Databases
Lab

DBMS Architecture

Flavio Bertini

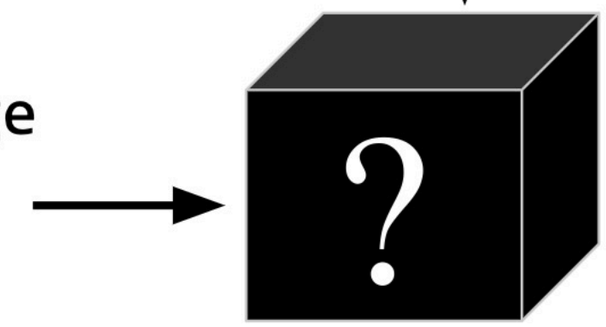
flavio.bertini@smartdata.cs.unibo.it

DataBase Management System

- A **database management system** (DBMS) is software for creating and managing databases. The DBMS **provides an efficient and persistent way to** create, retrieve, update and **manage huge amounts of data**.
- How does a DBMS work?

Name	Surname	Gender	Wage
Mary	Smith	F	1700
James	Johnson	M	1200
Michael	Williams	M	1550
Susan	Brown	F	1500

```
SELECT Surname, Wage
FROM EMPLOYEES
WHERE Wage > 1500
```



Surname	Wage
Smith	1700
Williams	1550

Implementing a DBMS

- What if we would like to implement a relational DBMS?
  It has to:
  - store relations as tables, like R(A,B,C), S(A,B) e T(A,B)
  - perform SQL queries, like:

```
SELECT A, B
FROM R
WHERE B = 2000
```

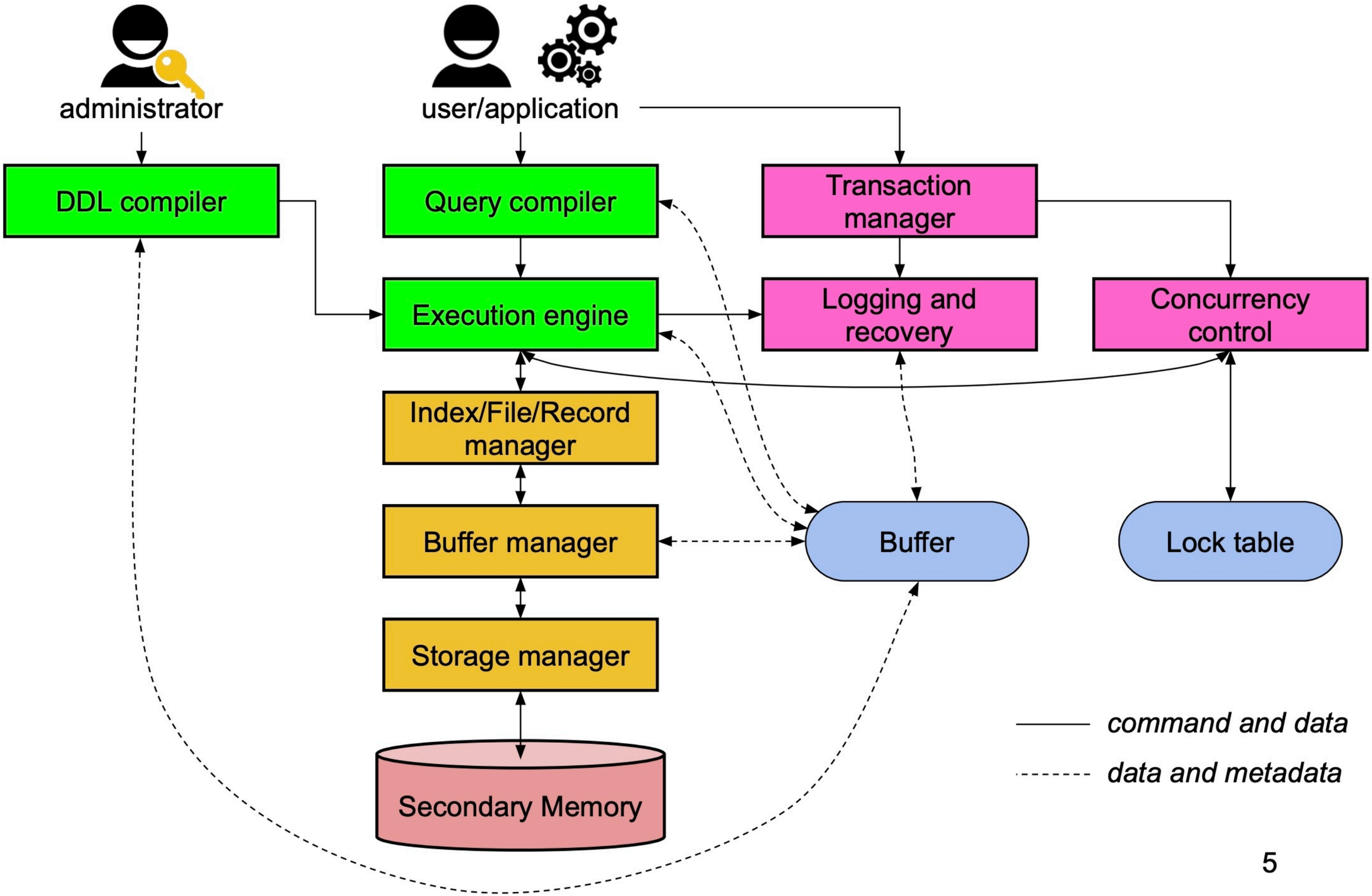
```
SELECT R.A, T.B
FROM R, S, T
WHERE R.B = S.A AND
      S.B = T.A
```

DBMS Functions

How does the system behave with respect to:

- tuples search?
- tuples update?
- running order of SQL queries?
- repeated access to the same disk locations?
- concurrent read & write operations?
- access control?
- disaster recovery?
- application programming interfaces?

Relational DBMS Architecture



DBMS: Main Components

The relational DBMS architecture is composed by three main parts:

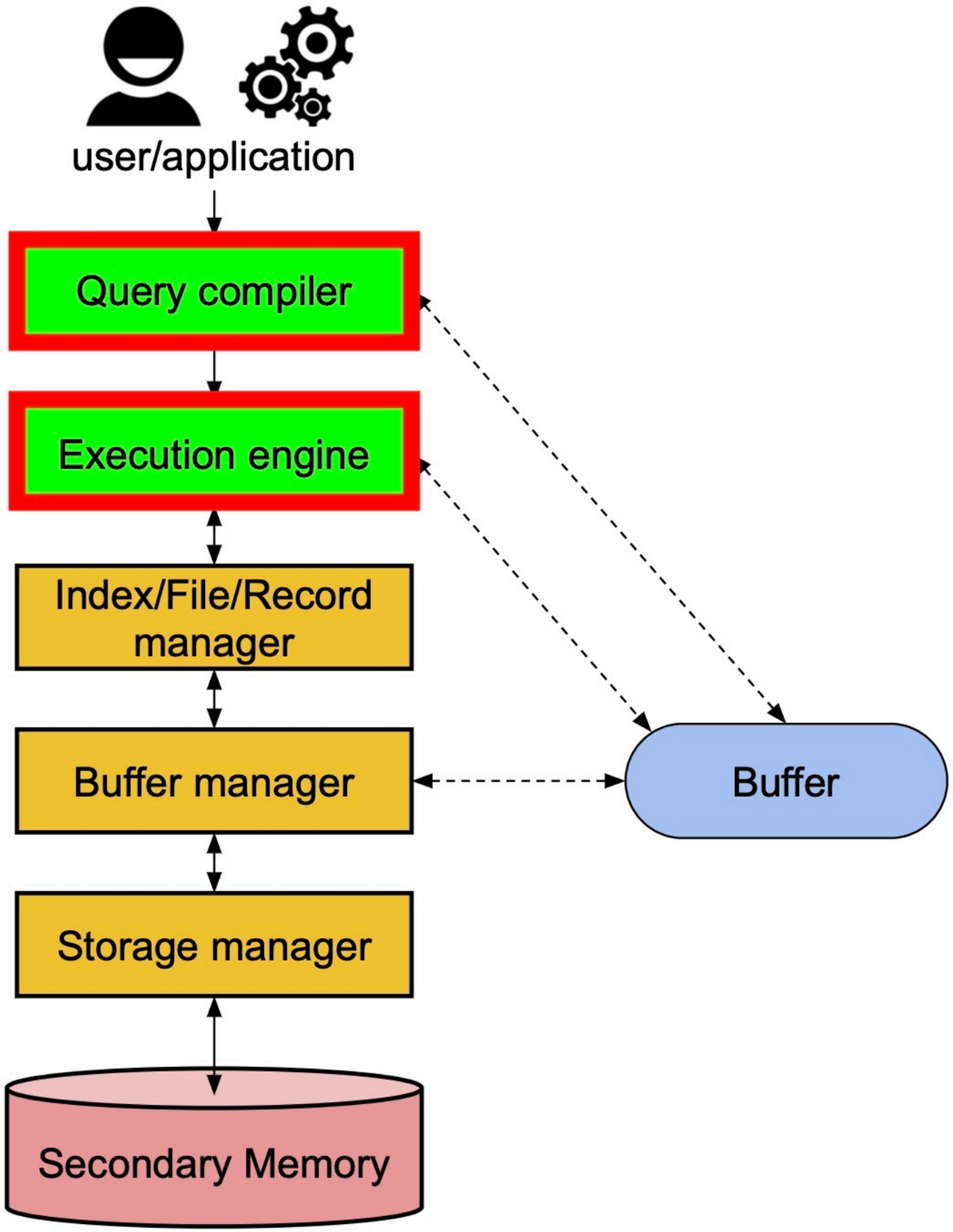
- **Query Processor**
- **Resource Manager**
- **Transaction Manager**

Query Processor (1)

The Query Processor is the component that most affects the performance. It is represented by two units.

- The **Query Compiler**, that translates the query in a query plan through these steps:
  - *parsing*, which performs the syntactic check and builds a tree structure;
  - *preprocessing*, which performs the semantic checks and transforms the tree in an algebraic operation tree;
  - *optimization*, which transforms the original query into the best sequence of operations (e.g., equivalent relational algebra expressions).
- The **Execution Engine**, which executes each of the steps in the chosen query plan interacting with most of the other components, like buffer, scheduler, and log manager.

Query Processor (2)




```
SELECT A, B
FROM R1, R2
WHERE C = D AND C = 'c'
```

$$\pi_{A,B}(\sigma_{C=D}(\sigma_{C='c'}(R1) \bowtie R2))$$

$$\sigma_{C='c'}(R1), \quad \bowtie R2, \quad \sigma_{C=D}, \quad \pi_{A,B}$$

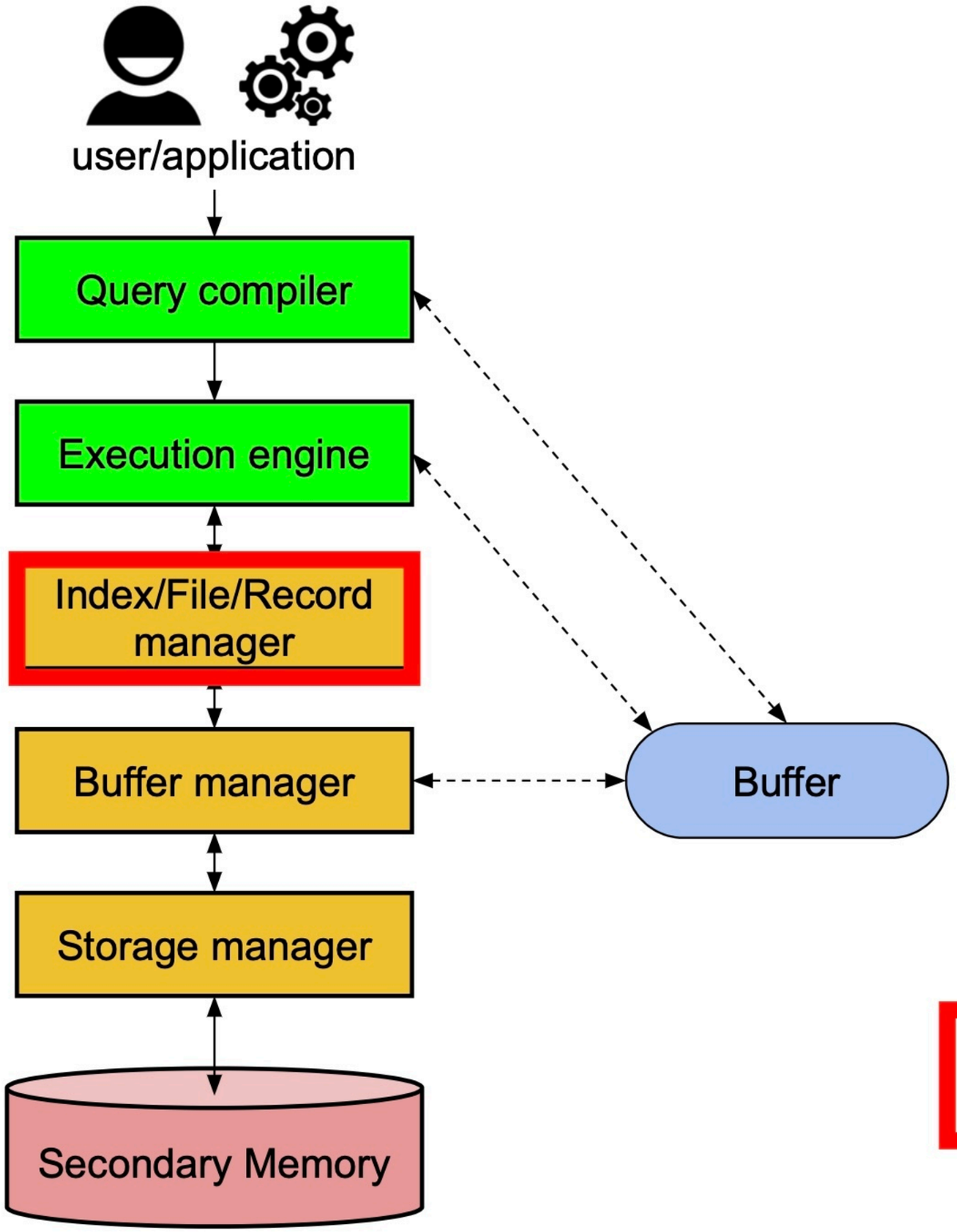
Resource Manager (1)

The data of a database normally resides in secondary storage to ensure persistency. The Resource Manager knows where the data is (i.e., file, records), and how to retrieve them quickly (i.e., indexes). It includes:

- the **Index/File/Record Manager**, which knows the database schema, and the data structures that support efficient access to the data;
- the **Buffer Manager**, which partitions the main memory into buffers into which disk blocks can be transfer;
- the **Storage Manager**, which keeps track of the location of files on the disk and obtains blocks containing a file on request from the buffer manager.

The kinds of information that various components may need include: data, metadata, index, log records, and statistics.

Resource Manager (2)

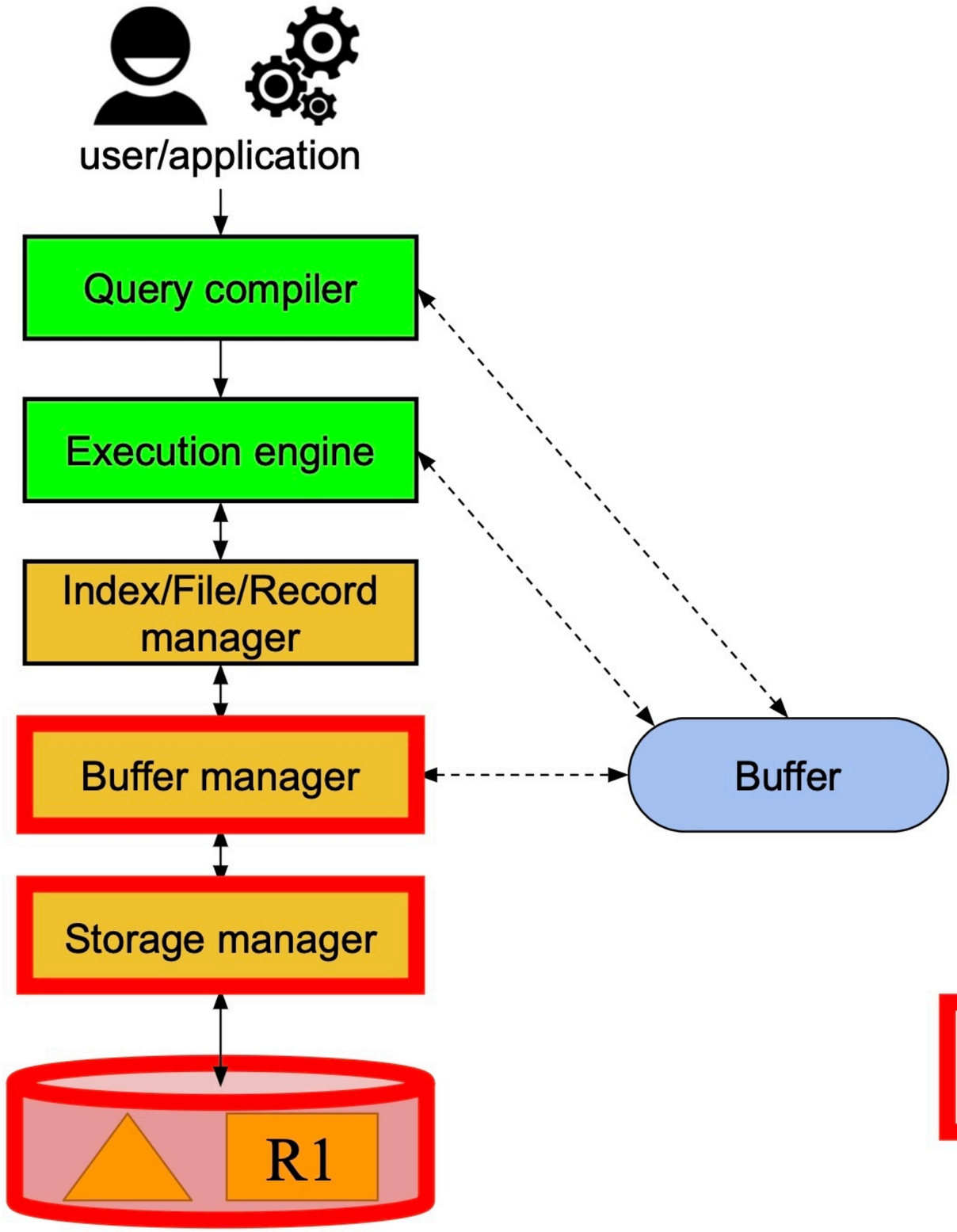



```
SELECT A, B
FROM R1, R2
WHERE C = D AND C = 'c'
```

$$\pi_{A,B}(\sigma_{C=D}(\sigma_{C='c'}(R1) \bowtie R2))$$

$\sigma_{C='c'}(R1)$, $\bowtie R2$, $\sigma_{C=D}$, $\pi_{A,B}$

Resource Manager (2)

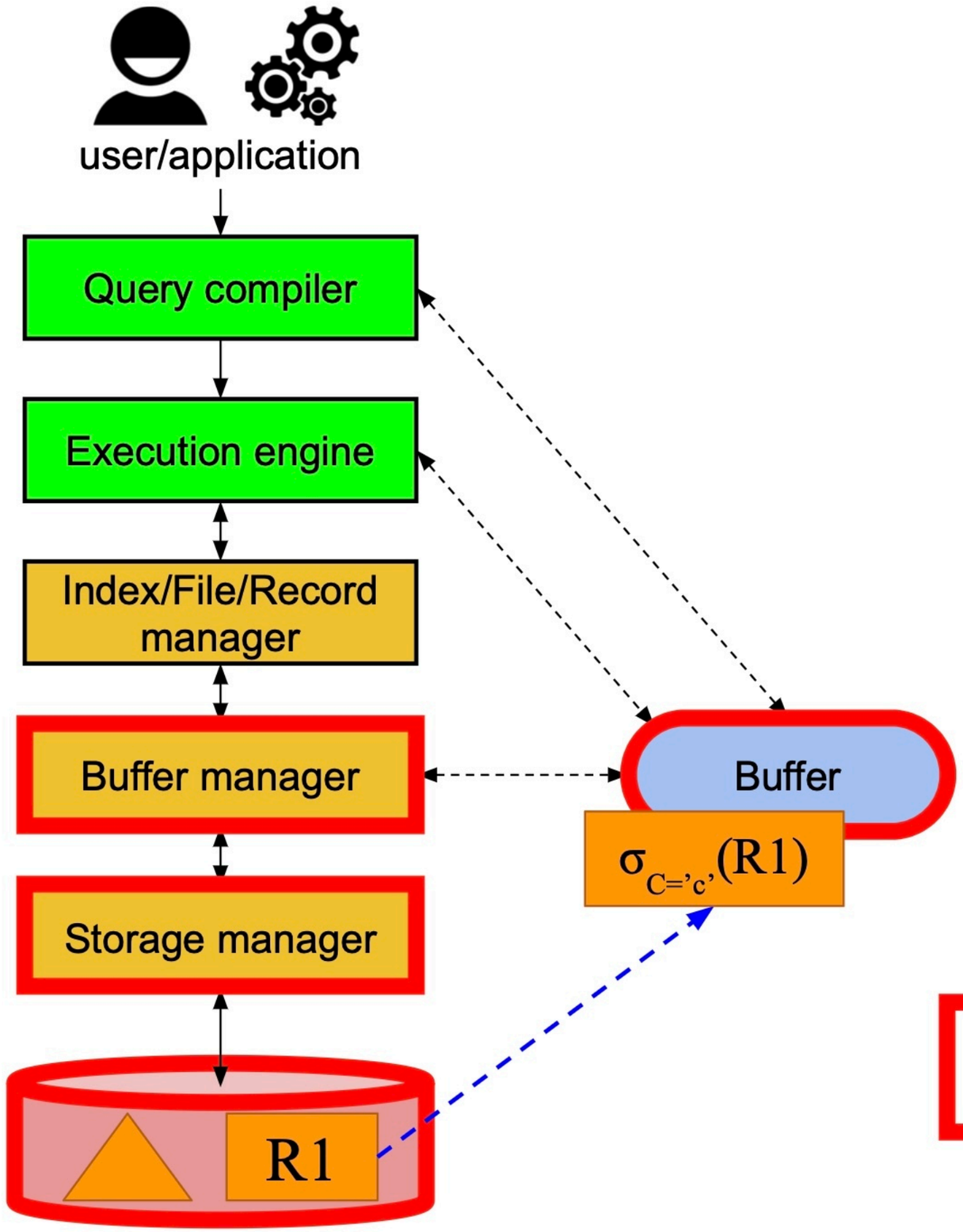



```
SELECT A, B
FROM R1, R2
WHERE C = D AND C = 'c'
```

$$\pi_{A,B}(\sigma_{C=D}(\sigma_{C='c'}(R1) \bowtie R2))$$

$\sigma_{C='c'}(R1)$, $\bowtie R2$, $\sigma_{C=D}$, $\pi_{A,B}$

Resource Manager (2)

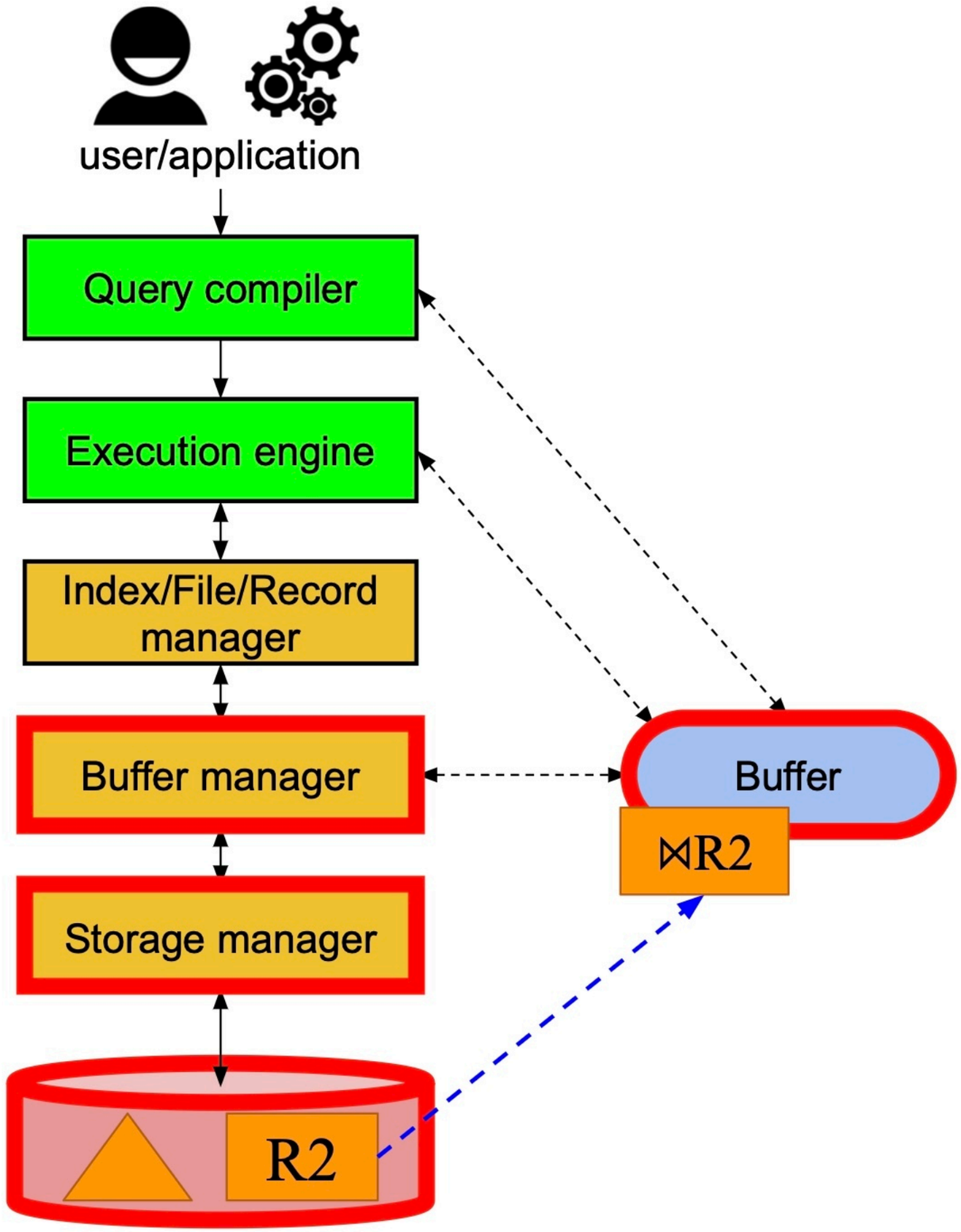



```
SELECT A, B
FROM R1, R2
WHERE C = D AND C = 'c'
```

$$\pi_{A,B}(\sigma_{C=D}(\sigma_{C='c'}(R1) \bowtie R2))$$

$\sigma_{C='c'}(R1)$, $\bowtie R2$, $\sigma_{C=D}$, $\pi_{A,B}$

Resource Manager (2)

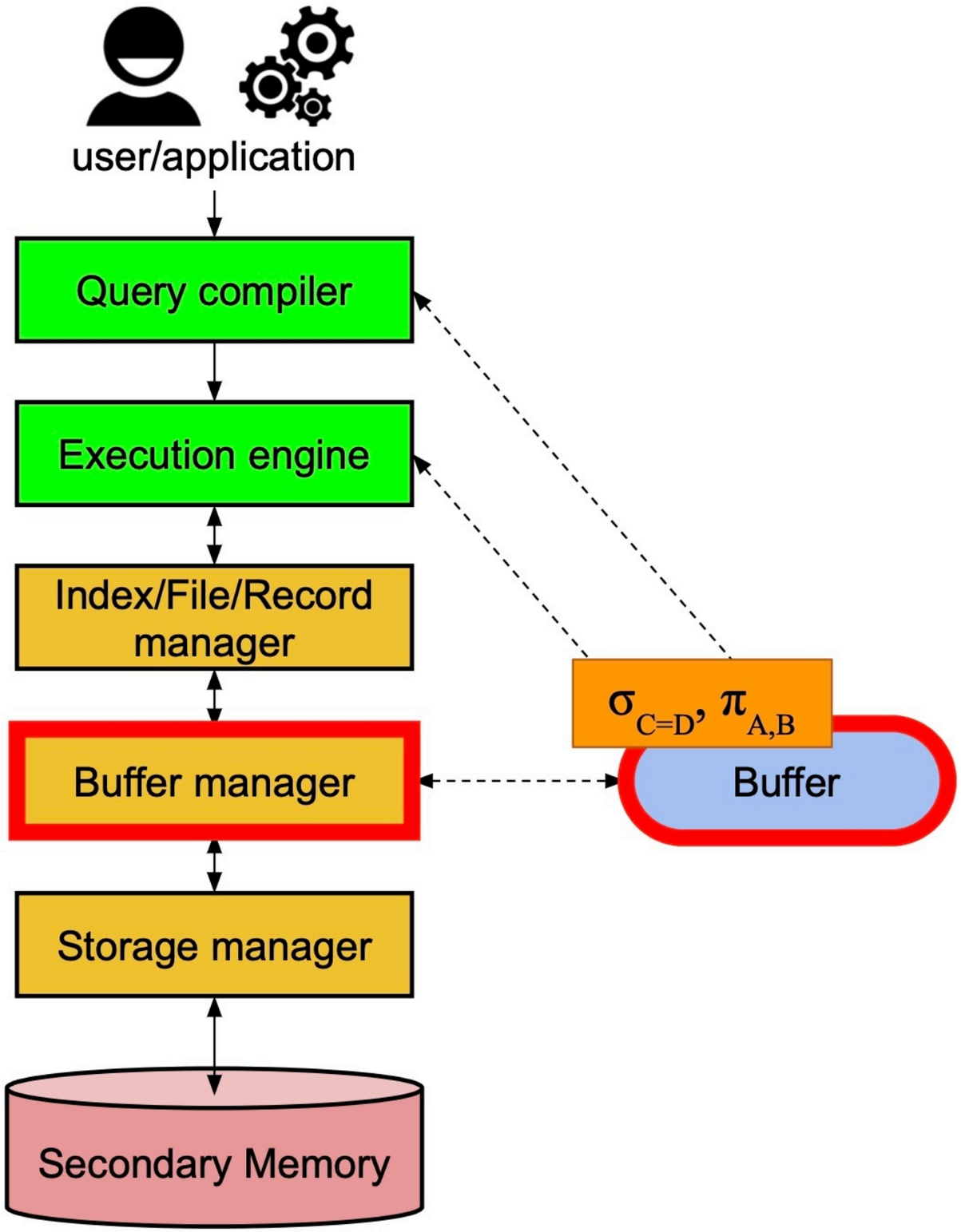



```
SELECT A, B
FROM R1, R2
WHERE C = D AND C = 'c'
```

$$\pi_{A,B}(\sigma_{C=D}(\sigma_{C='c'}(R1)\bowtie R2))$$

$$\sigma_{C='c'}(R1),\ \boxed{\bowtie R2,}\ \sigma_{C=D},\ \pi_{A,B}$$

Resource Manager (2)




```
SELECT A, B
FROM R1, R2
WHERE C = D AND C = 'c'
```

$$\pi_{A,B}(\sigma_{C=D}(\sigma_{C='c'}(R1) \bowtie R2))$$

$\sigma_{C='c'}(R1)$, $\bowtie R2$, $\sigma_{C=D}$, $\pi_{A,B}$

Transaction Manager (1)

- Operations on a database rarely involve a single SQL statement at a time, but typically include a series of SQL queries.
  Bank transfer example:

```
update BankAccount Set Balance = Balance + 500
       Where AccountNumber = 12202;
update BankAccount Set Balance = Balance - 500
       Where AccountNumber = 42177;
```

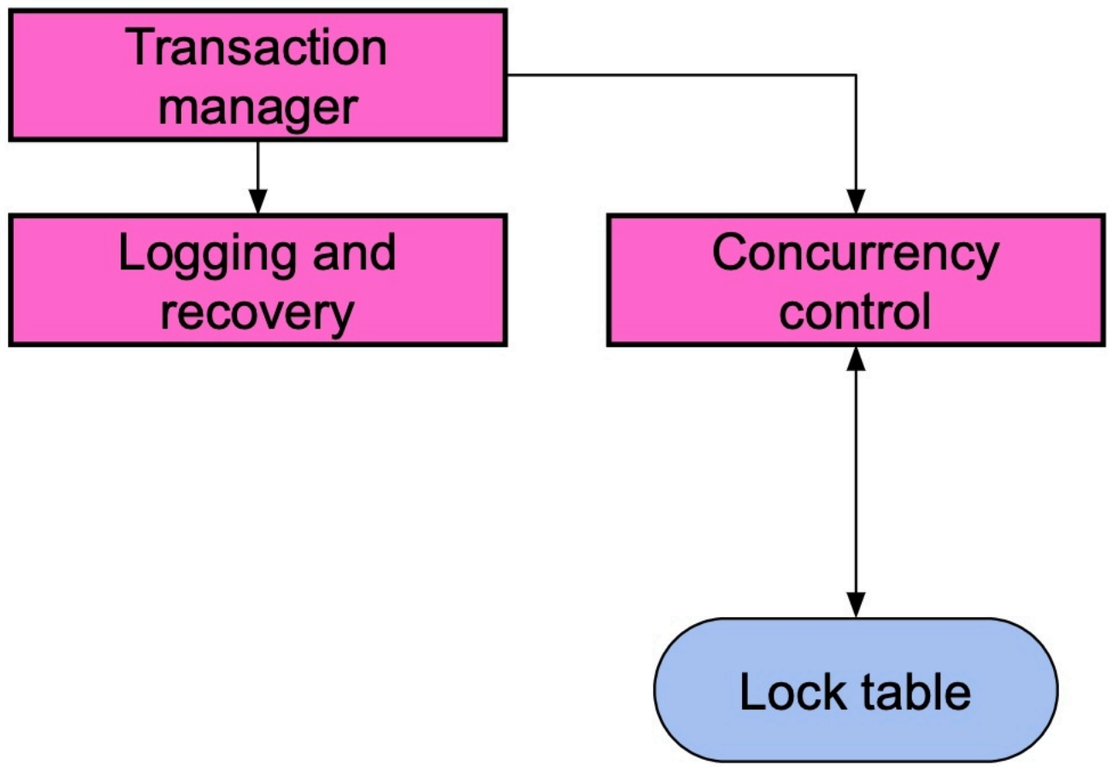
Transaction Manager (1)

- Operations on a database rarely involve a single SQL statement at a time, but typically include a series of SQL queries.
  Bank transfer example:

```
start transaction;
    update BankAccount Set Balance = Balance + 500
        Where AccountNumber = 12202;
    update BankAccount Set Balance = Balance - 500
        Where AccountNumber = 42177;
end transaction;
```

- It is normal to **group more SQL queries into a transaction**, which is a unit of work that must be **executed atomically**, and in apparent **isolation** from the other transactions.
- Moreover, a DBMS ensures the **durability**: the work of a transaction will never be lost.

Transaction Manager (2)



Logging and Recovery

- In order to assure durability, the **Log Manager write separately on disk every change in the database**.
- The Log Manager initially **writes the log in the buffers**, and negotiates with the Buffer Manager to **make sure that buffers are written on disk** (where data can survive a crash).
- No matter when a system failure or crash occurs, the **Recovery Manager** examines the log changes and **restore the database to a consistent state**.

Concurrency Control

- Transaction must appear to execute in isolation, but in most system, there will in truth be **many transactions executing at once**.
- The **Concurrency-Control Manager must assure** that the individual actions of multiple transactions are execute in such an order **that the net effect is the same** as if the transaction had in fact executed in their entirety, one-at-a-time.
- A typical scheduler maintains **locks** on certain pieces of the databases, **preventing two transactions from accessing the same piece of data in ways that interact badly**. The scheduler **affects the execution of the queries and other database operations** by forbidding the locked parts.

Deadlock resolution

- The **transactions compete for resources** through the locks that the scheduler grants.
- **Transactions can get into** a situation where none can proceed because each need something another transaction has, namely **deadlock**.
- The **Transaction Manager has the responsibility** to intervene and cancel, **“rollback” or “abort”, one or more transactions** to let the others proceed.

Deadlock resolution

- The **transactions compete for resources** through the locks that the scheduler grants.
- **Transactions can get into** a situation where none can proceed because each need something another transaction has, namely **deadlock**.
- The **Transaction Manager has the responsibility** to intervene and cancel, **“rollback” or “abort”, one or more transactions** to let the others proceed.