Object Oriented DBMS Architecture

Unit -3

Sub-unit-3.3

Performance Issue in OODBMS

1) Better Support for Complex Applications

- ODBMSs have provided better support for certain applications requiring high performance and modeling of complex relationships and heavily inter-related data that have traditionally not been well served by other DBMSs.

2)Enhance Programmability

- The seamless integration of an application programming language and database DDL/DML enables further savings in code.

Performance Issue in OODBMS

3)Improve Performance

- An ODBMS has more "knowledge" of the operations being performed, and can therefore provide better control of concurrently executing transactions. This leads to better performance.

4) Improve Navigational Access

- Using declarative query languages has proved to be very useful in relational systems.
- However, users should not be constrained in the way they interact with a database. It should be possible to support a variety of interaction styles (e.g. natural language, menus, prompts, graphical browsers, etc.) that can all be used to augment the formal command language of the database. In any case, there are many structures for which a navigational mode of access is more natural and intuitive than using a declarative query language.

Performance Issue in OODBMS

5)Simplify Concurrency Control

- Detailed locking strategies for some ODBMSs provide highly sophisticated control of objects (e.g. ORION).
- At a coarse level of granularity, it is possible to place a single lock on an entire object hierarchy.
- In contrast, an RDBMS would require multiple locks to get information from related but scattered data in the database.
- At a fine level of granularity, individual objects or attributes of objects could be locked.

Performance Issue in OODBMS

6) Reduce Problems of Referential Integrity

- One of the major problems with relational databases has been that of dangling references.
- This problem has been solved in object databases by giving the responsibility of pointer maintenance to the database itself, rather than to each application.

7) Abstraction Gain

- The object paradigm provides a better modeling framework that captures not only data and behavior, but raises the level of abstraction (an object is more than the sum of its attributes and methods).

Application Selection for Object oriented DBMS

- Basically, an OODBMS is an object database that provides DBMS capabilities to objects that have been created using an object-oriented programming language (OOPL).
- The basic principle is to add persistence to objects and to make objects persistent.
- Consequently application programmers who use OODBMSs typically write programs in a native OOPL such as Java, C++ or Smalltalk, and the language has some kind of Persistent class, Database class, Database Interface, or Database API that provides DBMS functionality as, effectively, an extension of the OOPL.

Application Selection for Object Oriented DBMS

- Object-oriented DBMSs, therefore, support advanced object-oriented database applications with features like support for persistent objects from more than one programming language, distribution of data, advanced transaction models, versions, schema evolution, and dynamic generation of new types.
- In today's world, Client-Server applications that rely on a database on the server as a data store while servicing requests from multiple clients are quite commonplace.
- Most of these applications use a Relational Database Management System (RDBMS) as their data store while using an object oriented programming language for development.

Application Selection for Object Oriented DBMS

- This causes a certain inefficency as objects must be mapped to tuples in the database and vice versa instead of the data being stored in a way that is consistent with the programming model.
- An OODBMS is thus a full scale object oriented development environment as well as a database management system. Features that are common in the RDBMS world such as transactions, the ability to handle large amounts of data, indexes, deadlock detection, backup and restoration features and data recovery mechanisms also exist in the OODBMS world.
- A primary feature of an OODBMS is that accessing objects in the database is done in a transparent manner such that interaction with persistent objects is no different from interacting with in-memory objects.
- An object-oriented database management system (OODBMS), sometimes shortened to *ODBMS* for *object database management system*), is a database management system (DBMS) that supports the modelling and creation of data as objects.

Database Design for Object oriented DBMS

- Idea is to not use a DBMS (Relational or Object-Oriented.)
- Instead, design data management classes which handle persistence, caching, etc.
- These classes decouple applications from their persistent storage.
- Use data management classes whenever you need to:
  - ✓ Store an application object;
  - ✓ Search for or retrieve stored objects;
  - ✓ Interface with an external database.
  - ✓ This solution won't work for large data sets!

Database Design for Object oriented DBMS

Data Storage layer:

- Options for locating the operations that handle the tasks of storing and retrieving objects:
- All persistent objects in the system could inherit methods for storage from an abstract superclass i.e.PersistentObject
- Introduce separate classes into the system whose role is to deal with the storage and retrieval of other classes i.e. (Database broker approach)

Database Design for Object oriented DBMS

1. PersistentObject Superclass Approach

- A superclass PersistentObject encapsulates the mechanisms for an object of any class to store itself in, or retrieve itself from a database.
- This superclass implements operations to get an object by object identifier, store, delete and update objects and to iterate through a set of objects (write and read operations).

2. *Database Broker Approach*

- Each persistent class could be responsible for its own storage...but

Database Design for Object oriented DBMS

- ✓highly coupled (to storage mechanism);
- ✓ lessens class cohesion;
- ✓ class must now have expert knowledge of storage tasks;
- ✓ these are unrelated to application tasks.

- Separates the business objects from their data storage implementation.
- The classes that provide the data storage services will be held in a separate package.
- For each business class that needs to be persistent, there will be an associated database broker class.

Database Design for Object oriented DBMS

The Broker Class:

- The broker class provides the mechanisms to materialize objects from the database and dematerialize them back to the database.

The Database Broker:

- The database broker object is responsible for:
  - ✓"materialising" objects,
  - ✓"dematerialising" objects,
  - ✓caching objects.
- Application classes are insulated from storage.
- Allows migration of storage sub-systems, e.g., implement on existing relational system.
- Replace this with OODBMS.
- Application programs unaffected by change.

Database Design for Object oriented DBMS

Caching Objects

- Objects can be cached for efficiency.
- The cache is a collection maintained by the database broker.
- When an object is requested, the cache is searched first.
- If the object sought is not in the cache it is materialised by the database broker from the database.

Collections

- In systems where collection classes are used in design, these may be replaced by database broker objects.
- Database objects provide collection-like services for large volumes of data (more than you would maintain in a collection class in memory).

Object identifier

- An object is defined by a triple (OID, type constructor, state) where OID is the unique object identifier, type constructor is its type (such as atom, tuple, set, list, array, bag, etc.) and state is its actual value.
- Example:
  (i1, atom, 'John')
  (i2, atom, 30)
- An object has structure or state (variables) and methods (behavior/operations)
- An object is described by four characteristics
  - ✓Identifier: a system-wide unique id for an object
  - ✓Name: an object may also have a unique name in DB (optional)
  - ✓Lifetime: determines if the object is persistent or transient
  - ✓Structure: Construction of objects using type constructors

Object identifier

Using an oid to refer to an object is similar to using a foreign key to refer to a record in another relation, but not quiet the same:

- An **oid** can point to an object of theater_t that is stored anywhere in the database, even in a field
- Whereas a **foreign key** reference is constrained to point to an object in a particular reference relation.

Storage and Access Methods

- The main purpose of a DBMS was to centralize and organize data storage. A DBMS program ran on a single, large machine.
- Data is often distributed among many systems, which is the consequence of an external program without changing the code in any way, or even recompiling it.
- But making such system scalable requires using other features of the ORDBMS: the distributed database functionality, commercially available gateways, and the open storage manager . Combining these facilities provides the kind of *location transparency necessary for the development of distributed information* systems.

Storage and Access Methods

- ORDBMSs possess storage manager facilities similar to RDBMSs.
- Disk space is taken under the control of the ORDBMS, and data is written into it according to whatever administrative rules are specified.
- All the indexing, query processing, and cache management techniques that are part of an RDBMS are also used in an ORDBMS.
- Further, distributed database techniques can be adapted to incorporate user-defined types and functions. However, all of these mechanisms must be re-implemented to generalize them so that they can work for user-defined types.

Storage and Access Methods

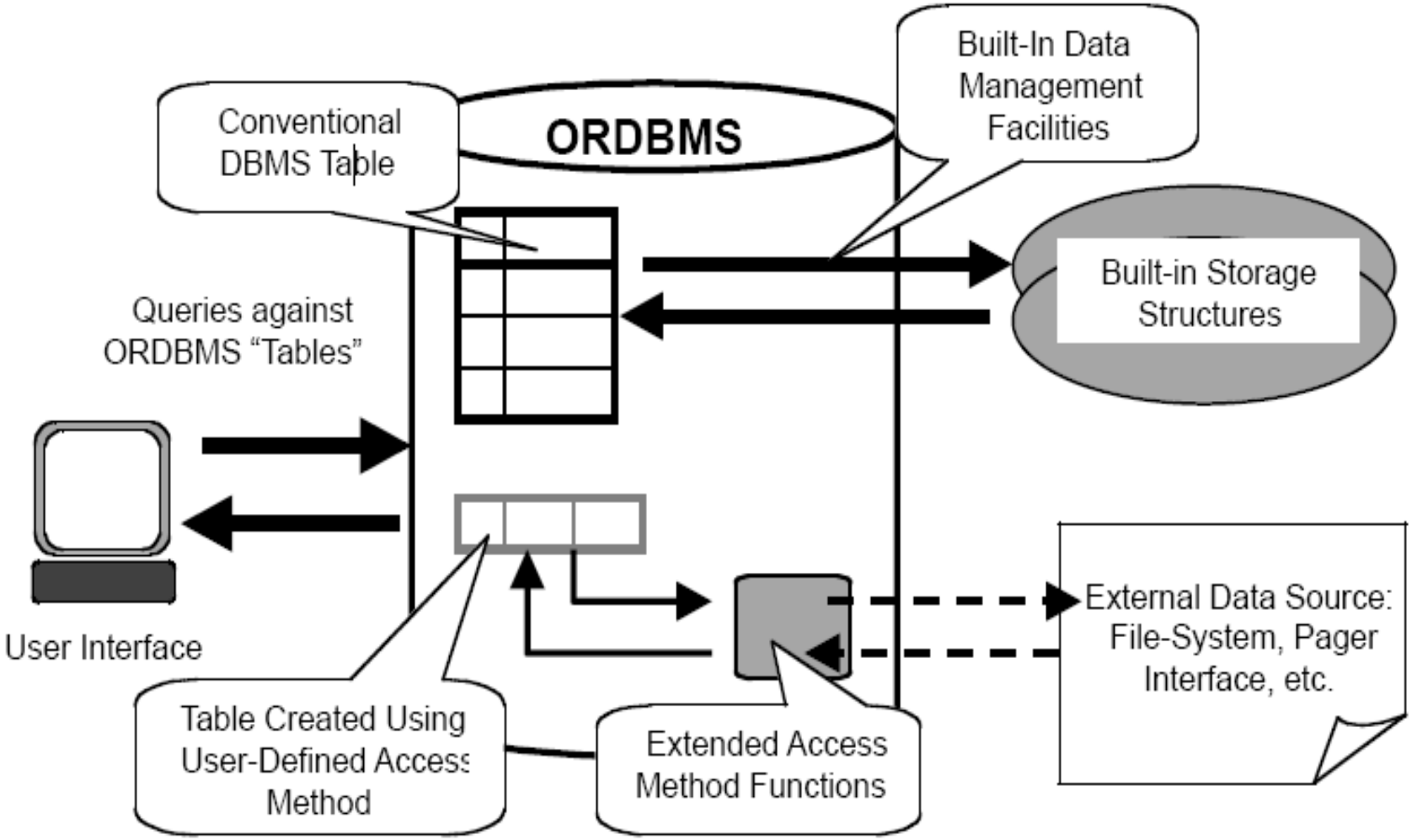


Figure 1–6. *Extensible Storage Manager*

New Data Types

- User-define abstract data type (ADT)
  - ✓ ADT include image, voice, and video footage, and these must be stored in the database - Must write compress functions to support (lower resolution).
  - ✓ ORDBMS key feature is allowing users to define arbitrary new data type. Such as:
    - compress, rotate, shrink and crop
  - ✓ Combination of an atomic data type and its associated methods is called: Abstract data type (ADT). Such object-relational systems, user allows to include ADT.

- Structured types
  - In this application, as needed in many traditional business data processing applications, we need new types built up from atomic types using constructors for creating sets, records, arrays, sequences, and so on..

New Data Types

- Structured types

✓ ROW

A type representing a row, or record, of n field with fields n of type n….etc

✓ Listof(base)

Type representing a sequence of base-type items

✓ Array(base)

A type representing an array of base-type items

✓ Setof(base)

A type representing a set of base-type items. Sets cannot contain duplicate elements.

✓ bagof(base)

A type resenting a bag or multiset of based-type items

Query Processing

- It was commonly believed that the application domains that OODBMS technology targets do not need querying capabilities.
- The issues related to the *optimization and execution of OODBMS query languages* which we collectively call *query processing*.

Query Processing Architecture

two architectural issues:

- the query processing methodology and
- the query optimizer architecture.

Query Processing Methodology

- A query processing methodology similar to relational DBMSs, but modified to deal with the difficulties can be followed in OODBMSs.

Steps of methodology are

- Queries are expressed in a declarative language which requires no user knowledge of object implementations, access paths or processing strategies.
- The calculus expression is first_reduced to a normalized form by eliminating duplicate predicates, applying identities and rewriting.

Query Processing Architecture

- The normalized expression is then converted to an equivalent object algebra expression.
- This form of the query is a nested expression which can be viewed as a tree whose nodes are algebra operators and whose leaves represent extents of classes in the database.
- The algebra expression is next checked for type consistency to insure that predicates and methods are not applied to objects which do not support the requested functions.
- This is not as simple as type checking in general programming languages since intermediate results, which are sets of objects, may be composed of heterogeneous types.
- The next step in query processing is the application of equivalence-preserving rewrite rules to the type consistent algebra expression. Lastly, an execution plan which takes into account object implementations is generated from the optimized algebra expression.

Query Processing Architecture

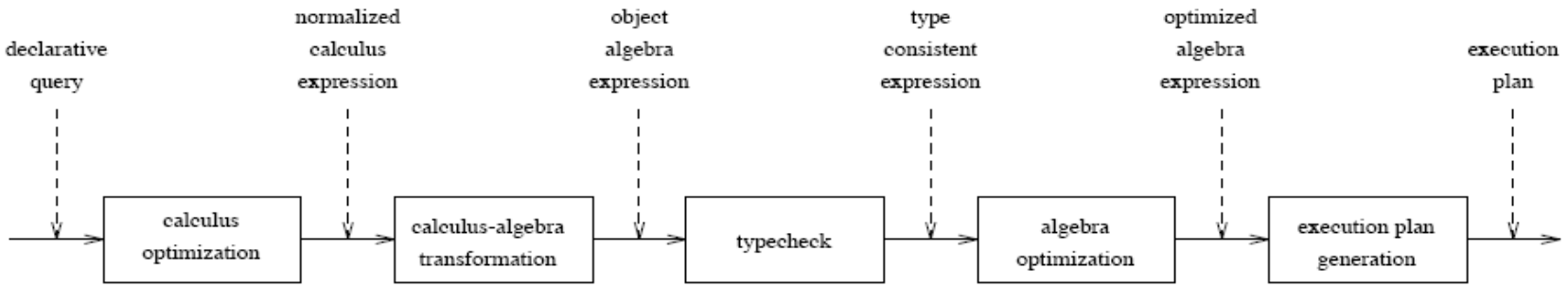


Figure 1: Object query processing methodology

Optimizer Architecture

- Query optimization can be modeled as an optimization problem whose solution is the choice of the “optimum” *state in* a *state space (also called search space). In query optimization, each state corresponds to an algebraic query indicating* an execution schedule and represented as a processing tree.
- The state space is a family of equivalent (in the sense of generating the same result) algebraic queries.
- Query optimizers generate and search a state space using a *search strategy applying a cost function to each state and finding one with minimal cost1.*

Optimizer Architecture

Thus, to characterize a query optimizer three things need to be specified:

1. the search space and the the transformation rules that generate the alternative query expressions which constitute the search space;
2. a search algorithm that allows one to move from one state to another in the search space; and
3. the cost function that is applied to each state.

Optimization Techniques

The optimization techniques for object queries follows two fundamental directions.

1. The first is the **cost-based optimization of queries based on algebraic manipulations.** Algebraic optimization techniques have been studied quite extensively within the context of the relational model.
2. The second is the optimization of ***path expressions.***

   Path expressions represent traversal paths between objects and are unique to OODBMSs. The optimization of path expressions is an important issue in OODBMSs and has been a subject of considerable investigation.

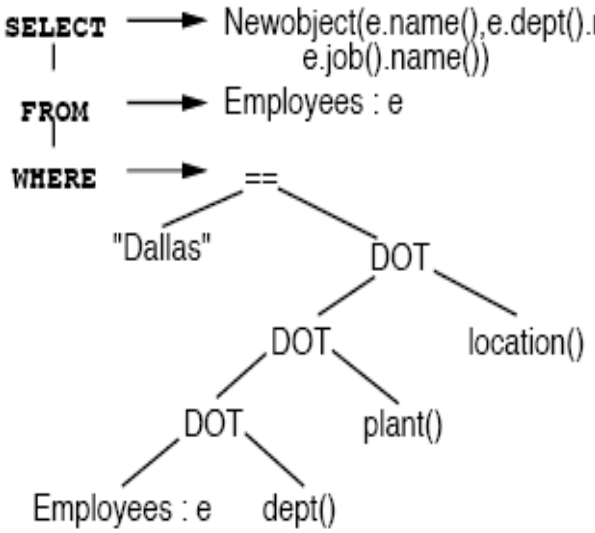
Optimization Techniques

1. **Algebraic Optimization**
   - ✓Search Space and Transformation Rules
   - ✓Search Algorithm
   - ✓Cost Function
   - ✓Parameterization
2. **Path Expressions**
   - ✓**Rewriting**
   - ✓**Algebraic optimization**
   - ✓**Path indexes**

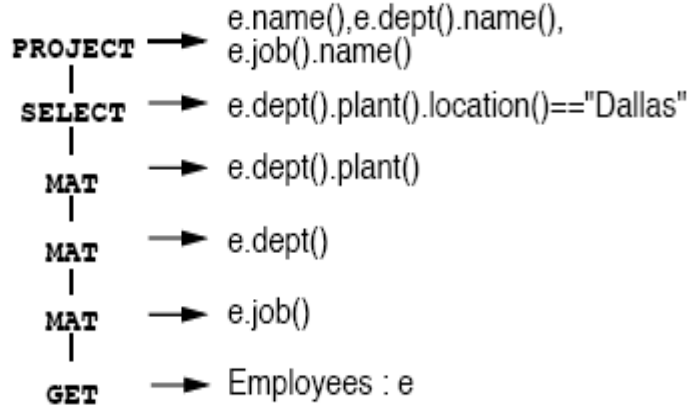
Optimization Techniques

```
SELECT Newobject(e.name(),e.dept().name(),
                 e.job().name())
FROM Employee e IN Employees
WHERE e.dept().plant().location()=="Dallas";
```

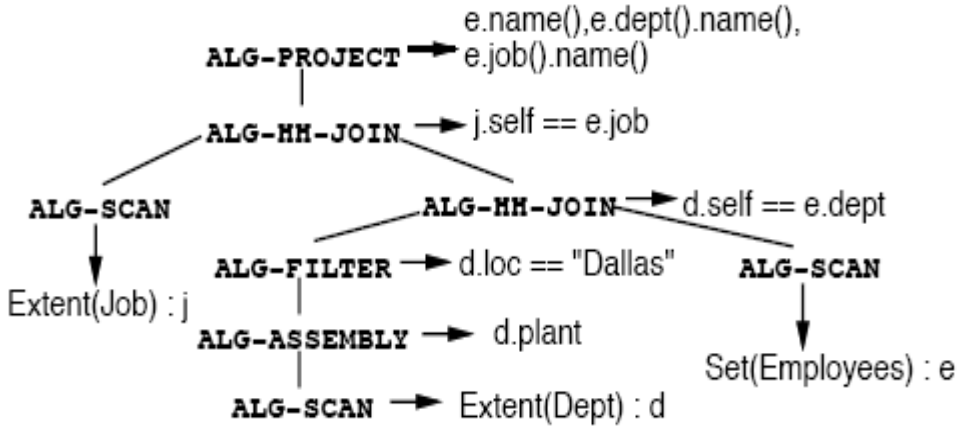
(a) User Query.



(b) After Parsing.



(c) After Rewriting.



(d) After Optimization.

Query Execution

- The relational DBMSs benefit from the close correspondence between the relational algebra operations and the access primitives of the storage system.
- Therefore, the generation of the execution plan for a query expression basically concerns the choice and implementation of the most efficient algorithms for executing individual algebra operators and their combinations.
- In OODBMSs the issue is more complicated due to the difference in the abstraction levels of behaviorally defined objects and their storage.
- Encapsulation of objects, which hides their implementation details, and the storage of methods with objects.
- A query-execution engine requires three basic classes of algorithms on collections of objects: (collection) *scan, indexed scan, and collection matching. Collection scan is a straightforward algorithm that sequentially accesses all* objects in a collection.

ODBC

- **Open Database Connectivity** (**ODBC**) is a standard application programming interface (API) for accessing database management systems (DBMS). The designers of ODBC aimed to make it independent of database systems and operating systems.
- An application written using ODBC can be ported to other platforms, both on the client and server side, with few changes to the data access code.
- ODBC accomplishes DBMS independence by using an *ODBC driver* as a translation layer between the application and the DBMS.
- The application uses ODBC functions through an *ODBC driver manager* with which it is linked, and the driver passes the query to the DBMS.
- ODBC provides a vendor-independent database-access API. The ODBC API has been wildly successful as a database-access standard. Virtually all products that require database access support it, and the most recent ODBC driver supports the SQL Server 7.0 enhancements. You use a call-level interface (CLI) to implement the ODBC API.

ODBC(Components of odbc)

- **ODBC API**

  A library of function calls, a set of error codes, and a standard SQL syntax for accessing data on DBMSs.

- **ODBC Driver Manager**

  A dynamic-link library (Odbc32.dll) that loads ODBC database drivers on behalf of an application. This DLL is transparent to your application.

- **ODBC database drivers**

  One or more DLLs that process ODBC function calls for specific DBMSs. For a list of supplied drivers, see ODBC Driver List.

- **ODBC Cursor Library**

  A dynamic-link library (Odbccr32.dll) that resides between the ODBC Driver Manager and the drivers and handles scrolling through the data.

- **ODBC Administrator**

  A tool used for configuring a DBMS to make it available as a data source for an application.

Db library

- DB-Library provides source code compatibility for older Sybase applications. Sybase encourages programmers to implement new applications with Client-Library or Embedded SQL.
- DB-Library/C includes C routines and macros that allow an application to interact with Adaptive Server and Open Server applications.
- It includes routines that send commands to Adaptive Server and Open Server applications and others that process the results of those commands. Other routines handle error conditions, perform data conversion, and provide a variety of information about the application's interaction with a server.
- DB-Library/C also contains several header files that define structures and values used by the routines. Versions of DB-Library have been developed for a number of languages besides C, including COBOL, FORTRAN, Ada, and Pascal.

DAO(Data Access Objects)

- In computer software, a **data access object** (**DAO**) is an object that provides an abstract interface to some type of database or other persistence mechanism.
- By mapping application calls to the persistence layer, the DAO provides some specific data operations without exposing details of the database.
- This isolation supports the Single responsibility principle. It separates what data access the application needs, in terms of domain-specific objects and data types (the public interface of the DAO), from how these needs can be satisfied with a specific DBMS, database schema, etc. (the implementation of the DAO).
- it is traditionally associated with Java EE applications and with relational databases (accessed via the JDBC API)

DAO(Data Access Objects)

- The advantage of using data access objects is the relatively simple and rigorous separation between two important parts of an application that can but should not know anything of each other, and which can be expected to evolve frequently and independently.
- Changing business logic can rely on the same DAO interface, while changes to persistence logic do not affect DAO clients as long as the interface remains correctly implemented. All details of storage are hidden from the rest of the application (information hiding).
- DAOs act as an intermediary between the application and the database. They move data back and forth between objects and database records.

JDBC(Java database connectivity)

- The Java Database Connectivity (JDBC) API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases SQL databases and other tabular data sources, such as spreadsheets or flat files. The JDBC API provides a call-level API for SQL-based database access.
- JDBC technology allows you to use the Java programming language to exploit "Write Once, Run Anywhere" capabilities for applications that require access to enterprise data. With a JDBC technology-enabled driver, you can connect all corporate data even in a heterogeneous environment.

Oledb

- **OLE DB** (*Object Linking and Embedding, Database*, sometimes written as **OLEDB** or **OLE-DB**), an API designed by Microsoft, allows accessing data from a variety of sources in a uniform manner. The API provides a set of interfaces implemented using the Component Object Model (COM); it is otherwise unrelated to OLE. Microsoft originally intended OLE DB as a higher-level replacement for, and successor to, ODBC, extending its feature set to support a wider variety of non-relational databases, such as object databases and spreadsheets that do not necessarily implement SQL.
- OLE DB is Microsoft's strategic low-level application program interface for access to different data sources. OLE DB includes not only the Structured Query Language capabilities of the Microsoft-sponsored standard data interface Open Database Connectivity but also includes access to data other than SQL data.

Distributed computing concept in COM/ CORBA

What is COM?

- Microsoft COM (Component Object Model) technology in the Microsoft Windows-family of Operating Systems enables software components to communicate. COM is used by developers to create re-usable software components, link components together to build applications, and take advantage of Windows services. COM objects can be created with a variety of programming languages. Object-oriented languages, such as C++, provide programming mechanisms that simplify the implementation of COM objects. The family of COM technologies includes COM+, Distributed COM (DCOM) and ActiveX® Controls.
- Microsoft provides COM interfaces for many Windows application programming interfaces such as Direct Show, Media Foundation, Packaging API, Windows Animation Manager, Windows Portable Devices, and Microsoft Active Directory (AD).
- COM is used in applications such as the Microsoft Office Family of products.

Distributed computing concept in COM/ CORBA

What is CORBA?

- The Common Object Request Broker Architecture (CORBA) is a standard developed by the Object Management Group (OMG) to provide interoperability among distributed objects.
- CORBA is the world's leading middleware solution enabling the exchange of information, independent of hardware platforms, programming languages, and operating systems.
- CORBA is essentially a design specification for an Object Request Broker (ORB), where an ORB provides the mechanism required for distributed objects to communicate with one another, whether locally or on remote devices, written in different languages, or at different locations on a network.
- CORBA is often described as a "software bus" because it is a software-based communications interface through which objects are located and accessed.

Distributed computing concept in COM/ CORBA

What is CORBA?

- Using CORBA, application components can communicate with one another no matter where they are located, or who has designed them. CORBA provides the location transparency to be able to execute these applications.
- In order to build a system that uses or implements a CORBA-based distributed object interface, a developer must either obtain or write the IDL code that defines the object-oriented interface to the logic the system will use or implement.
- Typically, an ORB implementation includes a tool called an IDL compiler that translates the IDL interface into the target language for use in that part of the system.
- A traditional compiler then compiles the generated code to create the linkable-object files for use in the application.

Distributed computing concept in COM/ CORBA

What is CORBA?

- One of the benefits of using CORBA to implement a system such as our shopping cart application is that the objects can be accessed by services written in different languages.
- To accomplish this task, CORBA defines an interface to which all languages must conform.
- The CORBA interface is called the *Interface Definition Language (IDL).*
- *For* CORBA to work, both sides of the wire, the client and server, must adhere to the contract as stated in the IDL.
- The main premise of CORBA (Common Object Request Broker Architecture) is this: Using a standard protocol,CORBA allows programs from different vendors to communicate with each other.This interoperability covers hardware and software.Thus, vendors can write applications on various hardware platforms and operating systems using a wide variety of programming languages, operating over different vendor networks.