HEADING: :	**Dark Purple 1, Light Gold 2**	**20**
TYPES:	**Green**	**18**
SUBTYPES:	**Yellow**	**16**
Important Points:	Orange	16
Important Points:	Gold	16
Important Points:	Lime	16

UNIT 1: Overview of DBMS

Database design
Query processing

Data modelling
- **ER, EER,**
- **Object Oriented Databases,**
- **Object Relational Databases,**
- **Document oriented Databases**

Background of NoSQL
XML document
Structure of XML Data
XML Document Schema
Querying and Transformation
API
Storage of XML Data
XML Applications

ER diagram to Relational Model (7 steps)
EER diagram to Relational Model (8 steps)
Relational Model to ER diagram

DESIGN

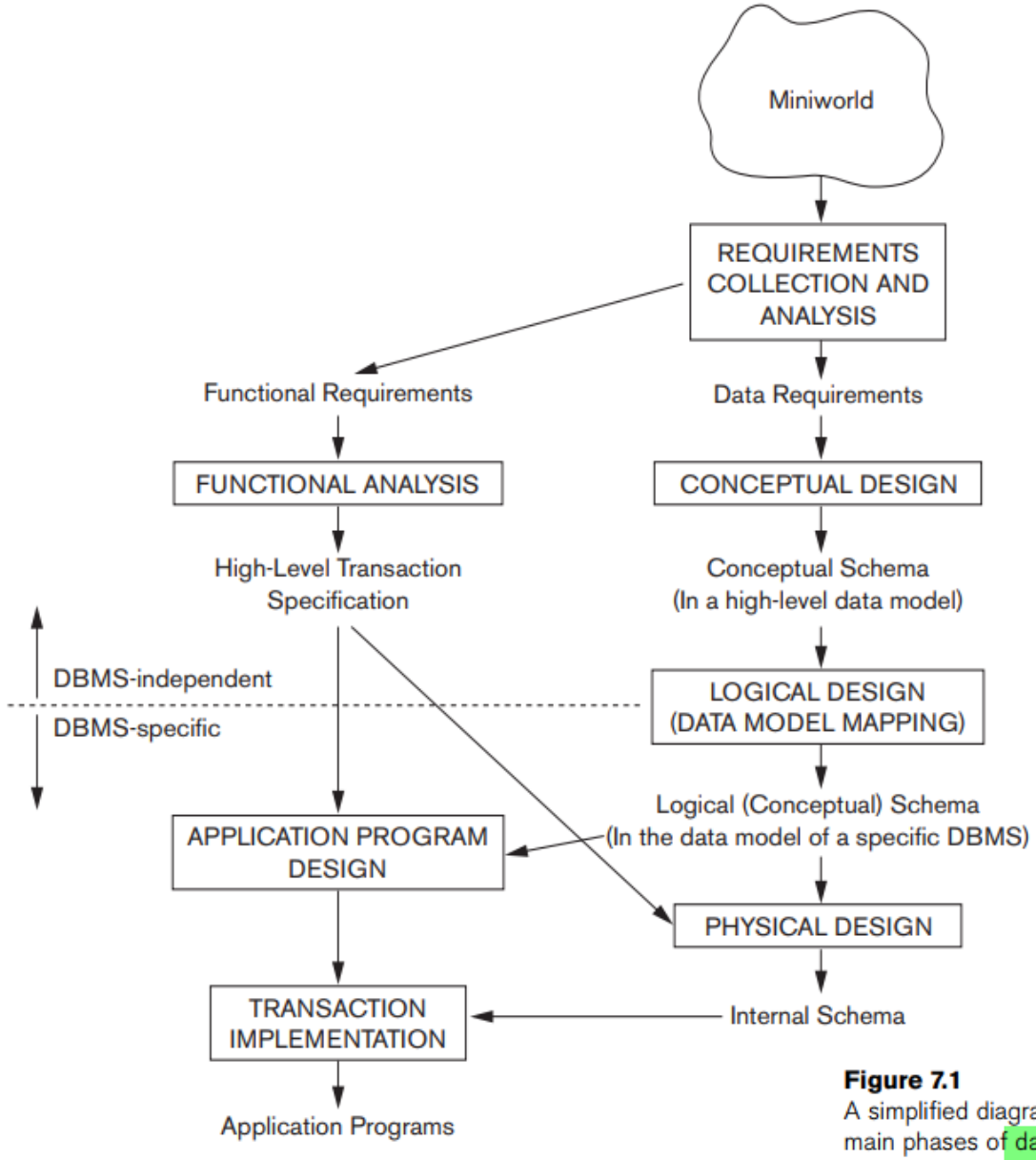


Figure 7.1
A simplified diagram to illustrate the main phases of database design.

EER to Relational

ER-to-Relational Mapping Algorithm

- **Step 1: Mapping of Regular Entity Types.**
  - For each regular (strong) entity type E in the ER diagram, create a relation R that includes all the simple attributes of E.
  - Choose one of the key attributes of E as the primary key for R.
  - If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.
- **Example:** We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram.
  - SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown.

- **Step 2: Mapping of Weak Entity Types**
  - For each weak entity type W in the ER schema with owner entity type E, create a relation R & include all simple attributes (or simple components of composite attributes) of W as attributes of R.
  - Also, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
  - The primary key of R is the *combination of* the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.
- **Example:** Create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT.
  - Include the primary key SSN of the EMPLOYEE relation as a foreign key attribute of DEPENDENT (renamed to ESSN).
  - The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT.

- **Step 3: Mapping of Binary 1:1 Relation Types**
  - For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.
  - Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S.
- **Example:** 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.

- **Step 4: Mapping of Binary 1:N Relationship Types.**
  - For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.
  - Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.
  - Include any simple attributes of the 1:N relation type as attributes of S.
- **Example:** 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION in the figure.
  - For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.

- **Step 5: Mapping of Binary M:N Relationship Types.**
  - For each regular binary M:N relationship type R, *create a new relation* S to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; *their combination will form the primary key* of S.
  - Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.
- **Example:** The M:N relationship type WORKS_ON from the ER diagram is mapped by creating a relation WORKS_ON in the relational database schema.
  - The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS_ON and renamed PNO and ESSN, respectively.
  - Attribute HOURS in WORKS_ON represents the HOURS attribute of the relation type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

- **Step 6: Mapping of Multivalued attributes.**
  - For each multivalued attribute A, create a new relation R.
  - This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.
  - The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.
- **Example:** The relation DEPT_LOCATIONS is created.
  - The attribute DLOCATION represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key-represents the primary key of the DEPARTMENT relation.
  - The primary key of R is the combination of {DNUMBER, DLOCATION}.

- **Step 7: Mapping of N-ary Relationship Types.**
  - For each n-ary relationship type R, where n>2, create a new relationship S to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
  - Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.
- **Example:** The relationship type SUPPY in the ER on the next slide.
  - This can be mapped to the relation SUPPLY shown in the relational schema, whose primary key is the combination of the three foreign keys {SNAME, PARTNO, PROJNAME}

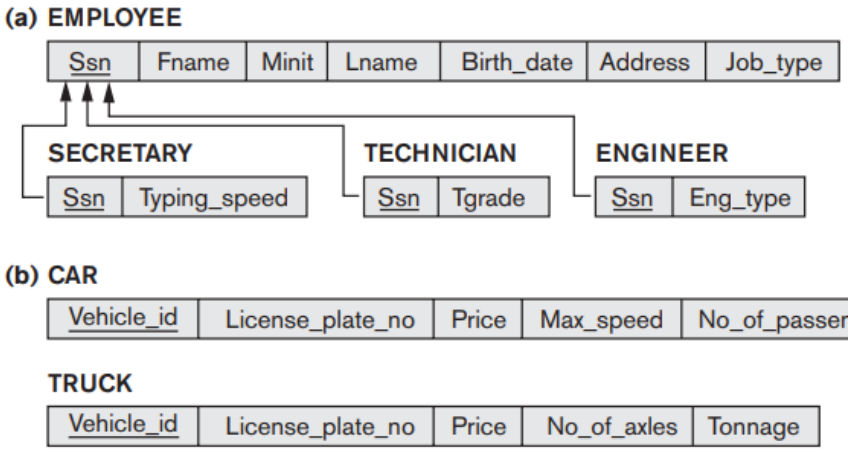


Figure 9.5
Options for mapping specialization or generalization. (a) Mapping the EER schema in Figure 8.4 using option 8A. (b) Mapping the EER schema in Figure 8.3(b) using option 8B. (c) Mapping the EER schema in Figure 8.4 using option 8C. (d) Mapping Figure 8.5 using option 8D with Boolean type fields Mflag and Pflag.

Table 9.1 Correspondence between ER and Relational Models

ER MODEL	RELATIONAL MODEL
Entity type	*Entity* relation
1:1 or 1:N relationship type	Foreign key (or *relationship* relation)
M:N relationship type	*Relationship* relation and *two* foreign keys
n-ary relationship type	*Relationship* relation and *n* foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

XML

- eXtensible Markup Language
- Markup Language
- Standard for structuring and exchanging data over the Web
- HTML isn’t suitable for specifying structured data extracted from DB

Data
- Structured
- Semi Structured
- Unstructured

XML Document (Element and Attribute)
- Data Centric
- Document Centric
- Hybrid

Well Formed XML
- syntactically correct
- allows to create DOM

Valid XML
- stronger criterion
- well formed, and must follow a particular schema
- element name’s structure is defined in a separate file
    - DTD (Document Type Definition) file
    - XSD (XML Schema Definition) file

DTD

XSD

XML QUERY:
- XPath
- XQuery

UNIT 2

Overview of OOP;
Complex objects;
Identity, structure etc.
Object model of ODMG,
Object definition Language ODL;
Object Query Language OQL;
Conceptual design of Object database.

Overview of object relational features of SQL;
Object-relational features of Oracle;
Implementation and related issues for extended type systems;
syntax and demo examples,
The nested relational model.

Overview of C++ language binding;

Object Databases (ODB)

ODBMS

Key feature of object databases is that they allow to specify both,
- **structure** of complex objects
- **operations** that can be applied to these objects

Objects

An object typically has two components:
- value
- operation

It can have a
- complex data structure
- specific operations defined

Unique Identity to each OBJECT in the database
Implemented via **Object Identifier**
Maintains integrity bw Real World and Databases
Every OBJECT has OID, Literal Value not have OID (stored in Object)

Type Constructors
Objects can have type structure of arbitrary complexity

Atom
- Built in datatypes
- int, string, bool, float

Struct / Tuple
- Compound or Composite type
- Not a type but a Type Generator
- Name <FName: string, MName: char, LName: string>

Collection
- Multivalued
-

SET: List of distinct elements
BAG: Similar to Set, but elements are not distinct
LIST: Similar to Bag, but elements are ordered
ARRAY: Similar to List, but has a maximum size
Dictionary: Key Value pairs

Declaration of Object types
define type DATE
tuple (Day: Integer, Month: Integer, Year: Integer);

ENCAPSULATION

The concepts of encapsulation is applied to database objects in ODBs by **defining the behaviour of a type of object** based on the operations that can be externally applied to objects of that type

Operations: Constructor, Destructor, Modifier, Retrieve

The external users of the object are only made aware of the interface of the operations, which defines the **name and arguments**

The interface part of an operation – **signature**
The operation implementation – **method**

The term class is often used to refer to a type definition, along with the definitions of the operations for that type

```
define class DEPARTMENT
type tuple (Dname: string, Dnum: integer);

operations
no_of_d: integer;
create_d: DEPARTMENT;
add(e: EMPLOYEE): boolean;
```

INHERITANCE

Inheritance allows the definition of new types based on other predefined types, leading to type hierarchy

A type in its simplest form has a type name and a list of visible (public) functions.

TYPE_NAME: function, function, ..., function

PERSON: Name, Address, Birth_date, Age, Ssn

Person is supertype
Employee and Student are subtype

Subtype has all the functions from supertype + functions of their own

EMPLOYEE: Name, Address, Birth_date, Age, Ssn, Salary, Hire_date, Seniority
STUDENT: Name, Address, Birth_date, Age, Ssn, Major, Gpa

Stored Attribute = Name, Birth_date, Hire_date, SSN
Operations = Age, Seniority, Gpa

EMPLOYEE **subtype-of** PERSON: Salary, Hire_date, Seniority
STUDENT **subtype-of** PERSON: Major, Gpa

Declaring subtypes doesn't create the objects.
We have create the objects.
The object, if it belongs to a subtype, should also belong to supertype

Multiple INHERITANCE

Multiple inheritance occurs when a certain subtype T is a subtype of two (or more) types and hence inherits the functions of both supertypes.

This leads to the creation of a type lattice

Subtype **ENGINEERING_MANAGER** that is a subtype of both **MANAGER** and **ENGINEER**.

Problem:
- Subtype can inherit function of same name in both M and E
- SALARY in M as well E
- It may be possible M and E inherit SALARY from Person

- Ask the user to choose from which type SALARY is being inherited
- Use system default type
- Disallow creation of subtype unless SALARY name is changed

Selective INHERITANCE

Selective inheritance occurs when a subtype inherits only some of the functions of a supertype. Other functions are not inherited.

An EXCEPT clause may be used to list the functions in a supertype that are not to be inherited by the subtype

Overview of ODMG Model
Page 406

EER to ODB Schmea
Page 426

Overview of the C++ Language Binding in the ODMG Standard
Page 436

UNIT 3

INFORMATION INTEGRATION

Definition:
The process of merging multiple databases (or information sources) to function as a single database.

- Physically (Warehouses)
- Virtually (Mediators or Middleware)

Challenges:
To resolve the conflicts, “Wrappers” are designed as translators between source data and integrated database schema

Query Optimization: ID require QO. Mediator systems,

Global-as-View:
ID data is defined by how its constructed from the **S**

Local-as-View:
SD is defined by the schema supported by **ID**

Why Information Integration?

- Ideal Scenario: Data could be put into a SINGLE db from scratch
- But many Databases are created independently
- Creating a new single database from scratch is costly

Middleware (layer of abstraction on top of existing db)
- Legacy system continues serving current applications
- New applications can access middleware for data

Heterogeneity Problem

When integrating independently developed information sources, problems arise!

Heterogeneous Sources and Heterogeneity Problems
Toyota Automobile has 1000 dealers.
Each dealer maintains separate database.
Toyota want to create integrated database.

1) Communication Heterogeneity

- Some dealers use HTTP (making db available on Web)
- Some dealers use FTP

2) Query Language Heterogeneity

How we Query or Modify the database
- SQL or not (Different versions of XML)
- Non relational, Object Oriented, Excel, XML db

3) Schema Heterogeneity

Variances in Schema

Cars (SerialNo, Model, Color, Available)
Auto (UniqueN, Model, Available) Options (UniqueN, Gears)

- Missing Values: Color
- Number of Relations: 1 and 2
- Atrribute Names difference: SerialNo, UniqueN

4) Data Type Differences (SerialNo)

- Strings (or) Integers
- Length of Strings might differ

5) Value Heterogeneity

Different representation of the same concept
Eg: Color: {STRING Black, INTEGER 20, CODE BL}

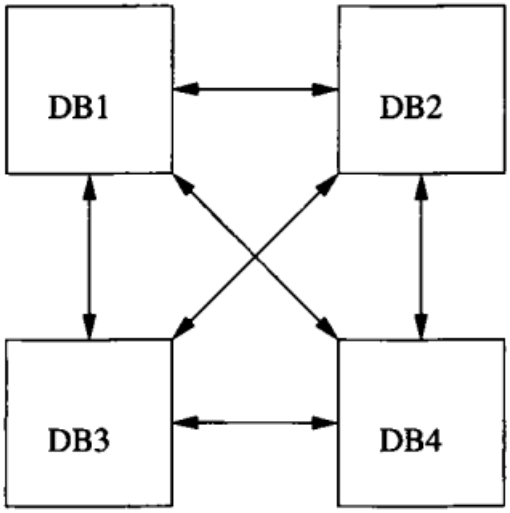
6) Semantics Heterogeneity

Differing interpretations of terms
- Trucks classified as Cars
- Trucks not classified as Cars

Modes of Integration (3)

FEDERATED Databases

- Sources are independent
- But one source can supply information to another source

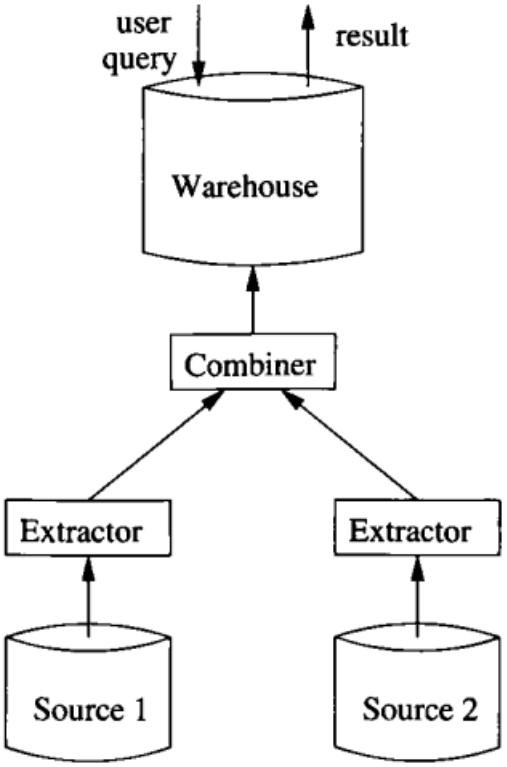


- One to One connections between all pairs
- Requires n(n-1) components to be coded

Data Warehouses

Data from several sources is extracted and combined into a global schema

Data is then stored at Warehouse, which looks like an ordinary database to the user



Two approaches to construct the data in the warehouse:

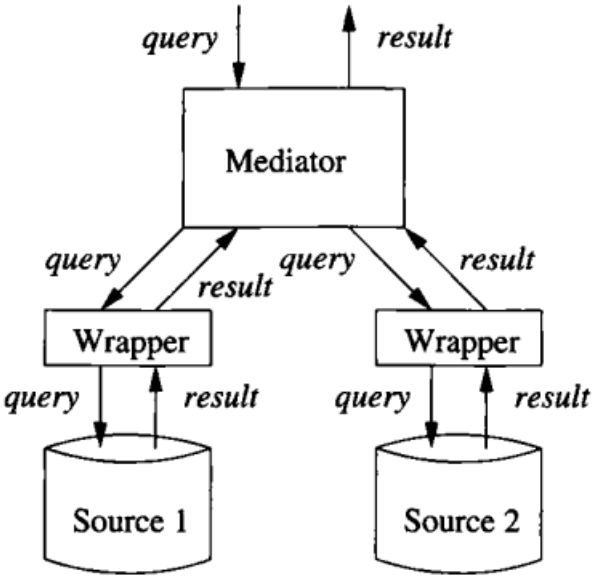
1)
- The warehouse is periodically closed to queries
- Reconstructed from the current data in the sources
- Reconstruction occurs once a night (or) at even longer intervals
- Most common approach

2)
- The warehouse is updated periodically (eg, each night)
- Based on the changes made to the sources
- Involves updation of smaller amounts of data
- Important if the warehouse is large needs quick modification
- Calculating changes required to the warehouse is complex

Mediators

A mediator is a software component that supports a virtual database, which the user may query as if it were materialized (as a warehouse)

- User issues a query to the mediator
- Mediator has no data of its own, it gets the data from its sources
- Mediator sends query to its Wrappers, which queries the sources
- Mediator combines the result and synthesizes the response



Models of Parallelism

Shared MEMORY

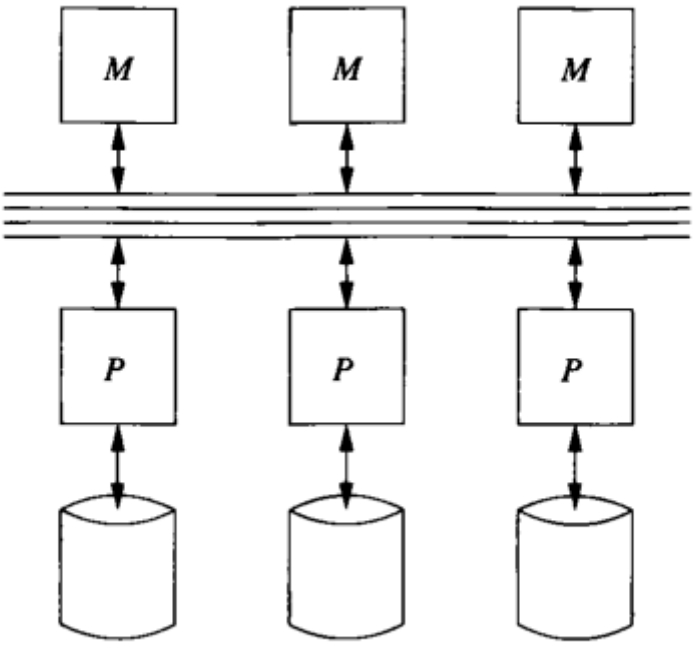


Figure 20.1: A shared-memory machine

- Each Processor has access to all Memory of all processors
- NUMA

Shared DISK

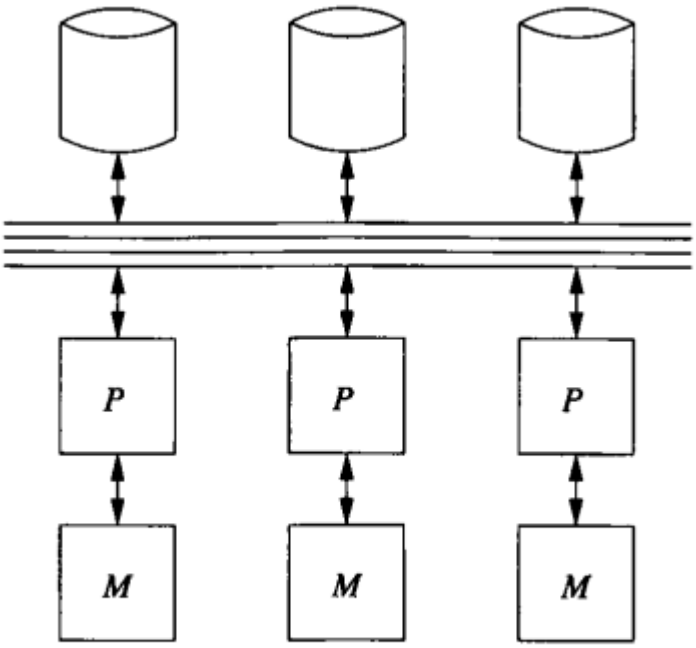


Figure 20.2: A shared-disk machine

- Memory is not accessible. But Processor can access Disk via CN
- NAS, SAN

Shared NOTHING

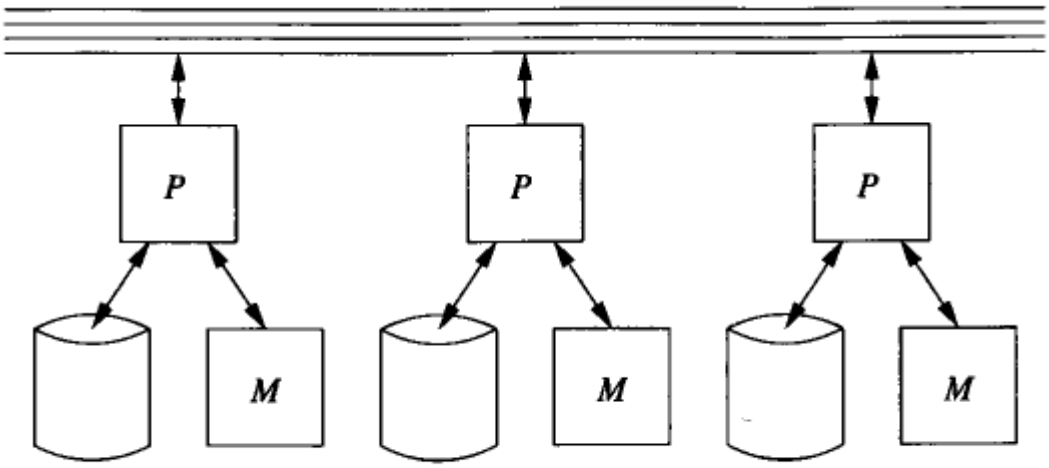


Figure 20.3: A shared-nothing machine

- All Processors have their own Memory and Disk
- Used for Database Systems

MAP REDUCE

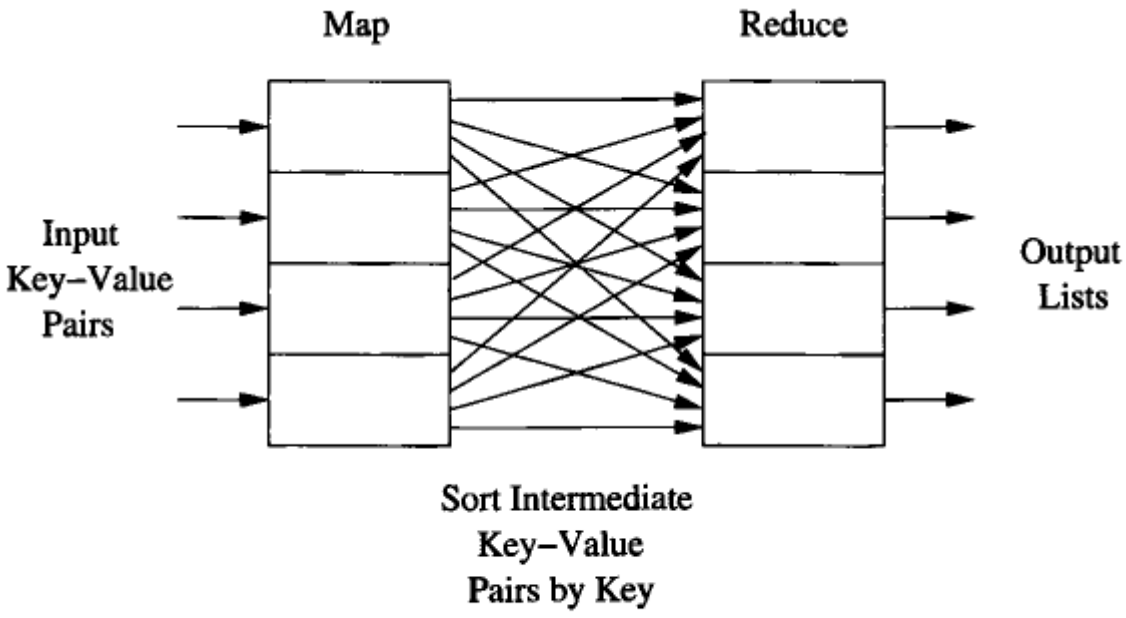


Figure 20.4: Execution of map and reduce functions

DD

UNIT 4

DD

UNIT 5

MOBILE DATABASE

Recent advances in **portable** and **wireless** technology led to mobile computing, a new dimension in data communication and processing.

Mobile coupled with wireless communications allow clients to access data from virtually anywhere and at any time.

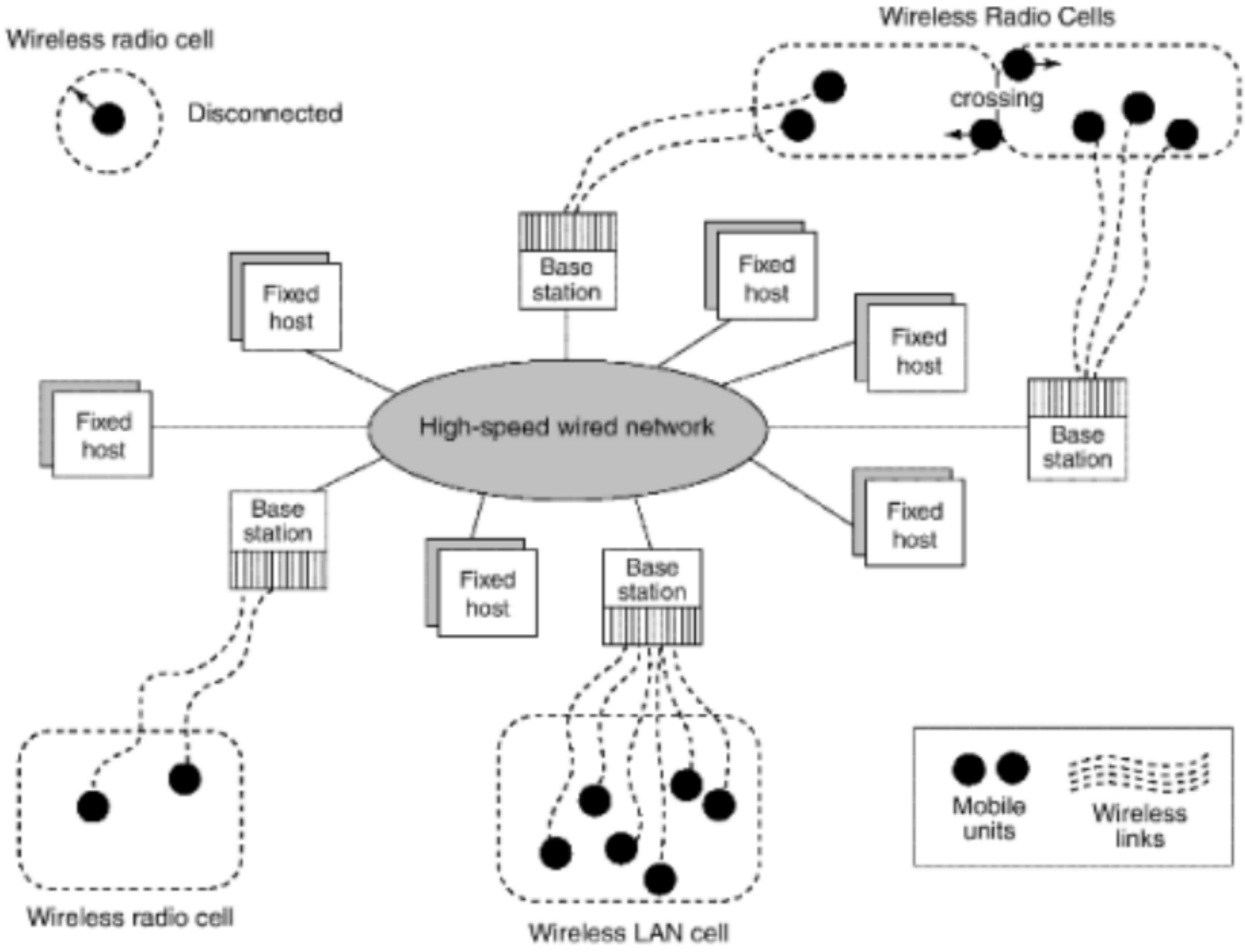
Challenges

- The limited and intermittent connectivity afforded by w/l comm
- The limited life of the power supply (**battery**)
- The changing topology of the network.
- New architectural possibilities and challenges.

Characteristics

- Intermittent Connectivity
- Limited Battery
- Changing Client Location
- Communication Latency

Mobile Computing Architecture



- Distributed Network
- Fixed Hosts and Base Stations are interconnected through HSWN

- FH: Configured to manage Mobile Units
- BS: Gateway to fixed network for Mobile Units

- Wireless Communications:

Bandwidth lower than wired. 2G, 3G, LTE
Range. Locality of Access. Interferences

- Client Network Relation:

Mobile can move around the geographic mobility domain.
Domain -> Cells. Each cell has a Base Station
Traditional Client Server

OTHER (Infrastructure LESS):
- MANET
- Mobile Ad Hoc Network
- Self Configuring mobiles that connect to other mobiles
- Bluetooth
- P2P

MULTIMEDIA DATA

TEXT
- Formatted, Unformatted
- For Parsing, structured documents (HTML)

GRAPHICS
- Drawings, Illustrations
- Encoded with Descriptive Standards
- CGM, PICT, Postscript

IMAGES
- Photographs, Pixel Data
- Bitmap, JPG, JPEG, PNG

ANIMATIONS
- Temporal sequence of images
- GIF

VIDEO
- Sequenced photographed data at specific rates
- 30 frames per second
- MP4, MKV, MOV

AUDIO
- Aural recordings
- Analog recording -> Digital format

COMPOSITE Data
- Mixed Multimedia
- Audio + Video

GEOGRAPHIC INFORMATION SYSTEMS

They are used to collect, model, analyze information describing physical properties of the geographical world.

Spatial Data
Maps, Political Boundaries, Roads, Rivers, Elevation

Non Spatial Data
Census counts, Sales and Marketing information, Economic data

Applications or Categories
It is possible to divide GIS into three categories:
- Cartographic applications
- Digital terrain modeling applications
- Geographic objects applications

- online map services
- vehicle-navigation systems (GPS)
- land usage information for ecologists and planners.
- distribution-network information for public-service utilities such as telephone, electric-power, and water-supply systems

Vector Data
Geometric objects such as Point, Line, Polygon
- Map data
- Road – Poly Lines
- Countries – Complex Polygons

Raster Data
Array of points, where each point represents value of a real world
- Bitmaps or Pixel maps (2 or more dimension)
- Satellite image
- + actual image, contains metadata (loc, lat, long)
- Represented by TILES, each covering certain area

TIN (Triangulated Irregular Network)
DTM (Digital Terrain Modelling)

(Page 1097: DC)

GENOME Database

Biology encompasses: Environment, Ecology, Anatomy, Histology

GENETICS: Genes are unit of heredity
- Mendelian
- Molecular
- Population

DNA (DeoxyriboNucleic Acid) – 1867
Watson and Crick – Double helix structure – 1953

GENOME
Total genetic information that can be obtained about an entity

Human Genome: Complete set of genes to create humans
30000 genes spread over 23 pairs of chromosomes, 3billn nucleotides

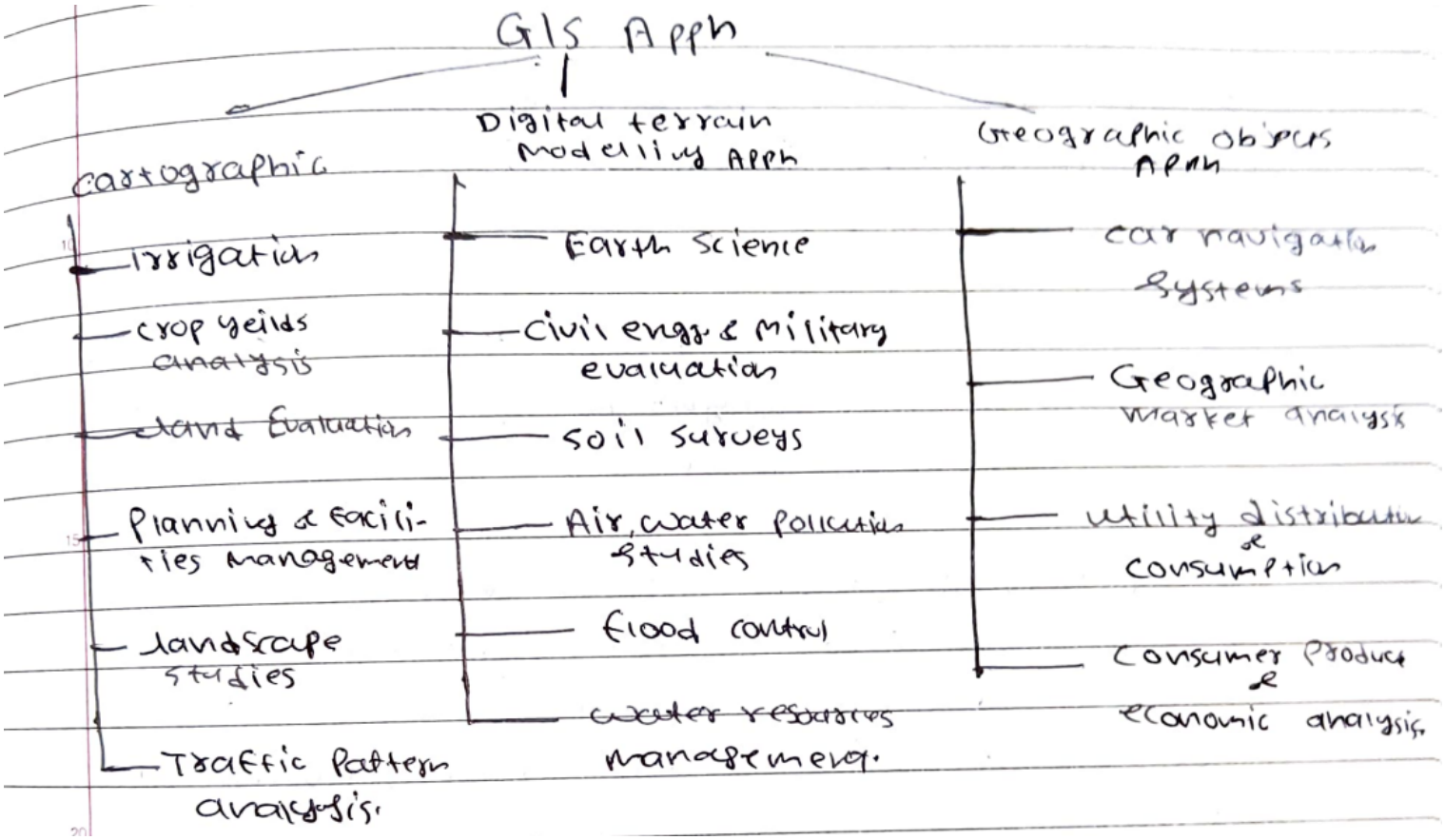
HGP:
- Obtain the ordering of the bases

GenBank
- DNA Sequence Database
- National Center for Biotechnology Information (NCBI)
- 1978
- 100 GB uncompressed
- 31 billion nucleotide bases
- 1400 organisms
- ASN.1

Genome Database (GDB)
- 1989
- Human gene mapping data

Online Mendelian Inheritance in Man
- genetic basis of human disease.

EcoCyc
- E coli K 12


GIS Apph
Cartographic
Irrigation
crop yeilds analysis
land Evaluation
Planning & facilities management
landscape studies
Traffic Pattern analysis.
Digital terrain Modelling Apph
Earth Science
Civil engg & Military evaluation
soil surveys
Air, water pollution studies
flood control
water resources management.
Geographic objects Apph
car navigation systems
Geographic market analysis
utility distribution & consumption
Consumer Product & economic analysis.