Database system

Prof. Suchitra Patil

Database Systems(DBS)

- DBS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database systems are used to manage collections of data that are:
  - Highly valuable
  - Relatively large
  - Accessed by multiple users and applications, often at the same time.
- A modern database system is a complex software system whose task is to manage a large, complex collection of data.
- Databases touch all aspects of our lives

Database Applications Examples

Enterprise Information

Sales: customers, products, purchases

Accounting: payments, receipts, assets

Human Resources: Information about employees, salaries, payroll taxes.

Manufacturing: management of production, inventory, orders, supply chain.

Banking and finance

customer information, accounts, loans, and banking transactions.

Credit card transactions

Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data

Universities: registration, grades

Database Applications Examples (Cont.)

Airlines: reservations, schedules

Telecommunication: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards

Web-based services

- Online retailers: order tracking, customized recommendations
- Online advertisements

Document databases

Navigation systems: For maintaining the locations of varies places of interest along with the exact routes of roads, train systems, buses, etc.

Main Characteristics of the Database Approach

- Self-describing nature of a database system(Metadata): A DBMS **catalog** stores the *description* of the database. The description is called **meta-data**). This allows the DBMS software to work with different databases.
- Insulation between programs and data: Called **program-data independence**. Allows changing data storage structures and operations without having to change the DBMS access programs.

Main Characteristics of the Database Approach

- Persistant Data: stored and retained unless deleted by explicit request.
- Data Abstraction: A **data model** is used to hide storage details and present the users with a *conceptual view* of the database.
- Access Flexibility and Security: Support of multiple views of the data. Each user may see a different view of the database, which describes *only* the data of interest to that user.

Main Characteristics of the Database Approach

- Sharing of data and multiuser transaction processing : allowing a set of concurrent users to retrieve and to update the database. Concurrency control within the DBMS guarantees that each **transaction** is correctly executed or completely aborted.

Disadvantages of the file processing system

- Data redundancy and inconsistency: duplicate files, inconsistency, wastage of storage

e.g. student with two majors

DB- single repository, consistent

- Difficulty in accessing data: specific application programs, manual extraction of data

e.g. university clerk need data of students in specific region

DB-Sharing of data among multiple users.

Disadvantages of the file processing system

- Data isolation: data scattered in various files, different formats, format specific application program writing in difficult.

DB- well organized, structured single repository

- Integrity problems: difficult to enforce integrity constraints in various files on different data items (code is required to be added to each application program)

DB- can be done easily using DDL.

- Security problems: enforcing security constraints is difficult.

DB- can easily do using views.

Database Users

Users may be divided into those who actually

Actors on the Scene : use and control the content

Workers Behind the Scene :who enable the database to be developed and the DBMS software to be designed and implemented

Users

Actors on the scene:

–**Database administrators:** responsible for authorizing access to the database, for co-ordinating and monitoring its use, acquiring software, and hardware resources, controlling its use and monitoring efficiency of operations.

–**Database Designers:** responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

–**End-users:** they use the data for queries, reports and some of them actually update the database content.

Categories of End-users

- **Casual** : access database occasionally when needed.

E.g. middle or high level managers

- **Naïve or Parametric** : they make up a large section of the end-user population. They use previously well-defined functions in the form of "canned transactions" against the database. Examples are bank-tellers or reservation clerks who do this activity for an entire shift of operations.

Categories of End-users

- **Sophisticated** : these include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities. Many use tools in the form of software packages that work closely with the stored database to meet their complex requirements.
- **Stand-alone** : mostly maintain personal databases using ready-to-use packaged applications. An example is a tax program user that creates his or her own internal database.

University Database Example

- In this text we will be using a university database to illustrate all the concepts
- Data consists of information about:
  - Students
  - Instructors
  - Classes
- Application program examples:
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts

View of Data

A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.

A major purpose of a database system is to provide users with an abstract view of the data.

- Data models
  - A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
- Data abstraction
  - Hide the complexity of data structures to represent data in the database from users through several levels of data abstraction.

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semi-structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Relational Model

All the data is stored in various tables.

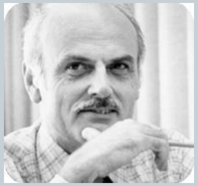
Example of tabular data in the relational model

Columns

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Rows

(a) The *instructor* table



Ted Codd
Turing Award 1981

A Sample Relational Database

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Levels of Abstraction

Physical level: describes how a record (e.g., instructor) is stored.

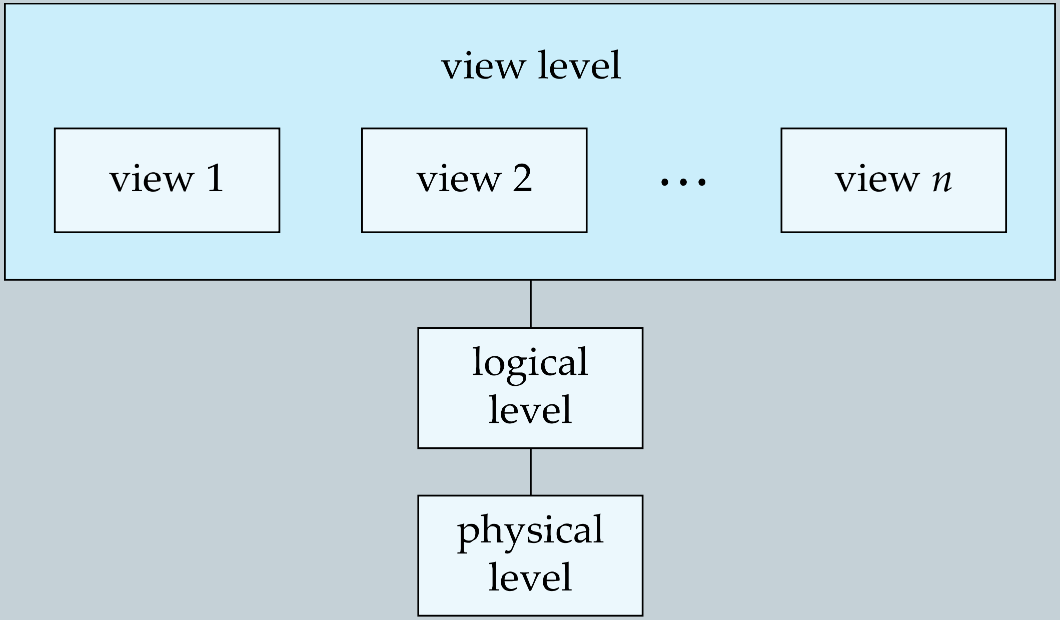
Logical level: describes data stored in database, and the relationships among the data.

```
type instructor = record
        ID : string;
        name : string;
        dept_name : string;
        salary : integer;
    end;
```

View level: application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

View of Data

An architecture for a database system



Instances and Schemas

Similar to types and variables in programming languages

Logical Schema – the overall logical structure of the database

- Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
  - Analogous to type information of a variable in a program

Physical schema – the overall physical structure of the database

Instance – the actual content of the database at a particular point in time

- Analogous to the value of a variable

DBMS ARCHITECTURE

- The *logical DBMS architecture*

- The *physical DBMS architecture*

DBMS ARCHITECTURE

- The *logical DBMS architecture*

The logical architecture deals with the way data is stored and presented to users.

- The *physical DBMS architecture*

DBMS ARCHITECTURE

- The *logical DBMS architecture*

- The *physical DBMS architecture*

The physical architecture is concerned with the s/w components that make up a DBMS.

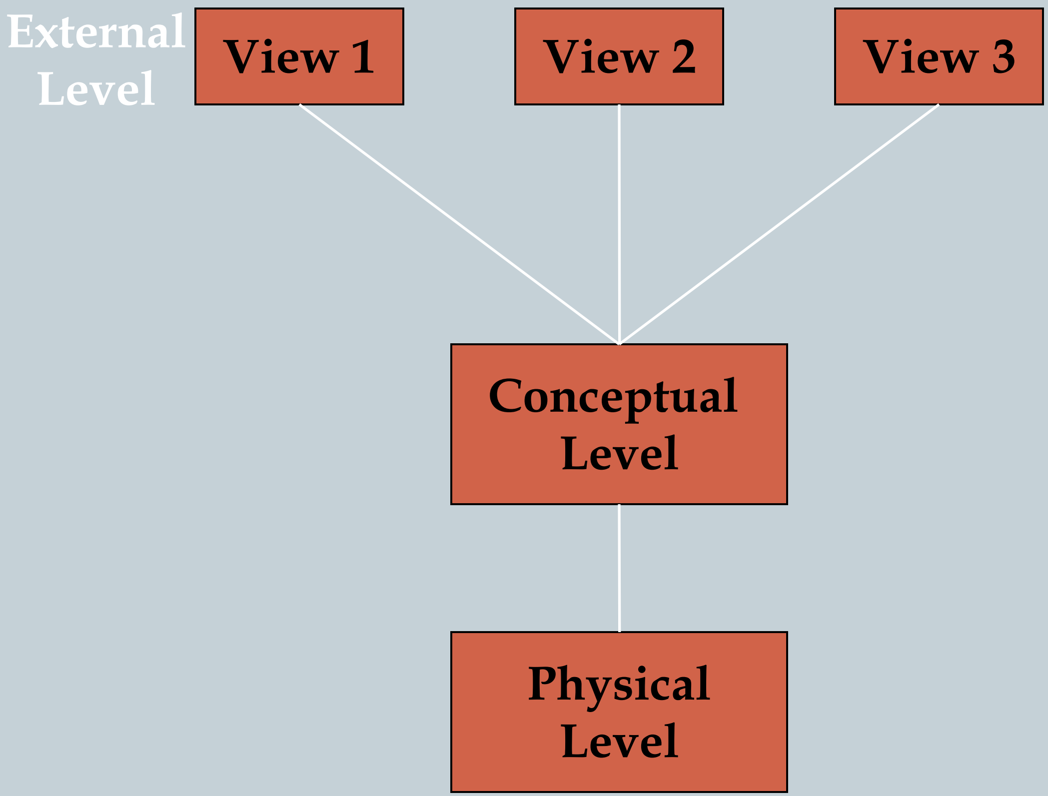
Three Level Architecture of DBMS

A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data is stored and maintained.

- *External or View Level*
- *Conceptual Level*
- *Internal or Physical Level*

Three Level Architecture of DBMS

continue…



Three Level Architecture of DBMS

continue…

External Level

Sales Officer

View 1
Item_Name
Price

Inventory Controller

View 2
Item_Name
Stock

Conceptual Level

Conceptual

Item_Number	**Character (6)**
Item_Name	**Character(30)**
Price	**Numeric(5,2)**
Stock	**Numeric(4)**

Physical Level

Physical

Stored_Item	**Length=50**
Item #	**Type = Byte(6), offset = 0, Index = Ix**
Name	**Type = Byte(30), offset = 6**
Price	**Type = Byte(8), offset = 36**
Stock	**Type = Byte(4), offset = 44**

External or View Level

This level is closest to the users and is concerned with the way in which the data is viewed by individual users. Most of the users are not concerned with all the information contained in the database. Instead they need only a part of the database relevant to them. The system provides many views for the same database.

External or View Level

- Highest level of abstraction of database.
- Allows to see only the data of interest to them.
- Users – Application programmers or end-users.
- Any no. of external views – external schema.

Conceptual Level

This level of abstraction describes what data are actually stored in the database. It also describes the relationships existing among data. At this level, the database is described logically in terms of simple data-structures. The users of this level are not concerned with how these logical data structures will be implemented at the physical level, rather they just are concerned about what information is to be kept in the database.

Conceptual Level

continue…

- The sum total of DBMS users view.
- Describes what data are actually stored in the database (ie,all the records and relationships included in the database).
- mapping between the conceptual schema and the internal schema

Conceptual Level

continue…

- The conceptual view is a representation of the entire information content of the database in a form that is some what abstract in comparison with the way in which the data is physically stored.

Conceptual Level

continue…

- The conceptual view is defined by means of the conceptual schema, which includes the definition of each of the various types of conceptual records and the mapping between the conceptual schema and the internal schema.

Internal or Physical Level

The internal level is closest to physical storage.

This level is also termed as physical level.

It describes how the data are actually stored on the storage medium.

At this level, complex low-level data structures are described in detail.

Internal or Physical Level

- Lowest level of abstraction.
- Describes how the data are physically stored.
- Internal view – internal schema (not only defines the various types of stored record but also specifies what indexes exists, how files are represented, etc.)

Data Independence

The ability to modify a schema definition in one level without affecting a scheme definition in the next higher level is called ***DATA INDEPENDENCE***

- ***Physical Data Independence***
- ***Logical Data Independence***

Physical Data Independence

It refers to the ability to modify the scheme followed at the physical level without affecting the scheme followed at the conceptual level.

The application programs remain the same even though the scheme at the physical level gets modified.

Modifications at the physical level are occasionally necessary in order to improve performance of the system.

Logical Data Independence

It refers to the ability to modify the conceptual scheme without causing any changes in the schemes followed at view levels.

The logical data independence ensures that the application programs remain the same.

Modifications at the conceptual level are necessary whenever logical structures of the database get altered because of some unavoidable reasons.

Physical & Logical and Data Independence

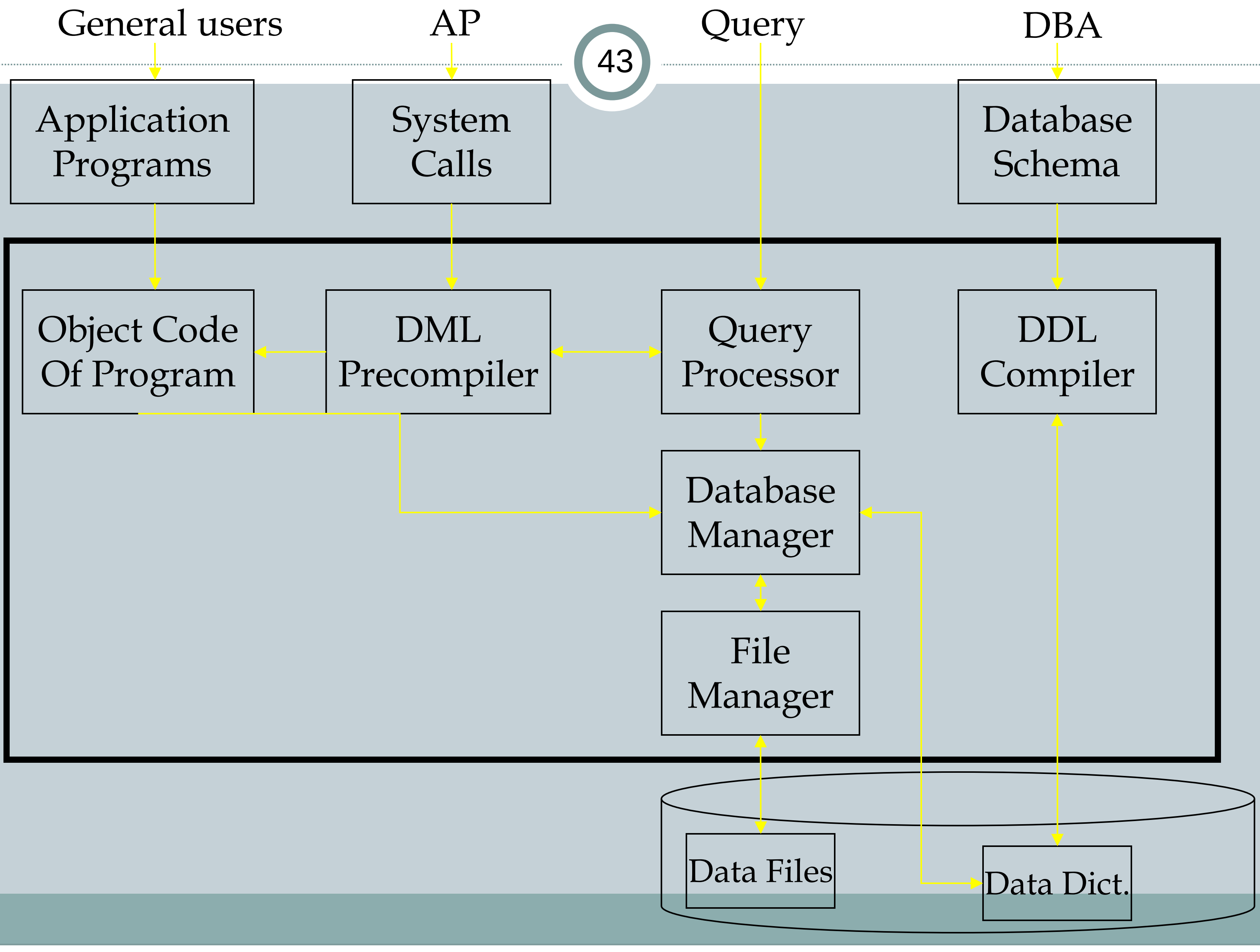
It is more difficult to achieve logical data independence than the physical data independence.

The reason being that the application programs are heavily dependent on the logical structure of the database.

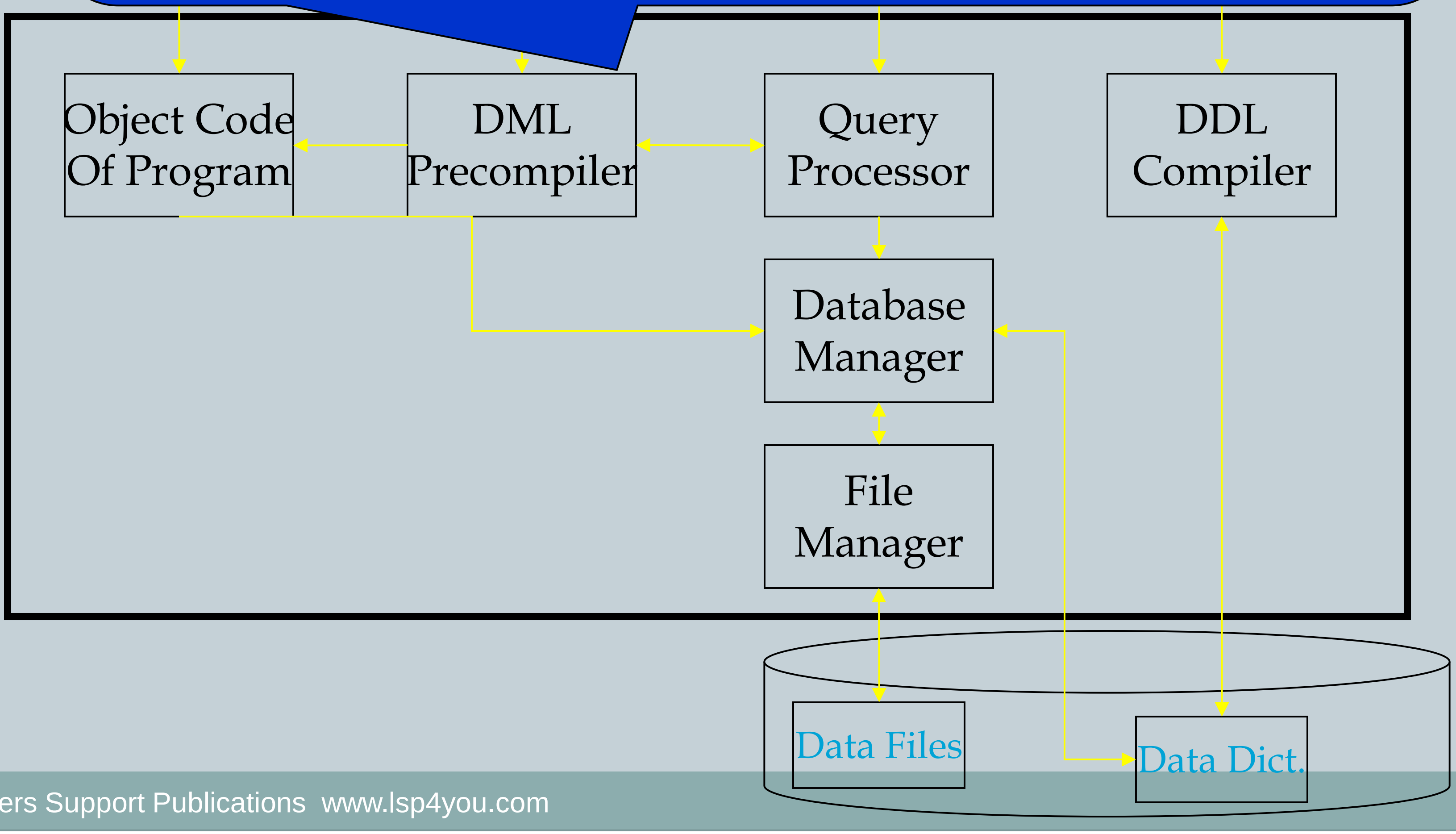
Physical DBMS Architecture

- Describes the software components used to enter and process data.
- How these s/w components are related and interconnected.

DBMS Structure

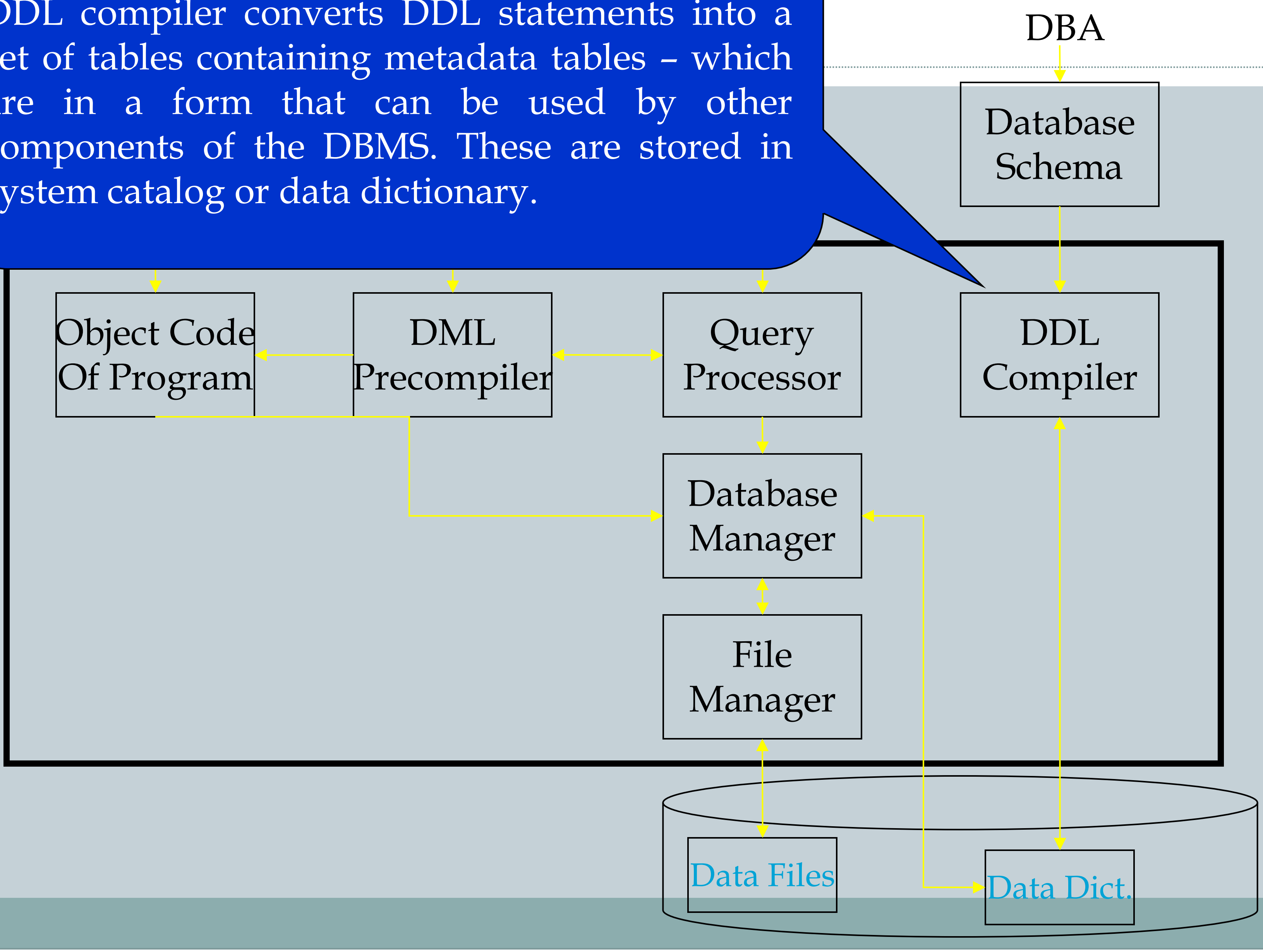


DDL – set of commands required to define the format of data.
DML – set of commands that modify, process data.
DML precompiler converts DML statements embedded in an application program to normal procedural calls in the host language. It interacts with the query processor in order to generate the appropriate code.

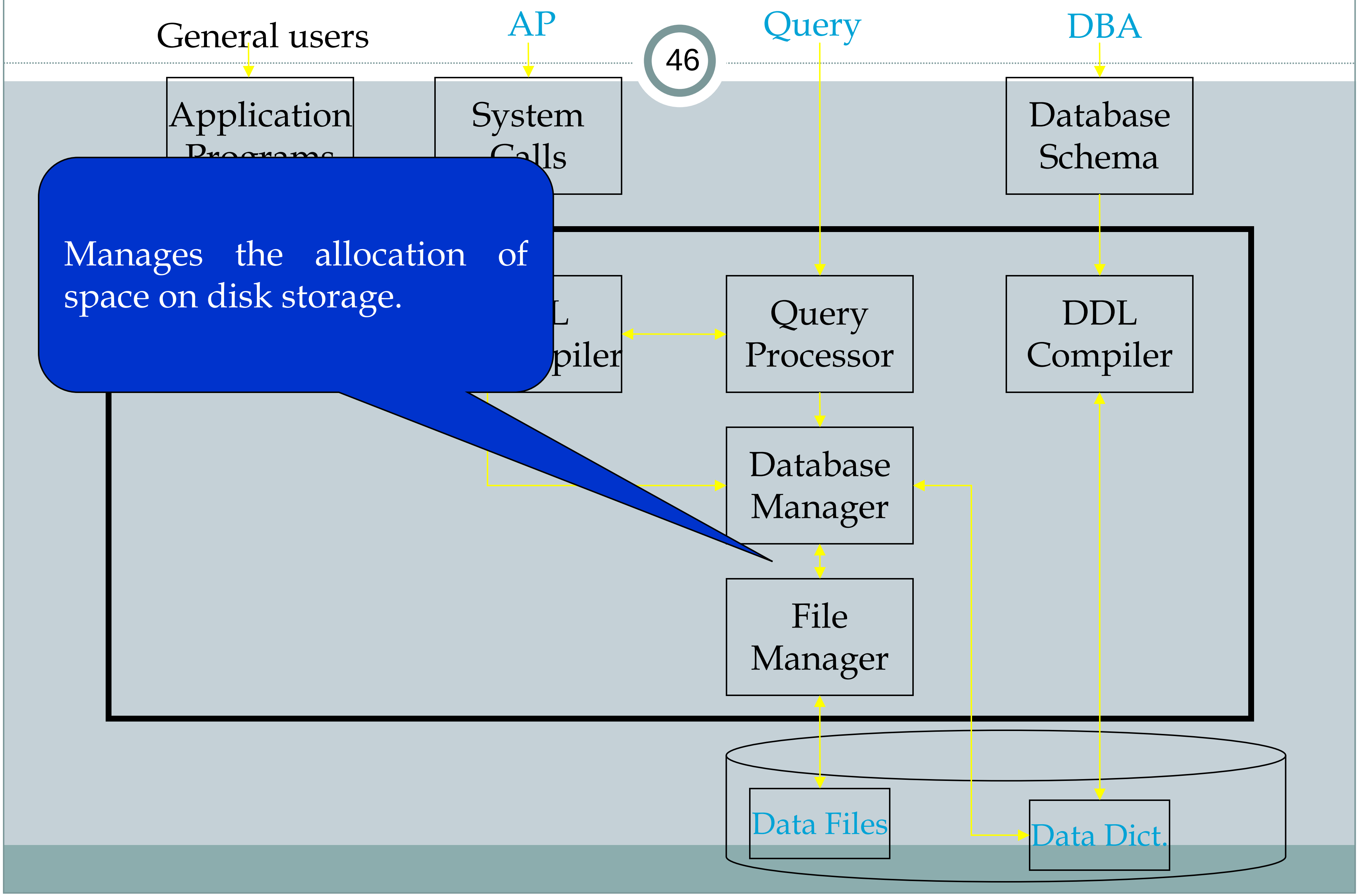


DBMS Structure

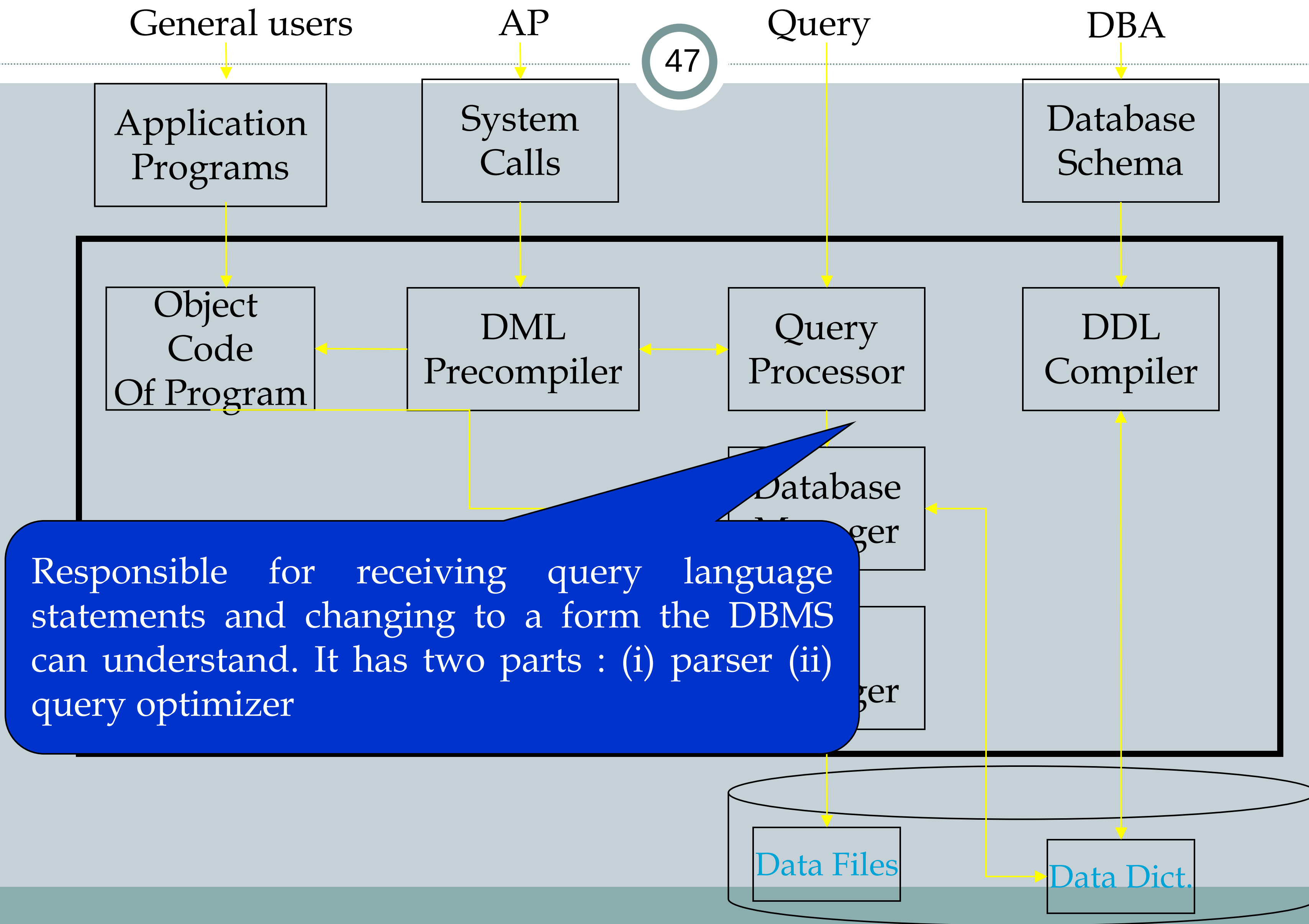
DDL compiler converts DDL statements into a set of tables containing metadata tables – which are in a form that can be used by other components of the DBMS. These are stored in system catalog or data dictionary.



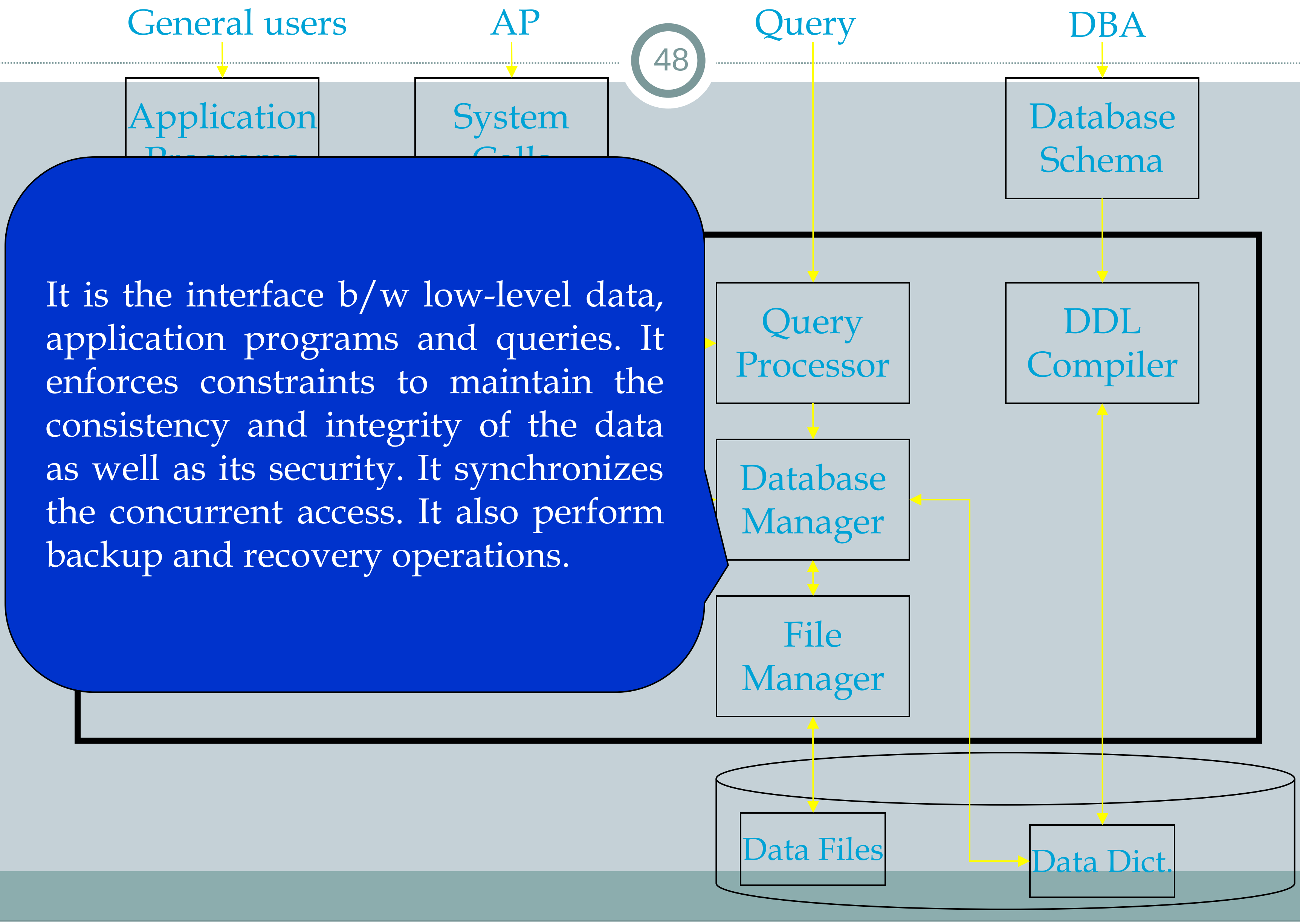
DBMS Structure



DBMS Structure

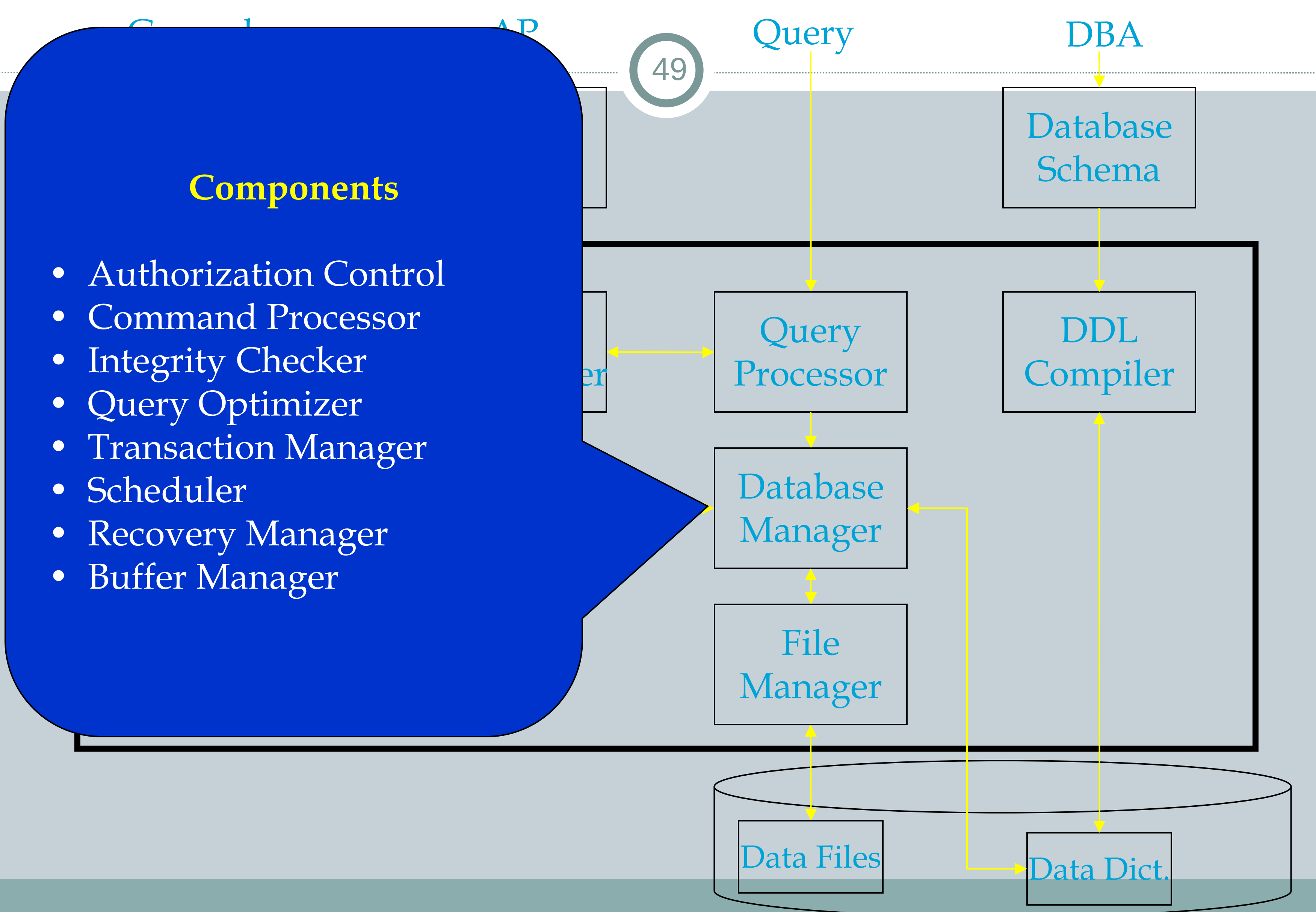


DBMS Structure

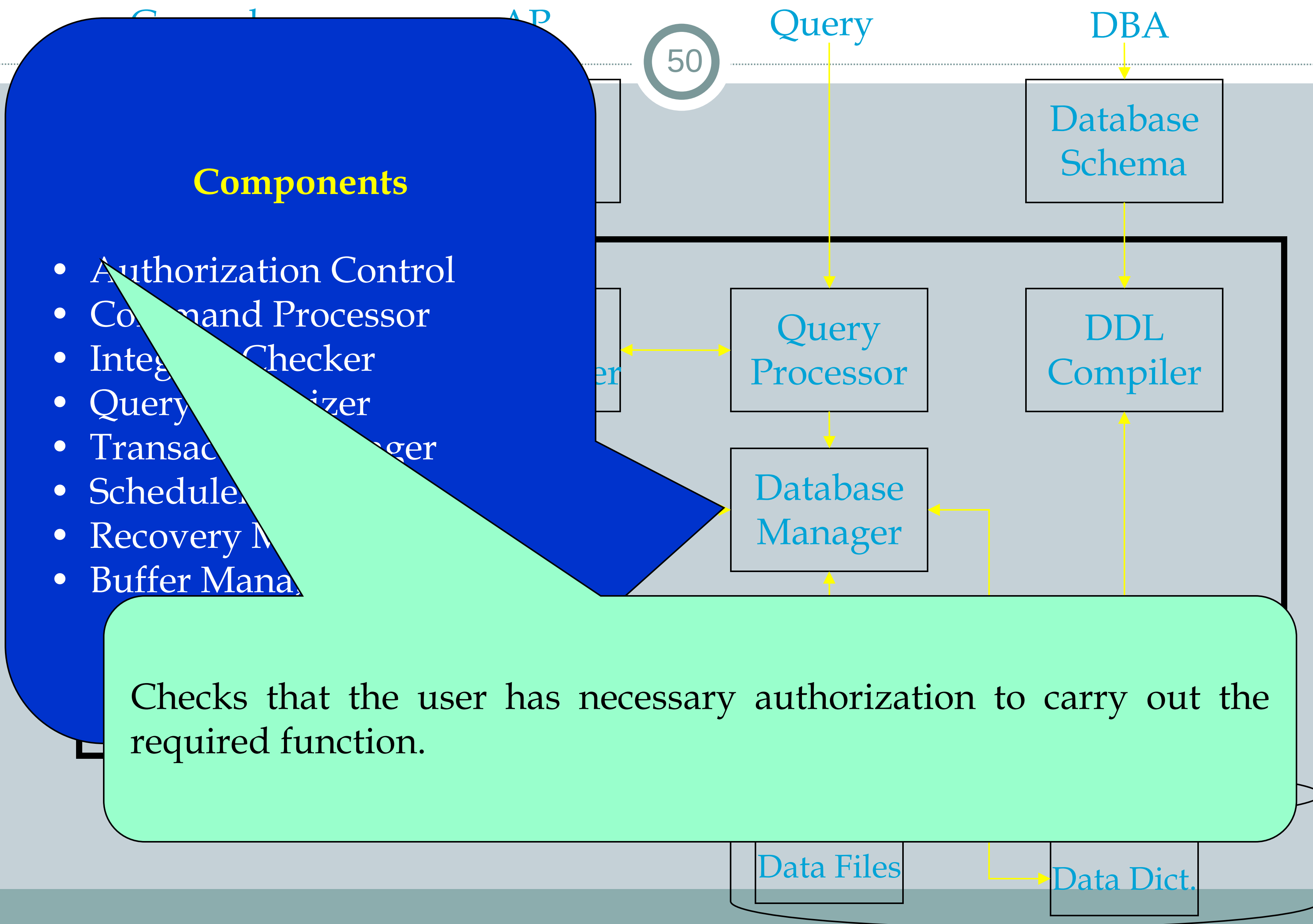


It is the interface b/w low-level data, application programs and queries. It enforces constraints to maintain the consistency and integrity of the data as well as its security. It synchronizes the concurrent access. It also perform backup and recovery operations.

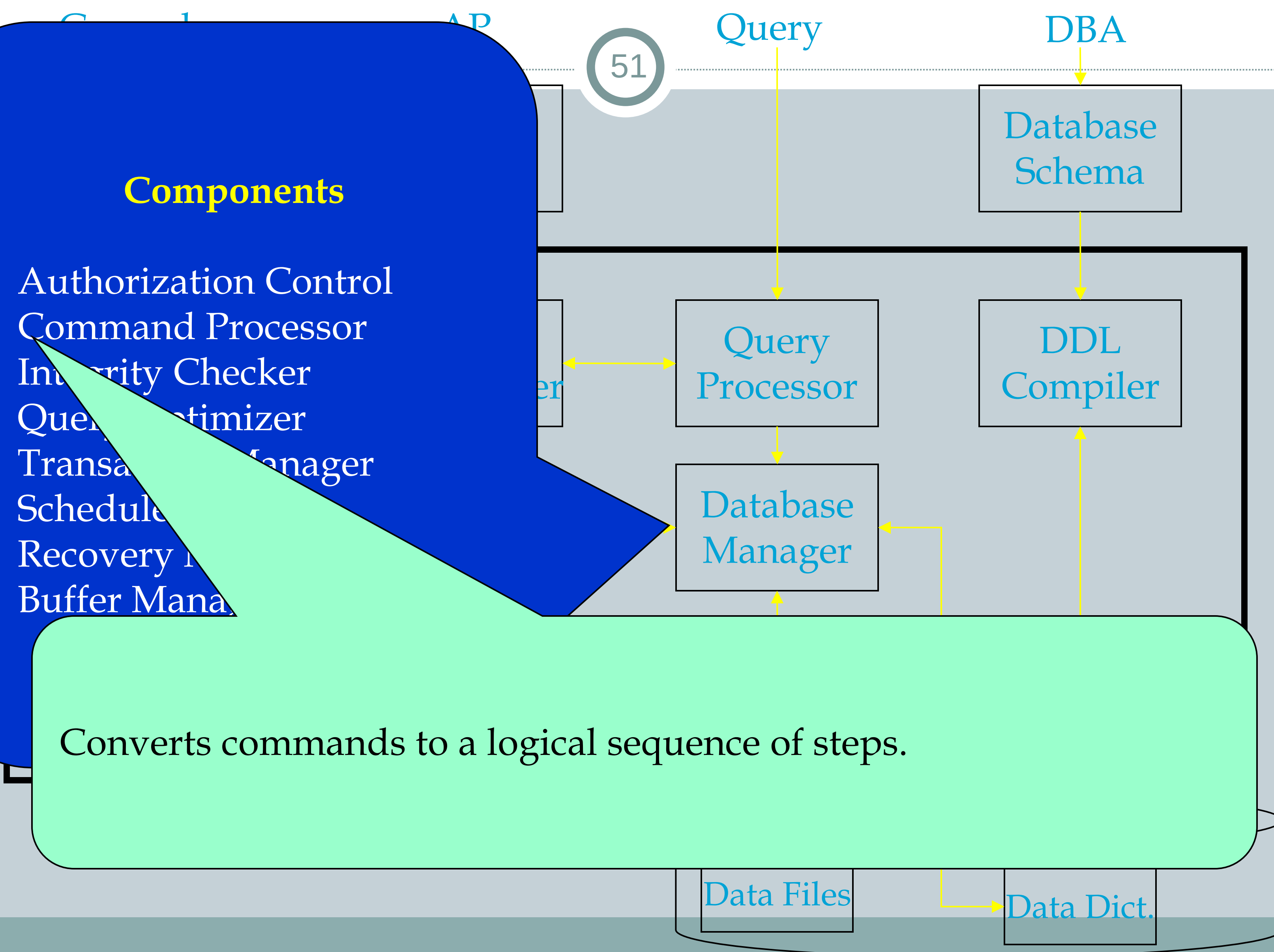
DBMS Structure



DBMS Structure



DBMS Structure

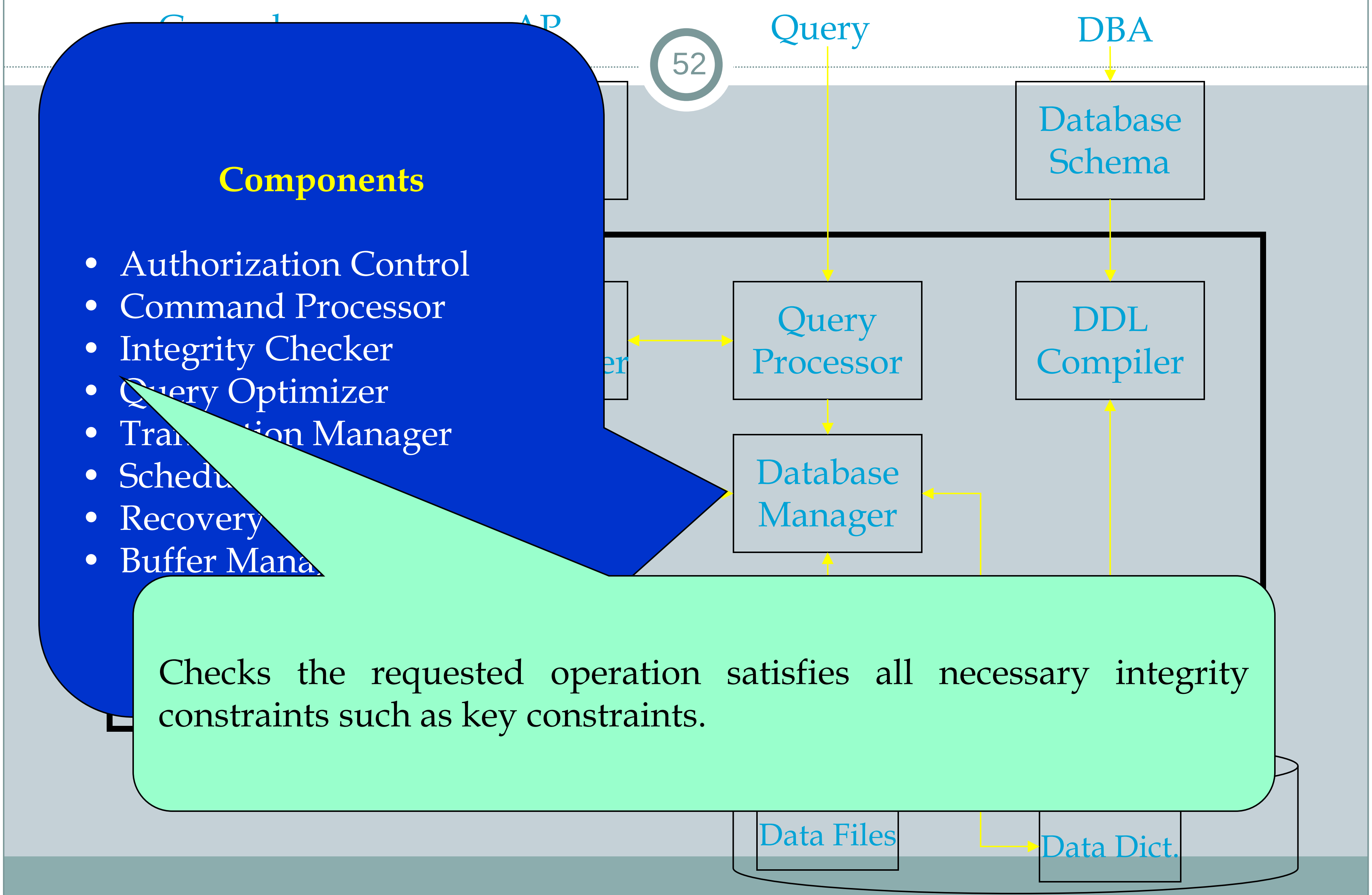


Components

- Authorization Control
- Command Processor
- Int rity Checker
- Quer timizer
- Transa anager
- Schedule
- Recovery I
- Buffer Mana

Converts commands to a logical sequence of steps.

DBMS Structure

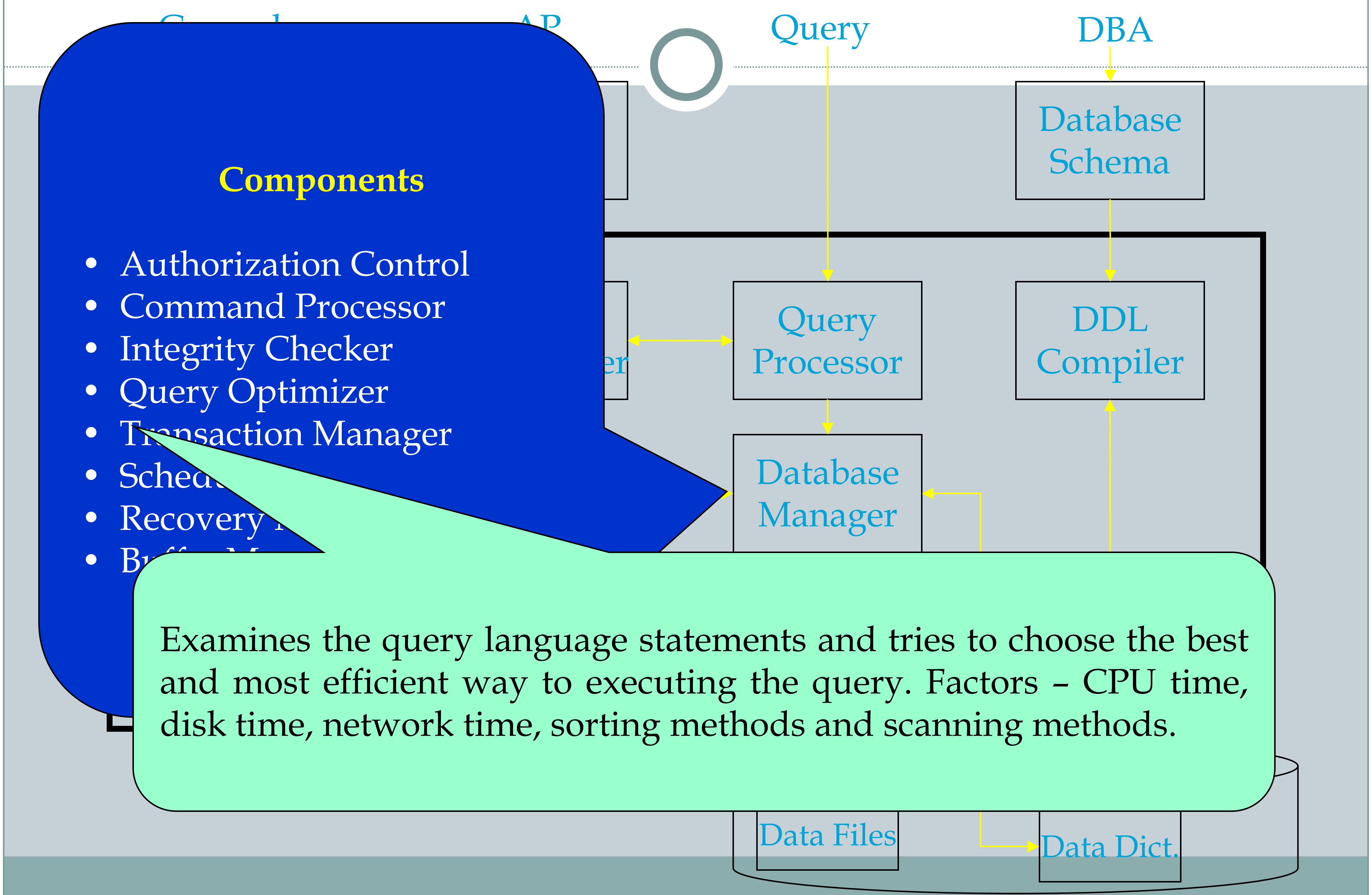


Components

- Authorization Control
- Command Processor
- Integrity Checker
- Query Optimizer
- Transaction Manager
- Schedu
- Recovery
- Buffer Mana

Checks the requested operation satisfies all necessary integrity constraints such as key constraints.

DBMS Structure

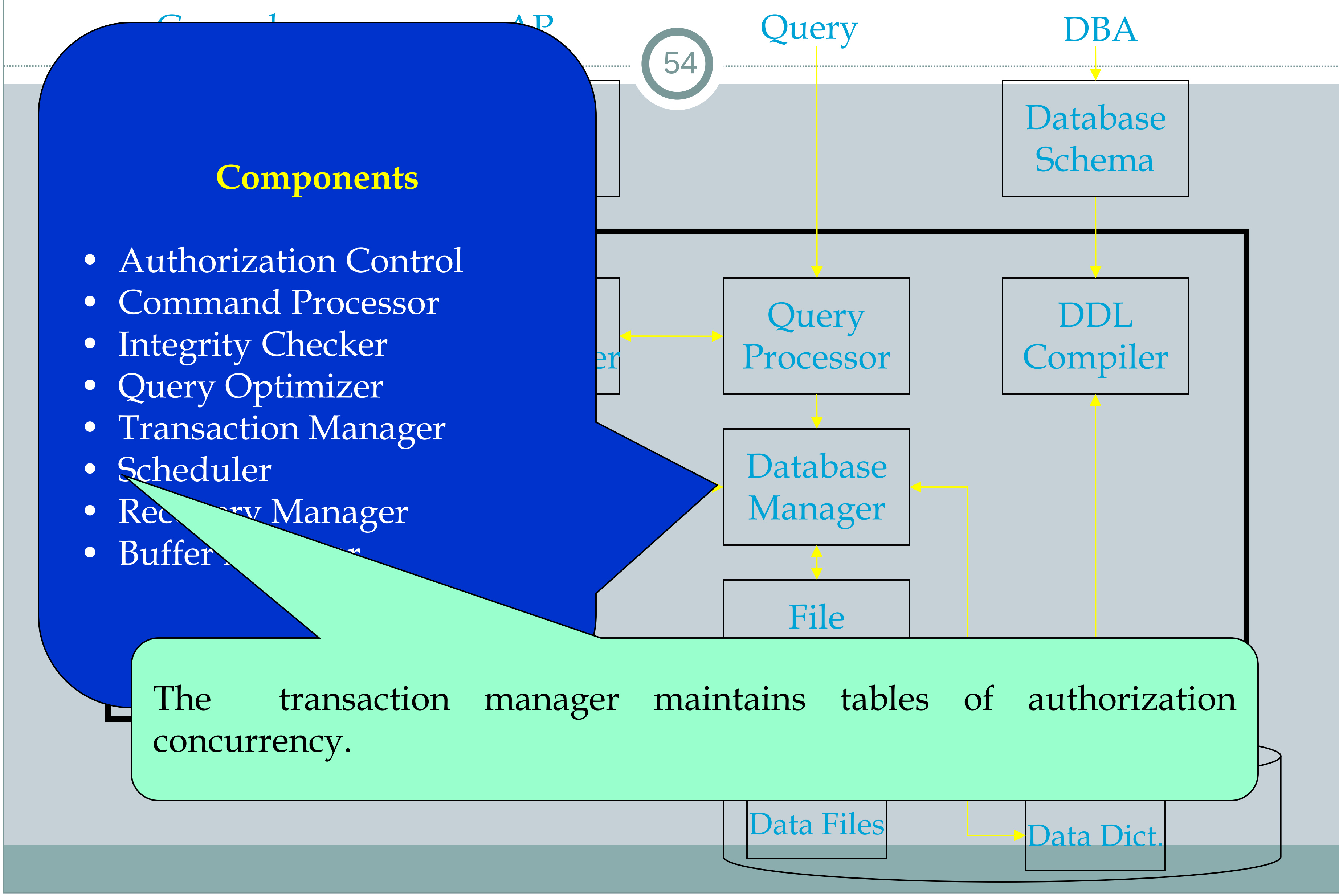


Components

- Authorization Control
- Command Processor
- Integrity Checker
- Query Optimizer
- Transaction Manager
- Sched
- Recovery
- Bu

Examines the query language statements and tries to choose the best and most efficient way to executing the query. Factors – CPU time, disk time, network time, sorting methods and scanning methods.

DBMS Structure

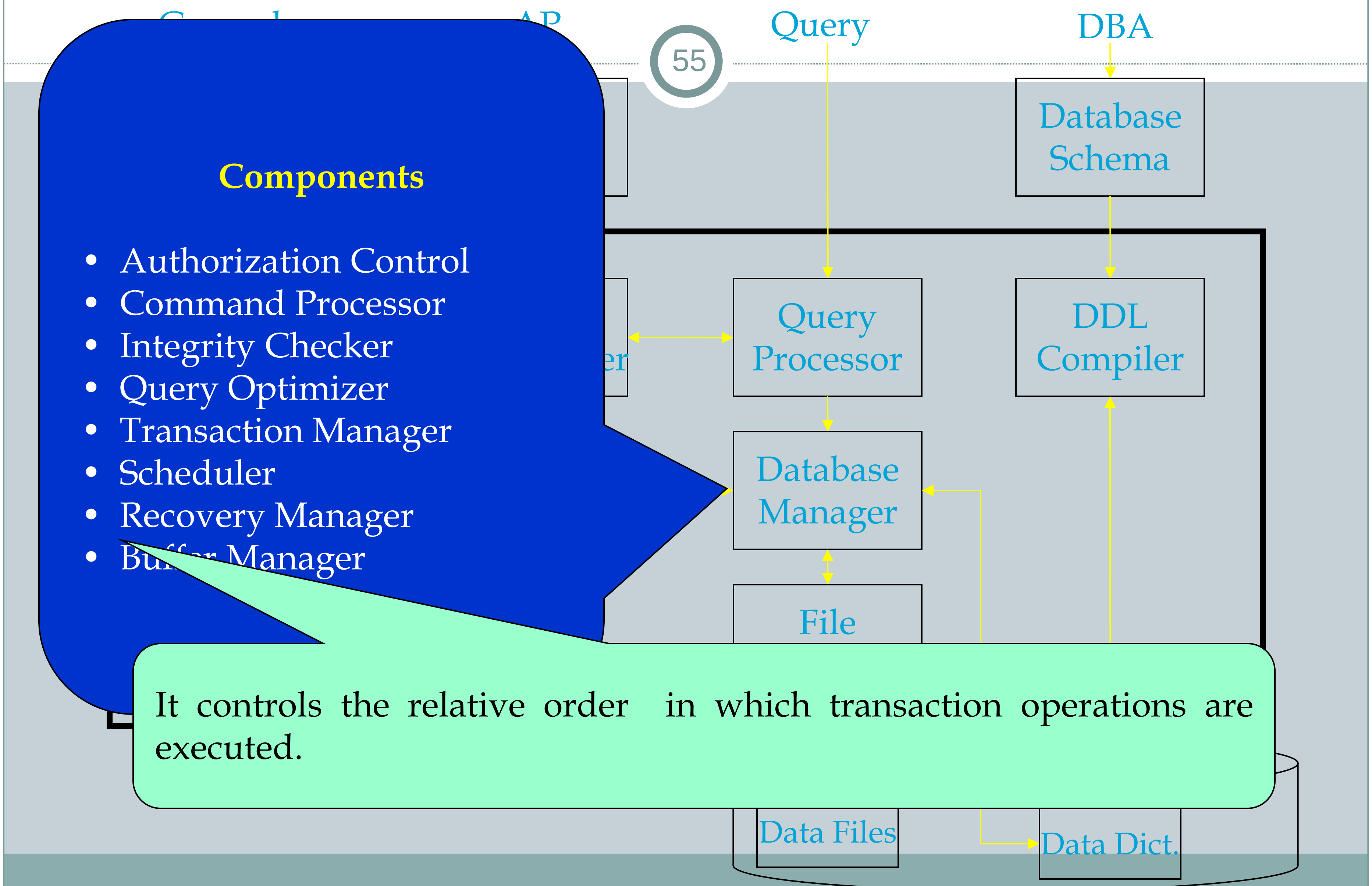


Components

- Authorization Control
- Command Processor
- Integrity Checker
- Query Optimizer
- Transaction Manager
- Scheduler
- Recovery Manager
- Buffer Manager

The transaction manager maintains tables of authorization concurrency.

DBMS Structure

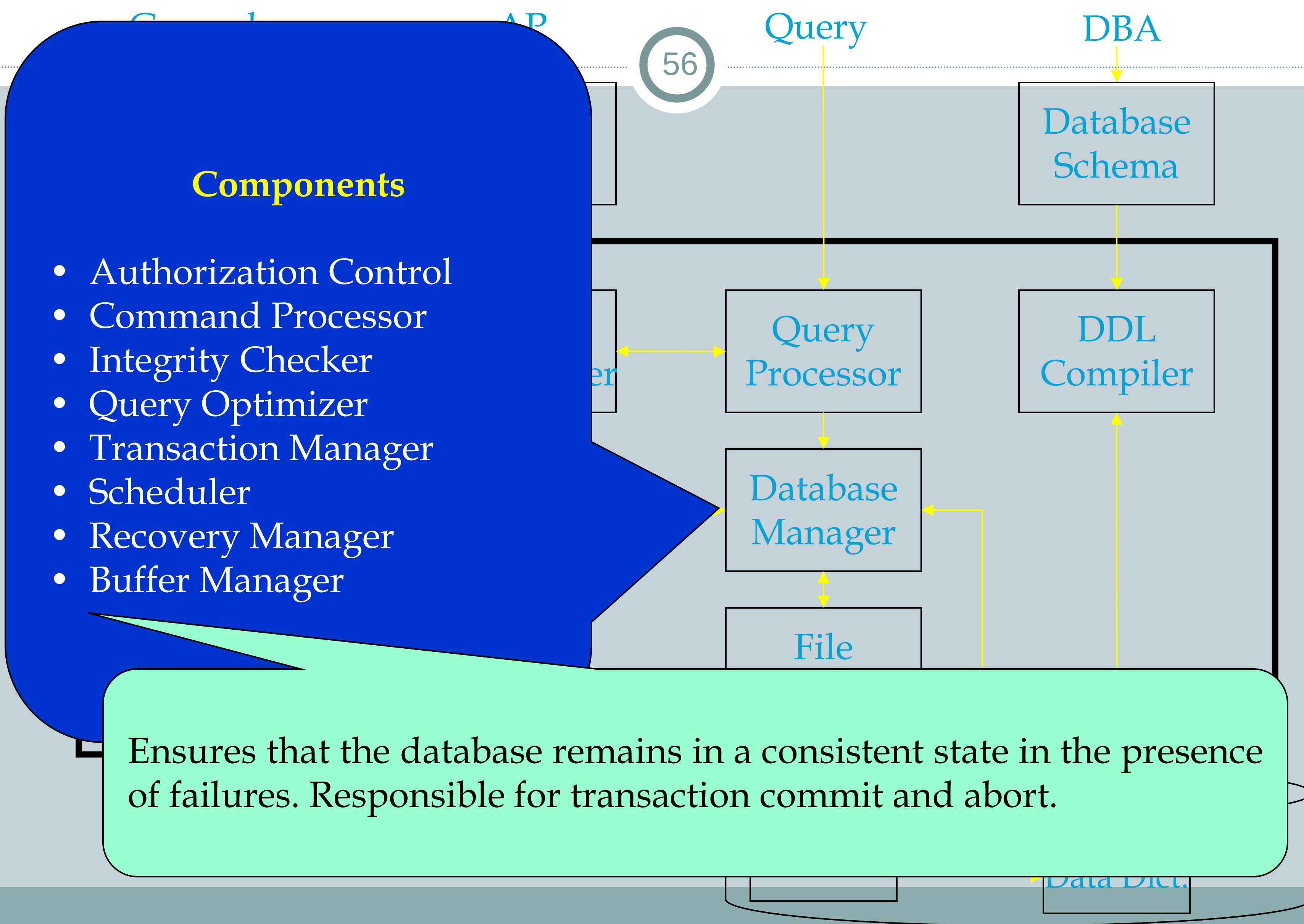


Components

- Authorization Control
- Command Processor
- Integrity Checker
- Query Optimizer
- Transaction Manager
- Scheduler
- Recovery Manager
- Buffer Manager

It controls the relative order in which transaction operations are executed.

DBMS Structure

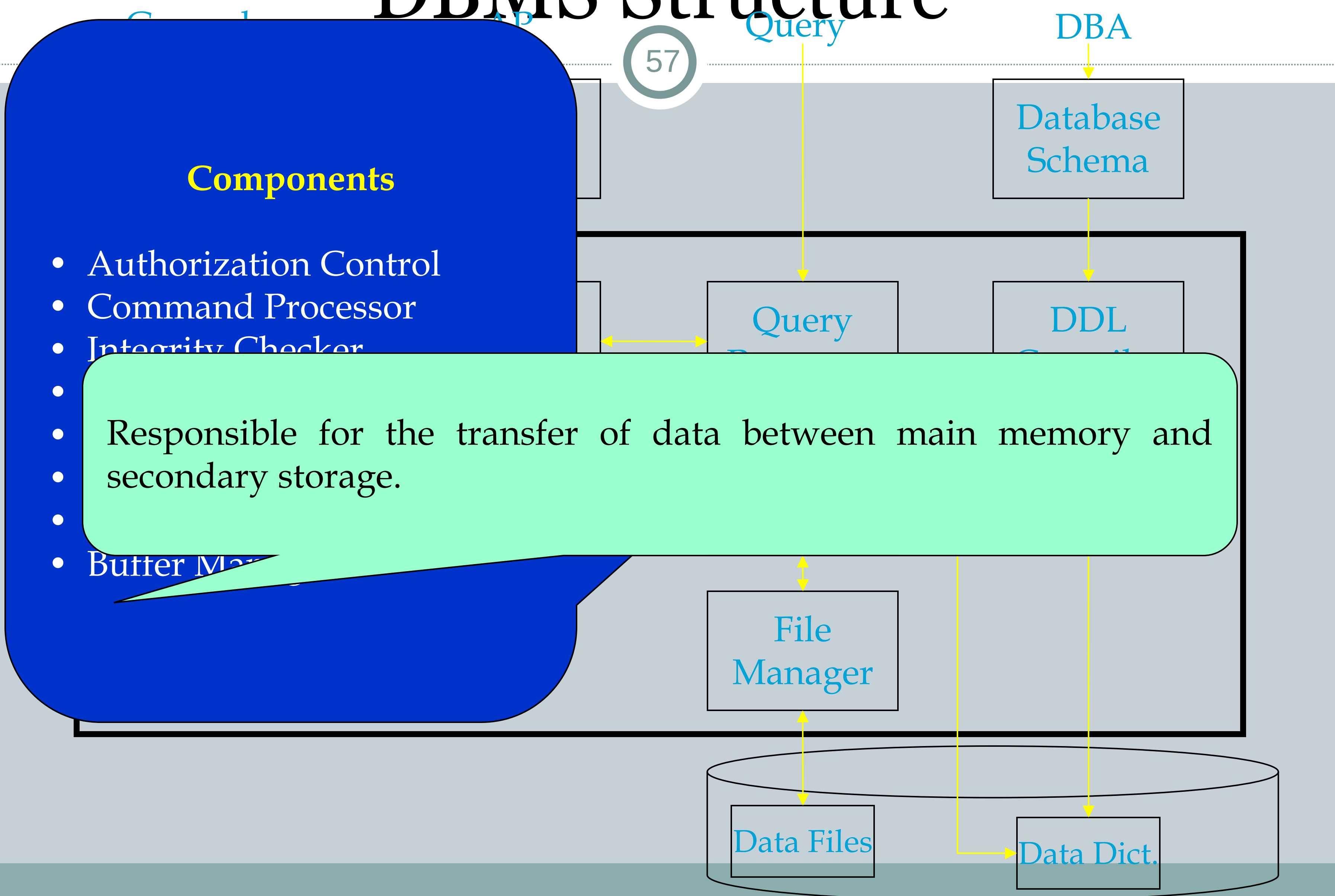


Components

- Authorization Control
- Command Processor
- Integrity Checker
- Query Optimizer
- Transaction Manager
- Scheduler
- Recovery Manager
- Buffer Manager

Ensures that the database remains in a consistent state in the presence of failures. Responsible for transaction commit and abort.

DBMS Structure



Components

- Authorization Control
- Command Processor
- Integrity Checker

Responsible for the transfer of data between main memory and secondary storage.

SQL Vs No SQL databases

•SQL databases are best for applications requiring complex queries, data integrity, and structured data.
•**NoSQL Databases** excel in handling large volumes of unstructured data, providing high scalability and flexibility for modern web applications.

Feature	SQL Databases	NoSQL Databases
Structure	Relational	Non-relational
Schema	Fixed schema	Dynamic schema
Data Integrity	ACID properties	BASE properties
Scalability	Vertical scalability	Horizontal scalability
Query Language	SQL (Structured Query Language)	Varies by database (e.g., MongoDB Query)
Data Model	Tables with rows and columns	Documents, key-value pairs, wide-columns, graphs
Transaction Support	Strong, with complex multi-row transactions	Varies, typically eventual consistency
Flexibility	Less flexible, requires predefined schema	More flexible, schema-less
Examples	MySQL, PostgreSQL, Oracle, SQL Server	MongoDB, Cassandra, Redis, Neo4j
Use Cases	Complex queries, data integrity (e.g., banking, enterprise applications)	Big data, real-time web apps, unstructured data (e.g., social networks, IoT)
Consistency	Strong consistency	Eventual consistency
Performance	Good for complex queries and transactions	High performance for read/write operations
Community and Support	Mature ecosystem with extensive support	Growing ecosystem, varying support