Chapter 2

Database Environment

Chapter 2 - Objectives

- Purpose of three-level database architecture.
- Contents of external, conceptual, and internal levels.
- Purpose of external/conceptual and conceptual/internal mappings.
- Meaning of logical and physical data independence.
- Distinction between DDL and DML.
- A classification of data models.

Chapter 2 - Objectives

- Purpose/importance of conceptual modeling.
- Typical functions and services a DBMS should provide.
- Function and importance of system catalog.
- Software components of a DBMS.
- Meaning of client–server architecture and advantages of this type of architecture for a DBMS.
- Function and uses of Transaction Processing Monitors.

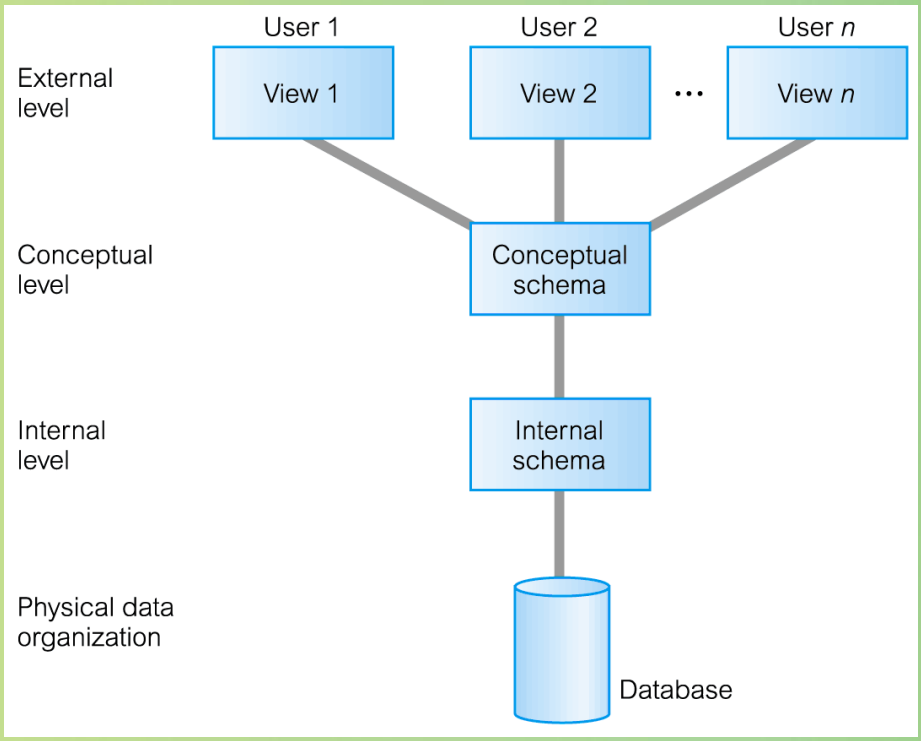
Objectives of Three-Level Architecture

- **The American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC) recognized the need for a three-level approach with a system catalog.**
- **Although the ANSI-SPARC model did not become a standard, it still provides a basis for understanding some of the functionality of a DBMS.**

ANSI-SPARC Three-Level Architecture

The **three-level approach** form a **three-level architecture** comprising an **external**, **a conceptual**, and an **internal** level, as depicted in Figure 2.1.

- The way users perceive the data is called the **external level**.
- **Conceptual Level:** Describes what data is stored in database and relationships among the data.
- **Internal Level:** Describes Physical representation of the database on the computer.



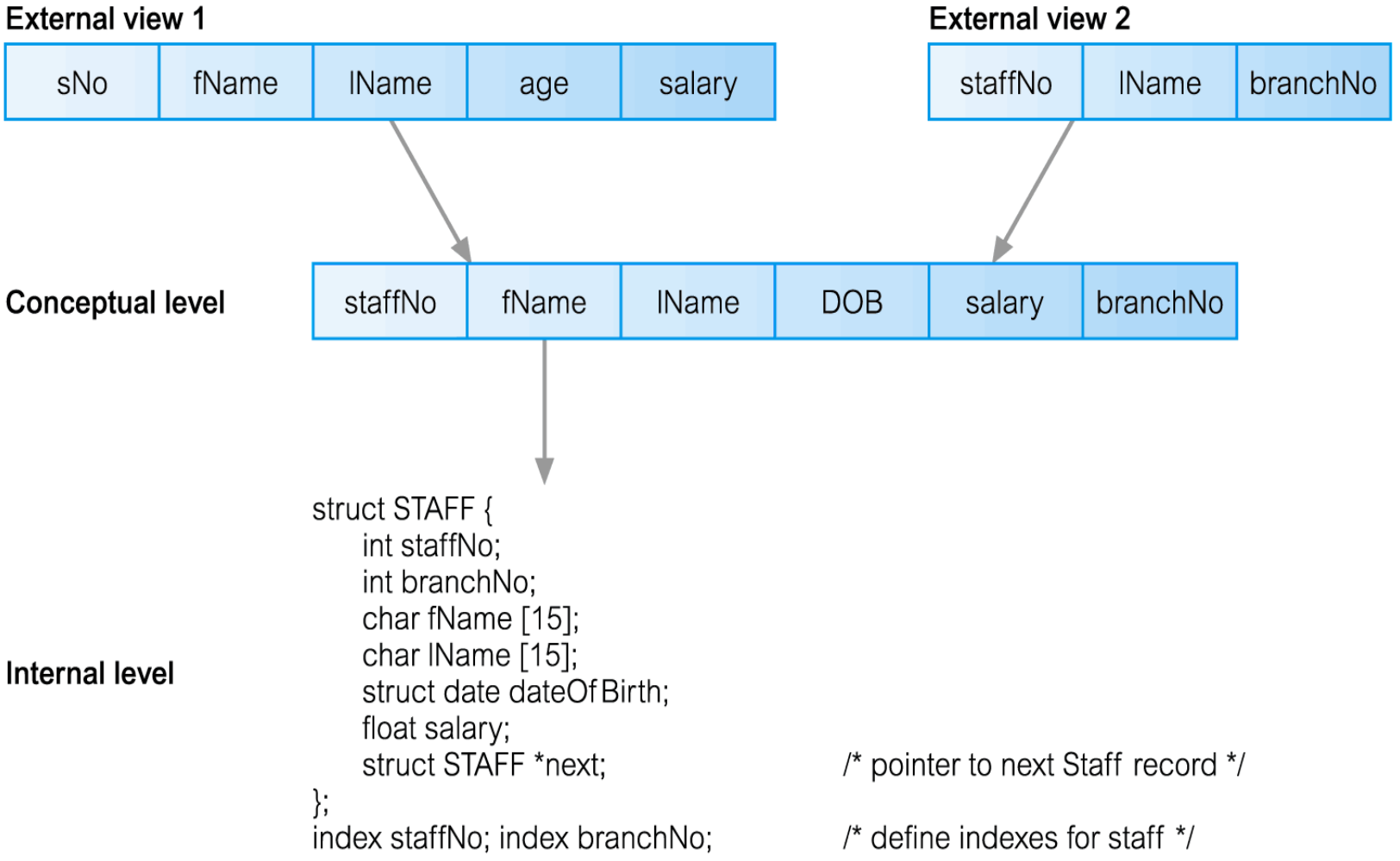
Objectives of Three-Level Architecture

- All users should be able to access same data.
- A user's view is immune to changes made in other views.
- Users should not need to know physical database storage details.

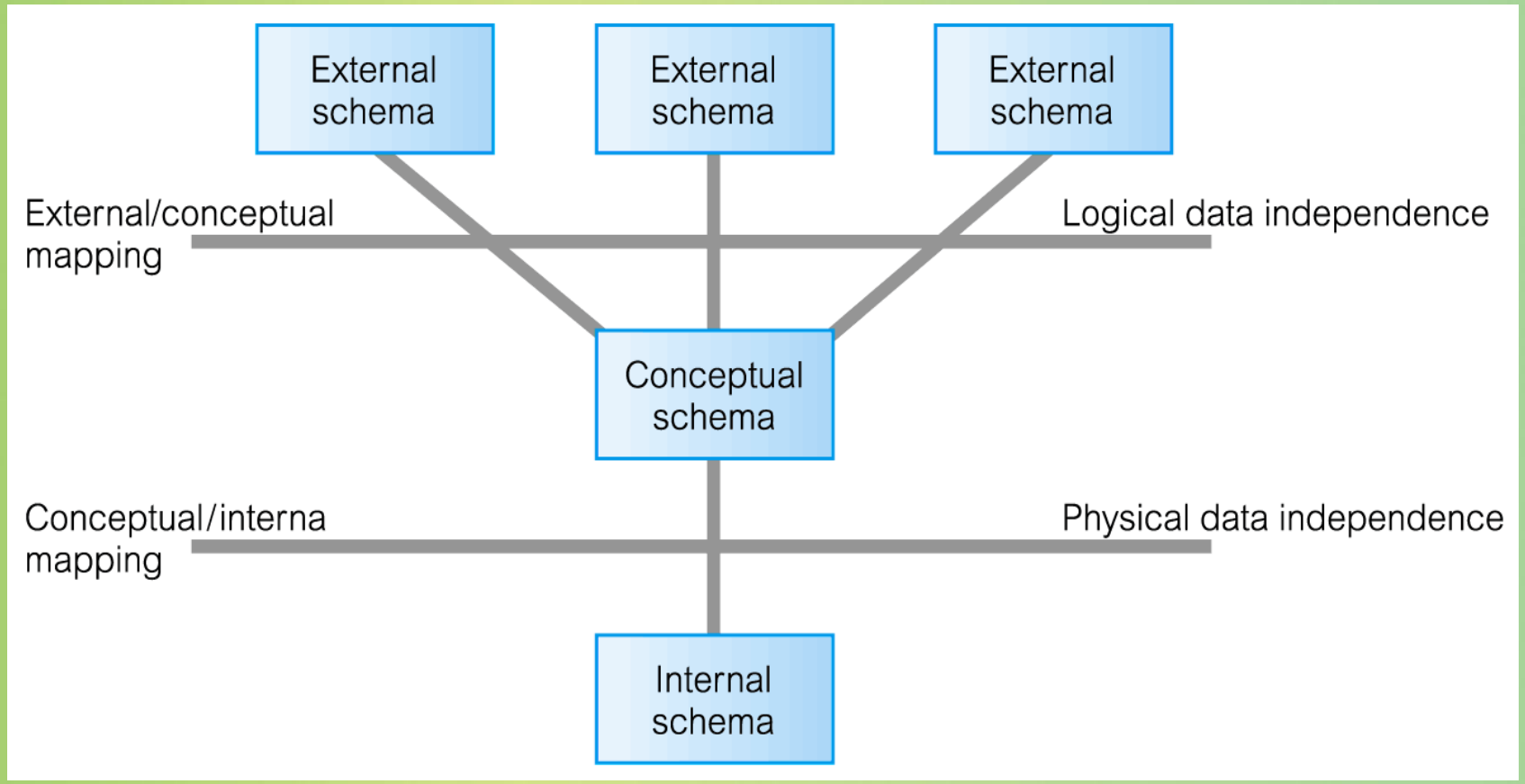
Objectives of Three-Level Architecture

- DBA should be able to change database storage structures without affecting the users' views.
- Internal structure of database should be unaffected by changes to physical aspects of storage.
- DBA should be able to change conceptual structure of database without affecting all users.

Differences between Three Levels of ANSI-SPARC Architecture



Data Independence and the ANSI-SPARC Three-Level Architecture



Data Independence

- A major objective for the three-level architecture is to provide **data independence**, which means that upper levels are unaffected by changes to lower levels.
- There are two kinds of data independence: **logical and physical**.

Data Independence

- **Logical Data Independence**
  - **It refers to immunity of external schemas to changes in conceptual schema.**
  - **Conceptual schema changes (e.g. addition/removal of entities).**
  - **Should not require changes to external schema or rewrites of application programs.**

Data Independence

- **Physical Data Independence**
  - **It refers to immunity of conceptual schema to changes in the internal schema.**
  - **Internal schema changes (e.g. using different file organizations, storage structures/devices).**
  - **Should not require change to conceptual or external schemas.**

Database Languages

- A data sublanguage consists of two parts: a **Data Definition Language (DDL)** and a **Data Manipulation Language (DML).**
- The **DDL** is used to specify the **database schema**, and the **DML** is used to both **read and update the database**.
- These languages are called *data sublanguages* because they do not include constructs for all computing needs, such as conditional or iterative statements, which are provided by the high-level programming languages.
- Many DBMSs have a facility for *embedding* the sublanguage in a high-level programming language such as COBOL, Fortran, Pascal, Ada, C, C++, C#, Java, or Visual Basic.
- In this case, the high-level language is sometimes referred to as the *host language*.

Database Languages

- **Data Definition Language (DDL)**
  - It allows the DBA or user to describe and name entities, attributes, and relationships required for the application plus any associated integrity and security constraints**.**
  - The result of the compilation of the DDL statements is a set of tables stored in special files collectively called the **system catalog**.
  - The system catalog integrates the **metadata**, which is data that describes the objects in the database and makes it easier for those objects to be accessed or manipulated.
  - The metadata contains definitions of records, data items, and other objects that are of interest to users or are required by the DBMS.

Database Languages

- Data Manipulation Language (DML)
  - Provides basic data manipulation operations on data held in the database.
  - Procedural DML
    - Allows user to tell system exactly how to manipulate data.
  - Non-Procedural DML
    - Allows user to state what data is needed rather than how it is to be retrieved.
- Fourth Generation Languages (4GLs)

Database Languages

- **Fourth Generation Languages (4GLs)**
  - **In essence, it is a shorthand programming language. An operation that requires hundreds of lines of code in a third-generation language (3GL), such as COBOL, generally requires significantly fewer lines in a 4GL.**
  - **Compared with a 3GL, which is procedural, a 4GL is nonprocedural: the user defines *what* is to be done, not *how*.**
  - **A 4GL is expected to rely largely on much higher-level components known as fourth-generation tools**

Data Model

- It is an integrated collection of concepts for describing data, relationships between data, and constraints on the data in an organization.
- It represents the real-world objects and events, and their associations.
- It is an abstraction that concentrates on the essential, inherent aspects of an organization.
- Data Model comprises:
  - a structural part (a set of rules according to which databases can be constructed);
  - a manipulative part (operation that are allowed on the data);
  - possibly a set of integrity rules (which ensures that the data is accurate).

Data Model

- Purpose
  - To represent data in an understandable way.
- Categories of data models include:
  - Object-based
  - Record-based
  - Physical.

Data Models

- **Object-based data models use concepts such as entities, attributes, and relationships.**
- **An entity is a distinct object (a person, place, thing, concept, event) in the organization that is to be represented in the database. An attribute is a property that describes some aspect of the object that we wish to record, and a relationship is an association between entities.**
- **Some of the more common types of object-based data model are:**
  - **Entity-Relationship**
  - **Semantic**
  - **Functional**
  - **Object-Oriented.**

Data Models

- In a record-based model, the database consists of a number of fixed-format records, possibly of differing types.
- Each record type defines a fixed number of fields, typically of a fixed length.
- There are three principal types of record-based logical data model:
  - Relational data model.
  - Network data model.
  - Hierarchical data model.

Relational Data Model

- **Data and relationships are represented as tables, each of which has a number of columns with a unique name.**
- **Figure 2.4 is a sample instance of a relational schema for part of the *DreamHome* case study, showing branch and staff details.**

Branch

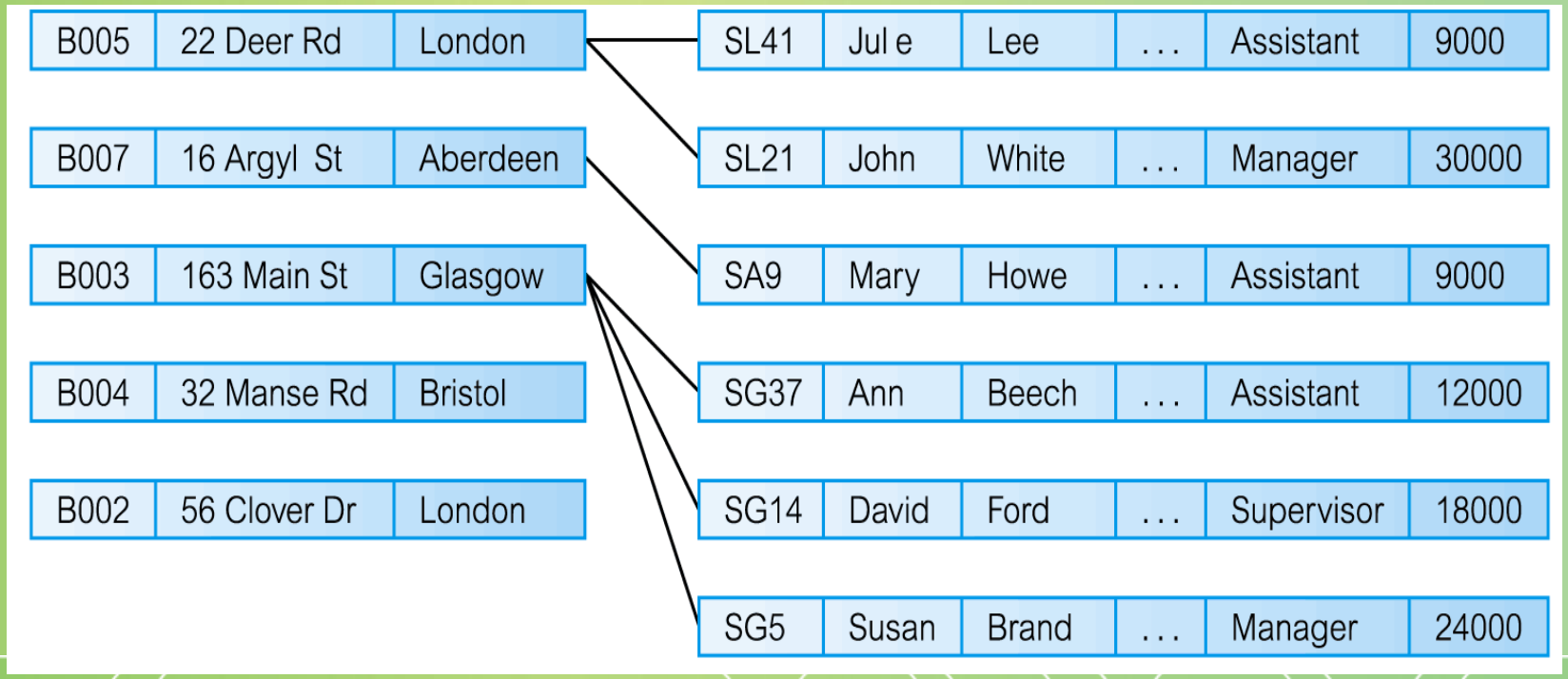
branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

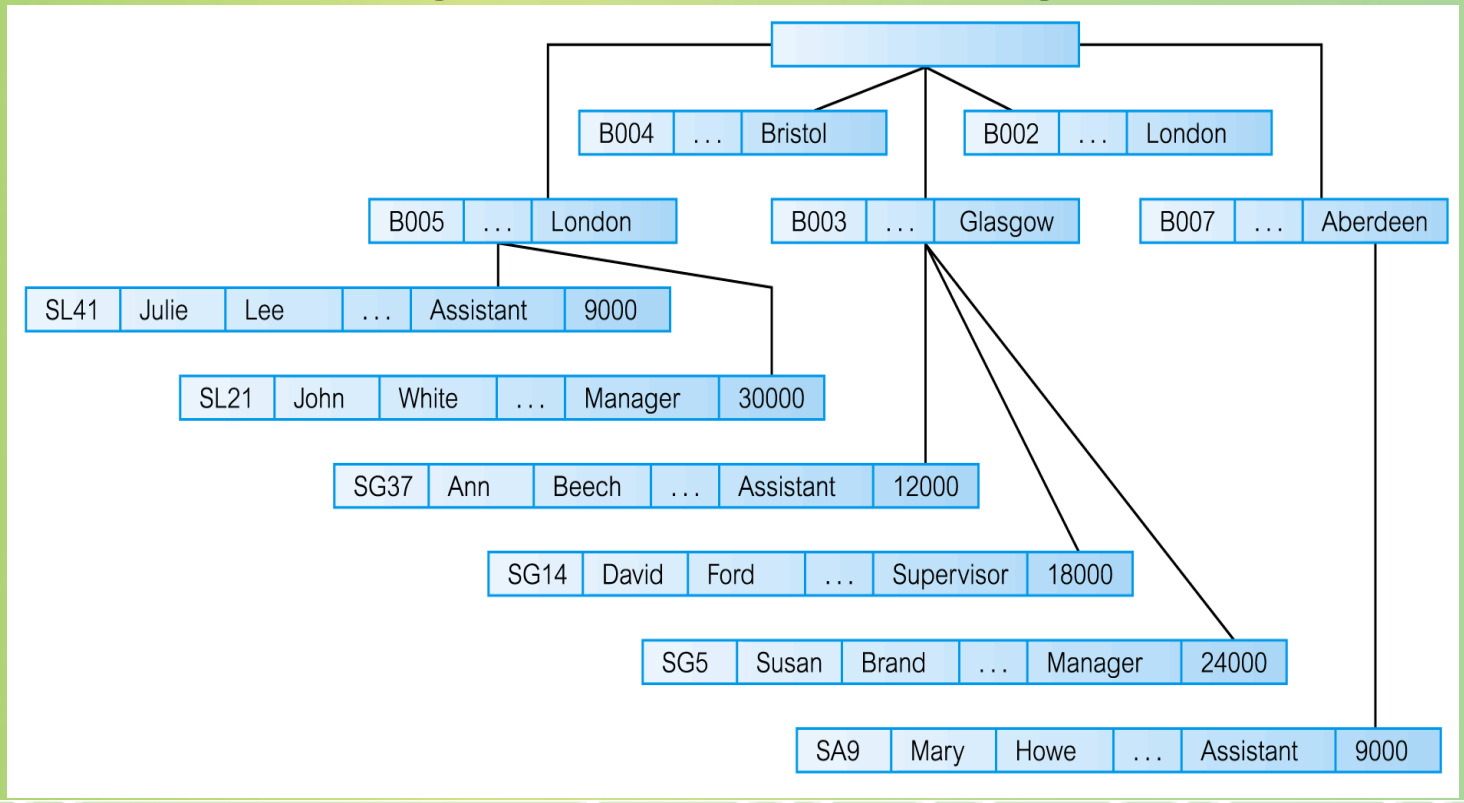
Network Data Model

- Data is represented as collections of **records**.
- Relationships are explicitly modeled by the **sets**, which become pointers in the implementation.
- The records are organized as graph structures with records appearing as **nodes** (also called **segments)** and sets as **edges** in the graph.



Hierarchical Data Model

- It is a restricted type of network model.
- It allows a node to have only one parent.
- It can be represented as a tree graph, with records appearing as nodes (also called segments) and sets as edges.



Physical Data Models

- **Physical data models describe how data is stored in the computer, representing information such as record structures, record orderings, and access paths.**
- **There are not as many physical data models as logical data models; the most common ones are the *unifying model* and the *frame memory*.**

Conceptual Modeling

- **Conceptual schema is the core of a system supporting all user views.**
- **Should be complete and accurate representation of an organization's data requirements.**
- **Conceptual modeling is process of developing a model of information use that is independent of implementation details.**
- **Result is a conceptual data model.**

System Catalog

- Repository of information (metadata) describing the data in the database.
- One of the fundamental components of DBMS.
- Typically stores:
  - names, types, and sizes of data items;
  - constraints on the data;
  - names of authorized users;
  - data items accessible by a user and the type of access;
  - external, conceptual, and internal schemas;
  - usage statistics.

Functions of a DBMS

- **Data Storage, Retrieval, and Update.**
- **A User-Accessible Catalog.**
  - A DBMS must furnish a catalog in which descriptions of data items are stored and which is accessible to users.
- **Transaction Support.**
  - A DBMS must ensure either that all the updates corresponding to a given transaction are made or that none of them is made.
- **Concurrency Control Services.**
  - A DBMS must ensure that the database is updated correctly when multiple users are updating concurrently.
- **Recovery Services.**

Functions of a DBMS

- **Authorization Services.**
  - A DBMS must ensure that only authorized users can access the database.
- **Support for Data Communication.**
  - A DBMS must be capable of integrating with communication software.
- **Integrity Services.**
  - A DBMS must ensure that both the data in the database and changes to the data follow certain rules.
- **Services to Promote Data Independence.**
  - A DBMS must support the independence of programs from the actual structure of the database.
- **Utility Services.**