Network Data Model
Hierarchical Data Model
Trends

First Generation DBMS
- Network Data Model
- Hierarchical Data Model

Network Model – Basic Concepts
- Data are represented as collection of records
- Relationships are represented as links
- Each record is a collection of fields:

  type customer = record
  customer-name: string;
  customer-street: string;
  customer-city: string;
  end

  type account = record
  account-number: string;
  balance: integer;
  end

Sample database.
Network Model – Data-Structure Diagrams

Data-structure diagram:
- Same purpose as an ER diagram.
- Boxes (record types)
- Lines (links)

One to Many from customer to account:

One to One:

Network Model – Relating 3 Record Types

A customer may have several accounts, each in a specific branch. An account may belong to several different customers.

A link can connect only 2 record types
⇒ We need to use a new record type (Rlink) as below:
Network Model – DBTG Model

- DBTG Model (late 1960s) – First database-standard specification.
- Using DBTG Sets to describe inter-record structures.
- Eg. a Depositor (1:M) link:

![Diagram of DBTG Model](image)

The DBTG Set (Depositor) is \{(a1,(b1,b2)), (a2,(b3,b4,b5)), (a3,(b6))\}

Network Model – DML (DBTG)

DML
- The DML consists of commands to be embedded in a host language (e.g., C).
- To select records based on a specified field value.
- To iterate over selected records by repeated comments.
- To find the owner record from a member.
- To find the members of an owner by iteration.
- To update the database.

Network Model – Implementation (DBTG)

- Links are implemented as pointer fields.
- Eg. M:1 from account to customer
  - In an account record, a pointer is stored to point to the customer record.
  - In a customer record, instead of using multiple pointers to point to the accounts, we use a ring structure.
**Network Model – Implementation (DBTG)**

- Difficult to implement M:N using pointers
- Therefore, DBTG Model does not allow M:N links.
- To describe M:N relationships, a new dummy record type is required (i.e. using two 1:M links):

  ![Diagram](image1.png)

**Network Model vs Relational Model**

The Network Model

- It is closely tied to the implementation.
- Increases the burden on the programmer for DB design and data manipulationship.
- Is efficient compared with early relational implementations.

**Hierarchical Model**

- Also use Records, Links (similar to Network Model)
- Database is collection of rooted trees => forest

  ![Diagram](image2.png)

**Hierarchical Model – Tree-Structure Diagrams**

Tree-structure diagram (cf. Data-structure diagram)

- No cycle (cf. arbitrary graph in Network Model)
- Only 1:1 and 1:M can be directly represented

  ![Diagram](image3.png)
Hierarchical Model – Tree-Structure Diagrams

- For M:N relationships, we need 2 separate tree-structure diagrams.

Hierarchical Model – Sample Database Trees

Hierarchical Model – Tree-Structure Diagrams

- For the following relationships between branch, account, and customer:

we need 2 separate tree-structure diagrams:

Hierarchical Model – Implementation

- To avoid record duplication, use *virtual records*:

Tree-structure diagram with virtual records
Hierarchical Model – Implementation

Implementation example:

IMS: Information Management System (IBM, mid-1960s)
- One of oldest and most widely used DB system
- The first to deal with the issues of concurrency, recovery, integrity, efficient query processing.
- Similar disadvantages and impact as Network Model.

Hierarchical Model – IMS

Trends
- Distributed DBMS
- Object DBMS

Trends – Distributed DBMSs

Distributed Database
A logically interrelated collection of shared data (and a description of this data), physically distributed over a computer network.

Distributed DBMS
Software system that permits the management of the distributed database and makes the distribution transparent to users.
Trends – Distributed DBMSs

Distributed DBMSs
Three key issues:

Fragmentation
Relation may be divided into a number of sub-relations, which are then distributed.

Allocation
Each fragment is stored at site with "optimal" distribution.

Replication
Copy of fragment may be maintained at several sites.

Advantages
- Organizational Structure
- Shareability & Local Autonomy
- Improved Availability
- Improved Reliability
- Improved Performance
- Economics
- Modular Growth

Disadvantages
- Complexity
- Cost
- Security
- Difficult Integrity Control
- Lack of Standards
- Lack of Experience
- Complex DB Design

Trends – Object DBMSs

The needs of advanced DB applications:
- Computer-Aided Design (CAD)
- Computer-Aided Manufacturing (CAM)
- Computer-Aided Software Engineering (CASE)
- Office Information Systems (OIS) and Multimedia Systems
- Digital Publishing
- Geographic Information Systems (GIS)
Object Oriented Concepts

Abstraction
- Identify essential aspects of an entity and ignore the unimportant properties.
- Concentrate on what an object is and what it does.
- Delay implementation details.

2 aspects:
- Encapsulation
  An object contains data structure and operations.
- Information hiding
  We present external aspects of an object to the outside world and hide its internal details.

Object : Employee

Attributes:
- Name
- Date of Birth
- Dept.
- Salary

Methods:
- Change Salary
- Calculate Age
- Delete

Objects communicate by sending messages:

```
method void ChangeSalary(int change)
{
    Salary = Salary + change;
}
```

Class

- Similar objects can be grouped as a class (cf. a type or a structure in C)
- Each such object is called an instance (cf. a variable)

Inheritance
- Allows one class to be defined as a special case of a more general class (subclass / superclass)

Inheritance diagram:

```
Person

Employee

Customer

Officer

Teller

Secretary

Superclass

Subclass
```

Debate: Object DBMSs vs Relational DBMSs

OODBMS proponents:
“Relational DBMSs are satisfactory for standard business applications but lack the capability of supporting more complex applications”

Relational supporters:
“Relational technology is a necessary part of any real DBMS, and complex applications can be handled by extensions to the relational model”
**Trends - Summary**

**First Generation DBMS - Network and Hierarchical**
- Required complex programs for even simple queries.
- Minimal data independence.
- No widely accepted theoretical foundation.

**Second Generation DBMS - Relational DBMS**
- Helped overcome these problems.

**Third Generation DBMS - OODBMS and ORDBMS**
- Response to increasing complexity of DB applications