Outline

- Database
  - What, Why, How
- Evolution of Database
  - File System
  - Data Models
    - Hierarchical
    - Network
    - Relational
    - Entity-Relationship
    - Object-Oriented
  - Web Database

Database management concepts

- Database Management Systems (DBMS)
  - An example of a database (relational)
  - Database schema (e.g. relational)
  - Data independence
  - Architecture of a DBMS
  - Types of DBMS
  - Basic DBMS types
  - Retrieving and manipulating data: query processing
  - Database views
- Data integrity
- Client-Server architectures
- Knowledge Bases and KBS (and area of AI)

- DBMS tasks:
  - Managing large quantity of structured data
  - Efficient retrieval and modification: query processing and optimization
  - Sharing data: multiple users use and manipulate data
  - Controlling the access to data: maintaining the data integrity
- An example of a database (relational):
  - Relations (tables)
  - Attributes (columns)
  - Tuples (rows)
  - Example query: Salesperson='Mary' AND Price>100.

Relation ITEM

Item#	ItemName	Quantity	Location
123	pump	25	A23.2
235	saw	42	B3.9
589	hose	110	A23.5
601	ladder	12	B14.6
...	...	...	...

Relation SUPPLIES

Item#	Supplier#
123	23
235	23
589	99
601	6
...	...

Relation SALES

Item#	Salesperson	Price
601	Sam	169.95
123	Sam	99.95
589	Mary	24.98
601	John	169.95
123	Mary	99.95
601	Mary	169.95
235	John	25.49
...	...	...

Relation SUPPLIER

Supplier#	City	Phone
6	Albany	518-555-1234
23	Troy	518-555-4321
48	Schenectady	518-555-6789
99	New York	201-555-9876
...	...	...

Figure 9.1 The inventory relational database

- Database schema (e.g. relational):
  - Names and types of attributes
  - Addresses
  - Indexing
  - Statistics
  - Authorization rules to access data etc.
- Data independence: separation of the physical and logical data
  - Particularly important for distributed systems
  - The mapping between them is provided by the schema
- Architecture of a DBMS - three levels: external, conceptual and internal schem
- Types of DBMS
  - The data structures supported: tables (relational), trees, networks, objec
  - Type of service provided: high level query language, programming primit

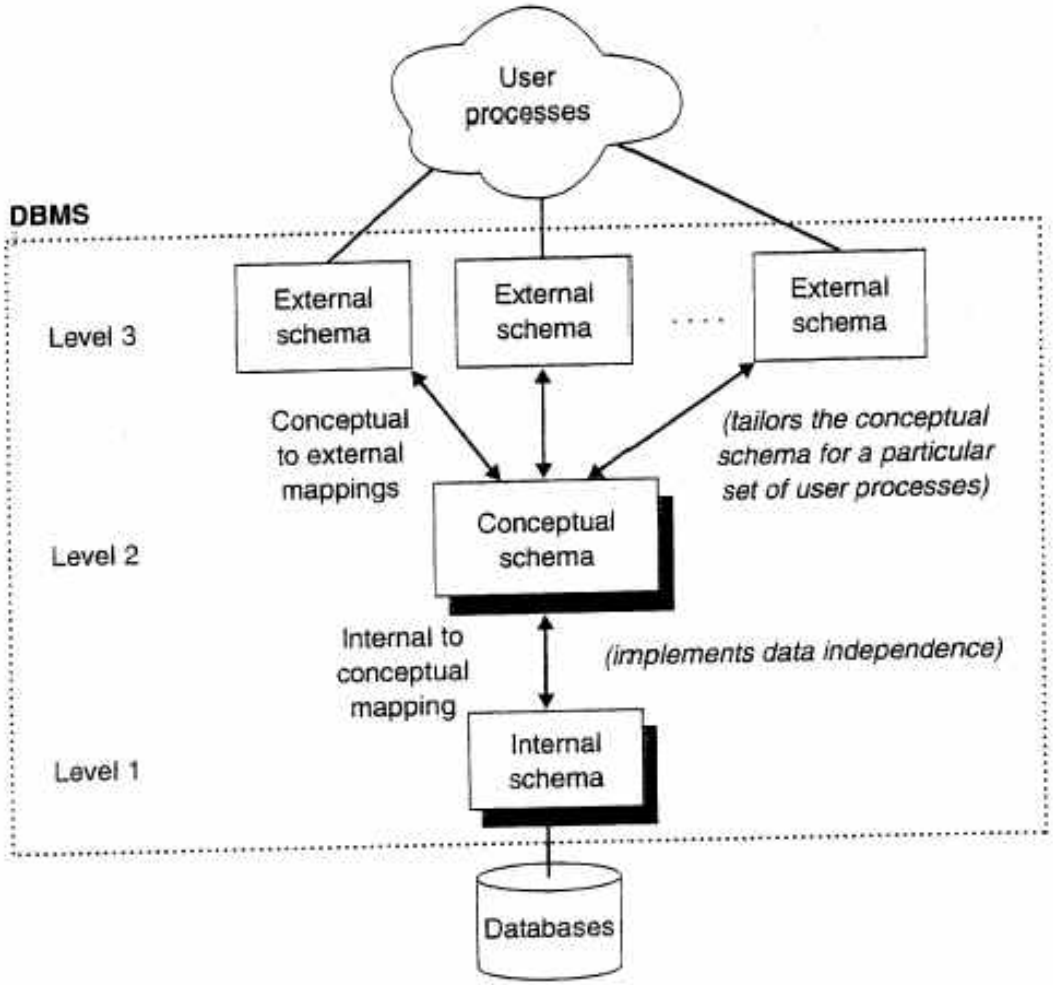


Figure 9.2 The three levels in a DBMS. *Source:* D. Tsichritzis and A. Klug, eds., *The ANSI/X3/SPARC DBMS Framework* (Montvale, NJ: AFIPS Press, 1978).

Basic DBMS types

- Linear files
  - Sequence of records with a fixed format usually stored on a single file
  - Limitation: single file
  - Example query: Salesperson='Mary' AND Price>100
- Hierarchical structure
  - Trees of records: one-to-many relationships
  - Limitations:
    - Requires duplicating records (e.g. many-to-many relationship)
    - Problems when updated
    - Retrieval requires knowing the structure (limited data independence): traversing the tree from top to bottom using a procedural language
- Network structure: similar to the hierarchical database with the implementation of many-to-many relationships
- Relational structure
- Object-Oriented structure
  - Objects (collection of data items and procedures) and interactions between
  - Is this really a new paradigm, or a special case of network structure?
  - Separate implementation vs. implementation on top of a RDBMS

Relational structure

- Relations, attributes, tuples
- Primary key (unique combination of attributes for each tuple)
- Foreign keys: relationships between tuples (many-to-many).

  Example: SUPPLIES defines relations between ITEM and SUPPLIER tuples.
- Advantages: many-to-many relationships, high level declarative query language (e.g. S
- SQL example (retrieve all items supplied by a supplier located in Troy):

```
SELECT ItemName
FROM ITEM, SUPPLIES, SUPPLIER
WHERE SUPPLIER.City = "Troy" AND
    SUPPLIER.Supplier# = SUPPLIES.Supplier# AND
    SUPPLIES.Item# = ITEM.Item#
```

- Programming language interfaces: including SQL queries in the code

Retrieving and manipulating data: query processing

- Parsing and validating a query: data dictionary - a relation listing all relations and relations listing the attributes
- Plans for computing the query: list of possible way to execute the query, estimated cost for each. Example:

```
SELECT ItemNames, Price
FROM ITEM, SALES
WHERE SALES.Item# = ITEM.Item# AND Salesperson="Mary"
```

- Index: B-tree index, drawbacks - additional space, updating; indexing not all relations (e.g. the keys only)
- Estimating the cost for computing a query: size of the relation, existence/size of the in
  Example: estimating Attribute=value with a given number of tuples and the size of the
- Query optimization: finding the best plan (minimizing the computational cost and the size of the intermediate results), subsets of tuples, projection and join.
- Static and dynamic optimization

Database views

- Creating user defined subsets of the database
- Improving the user interface
- Example:

```
CREATE VIEW MarySales(ItemName,Price)
AS SELECT ItemName, Price
FROM ITEM, SALES
WHERE ITEM.Item#=SALES.Item# AND Salesperson="Mary"
```

Then the query:

```
SELECT ItemName
FROM MarySales
WHERE Proce>100
```

translates to:

```
SELECT ItemName
FROM ITEM, SALES
WHERE ITEM.Item#=SALES.Item# AND Salesperson="Mary" AND Price>100
```

Data integrity

Integrity constraints: semantic conditions on the data

- Individual constraints on data items
- Uniqueness of the primary keys
- Dependencies between relations

Concurrency control

- Steps in executing a query
- Concurrent users of the database, interfering the execution of one query by another
- Transaction: a set of operations that takes the database from one consistent state to another
- Solving the concurrency control problem: making transactions atomic operations (one at a tim
- Concurrent transactions: serializability theory (two-phase locking), read lock (many), write lock
- Serializible transactions: first phase - accumulating locks, second phase - releasing locks.
- Deadlocks: deadlock detection algorithms.
- Distributed execution problems:
    - release a lock at one node (all locks accumulated at the other node?)
    - strict two-phase locking

The Transaction Model

Primitive	Description
BEGIN_TRANSACTION	Make the start of a transaction
END_TRANSACTION	Terminate the transaction and try to commit
ABORT_TRANSACTION	Kill the transaction and restore the old values
READ	Read data from a file, a table, or otherwise
WRITE	Write data to a file, a table, or otherwise

- Examples of primitives for transactions.

The Transaction Model

```
BEGIN_TRANSACTION
  reserve WP -> JFK;
  reserve JFK -> Nairobi;
  reserve Nairobi -> Malindi;
END_TRANSACTION
```

(a)

```
BEGIN_TRANSACTION
  reserve WP -> JFK;
  reserve JFK -> Nairobi;
  reserve Nairobi -> Malindi full =>
ABORT_TRANSACTION
```

(b)

a) Transaction to reserve three flights commits
b) Transaction aborts when third flight is unavailable

Data integrity

Backup and recovery

- The problem of keeping a transaction atomic: successful or failed

  What if some of the intermediate steps failed?
- Log of database activity: use the log to undo a failed transaction.
- More problems: when to write the log, failure of the recovery system executing the log.

Security and access control

- Access rules for relations or attributes. Stored in a special relation (part of the data diction
- Content-independent and content-dependent access control
- Content-dependent control: access to a view only or query modification

  (e.g. and-ing a predicate to the WHERE clause)
- Discretionary and mandatory access control

Knowledge Bases and KBS (and area of AI)

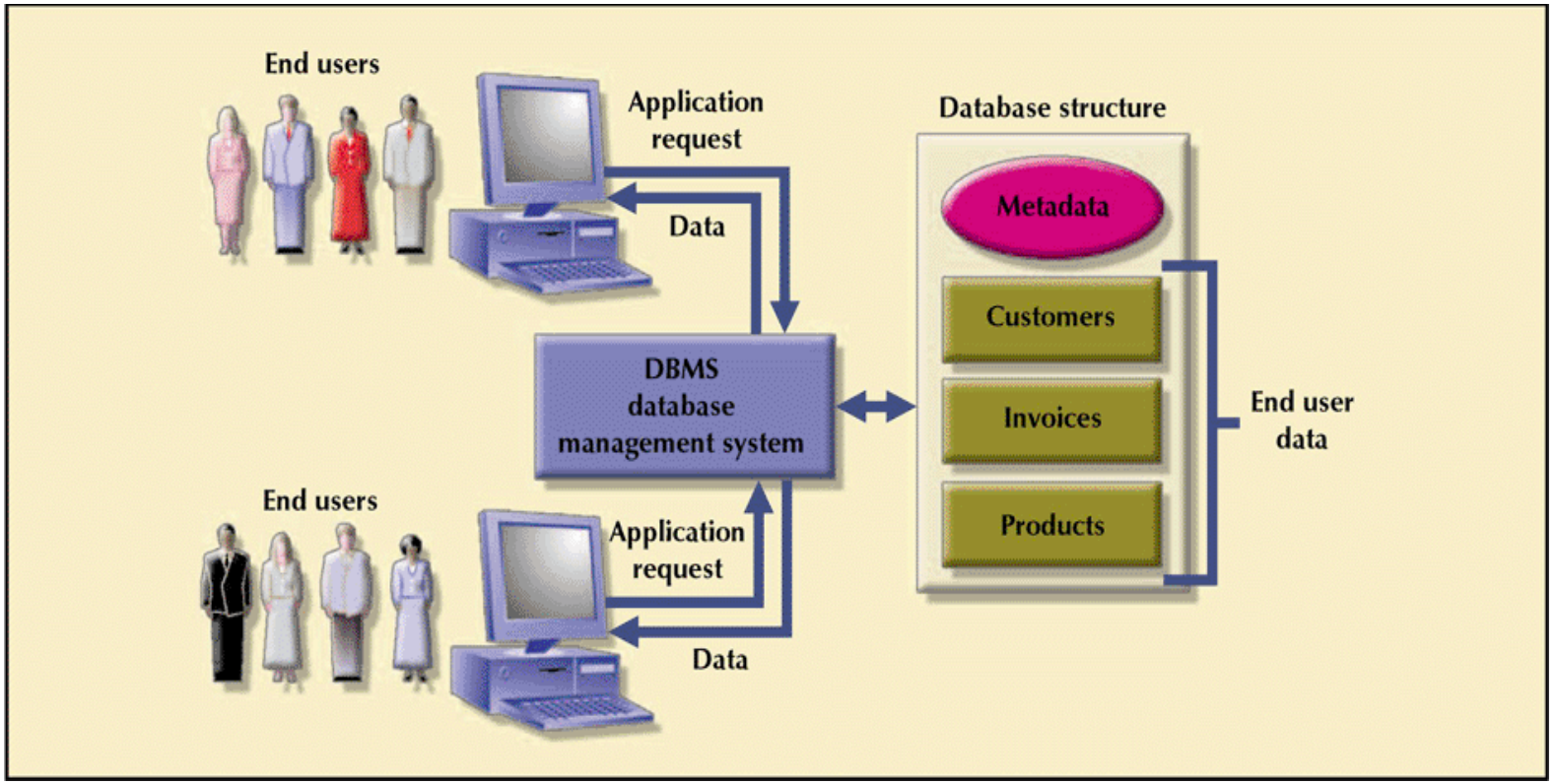
- Information, Data, Knowledge (data in a form that allows reasoning)
- Basic components of a KBS
    - Knowledge base
    - Inference (reasoning) mechanism (e.g. forward/backward chaining)
    - Explanation mechanism/Interface
- Rule-based systems (medical diagnostics, credit evaluation etc.)

Database: What

- Database
  - is collection of related data and its metadata organized in a structured format
  - for optimized information management
- Database Management System (DBMS)
  - is a software that enables easy creation, access, and modification of databases
  - for efficient and effective database management
- Database System
  - is an integrated system of hardware, software, people, procedures, and data
  - that define and regulate the collection, storage, management, and use of data within a database environment

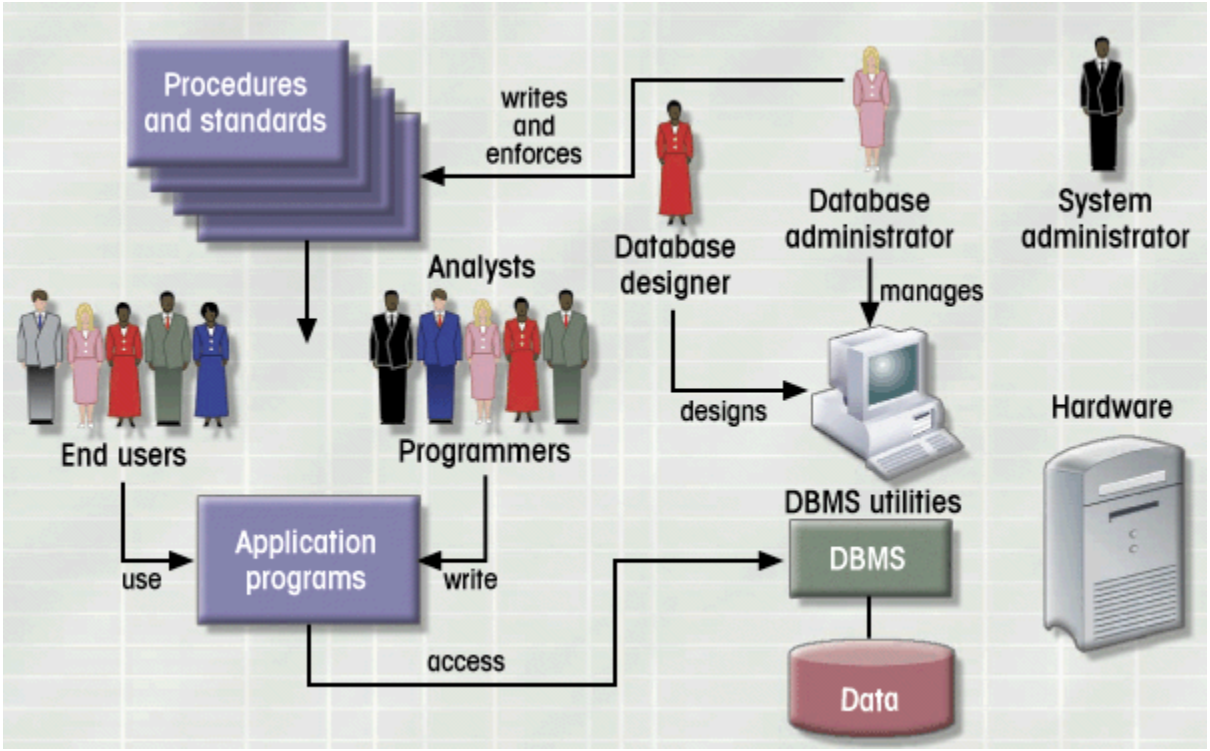
Database Management System

- manages interaction between end users and database



Database Systems: Design, Implementation, & Management: Rob & Coronel

Database System Environment



- **Hardware**
- **Software**
  - **OS**
  - **DBMS**
  - **Applications**
- **People**
- **Procedures**
- **Data**

Database Systems: Design, Implementation, & Management: Rob & Coronel

Database: Why

- Purpose of Database
  - Optimizes data management
  - Transforms data into information
- Importance of Database Design
  - Defines the database's expected use
    - different approach needed for different types of databases
  - Avoid data redundancy & ensure data integrity
    - data is accurate and verifiable
  - Poorly designed database generates errors
    - leads to bad decisions
    - can lead to failure of organization
- Functions of DBMS/Database System
  - Stores data and related data entry forms, report definitions, etc.
  - Hides the complexities of relational database model from the user
    - facilitates the construction/definition of data elements and their relationships
    - enables data transformation and presentation
  - Enforces data integrity
  - Implements data security management
    - access, privacy, backup & restoration

Database: How

- Planning & Analysis
  - Assess
    - Goal of the organization
    - Database environment
      - existing hardware, software, raw data, data processing procedures
  - Identify
    - Database needs
      - what database can do to further the goal of the organization
    - User needs and characteristics
      - who the users are, what they want to do, how they envision doing it
    - Database system requirements
      - what the database system should do to satisfy the database and user needs
- Design
  - From conceptual design to a detailed system specification
- Implementation
  - Create the database
- Maintenance
  - Troubleshoot, update, streamline the database

Business Rules

- What
  - Brief, precise, and unambiguous descriptions of operations in an organization
    - based on policies, procedures, or principles within a specific organization
    - help to create and enforce actions within that organization's environment
    - apply to any organization that stores and uses data to generate information
- Why
  - Enhance understanding & facilitate communication
    - Standardize company's view of data
    - Constitute a communications tool between users and designers
    - Allow designer to understand business process as well as the nature, role, and scope of data
  - Promote creation of an accurate data model
- How (sources)
  - Interviews
    - Company managers
    - Policy makers
    - Department managers
    - End users
  - Written documentation
    - Procedures, Standards, Operations manuals
  - Observation
    - Business operations

Database: User-centered

- Perspective
  - The user is always right. If there is a problem with the use of the system, the system is the problem, not the user.
- Compliance
  - The user has the right to a system that performs exactly as promised.
- Instruction
  - The user has the right to easy-to-use instructions (user guides, online or contextual help, error messages) for understanding and utilizing a system to achieve desired goals and recover efficiently and gracefully from problem situations.
- Usability
  - The user should be the master of software and hardware technology, not vice-versa. Products should be natural and intuitive to use.

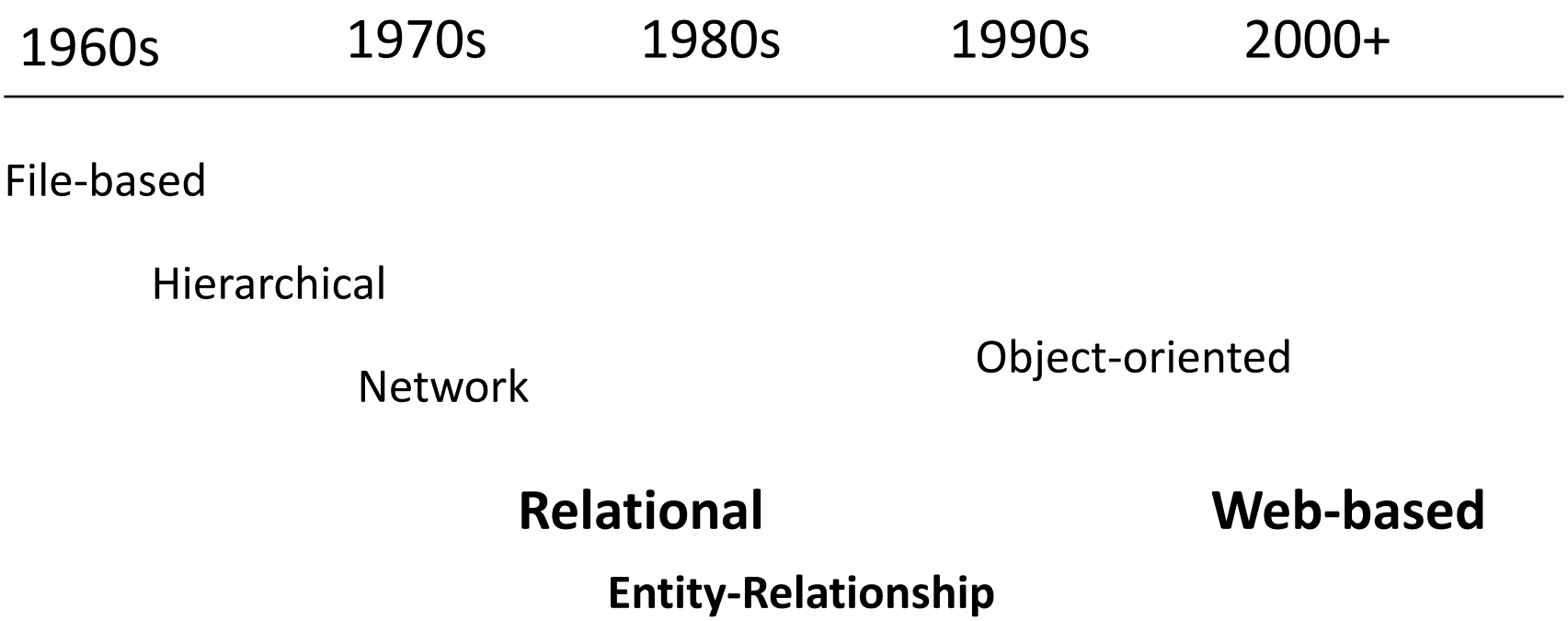
Database: Data Models

- Importance
  - Abstraction of complex real-word data structures in relative simple (graphical) representations
  - Facilitate interaction among the designer, the applications programmer, and the end user

- Basic Building Blocks
  - Entity
    - thing about which data are to be collected and stored
  - Attribute
    - a characteristic of an entity
  - Relationship
    - describes an association among entities
  - Constraint
    - restrictions placed on the data

Evolution of Data Models

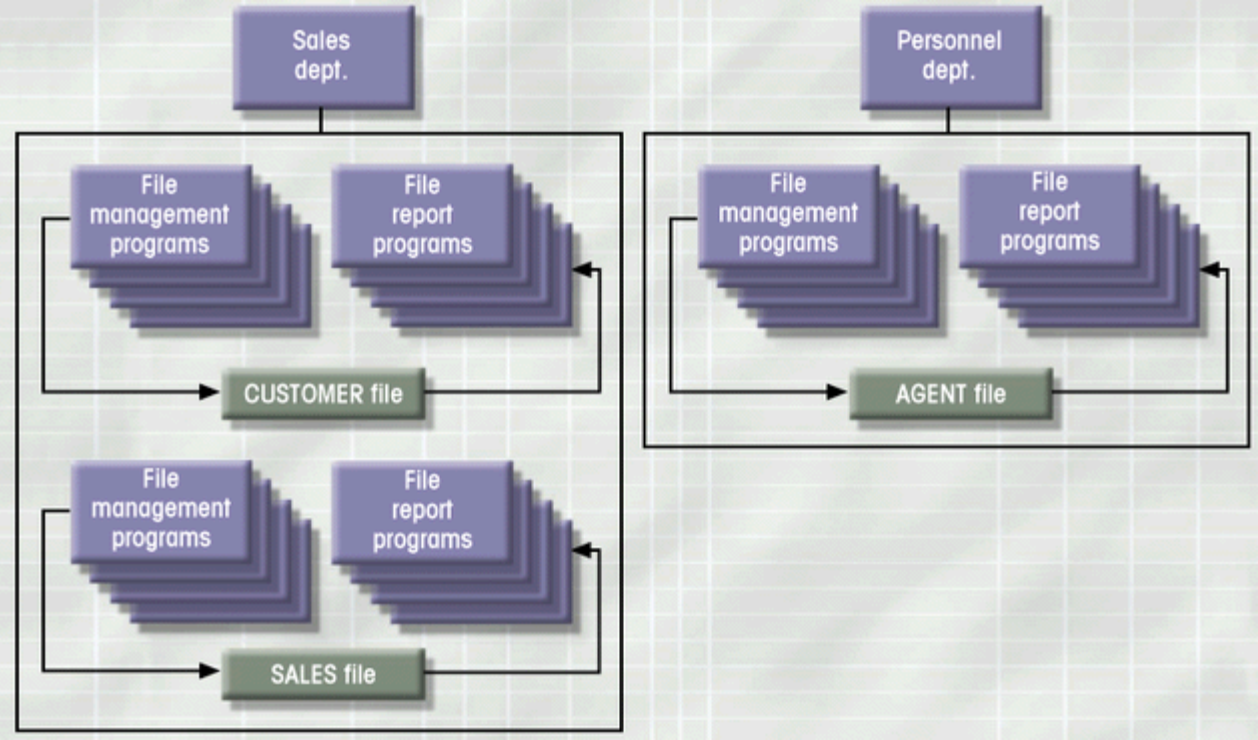
- Timeline



Database: Historical Roots

- Manual File System
  - to keep track of data
  - used tagged file folders in a filing cabinet
  - organized according to expected use
    - e.g. file per customer
  - easy to create, but hard to
    - locate data
    - aggregate/summarize data
- Computerized File System
  - to accommodate the data growth and information need
  - manual file system structures were duplicated in the computer
  - Data Processing (DP) specialists wrote customized programs to
    - write, delete, update data (i.e. management)
    - extract and present data in various formats (i.e. report)

File System: Example

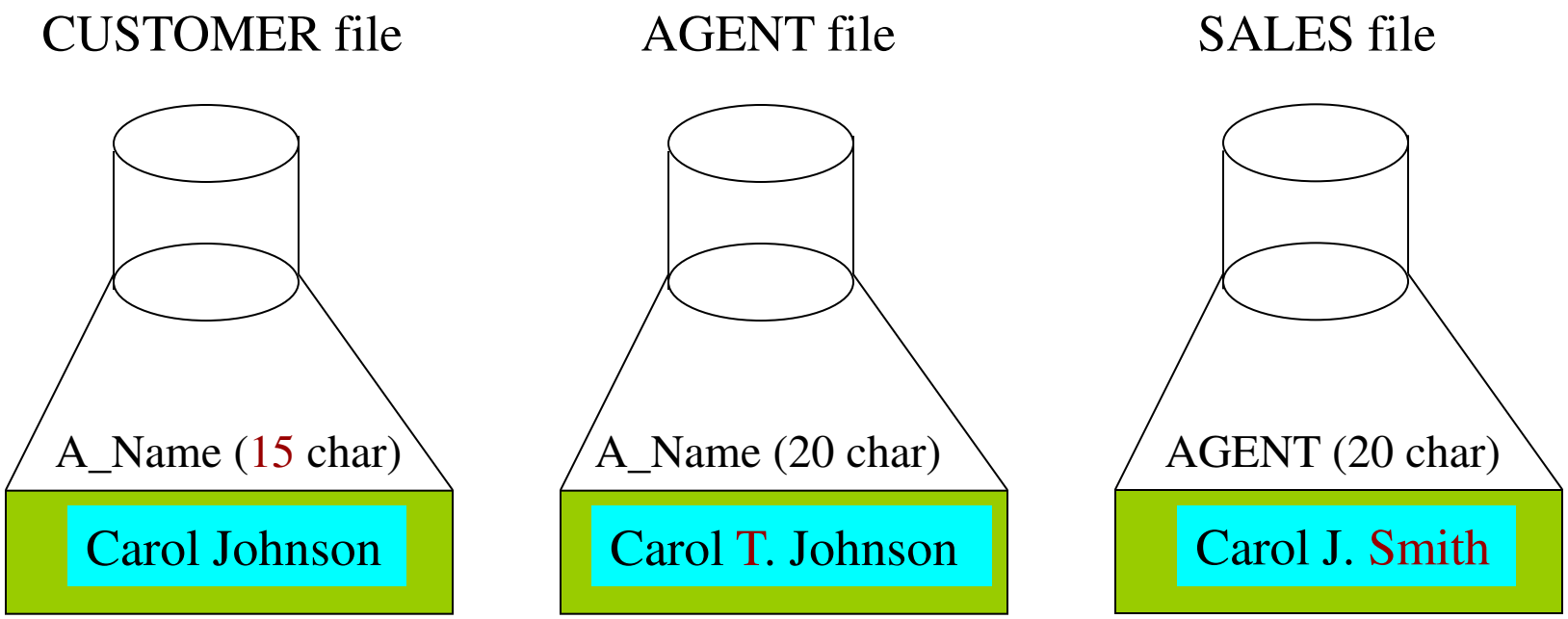


Database Systems: Design, Implementation, & Management: Rob & Coronel

File System: Weakness

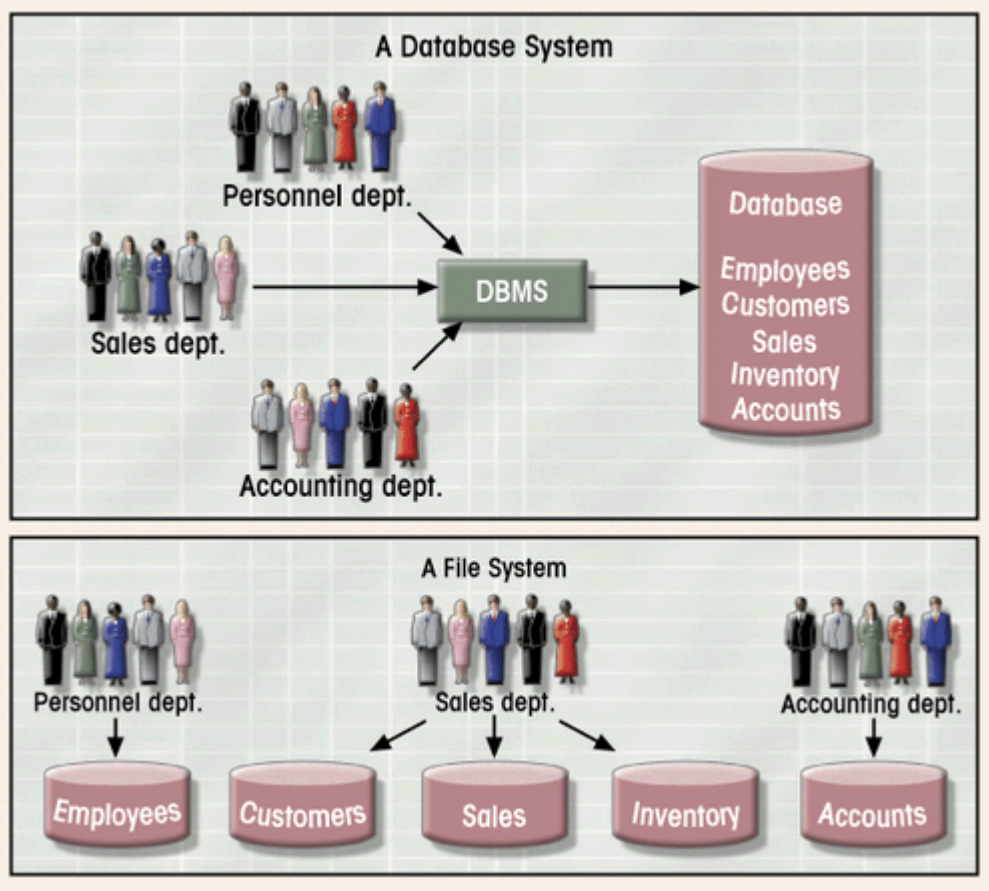
- Weakness
  - "Islands of data" in scattered file systems.
- Problems
  - Duplication
    - same data may be stored in multiple files
  - Inconsistency
    - same data may be stored by different names in different format
  - Rigidity
    - requires customized programming to implement any changes
    - cannot do ad-hoc queries
- Implications
  - Waste of space
  - Data inaccuracies
  - High overhead of data manipulation and maintenance

File System: Problem Case



- inconsistent field name, field size
- inconsistent data values
- data duplication

Database System vs. File System

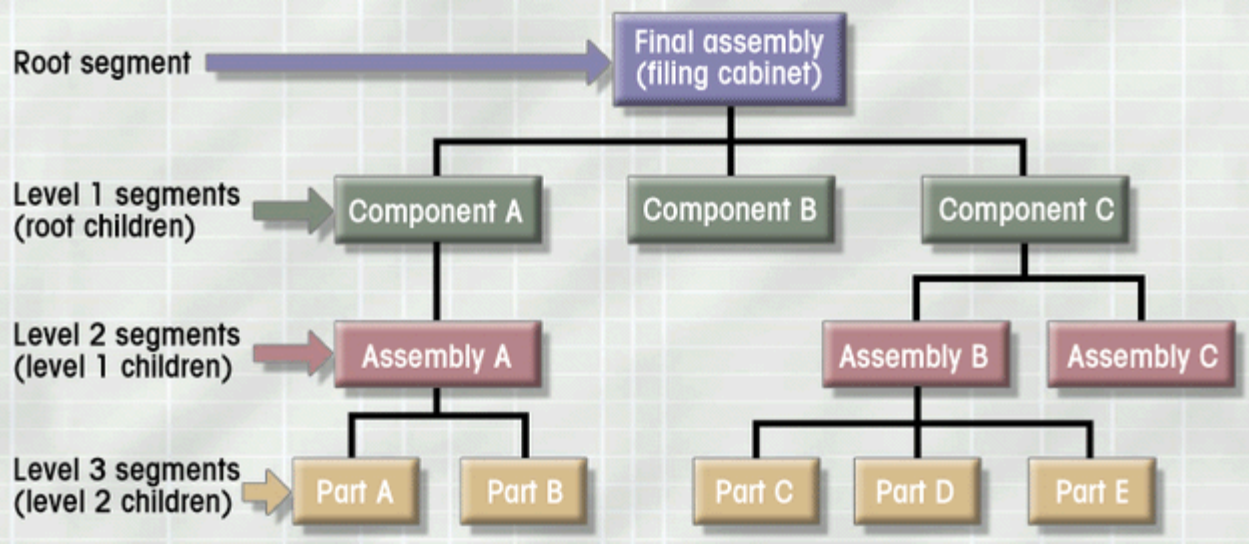


Database Systems: Design, Implementation, & Management: Rob & Coronel

Hierarchical Database

- Background
  - Developed to manage large amount of data for complex manufacturing projects
  - e.g., Information Management System (IMS)
    - IBM-Rockwell joint venture
    - clustered related data together
    - hierarchically associated data clusters using pointers
- Hierarchical Database Model
  - Assumes data relationships are hierarchical
    - One-to-Many (1:M) relationships
      - Each parent can have many children
      - Each child has only one parent
  - Logically represented by an upside down tree

Hierarchical Database: Example



Database Systems: Design, Implementation, & Management: Rob & Coronel

Hierarchical Database: Pros & Cons

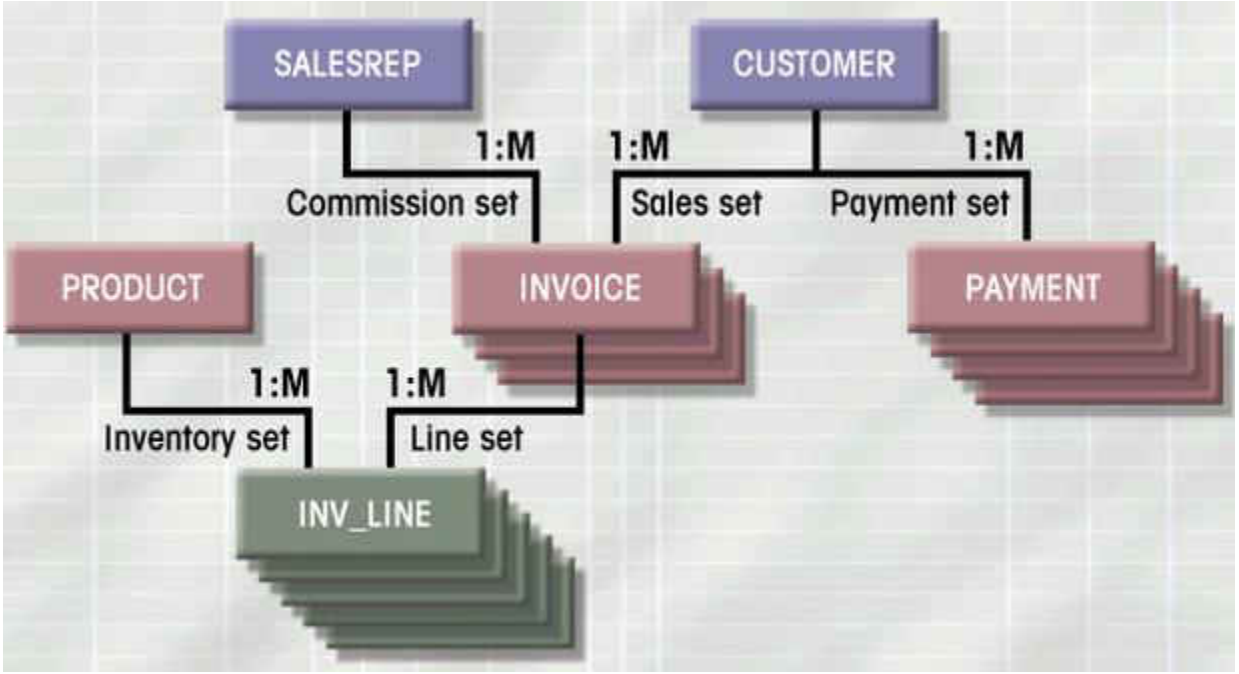
- Advantages
  - Conceptual simplicity
    - groups of data could be related to each other
    - related data could be viewed together
  - Centralization of data
    - reduced redundancy and promoted consistency
- Disadvantages
  - Limited representation of data relationships
    - did not allow Many-to-Many (M:N) relations
  - Complex implementation
    - required in-depth knowledge of physical data storage
  - Structural Dependence
    - data access requires physical storage path
  - Lack of Standards
    - limited portability

Network Database

- Objectives
  - Represent more complex data relationships
  - Improve database performance
  - Impose a database standard

- Network Database Model
  - Similar to Hierarchical Model
    - Records linked by pointers
  - Composed of sets
    - Each set consists of owner (parent) and member (child)
  - Many-to-Many (M:N) relationships representation
    - Each owner can have multiple members (1:M)
    - A member may have several owners

Network Database: Example



Database Systems: Design, Implementation, & Management: Rob & Coronel

Network Database: Pros & Cons

- Advantages
  - More data relationship types
  - More efficient and flexible data access
    - "network" vs. "tree" path traversal
  - Conformance to standards
    - enhanced database administration and portability
- Disadvantages
  - System complexity
    - require familiarity with the internal structure for data access
  - Lack of structural independence
    - small structural changes require significant program changes

Relational Database

- Problems with legacy database systems
  - Required excessive effort to maintain
    - Data manipulation (programs) too dependent on physical file structure
  - Hard to manipulate by end-users
    - No capacity for ad-hoc query (must rely on DB programmers).
- Evolution in Data Organization
  - E. F. Codd's Relational Model proposal
    - Separated the notion of physical representation (machine-view) from logical representation (human-view)
    - Considered ingenious but computationally impractical in 1970
  - Relational Database Model
    - Dominant database model of today
    - Eliminated pointers and used tables to represent data
    - Tables
      - flexible logical structure for data representation
      - a series of row/column intersections
      - related by sharing common entity characteristic(s)

Relational Database: Example

- Provides a logical "human-level" view of the data and associations among groups of data (i.e., tables)

Customer_ID	Customer_Account	Agent_ID
1224	4556	23
1225	4558	25

Agent_ID	Last_Name	First_Name	Phone
23	Sturm	David	334-5678
25	Long	Kyle	556-3421

Customer_ID	Last_Name	First_Name	Phone	Account_Balance
1224	Vira	Dyne	678-9987	1223.95
1225	Davies	Tricia	556-3342	234.25

Relational Database: Pros & Cons

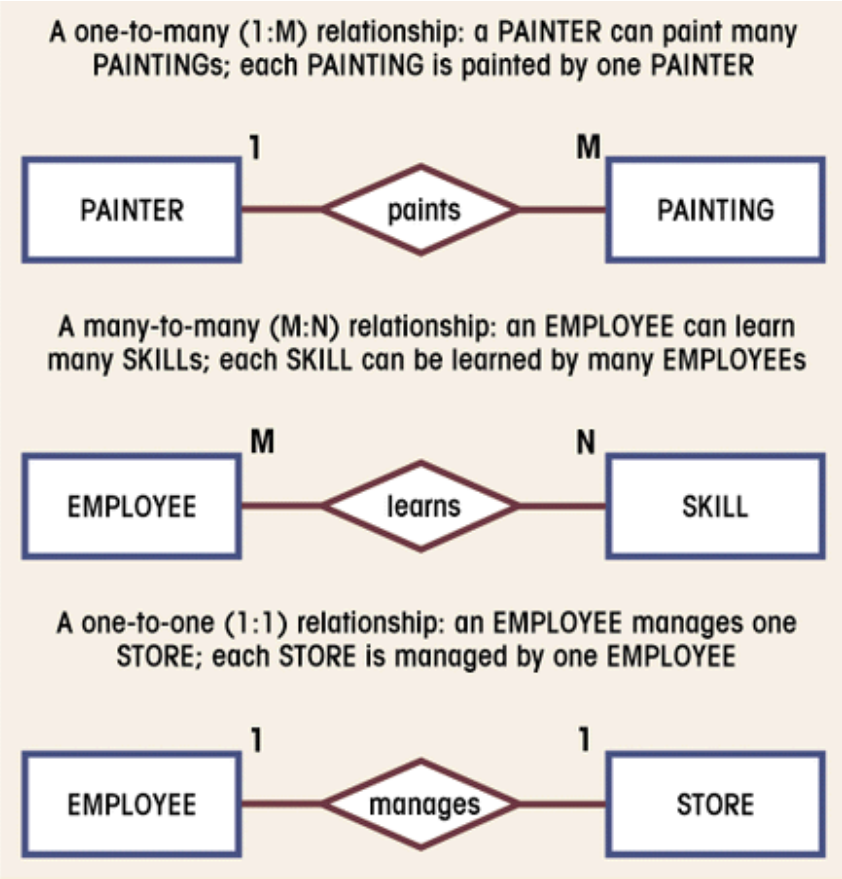
- Advantages
  - Structural independence
    - Separation of database design and physical data storage/access
    - Easier database design, implementation, management, and use
  - Ad hoc query capability with Structured Query Language (SQL)
    - SQL translates user queries to codes
- Disadvantages
  - Substantial hardware and system software overhead
    - more complex system
  - Poor design and implementation is made easy
    - ease-of-use allows careless use of RDBMS

Entity Relationship Model

- Peter Chen's Landmark Paper in 1976
  - "The Relationship Model: Toward a Unified View of Data"
  - Graphical representation of entities and their relationships

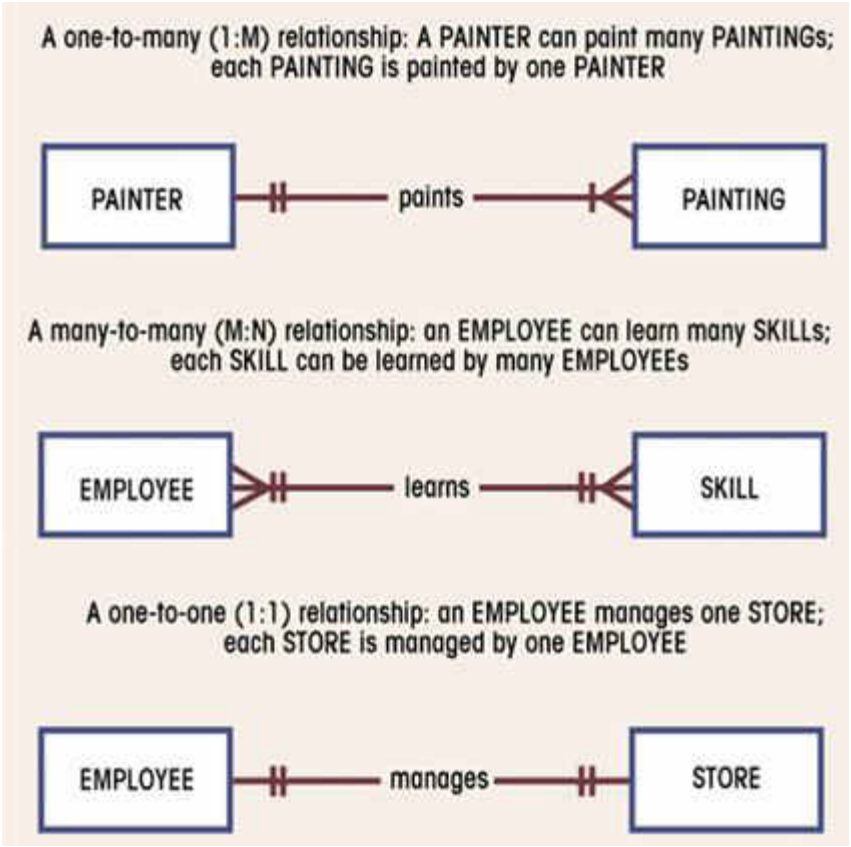
- Entity Relationship (ER) Model
  - Based on Entity, Attributes & Relationships
    - Entity is a thing about which data are to be collected and stored
      - e.g. EMPLOYEE
    - Attributes are characteristics of the entity
      - e.g. SSN, last name, first name
    - Relationships describe an associations between entities
      - i.e. 1:M, M:N, 1:1
  - Complements the relational data model concepts
    - Helps to visualize structure and content of data groups
      - entity is mapped to a relational table
    - Tool for conceptual data modeling (higher level representation)
  - Represented in an Entity Relationship Diagram (ERD)
    - Formalizes a way to describe relationships between groups of data

E-R Diagram: Chen Model



- Entity
  - represented by a rectangle with its name in capital letters.
- Relationships
  - represented by an active or passive verb inside the diamond that connects the related entities.
- Connectivities
  - i.e., types of relationship
  - written next to each entity box.

E-R Diagram: Crow's Foot Model



- Entity
  - represented by a rectangle with its name in capital letters.
- Relationships
  - represented by an active or passive verb that connects the related entities.
- Connectivities
  - indicated by symbols next to entities.
    - 2 vertical lines for 1
    - "crow's foot" for M

E-R Model: Pros & Cons

- Advantages
  - Exceptional conceptual simplicity
    - easily viewed and understood representation of database
    - facilitates database design and management
  - Integration with the relational database model
    - enables better database design via conceptual modeling
- Disadvantages
  - Incomplete model on its own
    - Limited representational power
      - cannot model data constraints not tied to entity relationships
        - » e.g. attribute constraints
      - cannot represent relationships between attributes within entities
    - No data manipulation language (e.g. SQL)
  - Loss of information content
    - Hard to include attributes in ERD

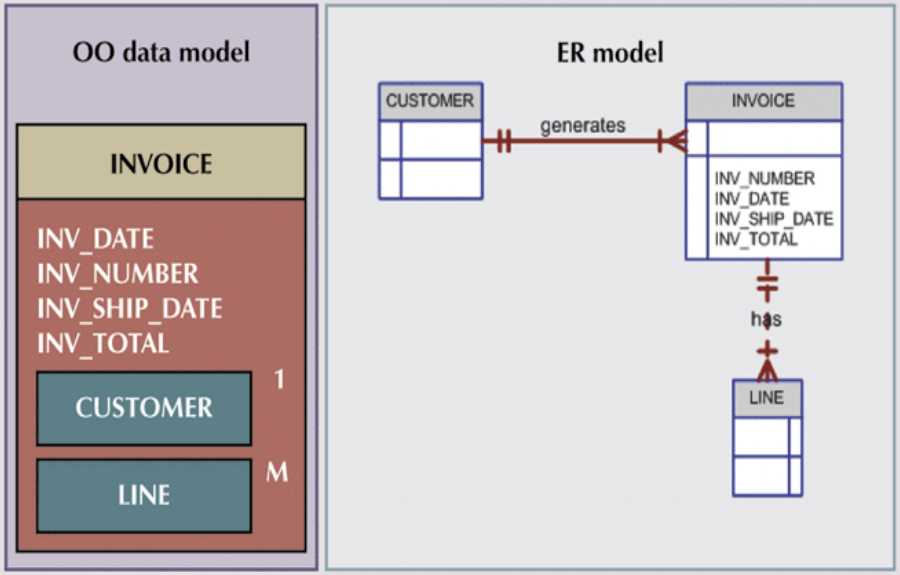
Object-Oriented Database

- Semantic Data Model (SDM)
  - Modeled both data and their relationships in a single structure (object)
    - Developed by Hammer & McLeod in 1981
- Object-oriented concepts became popular in 1990s
  - Modularity facilitated program reuse and construction of complex structures
  - Ability to handle complex data types (e.g. multimedia data)
- Object-Oriented Database Model (OODBM)
  - Maintains the advantages of the ER model but adds more features
  - Object = entity + relationships (between & within entity)
    - consists of attributes & methods
      - attributes describe properties of an object
      - methods are all relevant operations that can be performed on an object
    - self-contained abstraction of real-world entity
  - Class = collection of similar objects with shared attributes and methods
    - e.g. EMPLOYEE class = (employ1 object, employ2 object, …)
    - organized in a class hierarchy
      - e.g. PERSON > EMPLOYEE, CUSTOMER
  - Incorporates the notion of inheritance
    - attributes and methods of a class are inherited by its descendent classes

OO Database Model vs. E-R Model

OODBM:
- can accommodate relationships within a object
- objects to be used as building blocks for autonomous structures

FIGURE 2.7 A comparison of the OO model and the ER model



Database Systems: Design, Implementation, & Management: Rob & Coronel

Object-Oriented Database: Pros & Cons

- Advantages
  - Semantic representation of data
    - fuller and more meaningful description of data via object
  - Modularity, reusability, inheritance
  - Ability to handle
    - complex data
    - sophisticated information requirements

- Disadvantages
  - Lack of standards
    - no standard data access method
  - Complex navigational data access
    - class hierarchy traversal
  - Steep learning curve
    - difficult to design and implement properly
  - More system-oriented than user-centered
  - High system overhead
    - slow transactions

Web Database

- Internet is emerging as a prime business tool
  - Shift away from models (e.g. relational vs. O-O)
  - Emphasis on interfacing with the Internet

- Characteristics of "Internet age" databases
  - Flexible, efficient, and secure Internet access
  - Support for complex data types & relationships
  - Seamless interfaces with multiple data sources and structures
  - Ease of use for end-user, database architect, and database administrator
    - Simplicity of conceptual database model
    - Many database design, implementation, and application development tools
    - Powerful DBMS GUI

Lab: Access Automations

- MS Access Automations
  - can save effort & time
  - may not suit your needs
  - Templates & Wizards
- Group Project
  - Project Team formation
  - Project Description