

Database Management Systems

Dr. Atif Ali Mohamed

www.dratifnimir.info

Mob. 0123393000 ... e-mail: atif965@hotmail.com

Course for Computer Science, Information
Technology, Software Engineering Diploma Students

Introduction to Databases

Section Objectives ...

- Some common uses of database systems ...
- Characteristics of file-based systems ...
- Problems with file-based approach ...
- Meaning of the term database ...
- Meaning of the term Database Management System (DBMS) ...
- Typical functions of a DBMS ...
- Major components of the DBMS environment ...
- Personnel involved in the DBMS environment ...
- History of the development of DBMSs ...
- Advantages and disadvantages of DBMSs ...

Examples of Database Applications

- Purchases from the supermarket ...
- Purchases using your credit card ...
- Booking a holiday at the travel agents ...
- Using the local library ...
- Taking out insurance ...
- Renting a video ...
- Using the Internet ...
- Studying at university ...

File-Based Systems ...

- Collection of application programs that perform services for the end users (e.g. reports) ...
- Each program defines and manages its own data ...

File-Based Processing ...

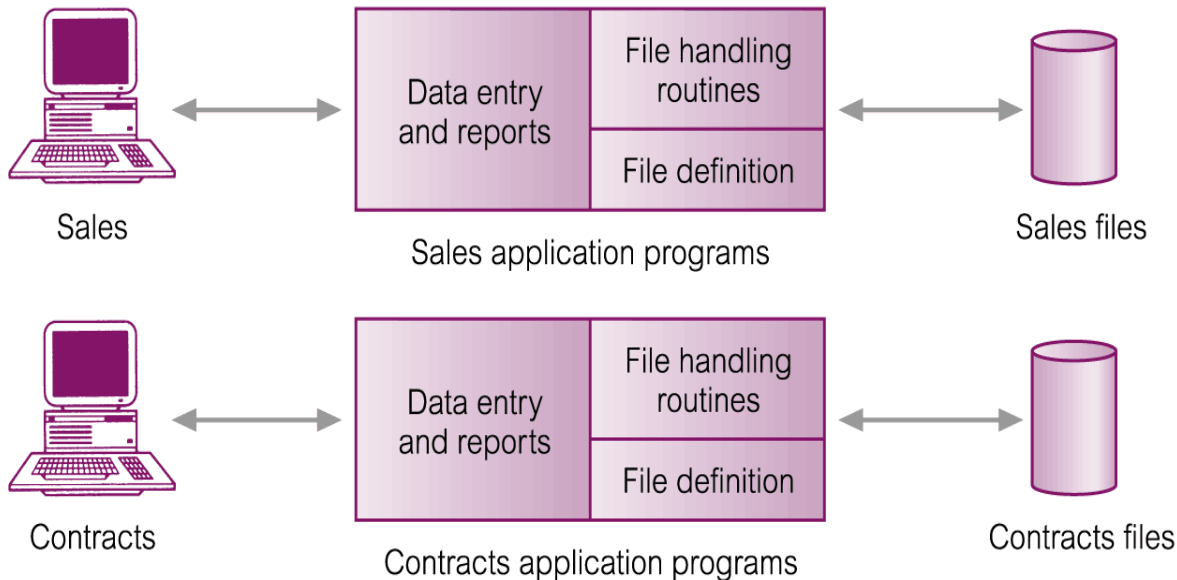


Figure 1.5

File-based processing.

Sales Files

PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)

PrivateOwner (ownerNo, fName, lName, address, telNo)

Client (clientNo, fName, lName, address, telNo, prefType, maxRent)

Contracts Files

Lease (leaseNo, propertyNo, clientNo, rent, paymentMethod, deposit, paid, rentStart, rentFinish, duration)

PropertyForRent (propertyNo, street, city, postcode, rent)

Client (clientNo, fName, lName, address, telNo)

Limitations of File-Based Approach

- Separation and isolation of data
 - Each program maintains its own set of data.
 - Users of one program may be unaware of potentially useful data held by other programs.
- Duplication of data
 - Same data is held by different programs.
 - Wasted space and potentially different values and/or different formats for the same item.

Limitations of File-Based Approach

- Data dependence
 - File structure is defined in the program code ...
- Incompatible file formats
 - Programs are written in different languages, and so cannot easily access each other's files ...
- Fixed Queries/Proliferation of application programs
 - Programs are written to satisfy particular functions ...
 - Any new requirement needs a new program ...

Database Approach ...

- Arose because:
 - Definition of data was embedded in application programs, rather than being stored separately and independently ...
 - No control over access and manipulation of data beyond that imposed by application programs ...
- Result:
 - The database and Database Management System (DBMS) ...

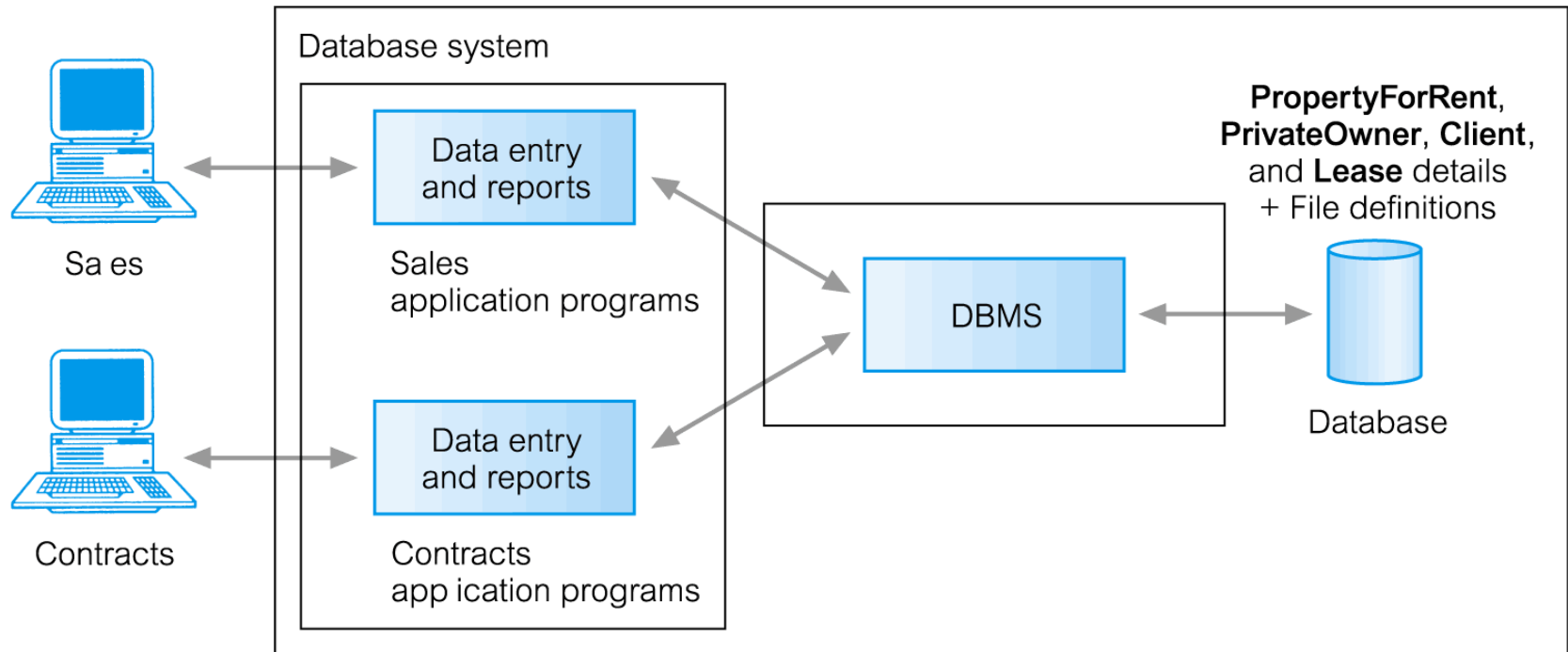
Database ...

- Shared collection of logically related data (and a description of this data), designed to meet the information needs of an organization ...
- System catalog (metadata) provides description of data to enable program–data independence ...
- Logically related data comprises entities, attributes, and relationships of an organization's information ...

Database Management System (DBMS)

- **DBMS**: a software system that enables users to define, create, maintain, and control access to the database ...
- **Database application program**: a computer program that interacts with database by issuing an appropriate request (SQL statement) to the DBMS ...

Database Management System (DBMS)



PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)

PrivateOwner (ownerNo, fName, lName, address, telNo)

Client (clientNo, fName, lName, address, telNo, prefType, maxRent)

Lease (leaseNo, propertyNo, clientNo, paymentMethod, deposit, paid, rentStart, rentFinish)

Database Approach ...

- Data definition language (DDL).
 - Permits specification of data types, structures and any data constraints ...
 - All specifications are stored in the database ...
- Data manipulation language (DML).
 - General enquiry facility (query language) of the data ...

Database Approach ...

- Controlled access to database may include:
 - A security system ...
 - An integrity system ...
 - A concurrency control system ...
 - A recovery control system ...
 - A user-accessible catalog ...

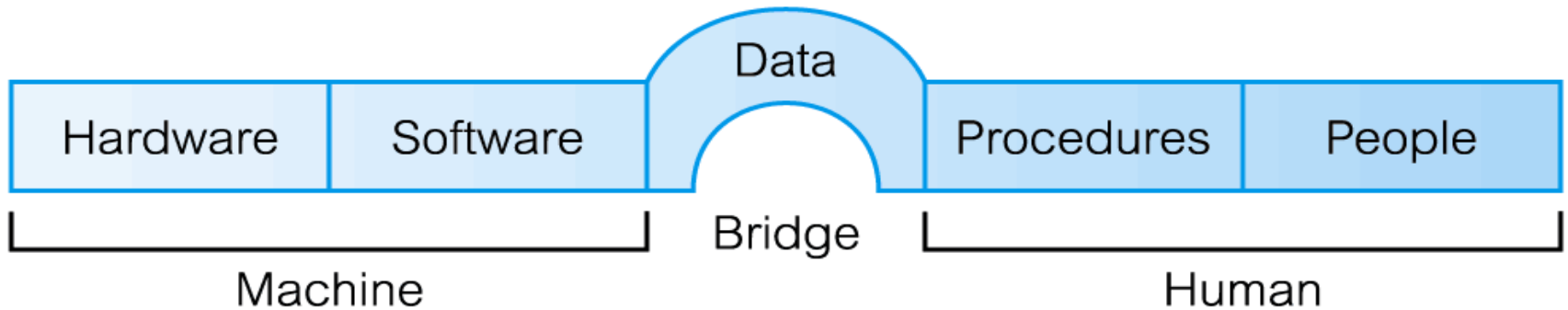
Views ...

- Allows each user to have his or her own view of the database ...
- A view is essentially some subset of the database ...

Views – Benefits ...

- Reduce complexity ...
- Provide a level of security ...
- Provide a mechanism to customize the appearance of the database ...
- Present a consistent, unchanging picture of the structure of the database, even if the underlying database is changed ...

Components of DBMS Environment



Components of DBMS Environment

- Hardware
 - Can range from a PC to a network of computers ...
- Software
 - DBMS, operating system, network software (if necessary) and also the application programs ...
- Data
 - Used by the organization and a description of this data called the schema ...

Components of DBMS Environment

- Procedures
 - Instructions and rules that should be applied to the design and use of the database and DBMS ...
- People

Roles in the Database Environment

- Data Administrator (DA) ...
- Database Administrator (DBA) ...
- Database Designers (Logical and Physical) ...
- Application Programmers ...
- End Users (naive and sophisticated) ...

Roles in the Database Environment

- **Data Administrator (DA)**

Is responsible for the management of the data resource including database planning, development and maintenance of standards, policies and procedures, and conceptual logical database design ...

- **Database Administrator (DBA)**

Is responsible for the physical realization of the database, including physical database design and implementation, security and integrity control, maintenance of the operational system, and ensuring satisfactory performance of the applications for users ...

- **Database Designers (Logical and Physical)**

Is concerned with identifying the data (that is, the entities and attributes), the relationships between the data, and the constraints on the data that is to be stored in the database.

- **Application Programmers**

Typically, the application developers work from a specification produced by systems analysts. Each program contains statements that request the DBMS to perform some operation on the database. This includes retrieving data, inserting, updating, and deleting data ...

- **End Users (naive and sophisticated)**

The end-users are the 'clients' for the database, which has been designed and implemented, and is being maintained to serve their information needs ...

History of Database Systems

- First-generation
 - Hierarchical and Network ...
- Second generation
 - Relational ...
- Third generation
 - Object-Relational ...
 - Object-Oriented ...

Advantages of DBMSs ...

- Control of data redundancy ...
- Data consistency ...
- More information from the same amount of data ...
- Sharing of data ...
- Improved data integrity ...
- Improved security ...
- Enforcement of standards ...
- Economy of scale ...

Advantages of DBMSs ...

- Balance conflicting requirements ...
- Improved data accessibility and responsiveness ...
- Increased productivity ...
- Improved maintenance through data independence ...
- Increased concurrency ...
- Improved backup and recovery services ...

Disadvantages of DBMSs ...

- Complexity ...
- Size ...
- Cost of DBMS ...
- Additional hardware costs ...
- Cost of conversion ...
- Performance ...
- Higher impact of a failure ...

Database Environment

Section 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 ...
- 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 ...

Introduction ...

- A major aim of a database system is:
 - To provide users with an abstract view of data ...
 - Hiding certain details of how data is stored and manipulated ...
- The starting point for the design of a database must be an abstract and general description of the information requirements of the organization that is to be represented in the database ...

Introduction ...

- The real world entities **Staff**, **PropertyforRent**, **PrivateOwner**, and **Client** ...
- Attributes describing properties or qualities of each entity (for example, Staff have a **name, position, and salary**) ...
- Relationships between these entities (for example, **Staff Manages PropertyForRent**).

- An early proposal for a standard terminology and general architecture for database systems was produced in 1971 by the DBTG (**Data Base Task Group**) appointed by the **Conference on Data Systems and Languages** (CODASYL, 1971).
- The DBTG recognized the need for a two-level approach with a system view called the **schema** and **user** views called **sub-schemas**.
- The **American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC)**, ANSI/X3/SPARC, produced a similar terminology and architecture in 1975 (ANSI, 1975). **ANSI-SPARC** recognized the need for a three-level approach with a system catalog.

- These proposals concentrated on the need for an implementation-independent layer to isolate programs from underlying representational issues.
- Although the ANSI-SPARC model did not become a standard, it still provides a basis for understanding some of the functionality of a DBMS.

Three-Level Architecture

- The levels form a three-level architecture comprising an ***external, a conceptual, and an internal*** level.
- The way users perceive the data is called the ***external level***.
- The way the DBMS and the operating system perceive the data is the ***internal level***, where the data is actually stored using the data structures and file organizations.
- The ***conceptual level*** provides both the mapping and the desired independence between the external and internal levels.

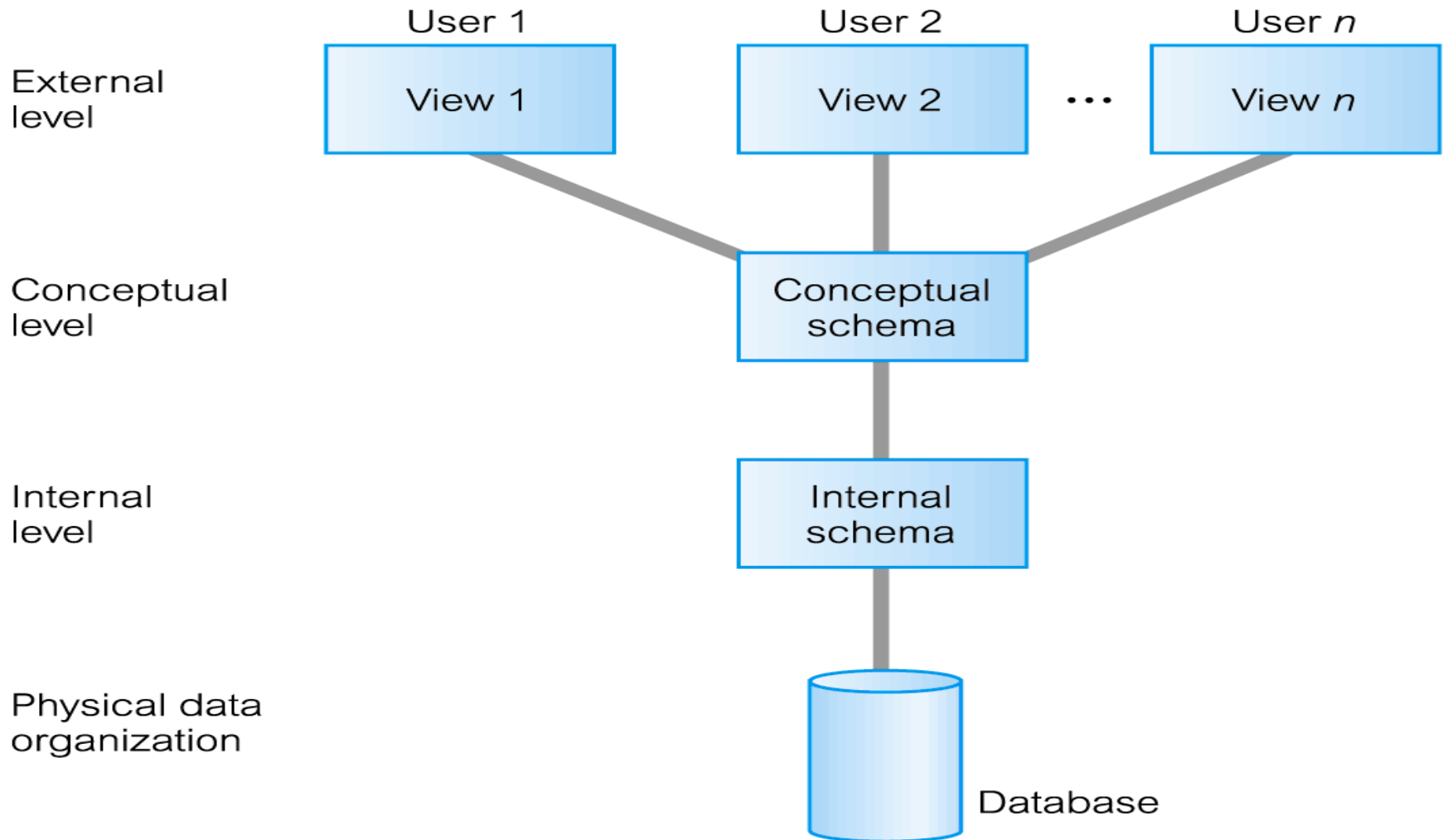
Main Objective

- The objective of the three-level architecture is to ***separate each user's view of the database from the way the database is physically represented.***
- There are several reasons why this separation is desirable:

- Each user should be able to access the same data, but have a different customized view of the data.
- Each user should be able to change the way he or she views the data, and this change should not affect other users.
- Users should not have to deal directly with physical database storage details, such as indexing or hashing. In other words, a user's interaction with the database should be independent of storage considerations.

- The Database Administrator (DBA) should be able to change the database storage structures without affecting the users' views.
- The internal structure of the database should be unaffected by changes to the physical aspects of storage, such as the changeover to a new storage device.
- The DBA should be able to change the conceptual structure of the database without affecting all users.

ANSI-SPARC Three-Level Architecture



ANSI-SPARC Three-Level Architecture

- **External Level**
 - Users' view of the database.
 - Describes that part of database that is relevant to a particular user.
- **Conceptual Level**
 - Community view of the database.
 - Describes what data is stored in database and relationships among the data.

ANSI-SPARC Three-Level Architecture

- **Internal Level**
 - Physical representation of the database on the computer.
 - Describes how the data is stored in the database.

Differences between Three Levels of ANSI-SPARC Architecture

External view 1

sNo	fName	lName	age	salary
-----	-------	-------	-----	--------

External view 2

staffNo	lName	branchNo
---------	-------	----------

Conceptual level

staffNo	fName	lName	DOB	salary	branchNo
---------	-------	-------	-----	--------	----------

Internal level

```
struct STAFF {  
    int staffNo;  
    int branchNo;  
    char fName [15];  
    char lName [15];  
    struct date dateOfBirth;  
    float salary;  
    struct STAFF *next;           /* pointer to next Staff record */  
};  
index staffNo; index branchNo; /* define indexes for staff */
```

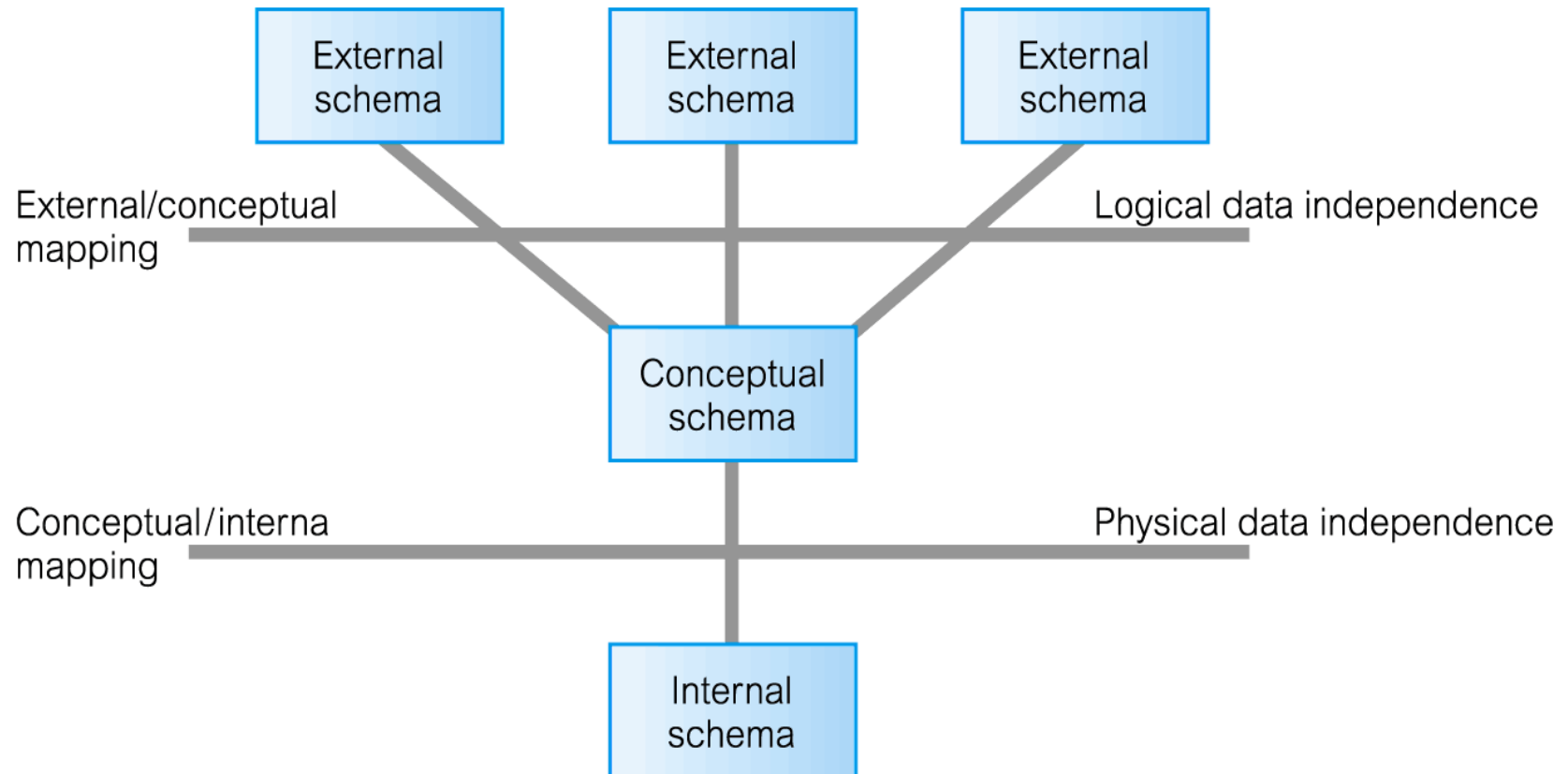
Data Independence ...

- **Logical Data Independence**
 - 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**
 - 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.

Data Independence and the ANSI-SPARC Three-Level Architecture



Database languages ...

- 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++, Java, or Visual Basic ...
- In this case, the high-level language is sometimes referred to as the host language ...

Database languages ...

- To compile the embedded file :
 - The commands in the data sublanguage are first removed from the host-language program and replaced by function calls ...
 - The pre-processed file is then compiled ...
 - placed in an object module ...
 - linked with a DBMS-specific library containing the replaced functions ...
 - Executed when required ...

Database Languages ...

- Data Definition Language (DDL)
 - Allows the DBA or user to describe and name entities, attributes, and relationships required for the application ...
 - Any associated integrity and security constraints ...

Database Languages ...

- Data Manipulation Language (DML)
 - Provides basic data manipulation operations on data held in the database ...
- Data manipulation operations usually include the following:
 - Insertion of new data into the database ...
 - Modification of data stored in the database ...
 - Retrieval of data contained in the database ...
 - Deletion of data from the database ...

Data Model ...

- Schema is written using a data definition language. DDL ...
- This type of language is too low level to describe the data requirements of an organization in a way that is readily understandable by a variety of users ...
- We require a higher-level description of the schema: that is, a **data model** ...

Data Model ...

Integrated collection of concepts for:

- Describing data ...
- Relationships between data ...
- Constraints on the data in an organization ...

Data Model ...

Data Model comprises:

- A structural part; consisting of a set of rules according to which databases can be constructed ...
- A manipulative part; defining the types of operation that are allowed on the data (updating or retrieving data from the database and changing the structure of the database) ...
- A set of integrity rules, which ensures that the data is accurate ...

Data Model ...

- Purpose
 - To represent data in an understandable way, then it can be easily used to design a database ...
- To reflect the ANSI-SPARC architecture, there must be three related data models:
 - **An external data model**, to represent each user's view of the organization ...
 - **A conceptual data model**, to represent the logical (or community) view that is DBMS independent ...
 - **An internal data model**, to represent the conceptual schema in such a way that it can be understood by the DBMS ...

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
 - Entity-Relationship ...
 - Semantic ...
 - Functional ...
 - Object-Oriented ...
- Record-Based Data Models
 - Relational Data Model ...
 - Network Data Model ...
 - Hierarchical Data Model ...
- Physical Data Models

Relational Data Model ...

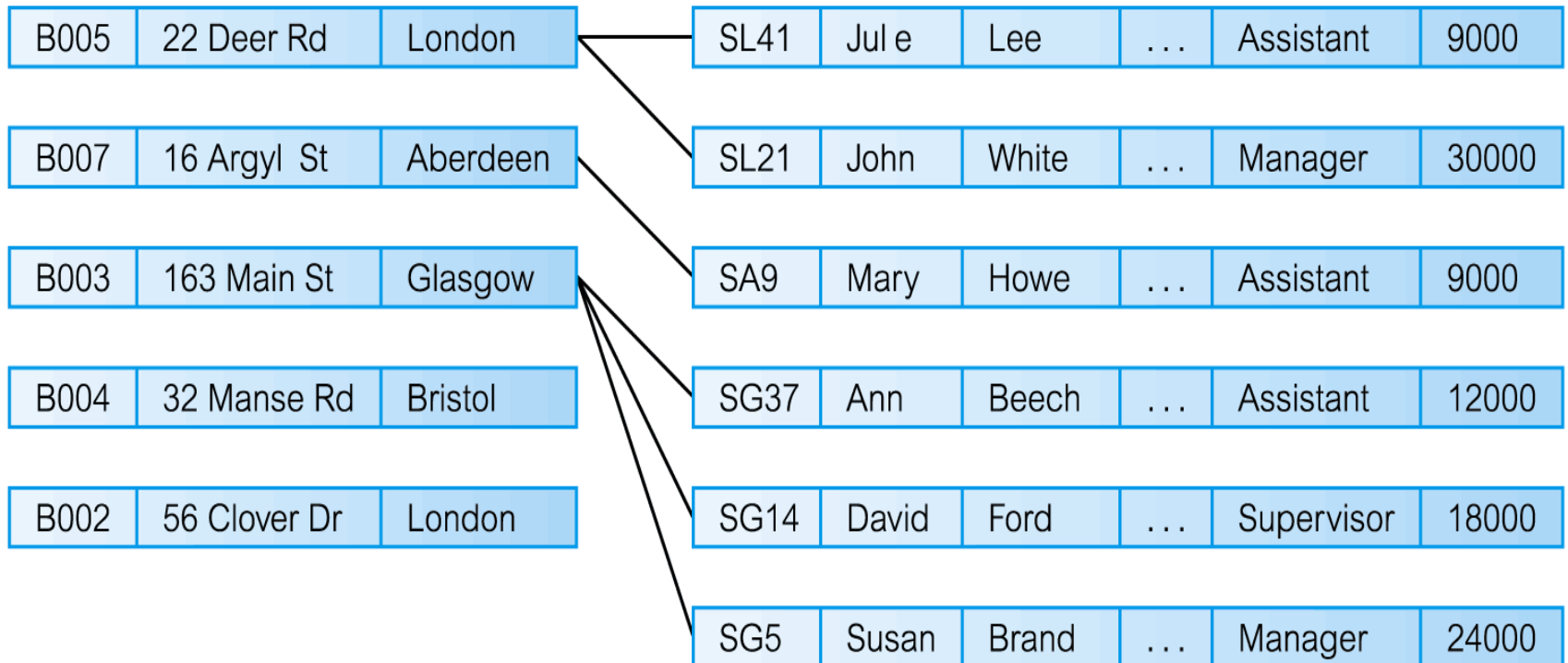
Branch

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

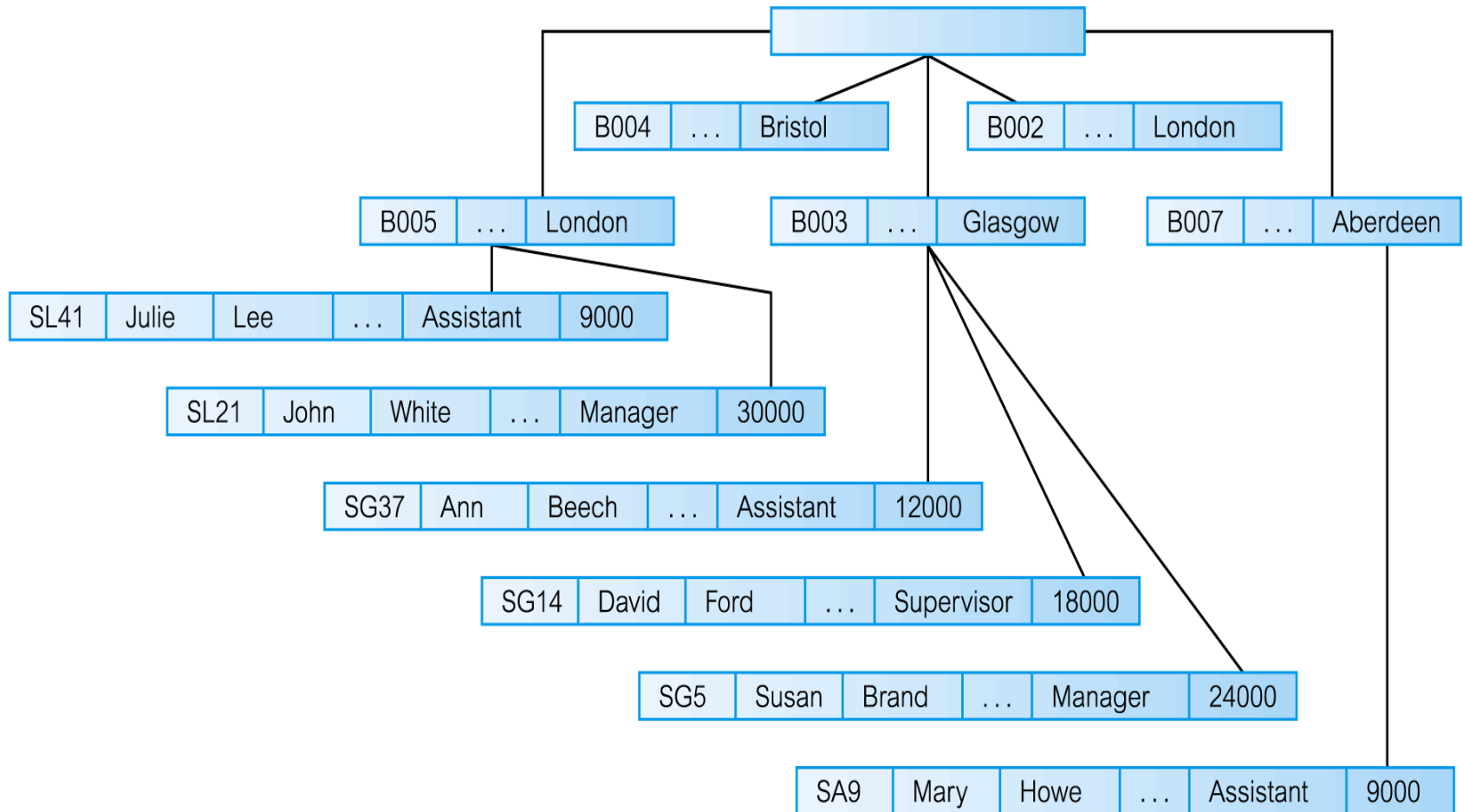
Staff

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 ...



Hierarchical Data Model ...



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 ...

Functions of a DBMS ...

- Data Storage, Retrieval, and Update ...
- A User-Accessible Catalog ...
- Transaction Support ...
- Concurrency Control Services ...
- Recovery Services ...
- Authorization Services ...
- Support for Data Communication ...
- Integrity Services ...
- Services to Promote Data Independence ...
- Utility Services ...

Functions of a DBMS: Data storage, retrieval, and update

- A DBMS must furnish users with the ability to store, retrieve, and update data in the database.
- This is the fundamental function of a DBMS. Clearly in providing this functionality the DBMS should hide the internal physical implementation details (such as file organization and storage structures) from the user ...

Functions of a DBMS: A user-accessible catalog...

- A DBMS must furnish a catalog in which descriptions of data items are stored and which is accessible to users ...
- A key feature of the ANSI-SPARC architecture is the recognition of an integrated system catalog to hold data about the schemas, users, applications, and so on ...
- The catalog is expected to be accessible to users as well as to the DBMS...

Functions of a DBMS: A user-accessible catalog...

- A system catalog, or data dictionary, is a repository of information describing the data in the database: it is, the 'data about the data' or metadata ...
- The amount of information and the way the information is used vary with the DBMS ...

Functions of a DBMS: A user-accessible catalog...

- The system catalog stores:
 - Names, types, and sizes of data items ...
 - Names of relationships ...
 - Integrity constraints on the data ...
 - Names of authorized users who have access to the data...
 - The data items that each user can access and the types of access allowed; for example, insert, update, delete, or read access ...
 - External, conceptual, and internal schemas and the mappings between the schemas ...
 - Usage statistics, such as the frequencies of transactions and counts on the number of accesses made to objects in the database ...

Functions of a DBMS: A user-accessible catalog...

- Benefits of a system catalog are:
 - Information about data can be collected and stored centrally. This helps to maintain control over the data as a resource ...
 - The meaning of data can be defined, which will help other users understand the purpose of the data ...
 - Communication is simplified, since exact meanings are stored. The system catalog may also identify the user or users who own or access the data ...
 - Redundancy and inconsistencies can be identified more easily since the data is centralized ...
 - Changes to the database can be recorded ...
 - The impact of a change can be determined before it is implemented, since the system catalog records each data item, all its relationships, and all its users ...
 - Security can be enforced ...
 - Integrity can be ensured ...
 - Audit information can be provided ...

Functions of a DBMS: A user-accessible catalog...

- Some times there is a distinction between system catalog and data directory, where a data directory holds information relating to where data is stored and how it is stored ...
- The International Organization for Standardization (ISO) has adopted a standard for data dictionaries called Information Resource Dictionary System (IRDS) (ISO, 1990, 1993) ...
- IRDS is a software tool that can be used to control and document an organization's information sources ...
- It provides a definition for the tables that comprise the data dictionary and the operations that can be used to access these tables ...

Functions of a DBMS: Transaction support...

- A DBMS must furnish a mechanism which will ensure either that all the updates corresponding to a given transaction are made or that none of them is made ...
- A transaction is a series of actions, carried out by a single user or application program, which accesses or changes the contents of the database ...

Functions of a DBMS: Concurrency control services...

- A DBMS must furnish a mechanism to ensure that the database is updated correctly when multiple users are updating the database concurrently ...
- One major objective in using a DBMS is to enable many users to access shared data concurrently ...
- Concurrent access is relatively easy if all users are only reading data, as there is no way that they can interfere with one another ...
- When two or more users are accessing the database simultaneously and at least one of them is updating data, there may be interference that can result in inconsistencies...

Functions of a DBMS: Recovery services...

- A DBMS must furnish a mechanism for recovering the database in the event that the database is damaged in any way ...
- If the transaction fails then the database has to be returned to a consistent state ...
- Fails of transaction may be a result of:
 - System crash ...
 - Media failure ...
 - A hardware or software error causing the DBMS to stop ...
 - The user detecting an error during the transaction and aborting the transaction before it completes ...
- In all these cases, the DBMS must provide a mechanism to recover the database to a consistent state ...

Functions of a DBMS: Authorization services...

- A DBMS must furnish a mechanism to ensure that only authorized users can access the database ...
- The term security refers to the protection of the database against unauthorized access, either intentional or accidental ...
- The DBMS provides mechanisms to ensure the data is secure ...

Functions of a DBMS: Support for data communication...

- Most users access the database from workstations. Sometimes these workstations are connected directly to the computer hosting the DBMS. In other cases, the workstations are at remote locations and communicate with the computer hosting the DBMS over a network ...
- In either case, the DBMS receives requests as communications messages and responds in a similar way ...
- All such transmissions are handled by a Data Communication Manager (DCM) ...
- The DCM is not part of the DBMS, it is necessary for the DBMS to be capable of being integrated with a variety of DCMs if the system is to be commercially viable ...
- Even DBMSs for personal computers should be capable of being run on a local area network so that one centralized database can be established for users to share, rather than having a series of disparate databases, one for each user ...

Functions of a DBMS: Integrity services...

- A DBMS must furnish a means to ensure that both the data in the database and changes to the data follow certain rules...
- Database integrity refers to the correctness and consistency of stored data ...
- While integrity is related to security, it has wider implications:
 - Integrity is concerned with the quality of data itself ...
 - Integrity is usually expressed in terms of constraints, which are consistency rules that the database is not permitted to violate ...

Functions of a DBMS: Services to promote data independence...

- A DBMS must include facilities to support the independence of programs from the actual structure of the database ...
- Data independence is normally achieved through a view or subschema mechanism ...
- Physical data independence is easier to achieve: there are usually several types of change that can be made to the physical characteristics of the database without affecting the views ...
- Complete logical data independence is more difficult to achieve...
- The addition of a new entity, attribute, or relationship can usually be accommodated, but not their removal ...
- In some systems, any type of change to an existing component in the logical structure is prohibited ...

Functions of a DBMS: Utility services...

- Utility programs help the DBA to administer the database effectively.
 - Some utilities work at the external level, and consequently can be produced by the DBA ...
 - Other utilities work at the internal level and can be provided only by the DBMS vendor ...

Functions of a DBMS: Utility services...

- Examples of utilities are:
 - Import facilities, to load the database from flat files, and export facilities, to unload the database to flat files ...
 - Monitoring facilities, to monitor database usage and operation ...
 - Statistical analysis programs, to examine performance or usage statistics ...
 - Index reorganization facilities, to reorganize indexes and their overflows ...
 - Garbage collection and reallocation, to remove deleted records physically from the storage devices, to consolidate the space released, and to reallocate it where it is needed ...

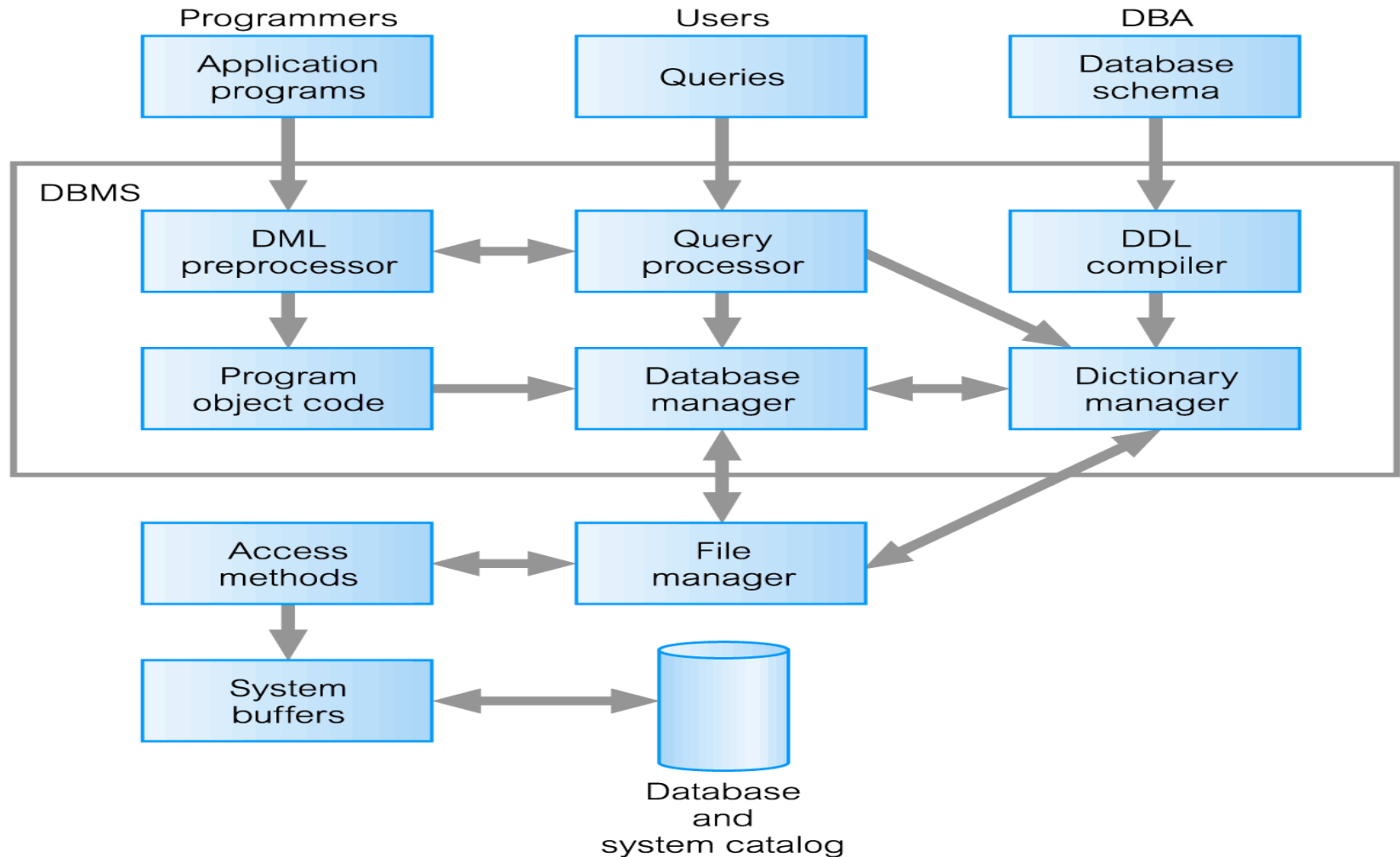
Components of a DBMS ...

- DBMSs are highly complex and sophisticated pieces of software that aim to provide the services discussed ...
- It is not possible to generalize the component structure of a DBMS as it varies greatly from system to system ...
- It is useful for understanding database systems to view the components and the relationships between them ...

Components of a DBMS ...

- A DBMS is partitioned into several software components (or *modules*), each of which is assigned a specific operation. Some of the functions of the DBMS are supported by the underlying operating system ...
- The operating system provides only basic services and the DBMS must be built on top of it ...
- The design of a DBMS must take into account the interface between the DBMS and the operating system ...

Components of a DBMS ...



- **Query processor** This is a major DBMS component that transforms queries into a series of low-level instructions directed to the database manager ...
- **Database manager (DM)**
 - Interfaces with user-submitted application programs and queries ...
 - Accepts queries and examines the external and conceptual schemas to determine what conceptual records are required to satisfy the request ...
 - Places a call to the file manager to perform the request ...

- **File manager**

- Manipulates the underlying storage files and manages the allocation of storage space on disk ...
- Establishes and maintains the list of structures and indexes defined in the internal schema ...
- Passes the requests on to the appropriate access methods, which either *read data from* or *write data into* the system buffer (or *cache*) ...

- **DML preprocessor**

- Converts DML statements embedded in an application program into standard function calls in the host language ...
- Must interact with the query processor to generate the appropriate code ...

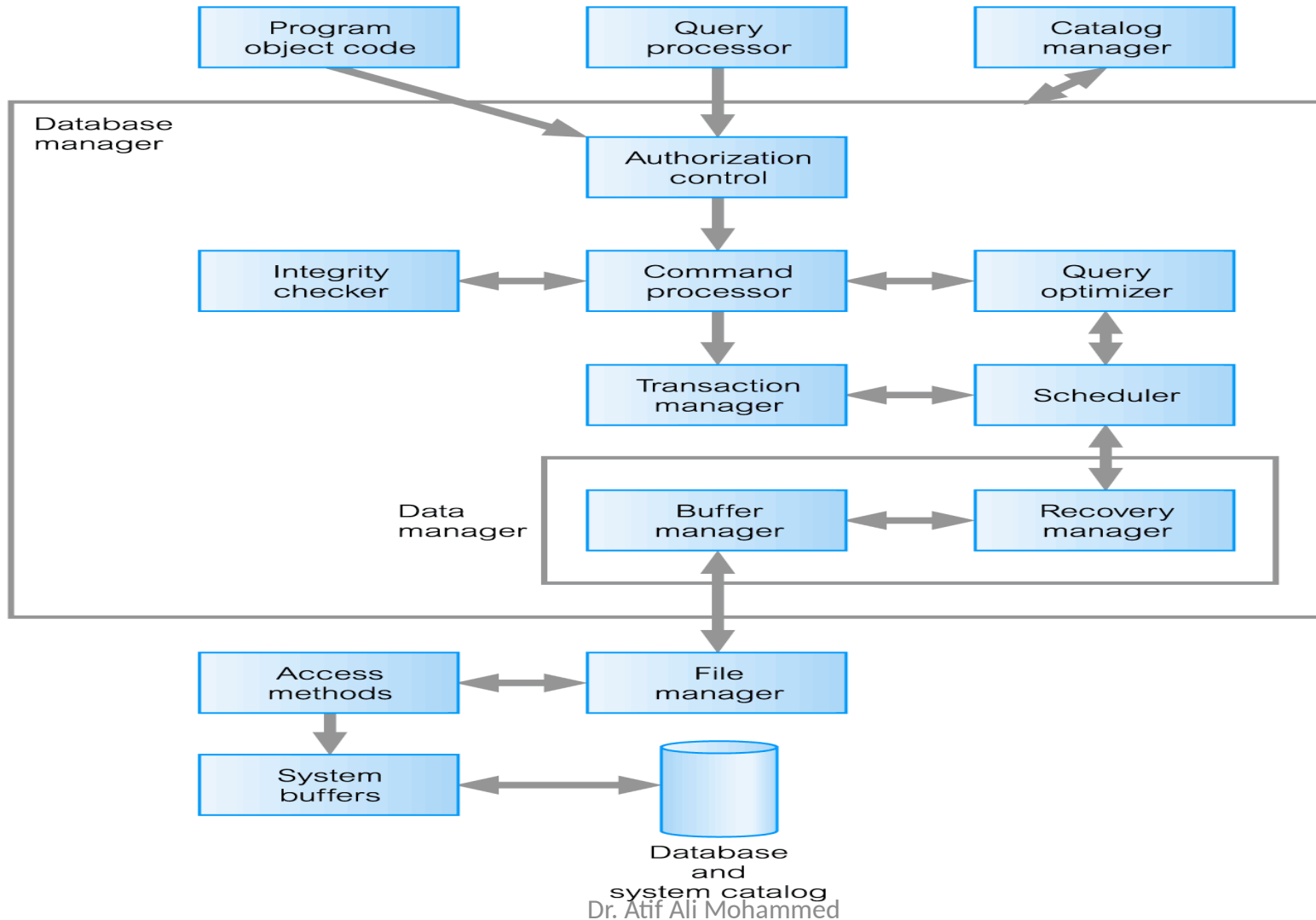
- **DDL compiler**

- Converts DDL statements into a set of tables containing metadata ...
- These tables are then stored in the system catalog while control information is stored in data file headers ...

- **Catalog manager**

- Manages access to and maintains the system catalog ...
- The system catalog is accessed by most DBMS components ...

Components of Database Manager (DM)



- **Authorization control** This module checks that the user has the necessary authorization to carry out the required operation ...
- **Command processor** Once the system has checked that the user has authority to carry out the operation, control is passed to the command processor ...
- **Integrity checker** For an operation that changes the database, the integrity checker checks that the requested operation satisfies all necessary integrity constraints (such as key constraints) ...
- **Query optimizer** This module determines an optimal strategy for the query execution ...

- **Transaction manager** This module performs the required processing of operations it receives from transactions ...
- **Scheduler**
 - Responsible for ensuring that concurrent operations on the database proceed without conflicting with one another ...
 - Controls the relative order in which transaction operations are executed ...
- **Recovery manager**
 - Ensures that the database remains in a consistent state in the presence of failures ...
 - Responsible for transaction commit and abort ...

- **Buffer manager**

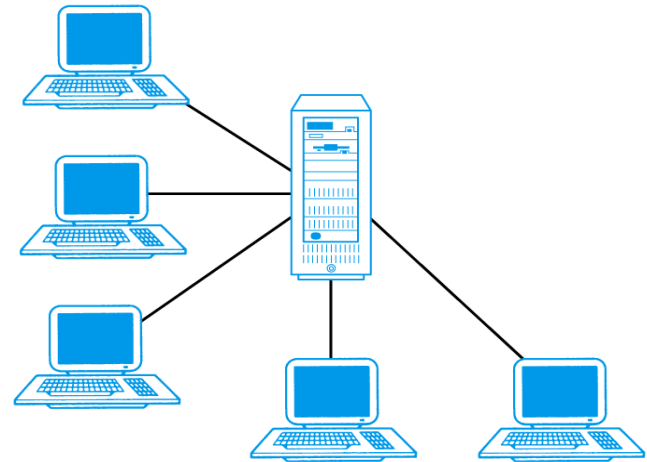
- Responsible for the transfer of data between main memory and secondary storage, such as disk and tape ...
- The recovery manager and the buffer manager are sometimes referred to collectively as the data manager ...
- The buffer manager is sometimes known as the cache manager ...

Multi-User DBMS Architectures

- The common architectures that are used to implement multi-user database management systems, are:
- Teleprocessing ...
- File-server ...
- Client-server ...

Teleprocessing ...

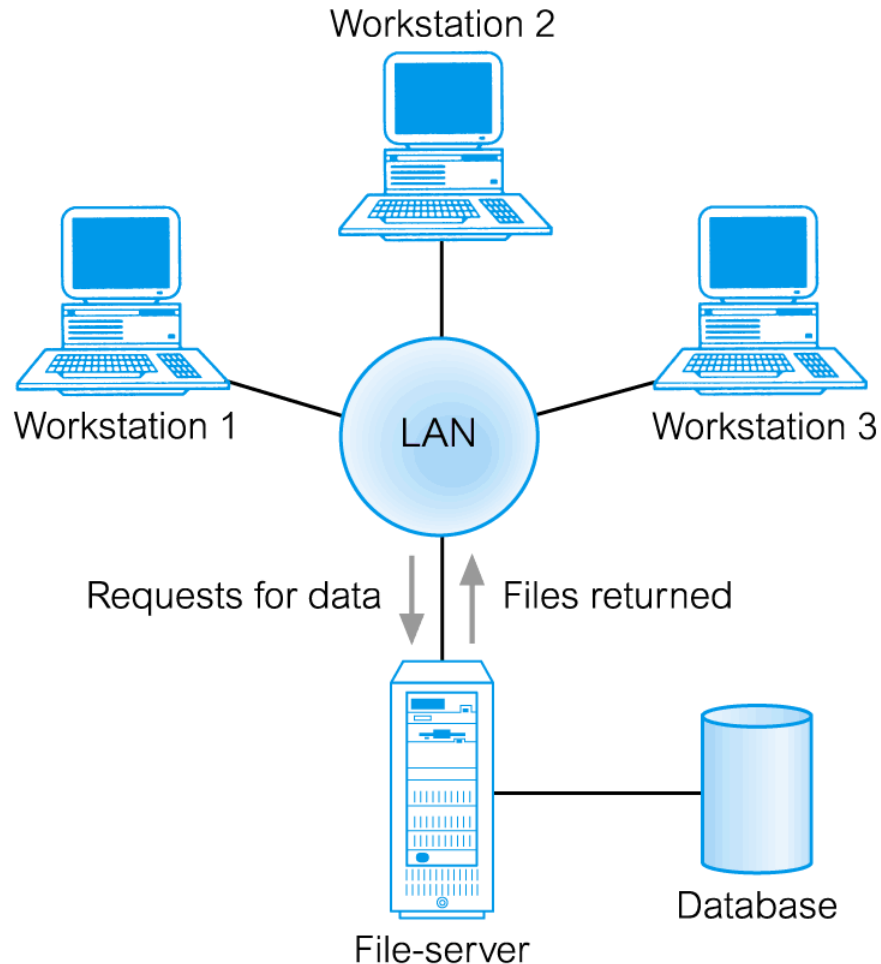
- Traditional architecture.
- Single mainframe with a number of terminals attached.
- Trend is now towards downsizing.



File-Server ...

- File-server is connected to several workstations across a network.
- Database resides on file-server.
- DBMS and applications run on each workstation.
- Disadvantages include:
 - Significant network traffic.
 - Copy of DBMS on each workstation.
 - Concurrency, recovery and integrity control more complex.

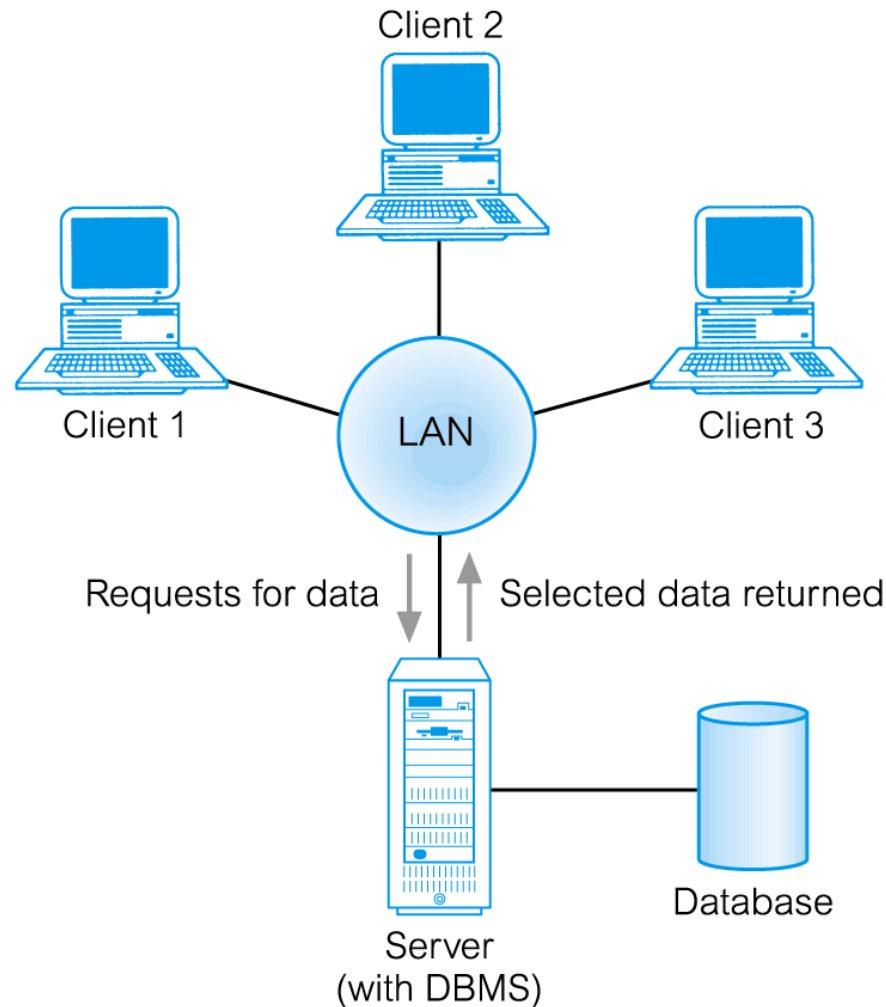
File-Server Architecture



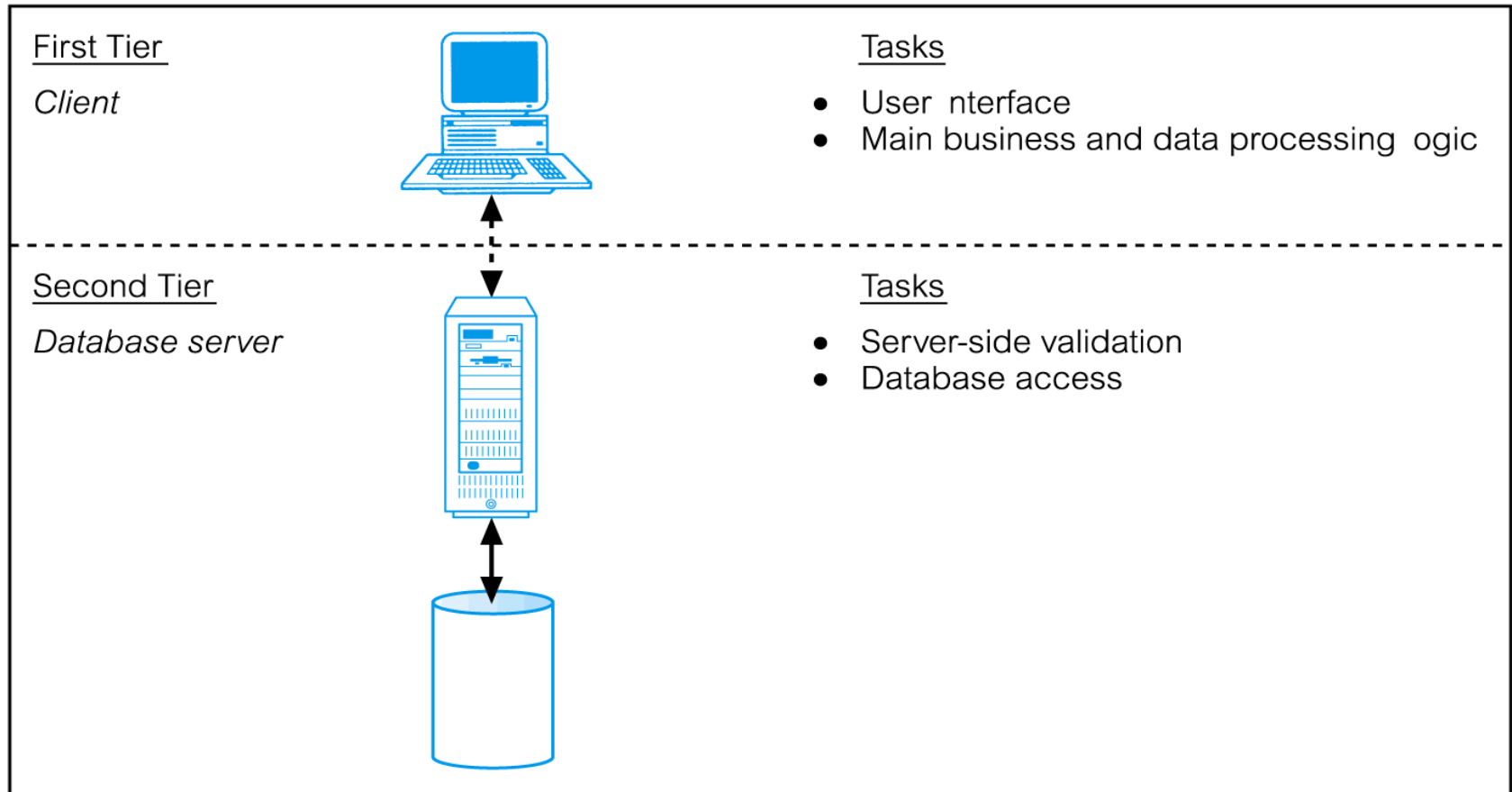
Traditional Two-Tier Client-Server

- Client (tier 1) manages user interface and runs applications.
- Server (tier 2) holds database and DBMS.
- Advantages include:
 - wider access to existing databases;
 - increased performance;
 - possible reduction in hardware costs;
 - reduction in communication costs;
 - increased consistency.

Traditional Two-Tier Client-Server



Traditional Two-Tier Client-Server



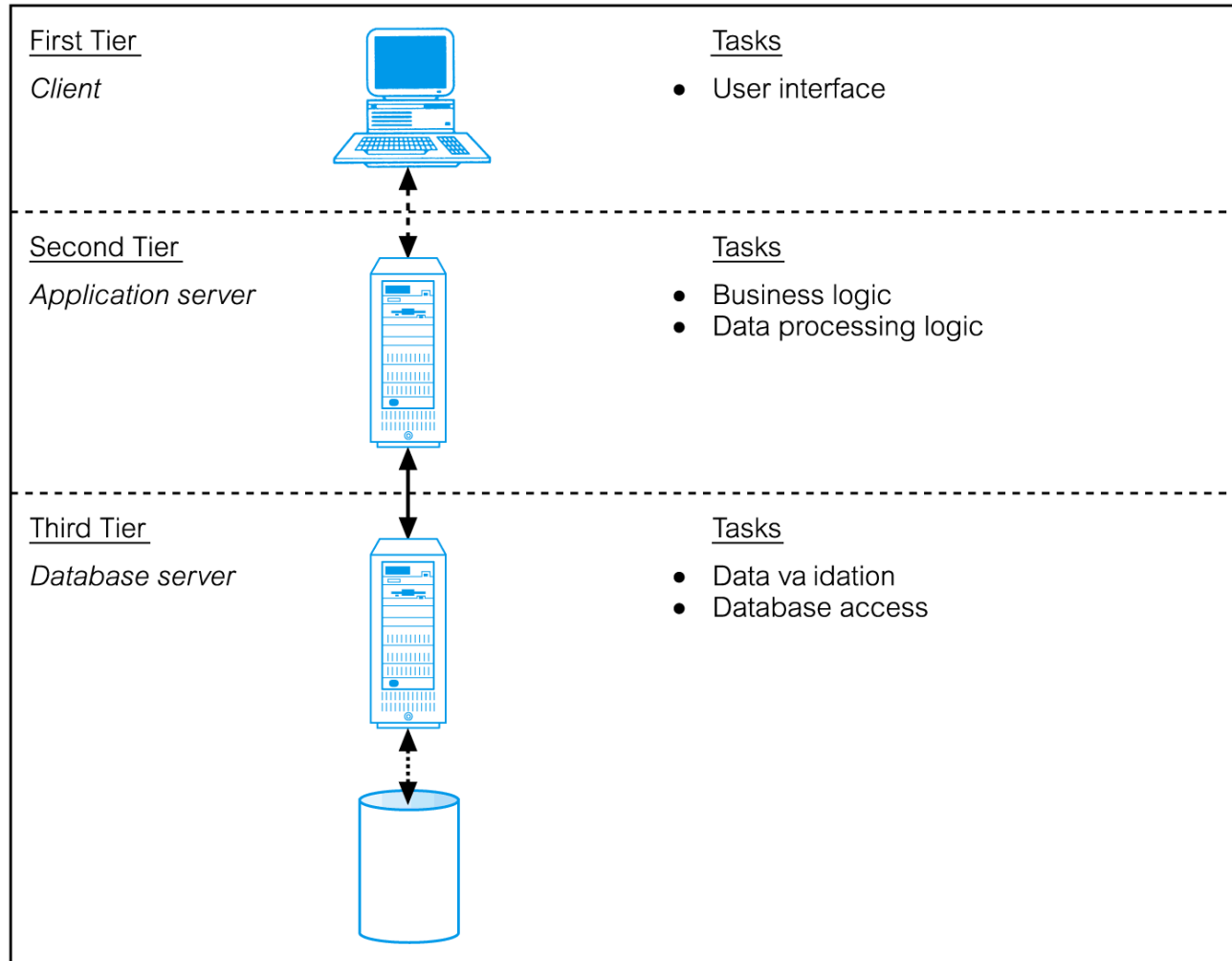
Three-Tier Client-Server

- **Client side presented two problems preventing true scalability:**
 - ‘Fat’ client, requiring considerable resources on client’s computer to run effectively.
 - Significant client side administration overhead.
- **By 1995, three layers proposed, each potentially running on a different platform.**

Three-Tier Client-Server

- **Advantages:**
 - ‘Thin’ client, requiring less expensive hardware.
 - Application maintenance centralized.
 - Easier to modify or replace one tier without affecting others.
 - Separating business logic from database functions makes it easier to implement load balancing.
 - Maps quite naturally to Web environment.

Three-Tier Client-Server



Transaction Processing Monitors

- Program that controls data transfer between clients and servers in order to provide a consistent environment, particularly for Online Transaction Processing (OLTP).

SQL: Data Manipulation

Section Objectives ...

- Purpose and importance of SQL.
- How to retrieve data from database using SELECT and:
 - Use compound WHERE conditions.
 - Sort query results using ORDER BY.
 - Use aggregate functions.
 - Group data using GROUP BY and HAVING.
 - Use subqueries.

Chapter 5 - Objectives

- Join tables together.
- Perform set operations (**UNION, INTERSECT, EXCEPT**).
- How to update database using **INSERT, UPDATE, and DELETE**.

Objectives of SQL

- **Ideally, database language should allow user to:**
 - create the database and relation structures;
 - perform insertion, modification, deletion of data from relations;
 - perform simple and complex queries.
- **Must perform these tasks with minimal user effort and command structure/syntax must be easy to learn.**
- **It must be portable.**

Objectives of SQL

- **SQL is a transform-oriented language with 2 major components:**
 - A DDL for defining database structure.
 - A DML for retrieving and updating data.
- **Until SQL:1999, SQL did not contain flow of control commands. These had to be implemented using a programming or job-control language, or interactively by the decisions of user.**

Objectives of SQL

- **SQL is relatively easy to learn:**
 - it is non-procedural - you specify *what* information you require, rather than *how* to get it;
 - it is essentially free-format.

Objectives of SQL

- **Consists of standard English words:**

```
1) CREATE TABLE Staff(staffNo VARCHAR(5),  
                        IName VARCHAR(15),  
                        salary DECIMAL(7,2));  
2) INSERT INTO Staff VALUES ('SG16', 'Brown', 8300);  
3) SELECT staffNo, IName, salary  
   FROM Staff  
   WHERE salary > 10000;
```

Objectives of SQL

- Can be used by range of users including DBAs, management, application developers, and other types of end users.
- An ISO standard now exists for SQL, making it both the formal and *de facto* standard language for relational databases.

History of SQL

- In 1974, D. Chamberlin (IBM San Jose Laboratory) defined language called 'Structured English Query Language' (SEQUEL).
- A revised version, SEQUEL/2, was defined in 1976 but name was subsequently changed to SQL for legal reasons.

History of SQL

- Still pronounced 'see-quel', though official pronunciation is 'S-Q-L'.
- IBM subsequently produced a prototype DBMS called *System R*, based on SEQUEL/2.
- Roots of SQL, however, are in SQUARE (Specifying Queries as Relational Expressions), which predates System R project.

History of SQL

- In late 70s, ORACLE appeared and was probably first commercial RDBMS based on SQL.
- In 1987, ANSI and ISO published an initial standard for SQL.
- In 1989, ISO published an addendum that defined an 'Integrity Enhancement Feature'.
- In 1992, first major revision to ISO standard occurred, referred to as SQL2 or SQL/92.
- In 1999, SQL:1999 was released with support for object-oriented data management.
- In late 2003, SQL:2003 was released.

Importance of SQL

- **SQL has become part of application architectures such as IBM's Systems Application Architecture.**
- **It is strategic choice of many large and influential organizations (e.g. X/OPEN).**
- **SQL is Federal Information Processing Standard (FIPS) to which conformance is required for all sales of databases to American Government.**

Importance of SQL

- **SQL is used in other standards and even influences development of other standards as a definitional tool. Examples include:**
 - **ISO's Information Resource Directory System (IRDS) Standard**
 - **Remote Data Access (RDA) Standard.**

Writing SQL Commands

- SQL statement consists of *reserved words* and *user-defined words*.
 - Reserved words are a fixed part of SQL and must be spelt exactly as required and cannot be split across lines.
 - User-defined words are made up by user and represent names of various database objects such as relations, columns, views.

Writing SQL Commands

- Most components of an SQL statement are *case insensitive*, except for literal character data.
- More readable with indentation and lineation:
 - Each clause should begin on a new line.
 - Start of a clause should line up with start of other clauses.
 - If clause has several parts, should each appear on a separate line and be indented under start of clause.

Writing SQL Commands

- **Use extended form of BNF notation:**
 - Upper-case letters represent reserved words.
 - Lower-case letters represent user-defined words.
 - | indicates a *choice* among alternatives.
 - Curly braces indicate a *required element*.
 - Square brackets indicate an *optional element*.
 - ... indicates *optional repetition* (0 or more).

Literals

- **Literals are constants used in SQL statements.**
- **All non-numeric literals must be enclosed in single quotes (e.g. 'London').**
- **All numeric literals must not be enclosed in quotes (e.g. 650.00).**

SELECT Statement

```
SELECT [DISTINCT | ALL]  
    { * | [columnExpression [AS newName]] [,...] }  
FROM      TableName [alias] [, ...]  
[WHERE    condition]  
[GROUP BY columnList] [HAVING condition]  
[ORDER BY columnList]
```

SELECT Statement

FROM Specifies table(s) to be used.

WHERE Filters rows.

GROUP BY Forms groups of rows with same column value.

HAVING Filters groups subject to some condition.

SELECT Specifies which columns are to appear in output.

ORDER BY Specifies the order of the output.

SELECT Statement

- **Order of the clauses cannot be changed.**
- **Only SELECT and FROM are mandatory.**

Example 5.1 All Columns, All Rows

List full details of all staff.

```
SELECT staffNo, fName, IName, address,  
       position, sex, DOB, salary, branchNo  
FROM Staff;
```

- **Can use * as an abbreviation for ‘all columns’:**

```
SELECT *  
FROM Staff;
```

Example 5.1 All Columns, All Rows

Table 5.1 Result table for Example 5.1.

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

Example 5.2 Specific Columns, All Rows

Produce a list of salaries for all staff, showing only staff number, first and last names, and salary.

```
SELECT staffNo, fName, lName, salary  
FROM Staff;
```

Example 5.2 Specific Columns, All Rows

Table 5.2 Result table for Example 5.2.

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG37	Ann	Beech	12000.00
SG14	David	Ford	18000.00
SA9	Mary	Howe	9000.00
SG5	Susan	Brand	24000.00
SL41	Julie	Lee	9000.00

Example 5.3 Use of DISTINCT

List the property numbers of all properties that have been viewed.

```
SELECT propertyNo  
FROM Viewing;
```

propertyNo
PA14
PG4
PG4
PA14
PG36

Example 5.3 Use of DISTINCT

- Use **DISTINCT** to eliminate duplicates:

SELECT DISTINCT propertyNo

**FROM **

propertyNo
PA14
PG4
PG36

Example 5.4 Calculated Fields

Produce list of monthly salaries for all staff, showing staff number, first/last name, and salary.

SE Table 5.4 Result table for Example 5.4.

, salary/12

staffNo	fName	lName	col4
SL21	John	White	2500.00
SG37	Ann	Beech	1000.00
SG14	David	Ford	1500.00
SA9	Mary	Howe	750.00
SG5	Susan	Brand	2000.00
SL41	Julie	Lee	750.00

Example 5.4 Calculated Fields

- To name column, use AS clause:

```
SELECT staffNo, fName, lName, salary/12  
      AS monthlySalary  
FROM Staff;
```

Example 5.5 Comparison Search Condition

List all staff with a salary greater than 10,000.

```
SELECT  staffNo, fName, lName, position,  
salary  
FROM Staff
```

√ **Table 5.5** Result table for Example 5.5.

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG37	Ann	Beech	Assistant	12000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

Example 5.6 Compound Comparison Search Condition

List addresses of all branch offices in London or Glasgow.

SELECT *

FROM Branch

Table 5.6 Result table for Example 5.6.

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B003	163 Main St	Glasgow	G11 9QX
B002	56 Clover Dr	London	NW10 6EU

Example 5.7 Range Search Condition

List all staff with a salary between 20,000 and 30,000.

```
SELECT staffNo, fName, lName, position, salary  
FROM Staff  
WHERE salary BETWEEN 20000 AND 30000;
```

- **BETWEEN test includes the endpoints of range.**

Example 5.7 Range Search Condition

Table 5.7 Result table for Example 5.7.

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG5	Susan	Brand	Manager	24000.00

Example 5.7 Range Search Condition

- Also a negated version NOT BETWEEN.
- BETWEEN does not add much to SQL's expressive power. Could also write:

```
SELECT staffNo, fName, lName, position, salary  
FROM Staff  
WHERE salary >= 20000 AND salary <= 30000;
```

- Useful, though, for a range of values.

Example 5.8 Set Membership

List all managers and supervisors.

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE position IN ('Manager', 'Supervisor');
```

Table 5.8 Result table for Example 5.8.

staffNo	fName	lName	position
SL21	John	White	Manager
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

Example 5.8 Set Membership

- There is a negated version (NOT IN).
- IN does not add much to SQL's expressive power. Could have expressed this as:

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE position='Manager' OR
       position='Supervisor';
```

- IN is more efficient when set contains many values.

Example 5.9 Pattern Matching

Find all owners with the string 'Glasgow' in their address.

SELECT ownerNo, fName, lName, address, telNo

Table 5.9 Result table for Example 5.9.

ownerNo	fName	lName	address	telNo
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

Example 5.9 Pattern Matching

- SQL has two special pattern matching symbols:
 - %: sequence of zero or more characters;
 - _ (underscore): any single character.
- LIKE '%Glasgow%' means a sequence of characters of any length containing '*Glasgow*'.

Example 5.10 NULL Search Condition

List details of all viewings on property PG4 where a comment has not been supplied.

- There are 2 viewings for property PG4, one with and one without a comment.
- Have to test for null explicitly using special keyword IS NULL:

```
SELECT clientNo, viewDate
FROM Viewing
WHERE propertyNo = 'PG4' AND
      comment IS NULL;
```

Example 5.10 NULL Search Condition

clientNo	viewDate
CR56	26-May-04

- **Negated version (IS NOT NULL) can test for non-null values.**

Example 5.11 Single Column Ordering

List salaries for all staff, arranged in descending order of salary.

```
SELECT staffNo, fName, lName, salary  
FROM Staff  
ORDER BY salary DESC;
```

Example 5.11 Single Column Ordering

Table 5.11 Result table for Example 5.11.

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG5	Susan	Brand	24000.00
SG14	David	Ford	18000.00
SG37	Ann	Beech	12000.00
SA9	Mary	Howe	9000.00
SL41	Julie	Lee	9000.00

Example 5.12 Multiple Column Ordering

Produce abbreviated list of properties in order of property type.

```
SELECT propertyNo, type, rooms, rent  
FROM PropertyForRent  
ORDER BY type;
```

Example 5.12 Multiple Column Ordering

Table 5.12(a) Result table for Example 5.12 with one sort key.

propertyNo	type	rooms	rent
PL94	Flat	4	400
PG4	Flat	3	350
PG36	Flat	3	375
PG16	Flat	4	450
PA14	House	6	650
PG21	House	5	600

Example 5.12 Multiple Column Ordering

- Four flats in this list - as no minor sort key specified, system arranges these rows in any order it chooses.
- To arrange in order of rent, specify minor order:

```
SELECT propertyNo, type, rooms, rent  
FROM PropertyForRent  
ORDER BY type, rent DESC;
```

Example 5.12 Multiple Column Ordering

Table 5.12(b) Result table for Example 5.12 with two sort keys.

propertyNo	type	rooms	rent
PG16	Flat	4	450
PL94	Flat	4	400
PG36	Flat	3	375
PG4	Flat	3	350
PA14	House	6	650
PG21	House	5	600

SELECT Statement - Aggregates

- **ISO standard defines five aggregate functions:**

COUNT returns number of values in specified column.

SUM returns sum of values in specified column.

AVG returns average of values in specified column.

MIN returns smallest value in specified column.

MAX returns largest value in specified column.

SELECT Statement - Aggregates

- **Each operates on a single column of a table and returns a single value.**
- **COUNT, MIN, and MAX apply to numeric and non-numeric fields, but SUM and AVG may be used on numeric fields only.**
- **Apart from COUNT(*), each function eliminates nulls first and operates only on remaining non-null values.**

SELECT Statement - Aggregates

- **COUNT(*)** counts all rows of a table, regardless of whether nulls or duplicate values occur.
- Can use **DISTINCT** before column name to eliminate duplicates.
- **DISTINCT** has no effect with **MIN/MAX**, but may have with **SUM/AVG**.

SELECT Statement - Aggregates

- **Aggregate functions can be used only in SELECT list and in HAVING clause.**
- **If SELECT list includes an aggregate function and there is no GROUP BY clause, SELECT list cannot reference a column out with an aggregate function. For example, the following is illegal:**

```
SELECT staffNo, COUNT(salary)  
FROM Staff;
```

Example 5.13 Use of COUNT(*)

How many properties cost more than £350 per month to rent?

```
SELECT COUNT(*) AS myCount  
FROM PropertyForRent  
WHERE rent > 350;
```

myCount
5

Example 5.14 Use of COUNT(DISTINCT)

How many different properties viewed in May '04?

```
SELECT COUNT(DISTINCT propertyNo) AS myCount  
FROM Viewing  
WHERE viewDate BETWEEN '1-May-04'  
AND '31-May-04';
```

myCount
2

Example 5.15 Use of COUNT and SUM

Find number of Managers and sum of their salaries.

```
SELECT COUNT(staffNo) AS myCount,  
        SUM(salary) AS mySum  
FROM Staff  
WHERE position = 'Manager'.
```

myCount	mySum
2	54000.00

Example 5.16 Use of MIN, MAX, AVG

Find minimum, maximum, and average staff salary.

```
SELECT MIN(salary) AS myMin,  
       MAX(salary) AS myMax,  
       AVG(salary) AS myAvg  
FROM Staff;
```

myMin	myMax	myAvg
9000.00	30000.00	17000.00

SELECT Statement - Grouping

- Use GROUP BY clause to get sub-totals.
- SELECT and GROUP BY closely integrated: each item in SELECT list must be *single-valued per group*, and SELECT clause may only contain:
 - column names
 - aggregate functions
 - constants
 - expression involving combinations of the above.

SELECT Statement - Grouping

- **All column names in SELECT list must appear in GROUP BY clause unless name is used only in an aggregate function.**
- **If WHERE is used with GROUP BY, WHERE is applied first, then groups are formed from remaining rows satisfying predicate.**
- **ISO considers two nulls to be equal for purposes of GROUP BY.**

Example 5.17 Use of GROUP BY

Find number of staff in each branch and their total salaries.

```
SELECT  branchNo,  
        COUNT(staffNo) AS myCount,  
        SUM(salary) AS mySum  
FROM Staff  
GROUP BY branchNo  
ORDER BY branchNo;
```

Example 5.17 Use of GROUP BY

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00
B007	1	9000.00

Restricted Groupings – HAVING clause

- **HAVING clause is designed for use with GROUP BY to restrict groups that appear in final result table.**
- **Similar to WHERE, but WHERE filters individual rows whereas HAVING filters groups.**
- **Column names in HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function.**

Example 5.18 Use of HAVING

For each branch with more than 1 member of staff, find number of staff in each branch and sum of their salaries.

```
SELECT branchNo,  
       COUNT(staffNo) AS myCount,  
       SUM(salary) AS mySum  
FROM Staff  
GROUP BY branchNo  
HAVING COUNT(staffNo) > 1  
ORDER BY branchNo;
```

Example 5.18 Use of HAVING

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00

INSERT

**INSERT INTO TableName [(columnList)]
VALUES (dataValueList)**

- ***columnList* is optional; if omitted, SQL assumes a list of all columns in their original CREATE TABLE order.**
- **Any columns omitted must have been declared as NULL when table was created, unless DEFAULT was specified when creating column.**

INSERT

- ***dataValueList* must match *columnList* as follows:**
 - number of items in each list must be same;
 - must be direct correspondence in position of items in two lists;
 - data type of each item in ***dataValueList*** must be compatible with data type of corresponding column.

Example 5.35 INSERT ... VALUES

Insert a new row into Staff table supplying data for all columns.

INSERT INTO Staff

**VALUES ('SG16', 'Alan', 'Brown', 'Assistant', 'M',
Date'1957-05-25', 8300, 'B003');**

Example 5.36 INSERT using Defaults

Insert a new row into Staff table supplying data for all mandatory columns.

```
INSERT INTO Staff (staffNo, fName, lName,  
                  position, salary, branchNo)  
VALUES ('SG44', 'Anne', 'Jones',  
        'Assistant', 8100, 'B003');
```

- **Or**

```
INSERT INTO Staff  
VALUES ('SG44', 'Anne', 'Jones', 'Assistant', NULL,  
        NULL, 8100, 'B003');
```

UPDATE

UPDATE TableName

SET columnName1 = dataValue1

[, columnName2 = dataValue2...]

[WHERE searchCondition]

- ***TableName*** can be name of a base table or an updatable view.
- **SET** clause specifies names of one or more columns that are to be updated.

UPDATE

- **WHERE clause is optional:**
 - if omitted, named columns are updated for all rows in table;
 - if specified, only those rows that satisfy *searchCondition* are updated.
- **New *dataValue(s)* must be compatible with data type for corresponding column.**

Example 5.38/39 UPDATE All Rows

Give all staff a 3% pay increase.

UPDATE Staff

SET salary = salary*1.03;

Give all Managers a 5% pay increase.

UPDATE Staff

SET salary = salary*1.05

WHERE position = 'Manager';

Example 5.40 UPDATE Multiple Columns

Promote David Ford (staffNo='SG14') to Manager and change his salary to £18,000.

UPDATE Staff

SET position = 'Manager', salary = 18000

WHERE staffNo = 'SG14';

DELETE

**DELETE FROM TableName
[WHERE searchCondition]**

- *TableName* can be name of a base table or an updatable view.
- *searchCondition* is optional; if omitted, all rows are deleted from table. This does not delete table. If *search_condition* is specified, only those rows that satisfy condition are deleted.

Example 5.41/42 DELETE Specific Rows

Delete all viewings that relate to property PG4.

DELETE FROM Viewing

WHERE propertyNo = 'PG4';

Delete all records from the Viewing table.

DELETE FROM Viewing;

SQL: Data Definition

Section Objectives

- **Data types supported by SQL standard.**
- **Purpose of integrity enhancement feature of SQL.**
- **How to define integrity constraints using SQL.**
- **How to use the integrity enhancement feature in the CREATE and ALTER TABLE statements.**

Section Objectives

- **Purpose of views.**
- **How to create and delete views using SQL.**
- **How the DBMS performs operations on views.**
- **Under what conditions views are updatable.**
- **Advantages and disadvantages of views.**
- **How the ISO transaction model works.**
- **How to use the GRANT and REVOKE statements as a level of security.**

ISO SQL Data Types

Table 6.1 ISO SQL data types.

Data type	Declarations			
boolean	BOOLEAN			
character	CHAR	VARCHAR		
bit	BIT	BIT VARYING		
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION	
datetime	DATE	TIME	TIMESTAMP	
interval	INTERVAL			
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT	

Integrity Enhancement Feature

- **Consider five types of integrity constraints:**
 - required data
 - domain constraints
 - entity integrity
 - referential integrity
 - general constraints.

Integrity Enhancement Feature

Required Data

position VARCHAR(10) NOT NULL

Domain Constraints

(a) CHECK

sex CHAR NOT NULL

CHECK (sex IN ('M', 'F'))

Integrity Enhancement Feature

(b) CREATE DOMAIN

```
CREATE DOMAIN DomainName [AS] dataType  
[DEFAULT defaultOption]  
[CHECK (searchCondition)]
```

For example:

```
CREATE DOMAIN SexType AS CHAR  
CHECK (VALUE IN ('M', 'F'));  
sexSexType NOT NULL
```

Integrity Enhancement Feature

- *searchCondition* can involve a table lookup:

```
CREATE DOMAIN BranchNo AS CHAR(4)  
CHECK (VALUE IN (SELECT branchNo  
FROM Branch));
```

- Domains can be removed using DROP DOMAIN:

```
DROP DOMAIN DomainName  
[RESTRICT | CASCADE]
```

IEF - Entity Integrity

- **Primary key of a table must contain a unique, non-null value for each row.**
- **ISO standard supports FOREIGN KEY clause in CREATE and ALTER TABLE statements:**

PRIMARY KEY(staffNo)

PRIMARY KEY(clientNo, propertyNo)

- **Can only have one PRIMARY KEY clause per table. Can still ensure uniqueness for alternate keys using UNIQUE:**

UNIQUE(telNo)

IEF - Referential Integrity

- **FK is column or set of columns that links each row in child table containing foreign FK to row of parent table containing matching PK.**
- **Referential integrity means that, if FK contains a value, that value must refer to existing row in parent table.**
- **ISO standard supports definition of FKs with FOREIGN KEY clause in CREATE and ALTER TABLE:**

FOREIGN KEY(branchNo) REFERENCES Branch

IEF - Referential Integrity

- Any INSERT/UPDATE attempting to create FK value in child table without matching CK value in parent is rejected.
- Action taken attempting to update/delete a CK value in parent table with matching rows in child is dependent on referential action specified using ON UPDATE and ON DELETE subclauses:
 - CASCADE- SET NULL
 - SET DEFAULT - NO ACTION

IEF - Referential Integrity

CASCADE: Delete row from parent and delete matching rows in child, and so on in cascading manner.

SET NULL: Delete row from parent and set FK column(s) in child to NULL. Only valid if FK columns are NOT NULL.

SET DEFAULT: Delete row from parent and set each component of FK in child to specified default. Only valid if DEFAULT specified for FK columns.

NO ACTION: Reject delete from parent. Default.

IEF - Referential Integrity

**FOREIGN KEY (staffNo) REFERENCES Staff
ON DELETE SET NULL**

**FOREIGN KEY (ownerNo) REFERENCES Owner
ON UPDATE CASCADE**

IEF - General Constraints

- Could use CHECK/UNIQUE in CREATE and ALTER TABLE.
- Similar to the CHECK clause, also have:

**CREATE ASSERTION AssertionName
CHECK (searchCondition)**

IEF - General Constraints

```
CREATE ASSERTION StaffNotHandlingTooMuch  
CHECK (NOT EXISTS (SELECT staffNo  
                     FROM PropertyForRent  
                     GROUP BY staffNo  
                     HAVING COUNT(*) > 100))
```

Data Definition

- **SQL DDL allows database objects such as schemas, domains, tables, views, and indexes to be created and destroyed.**
- **Main SQL DDL statements are:**

CREATE SCHEMA DROP SCHEMA

CREATE/ALTER DOMAIN DROP DOMAIN

CREATE/ALTER TABLE DROP TABLE

CREATE VIEW DROP VIEW

- **Many DBMSs also provide:**

CREATE INDEX DROP INDEX

Data Definition

- Relations and other database objects exist in an *environment*.
- Each environment contains one or more *catalogs*, and each catalog consists of set of schemas.
- Schema is named collection of related database objects.
- Objects in a schema can be tables, views, domains, assertions, collations, translations, and character sets. All have same owner.

CREATE SCHEMA

CREATE SCHEMA [Name |

AUTHORIZATION CreatorId]

DROP SCHEMA Name [RESTRICT | CASCADE]

- **With RESTRICT (default), schema must be empty or operation fails.**
- **With CASCADE, operation cascades to drop all objects associated with schema in order defined above. If any of these operations fail, DROP SCHEMA fails.**

CREATE TABLE

```
CREATE TABLE TableName
{(colName dataType [NOT NULL] [UNIQUE]
[DEFAULT defaultOption]
[CHECK searchCondition] [...]}
[PRIMARY KEY (listOfColumns),]
{[UNIQUE (listOfColumns),] [...,]}
{[FOREIGN KEY (listOfFKColumns)
REFERENCES ParentTableName [(listOfCKColumns)],
[ON UPDATE referentialAction]
[ON DELETE referentialAction ]] [...]}
{[CHECK (searchCondition)] [...] })
```

CREATE TABLE

- Creates a table with one or more columns of the specified *dataType*.
- With NOT NULL, system rejects any attempt to insert a null in the column.
- Can specify a DEFAULT value for the column.
- Primary keys should always be specified as NOT NULL.
- FOREIGN KEY clause specifies FK along with the referential action.

Example - CREATE TABLE

```
CREATE DOMAIN OwnerNumber AS VARCHAR(5)
    CHECK (VALUE IN (SELECT ownerNo FROM PrivateOwner));
CREATE DOMAIN StaffNumber AS VARCHAR(5)
    CHECK (VALUE IN (SELECT staffNo FROM Staff));
CREATE DOMAIN PNumber AS VARCHAR(5);
CREATE DOMAIN PRooms AS SMALLINT;
    CHECK(VALUE BETWEEN 1 AND 15);
CREATE DOMAIN PRent AS DECIMAL(6,2)
    CHECK(VALUE BETWEEN 0 AND 9999.99);
```

Example - CREATE TABLE

```
CREATE TABLE PropertyForRent (  
    propertyNo    PNumber NOT NULL, ....  
    rooms        PRooms      NOT NULL DEFAULT 4,  
    rent         PRent        NOT NULL,    DEFAULT 600,  
    ownerNo      OwnerNumber NOT NULL,  
    staffNo      StaffNumber  
                Constraint StaffNotHandlingTooMuch ....  
    branchNo     BranchNumber NOT NULL,  
    PRIMARY KEY (propertyNo),  
    FOREIGN KEY (staffNo) REFERENCES Staff  
    ON DELETE SET NULL ON UPDATE CASCADE ....);
```

ALTER TABLE

- **Add a new column to a table.**
- **Drop a column from a table.**
- **Add a new table constraint.**
- **Drop a table constraint.**
- **Set a default for a column.**
- **Drop a default for a column.**

Example (a) - ALTER TABLE

Change Staff table by removing default of 'Assistant' for position column and setting default for sex column to female ('F').

ALTER TABLE Staff

ALTER position DROP DEFAULT;

ALTER TABLE Staff

ALTER sex SET DEFAULT 'F';

Example (b) - ALTER TABLE

Remove constraint from PropertyForRent that staff are not allowed to handle more than 100 properties at a time. Add new column to Client table.

```
ALTER TABLE PropertyForRent  
    DROP CONSTRAINT StaffNotHandlingTooMuch;  
ALTER TABLE Client  
    ADD prefNoRooms PRooms;
```

DROP TABLE

DROP TABLE TableName [RESTRICT | CASCADE]

e.g. DROP TABLE PropertyForRent;

- **Removes named table and all rows within it.**
- **With RESTRICT, if any other objects depend for their existence on continued existence of this table, SQL does not allow request.**
- **With CASCADE, SQL drops all dependent objects (and objects dependent on these objects).**

Views

View

Dynamic result of one or more relational operations operating on base relations to produce another relation.

- **Virtual relation that does not necessarily actually exist in the database but is produced upon request, at time of request.**

Views

- Contents of a view are defined as a query on one or more base relations.
- With view resolution, any operations on view are automatically translated into operations on relations from which it is derived.
- With view materialization, the view is stored as a temporary table, which is maintained as the underlying base tables are updated.

SQL - CREATE VIEW

```
CREATE VIEW ViewName [ (newColumnName [,...]) ]  
    AS subselect  
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

- Can assign a name to each column in view.
- If list of column names is specified, it must have same number of items as number of columns produced by *subselect*.
- If omitted, each column takes name of corresponding column in *subselect*.

SQL - CREATE VIEW

- List must be specified if there is any ambiguity in a column name.
- The *subselect* is known as the defining query.
- **WITH CHECK OPTION** ensures that if a row fails to satisfy **WHERE** clause of defining query, it is not added to underlying base table.
- Need **SELECT** privilege on all tables referenced in subselect and **USAGE** privilege on any domains used in referenced columns.

Example - Create Horizontal View

Create view so that manager at branch B003 can only see details for staff who work in his or her office.

```
CREATE VIEW Manager3Staff  
AS SELECT *  
FROM Staff
```

Table 6.3 Data for view Manager3Staff.

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003

Example - Create Vertical View

Create view of staff details at branch B003 excluding salaries.

CREATE VIEW Staff3

AS SELECT staffNo, fName, lName, position, sex

FROM Staff

WHERE branc Table 6.4 Data for view Staff3.

staffNo	fName	lName	position	sex
SG37	Ann	Beech	Assistant	F
SG14	David	Ford	Supervisor	M
SG5	Susan	Brand	Manager	F

Example - Grouped and Joined Views

Create view of staff who manage properties for rent, including branch number they work at, staff number, and number of properties they manage.

```
CREATE VIEW StaffPropCnt (branchNo, staffNo, cnt)  
AS SELECT s.branchNo, s.staffNo, COUNT(*)  
FROM Staff s, PropertyForRent p  
WHERE s.staffNo = p.staffNo  
GROUP BY s.branchNo, s.staffNo;
```

Example - Grouped and Joined Views

Table 6.5 Data for view StaffPropCnt.

branchNo	staffNo	cnt
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

SQL - DROP VIEW

DROP VIEW ViewName [RESTRICT | CASCADE]

- **Causes definition of view to be deleted from database.**
- **For example:**

DROP VIEW Manager3Staff;

SQL - DROP VIEW

- **With CASCADE, all related dependent objects are deleted; i.e. any views defined on view being dropped.**
- **With RESTRICT (default), if any other objects depend for their existence on continued existence of view being dropped, command is rejected.**

View Resolution

Count number of properties managed by each member at branch B003.

```
SELECT staffNo, cnt  
FROM StaffPropCnt  
WHERE branchNo = 'B003'  
ORDER BY staffNo;
```

View Resolution

- (a) View column names in SELECT list are translated into their corresponding column names in the defining query:

```
SELECT s.staffNo As staffNo, COUNT(*) As cnt
```

- (b) View names in FROM are replaced with corresponding FROM lists of defining query:

```
FROM Staff s, PropertyForRent p
```

View Resolution

- (c) **WHERE** from user query is combined with **WHERE** of defining query using **AND**:

WHERE s.staffNo = p.staffNo AND branchNo = 'B003'

- (d) **GROUP BY** and **HAVING** clauses copied from defining query:

GROUP BY s.branchNo, s.staffNo

- (e) **ORDER BY** copied from query with view column name translated into defining query column name

ORDER BY s.staffNo

View Resolution

(f) Final merged query is now executed to produce the result:

```
SELECT s.staffNo AS staffNo, COUNT(*) AS cnt  
FROM Staff s, PropertyForRent p  
WHERE s.staffNo = p.staffNo AND  
       branchNo = 'B003'  
GROUP BY s.branchNo, s.staffNo  
ORDER BY s.staffNo;
```

Restrictions on Views

SQL imposes several restrictions on creation and use of views.

(a) If column in view is based on an aggregate function:

- Column may appear only in SELECT and ORDER BY clauses of queries that access view.**
- Column may not be used in WHERE nor be an argument to an aggregate function in any query based on view.**

Restrictions on Views

- For example, following query would fail:

```
SELECT COUNT(cnt)
FROM StaffPropCnt;
```

- Similarly, following query would also fail:

```
SELECT *
FROM StaffPropCnt
WHERE cnt > 2;
```

Restrictions on Views

- (b) Grouped view may never be joined with a base table or a view.**
- For example, StaffPropCnt view is a grouped view, so any attempt to join this view with another table or view fails.**

View Updatability

- All updates to base table reflected in all views that encompass base table.
- Similarly, may expect that if view is updated then base table(s) will reflect change.

View Updatability

- However, consider again view StaffPropCnt.
- If we tried to insert record showing that at branch B003, SG5 manages 2 properties:

```
INSERT INTO StaffPropCnt  
VALUES ('B003', 'SG5', 2);
```

- Have to insert 2 records into PropertyForRent showing which properties SG5 manages. However, do not know which properties they are; i.e. do not know primary keys!

View Updatability

- If change definition of view and replace count with actual property numbers:

```
CREATE VIEW StaffPropList (branchNo,  
                           staffNo, propertyNo)
```

```
AS SELECT s.branchNo, s.staffNo, p.propertyNo  
FROM Staff s, PropertyForRent p  
WHERE s.staffNo = p.staffNo;
```

View Updatability

- Now try to insert the record:

```
INSERT INTO StaffPropList  
VALUES ('B003', 'SG5', 'PG19');
```

- Still problem, because in PropertyForRent all columns except postcode/staffNo are not allowed nulls.
- However, have no way of giving remaining non-null columns values.

View Updatability

- **ISO specifies that a view is updatable if and only if:**
 - **DISTINCT is not specified.**
 - **Every element in SELECT list of defining query is a column name and no column appears more than once.**
 - **FROM clause specifies only one table, excluding any views based on a join, union, intersection or difference.**
 - **No nested SELECT referencing outer table.**
 - **No GROUP BY or HAVING clause.**
 - **Also, every row added through view must not violate integrity constraints of base table.**

Updatable View

For view to be updatable, DBMS must be able to trace any row or column back to its row or column in the source table.

WITH CHECK OPTION

- Rows exist in a view because they satisfy WHERE condition of defining query.
- If a row changes and no longer satisfies condition, it disappears from the view.
- New rows appear within view when insert/update on view cause them to satisfy WHERE condition.
- Rows that enter or leave a view are called *migrating rows*.
- **WITH CHECK OPTION** prohibits a row migrating out of the view.

WITH CHECK OPTION

- **LOCAL/CASCADDED** apply to view hierarchies.
- **With LOCAL**, any row insert/update on view and any view directly or indirectly defined on this view must not cause row to disappear from view unless row also disappears from derived view/table.
- **With CASCADDED (default)**, any row insert/ update on this view and on any view directly or indirectly defined on this view must not cause row to disappear from the view.

Example - WITH CHECK OPTION

```
CREATE VIEW Manager3Staff  
AS SELECT *  
FROM Staff  
WHERE branchNo = 'B003'  
WITH CHECK OPTION;
```

- **Cannot update branch number of row B003 to B002 as this would cause row to migrate from view.**
- **Also cannot insert a row into view with a branch number that does not equal B003.**

Example - WITH CHECK OPTION

- Now consider the following:

```
CREATE VIEW LowSalary
```

```
AS    SELECT * FROM Staff WHERE salary > 9000;
```

```
CREATE VIEW HighSalary
```

```
AS    SELECT * FROM LowSalary
```

```
      WHERE salary > 10000
```

```
      WITH LOCAL CHECK OPTION;
```

```
CREATE VIEW Manager3Staff
```

```
AS    SELECT * FROM HighSalary
```

```
      WHERE branchNo = 'B003';
```

Example - WITH CHECK OPTION

```
UPDATE Manager3Staff  
SET salary = 9500  
WHERE staffNo = 'SG37';
```

- This update would fail: although update would cause row to disappear from HighSalary, row would not disappear from LowSalary.
- However, if update tried to set salary to 8000, update would succeed as row would no longer be part of LowSalary.

Example - WITH CHECK OPTION

- **If HighSalary had specified WITH CASCADED CHECK OPTION, setting salary to 9500 or 8000 would be rejected because row would disappear from HighSalary.**
- **To prevent anomalies like this, each view should be created using WITH CASCADED CHECK OPTION.**

Advantages of Views

- **Data independence**
- **Concurrency**
- **Improved security**
- **Reduced complexity**
- **Convenience**
- **Customization**
- **Data integrity**

Disadvantages of Views

- **Update restriction**
- **Structure restriction**
- **Performance**