**St. joseph's college of engineering, Chennai**

**Information technology**

**DBMS**

**Unit-5 notes**

## Database Security

A DBMS typically includes a database security and authorization subsystem that is responsible for ensuring the security portions of a database against unauthorized access.

## Types of Security

- Legal and ethical issues
- Policy issues
- System-related issues
- The need to identify multiple security levels

- **Threats to databases**
  - Loss of **integrity**
  - Loss of **availability**
  - Loss of **confidentiality**
- To protect databases against these types of threats four kinds of countermeasures can be implemented:
  - Access control
  - Inference control
  - Flow control
  - Encryption

## Database Security and the DBA

The database administrator (**DBA**) is the central authority for managing a database system.

- The DBA's responsibilities include
  - granting privileges to users who need to use the system

- classifying users and data in accordance with the policy of the organization

- The DBA is responsible for the overall security of the database system.

**Data classification**

**-** The data classifications determine how the data will be secured, managed, retained, and disposed of

-The process that groups data that possess similar characteristics into categories.

-The value of each sample of data is determine and recorded according to organization standards

**Goals**

-Identify what information exists, and who needs it.

-Provide a system for protecting information critical to the organization.

**Benefits**

-Provide clear picture of the categories of data that exists in the organization

-Once data classification has been determined the appropriate control activities can be established.

**Database access control**

As the name Access control itself describes that this mechanism is all about user's access ro the database.

1. Discretionary access control

2. Mandatory Access control

3. Role based Access control

**Discretionary Access Control**

The typical method of enforcing discretionary access control in a database system is based on the granting and revoking privileges.

**Types of Discretionary Privileges**

- The **account level**:
    - At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database.
    - The **relation level** (or **table level**):
        - At this level, the DBA can control the privilege to access each individual relation or view in the database.
- The privileges at the **account level** apply to the capabilities provided to the account itself and can include
    - the **CREATE SCHEMA** or **CREATE TABLE** privilege, to create a schema or base relation;
    - the **CREATE VIEW** privilege;
    - the **ALTER** privilege, to apply schema changes such adding or removing attributes from relations;
    - the **DROP** privilege, to delete relations or views;
    - the **MODIFY** privilege, to insert, delete, or update tuples;
    - and the **SELECT** privilege, to retrieve information from the database by using a **SELECT** query.
- The second level of privileges applies to the **relation level**
    - This includes **base relations** and virtual (**view**) relations.
- The granting and revoking of privileges generally follow an authorization model for discretionary privileges known as the access matrix model where
    - The **rows** of a matrix M represents **subjects** (users, accounts, programs)
    - The **columns** represent **objects** (relations, records, columns, views, operations).
    - Each position **M(i,j)** in the matrix represents the types of privileges (read, write, update) that **subject i** holds on **object j**.

**Revoking Privileges**

- In some cases it is desirable to grant a privilege to a user temporarily. For example,

- The owner of a relation may want to grant the **SELECT** privilege to a user for a specific task and then revoke that privilege once the task is completed.

  Hence, a mechanism for **revoking** privileges is needed. In SQL, a **REVOKE** command is included for the purpose of **canceling privileges**

**Propagation of Privileges using the GRANT OPTION**

- Whenever the owner A of a relation R grants a privilege on R to another account B, privilege can be given to B with or without the GRANT OPTION.
- If the GRANT OPTION is given, this means that B can also grant that privilege on R to other accounts.
  - Suppose that B is given the GRANT OPTION by A and that B then grants the privilege on R to a third account C, also with GRANT OPTION. In this way, privileges on R can propagate to other accounts without the knowledge of the owner of R.
  - If the owner account A now revokes the privilege granted to B, all the privileges that B propagated based on that privilege should automatically be revoked by the system

**Example**

- Suppose that A1 wants to allow A3 to retrieve information from either of the two tables and also to be able to propagate the SELECT privilege to other accounts.
- A1 can issue the command:

  **GRANT SELECT ON** EMPLOYEE, DEPARTMENT
  **TO** A3 **WITH GRANT OPTION;**
- A3 can grant the **SELECT** privilege on the **EMPLOYEE** relation to A4 by issuing:

  **GRANT SELECT ON** EMPLOYEE **TO** A4;
- Suppose that A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3; A1 can issue:

  **REVOKE SELECT ON** EMPLOYEE **FROM** A3**;**

**Mandatory Access Control**

- In many applications, and additional security policy is needed that classifies data and users based on security classes.
  - This approach as mandatory access control, would typically be combined with the discretionary access control mechanisms.

- Typical **security classes** are top secret (TS), secret (S), confidential (C), and unclassified (U), where TS is the highest level and U the lowest: TS ≥ S ≥ C ≥ U
- The commonly used model for multilevel security, known as the Bell-LaPadula model, classifies each **subject** (user, account, program) and **object** (relation, tuple, column, view, operation) into one of the security classifications, T, S, C, or U:
    - **Clearance** (classification) of a subject S as **class(S**) and to the **classification** of an object O as **class(O)**.
- To incorporate multilevel security notions into the relational database model, it is common to consider attribute values and tuples as data objects.
- Hence, each attribute A is associated with a **classification attribute C** in the schema, and each attribute value in a tuple is associated with a corresponding security classification.
- In addition, in some models, a **tuple classification** attribute TC is added to the relation attributes to provide a classification for each tuple as a whole.
- Hence, a **multilevel relation** schema R with n attributes would be represented as
    - $R(A_1,C_1,A_2,C_2, \ldots, A_n,C_n,TC)$
- where each Ci represents the classification attribute associated with attribute $A_i$.
- The apparent key of a multilevel relation is the set of attributes that would have formed the primary key in a regular(single-level) relation.

**Role-Based Access Control**

- proven technology for managing and enforcing security in large-scale enterprisewide systems.
- Its basic notion is that permissions are associated with roles, and users are assigned to appropriate roles.
- Roles can be created using the CREATE ROLE and DESTROY ROLE commands.
    - The GRANT and REVOKE commands discussed under DAC can then be used to assign and revoke privileges from roles.
- **RBAC** appears to be a viable alternative to traditional discretionary and mandatory access controls; it ensures that only authorized users are given access to certain data or resources.
- Many DBMSs have allowed the concept of roles, where privileges can be assigned to roles.
- Role hierarchy in **RBAC** is a natural way of organizing roles to reflect the organization's lines of authority and responsibility.

**Statistical databases**

- **Statistical databases** are used mainly to produce statistics on various populations.
- The database may contain **confidential data** on individuals, which should be protected from user access.
- Users are permitted to retrieve **statistical information** on the populations, such as **averages, sums, counts, maximums, minimums,** and **standard deviations**.
- A **population** is a set of tuples of a relation (table) that satisfy some selection condition.
- Statistical queries involve applying **statistical functions** to a **population** of tuples.

**For example,**

- we may want to retrieve the *number* of individuals in a **population** or the *average income* in the population.
    - However, statistical users are not allowed to retrieve individual data, such as the income of a specific person.
- Statistical database security techniques must prohibit the retrieval of individual data.

**Flow Control**

- Flow control regulates the distribution or flow of information among accessible objects.
- A flow between object X and object Y occurs when a program reads values from X and writes values into Y.
    - Flow controls check that information contained in some objects does not flow explicitly or implicitly into less protected objects.
- A **covert channel** allows a transfer of information that violates the security or the policy.

**Cryptography**

The art of protecting information by transforming it (*encrypting* it) into an unreadable format, called cipher text. Only those who possess a secret *key* can decipher (or *decrypt*) the message into plain text.

**Cryptosystem Services**

- Confidentiality
- Integrity
- Authenticity

- Nonrepudiation
- Access Control

**Cryptographic Methods**

- ***Symmetric***
    - Same key for encryption and decryption

  **Example Algorithm**

  **-DES (data encryption standard)**

- ***Asymmetric***
    - Mathematically related key pairs for encryption and decryption
    - Public and private keys

  **Example Algorithms**

    - Diffie-Hellman
    - RSA

**Encryption and Public Key Infrastructures**

- Encryption is a means of maintaining secure data in an insecure environment.
- Encryption consists of applying an encryption algorithm to data using some prespecified encryption key.
    - The resulting data has to be decrypted using a decryption key to recover the original data.

**Public Key Encryption**

- Public key algorithms are based on mathematical functions rather than operations on bit patterns.
- They also involve the use of two separate keys
- The two keys used for public key encryption are referred to as the public key and the private key.
- Public key is made for public and private key is known only by owner

**A public key encryption scheme, or infrastructure, has six ingredients:**

- **Plaintext:** This is the data or readable message that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private keys:** These are pair of keys that have been selected so that if one is used for encryption, the other is used for decryption.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key.
- **Decryption algorithm**: This algorithm accepts the ciphertext and the matching key and produces the original plaintext

**Digital Signatures**

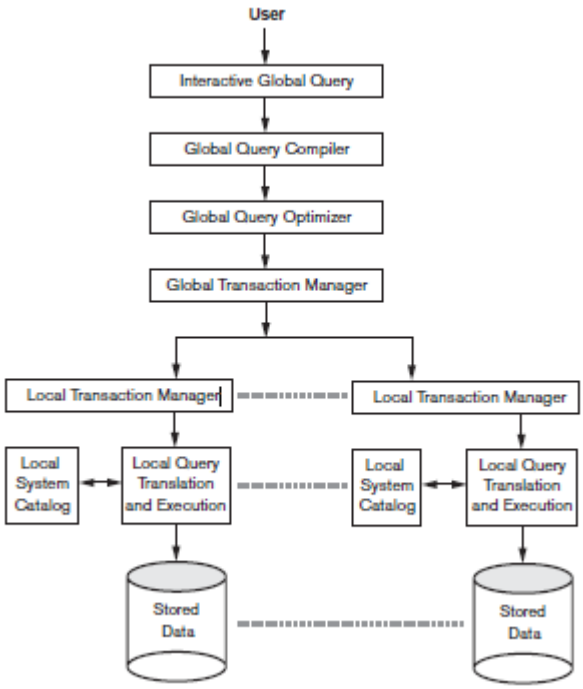
A digital signature is an example of using encryption techniques to provide authentication services in e-commerce applications.

- A digital signature is a means of associating a mark unique to an individual with a body of text.
    - The mark should be unforgettable, meaning that others should be able to check that the signature does come from the originator.
- A digital signature consists of a string of symbols.
    - Signature must be different for each use.
        - This can be achieved by making each digital signature a function of the message that it is signing, together with a time stam

**Distributed database**

**Distributed database (DDB)** as a collection of multiple logically interrelated databases distributed over a computer network, and a **distributed database management system (DDBMS)** as a software system that manages a distributed database while making the distribution transparent to the user

**Component architecture**




**Association Rule**

Finding frequent patterns, associations, correlations,or casual structures among sets of items or objects in transaction databases, relational databases, and other information repositories

**Generating association rule**

- Market-Basket Model, Support, and Confidence
- Apriori Algorithm
- Sampling Algorithm
- Frequent-Pattern Tree Algorithm
- Partition Algorithm

- Retail shops are often interested in associations between different items that people buy.
    - Someone who buys bread is quite likely also to buy milk
    - A person who bought the book *Database System Concepts* is quite likely also to buy the book *Operating System Concepts*.
- Association s information can be used in several ways.

- E.g. when a customer buys a particular book, an online shop may suggest associated books.

*bread* ⇒ *milk*  *DB-Concepts, OS-Concepts* ⇒ Networks

- Left hand side: antecedent, right hand side: consequent
- An association rule must have an associated population; the population consists of a set of instances
    - E.g. each transaction (sale) at a shop is an instance, and the set of all transactions is the population

- Rules have an associated support, as well as an associated confidence.

**Support** is a measure of what fraction of the population satisfies both the antecedent and the consequent of the rule**.**

- E.g. suppose only 0.001 percent of all purchases include milk and screwdrivers. The support for the rule is *milk* ⇒ *screwdrivers* is low.
- We usually want rules with a reasonably high support
    - Rules with low support are usually not very useful
    - $support = \frac{(X \cup Y).count}{n}$

**Confidence** is a measure of how often the consequent is true when the antecedent is true.

- E.g. the rule *bread* ⇒ *milk* has a confidence of 80 percent if 80 percent of the purchases that include bread also include milk.
- Usually want rules with reasonably large confidence.
    - A rule with a low confidence is not meaningful.

$$confidence = \frac{(X \cup Y).count}{X.count}$$

**Goal:**

Find all rules that satisfy the user-specified *minimum support* (minsup) and *minimum confidence* (minconf).

**Generating Association Rules:**
**Apriori Algorithm**

**Apriori principle:**

-If an itemset is frequent, then all of its subsets must also be frequent

-Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

Support of an itemset never exceeds the support of its subsets

This is known as the **anti-monotone** property of support

**Illustrating Apriori Principle**

Items (1-itemsets)

| Item | Count |
|---|---|
| Bread | 4 |
| Coke | 2 |
| Milk | 4 |
| Beer | 3 |
| Diaper | 4 |
| Eggs | 1 |

Pairs (2-itemsets)

No need to generate candidates involving Coe or Eggs

| Itemset | Count |
|---|---|
| {Bread,Milk} | 3 |
| {Bread,Beer} | 2 |
| {Bread,Diaper} | 3 |
| {Milk,Beer} | 2 |
| {Milk,Diaper} | 3 |
| {Beer,Diaper} | 3 |

Triplets (3-itemsets)No need to generate candidates involving {bread, beer} and {milk, beer}

| Itemset | Count |
|---|---|
| {Bread,Milk,Diaper} | 3 |

**Apriori Algorithm**

- N= Nomber of attributes
- Attribute_list = all attributes
- For i=1 to N
    - Frequent_item_set = generate item set of i attributes using attribute_list
    - Frequent_item_set = remove all infrequent itemsets

        - Rule = Rule U generate rule using Frequent_item_set
        - Attribute_list = Only attributes contained in Frequent_item_set
- End For

**Apriori candidate generation**

- The candidate-gen function takes $F_{k-1}$ and returns a superset (called the candidates) of the set of all frequent *k*-itemsets. It has two steps
    - ***join* step:** Generate all possible candidate itemsets $C_k$ of length *k*
    - ***prune* step:** Remove those candidates in $C_k$ that cannot be frequent.

**The Sampling Algorithm**

- The **sampling algorithm** selects samples from the database of transactions that individually fit into memory. Frequent itemsets are then formed for each sample.
    - If the frequent itemsets form a superset of the frequent itemsets for the entire database, then the real frequent itemsets can be obtained by scanning the remainder of the database.
    - In some rare cases, a second scan of the database is required to find all frequent itemsets.

**Frequent-Pattern Tree Algorithm**

- The **Frequent-Pattern Tree** Algorithm reduces the total number of candidate itemsets by producing a compressed version of the database in terms of an FP-tree.
- The FP-tree stores relevant information and allows for the efficient discovery of frequent itemsets.
- The algorithm consists of two steps:
    - Step 1 builds the FP-tree.
    - Step 2 uses the tree to find frequent itemsets.

**The Partition Algorithm**

- Divide the database into non-overlapping subsets.
- Treat each subset as a separate database where each subset fits entirely into main memory.
- Apply the Apriori algorithm to each partition.

- Take the union of all frequent itemsets from each partition.
- These itemsets form the global candidate frequent itemsets for the entire database.

**Complications seen with Association Rules**

- The cardinality of itemsets in most situations is extremely large.
- Association rule mining is more difficult when transactions show variability in factors such as geographic location and seasons.
- Data quality is variable; data may be missing, erroneous, conflicting, as well as redundant.

**Classification**

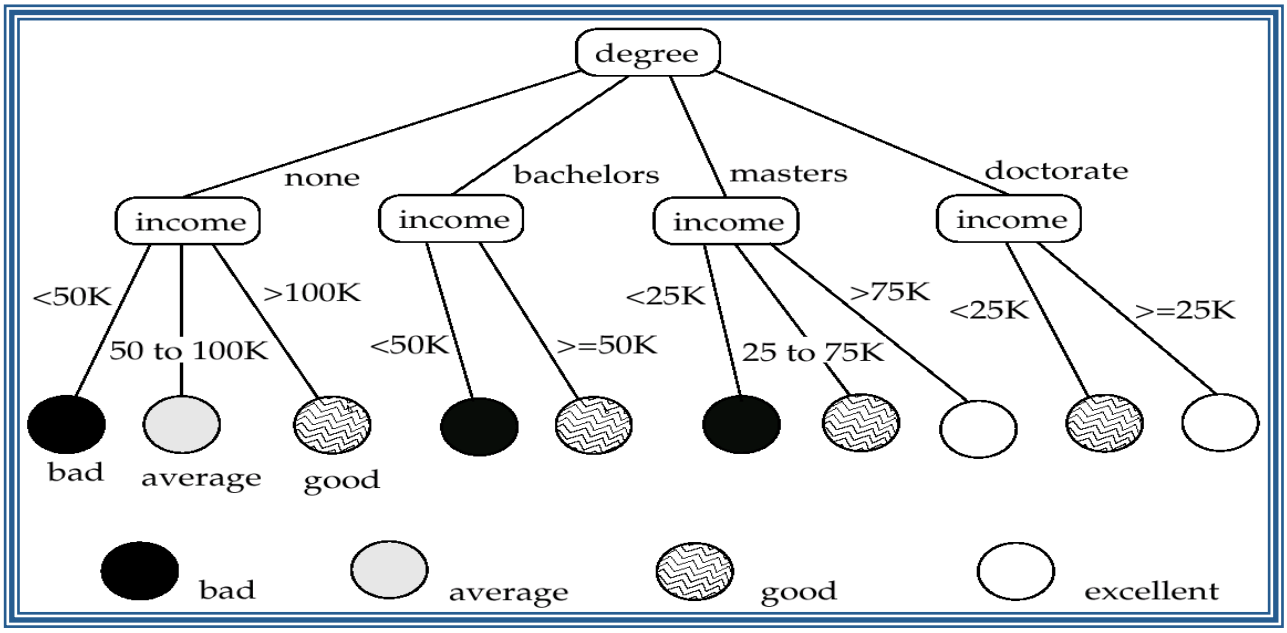
- Classification is the process of learning a model that is able to describe different classes of data.
- Learning is supervised as the classes to be learned are predetermined.
- Learning is accomplished by using a training set of pre-classified data.
- The model produced is usually in the form of a decision tree or a set of rules.

**Classification Rules**

- Classification rules help assign new objects to a set of classes. E.g., given a new automobile insurance applicant, should he or she be classified as low risk, medium risk or high risk?
- Classification rules for above example could use a variety of knowledge, such as educational level of applicant, salary of applicant, age of applicant, etc.
    - $\forall$ person P, P.degree = masters **and** P.income > 75,000

      $\Rightarrow$ P.credit = excellent
    - $\forall$ person P, P.degree = bachelors **and**
      (P.income $\geq$ 25,000 and P.income $\leq$ 75,000)

      $\Rightarrow$ P.credit = good
- Rules are not necessarily exact: there may be some misclassifications
- Classification rules can be compactly shown as a **decision tree.**

- 


**Clustering**

- Unsupervised learning or clustering builds models from data without predefined classes.
- The goal is to place records into groups where the records in a group are highly similar to each other and dissimilar to records in other groups.
- The k-Means algorithm is a simple yet effective clustering technique

**Agglomerative clustering algorithms**

- Build small clusters, then cluster small clusters into bigger clusters, and so on

**Divisive clustering algorithms**

- Start with all items in a single cluster, repeatedly refine (break) clusters into smaller ones

**Information Retrieval Systems**

- **Information retrieval** (**IR**) systems use a simpler data model than database systems
  - Information organized as a collection of documents
  - Documents are unstructured, no schema
- Information retrieval locates relevant documents, on the basis of user input such as keywords or example documents
  - e.g., find documents containing the words “database systems”

- Can be used even on textual descriptions provided with non-textual data such as images
- IR on Web documents has become extremely important
    - E.g. google, altavista, …

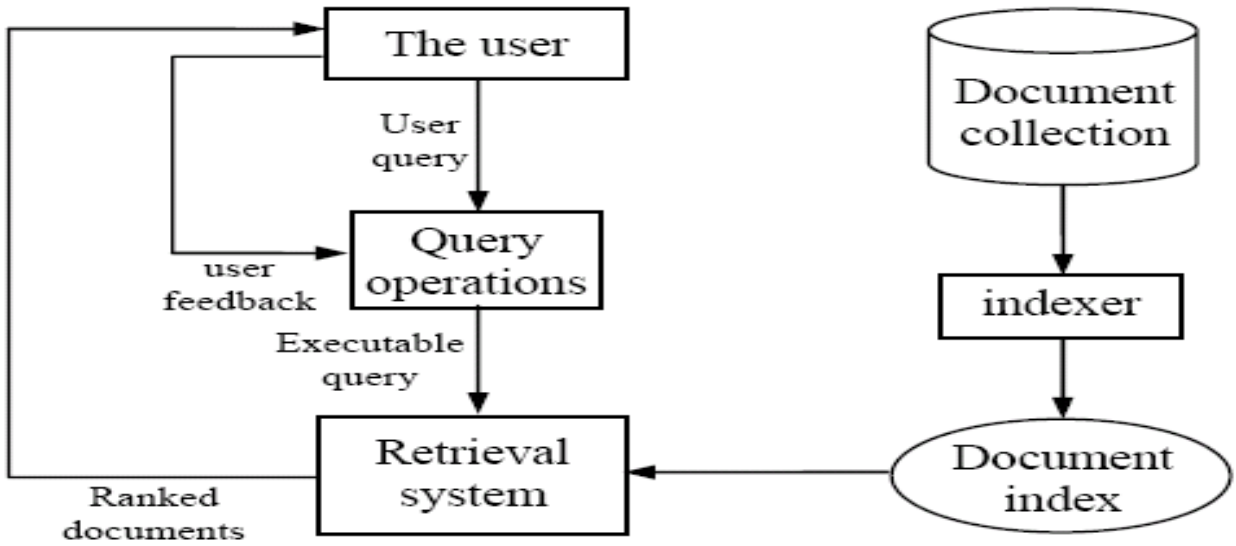
**Differences from database systems**

- IR systems don't deal with transactional updates (including concurrency control and recovery)
- Database systems deal with structured data, with schemas that define the data organization
- IR systems deal with some querying issues not generally addressed by database systems
    - Approximate searching by keywords
    - Ranking of retrieved answers by estimated degree of relevance

**Keyword Search**

- In **full text** retrieval, all the words in each document are considered to be keywords.
    - We use the word **term** to refer to the words in a document
- Information-retrieval systems typically allow query expressions formed using keywords and the logical connectives ***and, or,*** and ***not***
    - *And*s are implicit, even if not explicitly specified

**IR architecture**



**IR queries**

- Keyword queries
- Boolean queries (using AND, OR, NOT)
- Phrase queries
- Proximity queries
- Full document queries
- Natural language questions

**Information retrieval models**

- An IR model governs how a document and a query are represented and how the relevance of a document to a user query is defined.
- **Main models:**
    - Boolean model
    - Vector space model
    - Statistical language model

**Relevance ranking**

- Ranking of documents on the basis of estimated relevance to a query is critical
    - Relevance ranking is based on factors such as
        - **Term frequency**
            - Frequency of occurrence of query keyword in document
        - **Inverse document frequency**
            - How many documents the query keyword occurs in
                - Fewer ➔ give more importance to keyword
        - **Hyperlinks to documents**
            - More links to a document ➔ document is more important

**Relevance Ranking Using Terms**

- TF-IDF (Term frequency/Inverse Document frequency) ranking:
    - Let $n(d)$ = number of terms in the document $d$
    - $n(d, t)$ = number of occurrences of term $t$ in the document $d$.

Then relevance of a document $d$ to a term $t$

$$r(d, t) = log \left(1 + \frac{n(d, t)}{n(d)}\right)$$

- The log factor is to avoid excessive weightage to frequent terms

And relevance of document to query $Q$

$$r(d, Q) = \sum_{t \in Q} \frac{r(d, t)}{n(t)}$$

- Most systems add to the above model
    - Words that occur in title, author list, section headings, etc. are given greater importance
    - Words whose first occurrence is late in the document are given lower importance
    - Very common words such as “a”, “an”, “the”, “it” etc are eliminated
        - Called stop words
    - Proximity: if keywords in query occur close together in the document, the document has higher importance than if they occur far apart
- Documents are returned in decreasing order of relevance score
    - Usually only top few documents are returned, not all

**Relevance Using Hyperlinks**

- When using keyword queries on the Web, the number of documents is enormous (many billions)
    - Number of documents relevant to a query can be enormous if only term frequencies are taken into account
- Using term frequencies makes “spamming” easy

        - E.g. a travel agent can add many occurrences of the words “travel agent” to his page to make its rank very high
- Most of the time people are looking for pages from popular sites
- Idea: use **popularity** of Web site (e.g. how many people visit it) to rank site pages that match given keywords

**Web Crawling and indexing**

- Web crawlers are programs that locate and gather information on the Web
    - Recursively follow hyperlinks present in known documents, to find other documents
        - Starting from a *seed* set of documents
    - Fetched documents
        - Handed over to an indexing system
        - Can be discarded after indexing, or store as a *cached* copy
- Crawling the entire Web would take a very large amount of time
    - Search engines typically cover only a part of the Web, not all of it
    - Take months to perform a single crawl
    - most well-known crawler is called “Googlebot.”
- Crawling is done by multiple processes on multiple machines, running in parallel
    - Set of links to be crawled stored in a database
    - New links found in crawled pages added to this set, to be crawled later
- Indexing process also runs on multiple machines
    - Creates a new copy of index instead of modifying old index
    - Old index is used to answer queries
    - After a crawl is “completed” new index becomes “old” index
- Multiple machines used to answer queries
    - Indices may be kept in memory

- Queries may be routed to different machines for load balancing

**Types**

- Basic Crawling
- Selective crawling
- Focused crawling
- Distributed crawling

**Object oriented database**

**OODB = Object Orientation + Database Capabilities**

Object-Oriented Database Features:
-persistence
- support of transactions
- simple querying of bulk data
- concurrent access
- resilience
- security

Integration and Sharing

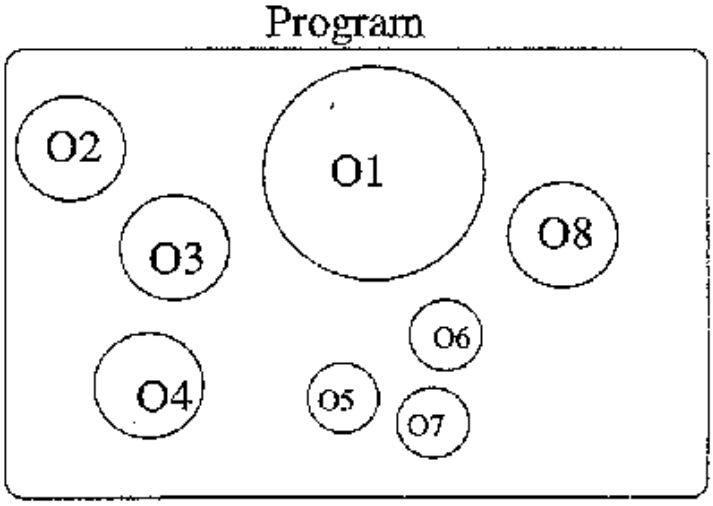
-Seamless integration of operating systems, databases, languages, spreadsheets, word processors, AI expert system shells.

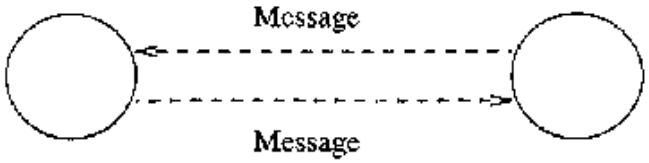
-Sharing of data, information, software components, products, computing environments.

-Referential sharing:
Multiple applications, products, or objects share common sub-objects.

Object-oriented databases allows referential sharing through the support of object identity and inheritance.

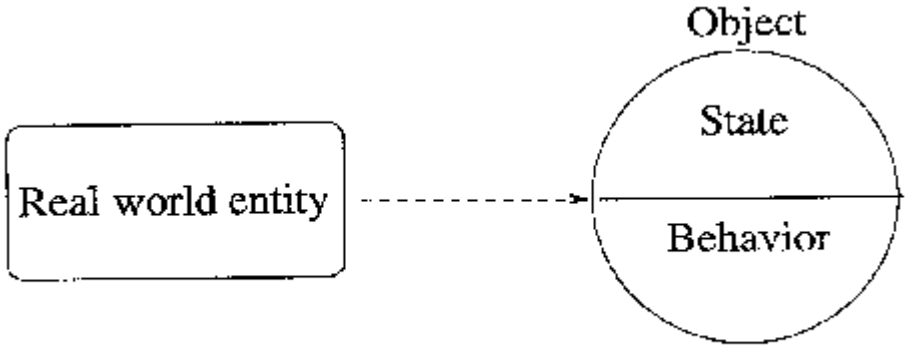
The object-oriented paradigm is illustrated below:



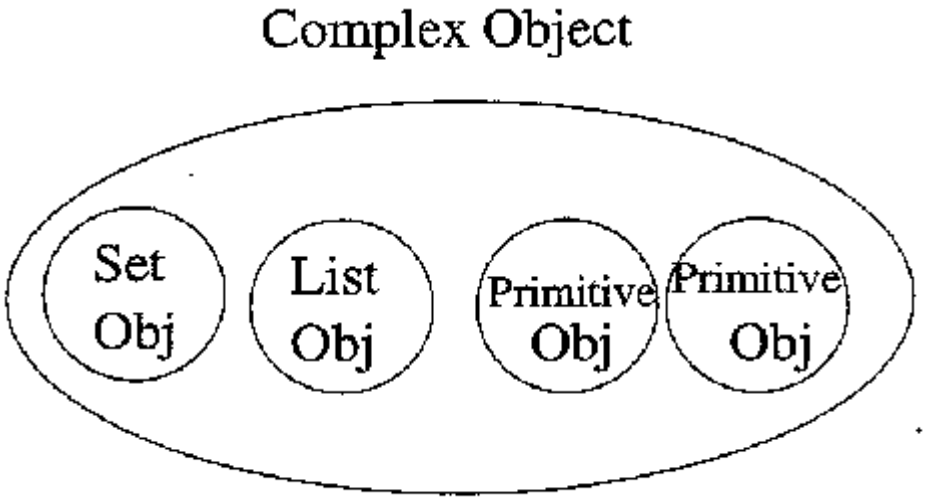

Objects and Identity

The following figure shows object with state and behavior. The state is represented by the values of the object's attributes, and the behavior is defined by the methods acting on the state of the object. There is a unique object identifier OID to identify the object.



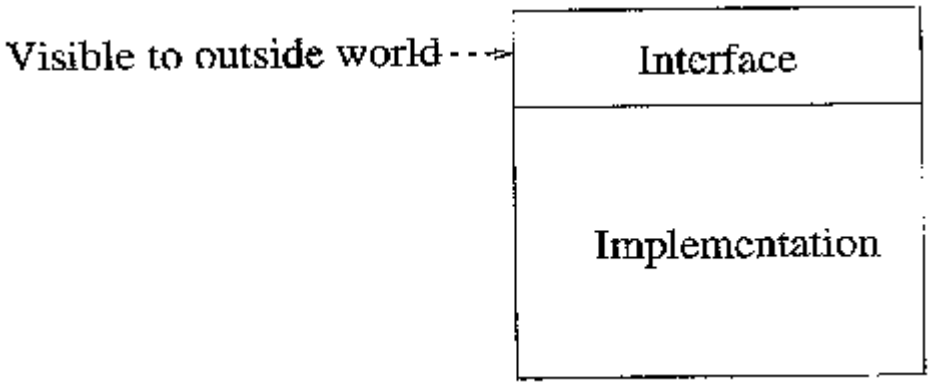
Complex Objects

Complex objects are built by applying constructors to simpler objects including: sets, lists and tuples. An example is illustrated below:




Encapsulation

Encapsulation is derived from the notion of Abstract Data Type (ADT). It is motivated by the need to make a clear distinction between the specification and the implementation of an operation. It reinforces modularity and provides a form of logical data independence.




**Class**

A class object is an object which acts as a template.

It specifies:

A structure that is the set of attributes of the instances

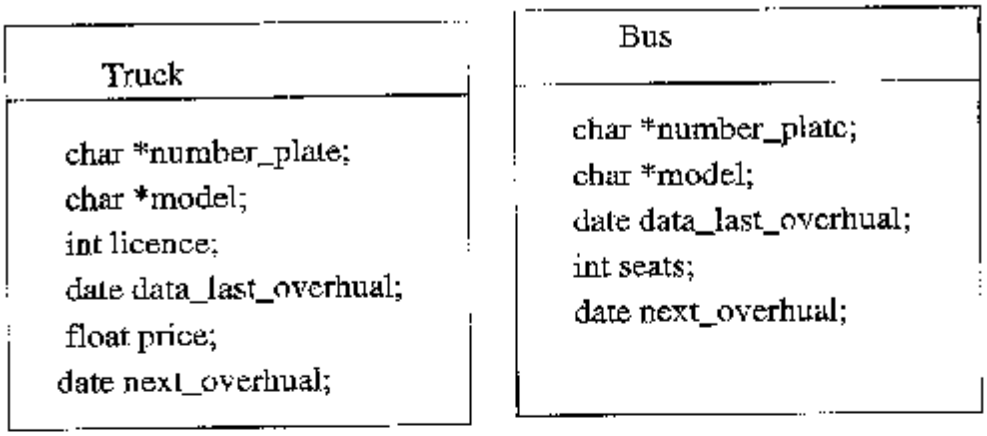
A set of operations
A set of methods which implement the operations

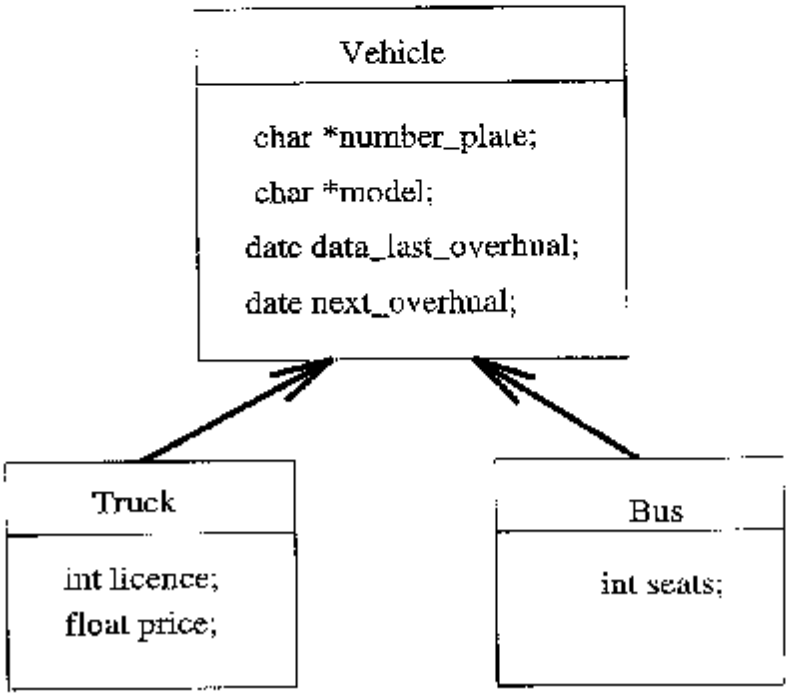
Instantiation means generating objects, Ex. 'new' operation in C++

Persistence of objects: Two approaches
An implicit characteristic of all objects
An orthogonal characteristic - insert the object into a persistent collection of objects

**Inheritance**

A mechanism of reusability, the most powerful concept of OO programming



**XML**

- XML stands for E**X**tensible **M**arkup **L**anguage
- XML was designed to describe data.
- XML tags are not predefined unlike HTML
- XML DTD and XML Schema define rules to describe data
- XML example of semi structured data

**Building Blocks of XML**

Elements (Tags) are the primary components of XML documents.

```
<AUTHOR id = 123>
        <FNAME> JAMES</FNAME>
      <LNAME>  RUSSEL</LNAME>
  </AUTHOR>                                          <!- I am comment ->
```

- Attributes provide additional information about Elements. Values of the Attributes are set inside the Elements
- Comments stats with <!- and end with ->

**XML DTD**

- A DTD is a set of rules that allow us to specify our own set of elements and attributes.
- DTD is grammar to indicate what tags are legal in XML documents.
- XML Document is valid if it has an attached DTD and document is structured according to rules defined in DTD.

**DTD Example**

**Xml Document And Corresponding DTD**

**Xml Document**

```
<BOOKLIST>
```

<BOOK GENRE = “Science” FORMAT = “Hardcover”>

<AUTHOR>

<FIRSTNAME> RICHRD </FIRSTNAME>

<LASTNAME> KARTER </LASTNAME>

</AUTHOR>

</BOOK>

</BOOKLIST>

**Corresponding DTD**

<!DOCTYPE BOOKLIST[

<!ELEMENT BOOKLIST(BOOK)*> <!ELEMENT BOOK(AUTHOR)>

<!ELEMENT AUTHOR(FIRSTNAME,LASTNAME)>

<!ELEMENT FIRSTNAME(#PCDATA)>

<!ELEMENT>LASTNAME(#PCDATA)>

<!ATTLIST BOOK GENRE (Science|Fiction)#REQUIRED>

<!ATTLIST BOOK FORMAT (Paperback|Hardcover) “PaperBack”>]>

**XML Schema**

**Serves same purpose as database schema**

- Schemas are written in XML
- Set of pre-defined simple types (such as string, integer)
- Allows creation of user-defined complex types

**RDBMS Schema (s_id integer, s_name string, s_status string)**

**XMLSchema**

***XML Document and Schema***

<Students>

<Student id=“p1”>

<Name>Allan</Name>

<Age>62</Age>

<Email>allan@abc.com

</Email>

</Student>

</Students>

***Schema***

<xs:schema>

<xs:complexType name = “StudnetType”>

<xs:attribute name=“id” type=“xs:string” />

<xs:element name=“Name” type=“xs:string />

<xs:element name=“Age” type=“xs:integer” />

<xs:element name=“Email” type=“xs:string” />

</xs:complexType>

<xs:element name=“Student” type=“StudentType” />

</xs:schema>

**XQuery**

- XQuery
- XQuery to XML is same as SQL to

RDBMS

- Most databases supports XQuery
- XQuery is built on XPath operators

(XPath is a language that defines path expressions to locate document data)

**XPath Example**

<Student id=“s1”>

<Name>John</Name>

<Age>22</Age>

<Email>jhn@xyz.com</Email>

</Student>

XPath: /Student[Name=“John”]/Email

Extracts: <Email> element with value “jhn@xyz.com