Course :

Advanced Databases

Engineer cycle, 3rd year, Networks.
2024-2025

Dr. Dilekh tahar

tahar.dilekh@univ-batna2.dz

Department of Computer Science. Faculty of Mathematics and Computer Science. Batna 2 University

Chapter 4 : Database administration

1. Physical structure of a database
2. Logical structure of a database
3. Backup and restore of a database
4. Administration of an Oracle DB

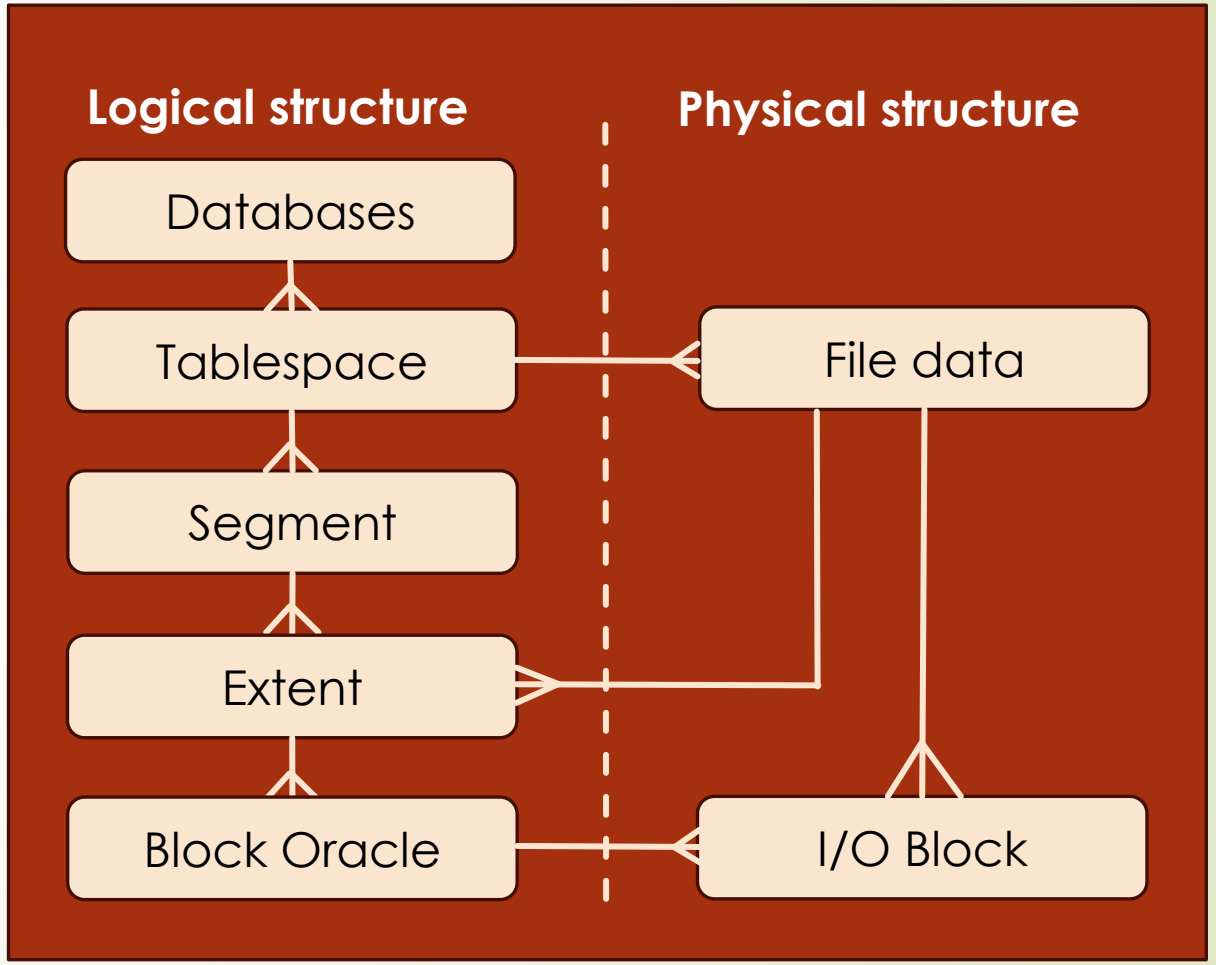
Ch 4 : Database administration / 1. Introduction

1. Physical Structure:

1. Data storage files at the operating system level.
2. Includes data files, recovery files, and control files.

1. Logical Structure:

1. Abstract representation for organizing and accessing data.
2. Includes tablespaces, segments, extents, and blocks.



Ch 4 : Database administration / 1. Physical structure of a database

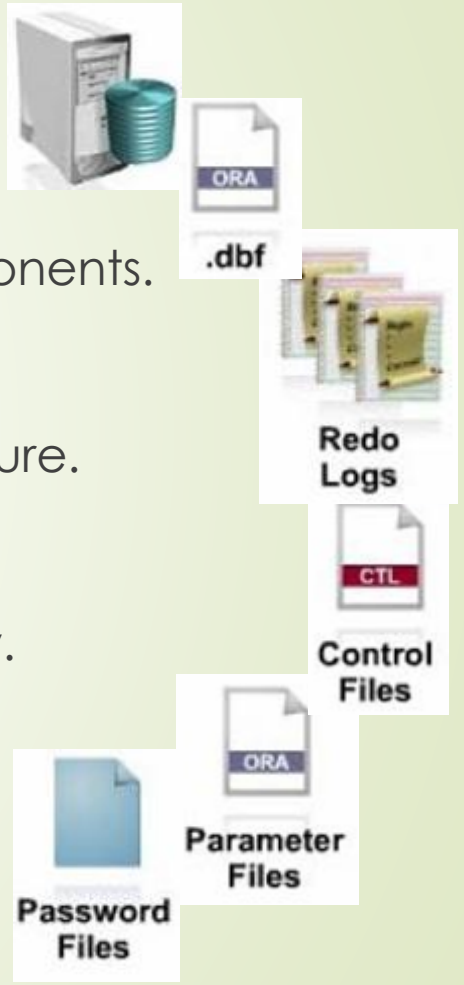
Physical structure

Objectives

- Comprehend the **physical storage structure** in DBMS databases.
- Explore the types of files used for storage and recovery.
- Understand the logical storage structure.

Ch 4 : Database administration / 1. Physical structure of a database

Physical Storage Structure



1. **Data Files:**
   - Store user objects, data dictionary, and other database components.
2. **Redo Log Files:**
   - Record all modifications to ensure data recovery in case of failure.
3. **Control Files:**
   - Maintain metadata about the database structure and integrity.
4. **Other Files:**
   - Parameter files, password files, and archived redo logs.

Ch 4 : Database administration / 1. Physical structure of a database

1. Data Files

- Contain all database objects (tables, indexes, etc.).
- They are used to store the **data dictionary** and database objects.
- These files are often very large.
- The data in the data buffer and the cache dictionary are retrieved from these files.

Ch 4 : Database administration / 1. Physical structure of a database

Data dictionary : is a set of tables and views where information on the logical and physical structure of the database is stored:

- Database users.
- Names and characteristics of objects stored in the database.
- Integrity constraints.
- Physical resources allocated to the database.

Data dictionary is created when the database is created, and is updated as objects are created.

Data dictionary used as a central source of information associated with a database.

Data dictionary has two components: (**a**) **base tables** (**b**) **data dictionary views**

Ch 4 : Database administration / 1. Physical structure of a database

a. Base tables

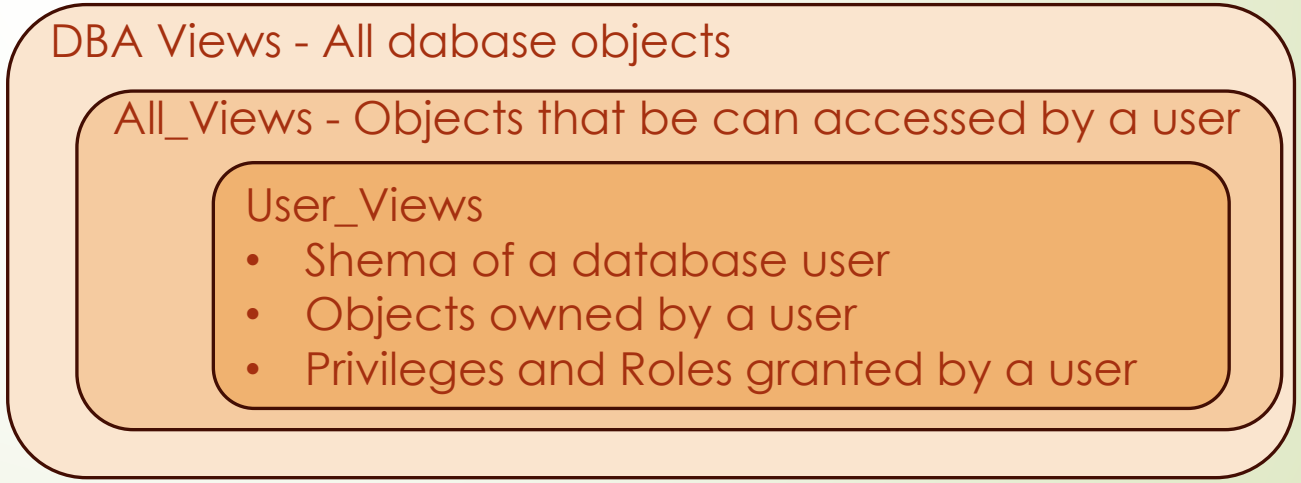
- These tables are created with a script (In Oracle, the script is the sql.bsq, it is stored in the ORACLE_HOME\rdbms\admin directory).
- The information stored in the base tables is **read** and **written** by the **Oracle server**.
- This information is **rarely accessed directly** by **users** because this information is normalized and stored in an encoded form.
- Users or database administrators should not use DML statements to update the base tables directly (**except** for the **audit trail table** when the audit feature is used).

Ch 4 : Database administration / 1. Physical structure of a database

b. Data dictionary views:

- Are views over the base tables (In Oracle, Created by the catalog.sql script).
- Simplify and summarize the information contained in the base tables.
- Store this information in a form that database users can easily read.
- These views allow the DBA to manage and administer the database.

Users can only access this dictionary by reading it through **three categories of views**:



Ch 4 : Database administration / 1. Physical structure of a database

b. Data dictionary views:

1. **USER_<views>**: Views on objects created by a user
   - For example, the USER_TABLES view contains information on tables belonging to a user.
2. **ALL_<views>:** Views on objects to which a user has access (not necessarily that he has created (through public or explicit obtaining of roles and privileges)
3. **DBA_<views>:** Views on all objects in the database of all users **accessible** by **administrators** (DBA) **or users** who **have the SELECT ANY TABLE** system privilege.

Ch 4 : Database administration / 1. Physical structure of a database

2. Redo Log Files

- Record every change made to the database.
- Used during recovery to restore committed data.
- Organized into **groups** for redundancy, Each redo log group can have multiple files, called **members**.

- Multiplexing recommended for fault tolerance.

  Example:

  - Group 1: Member 1 (/disk1/redo01.log), Member 2 (/disk2/redo01.log)
  - Group 2: Member 1 (/disk1/redo02.log), Member 2 (/disk2/redo02.log)

Ch 4 : Database administration / 1. Physical structure of a database

Redo Log File Operations

1. **Adding Redo Log Groups:**
   - ALTER DATABASE ADD LOGFILE GROUP 3 ('/path/log3.ora') SIZE 1000K;
2. **Adding Redo Log Members:**
   - ALTER DATABASE ADD LOGFILE MEMBER '/path/log3member.ora' TO GROUP 3;
3. **Removing Groups:**
   - ALTER DATABASE DROP LOGFILE GROUP 3;

- Minimum two groups required.
- Active groups cannot be removed.

Ch 4 : Database administration / 1. Physical structure of a database

Redo Log File Archiving

- **ARCHIVELOG Mode:** The database **archives redo logs after they are filled**. This mode is essential for ensuring full data recovery.
- **NOARCHIVELOG Mode:** The database **does not archive filled redo logs**, which **limits** the **recovery capabilities** to only the **most recent changes**. This mode is usually used in non-critical environments or where data recovery is less important.

Archiving Modes:

- **Manual**: DBA initiates archiving.
- **Automatic**: Enabled by LOG_ARCHIVE_START=TRUE.

Ch 4 : Database administration / 1. Physical structure of a database

3. Control Files

- Are used to **define the location** of disk components on the server.
- The **location of data files** and **redo logs appear** there.
- Are modified each time redo logs or data files are added or deleted.
- A DB requires at least one control file.

- Essential for:
  - Starting the database.
  - Restoring the database after a failure.

Ch 4 : Database administration / 1. Physical structure of a database

Multiplexing Control Files:

- Store multiple copies on different disks for redundancy.
- Change the SPFILE (Parameter Files):

```
ALTER SYSTEM SET control_files='/path/ctrl01.ctl', '/path/ctrl02.ctl' SCOPE=SPFILE;
```

- Shut down the database.
- Create additional control files: cp $HOME/ORADATA/u01/ctrl01.ctl $HOME/ORADATA/u02/ctrl02.ctl
- Restart the database
- Create additional control files: cp $HOME/ORADATA/u01/ctrl01.ctl $HOME/ORADATA/u02/ctrl02.ctl
- Restart the database

Ch 4 : Database administration / 1. Physical structure of a database

Multiplexing Control Files:

- With init.ora (Parameter files):
  - Shut down the database.
  - Copy the existing control file to another disk and change its name :

    cp control01.ctl …/DISK3/control02.ctl
  - Add the new control file to init.ora :

    CONTROL_FILES = (/DISK1/control01.ctl, /DISK3/control02.ctl)
  - Restart the database.

Control File Queries

Select * from V$CONTROLFILE;

Ch 4 : Database administration / 1. Physical structure of a database

4. Other Files

Encompasses all auxiliary files that support the operation, such as configuration, diagnostics, archiving, backup, and network communication.

These files are not part of the core database structures (datafiles, redo log files, and control files).

- **Parameter Files**: Contain configuration settings.
- **Password File**: Enables remote authentication of privileged database users.
- **Archived Redo Log Files**: Copies of filled redo log files.
- **Backup Files**: Copies of datafiles, control files, and redo log files.
- **Network Configuration Files**: Define how the database communicates.
- **Temp Files**: Support temporary operations.

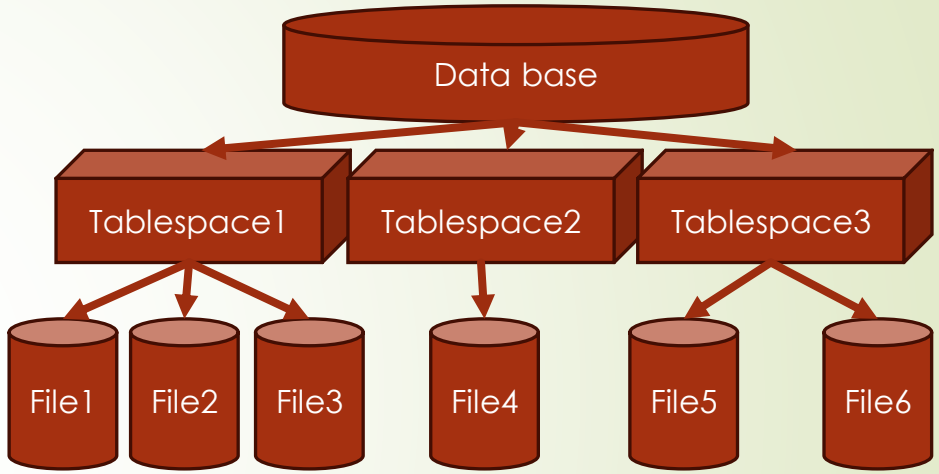
Ch 4 : Database administration / 2. Logical structure of a database

Overview of Logical Storage

- **Logical Units:** Abstract representations for managing physical storage.
- **Key Components:**
  1. **Tablespaces:** Logical storage units mapping to physical data files.
  2. **Segments:** Allocated storage for specific database objects.
  3. **Extents:** Contiguous storage blocks within segments.
  4. **Data Blocks:** Smallest storage units, tied to physical blocks.

Ch 4 : Database administration / 2. Logical structure of a database

1. Tablespaces



- Largest logical storage unit in Oracle.
- Comprised of one or more physical data files.
- **Key Features**:
  - Manages data allocation, user quotas, and availability.
  - Supports **online** and **offline** statuses (except SYSTEM tablespaces and those with active rollback segments).
  - Can switch between **read/write** and **read-only** modes.

Ch 4 : Database administration / 2. Logical structure of a database

1. **Tablespaces**
   - **Using Tablespaces**
     1. **Space Management:**
        - Allocate quotas to users : Controls how much space users can consume to avoid resource conflicts.
        - Optimize input/output by distributing data across multiple devices.
     2. **Backup and Maintenance:**
        - Support partial backups by toggling tablespace statuses.
     3. **Use Cases:**
        - Store static data in read-only tablespaces.

Ch 4 : Database administration / 2. Logical structure of a database

1. **Tablespaces**
   - **Types of Tablespaces**
     1. **SYSTEM Tablespace:**
        - Essential for database operations.
        - Stores data dictionary, stored procedures, and packages.
        - Should not store user data.
     2. **Non-SYSTEM Tablespaces:**
        - Offer flexibility for managing data and indexes.
        - Used for rollback segments and temporary operations.

Ch 4 : Database administration / 2. Logical structure of a database

1. **Tablespaces**
   - **Tablespace Management**
     - **Creating a Tablespace:**

```
CREATE TABLESPACE tbs_user
DATAFILE 'c:/oracle/oradata/userdata01.dbf' SIZE 100M
AUTOEXTEND ON NEXT 5M MAXSIZE 200M;
```

   - **Types of Tablespaces:**
     1. **Temporary:** Used for sorting and intermediate operations.

```
CREATE TEMPORARY TABLESPACE temp
TEMPFILE '/disk2/temp01.dbf' SIZE 500M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 4M;
```

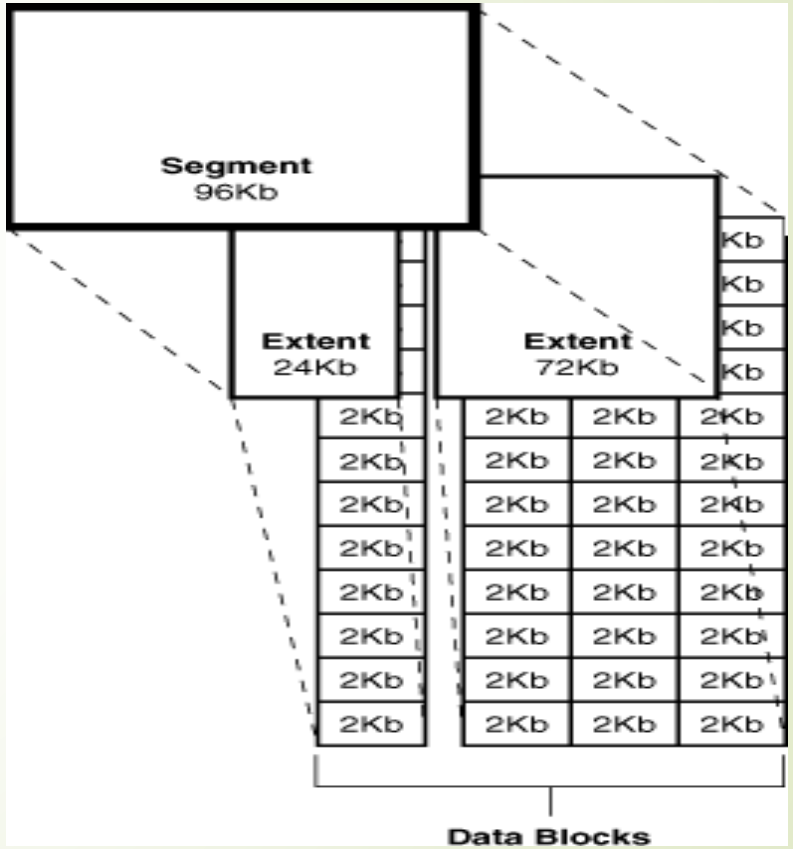
     2. **Undo:** Dedicated to undo segments.

```
CREATE UNDO TABLESPACE undo1
DATAFILE 'u01/oradata/undo101.dbf' SIZE 40M;
```

Ch 4 : Database administration / 2. Logical structure of a database

2. Segment

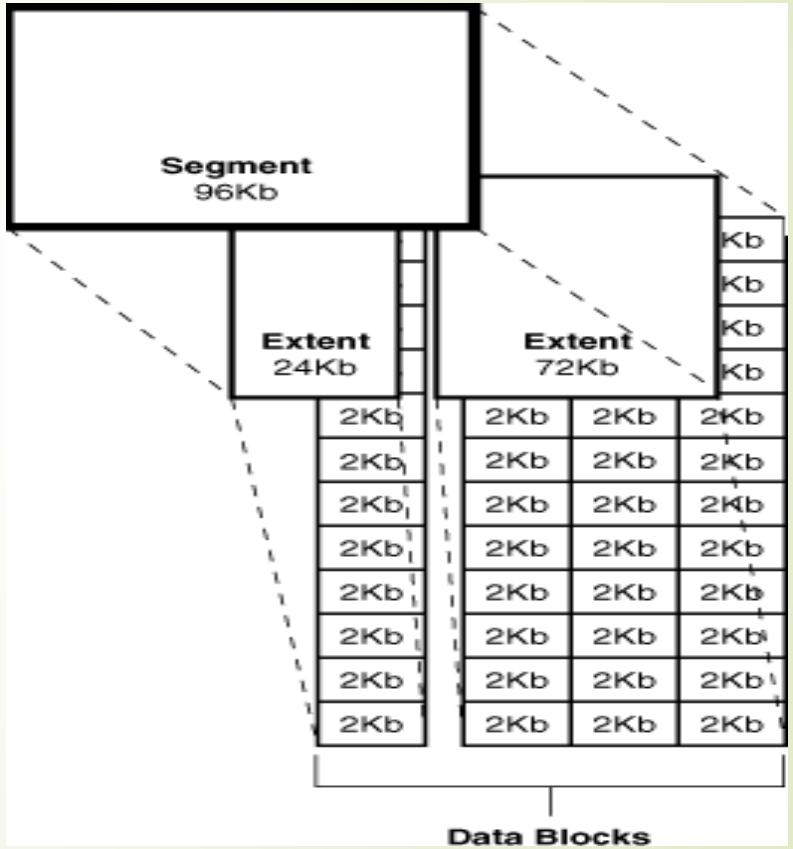
- Represents a specific database object (e.g., table, index).
- Made up of one or more extents.
- Types of Segments:
  - Data Segment: Stores table data.
  - Index Segment: Stores index data.
  - Temporary Segment: For temporary operations.
  - Undo Segment: Stores undo data for rollback.



Ch 4 : Database administration / 2. Logical structure of a database

3. Extent

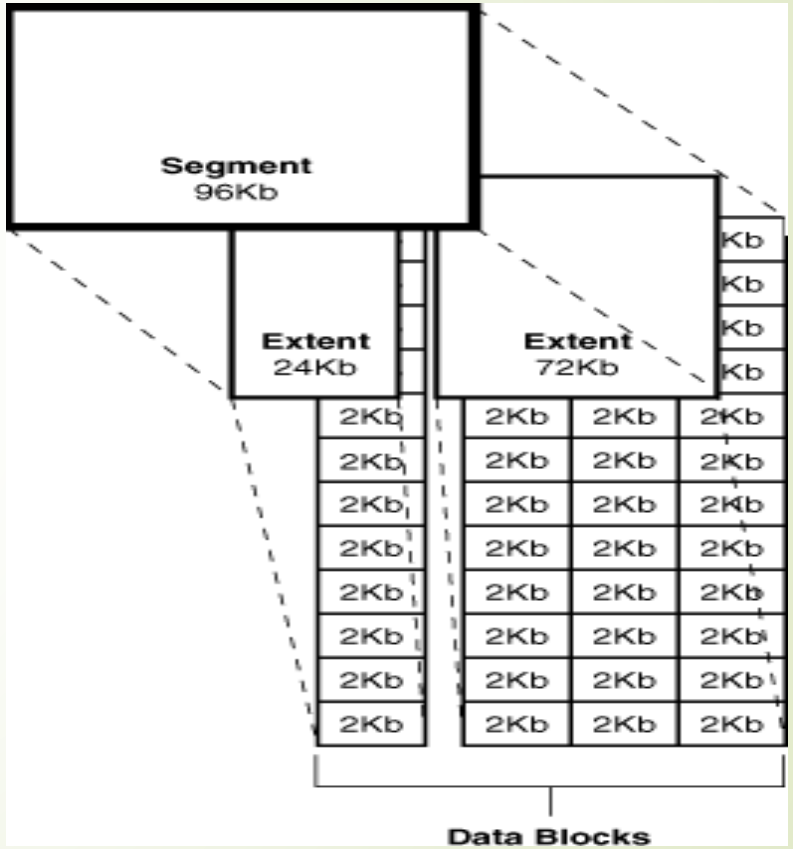
- Contiguous blocks of storage allocated for a segment.
- **Purpose**: To provide additional space for growing objects.
- Allocated dynamically as needed.
- Prevents fragmentation by ensuring contiguous storage.



Ch 4 : Database administration / 2. Logical structure of a database

4. Block

- Smallest unit of data storage in an Oracle database.
- Defined by the **DB_BLOCK_SIZE** parameter.
- Stores rows of data, header information, and overhead.
- **Key Features**:
  - Each block belongs to a specific tablespace.
  - Contains row data and metadata.



Ch 4 : Database administration / 3. Backup and restore of a database

Objectives:

- Understand essential DBMS recovery utilities.
- Explore backup mechanisms, journaling, checkpoints, and recovery techniques.

Topics covered:

- Backup mechanisms
- Journaling tools
- Checkpoints
- Recovery techniques

Ch 4 : Database administration / 3. Backup and restore of a database

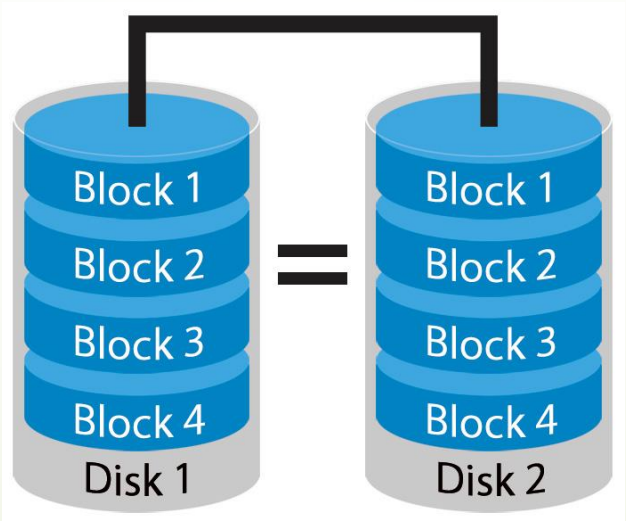
1. Backup mechanism

Purpose:

- Ensure data durability through redundant copies.
- Restore database state after hardware or software failures.

Key Types:

- Mirror Copies
- RAID Arrays



Show a simple database with backup copies and RAID structure

Ch 4 : Database administration / 3. Backup and restore of a database

1. Backup mechanism - Mirror copies

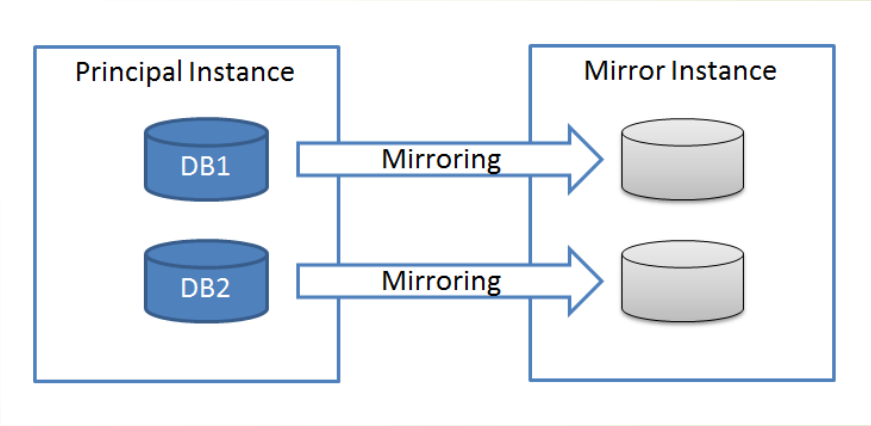
- Exact copies of data maintained in separate locations.

Example:

Data mirrored to two disks to prevent data loss in case of disk failure.

Benefits:

- High availability and fault tolerance.



Primary storage with two mirrored copies (DB1, DB2) stored on separate drives.

Ch 4 : Database administration / 3. Backup and restore of a database

1. Backup mechanism - RAID

RAID (Redundant Array of Independent Disks):

- Combines multiple disks to improve reliability and performance.

Common RAID Levels:

- **RAID 1**: Mirroring (high redundancy).
- **RAID 5**: Striping with parity (balances redundancy and storage- **See appendix**).
- **RAID 10**: Combination of RAID 1 and RAID 0

Ch 4 : Database administration / 3. Backup and restore of a database

1. Backup mechanism - RAID

- **RAID 1**: Mirroring (high redundancy).
  - **Usage**: Enhances data reliability.
  - **How it works**: Duplicates data on two disks. If one disk fails, the data is available on the other.
  - **Drawback**: Doubles storage cost.
  - **When to use**: For **critical applications** needing **high reliability** (e.g., transaction logs).

Ch 4 : Database administration / 3. Backup and restore of a database

1. **Backup mechanism - RAID**

    2. **RAID 5** (Striping with Parity)

        - **Usage**: Balances performance, storage efficiency, and fault tolerance.
        - **How it works**: Stripes data across disks and stores **parity** (error-checking) information on each disk.
        - **Drawback**: Requires at least three disks; slower write performance due to **parity calculations**.
        - **When to use**: For read-heavy workloads requiring fault tolerance.

Ch 4 : Database administration / 3. Backup and restore of a database

1. Backup mechanism - RAID

3. **RAID 10**: Combination of RAID 1 and RAID 0

RAID 0 (Striping)

- **Usage**: Improves performance.
- **How it works**: Splits data across multiple disks, allowing simultaneous read/write operations.
- **Drawback**: No fault tolerance; a single disk failure results in complete data loss.
- **When to use:** When **performance is critical**, and **data can be recreated** (e.g., caching or temporary storage).

Ch 4 : Database administration / 3. Backup and restore of a database

1. Backup mechanism - RAID

3. RAID 10: Combination of RAID 1 and RAID 0

- **Mirroring (RAID 1):**
  - Each piece of data is duplicated across a pair of disks. This ensures redundancy because if one disk fails, the data is still available on its mirror.
- **Striping (RAID 0):**
  - The mirrored pairs are then striped (data is spread across multiple mirrored pairs).
  - Striping improves performance by allowing simultaneous read and write operations across multiple disks.

Ch 4 : Database administration / 3. Backup and restore of a database

1. Backup mechanism - RAID

3. **RAID 10**: Combination of RAID 1 and RAID 0

- **Example**: Imagine a **RAID 10** system with **4 disks**:

Step 1 (Mirroring):	**Step 2 (Striping):**
• **Disk 1** is mirrored to **Disk 2**. • **Disk 3** is mirrored to **Disk 4**. So, **Disk 2** contains a copy of **Disk 1**, and **Disk 4** contains a copy of **Disk 3**.	•Data is striped across the **two** mirrored pairs. For example: • Part **A** of the data is written to **Disk 1** and mirrored on **Disk 2**. • Part **B** of the data is written to **Disk 3** and mirrored on **Disk 4**.

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

Purpose:

- Record transaction changes in a log (journal) for recovery purposes.

Types:

- **Physical Journal**: Logs exact data changes.
- **Logical Journal**: Logs high-level operations (e.g., “insert,” “delete”).

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

1. Physical Journal

Tracks low-level data block changes directly.

Example:

Before and after images of data blocks recorded during a transaction.

Benefits:

Suitable for exact, byte-level recovery of data.

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

1. Physical Journal - Example :

- A database page has 4KB (4096 bytes).
- Suppose the page ID is Page123, and it contains a list of integers: [10, 20, 30, 40].
- Each integer is stored in 4 bytes, so the structure is as follows:

Byte Offset	Data
0–3	10
4–7	20
8–11	30
12–15	40

Physical Log Entry Format

{
PageID: Page123,
Offset : 8-11,
BeforeImage: 30,
AfterImage: 50
}

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

1. Physical Journal – Example structure of a physical log entry :

Field	Description
Log Sequence Number (LSN)	12345 (unique ID for the log entry)
Transaction ID	T5678
Operation Type	UPDATE
Timestamp	2024-11-21 10:30:45
File ID	001 (file containing the modified page)
Page ID	0001 (specific page modified)
Pre-Image	Original state of the page before the update
Post-Image	Updated state of the page after the modification
Checksum	A hash for data integrity verification

Checksum : ensures the integrity of the log entry. A checksum or hash of the log entry to detect corruption.

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

1. Physical Journal – Usage of the Physical Log :

- **Undo Logging:** Uses the pre-image to revert changes if a transaction is rolled back.
- **Redo Logging:** Uses the post-image to reapply changes during crash recovery.
- **Checkpointing:** Ensures that log records up to a specific LSN have been safely written to disk.

By recording changes at the physical level, the physical log provides fine-grained control over data recovery, ensuring the integrity and consistency of the database system.

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

2. Logical Journal

Logs higher-level operations.

Example:

Logical journal entry:

INSERT INTO Employees (EmployeeID, Name, Salary) VALUES (3, 'Charlie', 70000);

Benefits:

Suitable for reconstructing data changes more abstractly, helpful for logical errors.

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

2. Logical Journal - Example structure of a physical log entry :

Field	Description
Log Sequence Number	45678 (unique identifier for this log entry)
Transaction ID	T7890
Operation Type	INSERT
Schema Name	public
Table Name	employees
Key Values	{employee_id: 1234}
New Values (After-Image)	{name: "Alice", salary: 75000}
Timestamp	2024-11-21 11:00:00
Checksum	A hash for integrity verification

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

2. Logical Journal - Key characteristics of logical logs :

- **Operation-Level Abstraction:** Logical logs focus on describing what the database operation is, not the underlying storage changes.
- **Application-Specific:** The log entries are tied closely to the schema and data model of the application (e.g., table names, column names).
- **Compactness:** Logical logs can be smaller than physical logs for certain workloads since they store only the logical operations instead of entire data pages or blocks.
- **Portability:** They can be replayed on a different database (e.g., for replication) as they do not depend on the physical structure of the storage.

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

- **UNDO and REDO Rules in Journaling**

1. **UNDO rule** : Ensures that the effects of uncommitted transactions are completely reversed if the system crashes or if a transaction is explicitly aborted. In logical logging, this involves reversing the operations recorded in the log.

- **Steps for UNDO in Logical Logs:**

1. Locate the log entries for the transaction that needs to be undone.
2. Process the log entries in **reverse order**, starting from the most recent (last logged operation).
3. Apply the **logical inverse** of each operation to revert the change.

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

- **UNDO and REDO Rules in Journaling**

1. UNDO rule Example for UNDO:

- **Initial State of Table:**

Logical Log Entries for Transaction T1:

1. LSN 1: UPDATE employees SET salary = 55000 WHERE Employee_ID = 101;
2. LSN 2: INSERT INTO employees (Employee_ID, Name, Salary) VALUES (103, 'Carol', 70000);

Employee_ID	Name	Salary
101	Alice	50000
102	Bob	60000

Crash Occurs Before Commit:

- Transaction T1 was not committed, so the changes must be undone.

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

- **UNDO and REDO Rules in Journaling**
  1. **UNDO rule Example for UNDO:**

UNDO Steps:

1. **Reverse the second operation:**
   - **Log Entry**: INSERT INTO employees (Employee_ID, Name, Salary) VALUES (103, 'Carol', 70000);
   - **Inverse Operation**: DELETE FROM employees WHERE Employee_ID = 103;
2. **Reverse the first operation:**
   - **Log Entry**: UPDATE employees SET salary = 55000 WHERE Employee_ID = 101;
   - **Inverse Operation**: UPDATE employees SET salary = 50000 WHERE Employee_ID = 101; (restore the original value).

Employee_ID	Name	Salary
101	Alice	50000
102	Bob	60000

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

- **UNDO and REDO Rules in Journaling**

2. **REDO Rule (Recovery)** : Ensures that the effects of committed transactions are fully applied to the database even if the system crashes before the changes reach the stable storage. In logical logging, this involves replaying the logical operations recorded in the log.

- **Steps for UNDO in Logical Logs:**

1. Locate the log entries for all committed transactions.
2. Process the log entries in **forward order**, starting from the earliest (oldest log entry).
3. Reapply each operation to ensure all changes are reflected in the database.

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

- **UNDO and REDO Rules in Journaling**

2. REDO rule - Example for REDO:

- **Initial State of Table:**

Logical log entries for transaction t2:

1. LSN 3: UPDATE employees SET salary = 62000 WHERE Employee_ID = 102;
2. LSN 4: INSERT INTO employees (Employee_ID, Name, Salary) VALUES (104, 'Dave', 45000);

Employee_ID	Name	Salary
101	Alice	50000
102	Bob	60000

Crash Occurs After Commit:

- Transaction T2 was committed, so the changes must be redone.

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

- **UNDO and REDO Rules in Journaling**

1. UNDO rule Example for UNDO:

REDO Steps:

1. **Replay the first operation:**
   - **Log Entry**: UPDATE employees SET salary = 62000 WHERE Employee_ID = 102;
   - **Operation**: Updates Bob's salary to 62000.;
2. **Replay the second operation:**
   - **Log Entry**: INSERT INTO employees (Employee_ID, Name, Salary) VALUES (104, 'Dave', 45000);
   - **Inverse Operation**: Adds Dave to the table.

Employee_ID	Name	Salary
101	Alice	50000
102	Bob	62000
104	Dave	45000

Ch 4 : Database administration / 3. Backup and restore of a database

2. Journaling Tool

UNDO and REDO Rules in Journaling

Key Differences Between UNDO and REDO in Logical Logs:

Aspect	UNDO	REDO
Purpose	Reverts changes made by uncommitted transactions.	Reapplies changes made by committed transactions.
Order of Logs	Processes logs in reverse order (most recent first).	Processes logs in forward order (oldest first).
Operations	Applies the logical inverse of logged operations.	Re-executes the original operations in the log.
When Used	During rollback or crash recovery for uncommitted transactions.	During crash recovery to ensure all committed changes are applied.

These rules ensure the database's **atomicity** (UNDO) and **durability** (REDO), key components of the **ACID** properties.

Ch 4 : Database administration / 3. Backup and restore of a database

3. Checkpointing

- Create points of consistency in the database log to speed up recovery.

- **Process**:
  - Periodically save current state to allow for faster recovery.

- **Checkpoint Benefits**:
  - Minimizes rollback work.
  - Limits REDO operations to after the last checkpoint.

Ch 4 : Database administration / 3. Backup and restore of a database

3. Checkpointing

- **Example**: **Log file with Checkpoints**
- Assume the following events occurred in a database system:

1. **T1**: Transaction 1 starts and inserts a row into Employees.
2. **T2**: Transaction 2 starts and updates a row in Departments.
3. A **checkpoint** is written after some transactions are committed.
4. A crash occurs.

Ch 4 : Database administration / 3. Backup and restore of a database

3. Checkpointing

- **Example: Log file with Checkpoints**

[LOG START]

[T1 BEGIN] **-- Transaction 1 begins.**

[T1 INSERT Employees (ID=3, Name='Charlie', Salary=70000)]

[T2 BEGIN] **-- Transaction 2 begins.**

[T2 UPDATE Departments SET Budget=50000 WHERE DeptID=2]

[T1 COMMIT] **-- Transaction 1 commits.**

[CHECKPOINT] **-- Checkpoint written here. All prior changes are flushed to disk.**

[T2 DELETE Employees WHERE ID=3] -- Transaction 2 deletes a row.

[T2 ROLLBACK] **-- Transaction 2 rolls back.**

[LOG END]

Ch 4 : Database administration / 3. Backup and restore of a database

3. Checkpointing

- **Example**: **Log file with Checkpoints**

  Recovery process:

  1. Identify checkpoint:

  - The **[CHECKPOINT]** marker indicates all changes logged **before this point** have already been applied to the database.
  - The recovery process can **ignore all entries before the checkpoint**.

  2. Process entries after the checkpoint:

  - Replay committed transactions and undo incomplete or rolled-back transactions:
    - **[T2 DELETE Employees WHERE ID=3]**: Ignored because Transaction 2 was rolled back.
    - **[T1 COMMIT]**: Already persisted, so no action is needed.

Ch 4 : Database administration / 4. Administration of an Oracle DB

Administering an Oracle Database

- Manage Oracle database instances effectively.
- Cover key administration tasks like startup, shutdown, parameter management, and access control.

Objectives

- Create and manage initialization parameter files.
- Start and stop database instances.
- Modify database operational modes.
- Restrict and manage connections.
- Monitor diagnostic files.

Ch 4 : Database administration / 4. Administration of an Oracle DB

Administrator Accounts

1. **SYS Account**:
   - Created during Oracle installation.
   - Has all system privileges.
   - Default password: change_on_install (must be changed).
2. **SYSTEM Account**:
   - Created during Oracle installation.
   - Has all system privileges.
   - Default password: manager (must be changed).

Ch 4 : Database administration / 4. Administration of an Oracle DB

Initialization Parameter Files

- **Purpose:** Contain settings necessary to start an Oracle database instance.

Types of Files:

1. **PFILE**: Static text file (e.g., initSID.ora).
   - Editable using a text editor.
   - Changes take effect after a restart.
2. **SPFILE**: Binary server file (e.g., spfileSID.ora).
   - Managed by Oracle.
   - Changes are persistent across restarts.

- **Key Parameters in PFILE:**

- instance_name=ORCL
- db_name=ORCL
- control_files=('/path/control01.ctl', '/path/control02.ctl')
- db_block_size=8192

Ch 4 : Database administration / 4. Administration of an Oracle DB

Managing Parameter Files

- **Creating a PFILE:**
  1. Copy and edit an existing init.ora file.
  2. Use the following command to create a PFILE:

```
CREATE PFILE='$ORACLE_HOME/dbs/initORCL.ora' FROM SPFILE;
```

- **Creating an SPFILE from a PFILE:**
  1. Convert using SQL commands:

```
CREATE SPFILE='$ORACLE_HOME/dbs/spfileORCL.ora'
FROM PFILE='$ORACLE_HOME/admin/dbname/pfile/initORCL.ora';
```

- **Modifying Parameters:**
  1. Use the ALTER SYSTEM command:

```
ALTER SYSTEM SET undo_tablespace='UNDO2' SCOPE=BOTH;
```

Ch 4 : Database administration / 4. Administration of an Oracle DB

Database Startup Process

- **Startup Modes:**

1. NOMOUNT:
- Reads initialization files (PFILE or SPFILE).
- Allocates memory and starts background processes.

2. MOUNT:
- Reads control files.
- Locates but does not open data files.

3. OPEN:
- Opens data files and online redo logs.
- Makes the database available for users.

- **Startup Command:**

```
STARTUP [FORCE] [RESTRICT] [PFILE=name] [OPEN | MOUNT | NOMOUNT];
```

Ch 4 : Database administration / 4. Administration of an Oracle DB

Restricting Database Access

- **Restrict Access at Startup:**

  STARTUP RESTRICT;

- **Enable Restricted Session:**

  ALTER SYSTEM ENABLE RESTRICTED SESSION;

- **Disable Restricted Session:**

  ALTER SYSTEM DISABLE RESTRICTED SESSION;

- **Kill a Session:**

  ALTER SYSTEM KILL SESSION 'sid,serial#';

Ch 4 : Database administration / 4. Administration of an Oracle DB

Shutting Down the Database

- **Shutdown Modes:**

1. NORMAL:	◦ Waits for all users to disconnect. ◦ Ensures database consistency.
2. TRANSACTIONAL:	◦ Waits for active transactions to complete. ◦ Does not allow new connections.
3. IMMEDIATE:	◦ Rolls back active transactions. ◦ Disconnects users without waiting.
4. ABORT:	◦ Forcibly stops the instance. ◦ Requires recovery at the next startup.

- **Shutdown Command:**

```
SHUTDOWN [NORMAL | IMMEDIATE | TRANSACTIONAL | ABORT];
```

Ch 4 : Database administration / 4. Administration of an Oracle DB

Managing Operational Modes

- **Change Database Modes:**

1. Switch from NOMOUNT to MOUNT:

   ALTER DATABASE MOUNT;

1. Switch from MOUNT to OPEN:

   ALTER DATABASE OPEN;

1. Open in Read-Only Mode:

   ALTER DATABASE OPEN READ ONLY;

1. Switch to Read/Write Mode:

   ALTER DATABASE OPEN READ WRITE;

Ch 4 : Database administration / 4. Administration of an Oracle DB

- **Troubleshooting Startup Issues**
  1. **Common Errors:**
     - Missing parameter files.
     - Corrupted control files or data files.
  2. **Steps to Resolve:**
     - Use the SHUTDOWN command before retrying.
     - Specify the correct parameter file:
     - STARTUP PFILE='$ORACLE_HOME/admin/dbname/pfile/initORCL.ora';
  3. **Use FORCE Mode:**
     - Stops any running instance before starting a new one:
     - STARTUP FORCE;

Ch 4 : Database administration / 4. Administration of an Oracle DB

Key Administrative Commands

1. View Sessions:

```
SELECT * FROM V$SESSION;
```

2. Check Initialization Parameters:

```
SHOW PARAMETERS;
```

3. List Data Files:

```
SELECT * FROM DBA_DATA_FILES;
```

4. List Control Files:

```
SELECT * FROM V$CONTROLFILE;
```

Ch 4 : Database administration / 4. Administration of an Oracle DB

Summary

- Key DBA tasks include managing parameter files, starting/stopping instances, and controlling access.
- Oracle provides tools for fine-grained control of operational modes.
- Always ensure database consistency during shutdowns to avoid recovery overhead.

Ch 4 : Database administration

Questions ?

Ch 4 : Database administration / Appendix

Appendix:

Parity:

Parity data in RAID refers to a mathematical technique used to ensure data redundancy and fault tolerance.

Example

Imagine a RAID 5 system with three disks: Disk 1, Disk 2, and Disk 3**.**

Step 1: Suppose we want to store the numbers **5** and **7**.

Data is split: Disk 1: **5** Disk 2 : **7** Disk 3: **2** **Parity = 5 ⊕ 7 = 2** (where ⊕ is XOR)	Recovering Data After a Failure: Suppose Disk 1 fails (losing 5) Missing data = Parity ⊕ Remaining data **5** = **2** ⊕ **7**