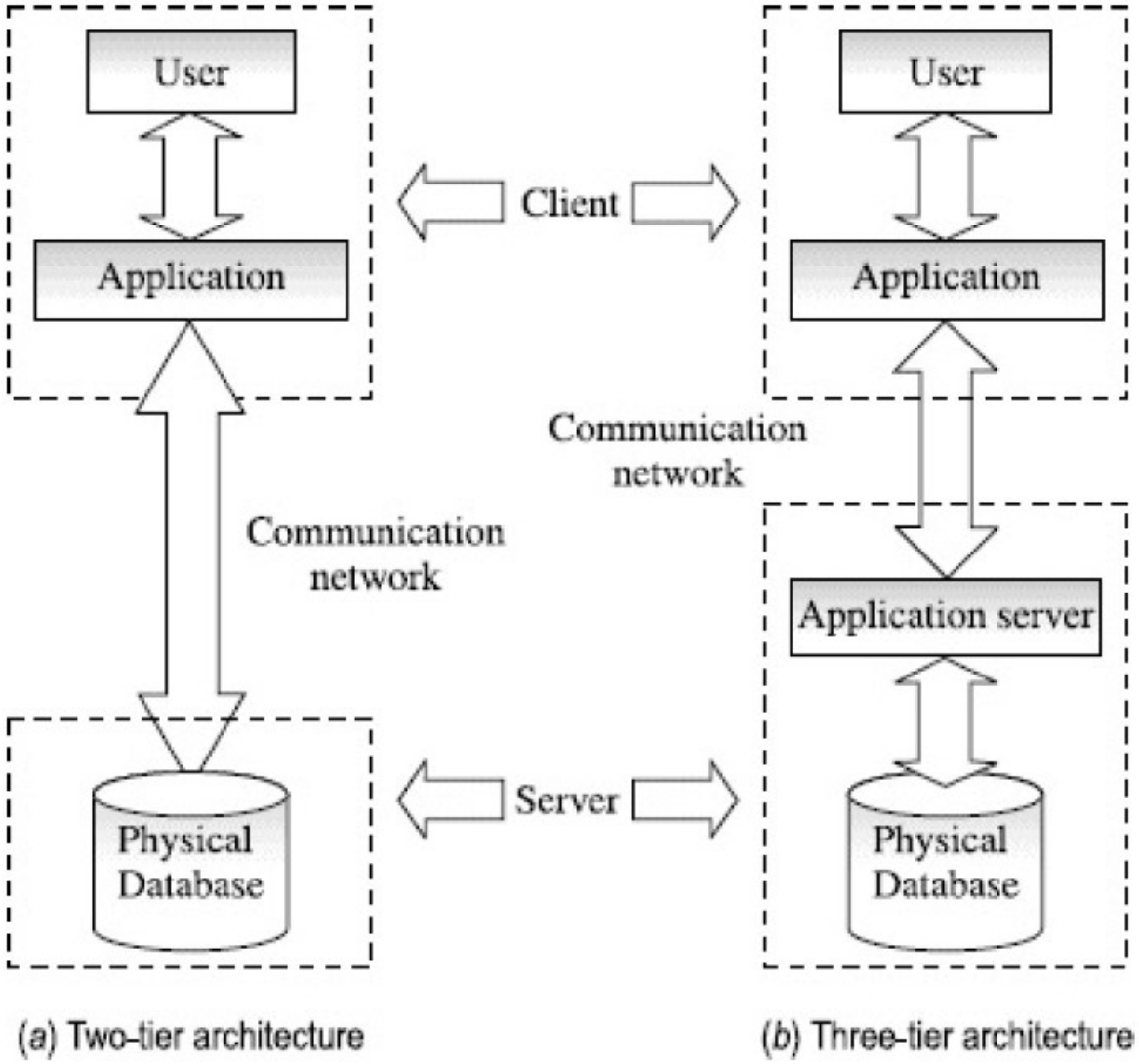
Database System Architecture

DataBase System Architecture



- organization requires accurate&reliable data -> maintains records for its varied operations by building appropriate database models (capturing essential properties of objects and relationship)
- users look for abstract view of data they are interested in
- since database is shared resource, each user may require different view of data
- we need to develop architecture for database systems - framework in which structure is described

- when database is designed to meet information needs of organization *scheme* of database becomes most important concern
- data in changes frequently, while schema remain the same over long time
- Schema consist of types of entities and relationships among these
- DBMS should do the translation between logical (users' view) organization and physical organization of data in database

Schema

- gives the names of entities and attributes
- specifies relationship among them
- database can have several schemas partitioned according to levels of abstraction

Logical / Physical Schema

- logical schema - concerned with exploiting data structures, to make scheme understandable to computer; important as programs use it to construct applications
  - database definition language (DDL)
- physical schema – deals with manner in which conceptual database shall get represented in computer as stored database; hidden beneath logical schema and can usually be changed easily without affecting application programs
  - database storage definition language (DSDl)

Subschema

- subset of schema and inherits same property
- refers to application programmer's (user's) view of data
- defines portion of database as "seen" by application programs

- different application programs can have different view of data; individual application programs can change their respective subschema without effecting subschema views of others
- application programs are not concerned about physical organization of data; physical organization of data in database can change without affecting application programs
- subschemas also act as unit for enforcing controlled access to database

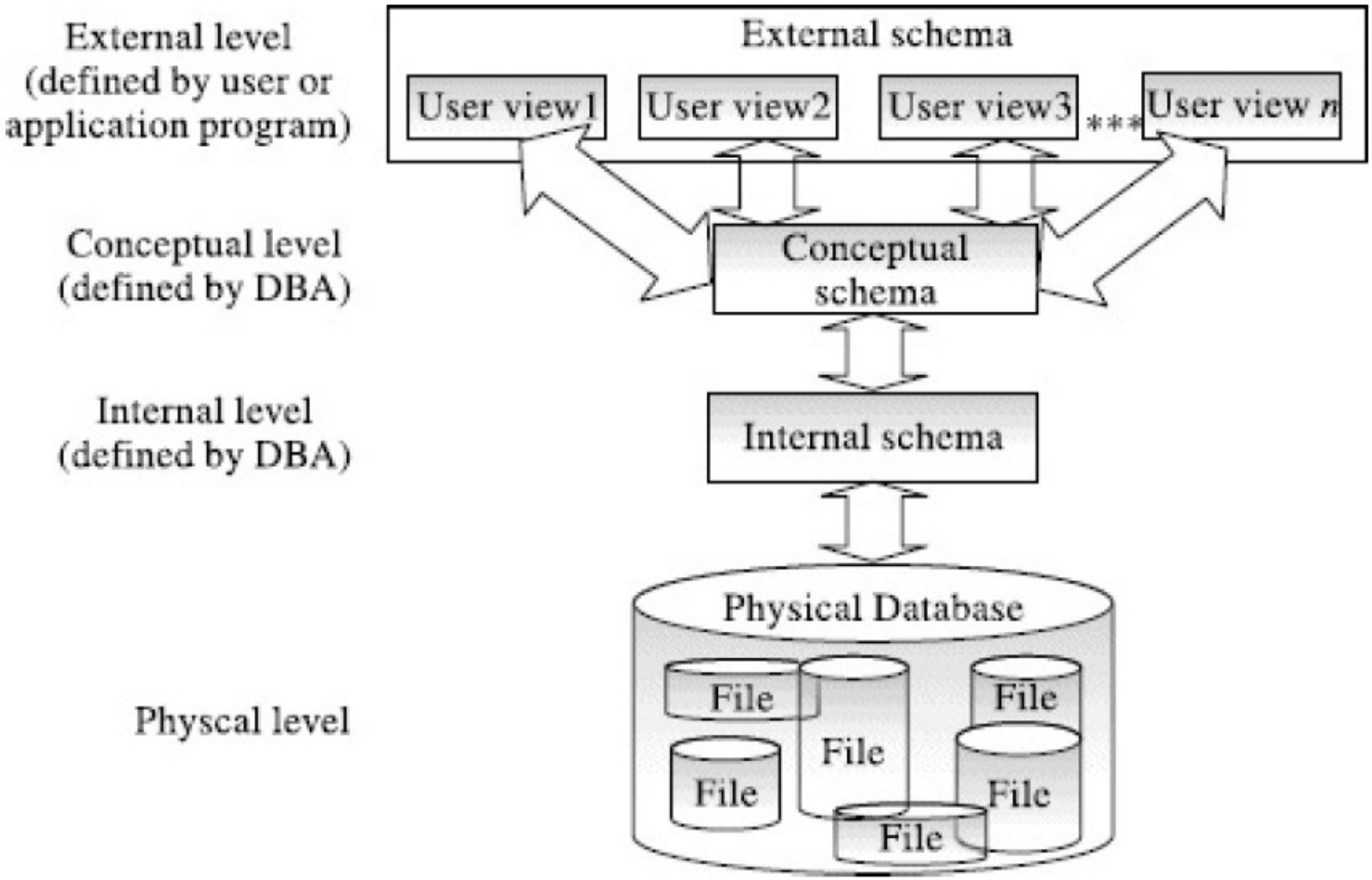
Instances

- when schema framework is filled in the data item values (contents of database at any point) referred to as an *instance* of database
- also called state of database or snapshot

Three-Level ANSI-SPARC DataBase Architecture

- in 1971, Database Task Group (DBTG) appointed by Conference on Data Systems and Languages (CODASYL), produced a proposal for general architecture for database systems
- two-tier architecture with system view called schema and user views called subschemas
- in 1975, ANSI-SPARC (American National Standards Institute — Standards Planning and Requirements Committee) produced three-tier architecture with system catalog
- architecture of most commercial DBMSs available today is based to some extent on this proposal

Three-Level ANSI-SPARC DataBase Architecture



Three-Level ANSI-SPARC DataBase Architecture consists of

1. Internal level
2. Conceptual level
3. External level

- data abstracted in three levels corresponding to three views (namely internal, conceptual and external views)
- lowest level of abstraction of data contains description of actual method of storing data
- third level is highest level of abstraction seen by user or application program

Internal Level

- physical representation of database on computer
- indicates how data will be stored
- describes data and file structures (definition of stored records, method of representing data fields), indexing and hashing schemes and access methods to be used on storage devices
- covered by internal level to achieve routine performance and storage space utilization

Internal Level concerned with

- storage space allocation for data
- record descriptions for storage with stored sizes for data items
- record placement
- data compression and data encryption techniques

Physical DataBase Design

- process arriving at good internal (or physical) schema

Conceptual Level

- all database entities and relationships
- provides community view of database, as seen by DBA
- *conceptual schema* defines conceptual view also called *logical schema*
- contains method of deriving objects in conceptual view from objects in internal view
- supports each external view, in that any data available to user must be derived from conceptual level
- level must not contain any storage-dependent details

Conceptual Level concerned with

- all entities, their attributes and their relationships
- constraint on data
- semantic information about data
- checks to retain data consistency and integrity
- security information

Conceptual DataBase Design

- process of arriving at good conceptual schema
- choice of relations (entities) and field (or data item) for each relation, is not always obvious

External Level

- is user's view of database, portions of concern to user or application program are included in form that is familiar for that user
- any number of user views, even identical, may exist for given conceptual view of database
- includes only those entities, attributes and relationships that user is interested in
- different views may have different representations of same data

External Level

- *external schema* describes each external view
- consists of definition of logical records and relationships in external view
- also contains method of deriving objects
- allow data access to be customized at level of individual users or groups of users
- given database has exactly one internal or physical schema and one conceptual schema but, it may have several external schemas, each tailored to particular group of users

Advantages of Three-tier Architecture

- main objective is to isolate each user's view of database from the way database is physically stored or logically represented
- each user is able to access same data but have different customized view as per their own needs; can change way views data and this change does not affect other users
- user is not concerned about physical data storage details - user's interaction with database is independent of physical data storage

Advantages of Three-tier Architecture

- internal structure of database is unaffected by changes to physical storage organization, such as changeover to new storage device
- database administrator (DBA) is able to change conceptual structure and database storage structures without affecting user's view

Data Independence

- major objective of implementing DBMS
- defined as immunity of application programs to change in logical and physical representation and access techniques
- characteristics of database system to change schema at one level without having to change schema at next higher level
- application programs do not depend on any one particular physical representation or access technique
- insulates application programs from changes in way data is structured and stored

Data Independence

- Physical data independence
- Logical data independence

Physical Data Independence

- immunity of conceptual (or external) schemas to changes in internal schema
- conceptual schema insulates users from changes in physical storage of data
  - such as using different file organizations or storage structures, using different storage devices, modifying indexes or hashing algorithms
- physical storage could be changed without necessitating change in conceptual view or any of external views; change is absorbed by conceptual/internal mapping

Logical Data Independence

- immunity of external schemas (application programs) to changes in conceptual schema
- users shielded from changes in logical structure of data
  - such as addition and deletion of entities, addition and deletion of attributes, or addition and deletion of relationships, without changing existing external schemas (having to rewrite application programs)
- only view definition and mapping need be changed; application programs that refers to external schema must work as before, after conceptual schema undergoes logical reorganization

mappings

- process of transforming requests and results between the three levels
- between internal, conceptual and external schemas

- Conceptual/Internal mapping
- External/Conceptual mapping

Conceptual/Internal Mapping

- defines correspondence between conceptual view and stored database
- specifies how conceptual records and fields are presented at internal level
- enables DBMS to find actual records in physical storage that constitute logical record in conceptual schema, together with any constraints to be enforced on operations)

Conceptual/Internal Mapping

- allows any differences in entity names, attribute names, attribute orders, data types, … to be resolved
- in case of any change in structure of stored database, conceptual/internal mapping is also changed, so that conceptual schema can remain invariant
- effects of changes to database storage structure are isolated below conceptual level in order to preserve physical data independence

External/Conceptual Mapping

- defines correspondence between external view and conceptual view
  - among records and relationships of external and conceptual views
- enables to map names in user's view to relevant part of conceptual schema
- there could be several mappings between external and conceptual levels
- conceptual/internal mapping is the key to physical data independence
- external/conceptual mapping is the key to the logical data independence

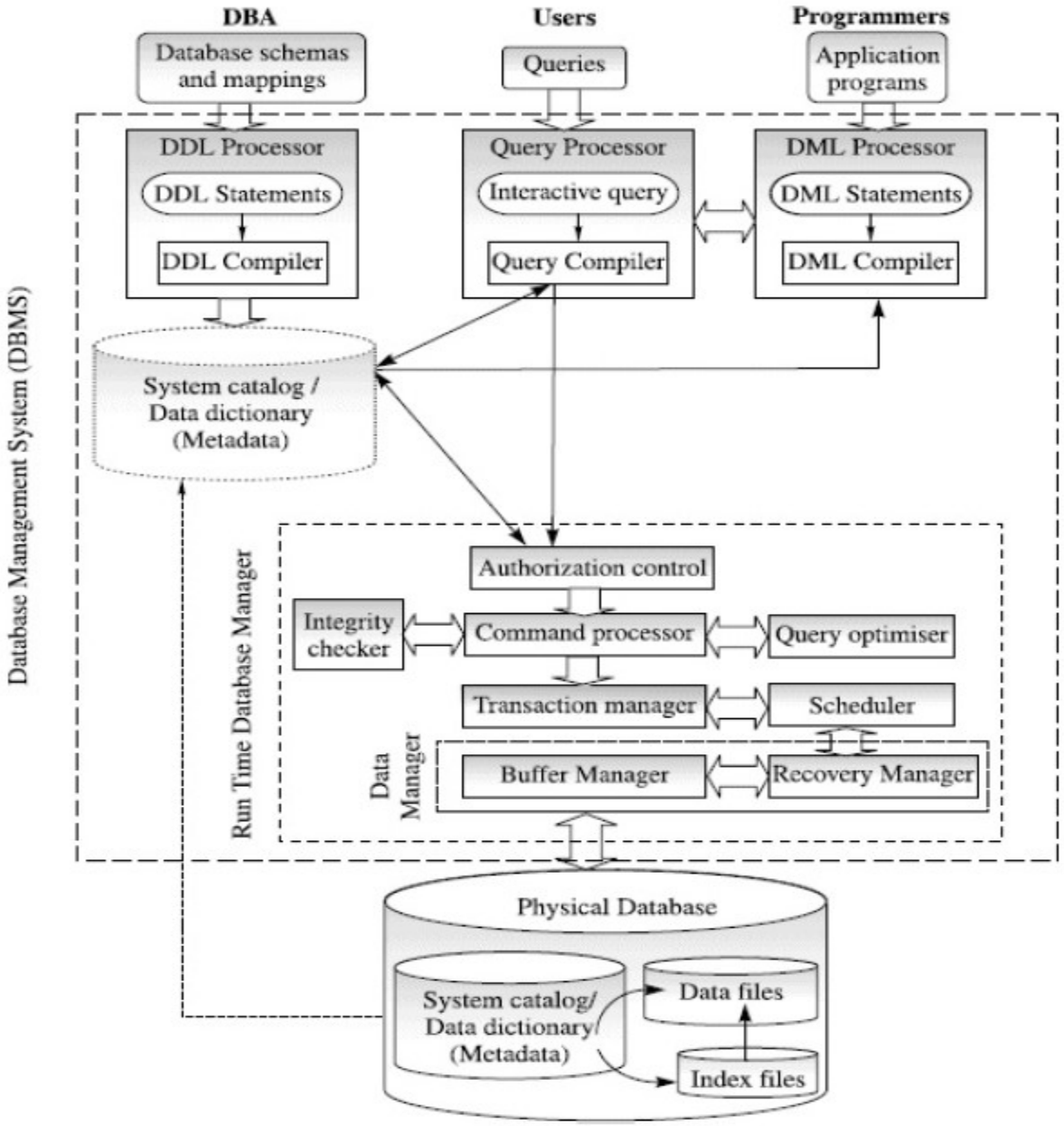
External/Conceptual Mapping

- information about mapping are included in system catalog of DBMS
- when schema is changed at some level, schema at next higher level remains unchanged; only mapping between levels is changed
- data independence is accomplished

Structure, Components, Functions of DBMS

Structure of a DBMS

- typical structure of DBMS with components and relationships between them
- DBMS partitioned into several modules
- some of functions of DBMS are supported by operating systems (OS) to provide basic services
  - physical data and system catalog are stored on physical disk, access to disk is controlled primarily by OS, therefore, interface with OS must be taken into account


DBA
Users
Programmers
Database schemas and mappings
Queries
Application programs
DDL Processor
DDL Statements
DDL Compiler
Query Processor
Interactive query
Query Compiler
DML Processor
DML Statements
DML Compiler
System catalog / Data dictionary (Metadata)
Database Management System (DBMS)
Run Time Database Manager
Authorization control
Integrity checker
Command processor
Query optimiser
Transaction manager
Scheduler
Data Manager
Buffer Manager
Recovery Manager
Physical Database
System catalog/ Data dictionary (Metadata)
Data files
Index files

Execution Steps of DBMS

1. users issue query using particular database language, for example, SQL commands
2. passed query is presented to query optimizer, which uses information about how data is stored to produce efficient execution plan for evaluating query
3. DBMS accepts users SQL commands and analyses them
4. DBMS produces query evaluation plans
   - external schema for user, corresponding external/conceptual mapping, conceptual schema, conceptual/internal mapping, storage structure definition
5. DBMS executes plans against physical database and returns answers to users

- DBMS supports concurrency and crash recovery by carefully scheduling users requests and maintaining log of all changes to database with components such as transaction manager, buffer manager, and recovery manager

Components of DBMS

- Query processor
- Run time database manager
  - Authorization control
  - Command processor
  - Integrity checker
  - Query optimizer
  - Transaction manager
  - Scheduler
  - Data manager
- DML processor
- DDL processor

Query processor

- transforms users queries into series of low-level instructions directed to run time database manager
- interpret online user's query and convert it into efficient series of operations in form capable of being sent to run time data manager for execution
- uses data dictionary to find structure of relevant portion of database and uses this information in modifying query and preparing optimal plan to access database

Run time database manager

- central software component of DBMS
- interfaces with user-submitted application programs and queries
- handles database access at run time
- converts operations in user's queries coming directly or indirectly via application program from user's logical view to physical file system
- accepts queries and examines external and conceptual schemas to determine what conceptual records are required to satisfy users request
- places call to physical database to perform request
- enforces constraints to maintain consistency and integrity of data, as well as its security; also performs backing and recovery operations
- sometimes referred to as *database control system*

Run time database manager / database control system components

- authorization control - checks that user has necessary authorization to carry out required operation
- command processor - processes queries passed by authorization control module
- integrity checker - checks for necessary integrity constraints for all requested operations that changes database
- query optimizer - determines optimal strategy for query execution; uses information on how data is stored to produce efficient execution plan for evaluating query

Run time database manager / database control system components

- transaction manager - performs required processing of operations it receives from transactions; ensures that transactions request and release locks according to suitable locking protocol and schedules execution of transactions
- scheduler - responsible for ensuring that concurrent operations on database proceed without conflicting with one another; controls relative order in which transaction operations are executed
- data manager - responsible for actual handling of data in database
  - recovery manager - ensures that database remains in consistent state in presence of failures; responsible for transaction commit and abort operations, maintaining log, and restoring system to consistent state after crash
  - buffer manager - responsible for transfer of data between main memory and secondary storage (disk); brings in pages from disk to main memory as needed in response to read user requests (sometimes referred as cache manager)

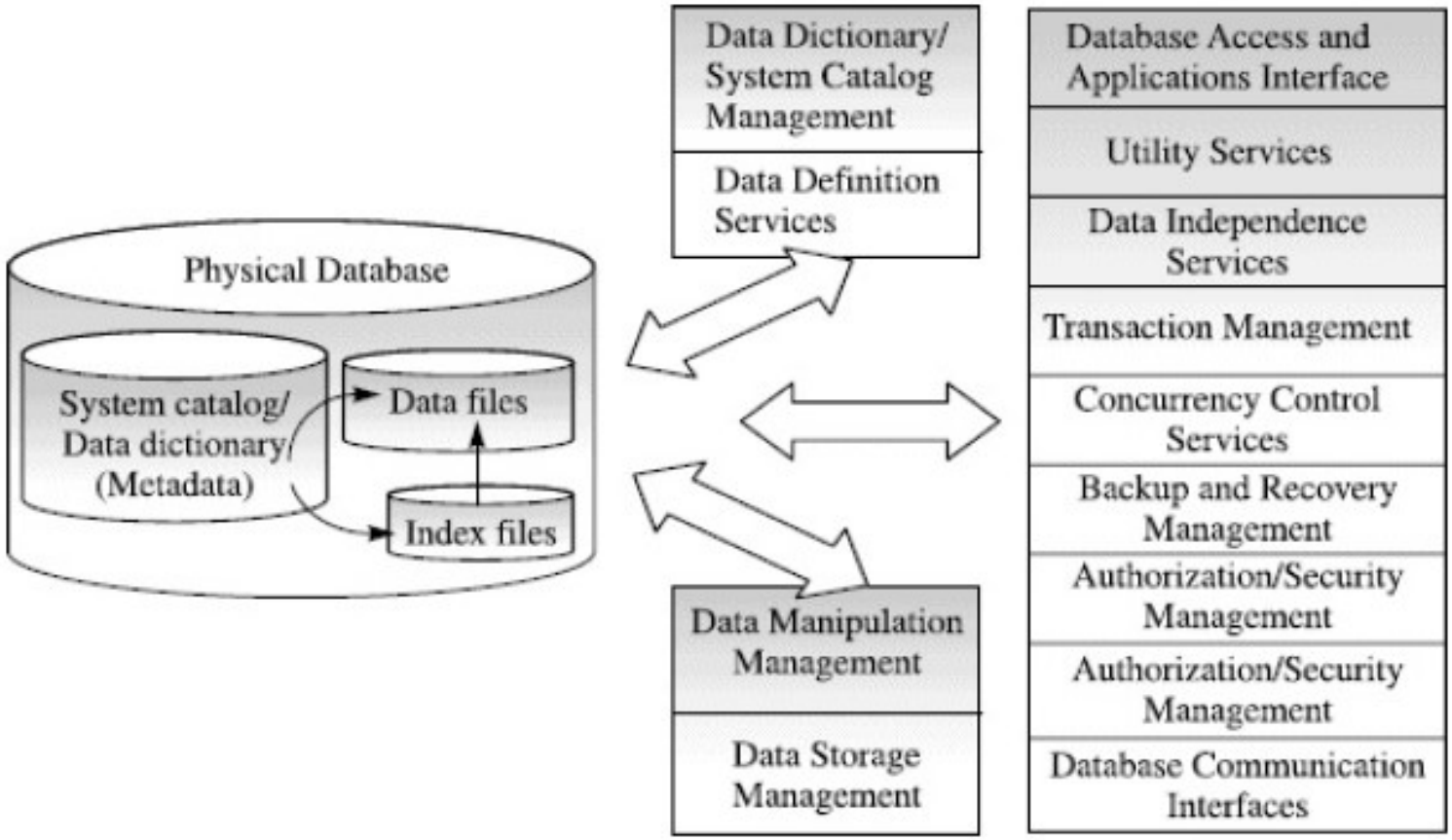
DML processor

- using DML compiler, converts DML statements embedded in application program into standard function calls in host language
- compiler converts DML statements written in host programming language into object code for database access
- processor must interact with query processor to generate appropriate code

DDL processor

- using DDL compiler, converts DDL statements into set of tables containing metadata; these tables contain metadata concerning database and are in form that can be used by other components of DBMS; tables are then stored in system catalog while control information is stored in data file headers
- DDL compiler processes schema definitions, specified in DDL and stores description of schema (metadata) in DBMS system catalog
- system catalog includes information such as names of data files, data items, storage details of each data file, mapping information amongst schemas, and constraints

Functions and Services of DBMS



Data Storage Management

- DBMS creates complex structures required for data storage in physical database, provides mechanism for management of permanent storage of data
- internal schema defines data should be stored by storage management mechanism and storage manager interfaces with operating system to access physical storage
- relieves users from task of defining and programming physical data characteristics
- DBMS provides not only for data, but also for related data entry forms or screen definitions, report definitions, data validation rules, procedural code, structure to handle pictures, sounds and video formats, and so on

Data Definition Services

- DBMS accepts data definitions such as external schema, conceptual schema, internal schema, and all the associated mappings
- converts them to appropriate object form using DDL processor component for each of various data definition languages (DDLs)

Data Dictionary
System Catalog Management

- DBMS provides data dictionary or system catalog in which descriptions of data items are stored and which is accessible to users; repository of information describing data
- data about data or metadata
- all schemas and mappings, all security and integrity constraints are stored
- automatically created by DBMS and consulted frequently to resolve user requests

Data Manipulation Management

- DBMS furnishes users with ability to retrieve, update and delete existing data
- includes DML processor component

Authorization / Security Management

- DBMS protects database against unauthorized access - mechanism to ensure that only authorized users can access database
- enforces user security and data privacy within database
- security rules determine which users can access database, which data items each user may access and which data operations (read, add, delete and modify) user may perform

Database Communication Interfaces

- user's requests for database are transmitted to DBMS in the form of communication messages
- DBMS provides special communication routines designed to allow database to accept user requests within computer network environment
- response to user is transmitted back from DBMS in form of such communication messages
- DBMS integrates with *data communication manager* (*DCM*), which controls such message transmission activities

Backup and Recovery Management

- DBMS provides mechanisms for backing up data periodically and recovering from different types of failures
- prevents loss of data; ensures that aborted or failed transactions do not create any adverse effect
- recovery mechanisms of DBMSs make sure that database is returned to consistent state after transaction fails or aborts due to system crash

Concurrency Control Services

- DBMSs support sharing of data among multiple users
- must provide mechanism for managing concurrent access to database
- DBMS ensure that database is kept in consistent state and that the integrity of data is preserved
- ensures that database is updated correctly when multiple users are updating database concurrently

Transaction Management

- transaction is series of database operations, carried out by single user or application program, which accesses or changes contents of database
- DBMS must provide mechanism to ensure either that all updates corresponding to given transaction are made or that none of them is made

Integrity Services

- database integrity refers to correctness and consistency of stored data (important in transaction-oriented database system)
- DBMS must provide means to ensure that both data in database and changes to data follow certain rules to maximizes data consistency
- data relationships stored in data dictionary are used to enforce data integrity
- various types of integrity mechanisms and constraints may be supported to ensure that data values are valid, that operations performed on those values are valid and database remains in consistent state

Utility Services

- DBMS provides set of utility services used by DBA and database designer to create, implement, monitor and maintain database
- help DBA to administer database effectively

Database Access and Application Programming Interfaces

- provide interface to enable applications to use DBMS services; provide data access via structured query language (SQL)
- DBMS query language contains two components:
  - data definition language (DDL) defines structure in which data are stored
  - data manipulation language (DML) allows users to extract data
- DBMS provides data access to application programmers via procedural languages such as C, PASCAL, COBOL, Visual BASIC, .NET, php, Java and others

Data Independence Services

- DBMS must support independence of programs from actual structure of database

Data Models

Model

- is an abstraction that concentrates essential aspects of organization's applications while ignores details
- is representation of real world objects and events and their associations
- *data model* is mechanism that provides this abstraction for database application

Data Model

- represents organization - provides basic concepts and notations to allow database designers and users unambiguously and accurately communicate their understanding of organizational data
- data modelling used for representing entities of interest and their relationships in database
- conceptual method of structuring data

Data Model

- collection of mathematically well-defined concepts that help organization to express static and dynamic properties of data intensive applications
- consists of:
  - static properties, for example, objects, attributes and relationships
  - integrity rules over objects and operations
  - dynamic properties, for example, operations or rules defining database states

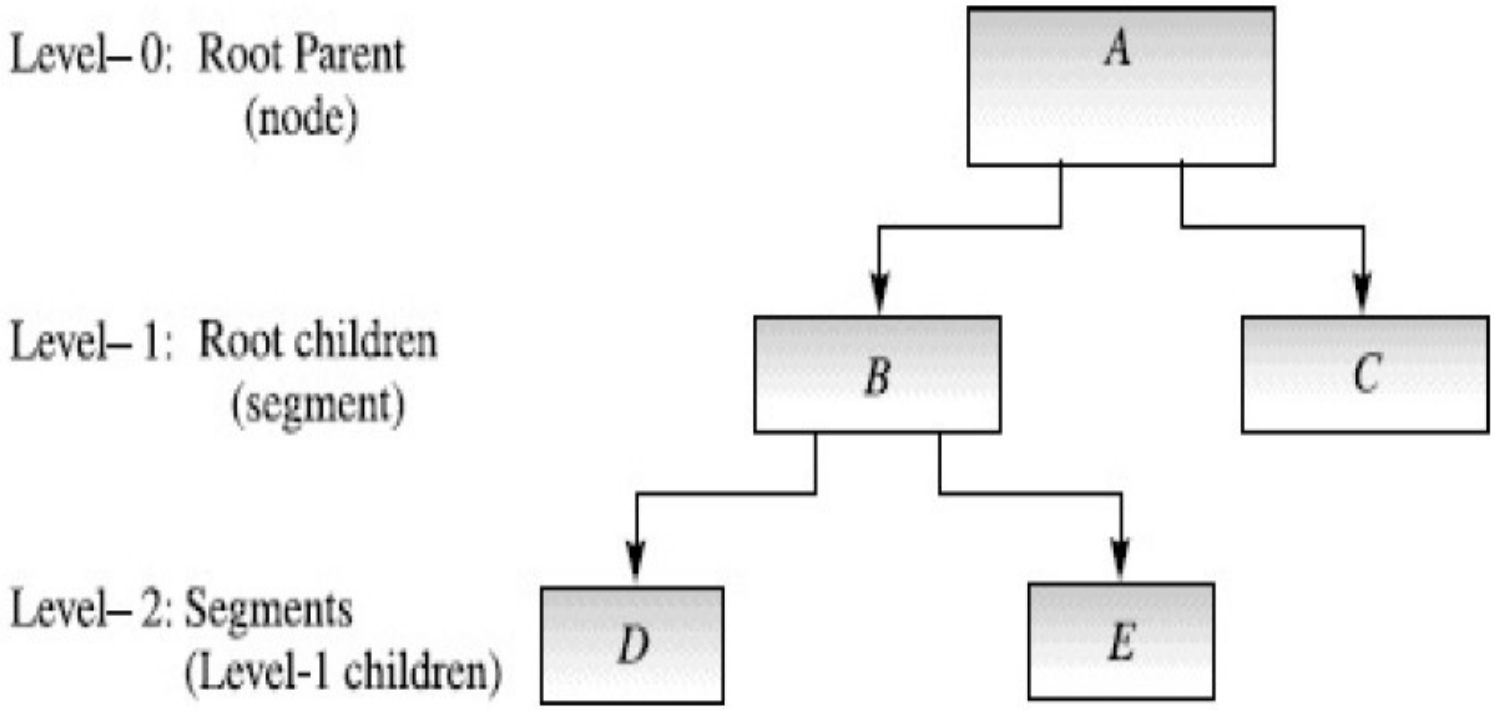
Data Model

- can be classified into:
  1. record-based data models
     - Hierarchical data model.
     - Network data model.
     - Relational data model.
  2. object-based data models
     - Entity-relationship.
     - Object-oriented.
- DBMS support a single data model

Hierarchical Data Models

- represented by upside-down tree
- user perceives as hierarchy of segments
- segment equivalent of file system's record type
- relationship between files or records forms a hierarchy

Hierarchical Data Models



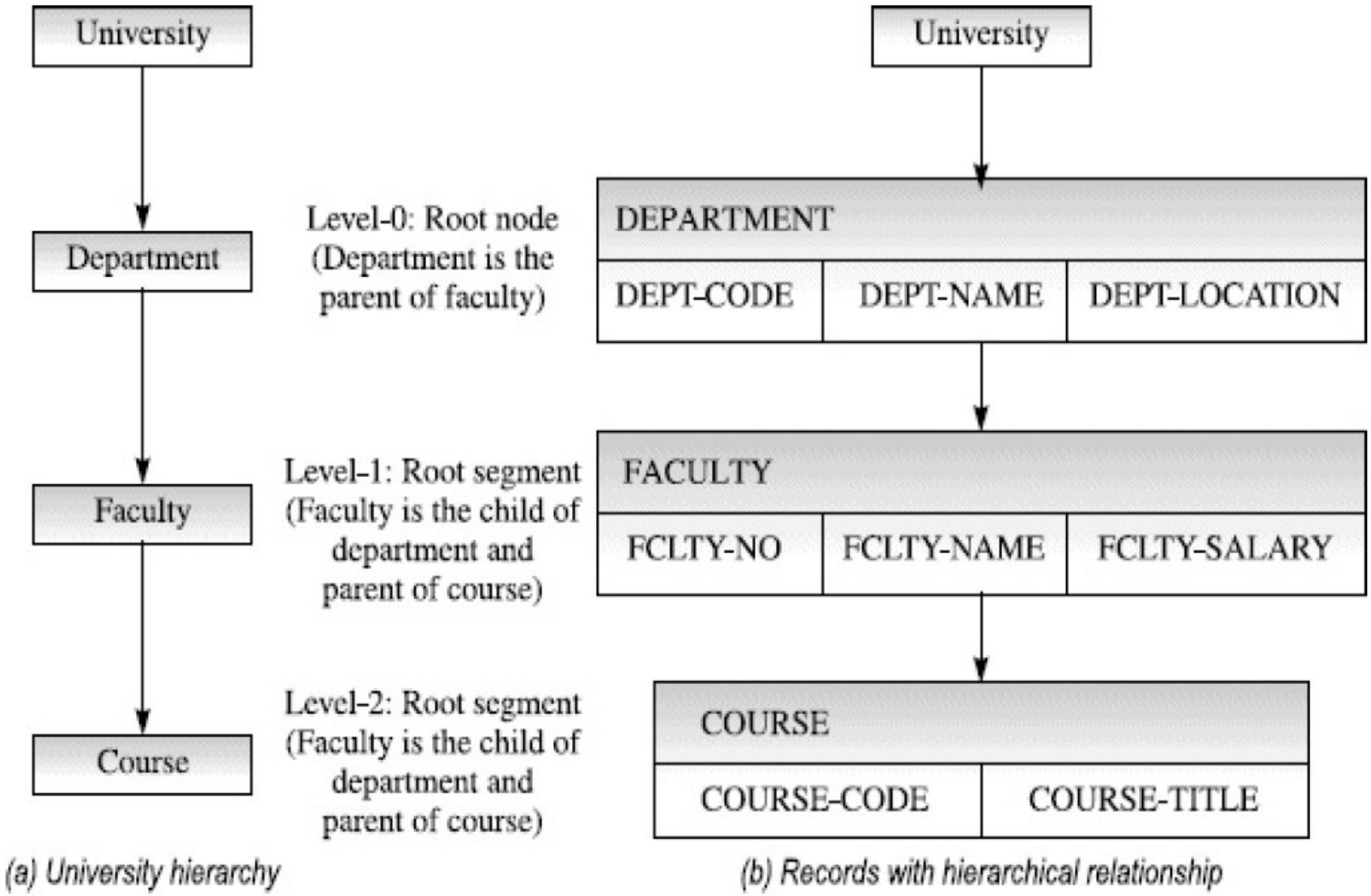
Hierarchical Data Models

- tree defined as set of nodes such that there is one specially designated node called root (node)
- remaining nodes are portioned into disjoint sets and perceived as children of segment above them
- each disjoint set in turn is a tree

Hierarchical Data Models

- can represent one-to-many relationship between two entities where the two are respectively parent and child
- nodes of tree represent record types

- hierarchical path that traces parent segments to child segments, beginning from left, defines the tree
- hierarchical path for segment ‘E’ can be traced as ABDE, tracing all segments from root starting at leftmost segment
- left-traced path is known as *preorder traversal* or *hierarchical sequence*


University
Department
Faculty
Course
Level-0: Root node (Department is the parent of faculty)
Level-1: Root segment (Faculty is the child of department and parent of course)
Level-2: Root segment (Faculty is the child of department and parent of course)
(a) University hierarchy
University
DEPARTMENT
DEPT-CODE
DEPT-NAME
DEPT-LOCATION
FACULTY
FCLTY-NO
FCLTY-NAME
FCLTY-SALARY
COURSE
COURSE-CODE
COURSE-TITLE
(b) Records with hierarchical relationship

Hierarchical Data Models

- one of oldest database models used by enterprise
- IBM *Information Management System* (*IMS*) became world's leading hierarchical database

advantages

- simplicity
- data sharing
- data security
- data independence
- data integrity
- efficiency
- available expertise
- tried business applications

disadvantages

- implementation complexity
- inflexibility
- database management problems
- lack of structural independence
- application programming complexity
- implementation limitation: many common relationships do not conform to one-to-many
- no standards
- extensive programming efforts

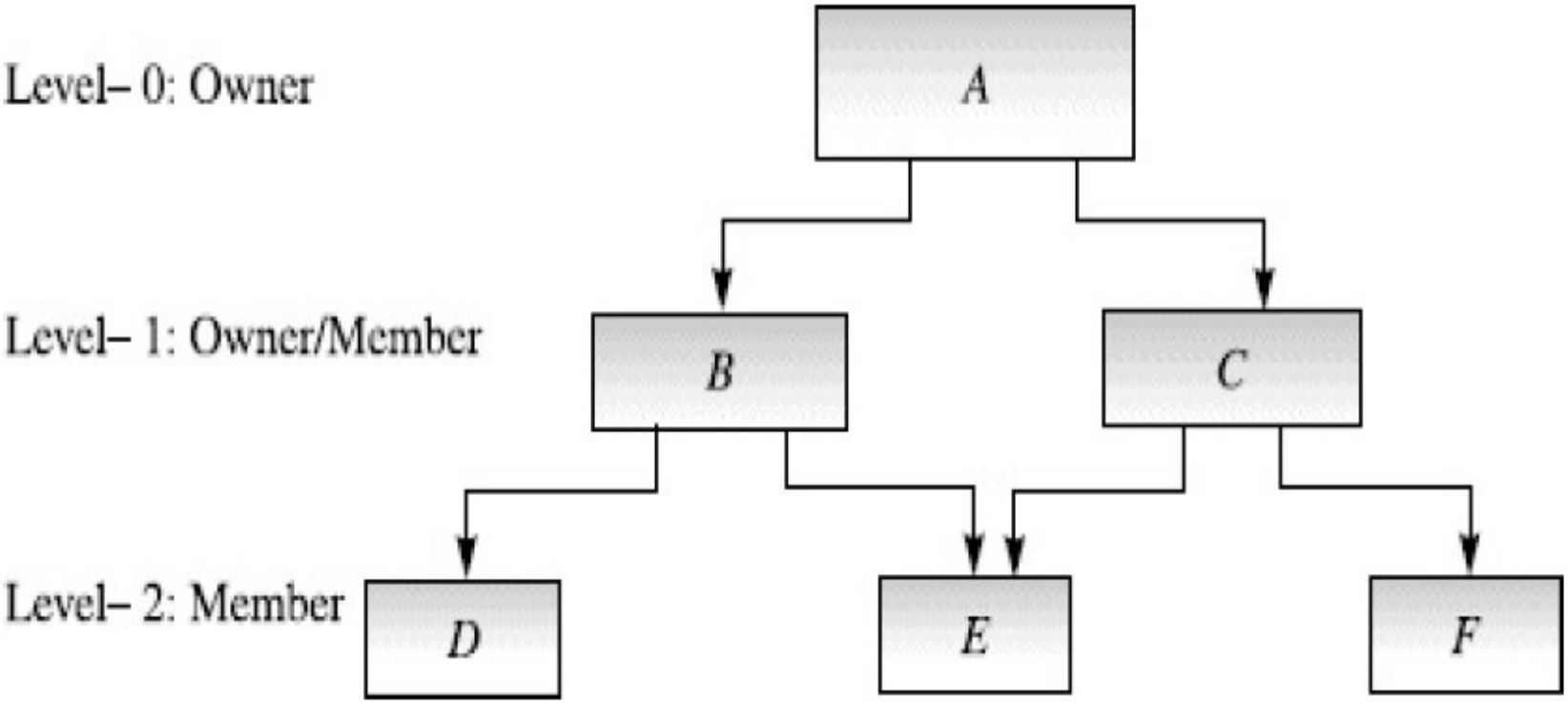
Network Data Model

- similar to hierarchical model except that record can have multiple parents
- has three basic components: record type, data items (or fields) and links
- relationship is called *set* in which each set is composed of at least two record types - first record type is *owner* equivalent to parent and second record type is *member* equivalent to child - connection between owner and its member records is identified by link to which database designers assign *set-name;* this set-name used to retrieve and manipulate data

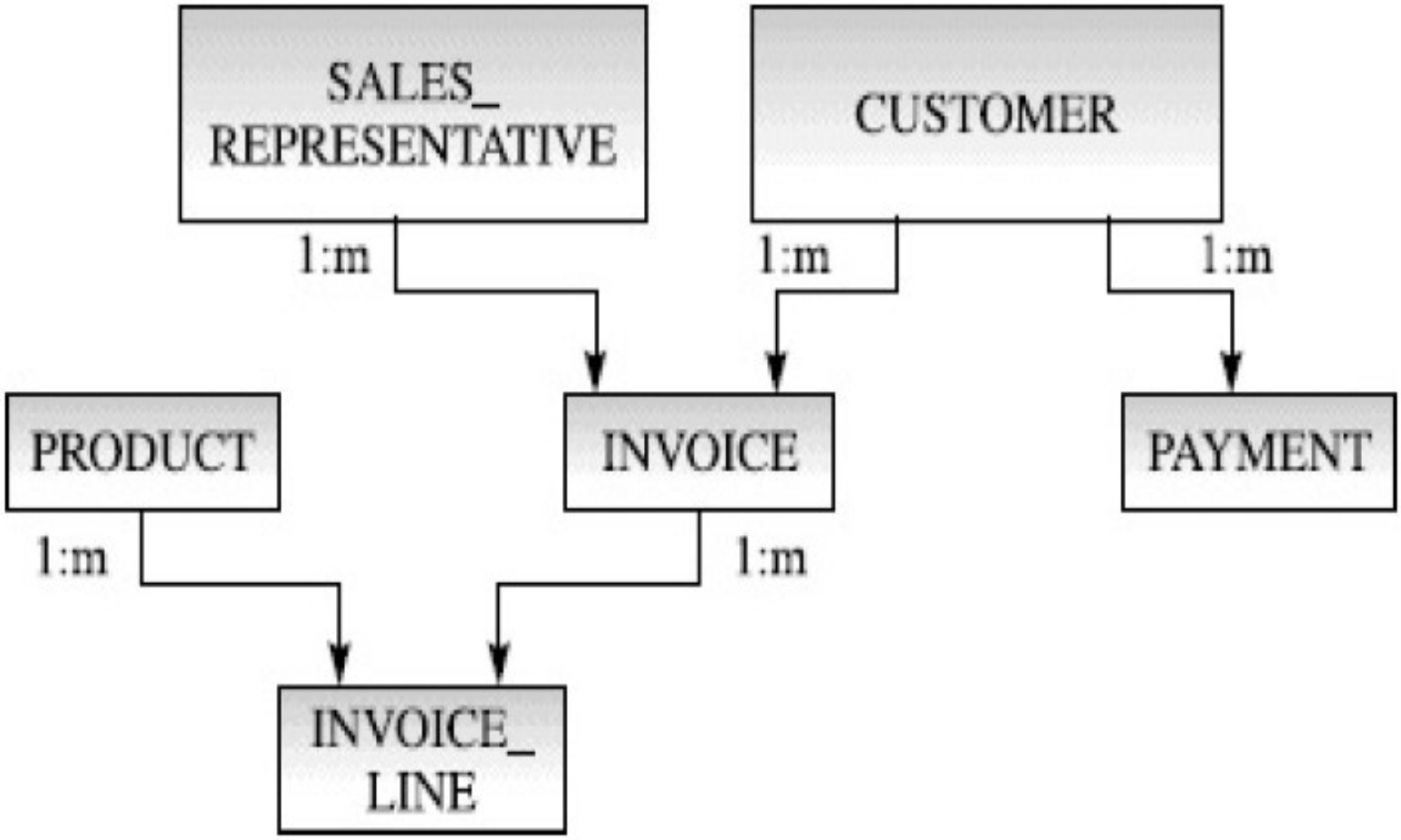
Network Data Model

- links between owners and their members indicate access paths in network models and are typically implemented with pointers
- member can appear in more than one set and thus may have several owners, and therefore, it facilitates many-to-many relationships

Network Data Model



Network Data Model for sales organization



advantages

- simplicity
- facilitating more relationship types
- superior data access
- database integrity
- data independence
- database standards

disadvantages

- system complexity
- absence of structural independence
- not a user-friendly

Relational Data Model

- *Relational Database Management System* (*RDBMS*) performs same basic functions
- relational data model simplifies user's view of database using simple tables instead of more complex tree and network structures
- in Relational Data Model database it is collection of tables (also called relations)
- each table is matrix of series of rows and columns
- tables are related by sharing common entity characteristic

advantages

- Simplicity - is even simpler than hierarchical and network models; frees designers from actual physical data storage details, thereby allowing them to concentrate on logical view of database
- structural independence - does not depend on navigational data access system; changes in database structure do not affect data access

advantages

- ease of design, implementation, maintenance and uses - provides both structural independence and data independence; it makes database design, implementation, maintenance and usage much easier
- flexible and powerful query capability - provides very powerful, flexible, and easy-to-use query facilities; structured query language (SQL) capability makes ad hoc queries a reality

disadvantages

- **hardware overheads -** need more powerful computing hardware and data storage devices to perform RDMS-assigned tasks
  - tend to be slower than other database systems
  - with rapid advancement in computing technology (1970-80) disadvantage of being slow is getting faded
- **easy-to-design capability leading to bad design -** easy-to-use feature of relational database results into untrained people generating queries and reports without much understanding and giving much thought to need of proper database design; with growth of database, poor design results into slower system, degraded performance and data corruption

disadvantages

- recent disadvantages (in the Internet era)
  - big data
  - un structured data

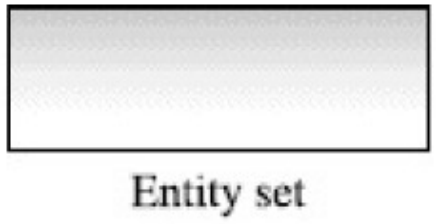
Entity-Relationship (E-R) Data Model

- is a logical database model, which has logical representation of data for enterprise, business
- is a collection of objects of similar structures called entity set
- relationship between entity sets is represented on basis of number of entities from entity set that can be associated with number of entities of another set
  - such as one-to-one (1:1), one-to-many (1:n), or many-to-many (n:n) relationships

E-R Data Model diagram

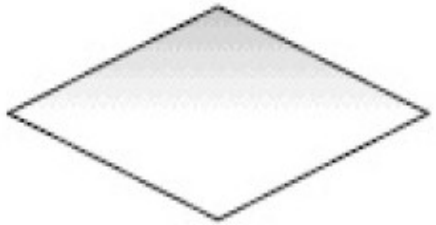
- E-R diagram is shown graphically
- E-R diagram has become widely acccepted data model used for designing of relational databases

Building blocks (symbols) of diagram of E-R Data Model



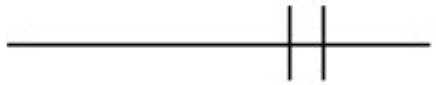
Entity set

Attributes

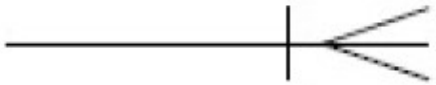


Relationship

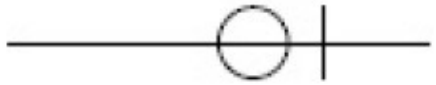
Link between attribute and entity set



Mandatory One



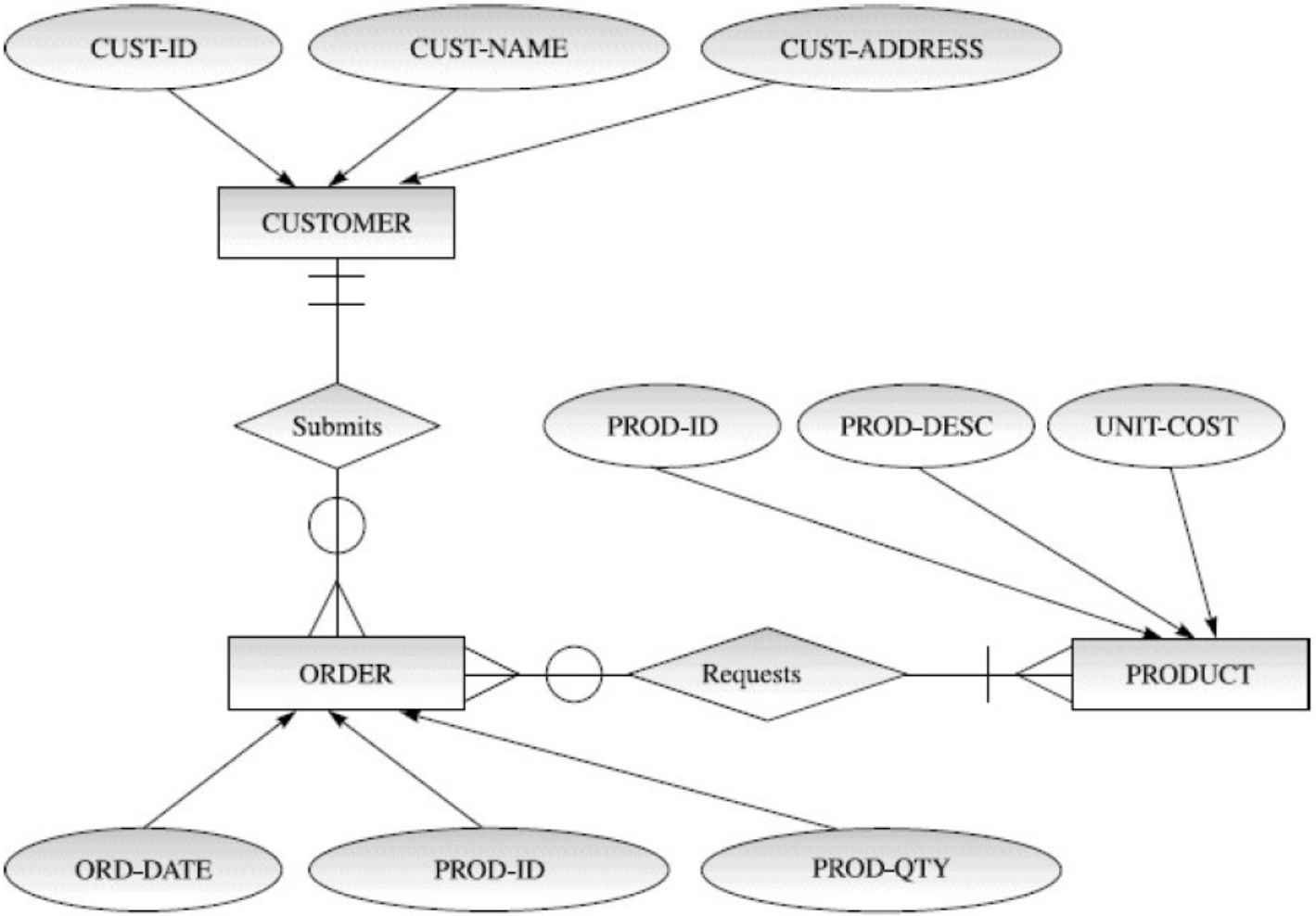
Mandatory Many



Optional One

Optional Many

E-R Data Model diagram for product sales organization



advantages

- straightforward relational representation
- easy conversion for E-R to other data model
- graphical representation for better understanding

disadvantages

- no industry standard for notation
- popular only for high-level database design

Object-Oriented Data Model

- logical data model that captures semantics of objects supported in OOP
- persistent and sharable collection of defined objects
- has the ability to model complete solution
- database models represent entity and class; class represents both object attributes as well as behavior of entity
- maintains relationships through *logical containment*

Object-Oriented Data Model

- structure is highly variable, unlike traditional databases (such as hierarchical, network or relational), it has no single inherent database structure
- structure for any given class could be anything programmer finds useful (for example, linked list, set, array, …); may contain varying degrees of complexity, making use of multiple types and multiple structures

advantages

- capable of handling a large variety of data types
- combining object-oriented programming with database technology
- improved productivity
- improved data access

disadvantages

- no precise definition
- difficult to maintain
- not suited for all applications

Review Questions

Review Questions

- Describe the three-tier ANSI-SPARC architecture. Why do we need mappings between different schema levels? How do different schema definition languages support this architecture?
- Discuss the advantages and characteristics of the three-tier architecture.
- Discuss the concept of data independence and explain its importance in a database environment.
- What is logical data independence and why is it important?
- What is the difference between physical data independence and logical data independence?

Review Questions

- How does the ANSI-SPARC three-tier architecture address the issue of data independence?
- Explain the difference between external, conceptual and internal schemas. How are these different schema layers related to the concepts of physical and logical data independence?
- Describe the structure of a DBMS.
- Describe the main components of a DBMS.
- Explain the structure of DBMS.

Review Questions

- What is a transaction?
- How does the hierarchical data model address the problem of data redundancy?
- What do you mean by a data model? Describe the different types of data models used.
- Explain the following with their advantages and disadvantages:
  - Hierarchical database model
  - Network database model
  - Relational database model
  - E-R data models
  - Object-oriented data model.

Review Questions

- Define the following terms:
  - Data independence
  - Query processor
  - DDL processor
  - DML processor.
  - Run time database manager.
- How does the hierarchical data model address the problem of data redundancy?
- Describe the basic features of the relational data model. Discuss their advantages, disadvantages and importance to the end-user and the designer.

- as always it is not possible to close whiteout **thank you for your kindly attention!**
- *If you get half as much pleasure - the guilty variety, to be sure - from reading this slides as I get from writing it, we're all doing pretty well.*