2) Object Oriented Databases

What is an Object-Oriented Database(OODB) ?

An object-oriented database (OODB) is a type of database management system (DBMS) that is designed to handle data as ***objects***, rather than as rows and columns in a traditional relational database. In an OODB, data is represented in the form of objects, which can encapsulate both data and the procedures that operate on that data. This approach is based on the principles of object-oriented programming (OOP).

Example of Object-oriented Database:

A Java program that uses an object-oriented database to store information about books. The program has a Book class that represents a book in the database. The class has the following fields:

– title

– author

– publisher.

Comparison with RDBMS:

In a relational database, you might have a table with fields for customer name, customer address, and customer age. In an object-oriented database, you would have a customer object with fields for first name, last name, and age. You could then access the customer's first name by calling the firstName() method on the customer object.

What are the components of an Object-oriented Database?

Here are the components of an Object-oriented Database:

Object: The fundamental unit of data, possessing a unique identifier and storing data in fields.

Class: A blueprint or template for creating objects, defining the structure and methods for object manipulation.

Method: A function associated with a class, used to manipulate the data stored in the fields of an object.

Collection: A container that holds a group of objects, allowing manipulation of the entire group as a single unit.

Design: Object-oriented Database serves as a design pattern, enabling the structuring of data in alignment with object-oriented programming principles.

What are the benefits of using an Object-oriented Database?

The benefits of using an Object-oriented Database include:

Modeling Real-world Entities: OODB provide a natural representation of complex, real-world relationships.

Improved Query Language: OODB often have a query language that supports object-oriented concepts.

Reusability: The classes and methods in an Object-oriented Database can be reused to create other applications.

High performance: ODBMS can provide high performance, especially for applications that require complex data access patterns, as objects can be retrieved with a single query.

Data integrity: ODBMS provides strong data integrity, as the relationships between objects are maintained by the database. This ensures that data remains consistent and correct, even in complex applications.

Concurrency control: ODBMS provides concurrency control mechanisms that ensure that multiple users can access and modify the same data without conflicts.

Scalability: ODBMS can scale horizontally by adding more servers to the database cluster, allowing it to handle large volumes of data

How is an Object-oriented Database implemented?

An object-oriented database is implemented using a programming language that supports object-oriented programming. The most popular languages for this are Java and C++. However, there are also implementations for Python and Ruby

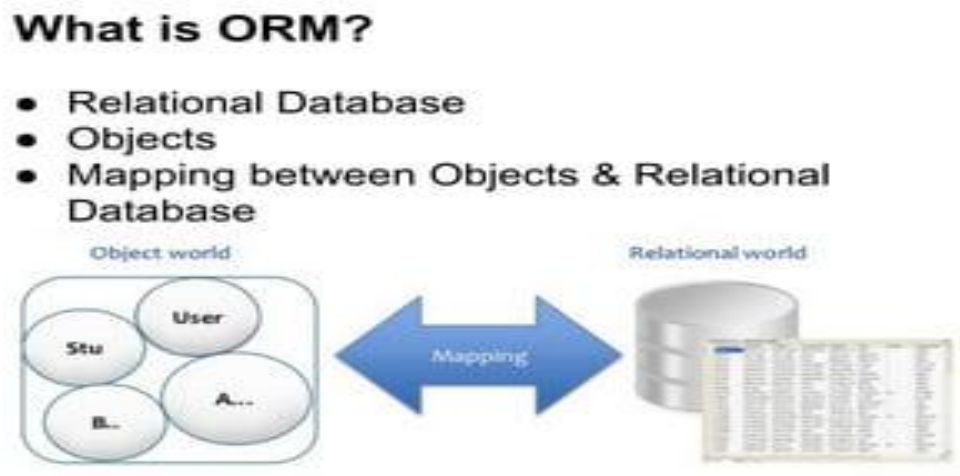
How is a relational database different from an Object-oriented Database?

One key difference between relational and object-oriented databases is their meaning of the word ***"object."*** In a relational database, an object is essentially a ***table row.*** In an object-oriented database, however, an object is a self-contained entity with its own data and methods.

Another key difference between relational and object-oriented databases is their relationships. In a relational database, you can only define relationships between tables.

Disadvantages:

1. Limited Adoption
2. Lack of Standardization
3. Cost:
4. Integration with Other Systems
5. Scalability Challenges



Object Relational Mapping (ORM) : Technique used in creating a "bridge" between *Databases* and *Object-Oriented Programming.*

When interacting with a database using OOP languages, we'll have to perform different operations like creating, reading, updating, and deleting (CRUD) data from a database. By design, we use SQL for performing these operations in relational databases.

Main idea ?

The idea of ORM is based on *abstraction.* The ORM mechanism makes it possible to address, access and manipulate objects without having to consider how those objects relate to their data sources.

Fundamentals of ORM

1. **Mapping:** **Bridging Objects and Tables**
   - ORM involves mapping object-oriented classes to corresponding database tables.
   - Achieves alignment between object-oriented and relational data models.
   - Reduces the need for developers to write raw SQL statements.
2. **Relationship Mapping:**

- Molds objects and classes into structure of database.
- One-to-One, One-to-Many, Many-to-One, and Many-to-Many relationships are commonly supported.
- translation of complex object relationships into relational database structures.

3. Query Abstraction:

- Provides frameworks query languages or methods allowing developers to interact with the database using object-oriented syntax.

What are the Key Components and Features of ORM ?

Relationship Handling:

ORM systems facilitate the management of relationships between objects and tables. This includes one-to-one, one-to-many, and many-to-many rel*ationships, offering a robust foundation for handling complex data structures.*

CRUD Operations:

CRUD operations—Create, Read, Update, and Delete—are fundamental to database interactions. ORM frameworks streamline these operations, offering methods and conventions that developers can use to manipulate data without directly resorting to raw SQL statements.

Lazy Loading:

Lazy loading is a crucial optimization feature in ORM. It defers the loading of related data until it is explicitly requested by the application, minimizing unnecessary database queries and improving overall performance.

Abstraction of Database Complexity:

Developers can interact with the database using a higher-level, object-oriented syntax, abstracting away the complexities of raw SQL queries. This results in a more intuitive and maintainable codebase.

Object-Oriented Modeling:
ORM aligns well with the principles of object-oriented programming, providing a natural and intuitive way to model and interact with data in modern applications.

What are the Challenges and Considerations ?

Performance Overhead:
ORM may introduce some performance overhead, especially in scenarios where fine-tuned control over SQL queries is required. Developers should be mindful of optimizing critical paths.

Learning Curve:
There can be a learning curve associated with mastering the intricacies of a specific ORM framework. However, the long-term benefits often outweigh the initial investment in learning.

Complex Mappings:
Handling complex database schemas and intricate relationships may require careful consideration and planning when designing the mapping between objects and database tables.

Pros

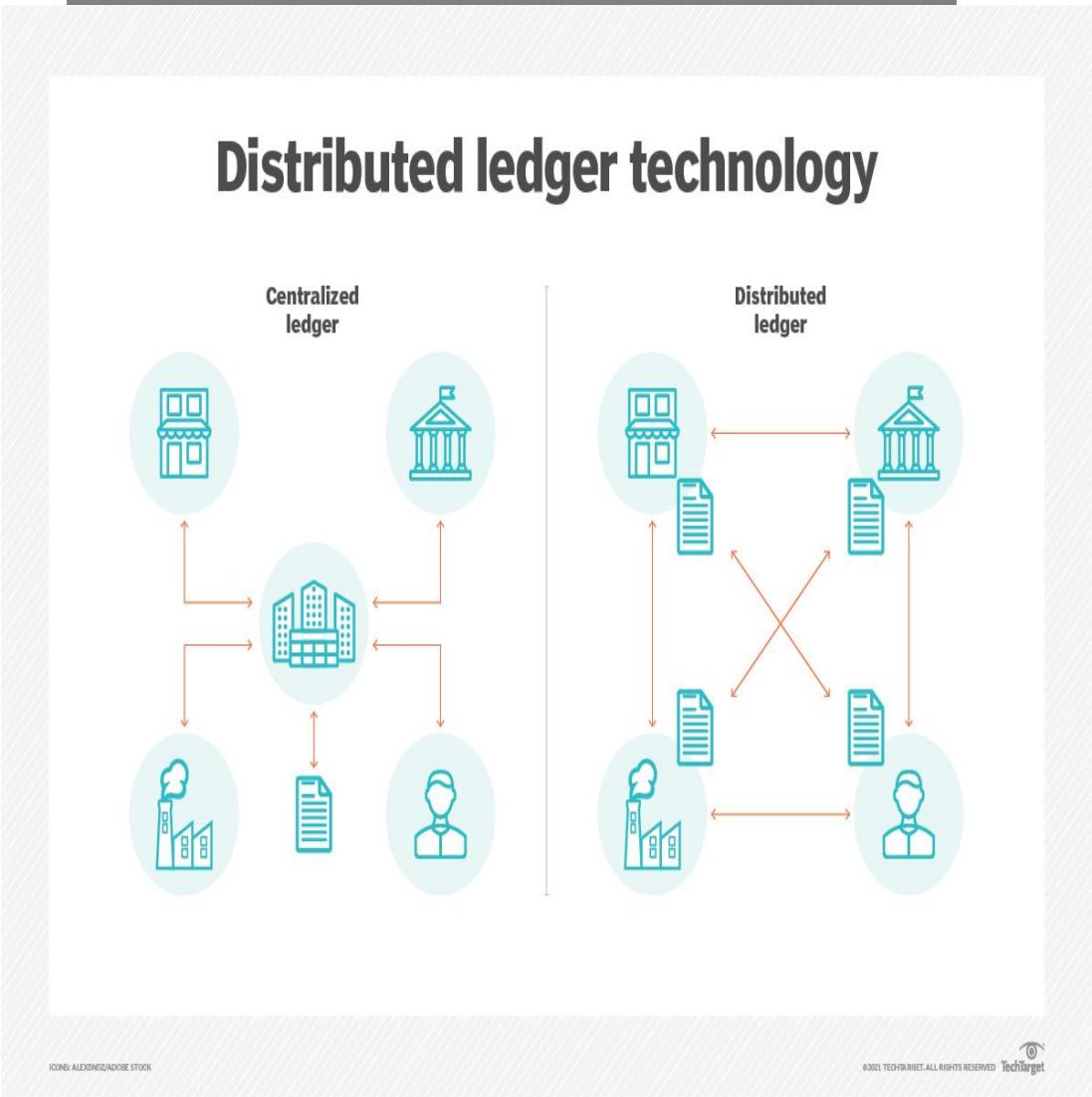
- It is easier to query using ORM than MySQL.
- Hides the complex processes under-the-hood and only lets you worry about the high-level implementation.
- There is no need to change the code if the underlying database changes.

Cons

- Learning an ORM from scratch is time-consuming.
- You cannot directly dive into an ORM without learning MySQL. Even though the ORM helps you abstract the complex details, it is pertinent to know the consequence of each command under-the-hood.

- You may run into performance issues while coding complex queries using an ORM.

DISTRIBUTED LEDGER TECHNOLOGY



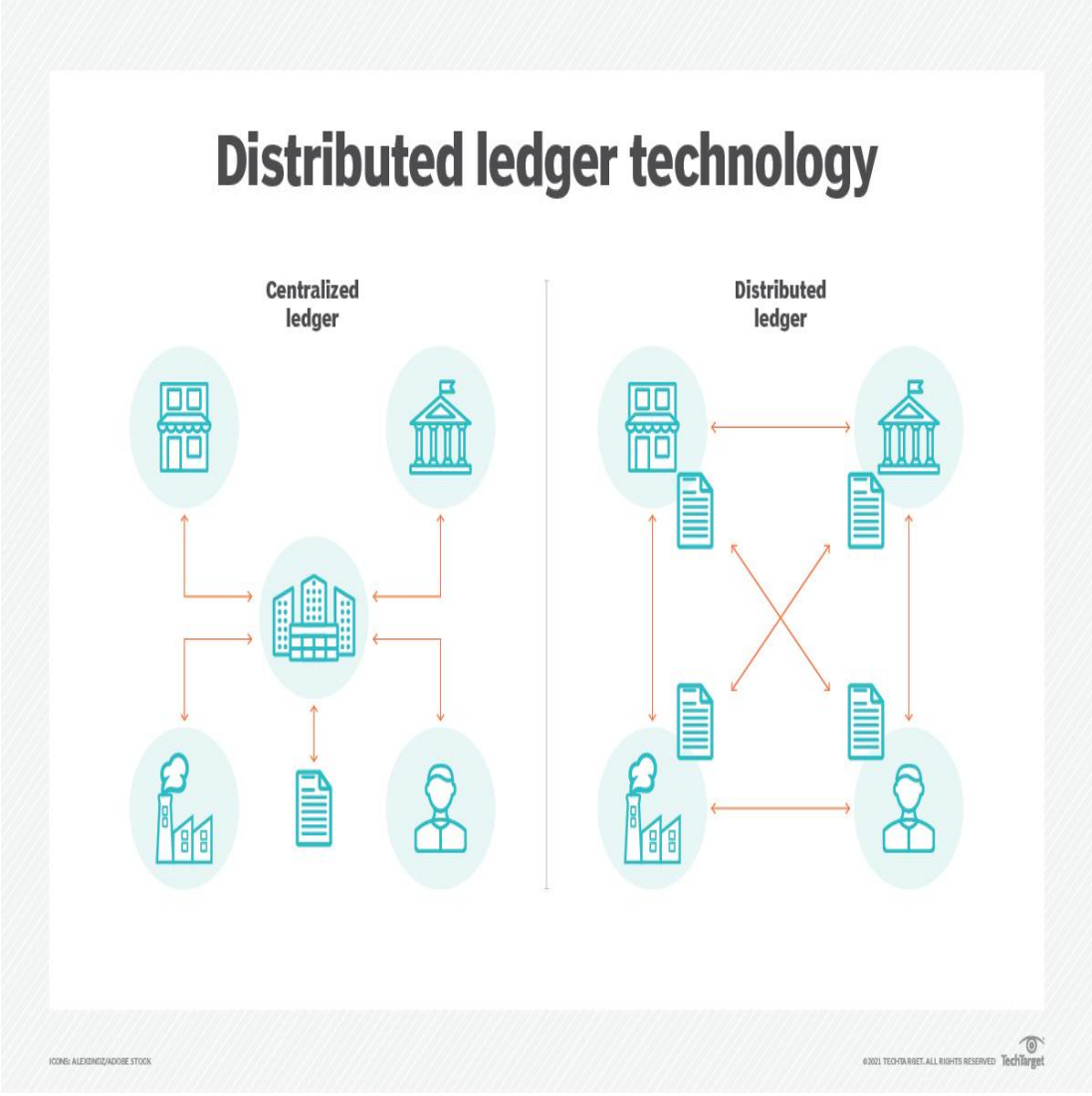
What is distributed ledger technology (DLT)?

Distributed ledger technology (DLT) is a digital system that stores, records, and shares data across multiple nodes in a network.

Unlike traditional centralized systems, where data is stored in a single location, DLTs leverage a decentralized architecture, which ensures that data is distributed across the network. This means that there is no central point of control, and no single entity can manipulate or alter the data. manipulate or alter the data.is no central point of control, and no single entity can manipulate or alter the data.

SOME KEY POINTS

1. **Network of Nodes**
2. **Evolution Since Bitcoin:**
3. **Versatile Industry Applications:**
4. **Blockchain Relationship:**
5. **Enhancements and Challenges:**



How do distributed ledgers work?

DLT works based on principles of decentralization. Unlike traditional centralized databases, DLT operates on a peer-to-peer (P2P) network, where multiple nodes store, validate and update the ledger simultaneously. This eliminates the need for a central authority and reduces the risk of a single point of failure.

BLOCKCHAIN