Introduction to Databases

CT042-3-1-IDB (version1)

Database Design & Implementation & Tuning

Topic & Structure of The Lesson

- Conceptual design
- Logical design
- Physical design
- Performance & security
- Backup and Recovery
- Maintenance & tuning

Learning Outcomes

- **At the end of this topic, You should be able to**
  - State the difference between conceptual, logical and physical design
  - understand the performance and security criteria
  - Explain the backup and recovery procedures

Key Terms You Must Be Able To Use

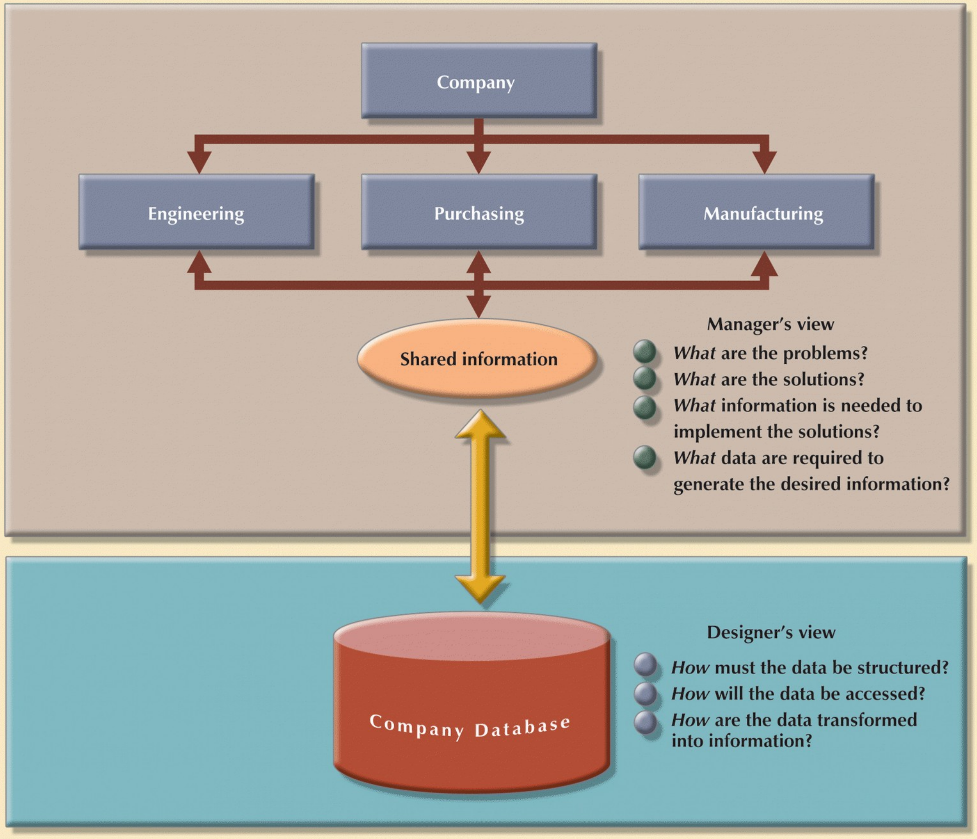
- If you have mastered this topic, you should be able to use the following terms correctly in your assignments and exams:
  - Conceptual model
  - Logical model
  - Physical model

Database Design

- Necessary to concentrate on data
- Characteristics required to build database model
- Two views of data within system:
  - Business view of data as information source
  - Designer's view of data structure, its access, and activities required to transform data into information

Database Design (continued)

FIGURE 9.5 Two views of data: business manager and designer



Manager's view

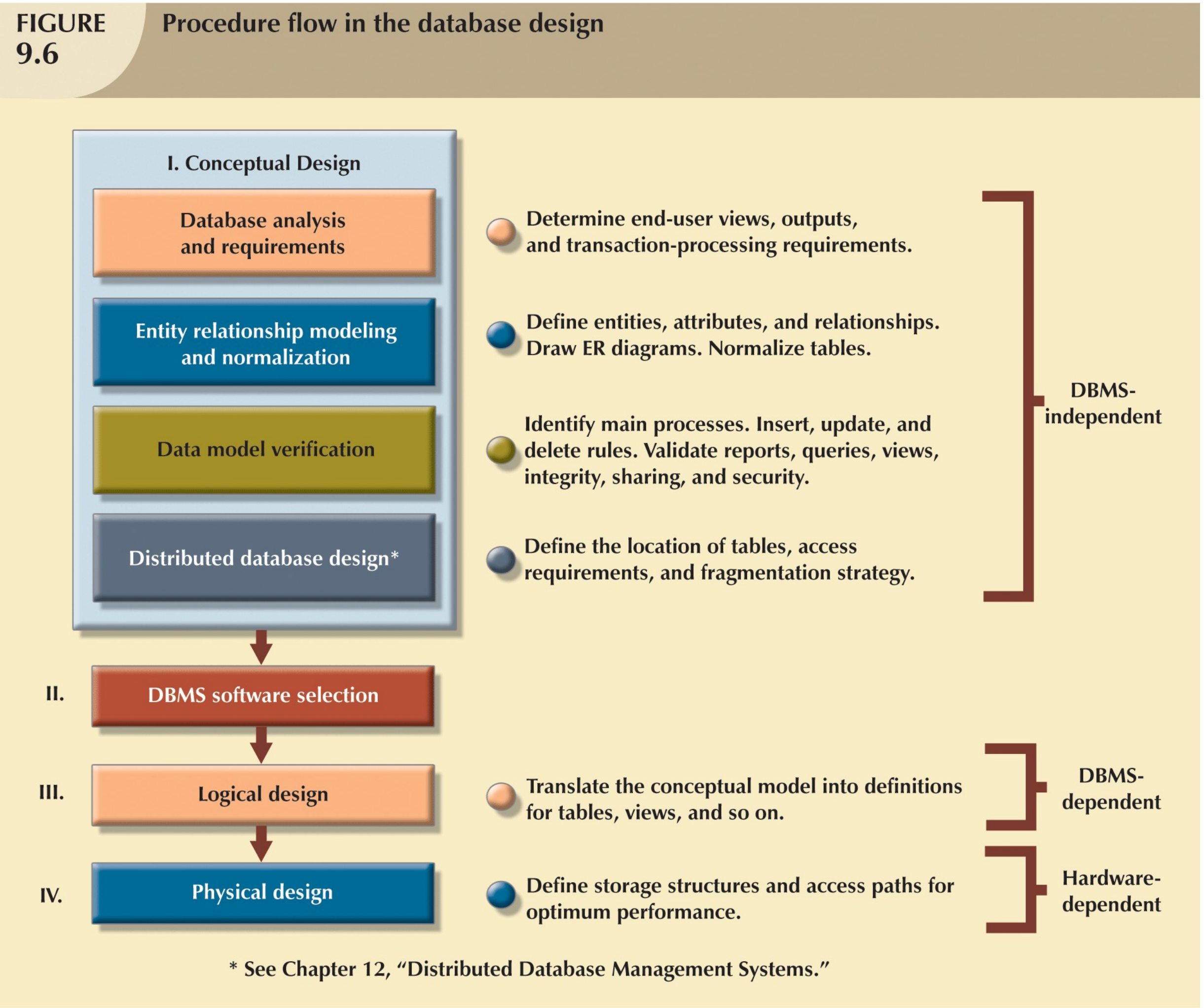
- *What* are the problems?
- *What* are the solutions?
- *What* information is needed to implement the solutions?
- *What* data are required to generate the desired information?

Designer's view

- *How* must the data be structured?
- *How* will the data be accessed?
- *How* are the data transformed into information?

Database Design (continued)

FIGURE 9.6 Procedure flow in the database design


I. Conceptual Design
Database analysis and requirements
Entity relationship modeling and normalization
Data model verification
Distributed database design*
Determine end-user views, outputs, and transaction-processing requirements.
Define entities, attributes, and relationships. Draw ER diagrams. Normalize tables.
Identify main processes. Insert, update, and delete rules. Validate reports, queries, views, integrity, sharing, and security.
Define the location of tables, access requirements, and fragmentation strategy.
DBMS-independent
II.
DBMS software selection
III.
Logical design
Translate the conceptual model into definitions for tables, views, and so on.
DBMS-dependent
IV.
Physical design
Define storage structures and access paths for optimum performance.
Hardware-dependent

* See Chapter 12, "Distributed Database Management Systems."

I. Conceptual Design

- Data modeling used to create an abstract database structure that represents real-world objects in most realistic way possible
- Must embody clear understanding of business and its functional areas
- Ensure that all data needed are in model, and that all data in model are needed

I. Conceptual Design (continued)

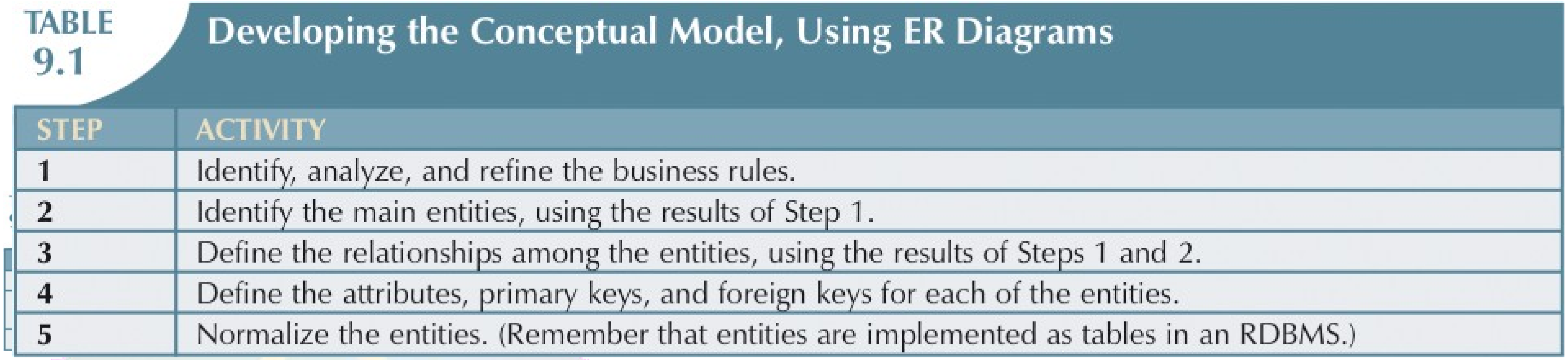
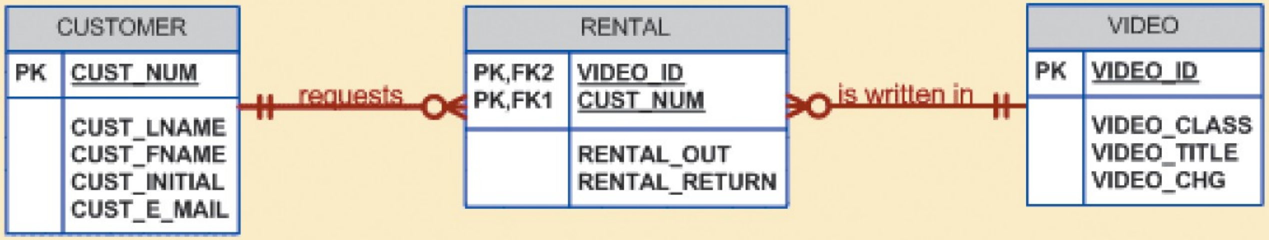


TABLE 9.1 Developing the Conceptual Model, Using ER Diagrams

STEP	ACTIVITY
1	Identify, analyze, and refine the business rules.
2	Identify the main entities, using the results of Step 1.
3	Define the relationships among the entities, using the results of Steps 1 and 2.
4	Define the attributes, primary keys, and foreign keys for each of the entities.
5	Normalize the entities. (Remember that entities are implemented as tables in an RDBMS.)

I. Conceptual Design (continued)

FIGURE 9.7 A composite entry



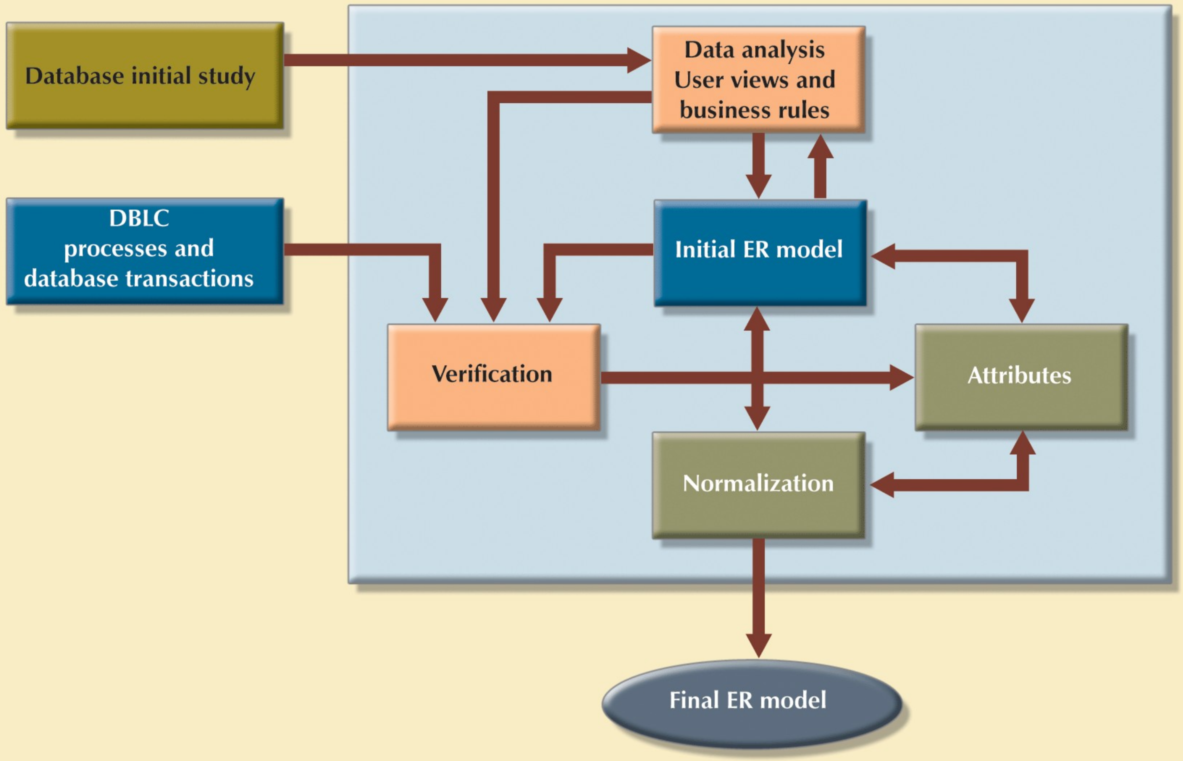
CUSTOMER	
PK	CUST_NUM
	CUST_LNAME CUST_FNAME CUST_INITIAL CUST_E_MAIL

RENTAL	
PK,FK2 PK,FK1	VIDEO_ID CUST_NUM
	RENTAL_OUT RENTAL_RETURN

VIDEO	
PK	VIDEO_ID
	VIDEO_CLASS VIDEO_TITLE VIDEO_CHG

I. Conceptual Design (continued)

FIGURE 9.8 ER modeling is an iterative process based on many activities



Database initial study

DBLC processes and database transactions

Data analysis User views and business rules

Initial ER model

Verification

Attributes

Normalization

Final ER model

II. DBMS Software Selection

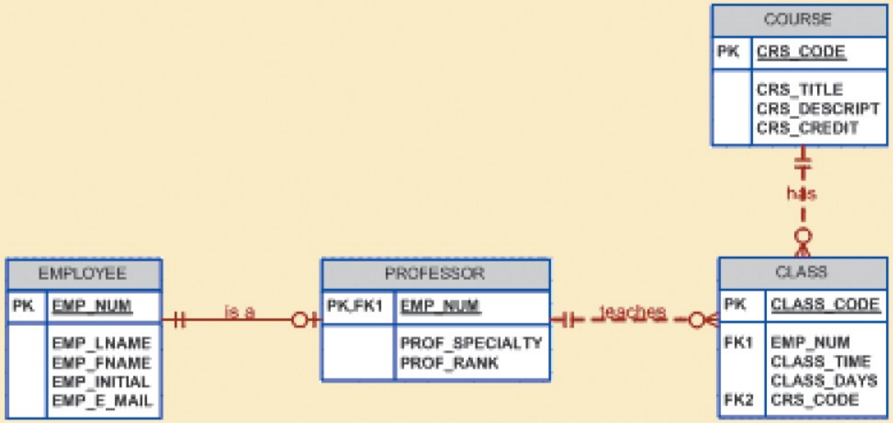
- Critical to information system's smooth operation
- Advantages and disadvantages should be carefully studied

III. Logical Design

- Used to translate conceptual design into internal model for selected database management system
- Logical design is software-dependent
- Requires that all objects in model be mapped to specific constructs used by selected database software

III. Logical Design (continued)

FIGURE 9.11 A simple conceptual model



IV. Physical Design

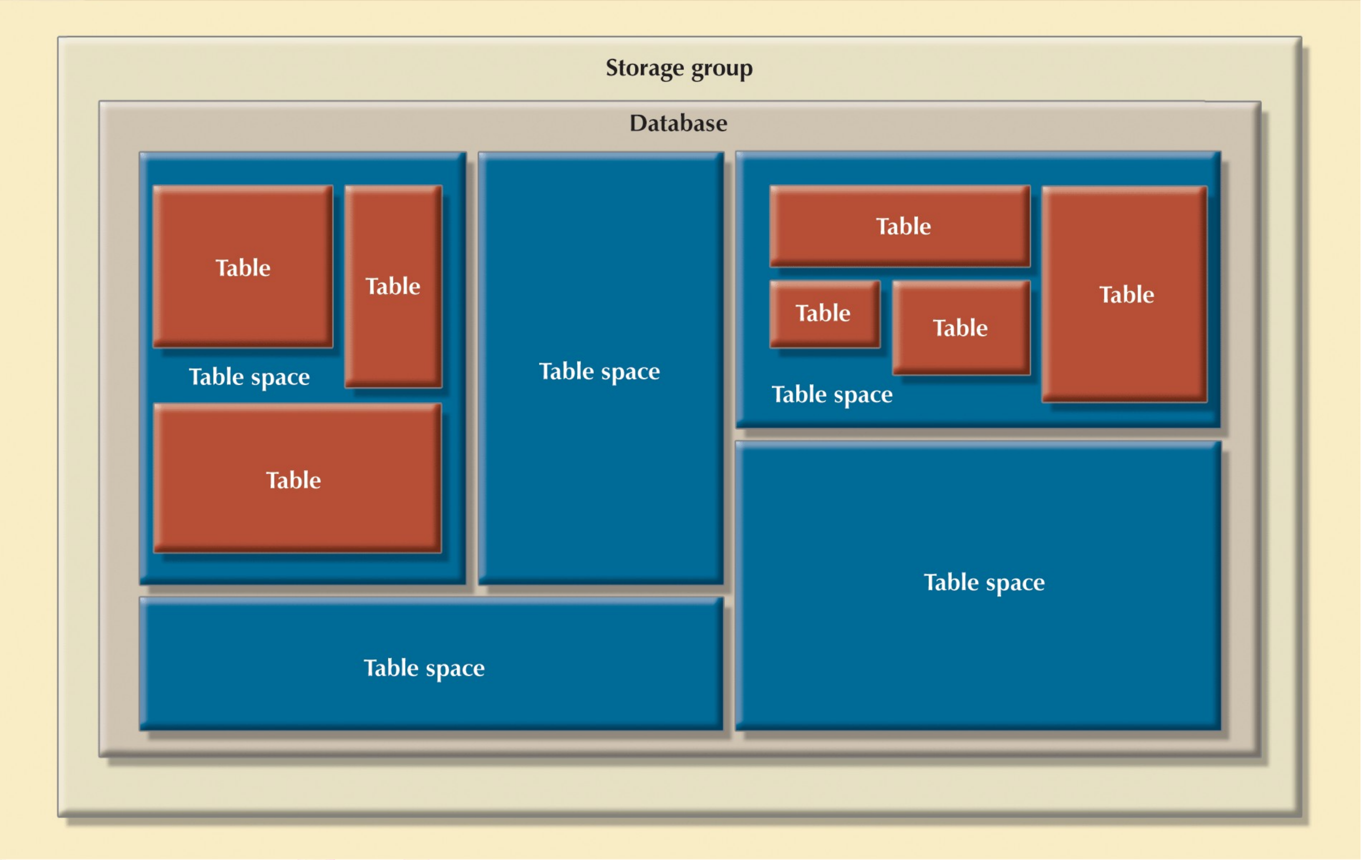
- Process of selecting data storage and data access characteristics of database
- Storage characteristics are function of device types supported by hardware, type of data access methods supported by system, and DBMS
- Becomes more complex when data are distributed at different locations

Implementation and Loading

- New database implementation requires creation of special storage-related constructs to house end-user tables

Implementation and Loading (continued)

FIGURE 9.12 Physical organization of a DB2 database environment



Performance

- One of most important factors in certain database implementations
- Not all DBMSs have performance-monitoring and fine-tuning tools embedded in their software
- There is no standard measurement for database performance
- Not only (nor even main) factor

Security

- Data must be protected from access by unauthorized users
- Must provide for following:
  - Physical security
  - Password security
  - Access rights
  - Audit trails
  - Data encryption

Backup and Recovery

- Database can be subject to data loss through unintended data deletion and power outages
- Data backup and recovery procedures

Integrity

- Enforced through proper use of primary and foreign key rules

Company Standards

- May partially define database standards
- Database administrator must implement and enforce such standards

Testing and Evaluation

- Occurs in parallel with applications programming
- Database tools used to prototype applications
- If implementation fails to meet some of system's evaluation criteria:
    - Fine-tune specific system and DBMS configuration parameters
    - Modify physical design
    - Modify logical design
    - Upgrade or change DBMS software and/or hardware platform

Operation

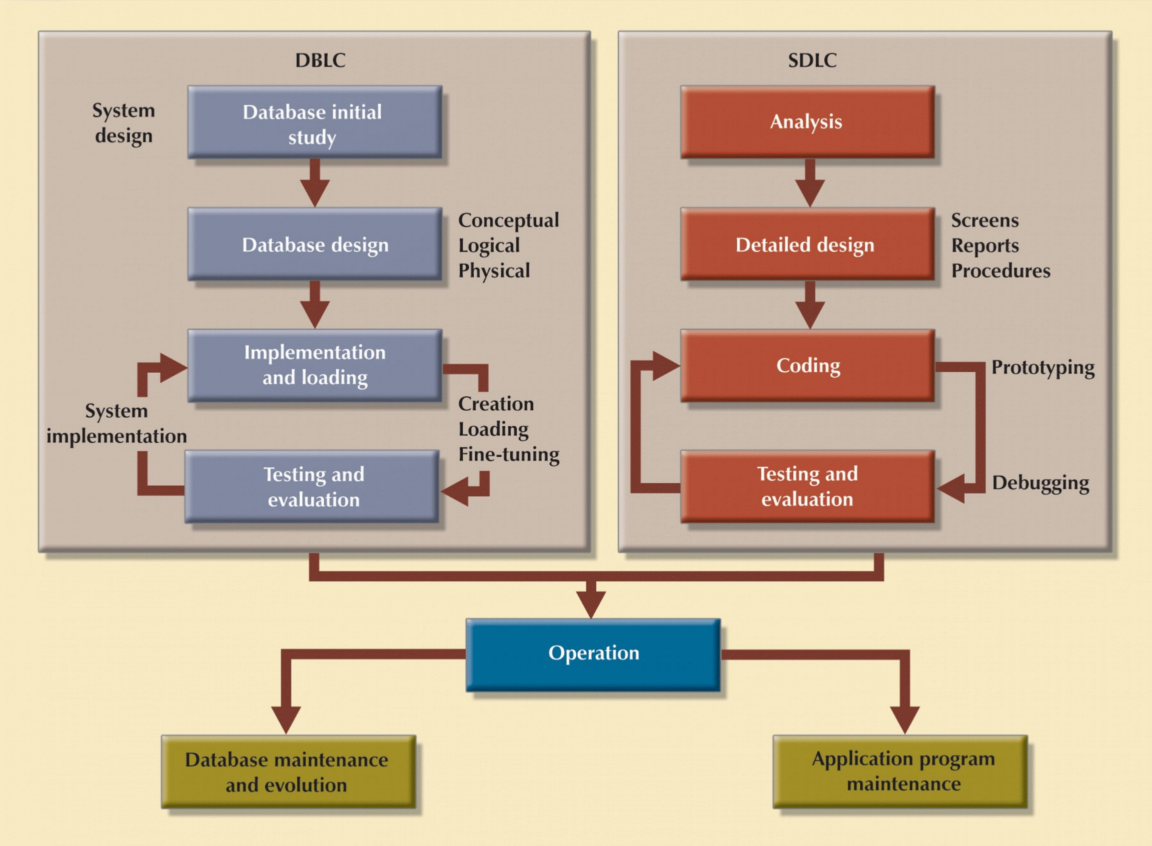
- Once database has passed evaluation stage, it is considered operational
- Beginning of operational phase starts process of system evolution

Maintenance and Evolution

- Required periodic maintenance:
  - Preventive maintenance (backup)
  - Corrective maintenance (recovery)
  - Adaptive maintenance
  - Assignment of access permissions and their maintenance for new and old users

Maintenance and Evolution (continued)

FIGURE 9.13 Parallel activities in the DBLC and the SDLC

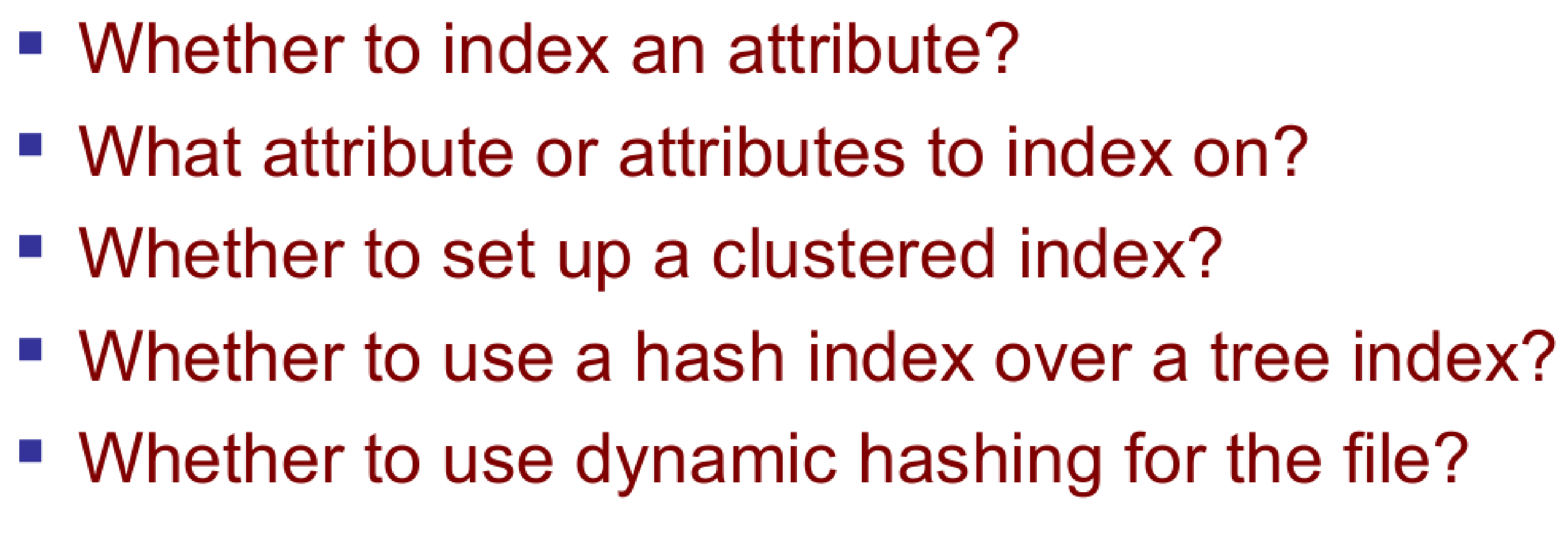


Physical Database Design in Relational Databases

- Factors that Influence Physical Database Design
  - Analyzing the database queries and transactions (file accessed, attributes, operation type(insert,join,...) etc)
  - Analyzing the expected frequency of invocation of queries and transactions
  - Analyzing the time constraints of queries and transactions
  - Analyzing the expected frequencies of update operations
  - Analyzing the uniqueness constraints on attributes

Physical Database Design in Relational Databases

- Design decisions about indexing



- Whether to index an attribute?
- What attribute or attributes to index on?
- Whether to set up a clustered index?
- Whether to use a hash index over a tree index?
- Whether to use dynamic hashing for the file?

Physical Database Design in Relational Databases

- Denormalization as a design decision for speeding up queries
  - Normalization- avoid redundancy and anomalies
  - Denormalization→
    - To improve the performance of frequently occurring queries and transactions.

Database Tuning

The process of continuing to adjust the physical database design by monitoring resource utilization as well as internal DBMS processing to reveal bottlenecks such as contention for the same data or devices.

- To make application run faster
- To lower the response time of queries/transactions
- To improve the overall throughput of transactions

Database Tuning

Problems to be considered in tuning:

- How to avoid excessive lock contention?
- How to minimize overhead of logging and unnecessary dumping of data?
- How to optimize buffer size and scheduling of processes
- How to allocate resources such as disks, RAM and processes for most efficient utilization?

Database Tuning

Tuning Indexes

- **Reasons to tuning indexes**
  - Certain queries may take **too long to run for lack of an index**
  - Certain **indexes may not get utilized** at all
  - Certain indexes may be causing excessive **overhead** because the **index** is on an attribute that undergoes **frequent changes**
- **Options to tuning indexes**
  - Drop or/and build new indexes
  - Change a non-clustered index to a clustered index (and vice versa)
  - Rebuilding the index

Database Tuning

Tuning the Database Design

- Dynamically changed processing requirements need to be addressed
  - by making changes to the conceptual schema if necessary
  - and to reflect those changes into the logical schema and physical design.
- Possible Changes:
  - Table join, normalization/denormalization, table partitioning and attribute relocation to different table, etc

Database Tuning

Tuning Queries

- Indications for tuning queries
  - A query issues too many disk accesses
  - The query plan shows that relevant indexes are not being used.
  - E.g.
    - Use of views: views become an overkill
    - The order of tables in the FROM clause may affect the join processing.