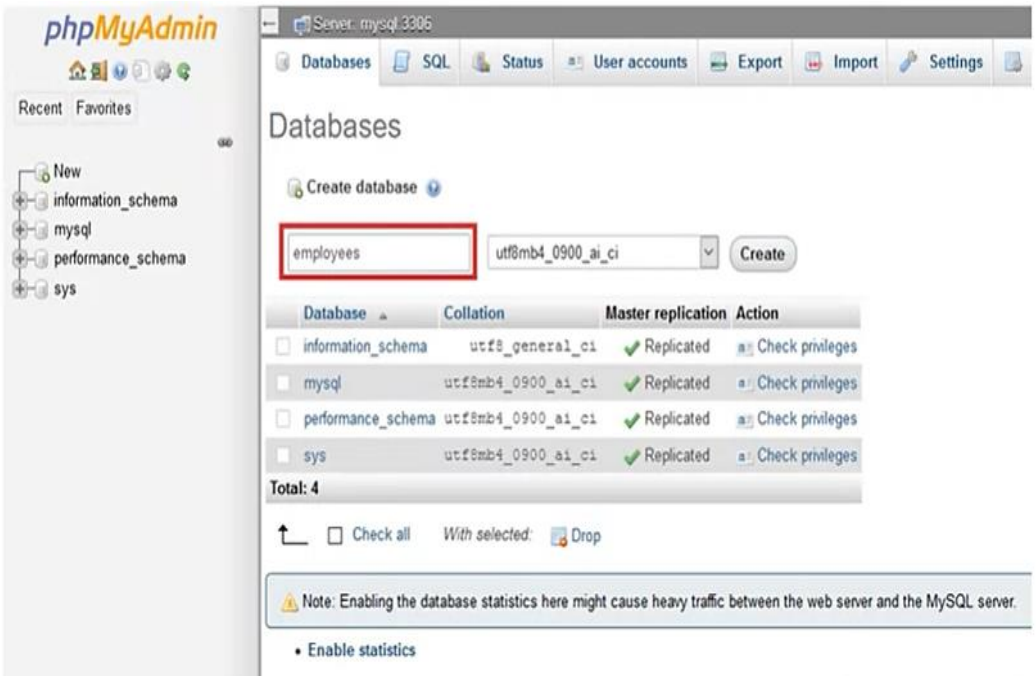
```
mysql> DESCRIBE employee_details;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| firstname | varchar(20)  | YES  |     | NULL    |       |
| lastname  | varchar(20)  | YES  |     | NULL    |       |
| startdate | date         | YES  |     | NULL    |       |
| salary    | decimal(10,0)| YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

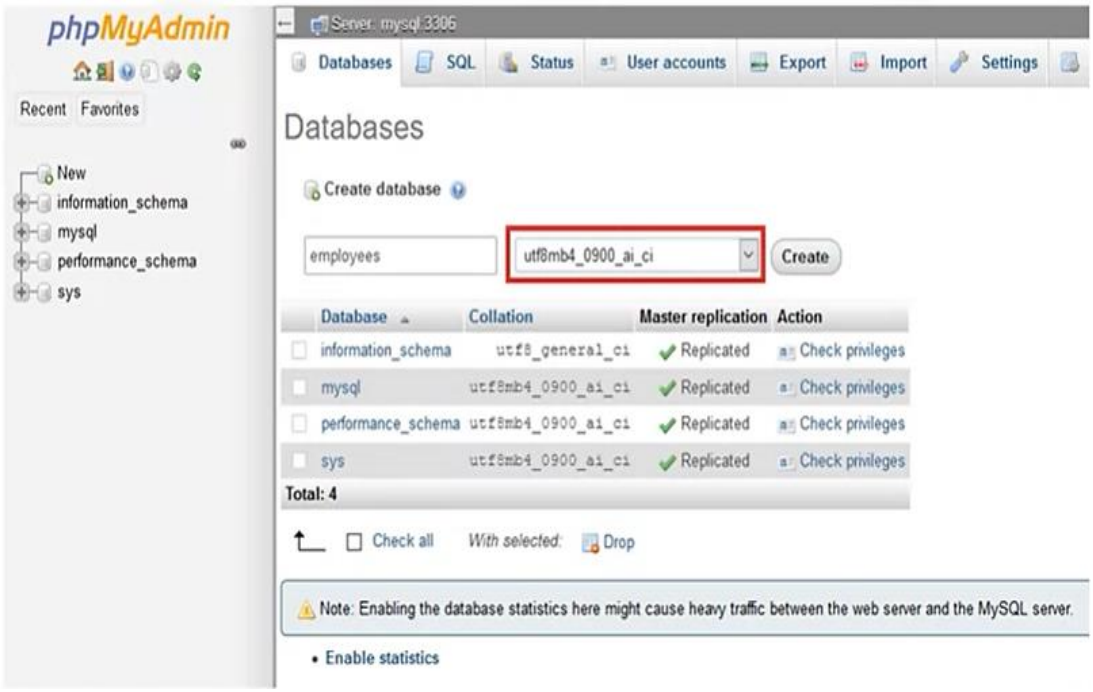
## 2. Using phpMyAdmin

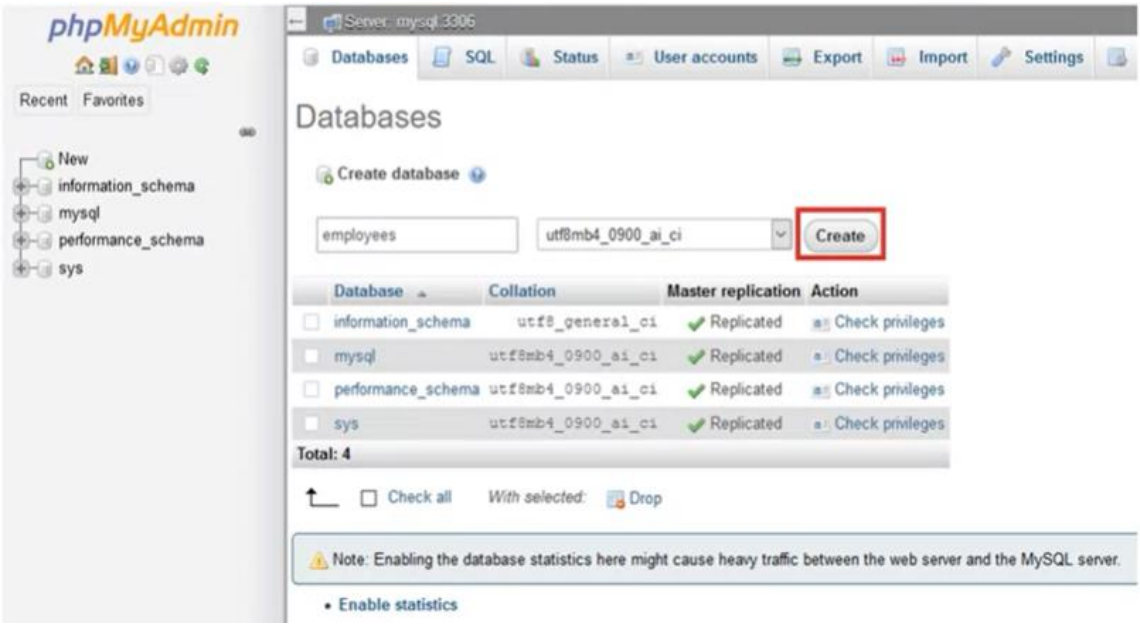
- Alternatively, you can use phpMyAdmin to create the same database and table. phpMyAdmin is a popular visual tool with a web interface for working with MySQL.

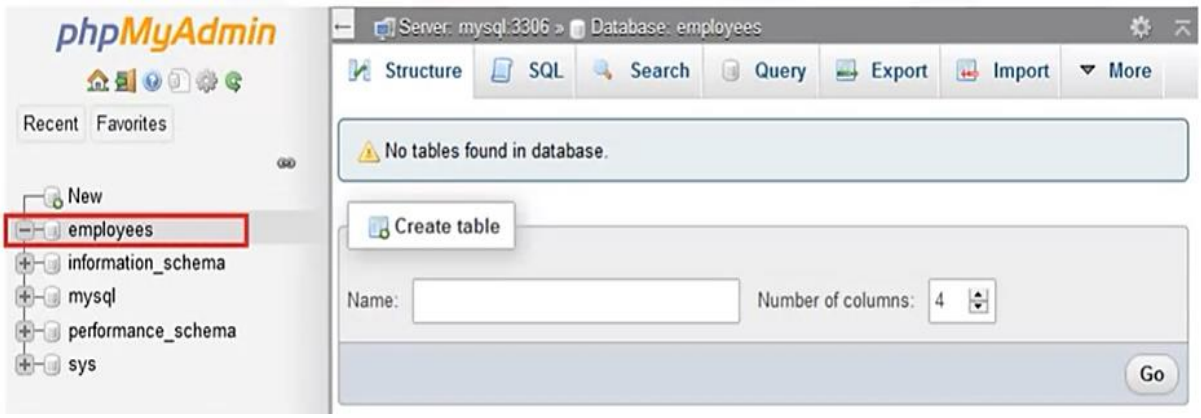
**Creating Database -**



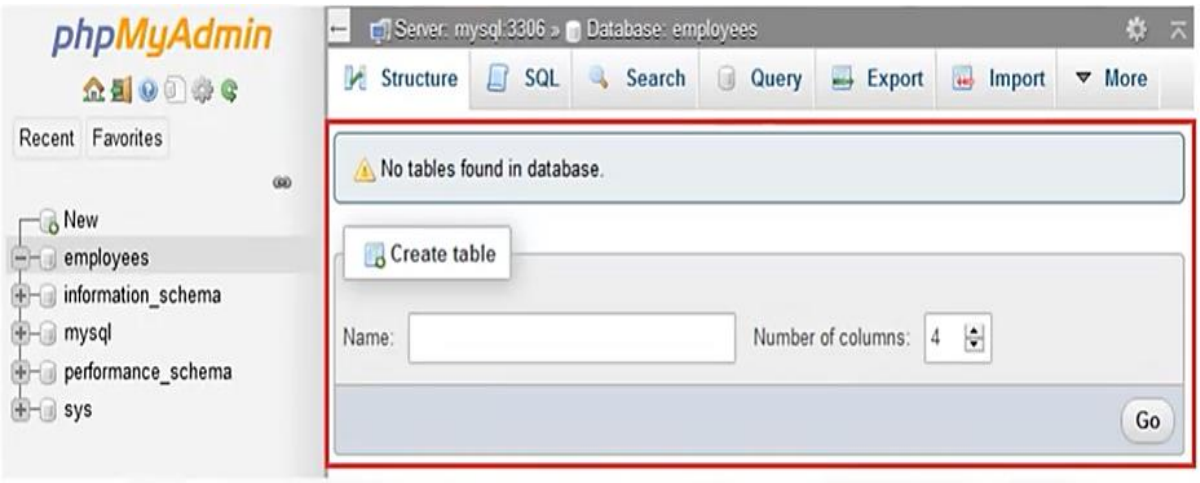

phpMyAdmin
Recent Favorites
New
information_schema
mysql
performance_schema
sys
Server: mysql:3306
Databases SQL Status User accounts Export Import Settings
Databases
Create database
employees
utf8mb4_0900_ai_ci
Create
Database Collation Master replication Action
information_schema utf8_general_ci Replicated Check privileges
mysql utf8mb4_0900_ai_ci Replicated Check privileges
performance_schema utf8mb4_0900_ai_ci Replicated Check privileges
sys utf8mb4_0900_ai_ci Replicated Check privileges
Total: 4
Check all With selected: Drop
Note: Enabling the database statistics here might cause heavy traffic between the web server and the MySQL server.
Enable statistics

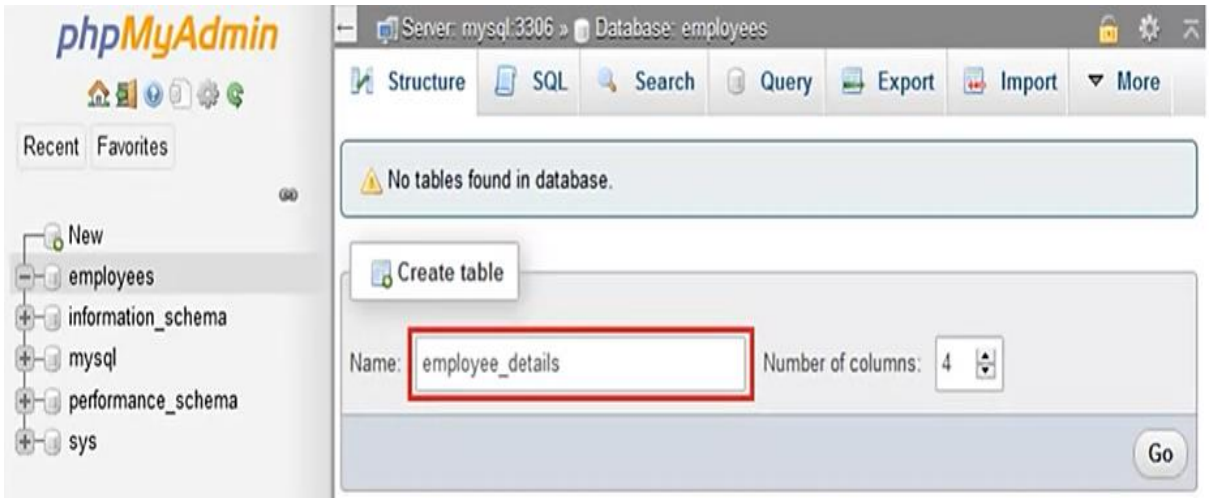

phpMyAdmin
Recent Favorites
New
information_schema
mysql
performance_schema
sys
Server: mysql:3306
Databases SQL Status User accounts Export Import Settings
Databases
Create database
employees
utf8mb4_0900_ai_ci
Create
Database Collation Master replication Action
information_schema utf8_general_ci Replicated Check privileges
mysql utf8mb4_0900_ai_ci Replicated Check privileges
performance_schema utf8mb4_0900_ai_ci Replicated Check privileges
sys utf8mb4_0900_ai_ci Replicated Check privileges
Total: 4
Check all With selected: Drop
Note: Enabling the database statistics here might cause heavy traffic between the web server and the MySQL server.
Enable statistics

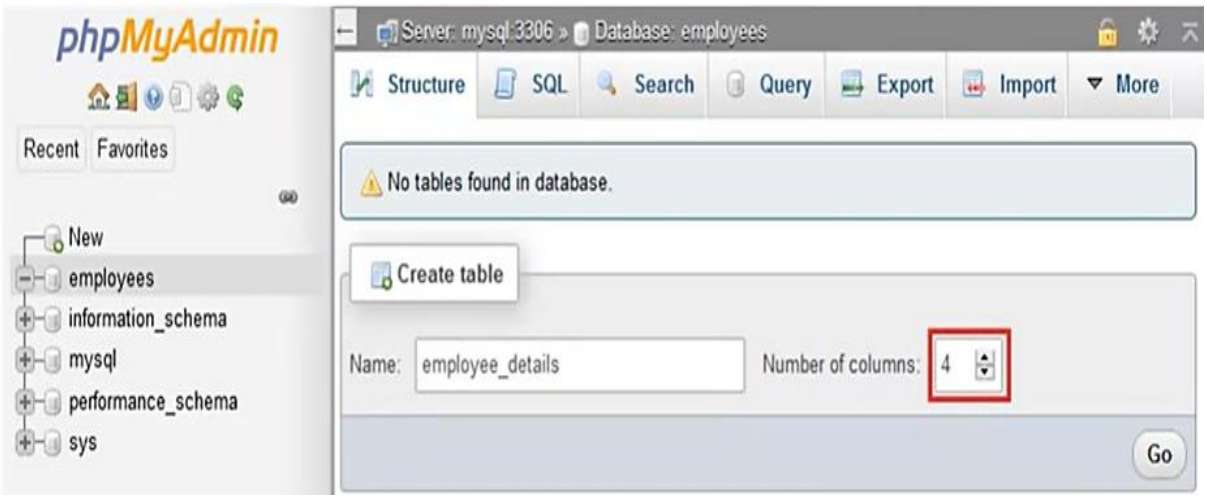


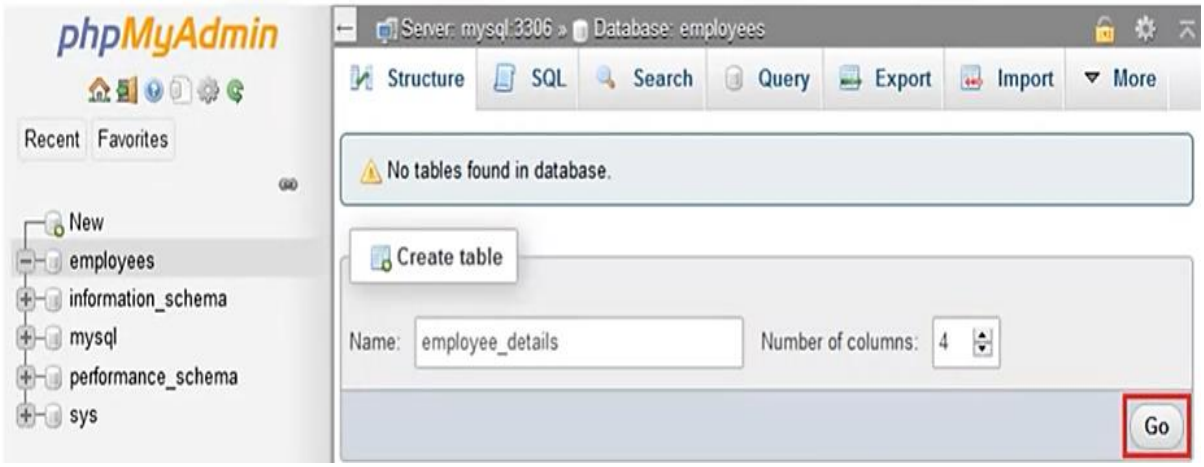




**Creating Tables -**

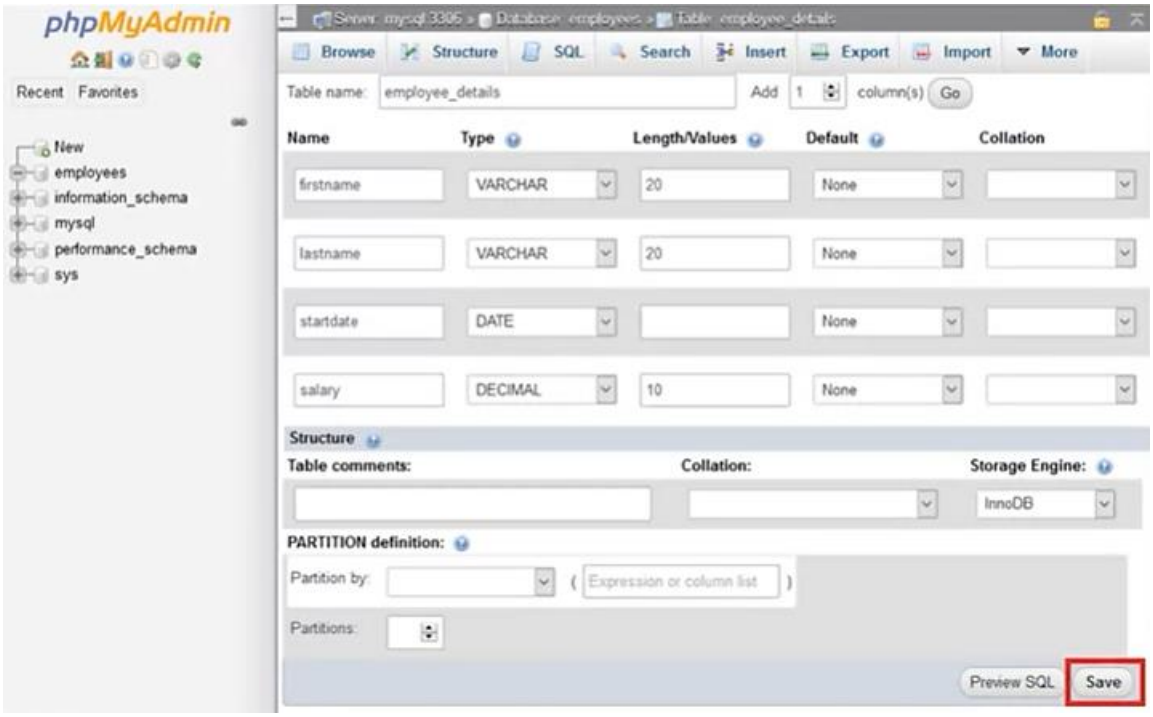



phpMyAdmin
Recent
Favorites
New
employees
information_schema
mysql
performance_schema
sys
Server: mysql:3306 » Database: employees
Structure
SQL
Search
Query
Export
Import
More
No tables found in database.
Create table
Name:
employee_details
Number of columns:
4
Go

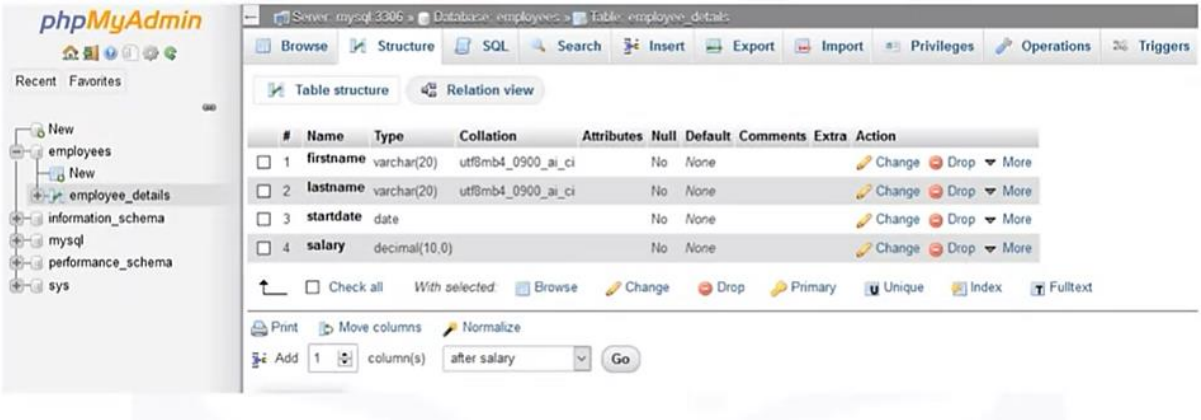

phpMyAdmin
Recent
Favorites
New
employees
information_schema
mysql
performance_schema
sys
Server: mysql:3306 » Database: employees
Structure
SQL
Search
Query
Export
Import
More
No tables found in database.
Create table
Name:
employee_details
Number of columns:
4
Go


phpMyAdmin
Recent
Favorites
New
employees
information_schema
mysql
performance_schema
sys
Server: mysql:3306 » Database: employees
Structure
SQL
Search
Query
Export
Import
More
No tables found in database.
Create table
Name:
employee_details
Number of columns:
4
Go

**Defining Columns -**




**Editing the table definition -**




**Summary: -**

- You can use a command line interface, a graphical user interface, or API calls to create databases and tables in MySQL.
- phpMyAdmin provides an easy-to-use interface to create databases, tables, and columns.
- You can add and modify columns after you create a table.

## *3] Loading Data in MySQL*

**Command line backup and restore: -**

- **Backup using mysqldump utility**

```
mysqldump -u root employees > employeesbackup.sql
```

  - The –u parameter specifies the username of root, employees is the name of the database, and employeesbackup.sql is the name of the file in which to create the backup.
  - If you want to just back up specific tables, you can list their names after the name of the database.

- **Restore using mysql**

```
mysql -u root restored_employees < employeesbackup.sql
```

  - This runs all of the SQL statements in the backup file to recreate the objects and restore the data in the destination database. Note that the greater than sign signifies output to the .sql file, or backup, and the less than sign signifies input to the database, or restore.

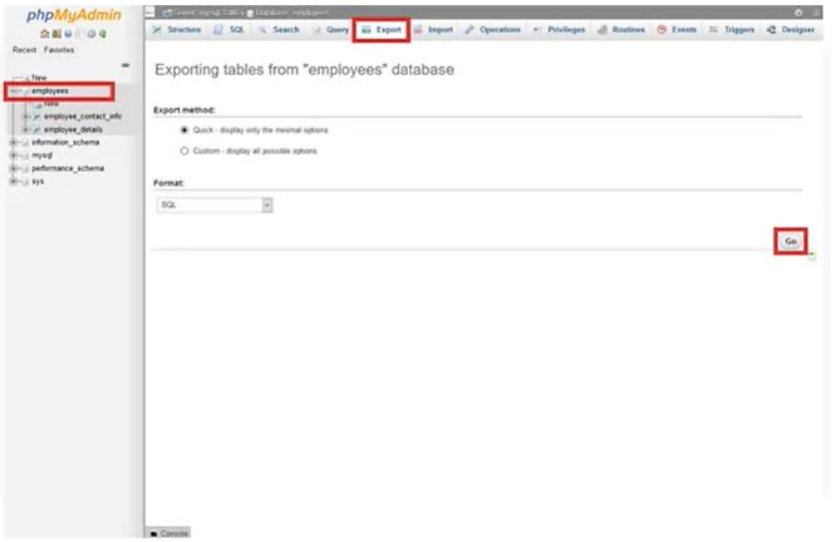
- **Restore using source command**

```
mysql> source employeesbackup.sql
```

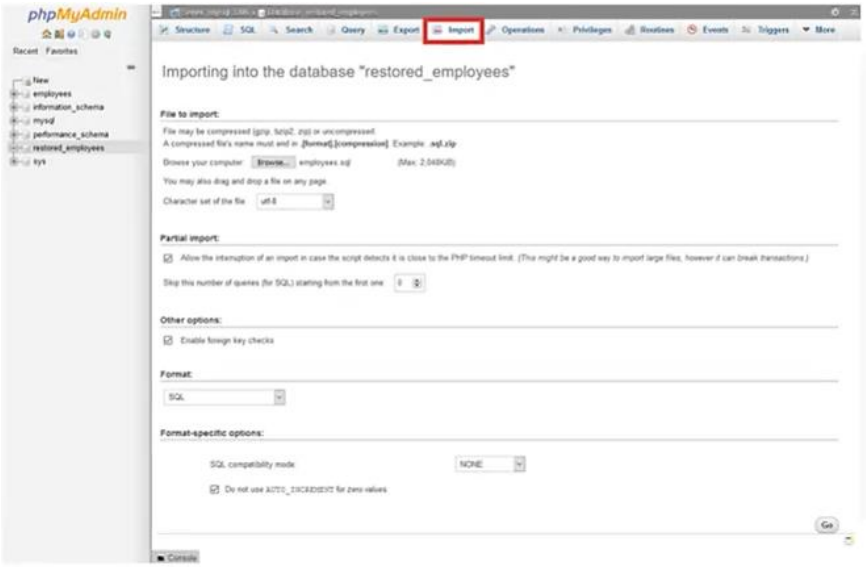
  - This method can also be used execute SQL scripts from file.

**Backup and restore in phpMyAdmin**

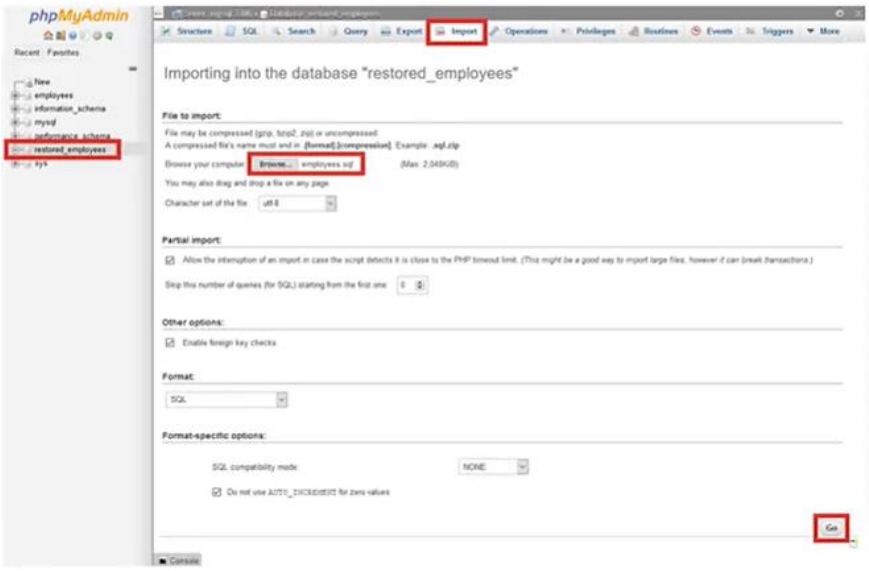
  - Select a database in the tree view, then on the Export tab, click Go. By default, this Quick export method generates an SQL file that contains all of the script necessary to completely recreate the contents of your database.




- You can then use the Import tab to restore this database to this or another instance of MySQL.




- Select your destination database, locate the backup file and then click Go. Again, this runs all of the SQL statements in the backup file to recreate the objects and restore the data in the destination database.



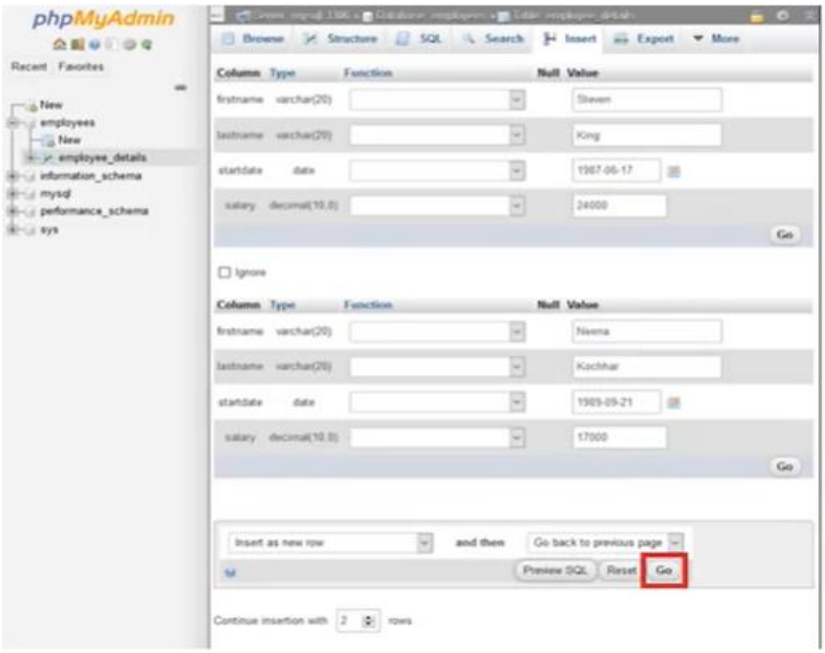

**Loading data: -**

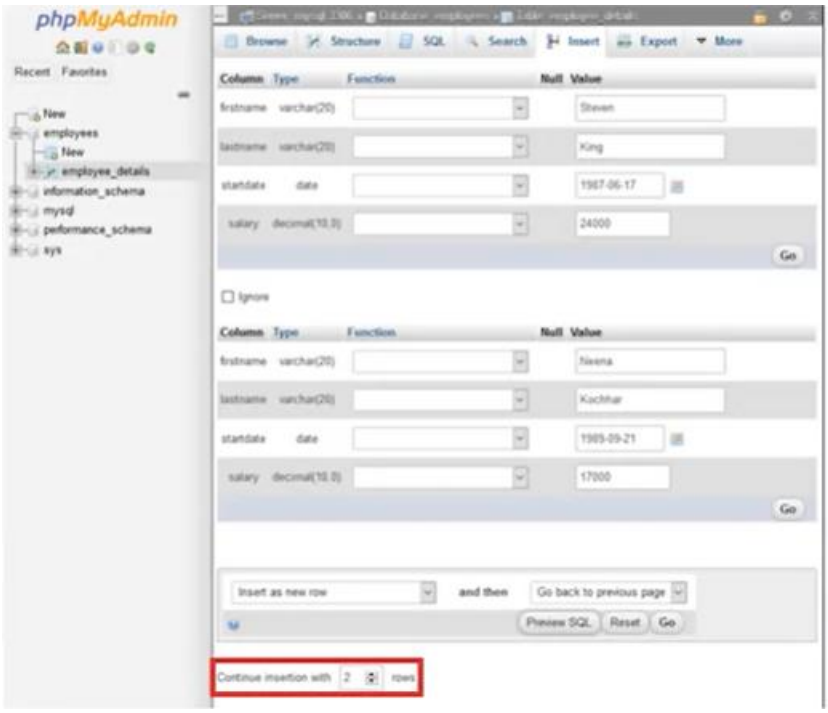
**phpMyAdmin tool: -**

- Manually enter rows
- Run INSERT statements

**Entering data -**

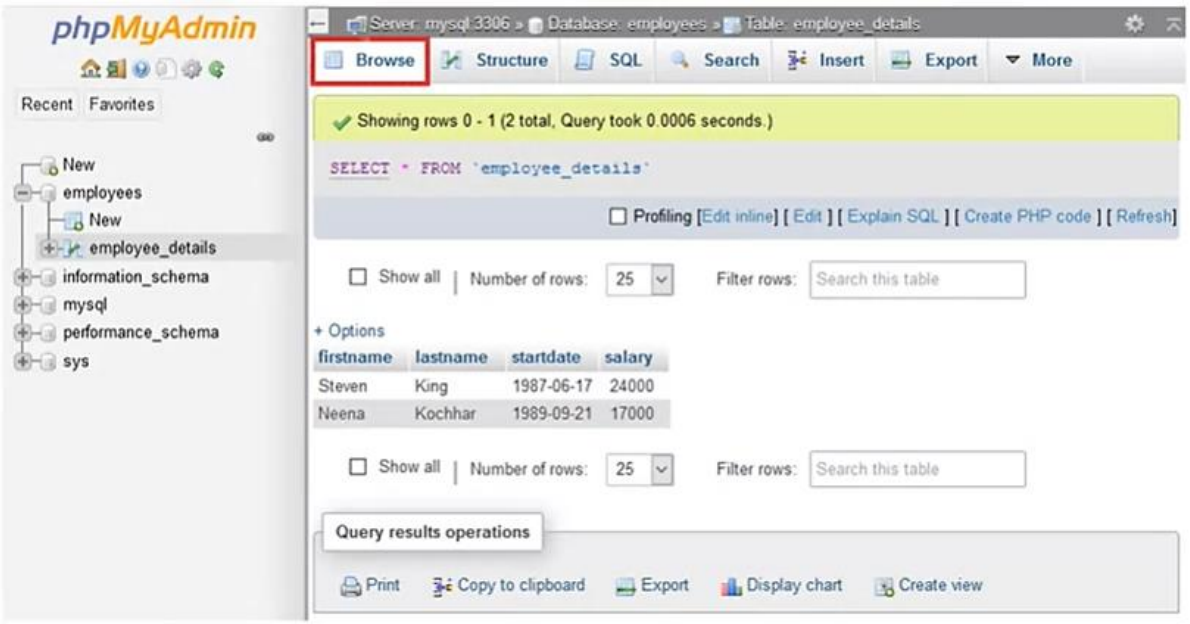
- Alternatively, if you only want to populate a small number of rows in an individual table instead of populating an entire database, you can use the phpMyAdmin tool to manually enter rows or run SQL INSERT statements To manually enter rows in phpMyAdmin, on the Insert tab, enter the data, and then click Go. By default, you can enter two rows at a time, but you can increase or decrease this as required.






**Viewing data –**

- After you have entered data into a table, you can view that data on the Browse tab.




- Manually entering rows and running individual SQL statements is fine for small amounts of data, but if you are loading a large number of rows, you'll find the import functionality is easier and quicker to use.

**Importing data files –**

- **Importing using load data infile statement**

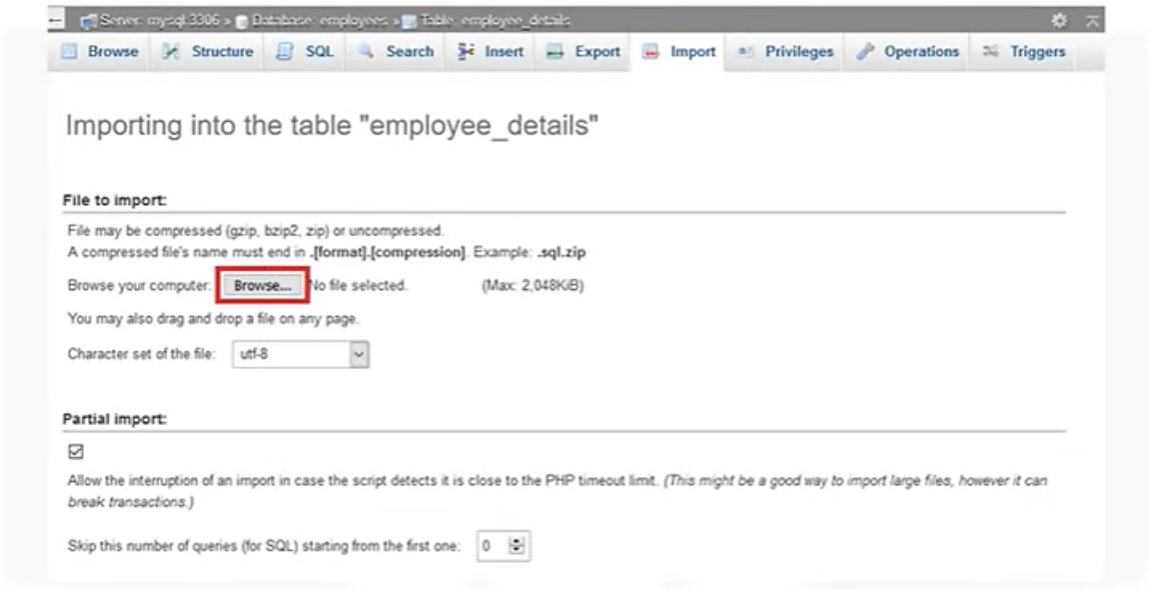
```
load data infile 'employeesdata.csv' into table employees_details
```

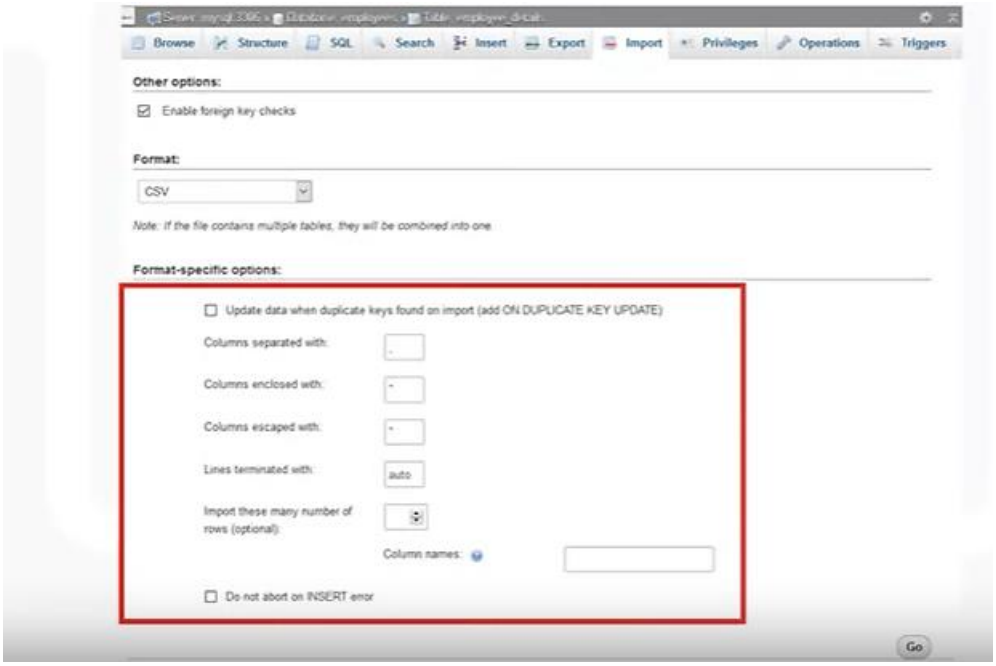
- **Importing using mysqlimport utility**

```
mysqlimport employees employees_details.csv
```

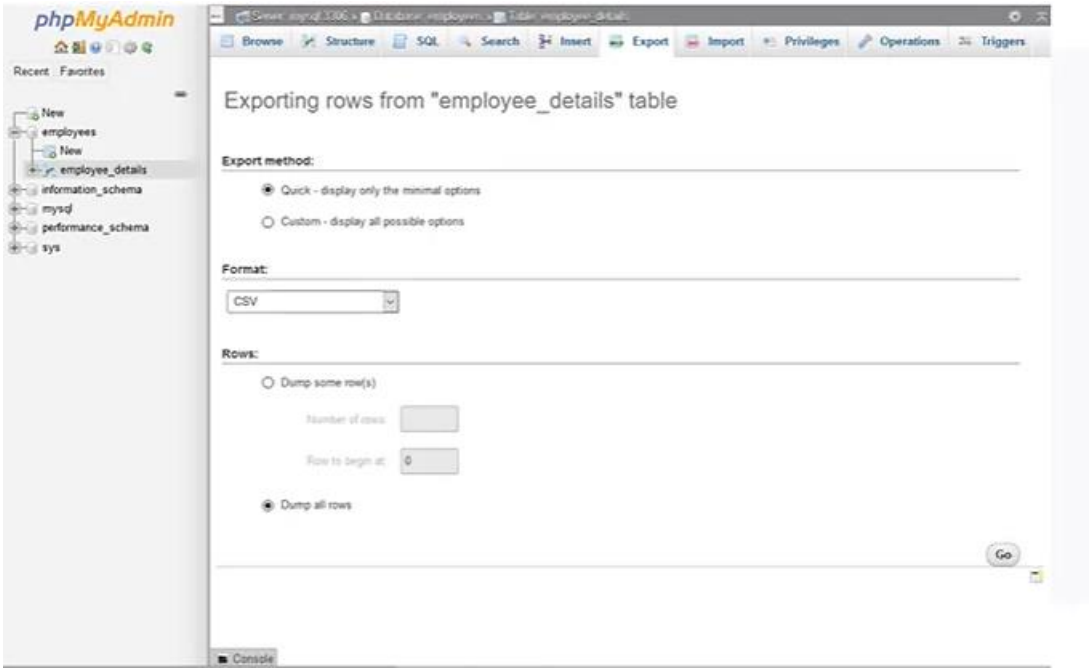
- **Importing data files in phpMyAdmin**

  - Alternatively, phpMyAdmin provides a visual interface for importing data into tables. On the Import tab, click Browse to select your file, check that the format and options have been correctly determined from the data file, and then click Go. Using this method, you can import up to 2 megabytes of data at a time.






- You can also use phpMyAdmin to export data from a table to CSV format. On the Export tab, change the Format to CSV, optionally specify which rows to export, and then click GO.




**Summary: -**

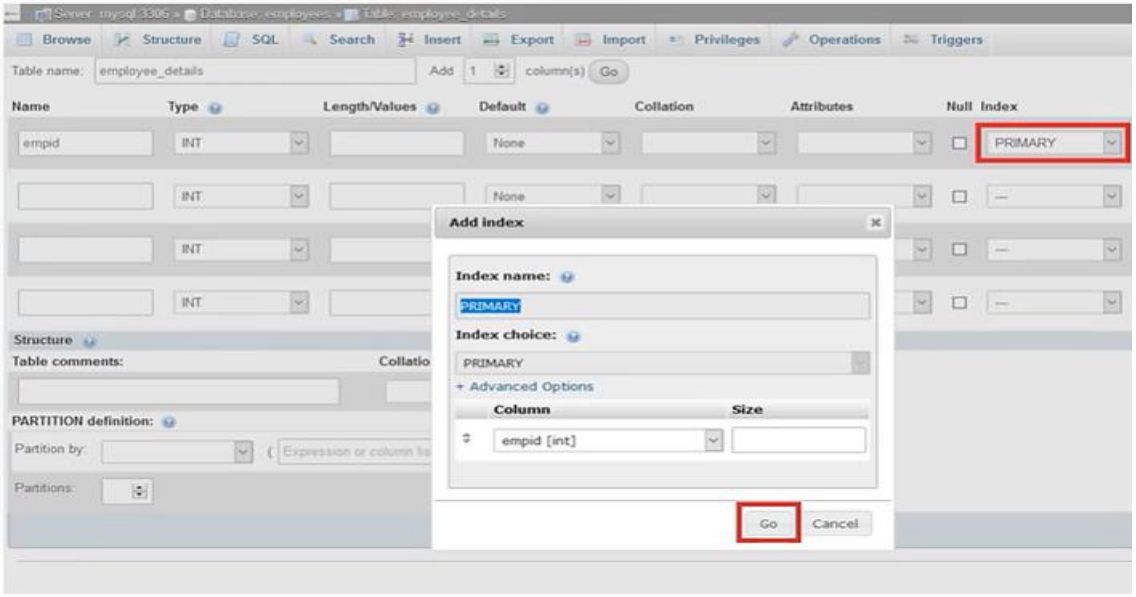
- You can use backup and restore functionality at the command line and in phpMyAdmin to populate databases.
- You can use manually insert small amounts of data using phpMyAdmin.

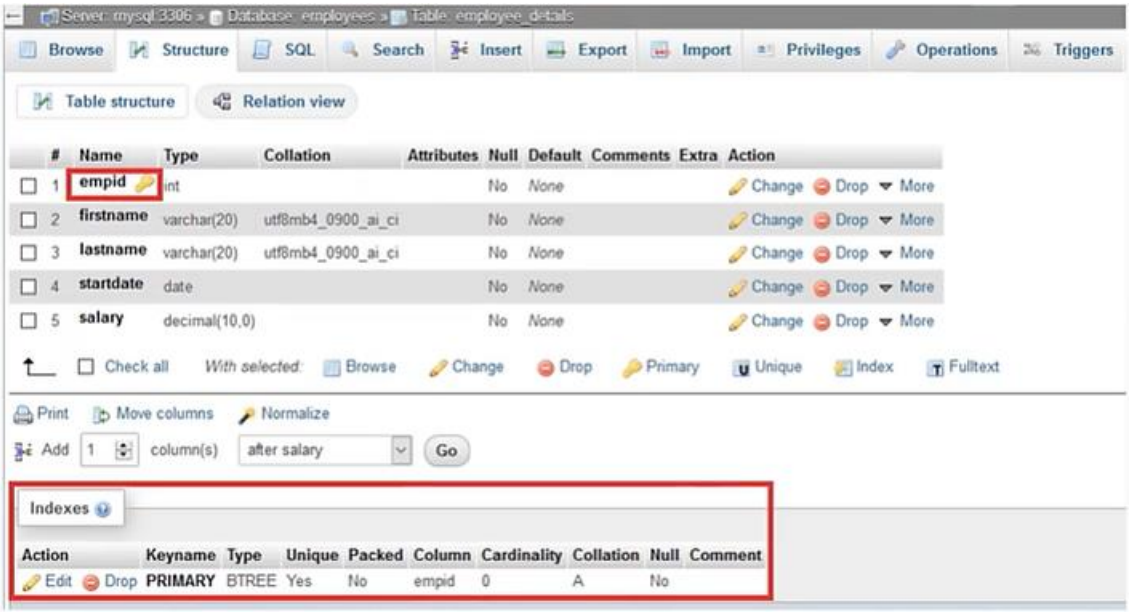
- You can use import and export functionality at the command line and in phpMyAdmin to populate tables and save their data to files.

## 4] Using Keys and Constraints in MySQL

**Types of keys and constraints: -**

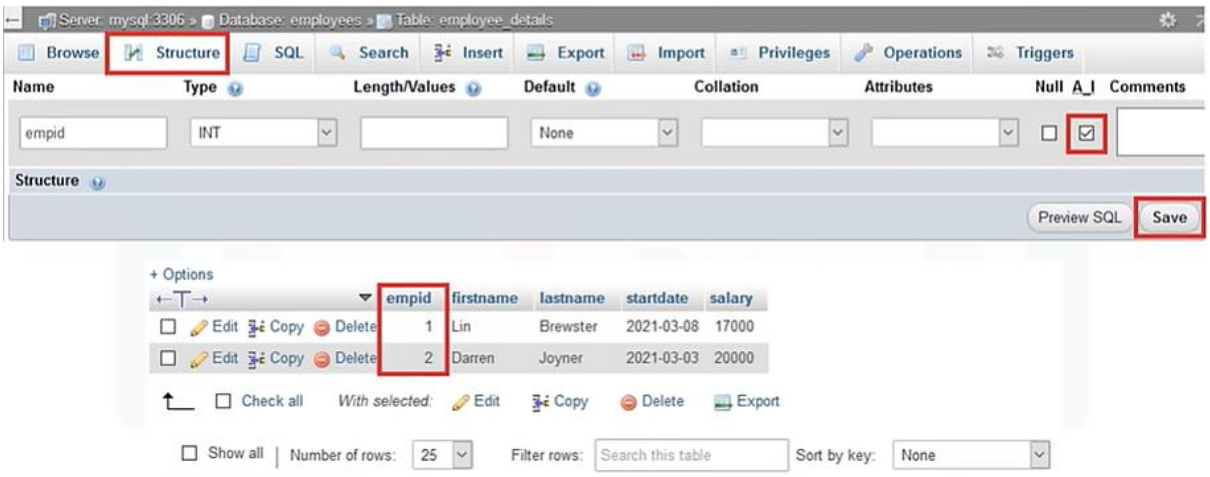
- **Primary keys**
  - One column or a combination of columns
  - Not null
  - Unique
  - Indexed



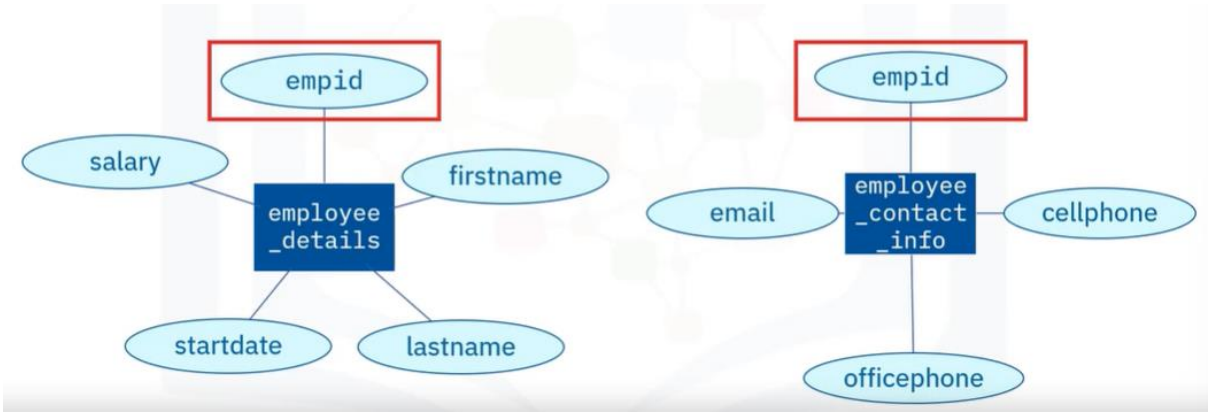
- **Auto increment**

  - Automatic generation of incrementing numbers.
  - On the Create table or the Structure tab, select the A_I checkbox for your primary key row and then click Save. Now when you add data to the table, the database engine automatically generates incrementing entries for the empid column.

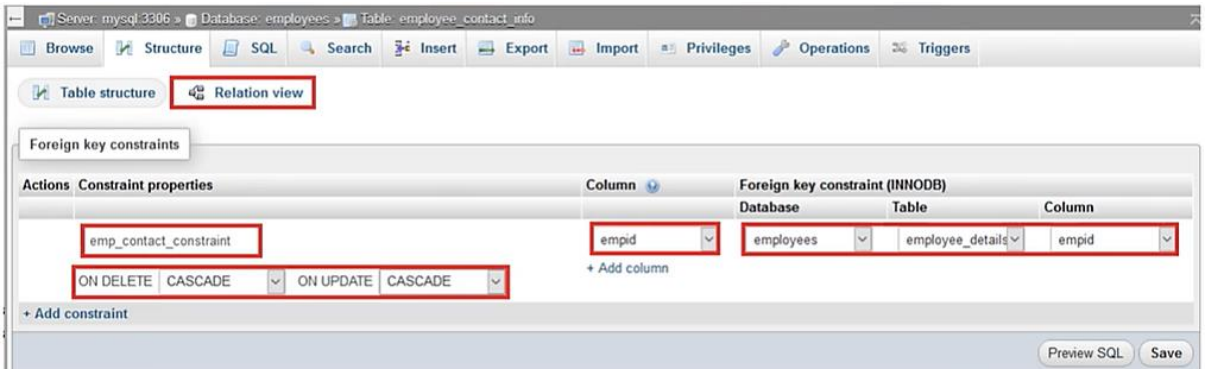



- **Foreign keys**

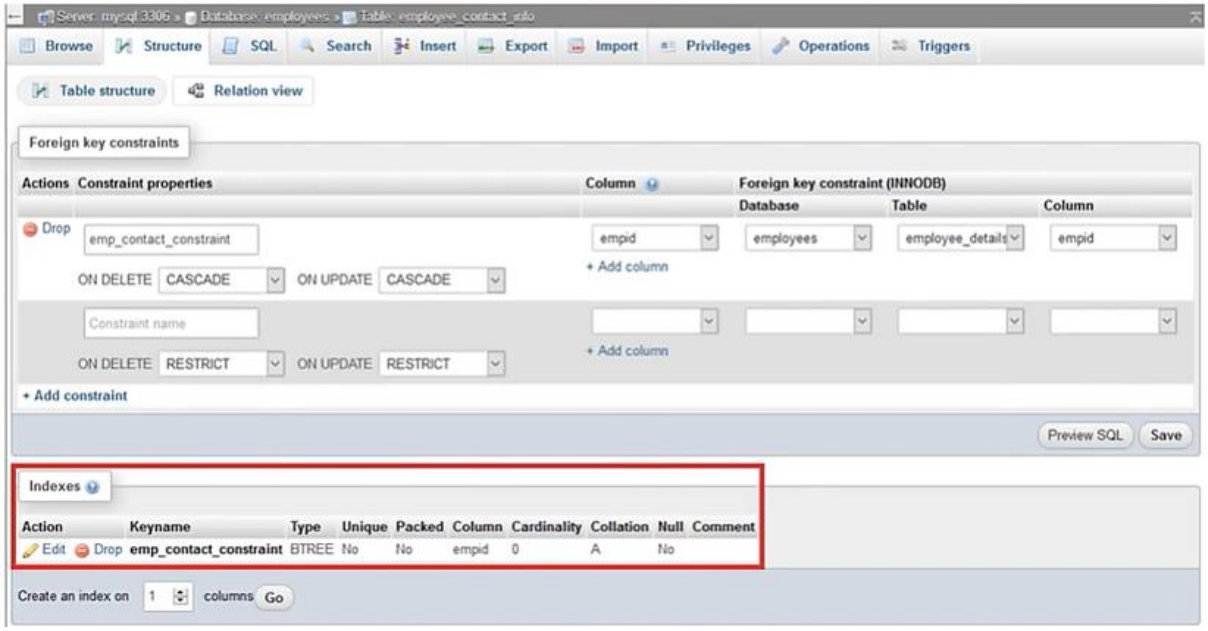
  - You can also create foreign keys to relate data across tables.
  - For example, you can use the empid column in the employee_details table to link to the empid column in the employee_contact_info table by creating a foreign key.




  - To create a foreign key, on the Relation view of the Structure tab, enter the name for your key and identify the columns that define it. You can also specify what action to take when a related row is deleted or updated.

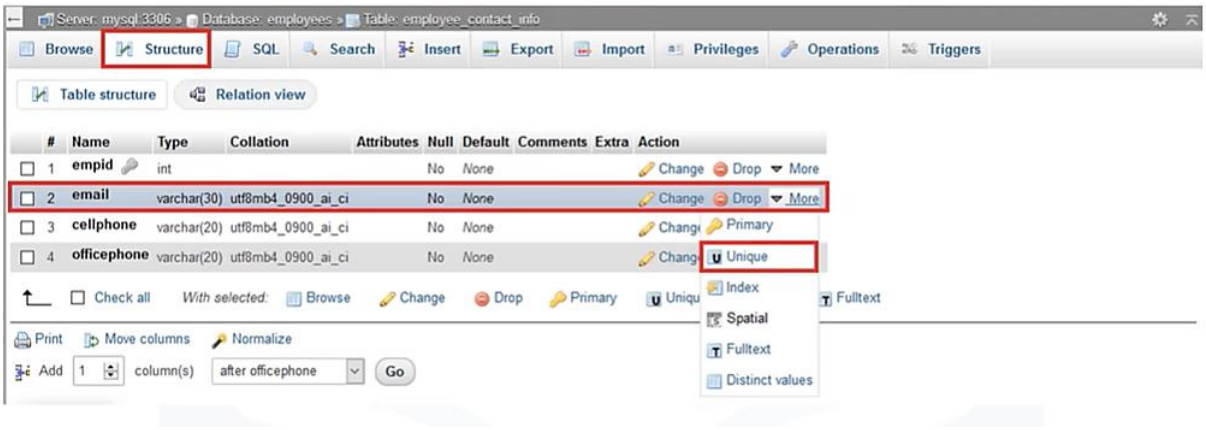



- And as with primary keys, the underlying index <click> now also shows on the Structure tab.



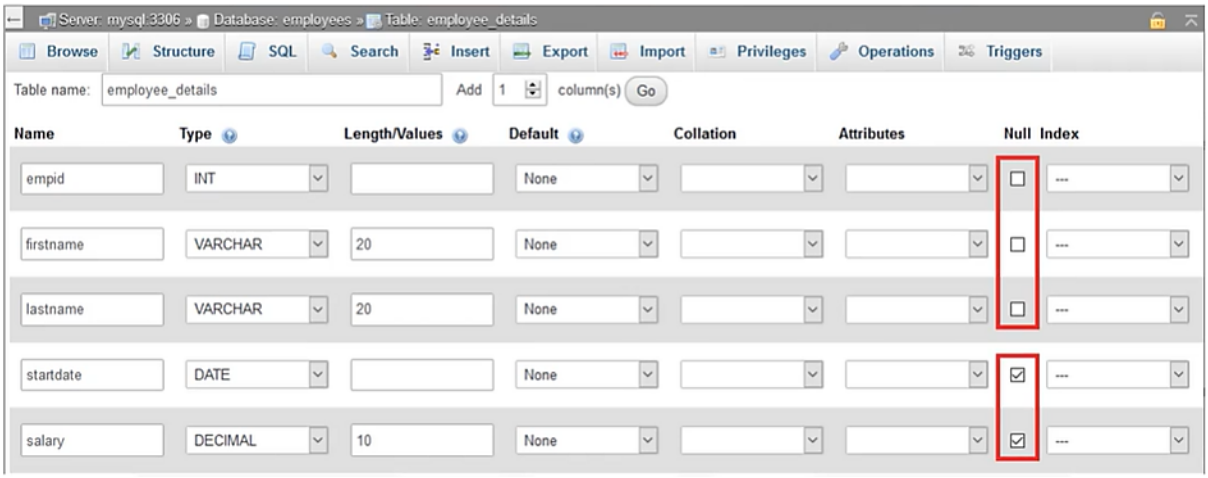

- **Unique constraints**

  - To ensure that the email address for each employee is unique, you can use the unique constraint.
  - On the Structure tab, click the More link for the relevant column, and then click Unique.




- **Null constraints**

  - By default, when you create a MySQL table in phpMyAdmin the columns are defined as not null. You can change this either when you create the table or by changing the column definition.
  - For example, to enable null values to be entered in the startdate and salary columns, select the Null check box. Leaving the empid, firstname, and lastname null checkboxes blank ensures that those columns require data.




**Summary: -**

- You create primary keys by defining a primary index on one or more columns.
- You use autoincrement to automatically generate sequential numeric data in a column
- When creating foreign keys, you can define ON DELETE and ON UPDATE actions.
- MySQL columns are NOT NULL by default.
- You can configure a column to only accept unique values.

## ***Summary & Highlights:-***

- MySQL is a free, open-source RDMS that you can download and install on your own systems or access on the Cloud. You can either self-manage a Cloud instance of MySQL or use a managed services provider, including IBM Cloud, Amazon RDS for MySQL, Azure Database for MySQL, or Google Cloud SQL for MySQL.
- MySQL includes several options for creating databases and tables, loading and querying data, and importing and exporting data relational databases:
    - **mysql and mysqladmin command line interfaces -** You use these CLIs to run SQL statements.
    - **MySQL Workbench -** A desktop application for designing, developing, and administering MySQL databases.
    - **phpMyAdmin -** An easy to use, third-party web interface for working with MySQL databases.
    - API calls.
- Using phpMyAdmin, you can:
    - Add and modify columns after you create a table.
    - Use backup and restore functionality to populate databases.
    - Use import and export functionality to populate tables and save their data to files.
    - Create primary keys by defining a primary index on one or more columns.
    - Use autoincrement to automatically generate sequential numeric data in a column.
- When creating foreign keys, you can define ON DELETE and ON UPDATE actions.
- MySQL columns are NOT NULL by default.
- You can configure a column to only accept unique values.

# *Module 2: - PostgreSQL*

## *1] Getting Started with PostgreSQL*

**PostgreSQL RDBMS: -**

- Postgres is an open-source object-relational database management system with:
  - reliability and flexibility
  - support for both relational and non-relational data types
- Postgres is a popular database choice for:
  - OLTP
  - data analytics
  - geographic information systems.

**PostgreSQL options: -**

- Download and install on:
  - macOS
  - UNIX, UNIX-based, and UNIX-like systems
  - Windows
- Cloud:
  - VM images or containers
  - Managed services

**PostgreSQL Tools: -**

- psql command line
- pgAdmin: provides an open-source graphical interface to the database server which is available as a desktop application or as a web application that you can install on your web servers.
- Navicat, DBeaver: commercial graphical interface options that you can use to access Postgres, MySQL, and other types of databases
- Cloud vendor tools and APIs: For example, Amazon RDS for PostgreSQL provides a web-based management console and RDS APIs.

**Using psql –**

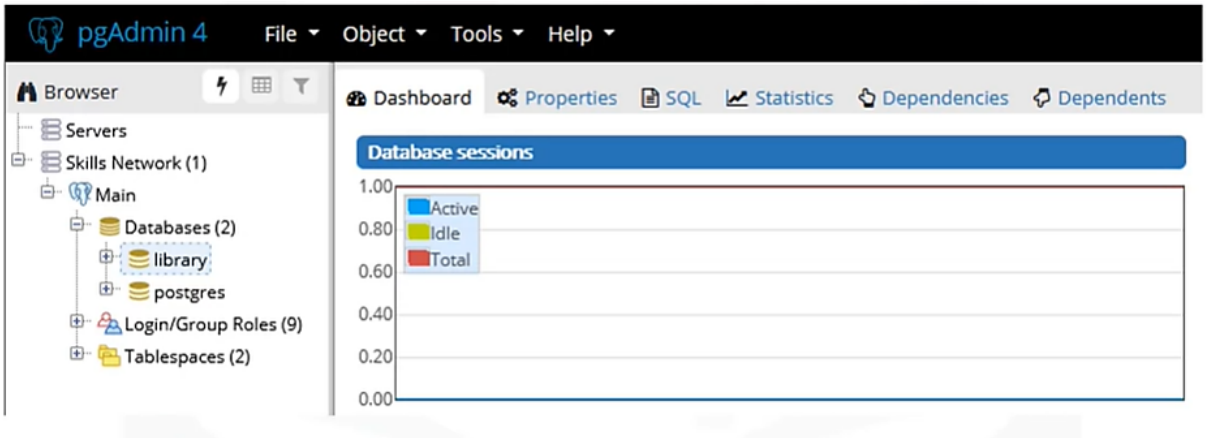
- An interactive command line tool
- You can run interactive queries and display information about the objects in your database

- The screenshot here shows the library and postgres user databases and the internal template0 and template1 databases which postgres uses as templates when you create a new database.

```
postgres=# \l
                                 List of databases
   Name    |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges
-----------+----------+----------+------------+------------+-----------------------
 library   | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 postgres  | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
           |          |          |            |            | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
           |          |          |            |            | postgres=CTc/postgres
(4 rows)
```

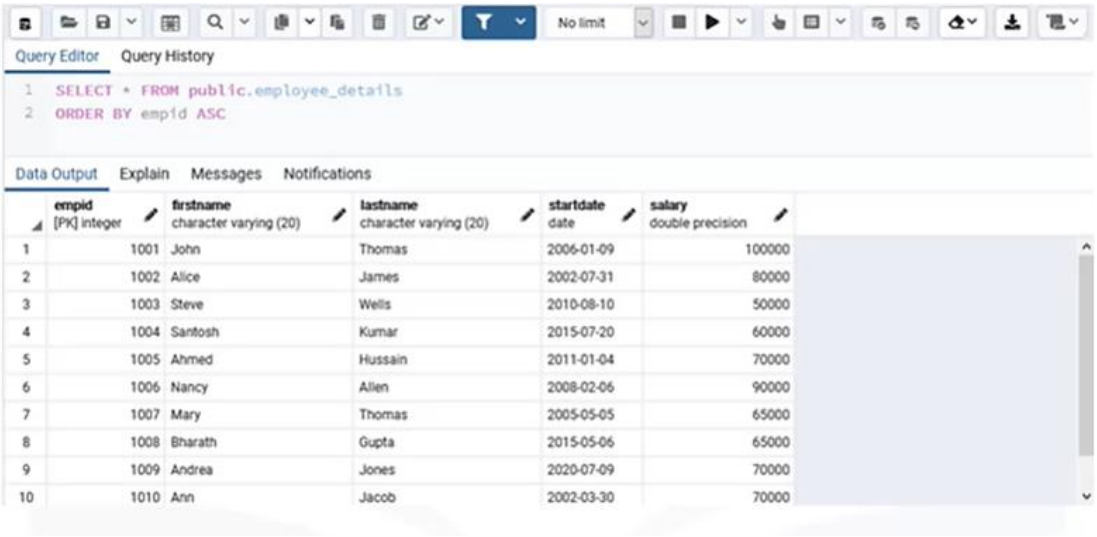
**Using pgAdmin –**

- You can use pgAdmin to complete all your development and administrative tasks, including creating databases and tables, loading data, querying data, writing stored procedures and functions, managing database objects, managing security, and monitoring usage.
- Includes:
  - Query Tool: you can use to run SQL commands and view or interact with their results
  - ERD Tool: you can use to create an ERD for an existing database or to create a new ERD and generate the SQL statements for creating the underlying database objects.
- When you first connect to the server, you will connect to the default postgres database. After you have created other databases, you can specify to connect directly to them. You can use pgAdmin to complete all your development and administrative tasks, including creating databases and tables, loading data, querying data, writing stored procedures and functions, managing database objects, managing security, and monitoring usage.



**Using pgAdmin Query Tool –**

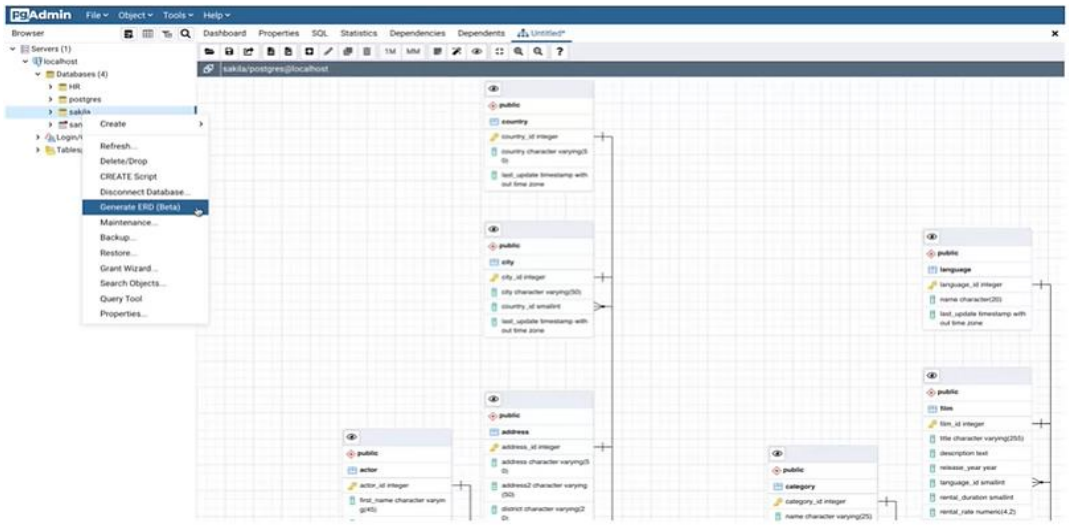
- You can use to run SQL commands and view or interact with their results. In the upper pane, you can type or paste SQL queries and the results are shown below. If the results are editable, you can use this area to edit the data set. You can also use the tabs to see an explanation of the query plan, server messages, and asynchronous server notifications in this lower pane.




| | empid [PK] integer | firstname character varying (20) | lastname character varying (20) | startdate date | salary double precision |
|---|---|---|---|---|---|
| 1 | 1001 | John | Thomas | 2006-01-09 | 100000 |
| 2 | 1002 | Alice | James | 2002-07-31 | 80000 |
| 3 | 1003 | Steve | Wells | 2010-08-10 | 50000 |
| 4 | 1004 | Santosh | Kumar | 2015-07-20 | 60000 |
| 5 | 1005 | Ahmed | Hussain | 2011-01-04 | 70000 |
| 6 | 1006 | Nancy | Allen | 2008-02-06 | 90000 |
| 7 | 1007 | Mary | Thomas | 2005-05-05 | 65000 |
| 8 | 1008 | Bharath | Gupta | 2015-05-06 | 65000 |
| 9 | 1009 | Andrea | Jones | 2020-07-09 | 70000 |
| 10 | 1010 | Ann | Jacob | 2002-03-30 | 70000 |

**Using pgAdmin ERD Tool –**

- You can use to create an ERD for an existing database or to create a new ERD and generate the SQL statements for creating the underlying database objects. To create an entity-relationship diagram from an existing database, right-click the database, and then click Generate ERD. The tool reviews your database structure and generates a visual diagram of the tables in the database and the relationships between them. You can reorganize the tables, add, edit and delete relationships, add notes, and generate SQL statements in the tool.

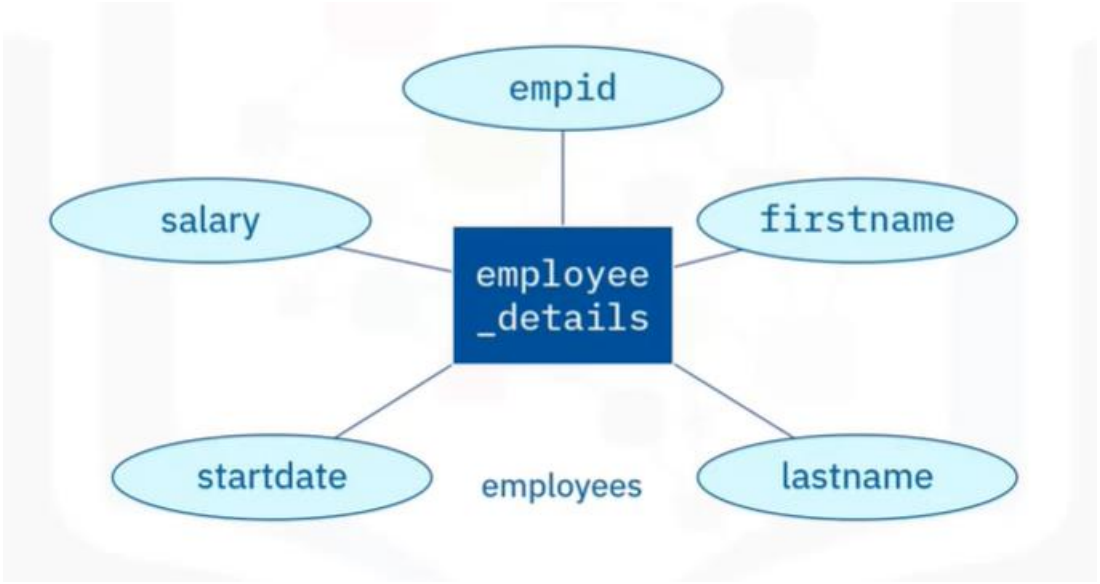


**Summary: -**

- You can install PostgreSQL on your own servers or work with it in the cloud.
- psql provides a command-line interface to a Postgres server.
- pgAdmin is a popular database management tool for PostgreSQL which includes object navigation, the Query Tool, and the ERD Tool

## *2] Creating Databases and Loading data in PostgreSQL*

**Creating databases and tables : -**




1. **Using psql to create a database –**

    - You can use psql to issue commands to create and interact with databases objects and data.
    - Use the CREATE DATABASE command to create the database and CREATE TABLE for tables, specifying column names and data types.
    - You can use the \d command to show the structure of the newly created table

```
CREATE DATABASE employees;
\connect employees;
CREATE TABLE employee_details (firstname VARCHAR(20),
lastname VARCHAR(20), startdate DATE, salary DECIMAL);
```

```
employee=# \d employee_details
                Table "public.employee_details"
  Column   |         Type          | Collation | Nullable | Default
-----------+-----------------------+-----------+----------+---------
 firstname | character varying(20) |           |          |
 lastname  | character varying(20) |           |          |
 startdate | date                  |           |          |
 salary    | numeric               |           |          |

employee=#
```

2. **Using psql to restore a data –**

- Retore a previously backed up database

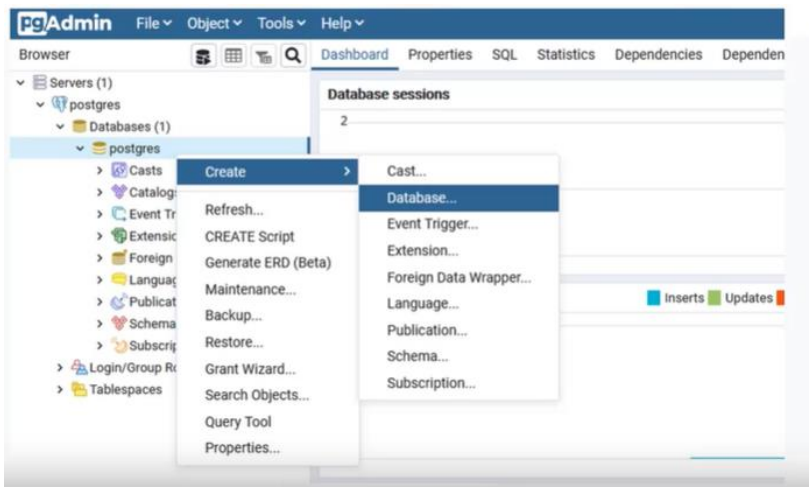
```
psql restored_employees < employeesbackup.sql
```

  - **restored_employees:** the name of the destination database
  - **employeesbackup.sql:** the name of the dump file
- Recreates:
  - Tables
  - Other database objects
  - Data

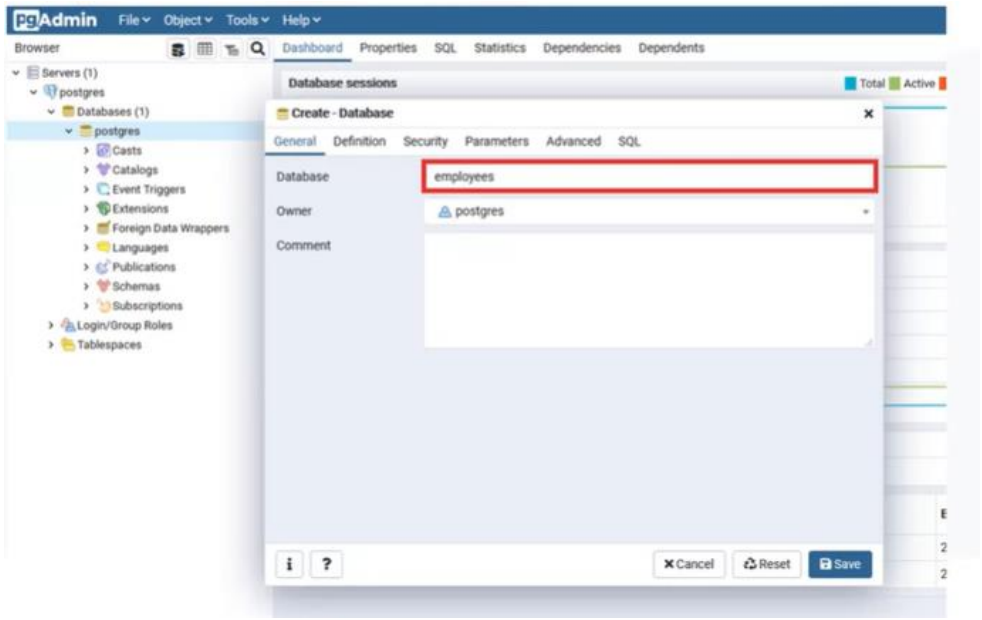
3. **Using pgAdmin –**

**Creating database in pgAdmin –**

- pgAdmin is a web based visual tool you can use to work with PostgreSQL.
- To create the database, in the treeview in the left-hand pane, right-click Databases, click Create, and then click Database.

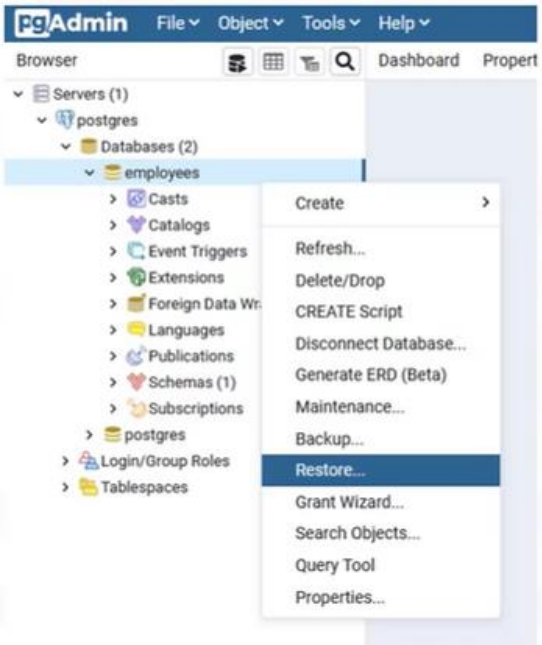


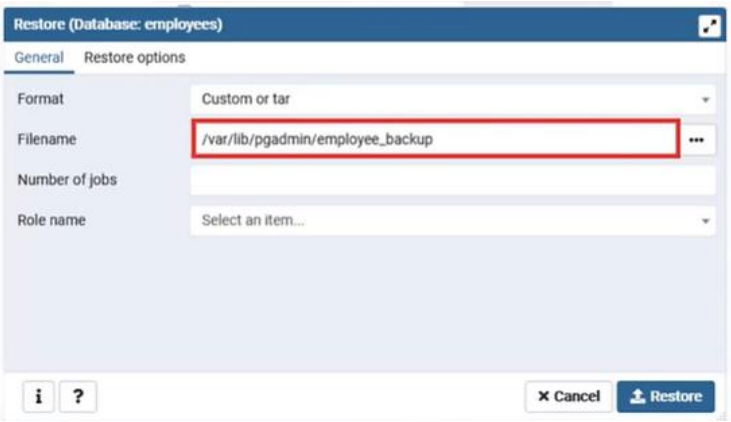
- Enter the name for the new database, and then click Save.




**Restoring database in pgAdmin –**

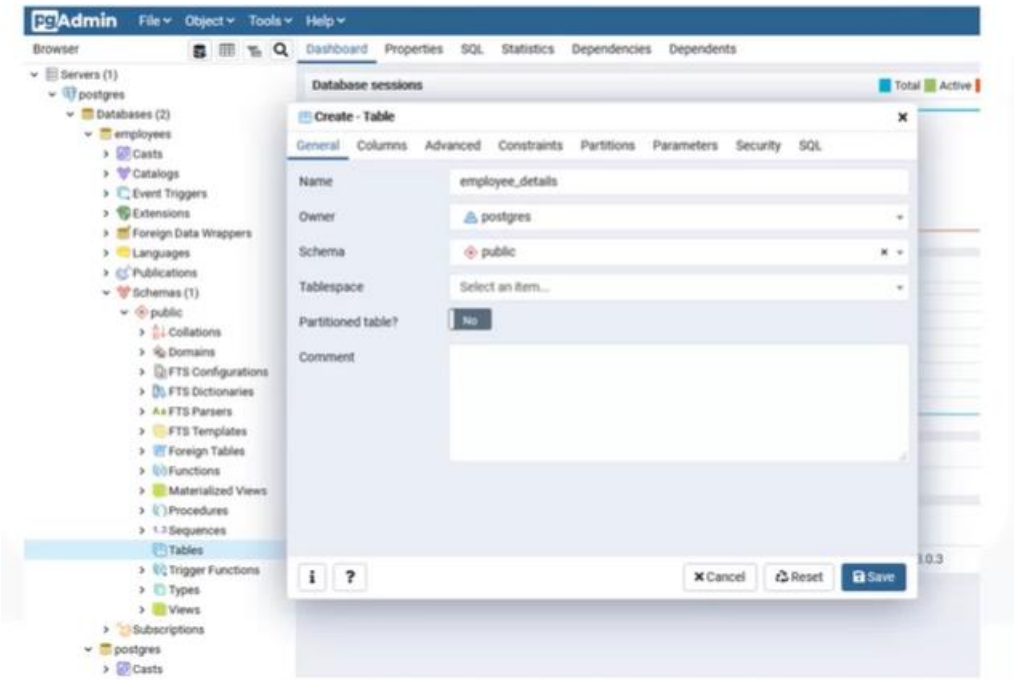
- If you want to restore the whole database from a dump file, select the database that you want to restore to in the treeview, click Restore, and then in the Restore box, enter the location of your dump file. This runs the SQL statements contained in that file to recreate the database objects and data in the current database.



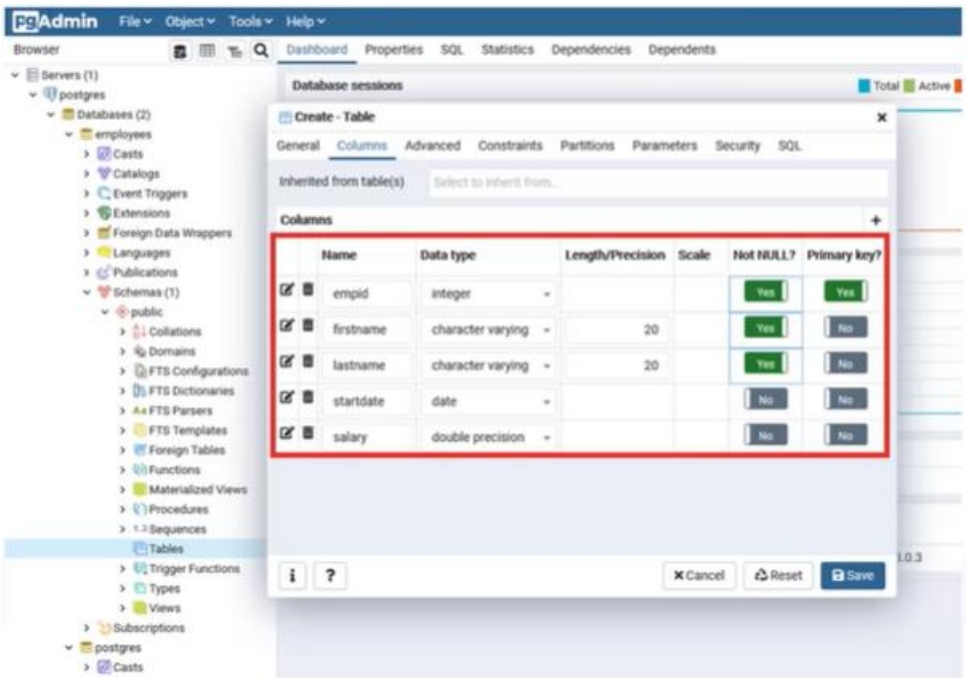
**Creating a table in pgAdmin: –**

- Alternatively, you can manually create tables using pgAdmin. In the treeview, right-click Tables, click Create, and then click Table. On the General page, enter the name for your table, in this case, employee_details.




**Defining Columns -**

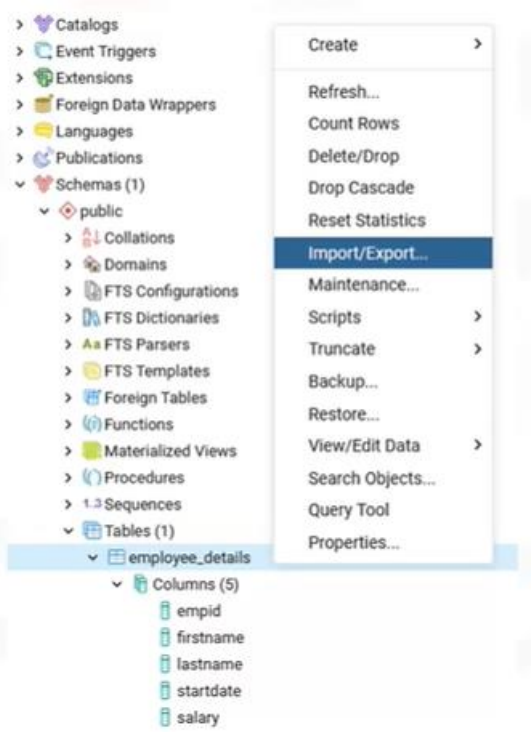
- And then on the Columns tab, enter the details for your columns, and then click Save.



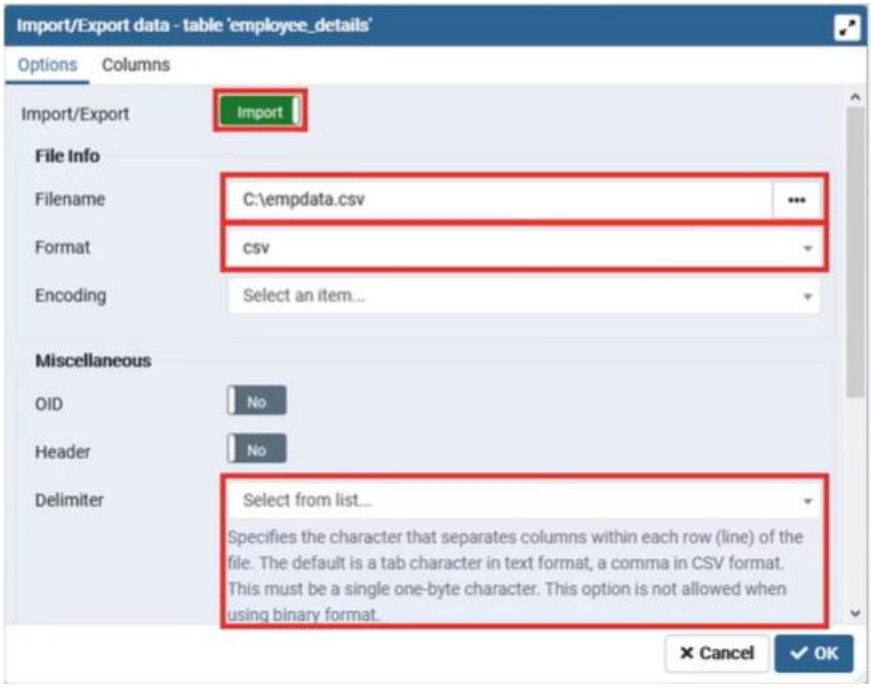
- You'll then see your table and columns in the tables section of the treeview pane.


**Loading data –**

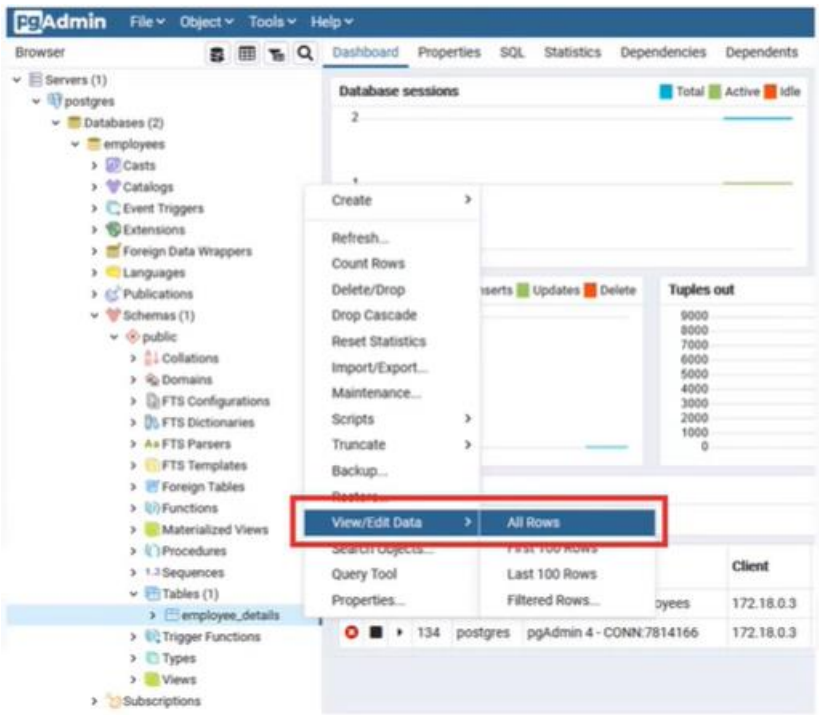
- From tables section of the treeview pane, you can use the Import/Export function to load data into your new table.



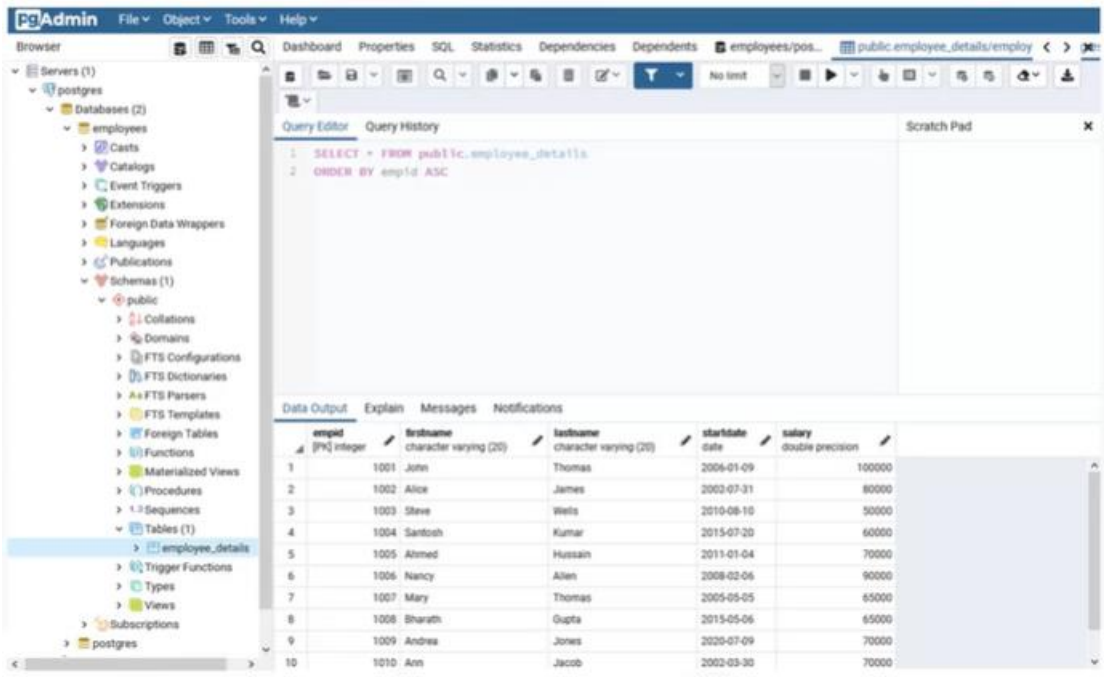
- In the Import/Export data box, select Import, and then enter the location and filename for your data file. If you're loading data from a CSV file, you don't need to specify the delimiter because it's the default option for CSV files.




- You can then use the View/Edit Data option to review the loaded data.

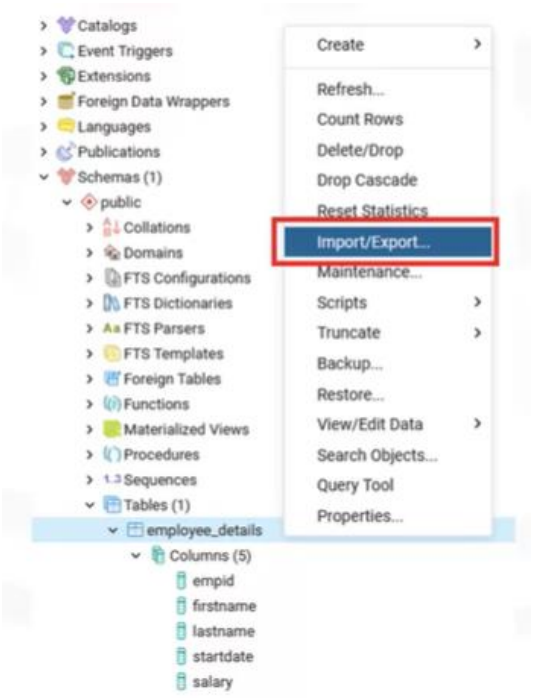


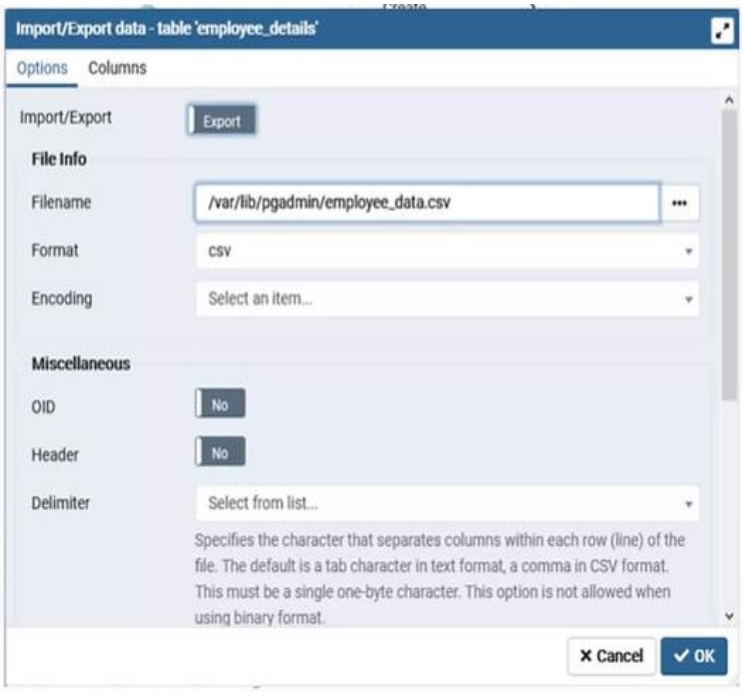
- This runs an SQL query to display the data in the selected table.




**Exporting data –**

- You can also use the Import/Export function to export existing data in your database to a CSV file for use elsewhere. CSV is the default format, so you can just specify the filename and then click OK.



4. **Using pg_dump**

- If you want to back up the entire database, you can use the pg_dump utility.

```
pg_dump employees > employeesbackup.sql
```

  - **employees:** the database name
  - **employeesbackup.sql:** the filename for the dump file

- By default, pg_dump creates an SQL file containing a script that describes all of the objects and data in the database. You can customize the command to output to a compressed archive file instead. By default, pg_dump backs up the entire schema and data in the database.
- By default, pg_dump backs up:
  - Schema
  - Data

**Summary: -**

- You can use many tools to create PostgreSQL objects, including the psql command line utility and pgAdmin.
- You can use pg_dump to back up databases and psql to restore them.
- You can use the pgAdmin Import/Export tool to load data into and export data from tables.

## *3] Views*

**What is a view?**

- A view is an alternative way of representing data from one or more tables or other views.
- Can interact with views in the same way as you interact with tables
- Use to:
  - Limit access to sensitive data
  - Simplify data retrieval
  - Reduce access to underlying tables

| empid | firstname | lastname | startdate | salary |
|---|---|---|---|---|
| 1001 | John | Thomas | 2002-07-31 | 100000 |
| 1002 | Alice | James | 2010-08-10 | 80000 |
| 1003 | Steve | Wells | 2015-07-20 | 50000 |
| 1004 | Santosh | Kumar | 2011-01-04 | 60000 |
| 1005 | Ahmed | Hussain | 2008-02-06 | 70000 |
| 1006 | Nancy | Allen | 2005-05-05 | 90000 |
| 1007 | Mary | Thomas | 2005-05-06 | 65000 |
| 1008 | Bharath | Gupta | 2020-07-09 | 65000 |
| 1009 | Andrea | Jones | 2002-03-30 | 70000 |
| 1010 | Ann | Jacobs | 2006-09-01 | 70000 |

| empid | email |
|---|---|
| 1001 | johnt@ |
| 1002 | alicej@ |
| 1003 | stevew@ |
| 1004 | santoshk@ |
| 1005 | ahmedh@ |
| 1006 | nancya@ |
| 1007 | maryt@ |
| 1008 | bharathg@ |
| 1009 | andreaj@ |
| 1010 | annj@ |

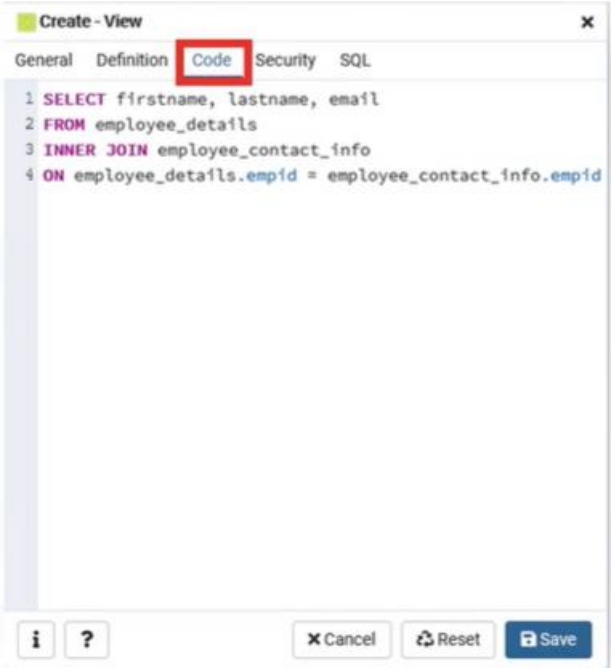
| firstname | lastname | email |
|---|---|---|
| John | Thomas | johnt@ |
| Alice | James | alicej@ |
| Steve | Wells | stevew@ |
| Santosh | Kumar | santoshk@ |
| Ahmed | Hussain | ahmedh@ |
| Nancy | Allen | nancya@ |
| Mary | Thomas | maryt@ |
| Bharath | Gupta | bharathg@ |
| Andrea | Jones | andreaj@ |
| Ann | Jacobs | annj@ |

**Creating a view in pgAdmin –**

- You create views within a schema, so in the tree view on the left-hand side, right-click Views, click Create, and then click View.

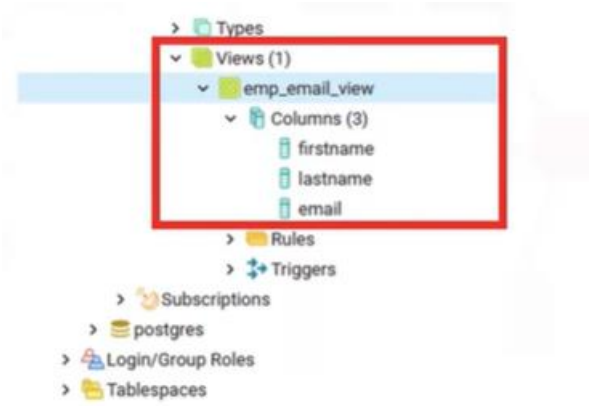

- This opens the Create – View box in which you first need to name the view.

- And then on the Code page, you can enter the SQL code that defines the view, and then click Save.

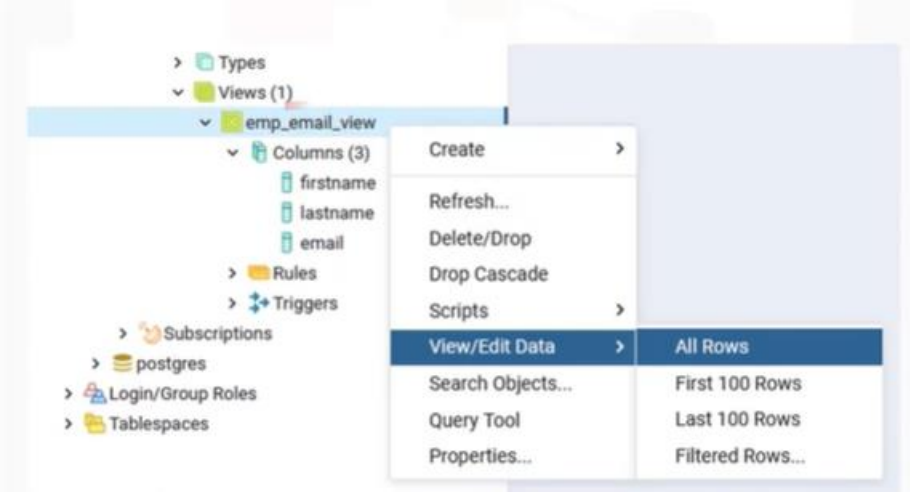



**Using a view in pgAdmin –**

- You will then see your view in the Views folder and you can expand the view to display the columns included in it.



- To run the view, right-click the view name, click View/Edit Data, and then click All Rows.




- You'll then see all of the rows included in the view.

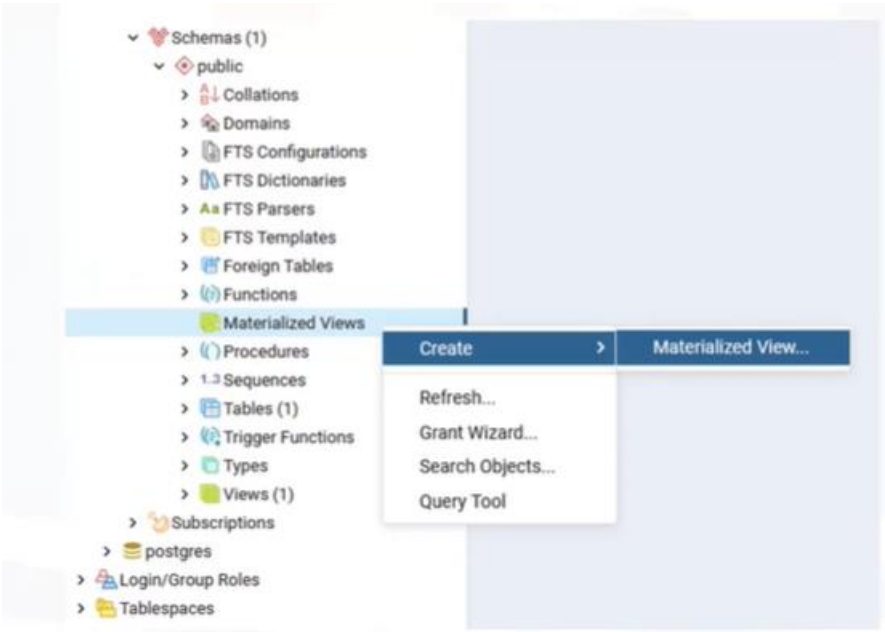

| | firstname character varying (20) | lastname character varying (20) | email character varying (50) |
|---|---|---|---|
| 1 | John | Thomas | johnt@ |
| 2 | Alice | James | alicej@ |
| 3 | Steve | Wells | stevew@ |
| 4 | Santosh | Kumar | santoshk@ |
| 5 | Ahmed | Hussain | ahmedh@ |
| 6 | Nancy | Allen | nancya@ |
| 7 | Mary | Thomas | maryt@ |
| 8 | Bharath | Gupta | bharathg@ |
| 9 | Andrea | Jones | andreaj@ |
| 10 | Ann | Jacob | annj@ |

**Materialized View: -**

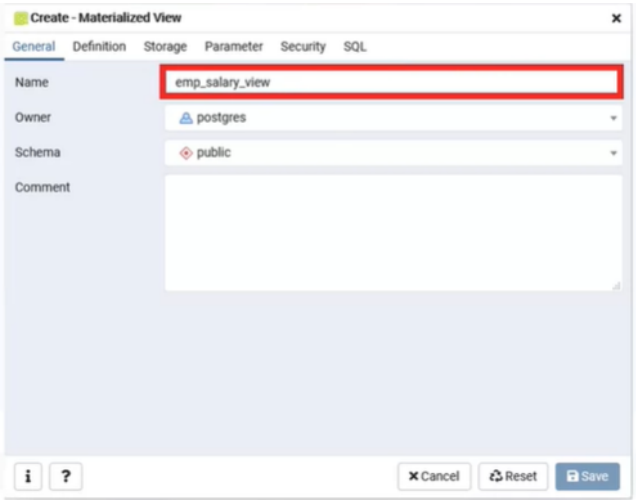
- Behave differently to regular views
- Result set is materialized, or saved, for future use
- Cannot insert, update, or delete rows
- Can improve performance

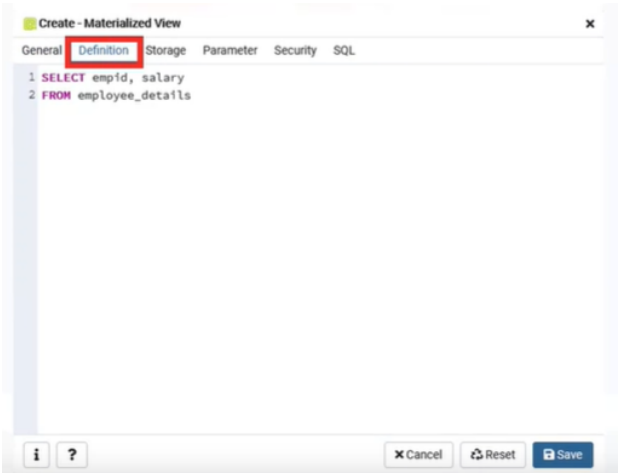
**Creating a materialized view in pgAdmin –**

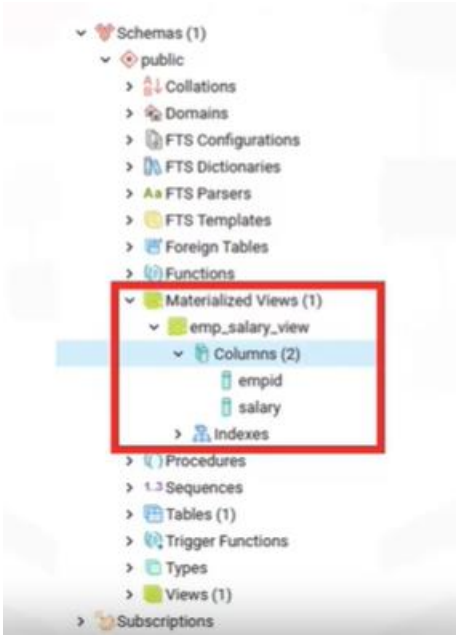
- Creating a materialized view is a similar process to creating a regular view, but you must ensure you start in the materialized views folder in the tree view.




- After entering the name for the view, on the Definition page, enter the code to define the view. Then click Save… … and the materialized view is added to the folder.

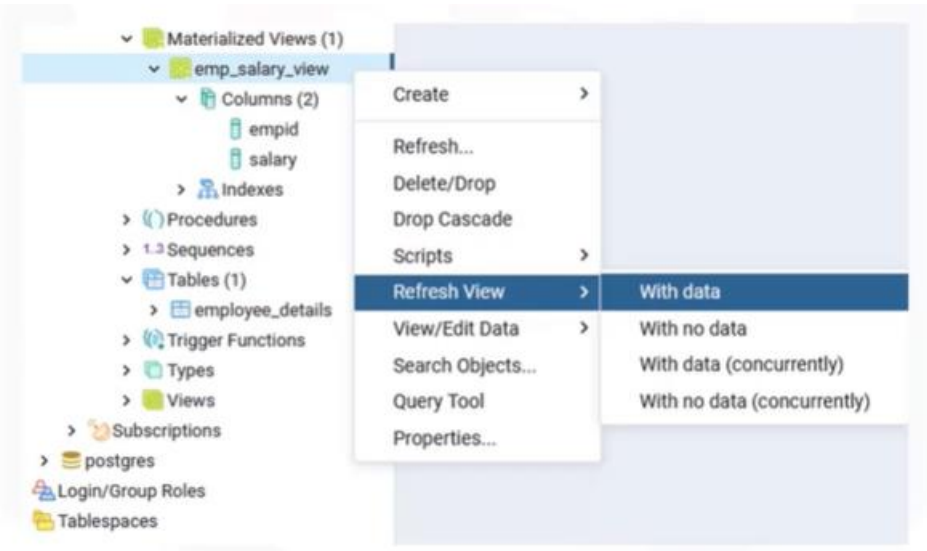




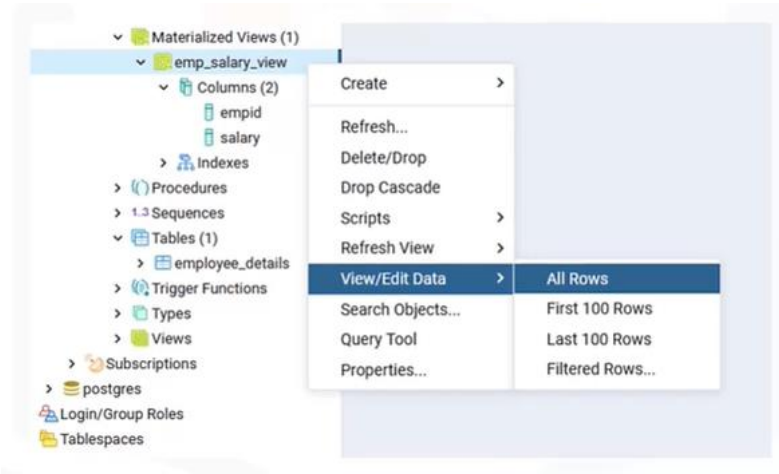

**Refreshing a materialized view in pgAdmin –**

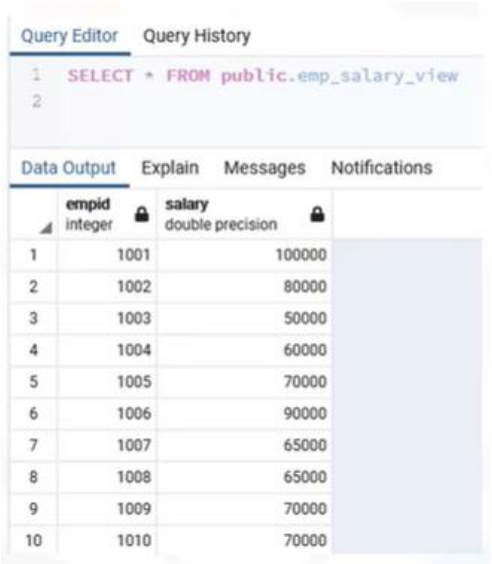
- Because the view stores the data, before using it you need to refresh it with the current rows.



**Using a materialized view in pgAdmin –**

- And then you can use the view to access the data it holds. You can refresh the data in the view at any time to update it with the data in the underlying tables.




**Summary: -**

- Views are an alternative way of representing data.
- You can use views to limit access to sensitive data and simplify data retrieval.
- Materialized views store the result set for quicker subsequent access.
- You cannot insert, update, or delete rows via a materialized view.

## ***Summary & Highlights:-***

- PostgreSQL is an open-source object-relational database management system that you can download and install on your own systems or access on the Cloud.
- You can either self-manage a Cloud instance of PostgreSQL or use a managed services provider, including IBM Cloud Databases for PostgreSQL, Amazon RDS, Google Cloud SQL for PostgreSQL, EnterpriseDB cloud, or Microsoft Azure for PostgreSQL.
- PostgreSQL includes several options for creating databases and tables, loading and querying data, and importing and exporting data relational databases:
  - The psql command line interface. You use this CLI to run SQL statements.
  - pgAdmin. A graphical interface to the database server, which is available as a desktop application or as a web application that you can install on your web servers.
  - Navicat and Dbeaver. Commercial graphical interface options that you can use to access PostgresSQL, MySQL, and other types of databases.
  - Cloud vendor tools and APIs.

- Using pgAdmin, you can:
  - Use pg_dump to back up databases and psql to restore them.
  - Use the Import/Export tool to load data into and export data from tables.

- Using views:
  - You can use views to limit access to sensitive data and simplify data retrieval.
  - Views can be materialized, which means that the view store the result set for quicker subsequent access.
  - Materialized views enhance performance because the view is saved and often stored in memory. However, you cannot insert, update, or delete rows in a materialized view, and they must be refreshed before you can see updated data.

# *Week 4*

## *Course Assignment*

## *Approach to Database Design (Including ERD)*

**Importance of database design: -**

- Crucial to the success of a project
- Contributes to:
  - Integrity of data
  - Reduction of redundancy
  - Performance
  - User satisfaction
- Avoids costly problems

**Database design process: -**

- **Requirements analysis –** where you analyse the data you are working with and the requirements for the use of that data
- **Logical Design –** where you plan how to organise your data
- **Physical Design –** where you plan how to implement your logical design in the database management system you will be using

1. **Requirements analysis: -**

   - During the first stage you gather and analyze real-world business information and policies. You need to identify the base objects in the data and the relationships between these objects.
   - **For example,** in a library scenario, a person borrows a book. You also need to identify the information associated with these objects that you will use to interact with them. For a book, this could be the title, description, ISBN, and authors. And for a person, it could be their name, address, and contact details.
   - How you obtain this information can vary from project to project, but is likely to include:
     - Reviewing any existing stores of this data, be that in a database, other electronic format, or even a paper-based system. Interviewing users to determine how the business currently uses this data.
     - Interviewing users and potential users to determine how the business could make better use of this data. If the data currently resides in a database, when reviewing the existing structure be sure to use it as a source of information about the data rather than a starting template for your own database design.

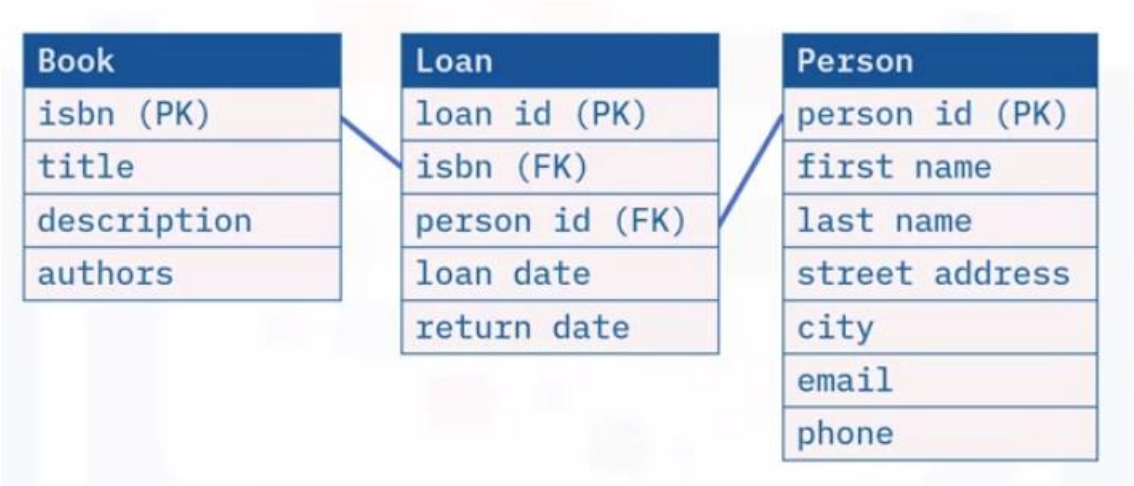
- The output from your requirements analysis could be a report, a data diagram, or a presentation that you can share with stakeholders to validate your understanding of the system.

**2. Logical Design: -**

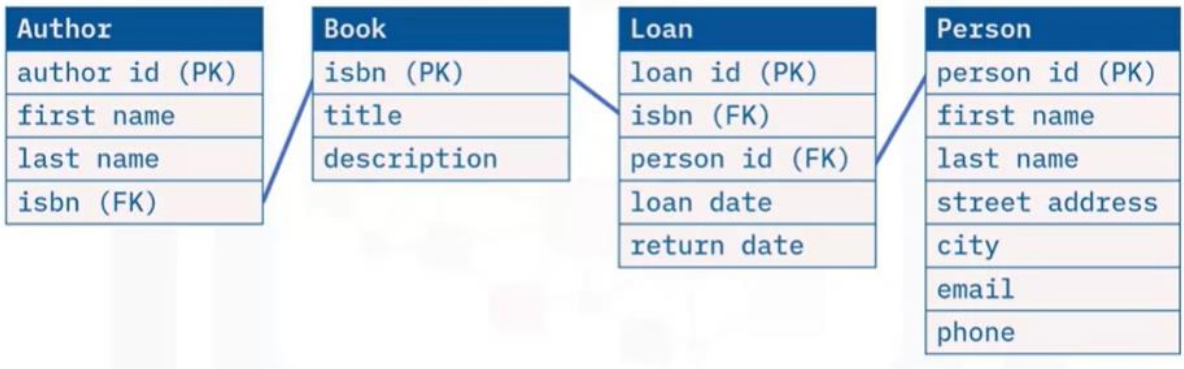
- In the logical design phase, you take the requirements you identified in the analysis stage and map them to entities, attributes, and relationships.
- However, logical models should not specify any technical requirements, so you should not be thinking about any implementation considerations at this stage.
- The objects that you identified in the previous stage become entities. Generally, entities are people, events, locations, or things and if you find you have an object that doesn't fit into one of these categories, you should double-check that it really is an object rather than a characteristic of another object. The characteristics of the objects will become attributes. So, the book object becomes a book entity and the person object becomes a person entity.
- At this stage you should also think about the attributes of your entities.
- **Example,** you may think of a person having a name, but in database terms it is better to think of them having a first name and a last name. This makes it easier to search on and sort by either first or last name. And although the requirements analysis identified that a person has an address, when you think about storing that address, you should divide it up into its constituent parts. When analyzing the requirements, you identified that a person borrows a book. However, a person may borrow many books and a book may be borrowed by many people, so this is a many to many relationships which can lead to ambiguity in a database. The easiest way to solve this is by creating two separate one to many relationships by introducing an associative entity in between the existing ones. In the book to person scenario, you can add in a loan entity. One person can have many loans and one book can be loaned many times. The loan entity will contain attributes from the book table and the person table, as well as some loan-specific ones.
- You could relate these entities by their matching attributes, however none of those are guaranteed to be unique. In the Book entity, the ISBN will be unique, so you could use that as the primary key. You can then use the ISBN instead of the book title in the Loan entity to identify the borrowed book. There are no unique attributes in the person entity, so you can add an identifier attribute to that entity and use that to link the loan to the person. And you can add a loan id to the Loan entity to uniquely identify each loan. Now that you have a view of your entities and attributes, you can consider normalization.

**Normalization: -**

- Most OLTP systems are normalized to the third normal form for optimal transactional performance. Whereas OLAP systems are generally denormalized to enhance read performance.




- To adhere to 1st normal form, you need to remove the potential of two (or more) authors names being listed in the authors attribute for an individual book. One option is to split this into author 1 and author 2 attributes, however there's no guarantee of a maximum number of authors for a book and this would not adhere to 2nd normal form.
- So, a better way is to create a separate authors entity with a many to one relationship to the Book entity.



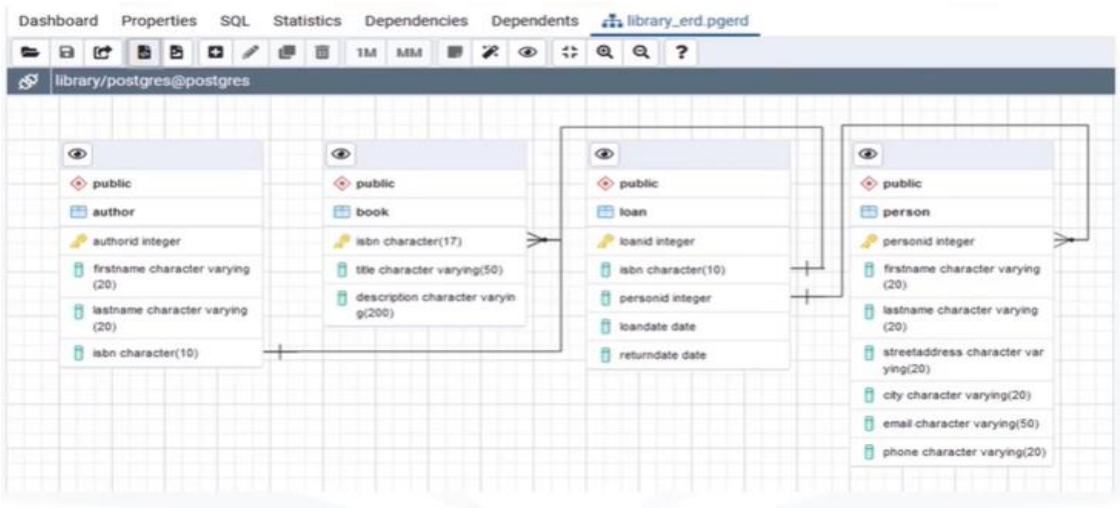
### 3. Physical Design: -

- When you have finished normalizing your entities, you can move on to the physical design stage – how your database will actually look. At this stage, you can start to consider the impact that your choice of database management system will have on your design.
- Impact of database management system:
  - Data types
  - Naming rules
  - Indexes
  - Constraints
- When you are thinking about naming rules, you should also consider implementing your own convention so that anyone working with your data will understand your schema. So, the Person entity from your logical design will become a person table in your physical design with each attribute becoming a typed column and keys being defined.

| Person |
|---|
| person id (PK) |
| first name |
| last name |
| street address |
| city |
| email |
| phone |

| person | | |
|---|---|---|
| personid | integer | PK |
| firstname | character varying (20) | |
| lastname | character varying (20) | |
| streetaddress | character varying (20) | |
| city | character varying (20) | |
| email | character varying (50) | |
| phone | character varying (20) | |

- You can use an ERD designer to create your entity relationship diagrams. pgAdmin includes the ERD Tool in which you can design your ERD and then generate an SQL script that will create your database and objects based on your design.



**Summary: -**

- It is important to spend time designing your database before you start the implementation
- There are three stages of database design: requirements analysis, logical design, and physical design.
- Using an ERD designer can simplify the design process.