UNIT-5

Physical File	Logical File
It occupies the portion of memory. It contains the original data.	It does not occupy memory space. It does not contain data.
A physical file contains one record format.	It can contain up to 32 record formats.
It can exist without a logical file.	It cannot exist without a physical file.
If there is a logical file for the physical file, the physical file cannot be deleted until and unless we delete the logical file.	If there is a logical file for a physical file, the logical file can be deleted without deleting the physical file.
CRTPF command is used to make such an object.	CRTLF command is used to make such an object.
Physical files represent the real data saved on an iSeries system and describe how the data is to be displayed to or retrieved from a program.	The logical file represents one or multiple physical files. It also has a description of the records found in one or multiple physical files.
If there is a logical file for a physical file, the physical file can’t be deleted until and unless we delete the Logical file.	If there is a logical file for a physical file, the logical file can be deleted without deleting the physical file.

Logical Storage Views: viewed by users are a collection of files organized within directories and storage volumes.

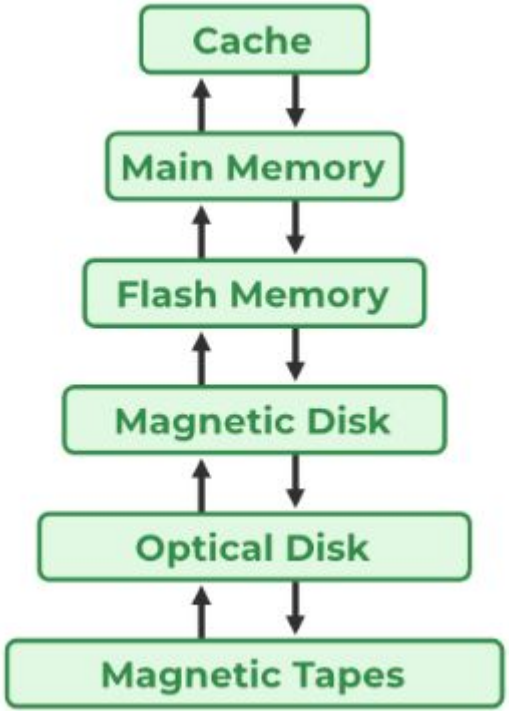
- The logical file structure is independent of its physical implementation.
- Logical file structure “ignores”.

Physical storage allocations: records can be stored in separate file locations. Data access methods and Data encoding methods.

Physical Storage Views: a collection of physical storage locations organized as a linear address space.

- The file is subdivided into records.
- The record usually contains information about a single customer, things such as a product in inventory, or an event.
- Records are divided into fields.
- Fields are individual units of data.

Memory structures



Memory Hierarchy (Volatile to Non-Volatile)

1. Cache Memory (L1, L2, L3): Extremely fast, small, CPU-level memory for frequently used data blocks, reducing disk I/O.
2. Main Memory (RAM): Holds the database's working set, including buffer pools, indexes, and temporary tables, offering quick access but losing data on power loss.
3. Secondary Storage (SSD/HDD): Persistent, non-volatile storage for large databases, slower than RAM but essential for long-term data.
4. Tertiary Storage: Archival, offline storage (tapes, optical disks) for massive, rarely accessed data.

Key DBMS Memory Components

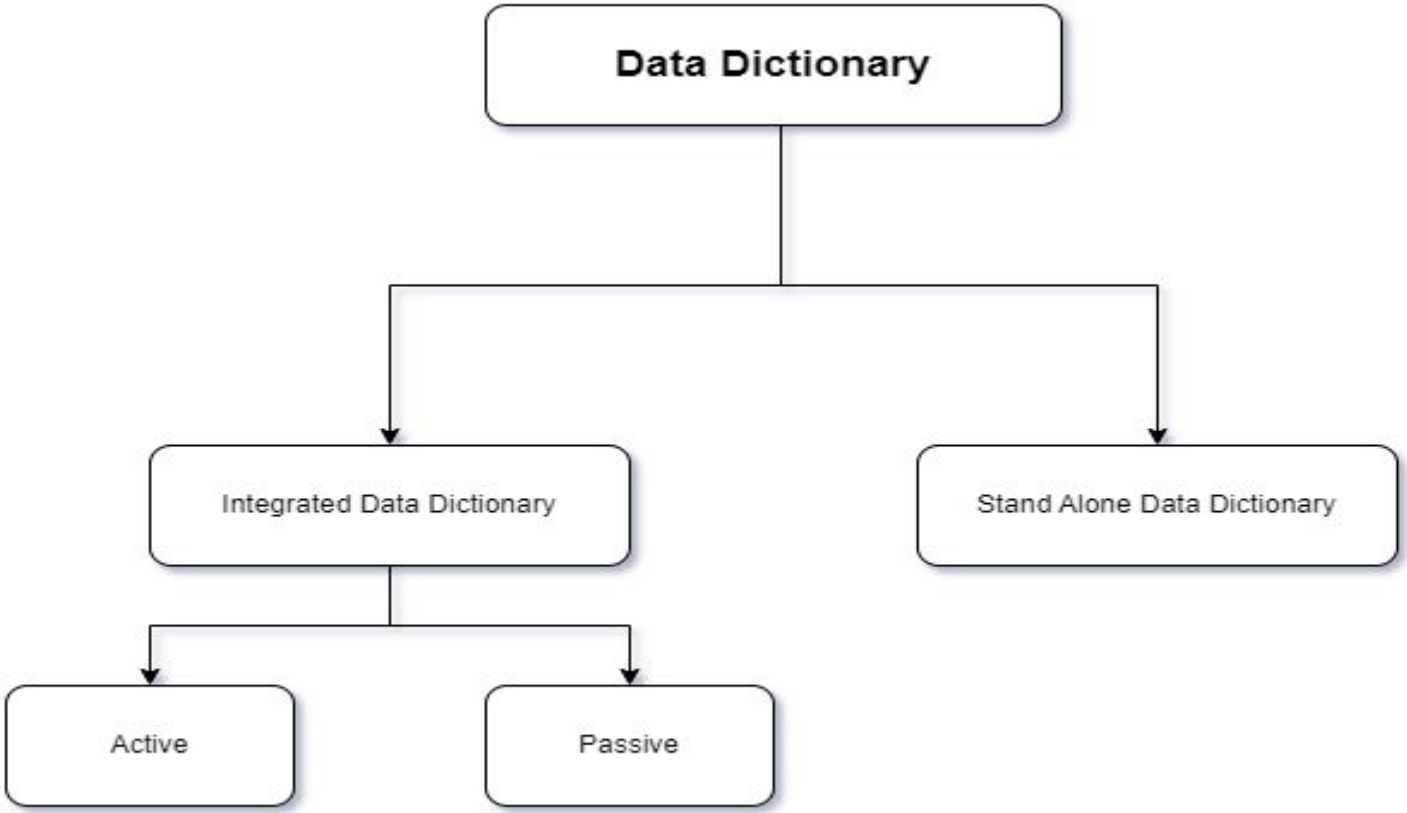
- System Global Area (SGA): Shared memory for an Oracle instance, holding shared SQL areas, data blocks, and control info for all processes.
- Program Global Area (PGA): Private memory for each process, containing private SQL areas, session info, and work areas (sorting, hashing).
- Buffer Manager/Pool: Manages the RAM buffer, caching data pages from disk; uses policies (like LRU) to decide what to evict.

Data Dictionary

The data dictionary consists of two words, data, which represents data collected from several sources, and dictionary, which represents where this data is available. The data dictionary is an important part of the relational database because it provides additional information about the relationship between several tables in the database. A data dictionary in a DBMS helps users manage data in an orderly and orderly manner, thereby preventing data redundancy.

Some of the advantages of using a data dictionary are:

- In DBMS, the data model provides very little information about the database, so the data dictionary is very important to get the right knowledge about the entities, relationships, and attributes that exist in the data model.
- The data dictionary provides consistency by reducing data redundancy in data collection and use among different team members.
- Data dictionaries provide structured analysis and design tools by implementing data standards. Data standards are sets of rules that govern the collection, recording, and presentation of data.
- Using a data dictionary helps define the naming convention used in the model.


Data Dictionary
Integrated Data Dictionary
Stand Alone Data Dictionary
Active
Passive

SQL Triggers

A trigger is a special stored procedure in a database that automatically executes when specific events (like `INSERT`, `UPDATE`, or `DELETE`) occur on a table. Triggers help automate tasks, maintain data consistency, and record database activities. Each trigger is tied to a particular table and runs without manual execution.

Types of Triggers

Triggers can be classified based on the event that fires them and their scope:

By Triggering Event

- DML Triggers (Data Manipulation Language): Fired in response to `INSERT`, `UPDATE`, or `DELETE` statements on a table or view.
- DDL Triggers (Data Definition Language): Fired in response to DDL events like `CREATE`, `ALTER`, or `DROP` table/view/etc. They are useful for auditing schema changes or preventing certain modifications.
- Logon Triggers: Fired when a user session is established with the database server. They can be used for auditing login activity or controlling the number of sessions.

SQL Triggers

A trigger is a special stored procedure in a database that automatically executes when specific events (like `INSERT`, `UPDATE`, or `DELETE`) occur on a table. Triggers help automate tasks, maintain data consistency, and record database activities. Each trigger is tied to a particular table and runs without manual execution.

Step 1: Create Main Table

```
CREATE TABLE users (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    email VARCHAR(100),
    updated_at TIMESTAMP
);
```

Step 2: Create Trigger

This trigger automatically updates the `updated_at` field whenever the user record is modified.

```
CREATE TRIGGER update_timestamp
BEFORE UPDATE ON users
FOR EACH ROW
BEGIN
   SET NEW.updated_at = CURRENT_TIMESTAMP;
END;
```

Step 3: Insert Sample Data

```
INSERT INTO users (id, name, email) VALUES (1, 'Amit', 'amit@example.com');
```

Output

id	name	email	updated_at
1	Amit	amit@example.com	**NULL** (not updated yet)

Step 4: Update a Record

```
UPDATE users SET email = 'amit_new@example.com' WHERE id = 1;
```

id	name	email	updated_at
1	Amit	amit_new@example.com	**2025-10-27 12:45:23** (current timestamp auto-updated)