Data: Abstraction, Independence, & Modeling

Data Abstraction

Why it is Important?

- How is it provided by a DBMS?
- 3 levels of abstraction
  - Physical or Internal Level
  - Logical Level
  - View or External Level

Data Independence

- What is Data Independence?
- Why is it Important?
- How is it provided by a DBMS?
- Types of Independence
  - Physical Data Independence
  - Logical Data Independence

Data Modeling

- What is Data Modeling?
- An integrated collection of concepts for describing & manipulating data, relationships between data, & constraints on the data in an organization
- Used for defining Database Schemas
- Databases have several schemas, partitioned according to levels of abstraction
- Schema Levels
  - Physical Schema
  - Conceptual/Logical Schema
  - Sub-schemas or external schemas

Data Model

- A data model in software engineering is an abstract model that describes how data is represented and accessed
- Data models formally define data elements and relationships among data elements for a domain of interest
- If different data models are used for describing different systems, complex interfaces are required to share data among them

Popular Data Models

- Entity-Relationship Model
- Relational Model
- Hierarchical Model
- Network Model
- Inverted File Model
- Object-Oriented Model
- Object-Relational Model

Data Abstraction

- Major aim of a DBMS is to provide users with an abstract view of data
- Hides certain details of how the data are stored & maintained
- DBMS must retrieve data efficiently
- Need for efficiency has led designers to use complex data structures to represent the data in the database
- Most DB users are not computer trained, developers hide complexity through several levels of abstraction to simplify user's interaction with the systems

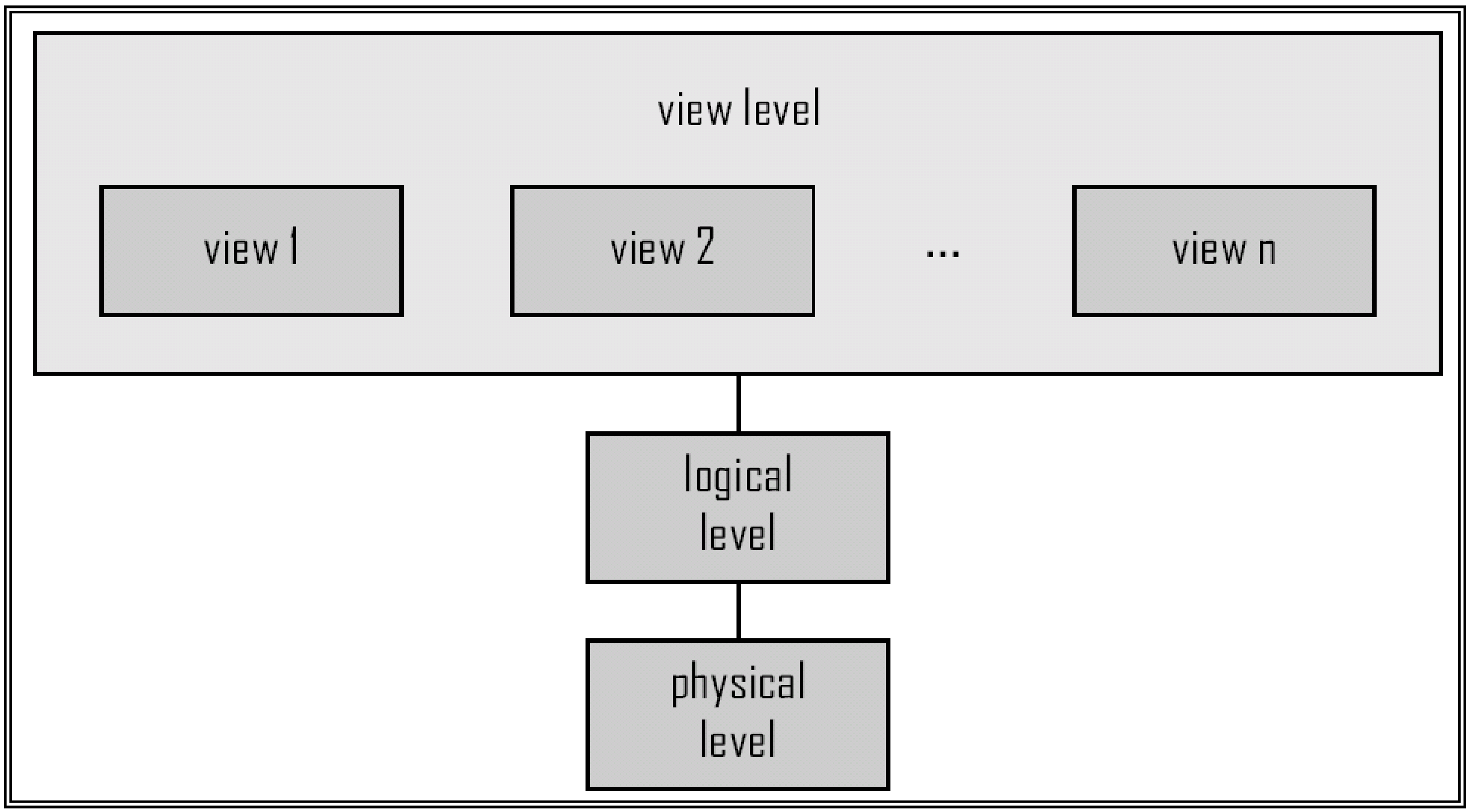
3 Levels of Abstraction

- Physical or Internal Level
  - Lowest level of abstraction describes how data are actually stored
  - Describes complex low-level data structures in detail
- Logical or Conceptual Level
  - Describes what data are stored in the DB & what relationships exist among those data
  - Describes the entire DB in terms of relatively simpler structures
- View or External Level
  - Highest level of abstraction which describes only a part of the DB
  - User’s view of the DB. This level describes part of the DB that is relevant to each user

3 Levels of Abstraction

- Logical or Conceptual Level
  - Describes what data are stored in the DB & what relationships exist among those data
  - Describes the entire DB in terms of relatively simpler structures
  - Implementation of these simple structures at this level may involve complex physical-level structures
  - Users of the logical level need not be aware of this complexity
  - DBAs, who decide what information to keep in DB, use the logical level of abstraction

Levels of Abstraction (1)



Levels of Abstraction (2)

- Many *views*, single *conceptual (logical) schema* and *physical schema*
  - Views describe how users see the data
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used

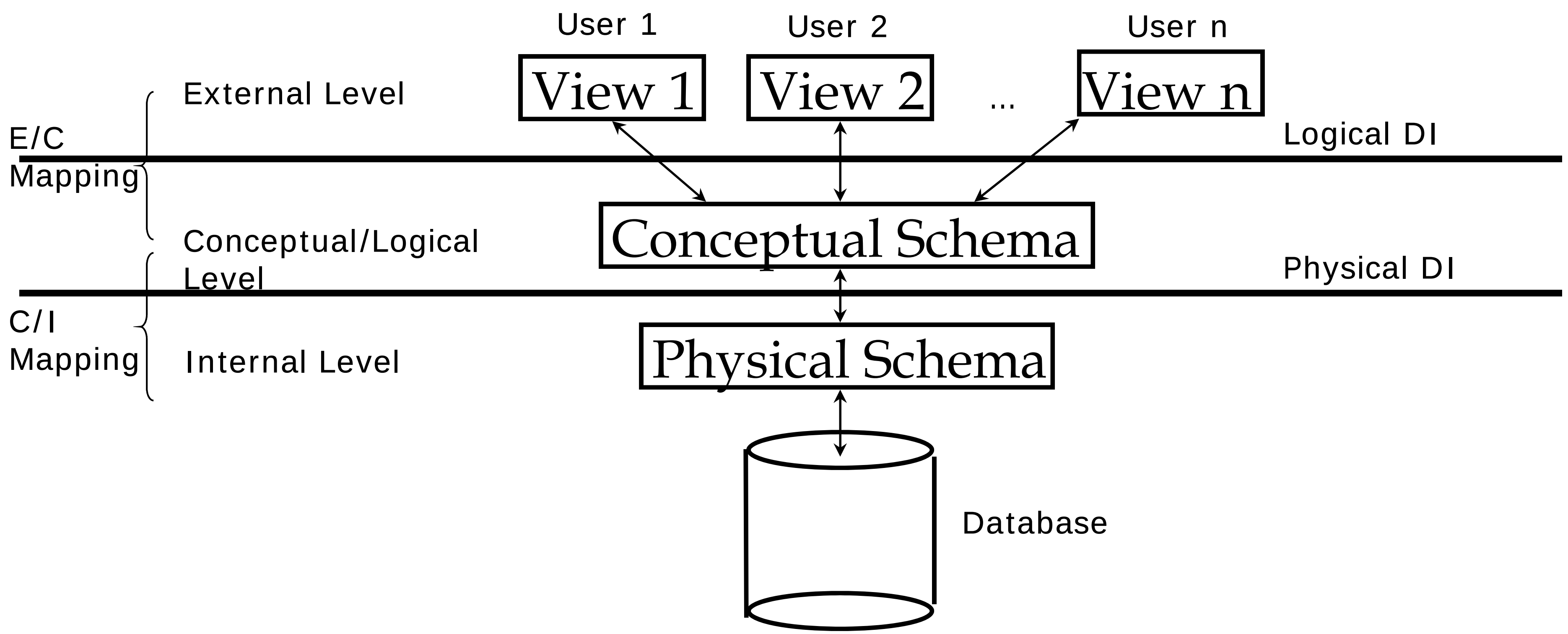
- Schemas are defined using DDL, data is modified/queried using DML

ANSI/SPARC 3-Tier Architecture (1)

- Proposal for standard terminology & general architecture for DBSs produced in 1971 by DBTG (Data Base Task Group) appointed by Conference on DBSs & Languages (CODASYL)
- DBTG recognized the need for a 2-tier architecture with system view (schema) & user view (subschema)
- ANSI (American National Standards Institute)-SPARC (Standards Planning & Requirements Committee) produced similar terminology & architecture in 1975(ANSI/X3/SPARC)* in 1975
- ANSI-SPARC recognized the need for a 3-tier architecture

**ANSI/X3/SPARC study group on DBMSs. Interim Report, FDT. ACM SIGMOD Bulletin, 7(2), 1975.*

ANSI/SPARC 3-Tier Architecture (2)



Levels of Abstraction

- Many *views*, single *conceptual (logical) schema* and *physical schema*
  - Views describe how users see the data
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used

- Schemas are defined using DDL, data is modified/queried using DML

Internal Level

- space allocation, data compression, encryption, access paths
- DBMS: Data access optimized and storage minimized

ANSI-SPARC Architecture (3)

- Separates users' view from the way in which the data is arranged physically or logically
- It hides the physical storage details from users: Users should not have to deal with physical database storage details. They should be allowed to work with the data itself, without concern for how it is physically stored
- The database administrator should be able to change the database storage structures without affecting the users' views : From time to time changes to the structure of an organisation's data will be required.
- The internal structure of the database should be unaffected by changes to the physical aspects of the storage : For example, a changeover to a new disk.

ANSI-SPARC Architecture (3)

- The database administrator should be able to change the conceptual or global structure of the database without affecting the users : This should be possible while still maintaining the desired individual users' views.
- Implementation of architecture using SQL updateable views was done in 1998 which uses triggers

Instances & Schemas (1)

- Collection of information stored in the DB at a particular moment is called an INSTANCE
- The overall design of the DB is called a SCHEMA
- A DB has many schemas
  - Physical
  - Conceptual/Logical
  - Sub-schemas
- DB design with requirements analysis
- Requirements of individual users are integrated into a single community view, called “conceptual schema”
- Represents “entities”, their “attributes”, & their “relationships”

Instances & Schemas (2)

- Conceptual design is independent of the DBMS, application programs, & physical considerations
- Conceptual schema is translated into a schema that is compatible with the chosen DBMS
- Relationships between entities as reflected in the conceptual schema may not be implementable with the chosen DBMS
- Version of the conceptual schema that can be presented to the DBMS is called the “Logical Schema”
- In a RDBMS, the logical schema describes all relations stored in the DB

Instances & Schemas (3)

- Users are presented with the subsets, called “subschemas”, of the logical schema
- Subschemas are also in terms of the data model of the DBMS
- Allow data access to be customized & authorized at the level of individual users or group of users
- Each subschema consists of a collection of one or more “views” & relations from the logical schema
- Logical schema is mapped to physical storage such as disk or tape

Example: University Database

- Logical schema:
  - *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
  - *Faculty(fid:string, fname:string, sal:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- External Schema (View):
  - *Course_info(cid:string,fname:string, enrollment:integer)*

Data Independence

- Major objective of the 3-tier architecture is to provide data independence (DI)
- Upper levels are unaffected by changes at the lower level
- Two kinds of DI:
  - Logical DI
  - Physical DI

Data Independence (1)

Logical DI

- Immunity of the external schemas to changes in the conceptual schema
- Addition or removal of entities, attributes, or relationships, should be possible without having to change the external schemas or having to rewrite the application programs

Data Independence (2)

Logical DI

Faculty(fid:string, fname:string, sal:real)

Faculty_public(fid:string, fname: string, office:integer)

Faculty_private(fid:string, sal: real)

View course_info can be redefined in terms of Faculty_public & Faculty_private so that users who queries course_info gets the same answer as before

Data Independence (3)

Physical DI

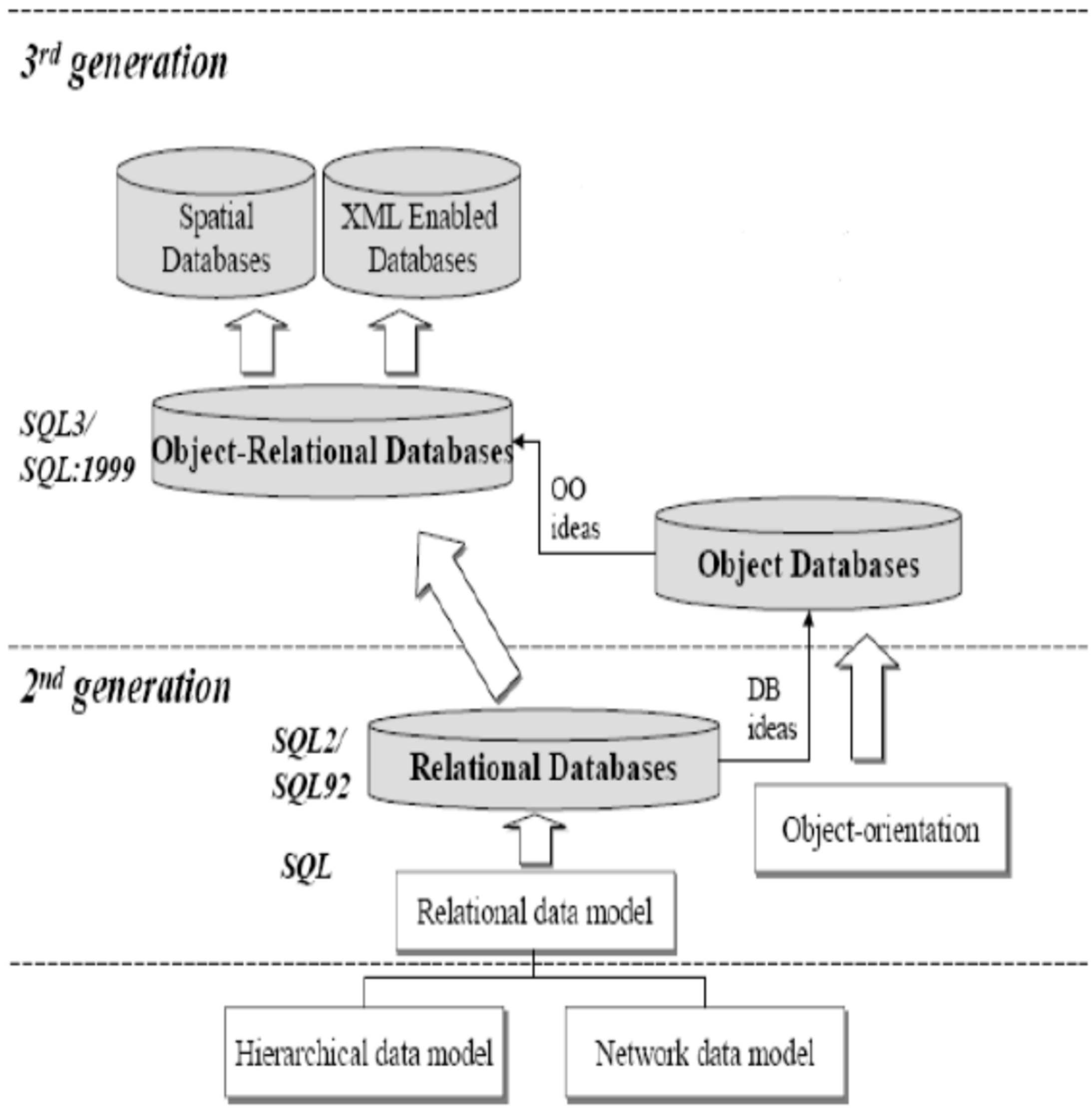
- Immunity of the conceptual schema to changes in the internal schema
- Using different file organizations or storage structures, using different storage devices, modifying indexes, or changing hashing algorithms should be possible without having to change the upper schemas
- Deterioration in performance is the most common reason for internal schema changes

Data Modeling

Three broad categories

- Object-based
  - Use concepts such as entities, attributes, & relationships
  - Entity-relationship Model
  - Object-oriented Model
- Record-based
  - DB consists of fixed format records of different types
  - Each record has a fixed number of fields, each typically of fixed length
  - Relational, Hierarchical, & Network
- Physical

Evolution of DBMS



Flat Data Model (before 1960)

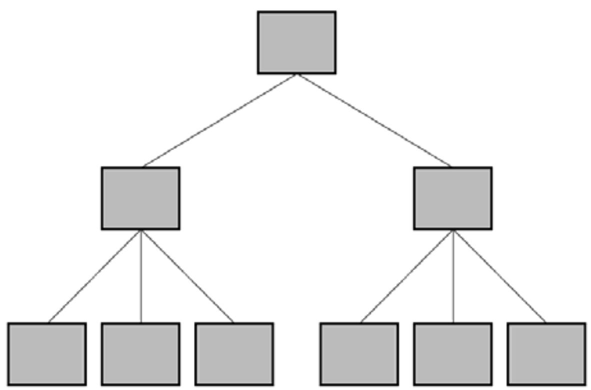
- 2-D array of data elements

Flat File Model

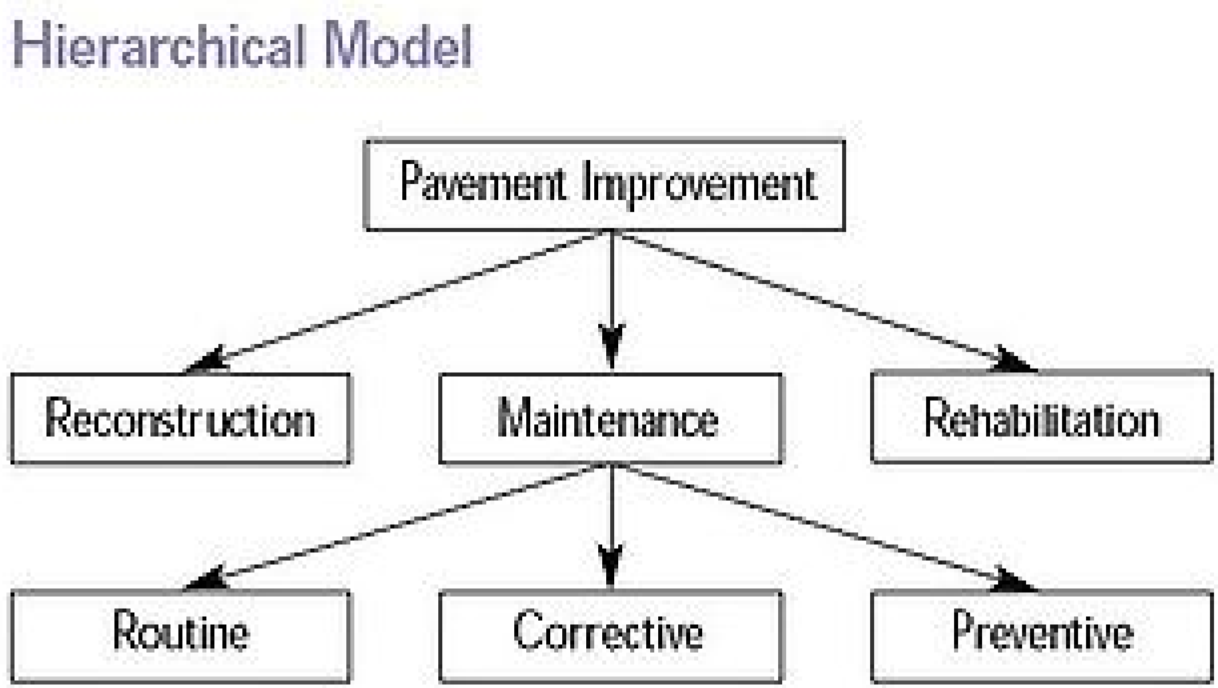
	Route No.	Miles	Activity
Record 1	I-95	12	Overlay
Record 2	I-495	05	Patching
Record 3	SR-301	33	Crack seal

Hierarchical Model (1960)

- Data is organized into tree like structure
- Links describe nesting
- Used in IBM's IMS DBMS

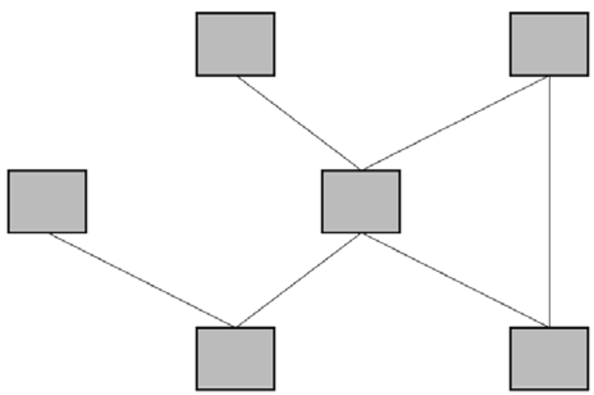


Example

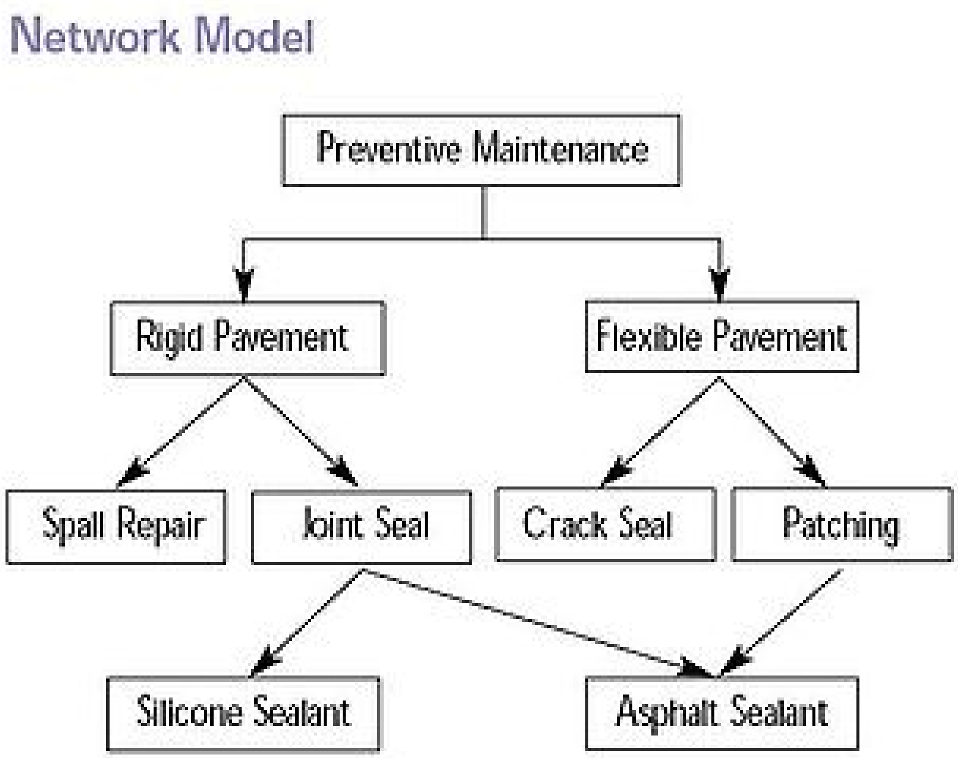

Hierarchical Model
Pavement Improvement
Reconstruction
Maintenance
Rehabilitation
Routine
Corrective
Preventive

Network Model (1970)

- Each record can have multiple parents and child forming lattice structure
- Better than hierarchical model
- IDMS (Integrated DMS)



Example



Network Model
Preventive Maintenance
Rigid Pavement
Flexible Pavement
Spall Repair
Joint Seal
Crack Seal
Patching
Silicone Sealant
Asphalt Sealant

Relational Model

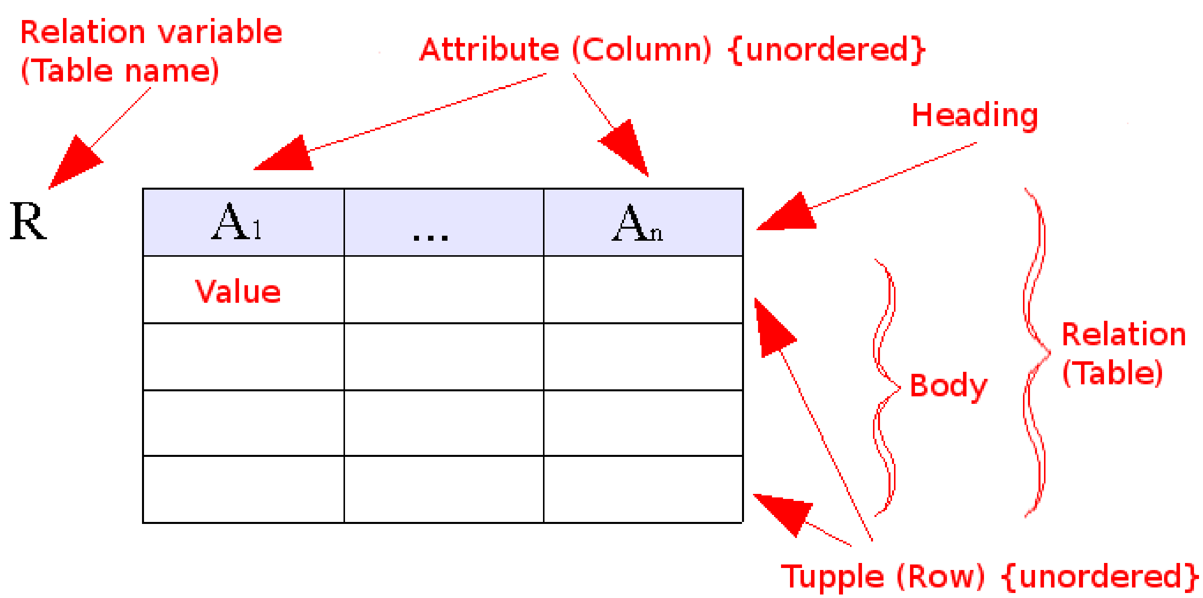
- Relational Model (E.F.Codd 1969-70)*
  - Relation is a table with rows and columns
  - Every relation has a schema that describes the columns or fields
- Relational model is good for:
  - Large amounts of data —> simple operations
  - Navigate among small number of relations
  - Difficult Applications for relational model
    - VLSI Design (CAD in general)
    - CASE
    - Graphical Data

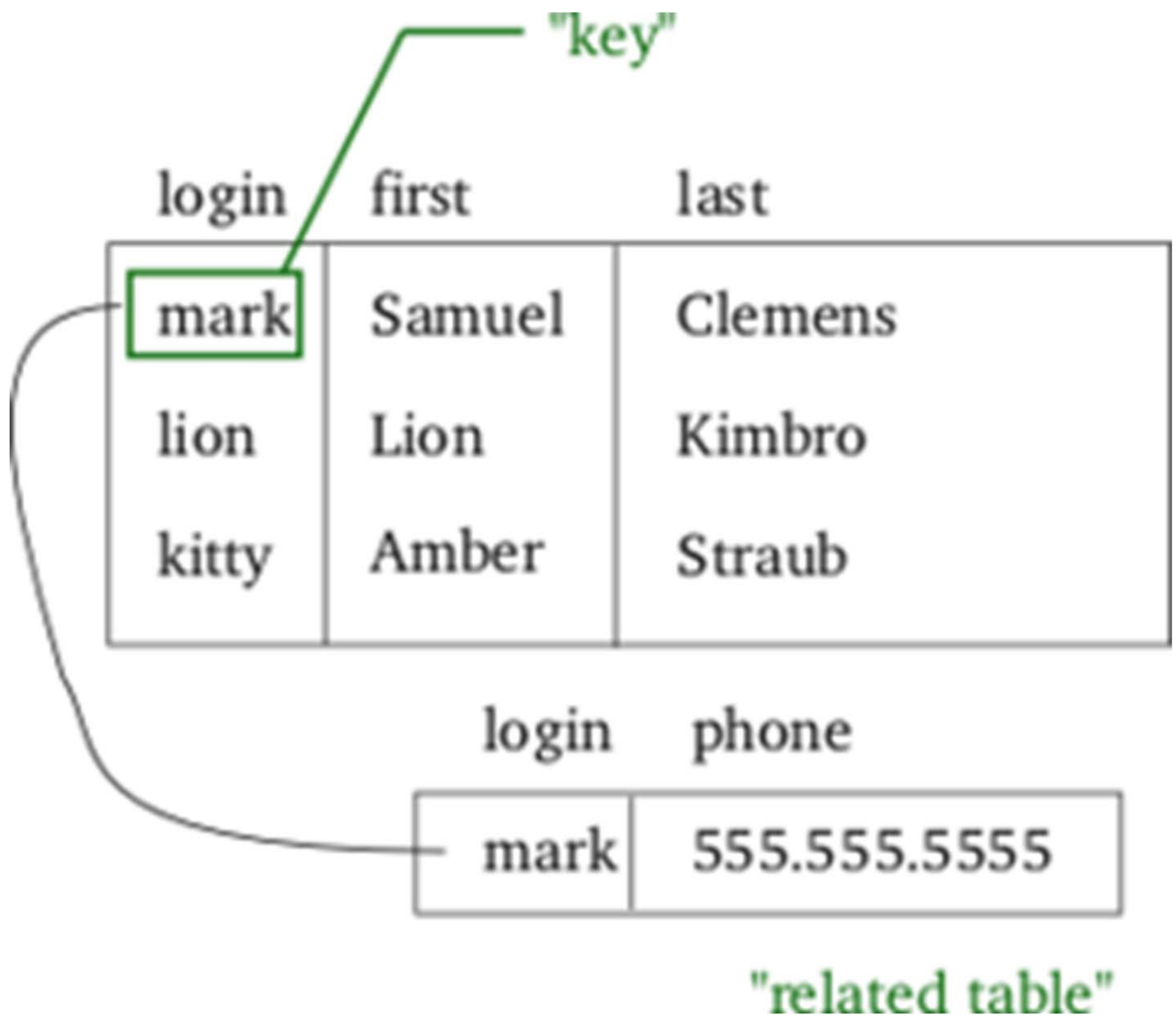
*_E.F. Codd (1970). A relational model of data for large shared data banks, Communications of the ACM, Vol 13. Issue 6(June 1970). pp.377-387_

Turing Award 1981 for his fundamental and continuing contributions to the theory and practice of database management system (Relational databases)

Relational Model (1980)

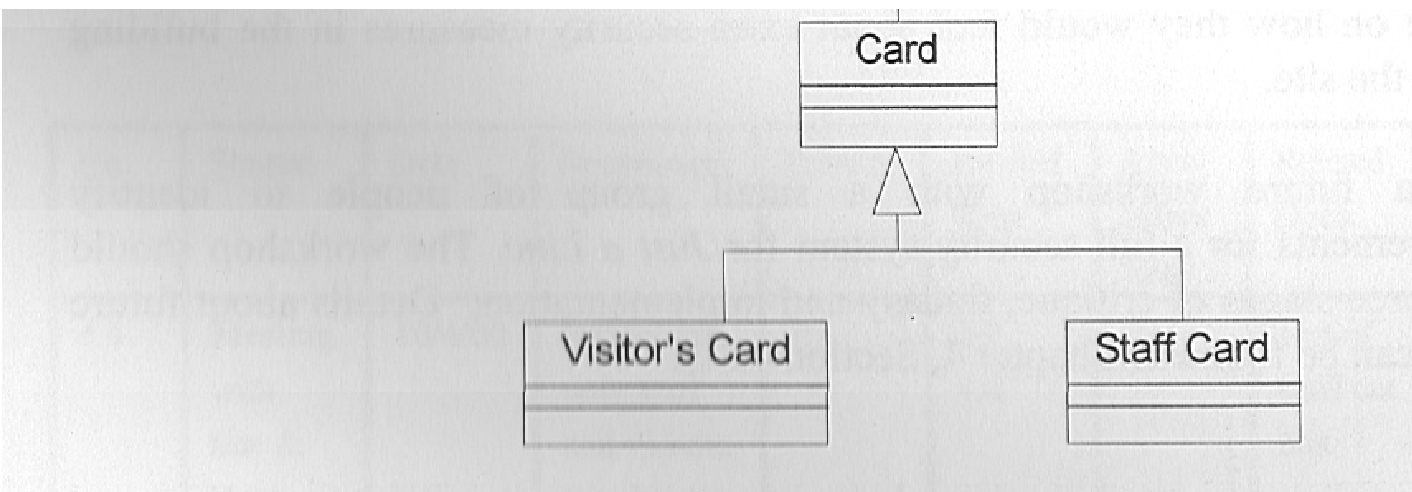
- Data as set of tables
- Having constraints and relationships
- Oracle, DB2,Sybase




"key"
login
first
last
mark
Samuel
Clemens
lion
Lion
Kimbro
kitty
Amber
Straub
login
phone
mark
555.555.5555
"related table"

Object-Oriented & Object-Relational Data Model (1990)

- Entities as objects, Grouped together as classes
- Every class knows things, able to do certain things
- Classes related to each other, Communicate through message passing
- Complex SQL
- UML is a language used to model data using OO model
- Oracle9i, O2
- ObjectStore, Versant



Object-Oriented Model

Object 1: Maintenance Report

Date	
Activity Code	
Route No.	
Daily Production	
Equipment Hours	
Labor Hours	

Object 1 Instance

01-12-01
24
I-95
2.5
6.0
6.0

Object 2: Maintenance Activity

Activity Code	
Activity Name	
Production Unit	
Average Daily Production Rate	

3rd Generation Data Models

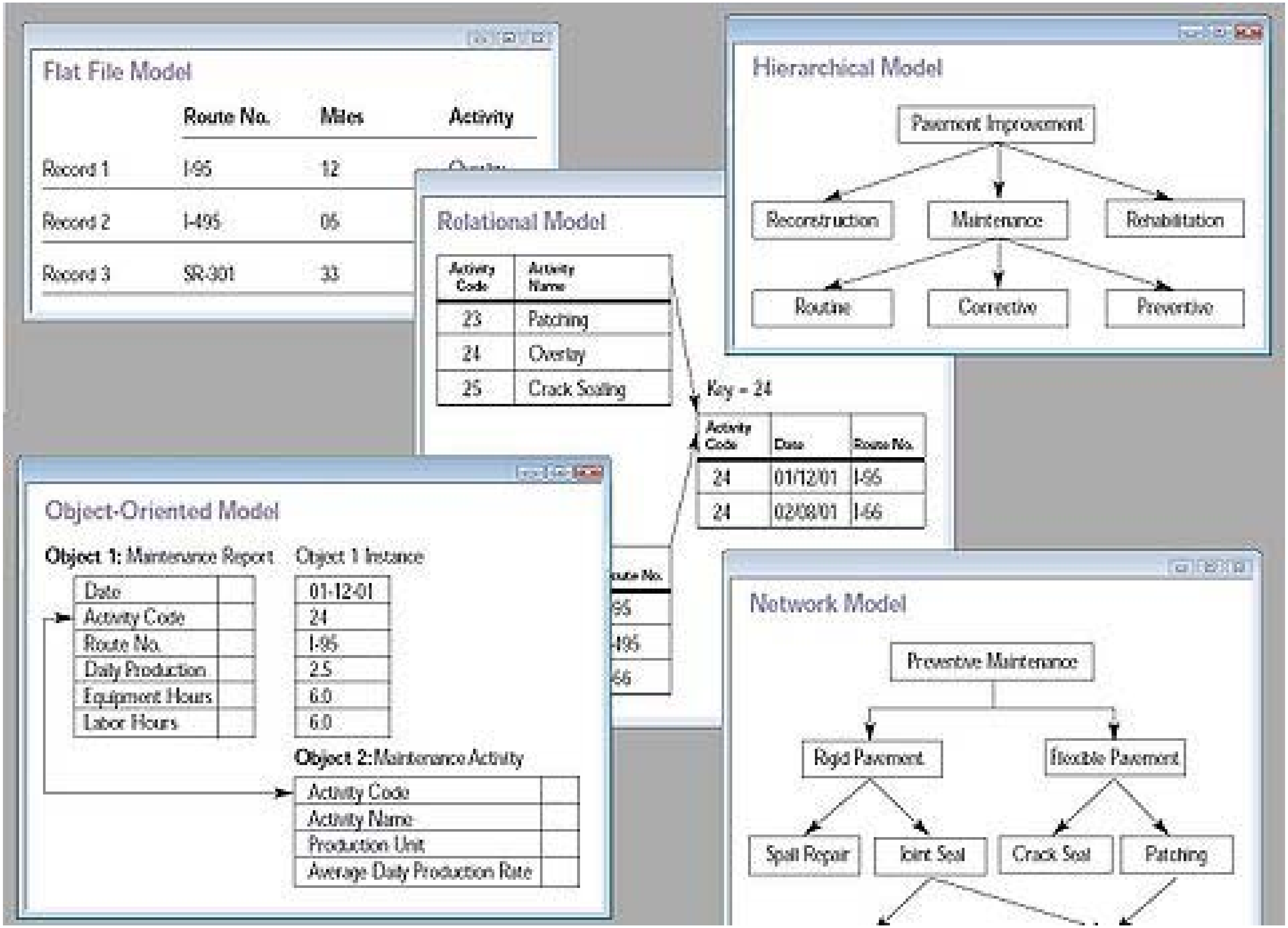
- XML Data Model (2005)
  - Stores data in XML format
  - Can be queried, exported, and serialized to desired format
  - Oracle XML DB (10g), GmStone System’s Gemfire Enterprise
- Spatial Data Model (2008)
  - Stores data and answers queries related to objects in space
  - Open Geospatial Consortium OGS
  - Oracle Spatial, PostgreSpatial

Concept Based Models (present)

- Syntax search
- Semantic search
- RDF, RDFS, OWL
- Technology support

RDF Model


http://www.w3.org/2000/10/swap/pim/contact#Person
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/People/EM/contact#me
http://www.w3.org/2000/10/swap/pim/contact#fullName
Eric Miller
http://www.w3.org/2000/10/swap/pim/contact#mailbox
mailto:em@w3.org
http://www.w3.org/2000/10/swap/pim/contact#personalTitle
Dr.


Flat File Model
Route No.
Miles
Activity
Record 1
I-95
12
Record 2
I-495
05
Record 3
SR-301
33
Relational Model
Activity Code
Activity Name
23
Patching
24
Overlay
25
Crack Sealing
Key = 24
Activity Code
Date
Route No.
24
01/12/01
I-95
24
02/08/01
I-66
Hierarchical Model
Pavement Improvement
Reconstruction
Maintenance
Rehabilitation
Routine
Corrective
Preventive
Object-Oriented Model
Object 1: Maintenance Report
Object 1 Instance
Date
Activity Code
Route No.
Daily Production
Equipment Hours
Labor Hours
01-12-01
24
I-95
2.5
6.0
6.0
Object 2: Maintenance Activity
Activity Code
Activity Name
Production Unit
Average Daily Production Rate
Network Model
Preventive Maintenance
Rigid Pavement
Flexible Pavement
Spall Repair
Joint Seal
Crack Seal
Patching