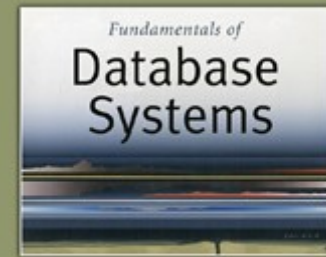


Chapter - 5

Physical Database Design



5th Edition

Elmasri / Navathe

Physical Database Design

- **Implementing representation**

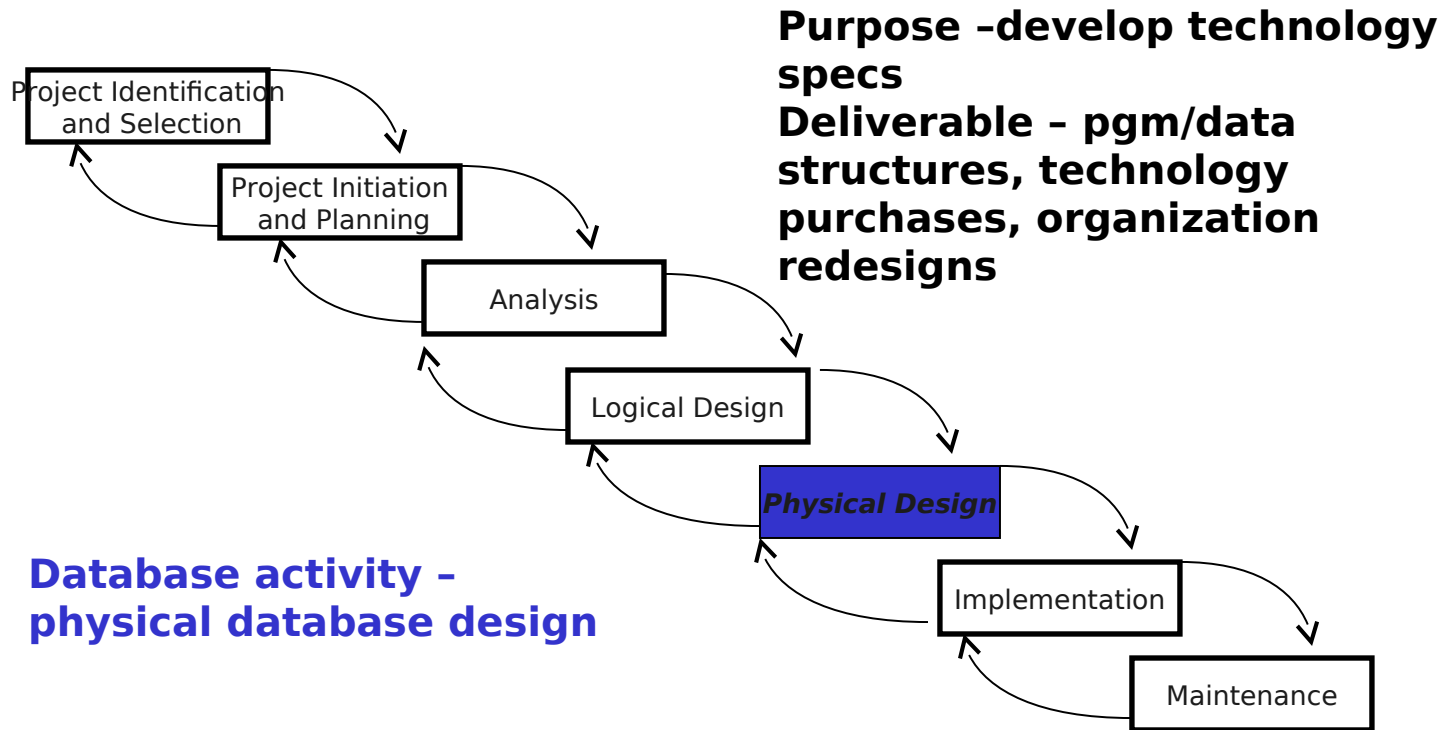
- Analysis transactions
- File organization
 - Indexes
 - Disk space

- **Security**

- Policy & procedure
- User view
- Access rules
- Transmission

physical

The Physical Design Stage of SDLC



Physical Database Design

- **Purpose** - translate the logical description of data into the *technical specifications* for storing and retrieving data.
- **Goal** - create a design for storing data that will provide *adequate performance* and insure *database integrity, security* and *recoverability*.

Choosing Data Types

- **CHAR** – fixed-length character
- **VARCHAR** – variable-length character (memo)
- **LONG** – large number
- **NUMBER** – positive/negative number
- **DATE** – actual date
- **BLOB** – binary large object (good for graphics, sound clips, etc.)

Physical Records

- **Physical Record:** A group of fields stored in adjacent memory locations and retrieved together as a unit.
- **Page:** The amount of data read or written in one I/O operation
- **Blocking Factor:** The number of physical records per page

Denormalization

- Transforming **normalized** relations into **unnormalized** physical record specifications
- **Benefits:**
 - Can improve performance (speed) by reducing number of table lookups (i.e. *reduce number of necessary join queries*)
- **Costs (due to data duplication)**
 - Wasted storage space
 - Data integrity/consistency threats
- **Common denormalization opportunities**
 - One-to-one relationship
 - Many-to-many relationship with attributes
 - Reference data (1:N relationship where 1-side has data not used in any other relationship)

Partitioning

- **Horizontal Partitioning:** Distributing the rows of a table into several separate files.
 - Useful for situations where different users need access to different rows
 - Three types: Key Range Partitioning, Hash Partitioning, or Composite Partitioning
- **Vertical Partitioning:** Distributing the columns of a table into several separate files
 - Useful for situations where different users need access to different columns
 - The primary key must be repeated in each file
- Combinations of Horizontal and Vertical Partitioning often correspond with User Schemas (user views)

Partitioning

- **Advantages of Partitioning:**

- Records used together are grouped together
- Each partition can be optimized for performance
- Security, recovery
- Partitions stored on different disks: contention
- Take advantage of parallel processing capability

- **Disadvantages of Partitioning:**

- Slow retrievals across partitions
- Complexity

Data Replication

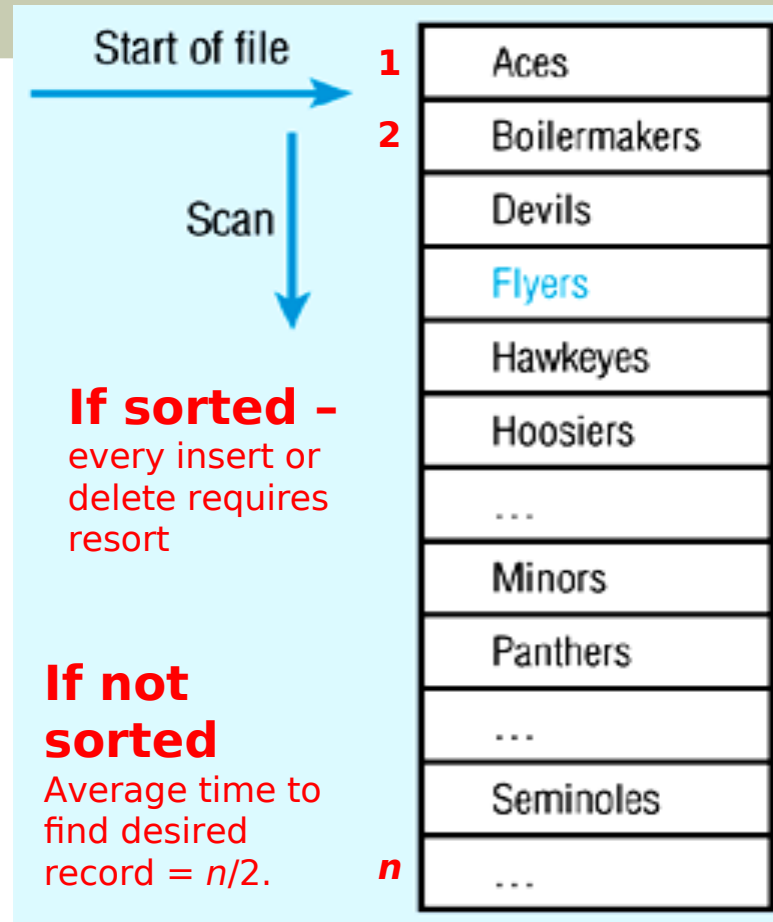
- Purposely storing the same data in multiple locations of the database
- Improves performance by allowing multiple users to access the same data at the same time with minimum contention
- Sacrifices data integrity due to data duplication
- Best for data that is not updated often

Designing Physical Files

- **Physical File:**
 - A named portion of secondary memory allocated for the purpose of storing physical records
- **Constructs to link two pieces of data:**
 - Sequential storage.
 - Pointers.
- **File Organization:**
 - How the files are arranged on the disk.
- **Access Method:**
 - How the data can be retrieved based on the file organization.

Figure 6-7 (a)
**Sequential
file
organization**

Records of the
file are stored
in sequence
by the
primary key
field values.



Indexed File Organizations

- **Index** – a separate table that contains organization of records for quick retrieval
- Primary keys are automatically indexed.
- Oracle has a CREATE INDEX operation, and MS ACCESS allows indexes to be created for most field types.
- Indexing approaches:
 - B-tree index,
 - Bitmap index,
 - Hash Index,
 - Join Index,

Clustering Files

- In some relational DBMSs, related records from different tables can be stored together in the same disk area.
- Useful for improving performance of join operations.
- **Primary key records of the main table are stored adjacent to associated foreign key records of the dependent table.**

Rules for Using Indexes

1. Use on larger tables
2. Index the primary key of each table
3. Index search fields (fields frequently in WHERE clause)
4. Fields in SQL ORDER BY and GROUP BY commands
5. When there are >100 values but not when there are <30 values
6. DBMS may have limit on number of indexes per table and number of bytes per indexed field(s)
7. Null values will not be referenced from an index
8. Use indexes heavily for non-volatile databases; limit the use of indexes for volatile databases

Why? Because modifications (e.g. inserts, deletes) require updates to occur in index files



Fundamentals of
**Database
Systems**

5th Edition

Elmasri / Navathe