



Databases & 4GLs

DATABASE ARCHITECTURE



Topics of Discussion

- 1. ANSI - SPARC Three-Level Database Architecture**
- 2. Logical and Physical Data Independence**
- 3. Data Model**
- 4. Database Languages (DDL & DML)**
- 5. Components of DBMS**
- 6. File Server vs Client Server Database Architecture**



Overview

- Major aim of a DBMS is to provide users with an **abstract** view of data, by hiding certain details of how data is stored and manipulated.
- Starting point for the database design must be an abstract and general description of the information requirements of the organization.
- Database is a shared resource, each user may require a different view of the data held in the database.
- Most commercial DBMS available today is based on the framework so-called **ANSI-SPARC** architecture.



Database Architecture Proposal

- First standard terminology and general architecture proposal for database system was done in 1971.
- By **DBTG (Data Base Task Group)** appointed by the Conference on Data Systems and Languages (**CODASYL**, 1971).
- Two-level approach :-
 - system view - ***schema***
 - user view - ***subschemas***
- Similar proposal was done later in 1975 by ANSI-SPARC.



ANSI-SPARC Architecture

- American National Standards Institute - Standards Planning and Requirements Committee.
- Based on Three-Level Approach.
- Three-Level approach with an implementation-independent layer to isolate programs from underlying data representational issues.
- The three-level architecture comprises :-
 - 1) External level**
 - 2) Conceptual level**
 - 3) Internal level**



Objectives for Three-Level Approach

- All users should be able to access same data, yet each should have their own customized view of it.
A user's view is immune to changes made in other views.
- Users should not need to know physical database storage details.
- 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.



External Level

- The users' view of database
- Describes the part of database that is relevant to a particular user.
- Consists of a number of different external views of the database.
- Each consists those ***entities***, ***attributes*** and ***relationships*** that the user is interested in.
- Different representation of same data eg. date - dmy, mdy.
- Derived or calculated data, eg. DOB, age.



Conceptual Level

- The community view of the database.
- Describes what data is stored in the database and the relationships among the data.
- Contains the logical structure of the entire database as seen by the DBA.
- It represents:
 - all entities, their attributes and relationships
 - the constraints on the data
 - semantic information about the data
 - security and integrity information
- Must not contain any storage-dependent details.



Internal Level

- The physical representation of the database on the computer.
- Describes how the data is stored in the database.
- Physical implementation to achieve optimal run-time performance and disk space utilization.
- Concern about data structure, file organization, record placement, data compression and encryption techniques.
- Some DBMS interfaces with the operating system access methods.



Database Schemas

- Overall description of the database.
- Three schemas exists corresponding to each level.
- **External Schemas** (subschemas), correspond to different view of data, might contain multiple schemas.
- **Conceptual Schemas**, describes all the data items and relationships between data items, together with integrity constraints. Only one conceptual schema per database.
- **Internal Schemas**, description of internal model, contains definitions of stored records, methods of representation, data fields, indexes and hashing schemes used.



Mappings

- DBMS is responsible to map between three types of schemas.
- Process to check schemas for consistency.

External schema is derivable from the conceptual schema and conceptual schema must be able to map between external and internal schema.

- **Conceptual / Internal Mapping**, enable DBMS to find the actual records in the physical storage that constitute to a logical record in the conceptual schema.
- **External / Conceptual Mapping**, enables DBMS to map names in the user's view onto the relevant part of the conceptual schema.



Database Instances

- Different between the description of database and the database itself.
- Description of database (schema) is created during the database design process. Not expected to change frequently.
- Actual data in database might change frequently.
- Data in the database at any particular point in time is called a ***database instance***.
- Many database instances can correspond to the same database schema.
- **Schema** - *intension* of database. **Instance** - *extension* (state) of database.



Data Independence

- Major objective of ANSI/SPARC three level architecture is to achieve **data independence**.
- Changes to lower level will not affect upper level.

Logical Data Independence

Immunity of external schemas to changes in conceptual schema.

Conceptual schema changes, such as addition/removal of entities, attributes, or relationships 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, such as using different file organizations, storage structure/devices, modifying indexes of hashing algorithms, should not require change to conceptual or external schemas.

Database Languages

- There are three types of database languages:
 - ❑ Data Definition Language (DDL)
 - ❑ Data Manipulation Language (DML)
 - ❑ 4th Generation Languages
- The Data Definition Language (DDL)
 - ❑ A descriptive language that allows the DBA or the user to *describe* and name the *entities* required for the application and the *relationships* that may exist between different entities
 - ❑ A DDL is used to define a schema or modify an existing one
 - ❑ A DDL cannot be used to manipulate data

Database Languages

- ❑ The result of the compilation of the DDL statements is a set of tables stored in special files collectively called data dictionary
- ❑ The DBMS normally consults the data dictionary before the actual data is accessed in the database
- ❑ The terms *system catalog*, *directory*, *metadata*, and *data dictionary* are used interchangeably
- The Data Manipulation Language (DML)
 - ❑ A language that provides a set of operations that support the basic data manipulation

Database Languages

- ❑ Data manipulation operations usually include the following:
 - the insertion of new data in the database;
 - the modification of data stored in the database
 - the retrieval of data contained in the database
 - the deletion of data in the database.
- ❑ There are two types of data manipulation languages:
 - Procedural DML
 - Non-Procedural DML
- ❑ Procedural DML
 - A language that allows the user to tell the system exactly *how to manipulate the data*
 - In the procedural languages the database statements treat records individually (sequentially)

Database Languages

❑ Non-Procedural DML

- A language that allows the user to state *what* data is needed rather than *how* to it is to be retrieved
- In non-procedural languages the statements operate on a set of records
- This will free the user from having to know:
 - 1) how data structures are internally implemented,
 - 2) what algorithms are required to retrieve data, and
 - 3) what algorithms are required to transform data
- Relational database systems make use of the non-procedural languages
- Examples of non-procedural languages are SQL and QBE

Database Languages

■ 4th Generation Languages

- ❑ 4GLs known as *shorthand* programming languages
- ❑ E.g. An operation that requires hundreds of LOC in a 3GL (COBOL), can be implemented in 20-30 LOC in 4GL
- ❑ 4GLs have improved productivity by a factor of 10
- ❑ While 3GLs are procedural, 4GLs are non-procedural (i.e. user defines *what* is to be done, not *how* is to be done)
- ❑ 4GLs rely heavily on 4GL tools (e.g. libraries)
- ❑ There are different types of 4GLs:

Database Languages

➤ FORM GENERATORS

- 1) create data input or screen layouts for screen forms
- 2) define what the screen looks like
- 3) can specify colors and typeface features (B, *I*, U)
- 4) allow creation of derived attributes
- 5) specification of validation check for data input

➤ REPORT GENERATION

- 1) create reports from data stored in the database
- 2) similar to query language
- 3) the user has a greater control of what the output looks like

4) control layout can be automatic or customized

Database Languages

5) two types of report generators

- language oriented (uses commands)

- visually oriented (user facility similar to form generator)

➤ GRAPHICS GENERATORS

- 1) retrieves data from the database and display data as a graph showing trends and relationships in the data

- 2) allows the users to create bar charts, pie charts, line chats, scatter charts, etc.

➤ APPLICATION GENERATIONS

- 1) to produce a program that interfaces with database

- 2) reduces time to design the entire software applitns.

- 3) consist of pre-written modules that comprise fundamental functions that most programs use
(library) (*WHAT VS HOW*)

DBMS Functions (Codd, 1982)

- Data storage, retrieval and update
- User accessible catalog (data about data)
 - ❑ names, types, sizes of data items;
 - ❑ names of relationships;
 - ❑ integrity constraints on the data;
 - ❑ names of authorized users;
 - ❑ external, conceptual, and internal schemas and the mappings;
 - ❑ usage statistics (frequencies of transaction)

DBMS Functions (Codd, 1982)

Benefits of using a data dictionary are:

- ❑ Information about data can be collected and stored centrally
- ❑ Meaning of the data can be defined
- ❑ Communication is simplified since exact meanings are stored
- ❑ Redundancy and inconsistencies can be identified more easily
- ❑ Changes to the database can be recorded
- ❑ The impact of change can be determined before it is implemented

DBMS Functions (Codd, 1982)

- ☐ Security can be enforced
- ☐ Integrity can be ensured
- ☐ Audit information can be provided

- Transaction support

- Concurrency control services

- ☐ Ensure that the database is updated correctly when multiple users are updating the database concurrently

- Recovery services

- Authorization services

- Support for data communication

DBMS Functions (Codd, 1982)

- Integrity services
- Services to promote data independence (physical and logical)
- Utility services
 - ❑ import/export facilities;
 - ❑ monitoring facilities;
 - ❑ statistical analysis programs;
 - ❑ index re-organization facilities; and
 - ❑ garbage collection.

COMPONENTS OF DBMS

A DBMS is partitioned into several software components each of which is assigned to a specific operation.

Some of the components in DBMS supported by an Operating System

COMPONENTS OF DBMS

1) QUERY PROCESSOR:

Used to transform queries into a series of low-level instructions directed to the Database Manager

2) Database Manager :

The DM interfaces with user-submitted application programs and queries.

It accepts the queries and examines the external and conceptual schema

Then places a call to the **File Manager** to perform the request

COMPONENTS OF DBMS

3) FILE MANAGER:

It manipulates the underlying storage files and manages the allocation of storage space on the disk.

It does not directly manage the physical input and output of data.

Rather it passes the requests on the appropriate access methods, which either read / write data into the system buffer.

COMPONENTS OF DBMS

3)DML PREPROCESSOR:

This is converts DML statements embedded in an application program into standard function calls in the host language.

4)DDL COMPILER :

It converts DDL statement into a set of tables containing METADATA.

5)DICTIONARY MANAGER:

It access to and maintains the data dictionary.

Major SW Components For the DM

- **Authorization Control :**
- **Command Processor :**
- **Integrity Checker**
- **Query Optimizer**
- **Transaction Manager**
- **Scheduler**
- **Recovery Manager**
- **Buffer Manager**



Data Model

- An integrated collection of concepts for describing data, relationships between data and constraints on the data in an organization.
- High level description of the schemas.
- Comprises of three components:

Structural part, consisting of a set of rules according to which database can be constructed.

Manipulative part, defining the types of operations that are allowed on the data e.g.. Updating, retrieving.

Set of integrity rules, ensure that data is accurate.



Record-Based Data Model

- Database consists of a number of fixed-format records, of possibly different types.
- Each record type defines a fixed number of fields of fixed length.

3 Types :

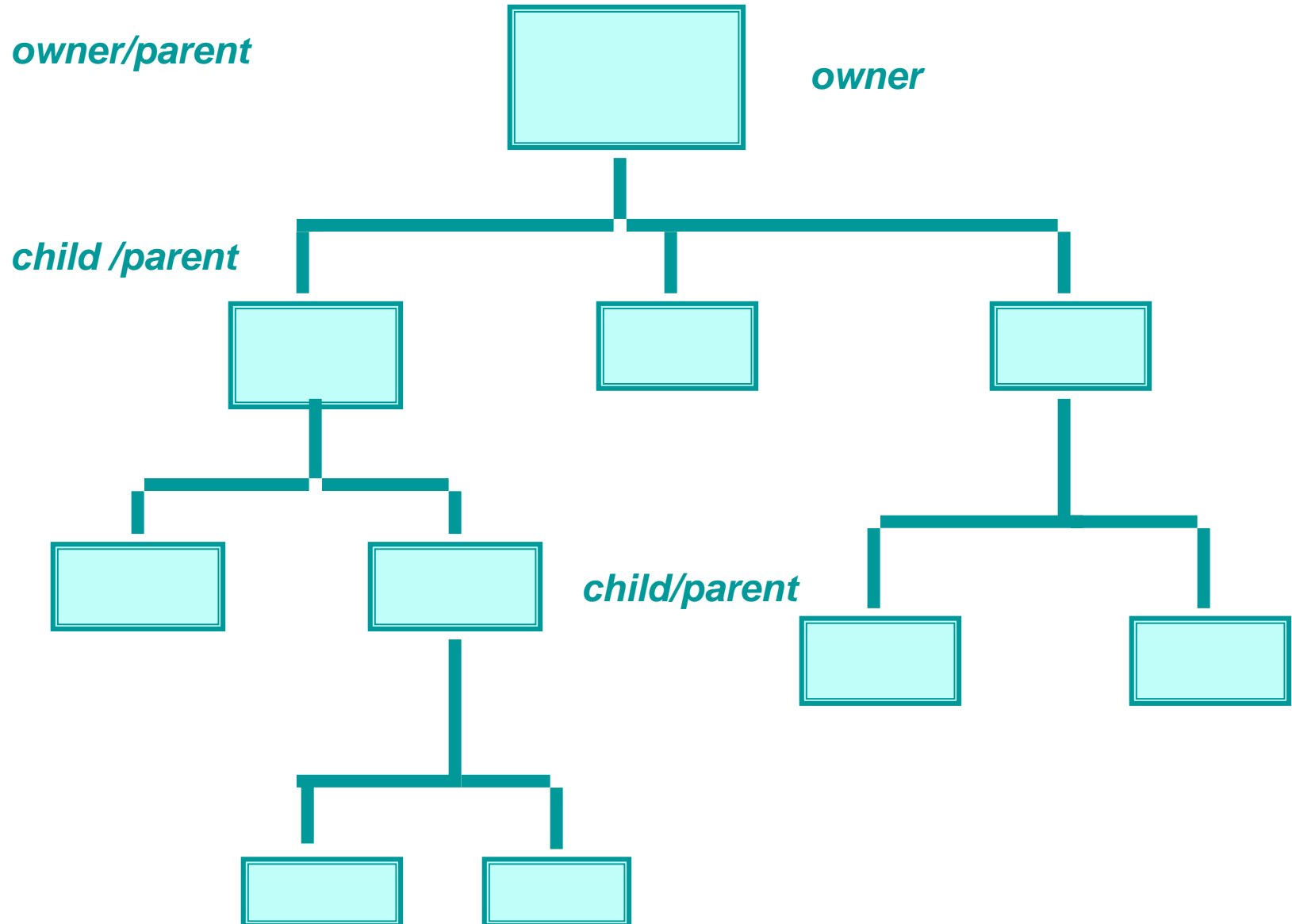
- 1) Hierarchical data model
- 2) Network data model
- 3) Relational data model



Hierarchical Data Model

- Restricted type of network data model.
- Data is represented as collections of records.
- Relationships are represented by sets.
- Typical tree graph, with records appearing as nodes (segment), and sets as edges.
- One to many relation.
- IBM's IMS.

Hierarchical Data Model

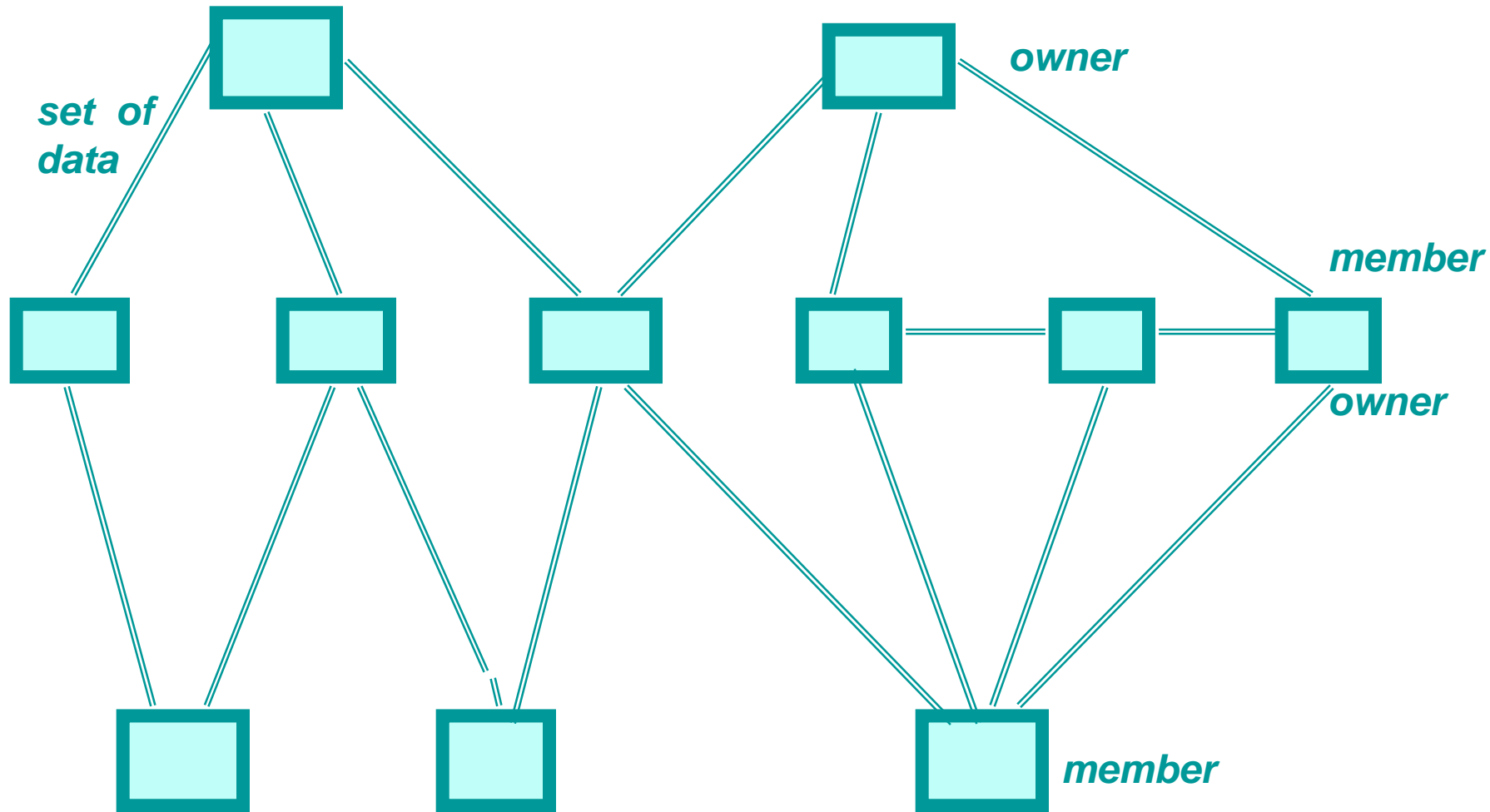




Network Data Model

- Data is represented as collections of records.
- Relationships are represented by sets.
- Records are organized as generalized graph structure with records appearing as nodes and sets as edges in the graph.
- Can have many to many relations.
- Computer Associates' IDMS/R

Network Data Model



set of data

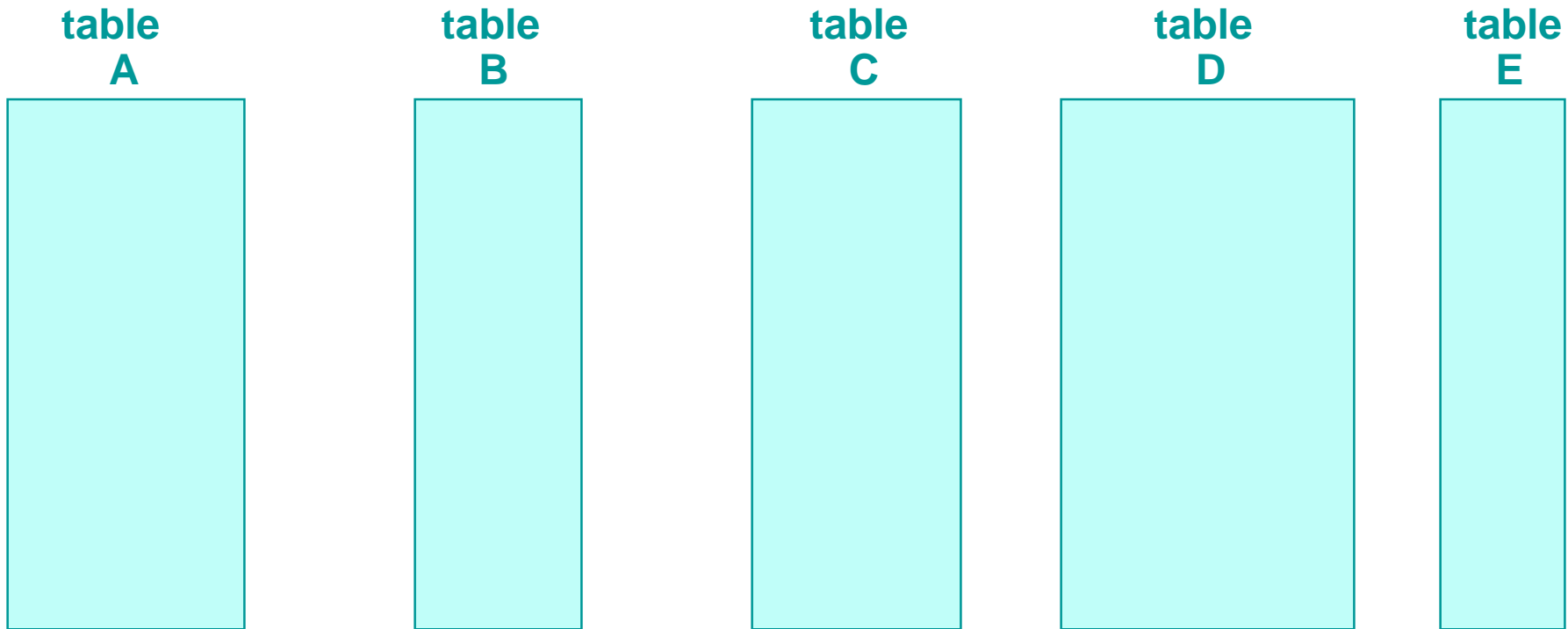
Note: Only linked sets can be accessed



Relational Data Model

- Data is represented in **ROW** and **COLUMN** form (matrix)
(tuple) (attribute)
- Collections of related data ---> TABLES (relations)
- Collection of 1 or more tables ----> DATA BASE
- ATTRIBUTES are generally static
- ROWS are DYNAMIC and Time-Varying

Relational Data Model



Any table(s) can be joined to any other table(s), *provided* there is a means of effecting the join

Primary key / Foreign key concept. Data redundancy

No fixed linkages

Table Relation

EMPNUM	NAME	Date of Birth	DEPTNUM
3	JONES	16-05-56	605
7	SMITH	23-09-65	432
11	ADAMS	11-08-72	201
15	NGUYEN	23-10-64	314
18	PHAN	16-11-76	201

Relation (Table) Name : EMP

Relation Schema: EMP(empnum,name,date of birth,deptnum)

DEPTNUM	DEPTNAME
201	Production
314	Finance
432	Information Systems
605	Administration

Relation (Table) Name : DEPT

Relation Schema: DEPT(deptnum, deptname)



Comparison of Data Model

- Relational model - data processing approach (what data to be retrieved).
- Network & Hierarchical - navigational approach (how data is retrieved).



Multi-User DBMS Architecture

- **Teleprocessing**
- **File-Server**
- **Client-Server**



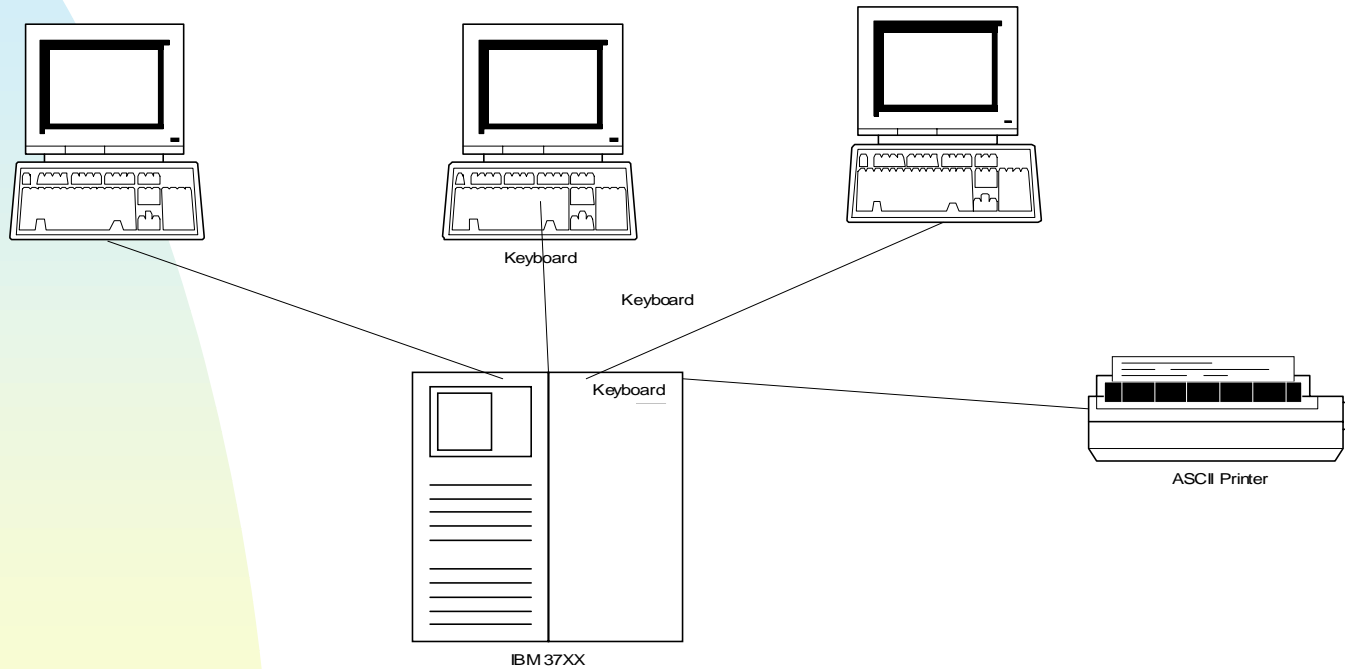
Teleprocessing Architecture

Teleprocessing Characteristics

- Traditional architecture.
- Single mainframe with a number of dumb terminals attached.
- Tremendous burden on the central computer.
- Now a trend towards downsizing - replacing expensive mainframe with more cost-effective networks & PCs.
- Give rise to File-server & Client-server architectures.



Teleprocessing





File-Server Architecture

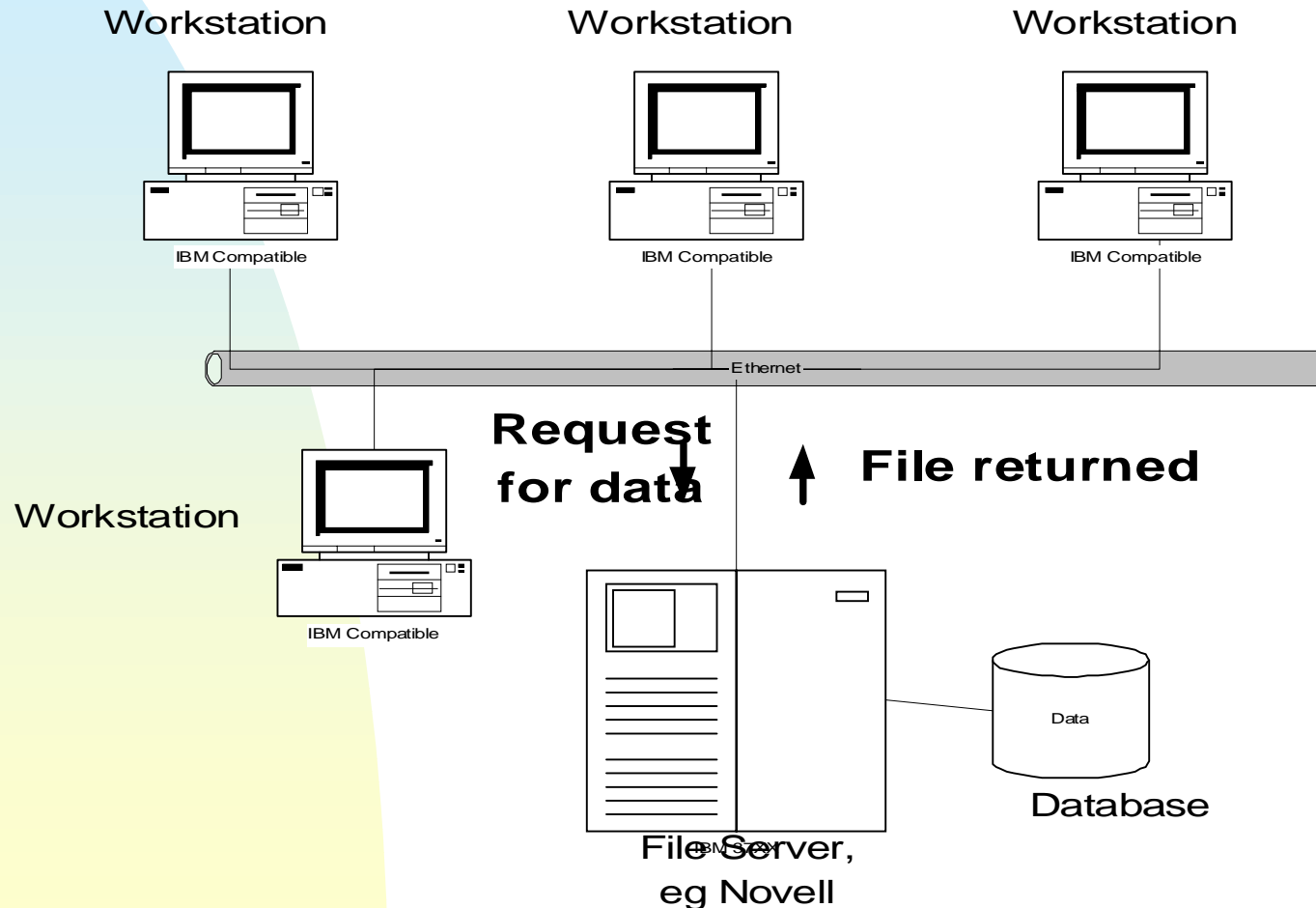
File-Server Characteristics

- File-server is connected to several workstations across a network.
- Database resides on file-server, and DBMS and applications run on each workstation.
- Processing done by workstation.

Disadvantages of File-Server Architecture:

1. Significant network traffic
2. Copy of DBMS on each workstation.
3. Concurrency, recovery and integrity control more complicated.

File-Server Architecture





Client-Server Architecture

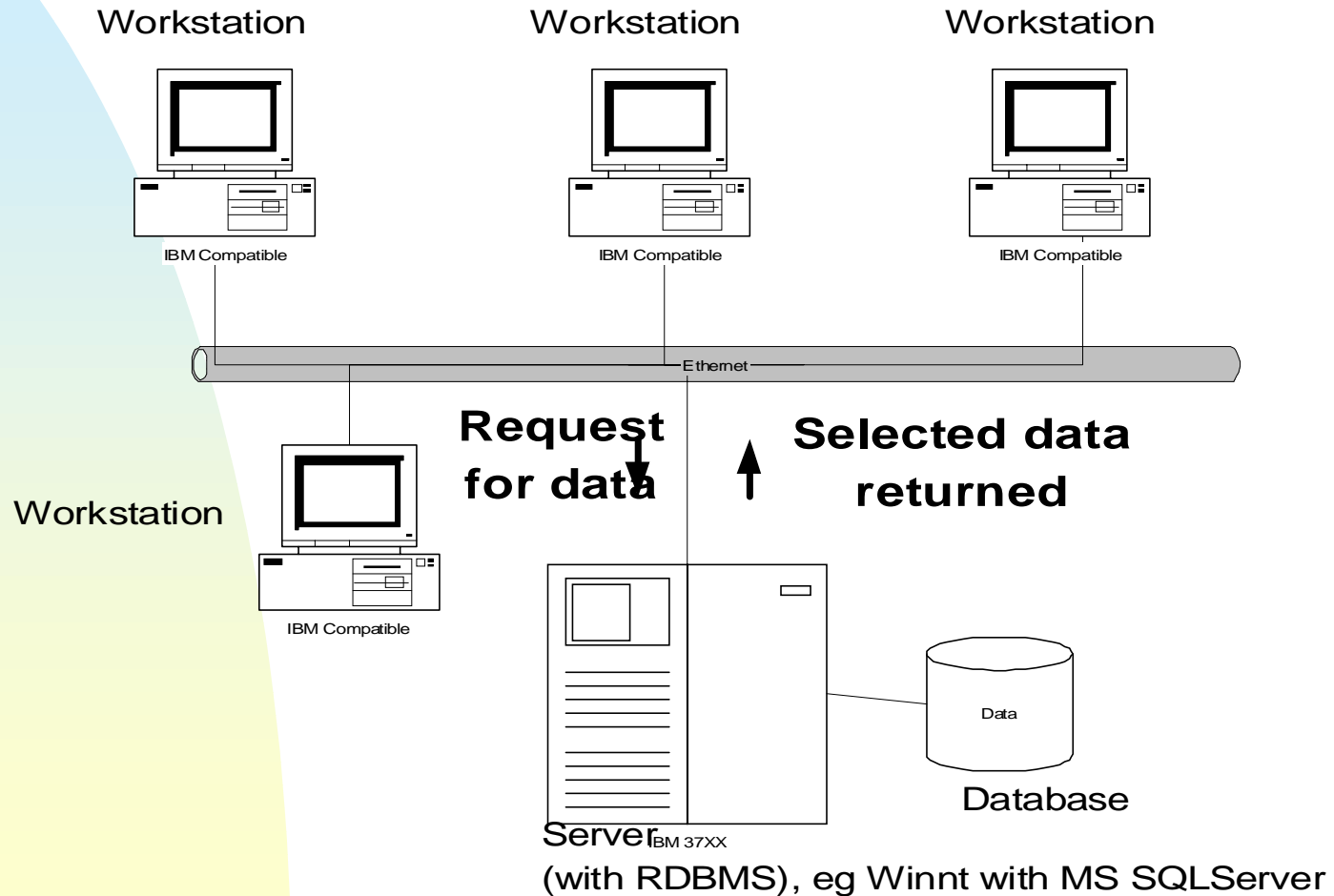
Client Server Characteristics

- Server holds the database and the DBMS.
- Client manages the user interface and run applications.
- Processing done by server.

Advantages of Client Server Architecture

1. Wider access to existing databases.
2. Increased performance.
3. Possible reduction in hardware costs.
4. Reduction in communication costs.
5. Increased consistency.

Client-Server Architecture





Client-Server Business Solution

- A **client-server business solution** is usually defined as an application distributed between a client machine and a server machine, connected over a network.
- A client-server solution does not have to be thought of in terms of physical processors and networks.
- It is the business solution that is divided into distinct parts — a ***logical*** rather than ***physical*** model for client-server computing.



Multitier Application Model

- A multitier model allows developers to break down complex business processes into digestible pieces, allowing for reusability.
- The simplest form of a multitier model is the three-tier model.



Multitier Application Model

3-tier architecture uses a services model that consists of the following:

- 1. User-Services Tier.** The front-end client that communicates with the user through a graphical user interface. Client-side code at this level calls upon the available business-services level to provide encapsulated business functionality.



Multitier Application Model

2. Business-Services Tier. A collection of services that enforce business rules, process information and manage transactions.

3. Data-Services Tier. A collection of decision-independent data used by the business-services level to make decisions. This can be a database management system (DBMS). For example, it could be data stored on a mainframe or accessed from a bulletin board service or the Internet.

3-Tier Client-Server Architecture

3-Tier Client/Server Overview

