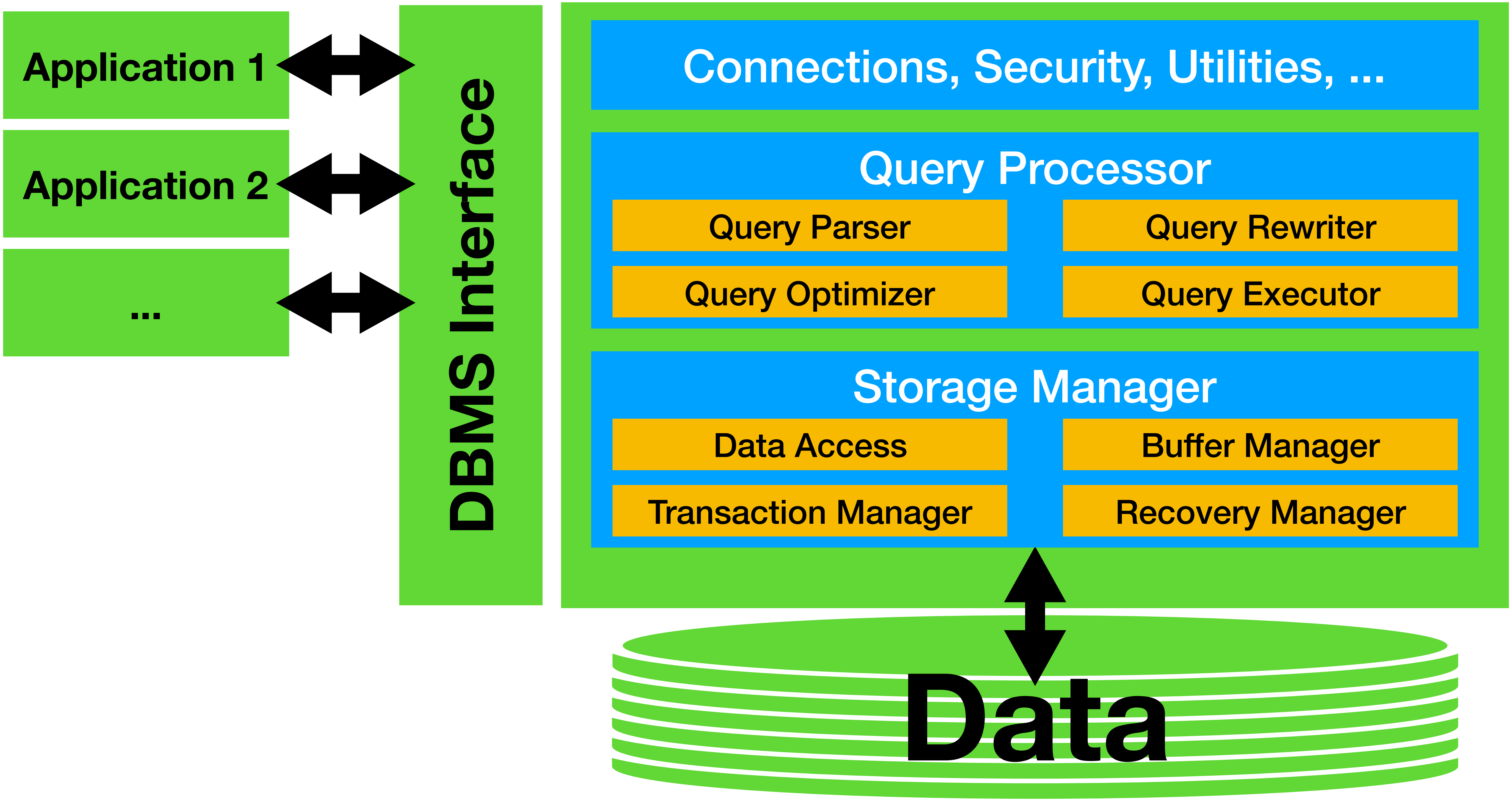
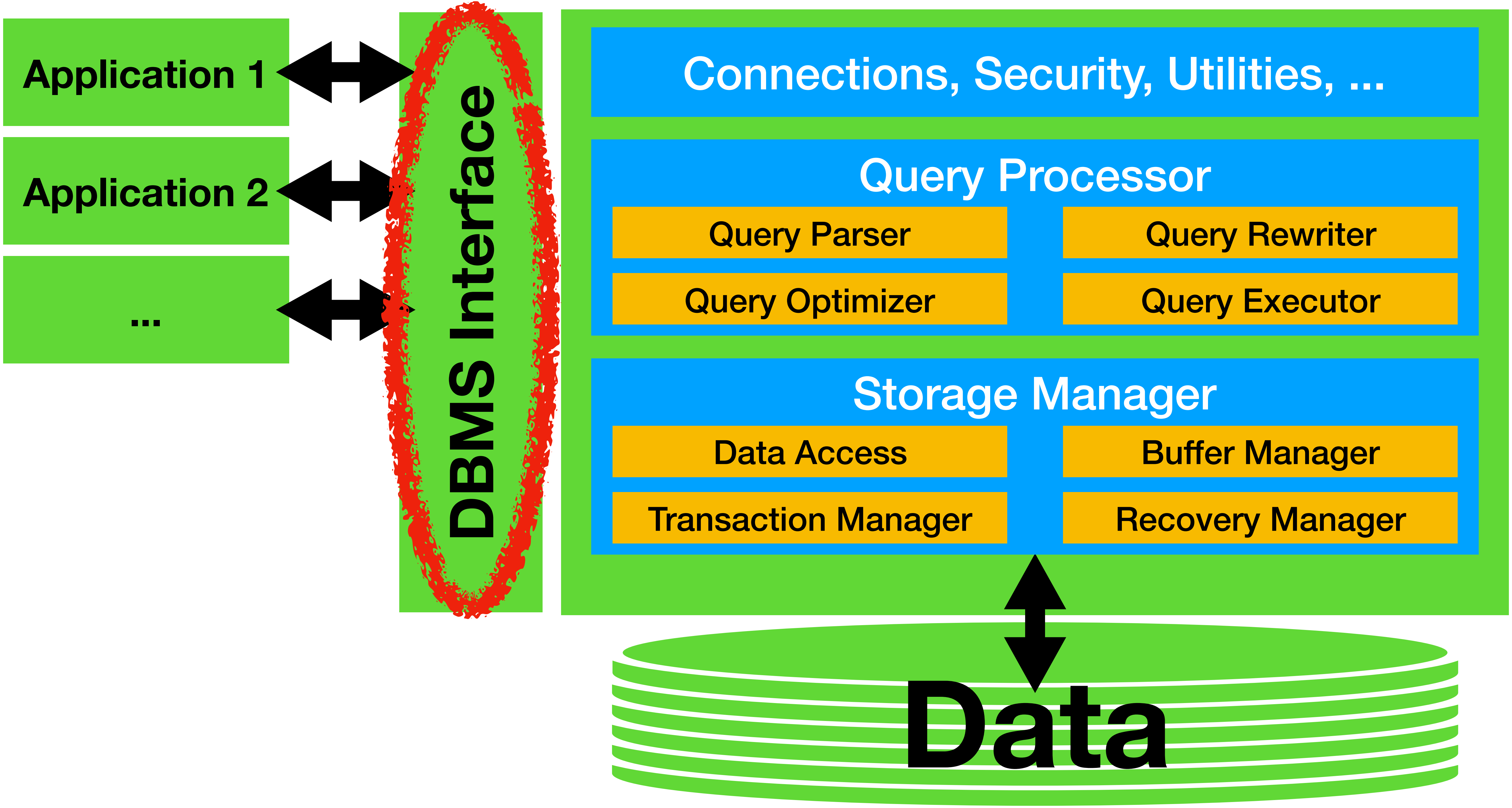
Repeat Session

Immanuel Trummer
itrummer@cornell.edu
www.itrummer.org

Database Management Systems (DBMS)



Database Management Systems (DBMS)



SQL Command Types

- **DDL: Data Definition Language**
  - Define admissible database content (schema)
- **DML: Data Manipulation Language**
  - Change and retrieve database content
- **TCL: Transaction Control Language**
  - Groups SQL commands (transactions)
- **DCL: Data Control Language**
  - Assign data access rights

Grouping Example

Database Relations:

Students(Sid, Sname)
Enrollment(Sid, Cid)
Courses(Cid, Cname)

```
SELECT Count(*), Cname
FROM Students
  JOIN Enrollment ON (Students.sid = Enrollment.sid)
  JOIN Courses ON (Enrollment.cid = Courses.cid)
WHERE Cname IN ('CS4320', 'CS5320')
GROUP BY Cname
```

Grouping Example

Database Relations:

Students(Sid, Sname)
Enrollment(Sid, Cid)
Courses(Cid, Cname)

Find pairs of students and enrollment tuples where Sid (i.e., student ID) is the same ...

```
SELECT Count(*), Cname
FROM Students
  JOIN Enrollment ON (Students.sid = Enrollment.sid)
  JOIN Courses ON (Enrollment.cid = Courses.cid)
WHERE Cname IN ('CS4320', 'CS5320')
GROUP BY Cname
```

Grouping Example

Database Relations:

Students(Sid, Sname)
Enrollment(Sid, Cid)
Courses(Cid, Cname)

```
SELECT Count(*), Cname
FROM Students
  JOIN Enrollment ON (Students.sid = Enrollment.sid)
  JOIN Courses ON (Enrollment.cid = Courses.cid)
WHERE Cname IN ('CS4320', 'CS5320')
GROUP BY Cname
```

... pair that with courses where Cid (i.e., course ID) matches the one in enrollment ...

Grouping Example

Database Relations:

Students(Sid, Sname)
Enrollment(Sid, Cid)
Courses(Cid, Cname)

```
SELECT Count(*), Cname
FROM Students
  JOIN Enrollment ON (Students.sid = Enrollment.sid)
  JOIN Courses ON (Enrollment.cid = Courses.cid)
WHERE Cname IN ('CS4320', 'CS5320')
GROUP BY Cname
```

... filter to rows where Cname (course name) is 'CS4320' or 'CS5320' ...

Grouping Example

Database Relations:

Students(Sid, Sname)
Enrollment(Sid, Cid)
Courses(Cid, Cname)

```
SELECT Count(*), Cname
FROM Students
  JOIN Enrollment ON (Students.sid = Enrollment.sid)
  JOIN Courses ON (Enrollment.cid = Courses.cid)
WHERE Cname IN ('CS4320', 'CS5320')
GROUP BY Cname
```

... group remaining rows by
Cname (Course name) ...

Grouping Example

Database Relations:

Students(Sid, Sname)
Enrollment(Sid, Cid)
Courses(Cid, Cname)

... count rows in each group and report count with course name (unique per group).

```
SELECT Count(*), Cname
FROM Students
  JOIN Enrollment ON (Students.sid = Enrollment.sid)
  JOIN Courses ON (Enrollment.cid = Courses.cid)
WHERE Cname IN ('CS4320', 'CS5320')
GROUP BY Cname
```

Multiple Nesting Levels

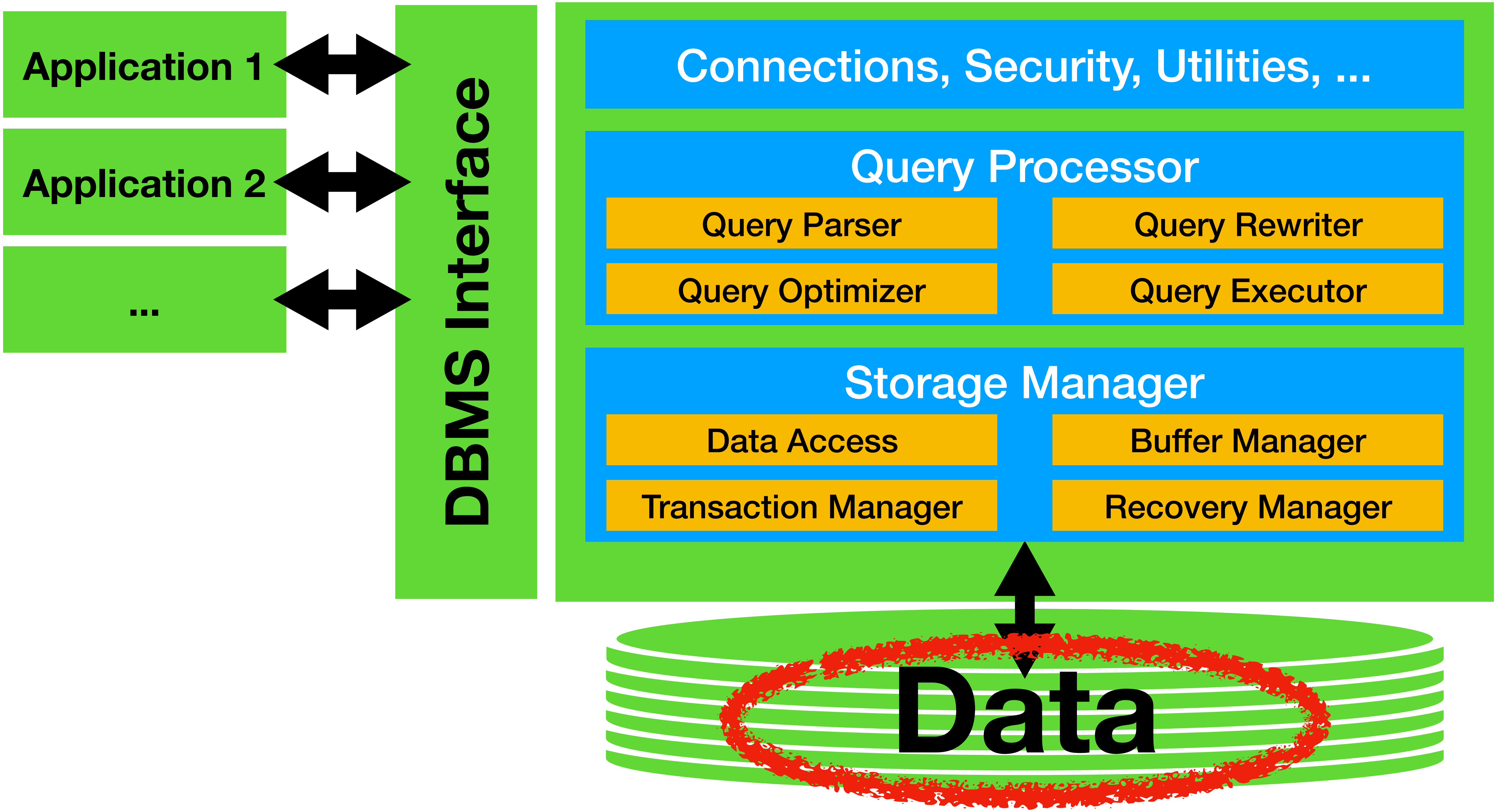
Database Relations:

Students(Sid, Sname)
Enrollment(Sid, Cid)
Courses(Cid, Cname)

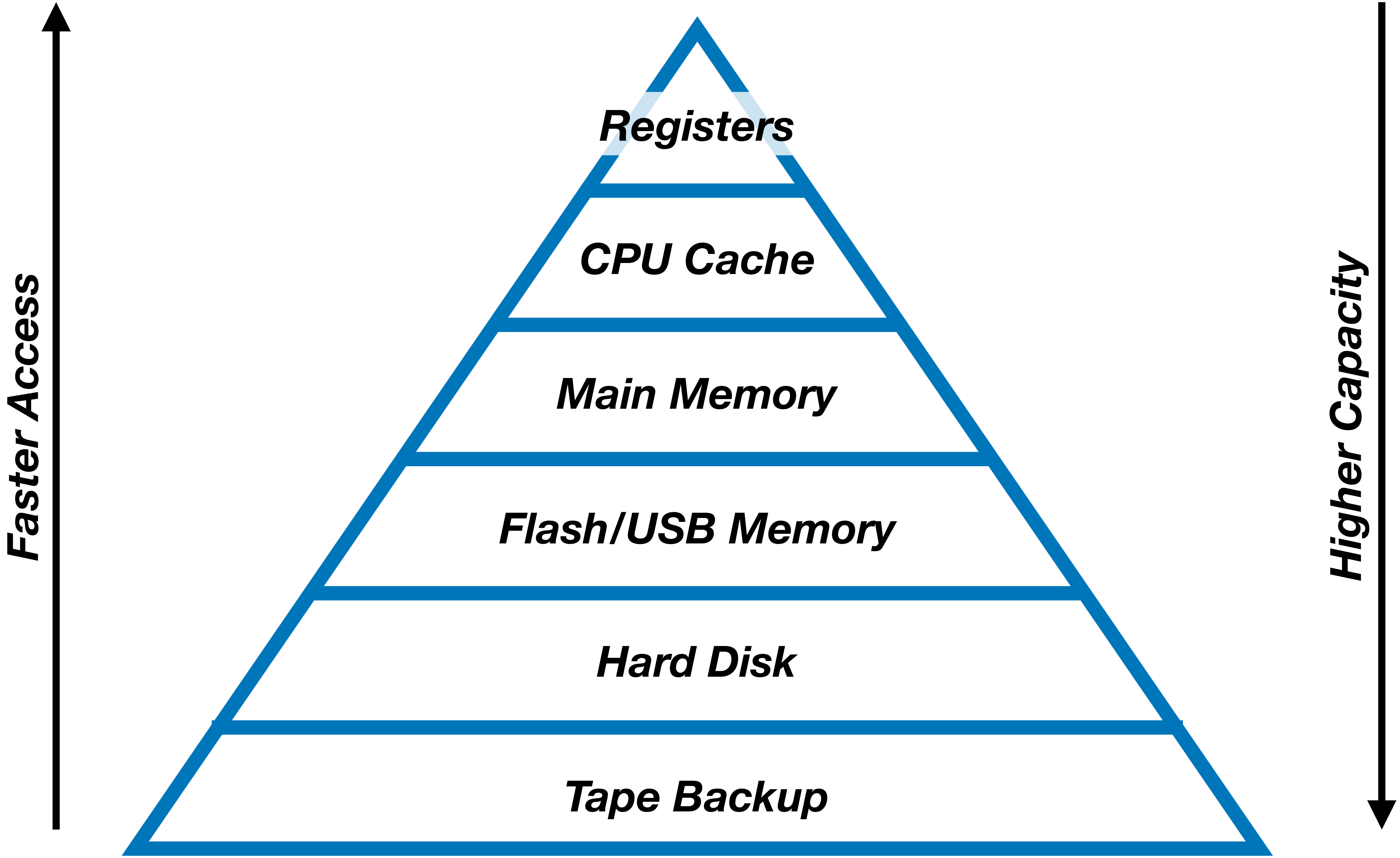
```
SELECT C.Cname FROM Courses C WHERE NOT EXISTS (
  SELECT * FROM Students S WHERE NOT EXISTS(
      SELECT * FROM Enrollment E
        WHERE E.cid = C.cid AND E.sid = S.sid
    )
  )
```

What does this do ... ?

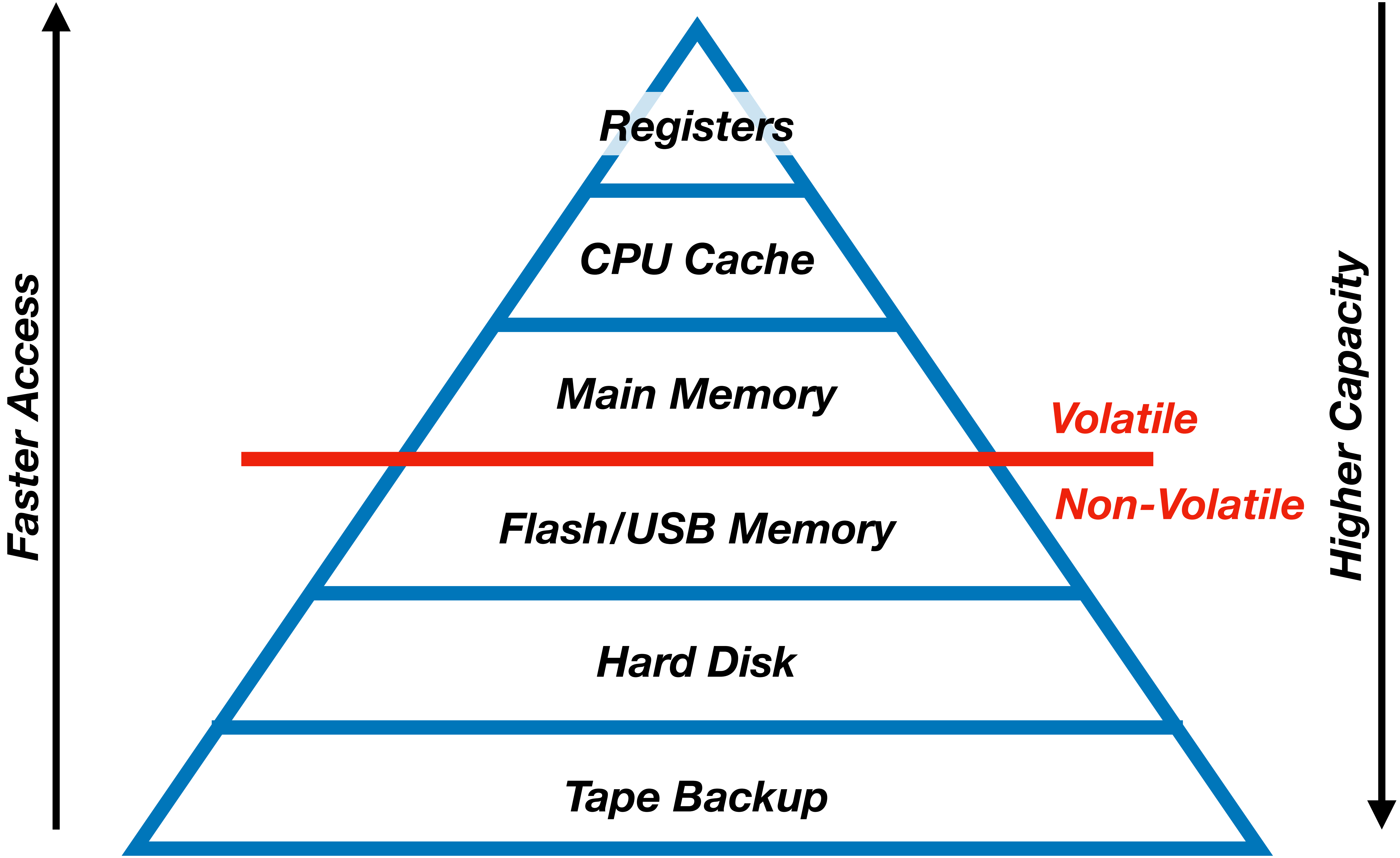
Database Management Systems (DBMS)



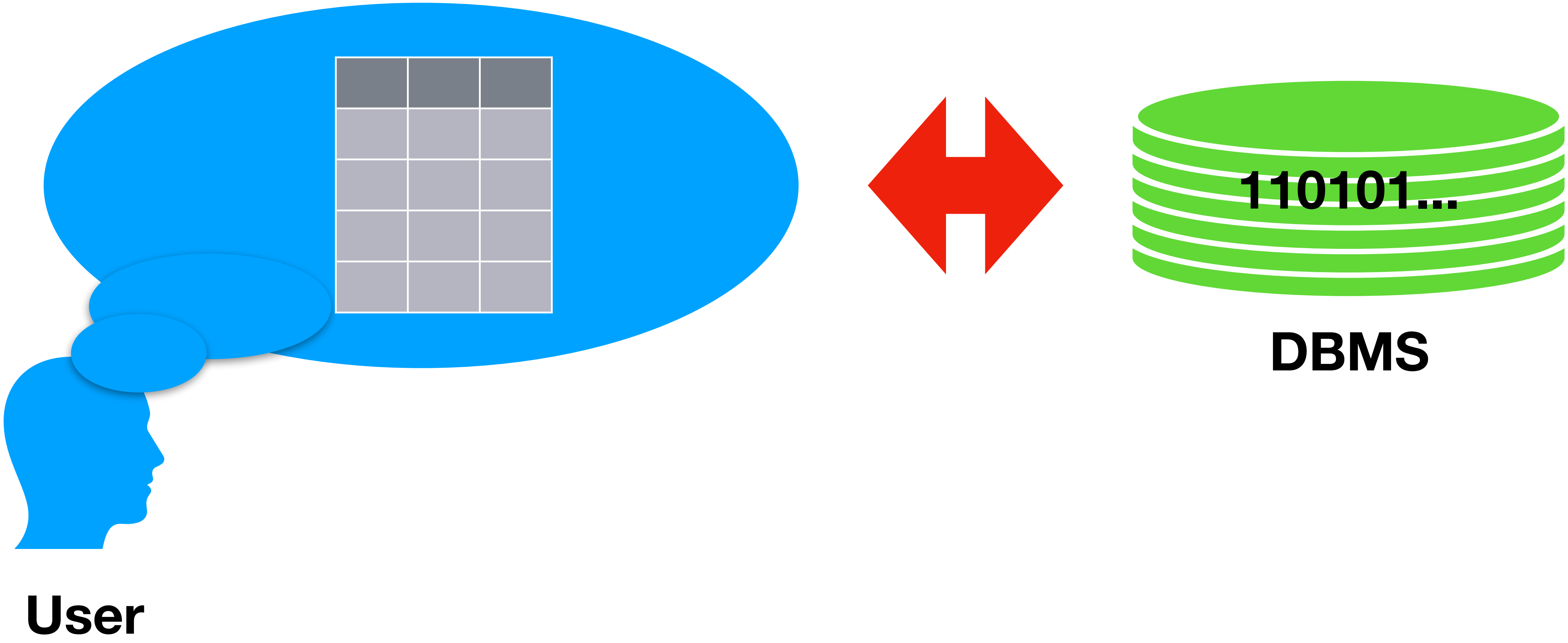
The Memory Hierarchy



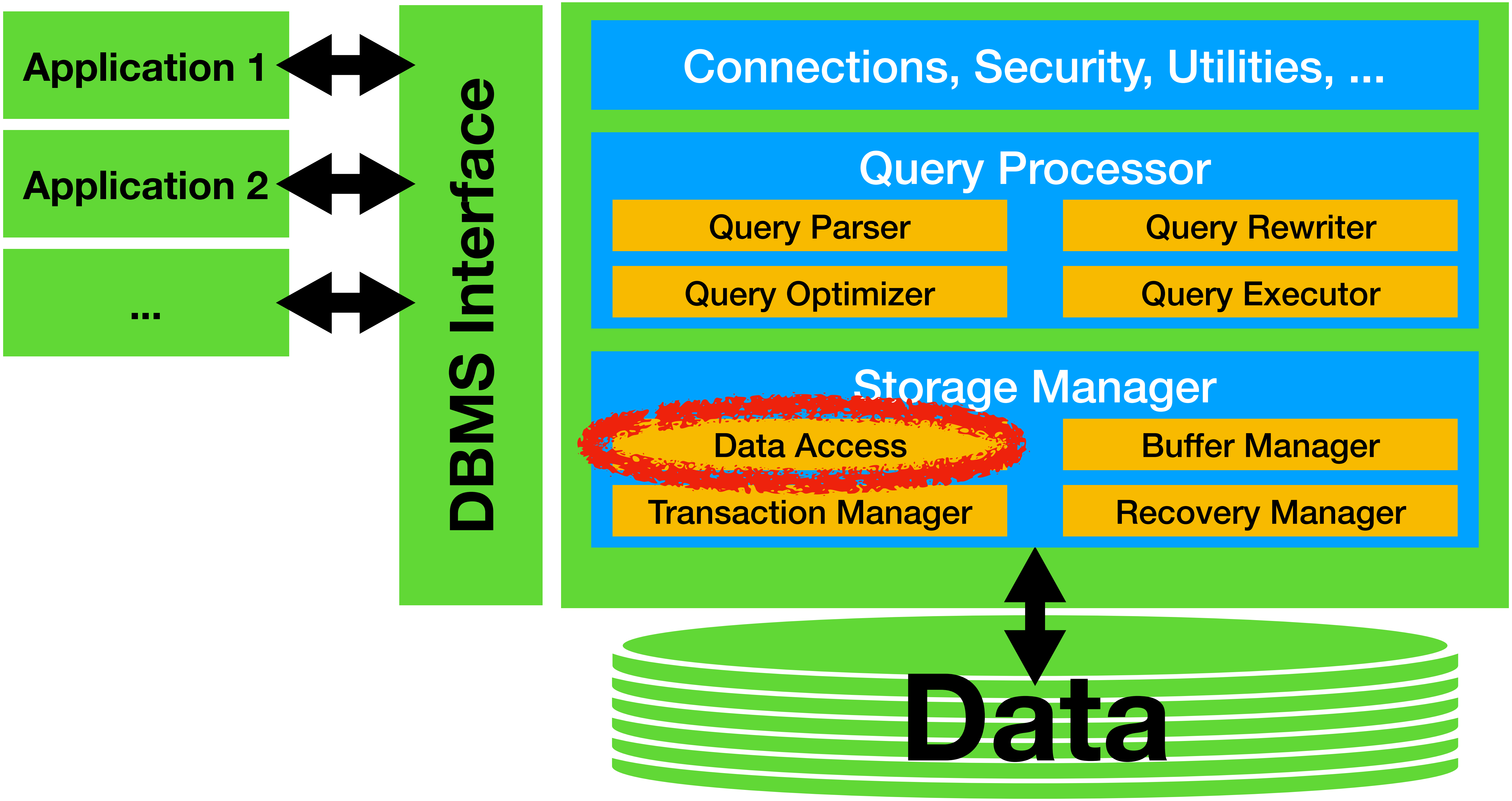
The Memory Hierarchy



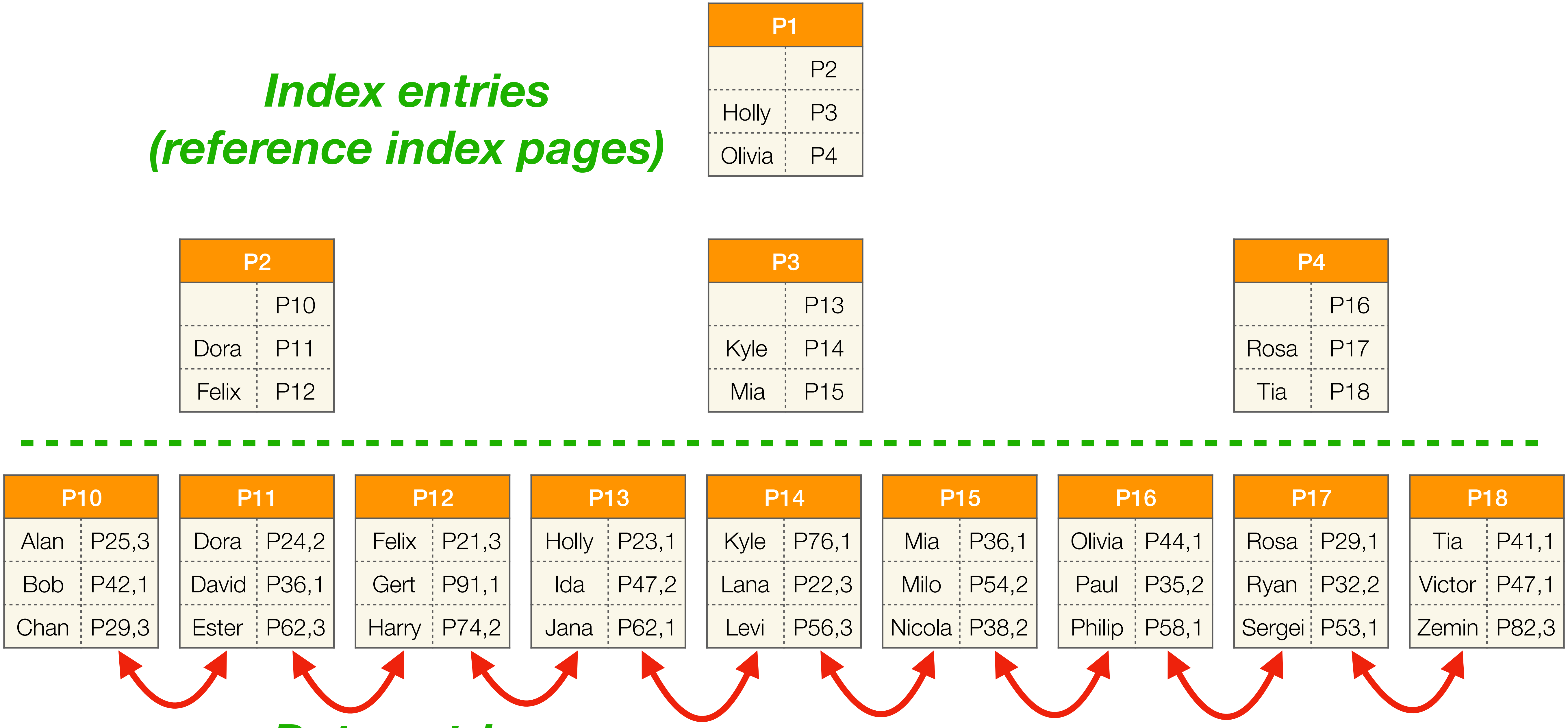
Logical Perspective vs. Physical Storage



Database Management Systems (DBMS)



B+ Tree Index



Index entries (reference index pages)

P1	
	P2
Holly	P3
Olivia	P4

P2	
	P10
Dora	P11
Felix	P12

P3	
	P13
Kyle	P14
Mia	P15

P4	
	P16
Rosa	P17
Tia	P18

P10	
Alan	P25,3
Bob	P42,1
Chan	P29,3

P11	
Dora	P24,2
David	P36,1
Ester	P62,3

P12	
Felix	P21,3
Gert	P91,1
Harry	P74,2

P13	
Holly	P23,1
Ida	P47,2
Jana	P62,1

P14	
Kyle	P76,1
Lana	P22,3
Levi	P56,3

P15	
Mia	P36,1
Milo	P54,2
Nicola	P38,2

P16	
Olivia	P44,1
Paul	P35,2
Philip	P58,1

P17	
Rosa	P29,1
Ryan	P32,2
Sergei	P53,1

P18	
Tia	P41,1
Victor	P47,1
Zemin	P82,3

Data entries (reference data pages)

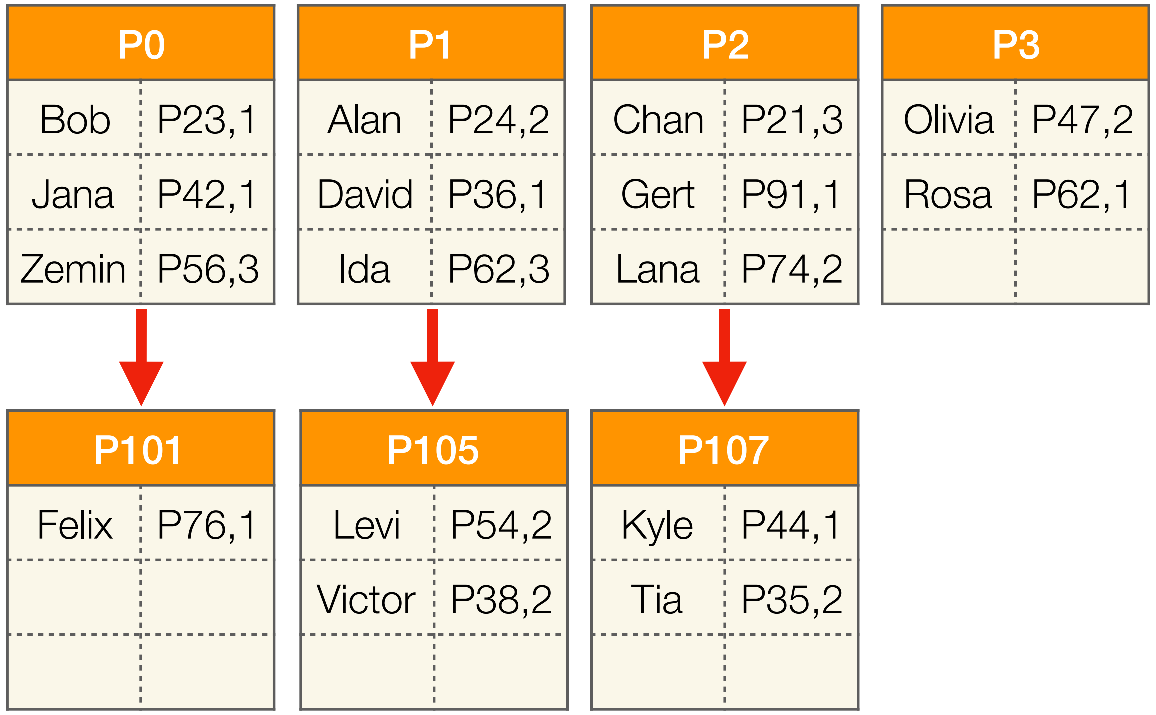
Index by Student Name

Static Hash Index

Hash Function (Not Stored)

Key	Hash
Alan	1
Bob	0
Chan	2
Dora	5
David	1
Ester	7
Felix	4
Gert	2
Holy	7
Ida	1
Jana	0
Kyle	6
Lana	6
Levi	5
Olivia	3
Philip	7
Rosa	3
Tia	6
Victor	5
Zemin	4

PageID = Hash % NrBuckets



P0	
Bob	P23,1
Jana	P42,1
Zemin	P56,3

P1	
Alan	P24,2
David	P36,1
Ida	P62,3

P2	
Chan	P21,3
Gert	P91,1
Lana	P74,2

P3	
Olivia	P47,2
Rosa	P62,1

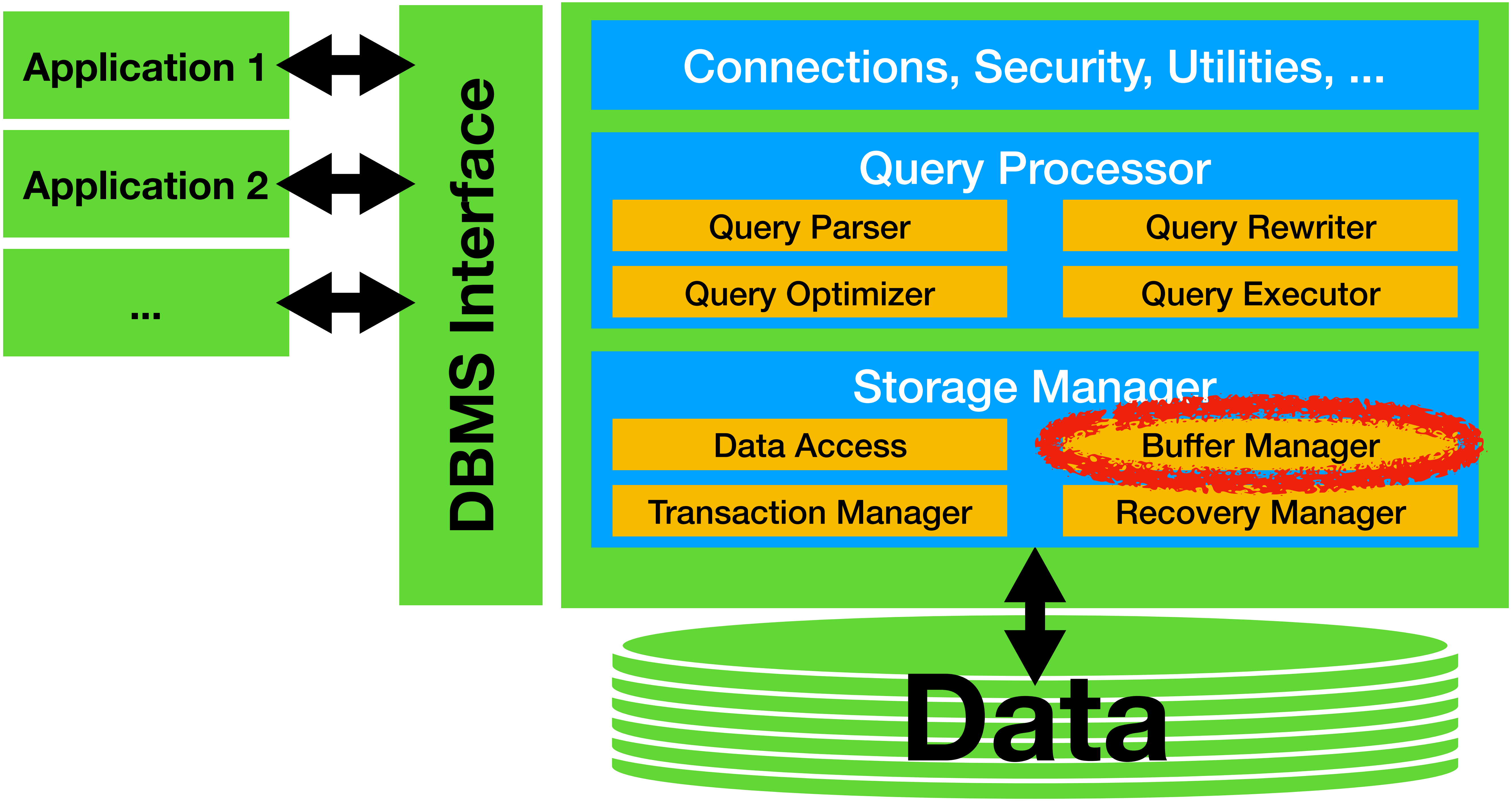
P101	
Felix	P76,1

P105	
Levi	P54,2
Victor	P38,2

P107	
Kyle	P44,1
Tia	P35,2

Overflow Pages

Database Management Systems (DBMS)



Application 1

Application 2

...

DBMS Interface

Connections, Security, Utilities, ...

Query Processor

Query Parser

Query Rewriter

Query Optimizer

Query Executor

Storage Manager

Data Access

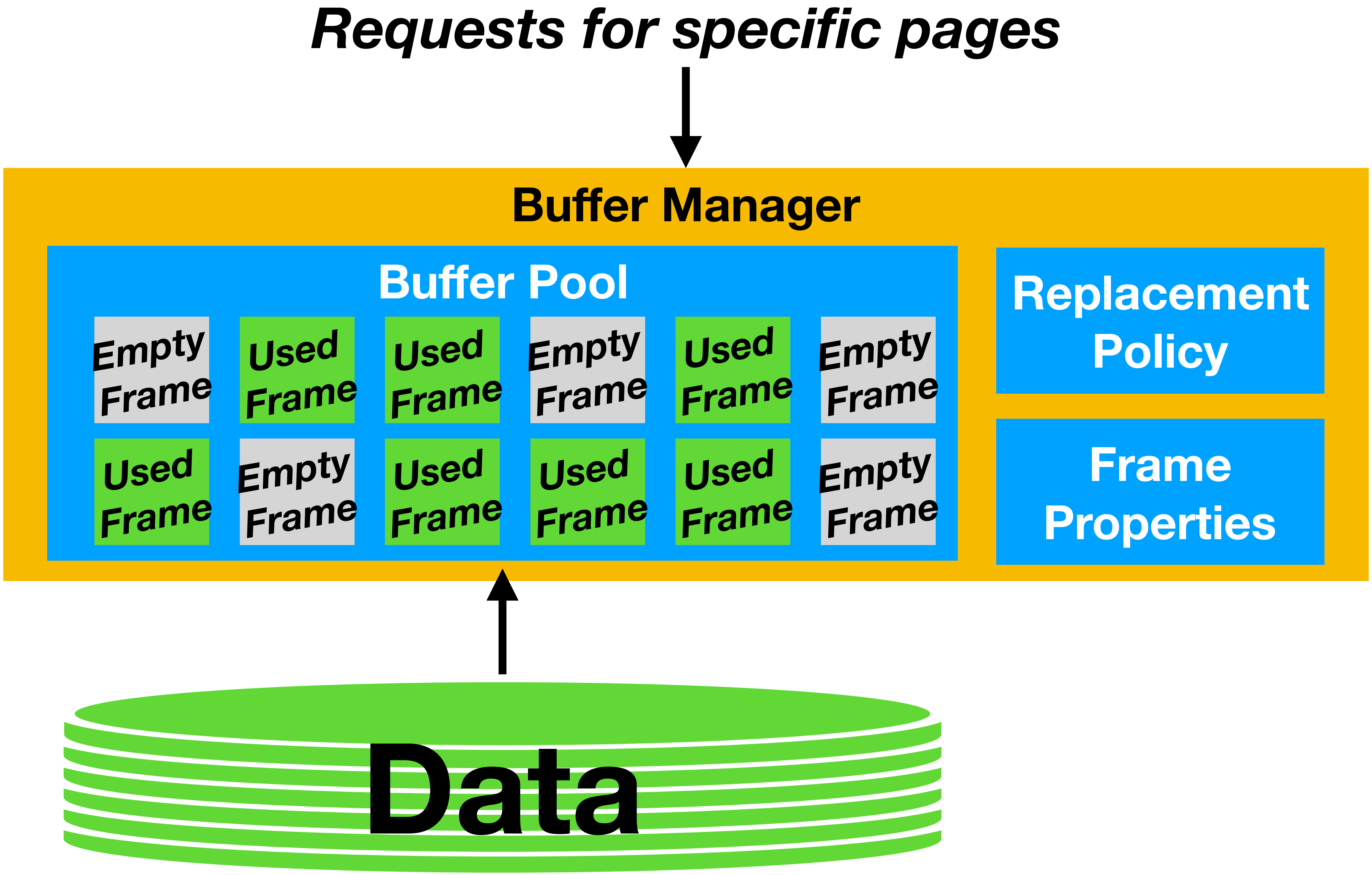
Buffer Manager

Transaction Manager

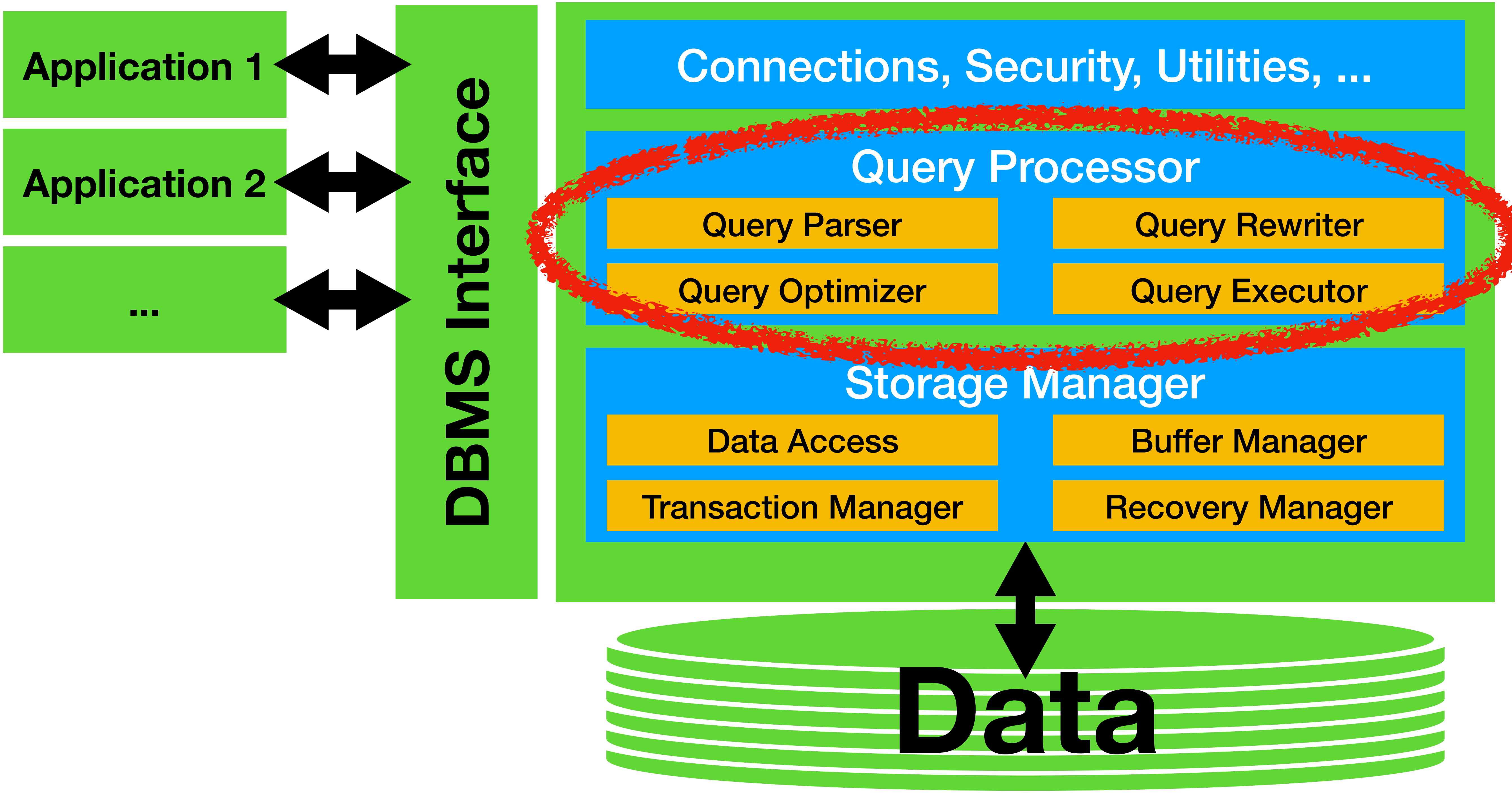
Recovery Manager

Data

Buffer Manager Illustration



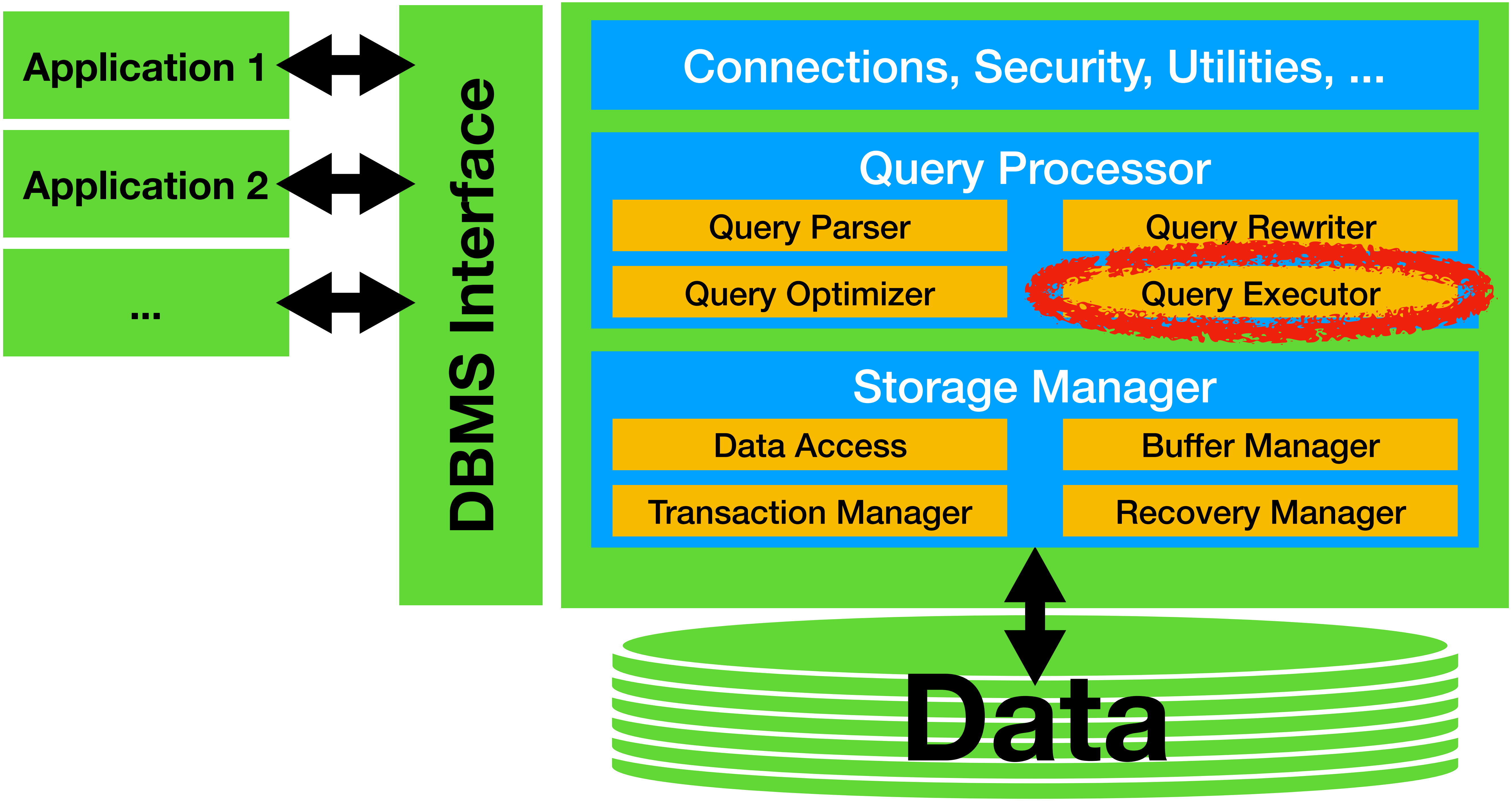
Database Management Systems (DBMS)



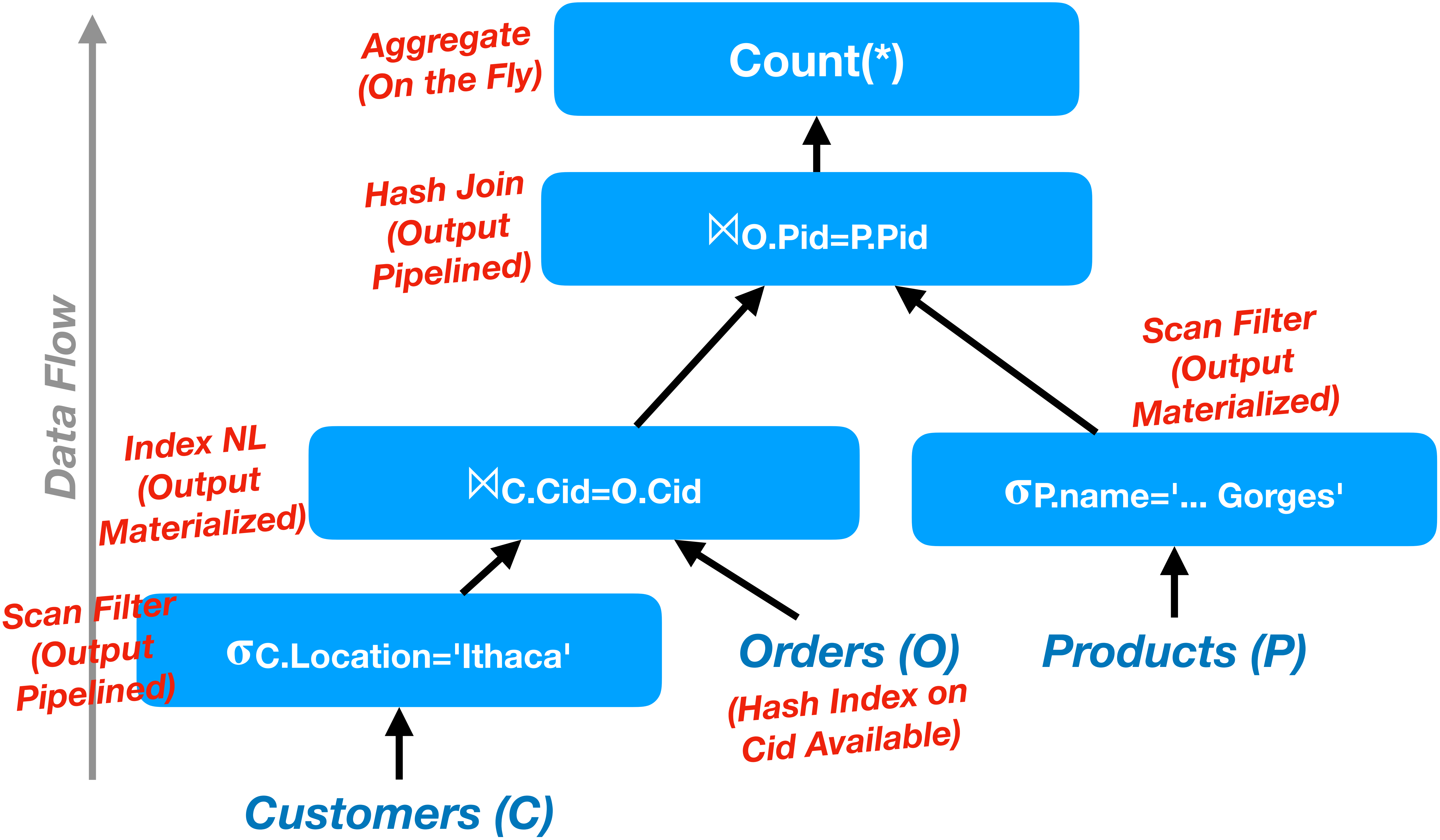
Query Processing

- Input query is parsed (**Parser**) and simplified (**Rewriter**)
- **Query optimizer** generates optimized execution plan
- Executing plan (**Executor**) produces query result

Database Management Systems (DBMS)



Example Query Plan


Data Flow
Aggregate (On the Fly)
Count(*)
Hash Join (Output Pipelined)
⋈O.Pid=P.Pid
Scan Filter (Output Materialized)
Index NL (Output Materialized)
⋈C.Cid=O.Cid
σP.name='... Gorges'
Scan Filter (Output Pipelined)
σC.Location='Ithaca'
Orders (O)
(Hash Index on Cid Available)
Products (P)
Customers (C)

Block Nested Loop Join

$\bowtie_{E.Sid=S.Sid}$

```
For ep in PageBlocks(E, b):
 LoadPages(ep)
 For sp in Pages(S):
  LoadPage(sp)
  For et in Tuples(ep), st in Tuples(sp):
   If (et.Sid=st.Sid):
    Output(et ⋈ st)
```

Hash Join: Phase 1

$\bowtie_{E.Sid=S.Sid}$

```
For ep in Pages(E):
 LoadPage(ep)
 For et in Tuples(ep):
  Add et to EB[Hash(et)]
   If (Full(EB[Hash(et)])):
    WriteAndClear(EB[Hash(et)])
```

Hash Join: Phase 1

$\bowtie_{E.Sid=S.Sid}$

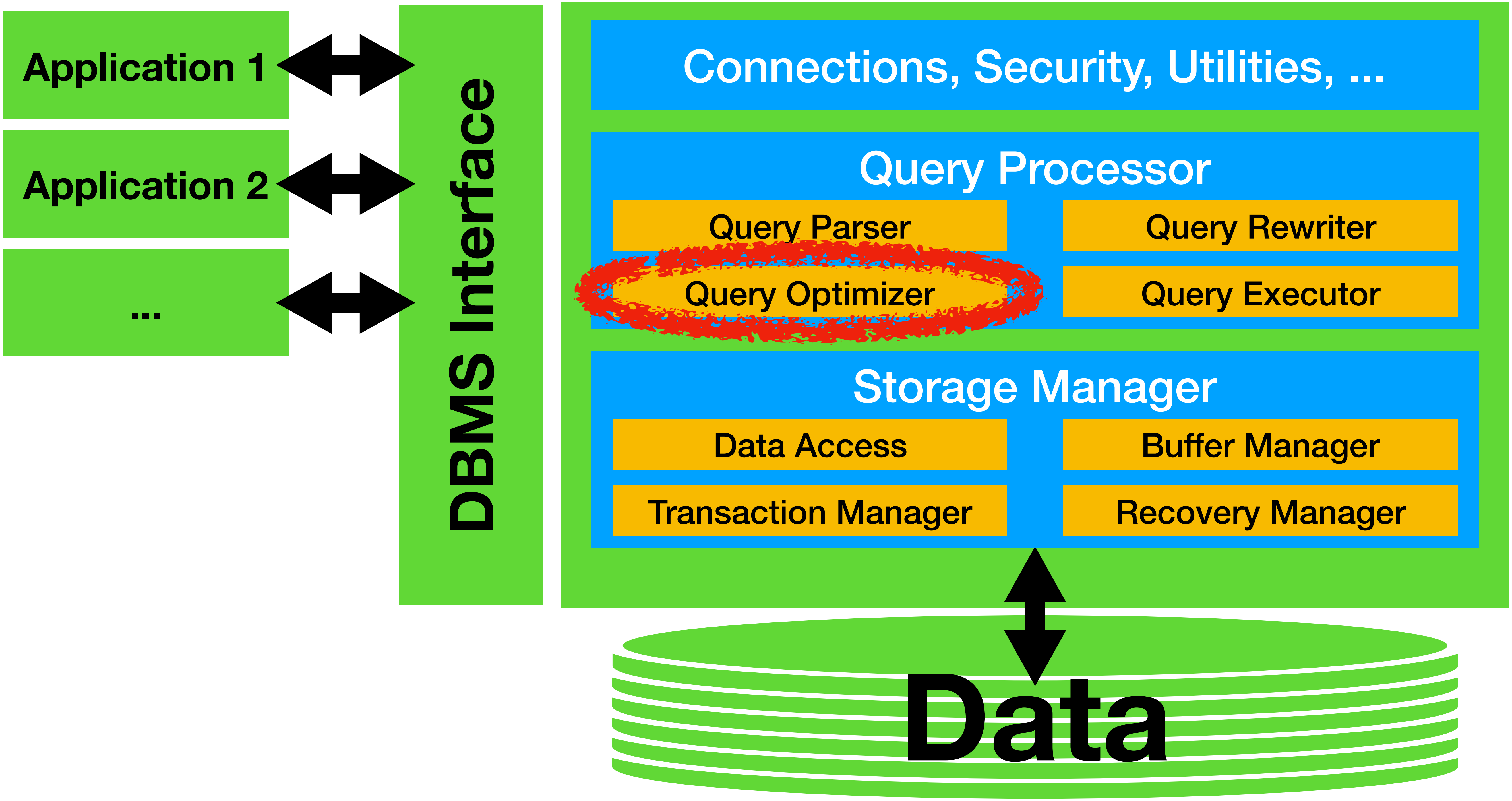
```
For sp in Pages(S):
 LoadPage(sp)
 For st in Tuples(sp):
  Add st to SB[Hash(st)]
   If (Full(SB[Hash(st)])):
    WriteAndClear(SB[Hash(st)])
```

Hash Join: Phase 2

$\bowtie_{E.Sid=S.Sid}$

```
For h in Hash Values:
  LoadPages(EB[h])
  For sp in Pages(SB[h]):
    Load(sp)
    For ep in Pages(EB[h]), st in sp, et in ep:
      If (et.Sid=st.Sid):
        Output(et ⋈ st)
```

Database Management Systems (DBMS)



Application 1

Application 2

...

DBMS Interface

Connections, Security, Utilities, ...

Query Processor

Query Parser

Query Rewriter

Query Optimizer

Query Executor

Storage Manager

Data Access

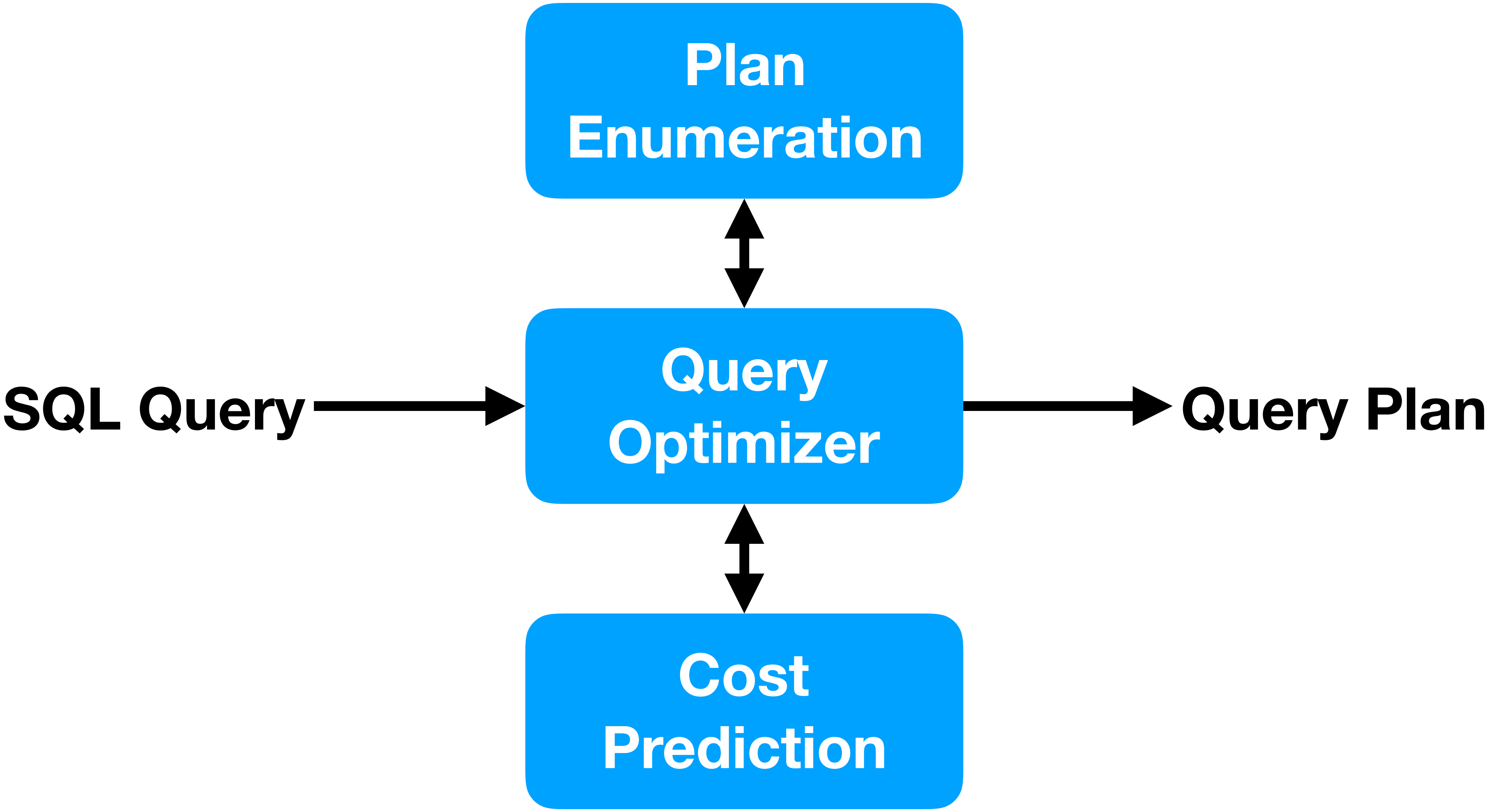
Buffer Manager

Transaction Manager

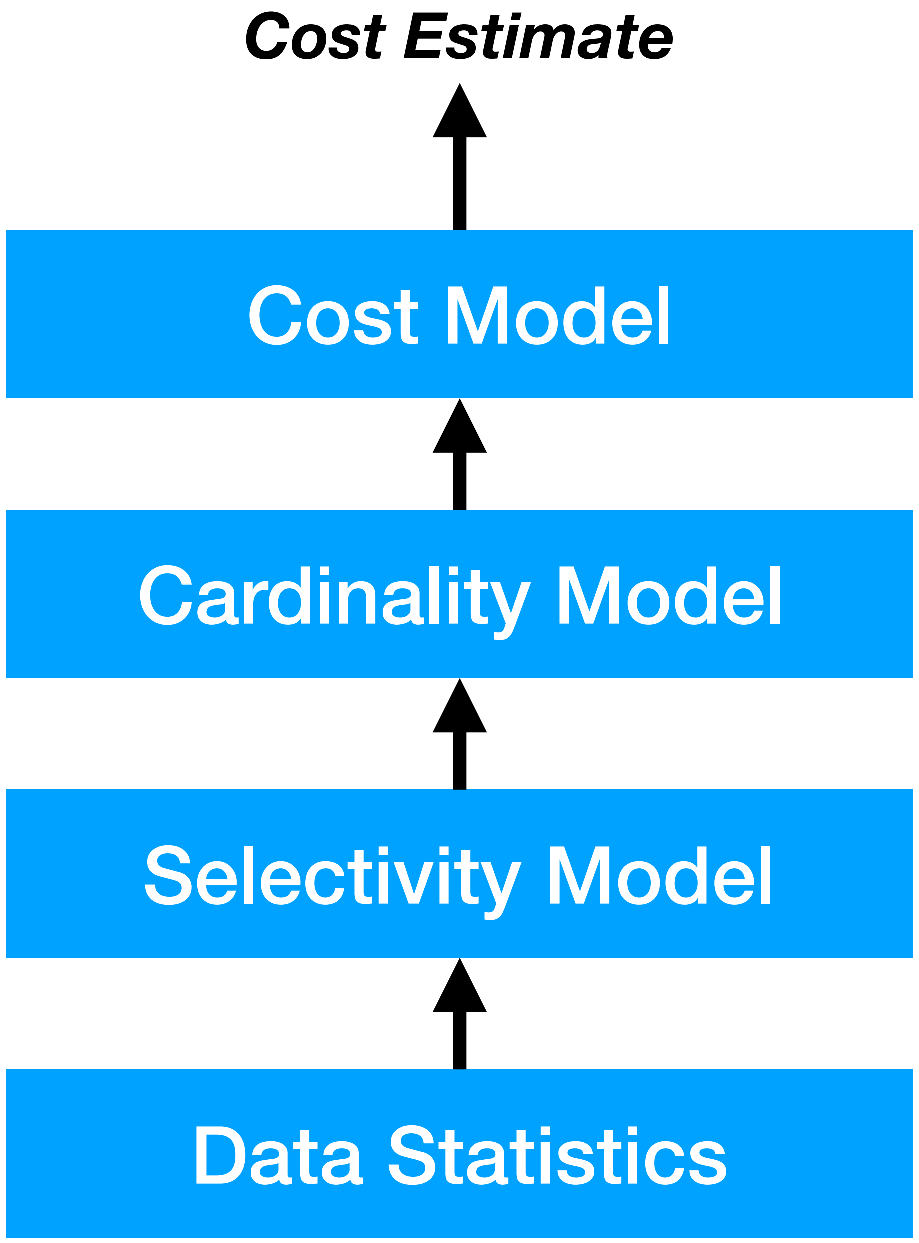
Recovery Manager

Data

Optimizer Overview



Cost Model Structure



Cost Estimate

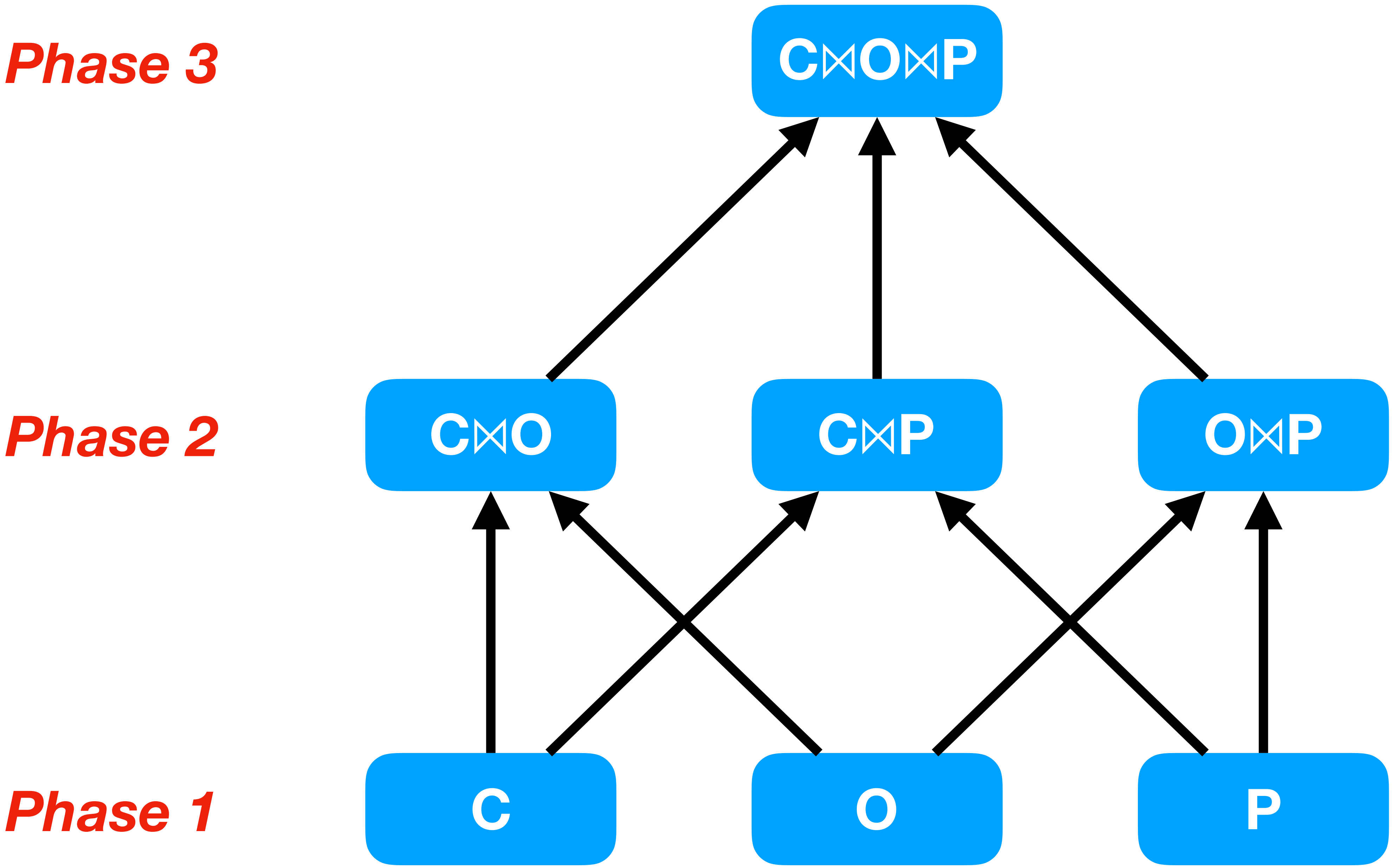
Cost Model

Cardinality Model

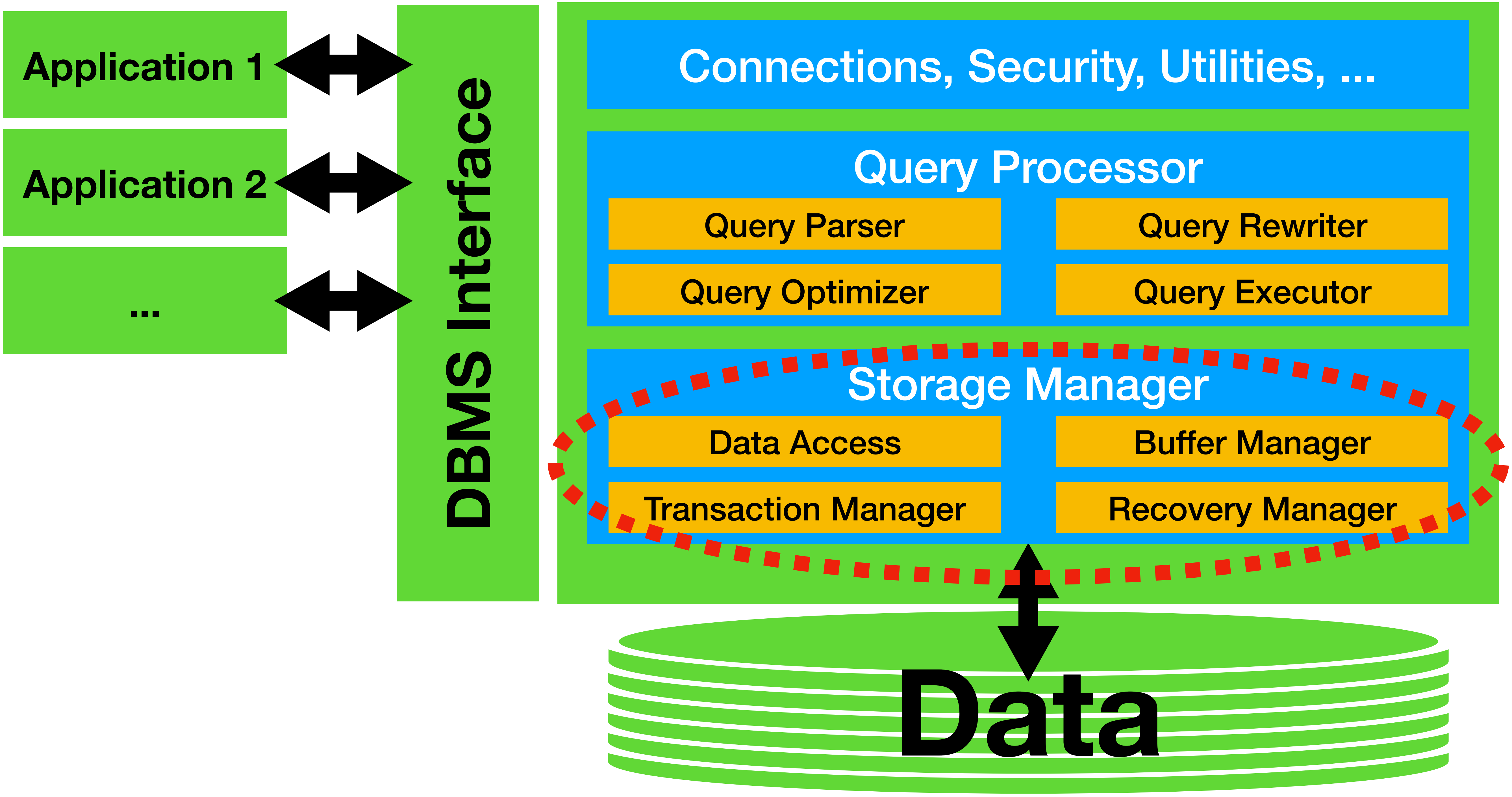
Selectivity Model

Data Statistics

Dynamic Programming



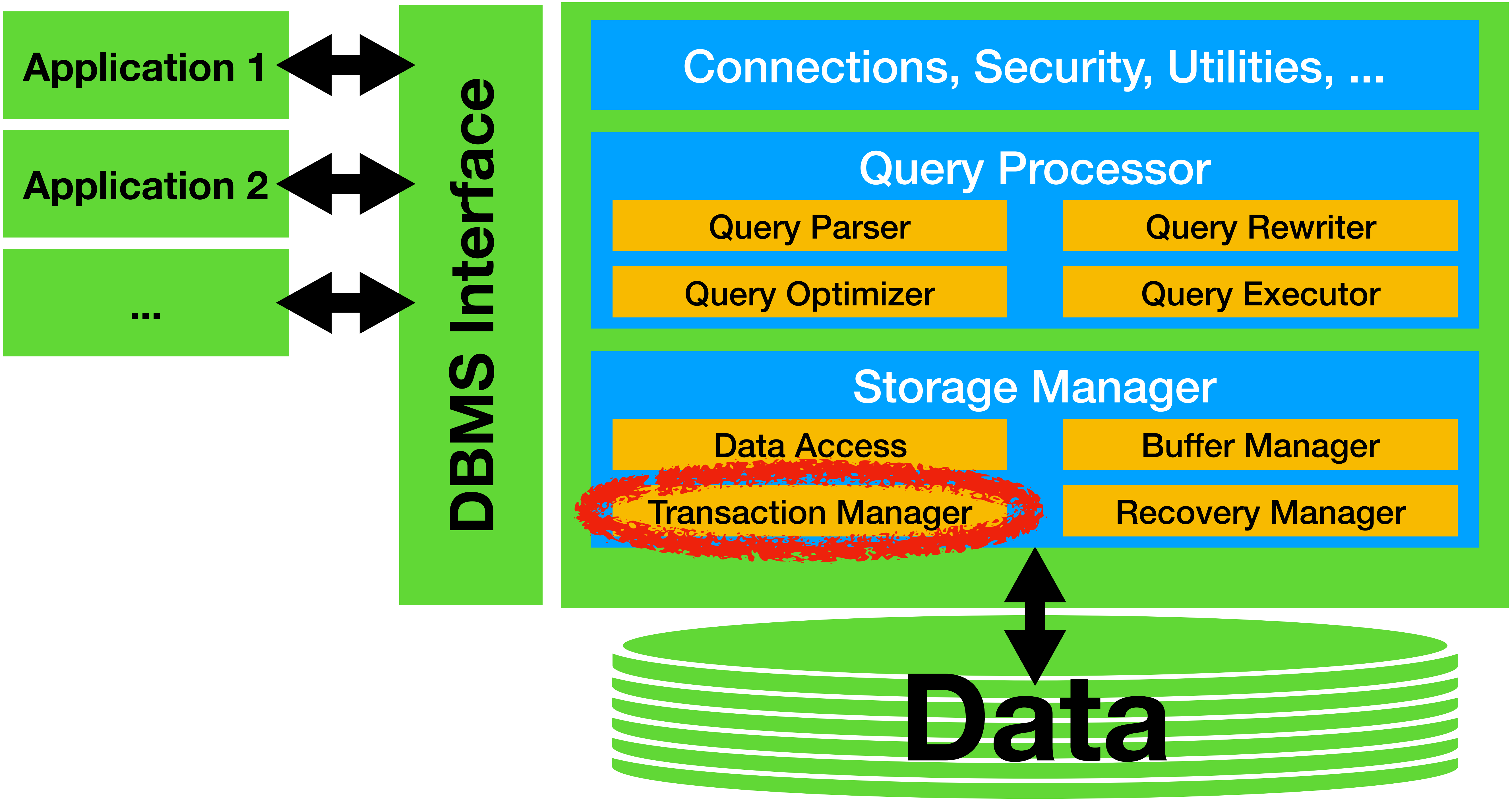
Database Management Systems (DBMS)



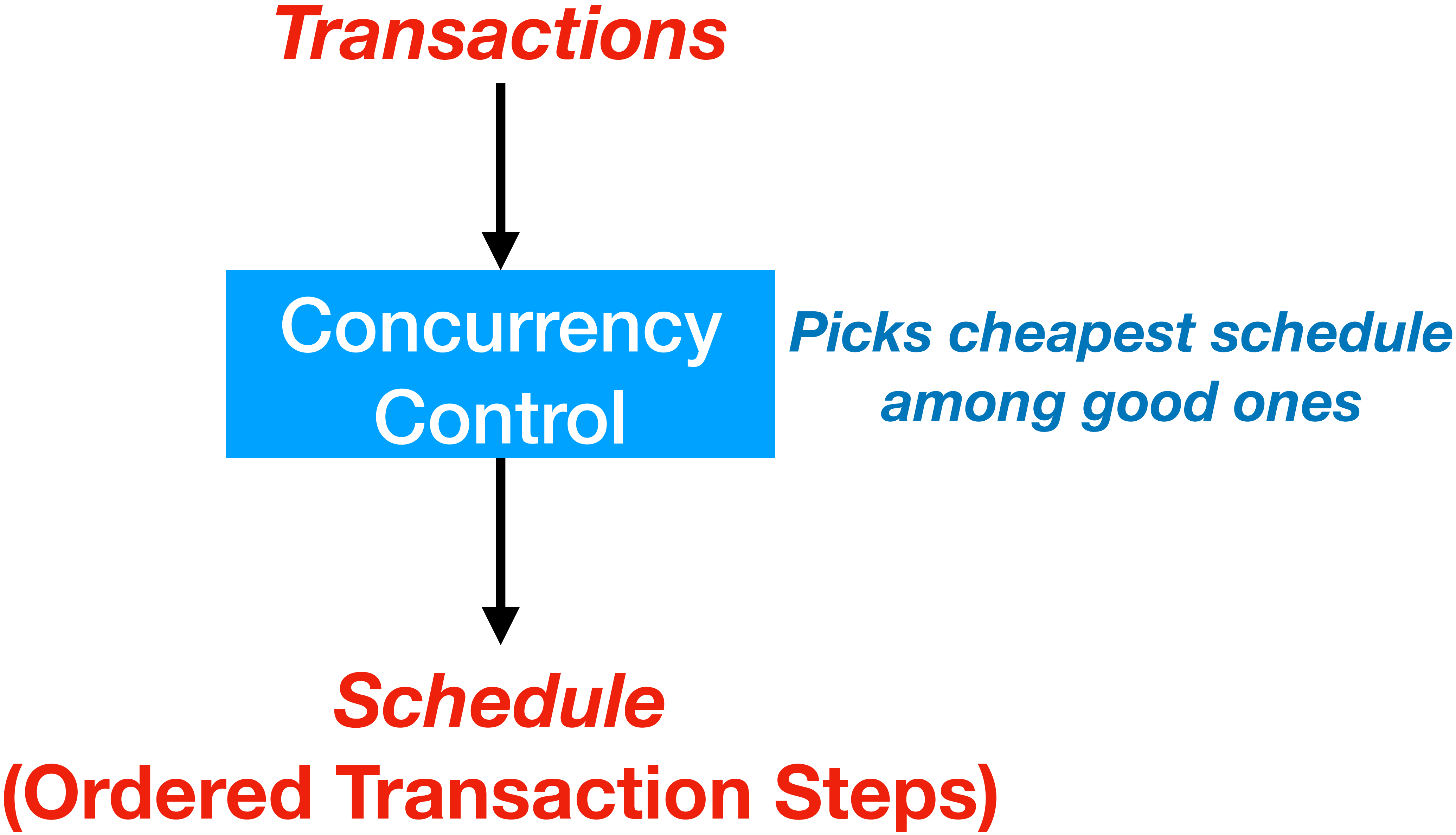
ACID Guarantees

- Most RDBMS give **ACID guarantees** for transactions
- **A: Atomicity** (either execute all or nothing)
- **C: Consistency** (enforce all integrity constraints)
- **I: Isolation** (avoid interleaving transactions badly)
- **D: Durability** (ensure that updates are not lost)

Database Management Systems (DBMS)



Concurrency Control



Transactions

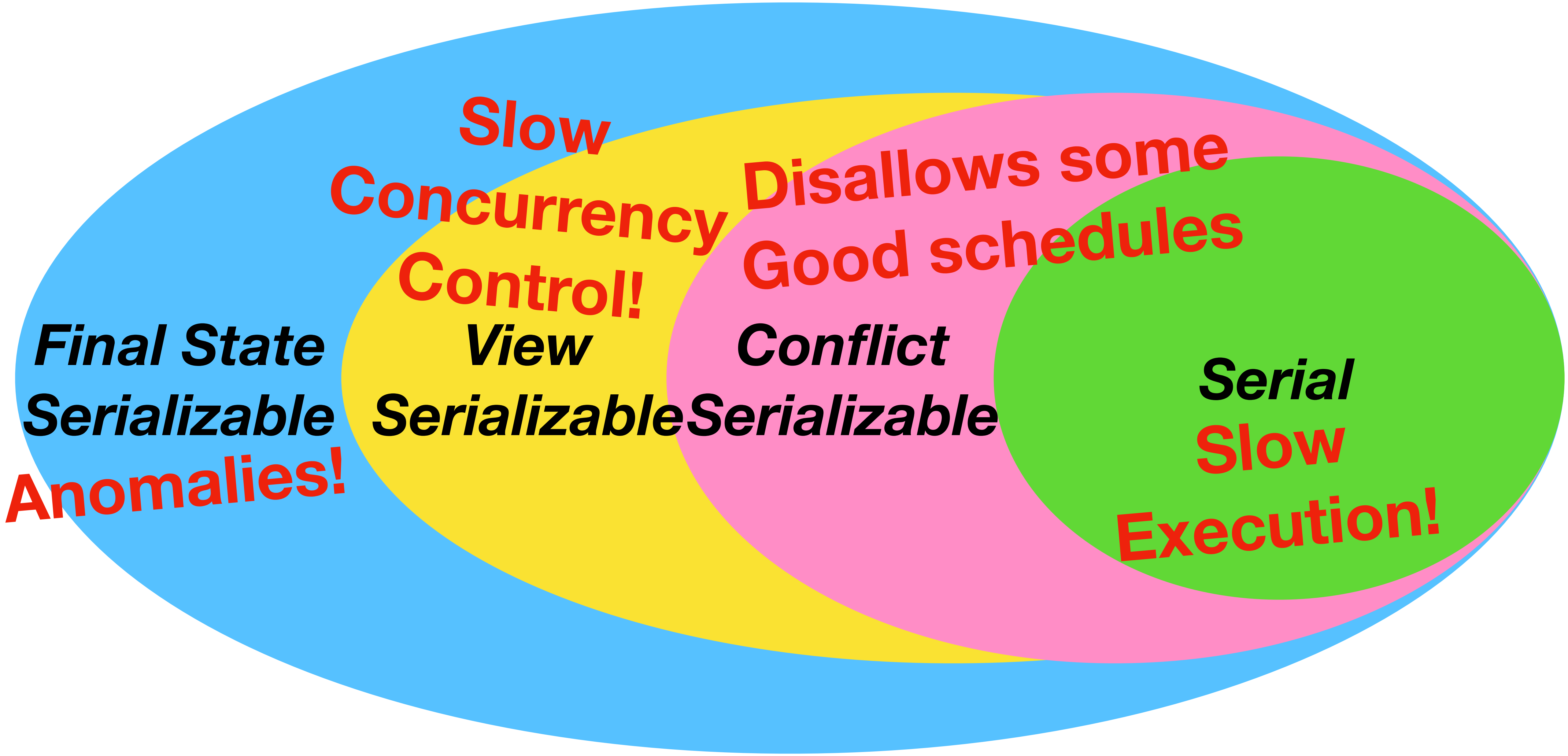
Concurrency Control

Picks cheapest schedule among good ones

Schedule

(Ordered Transaction Steps)

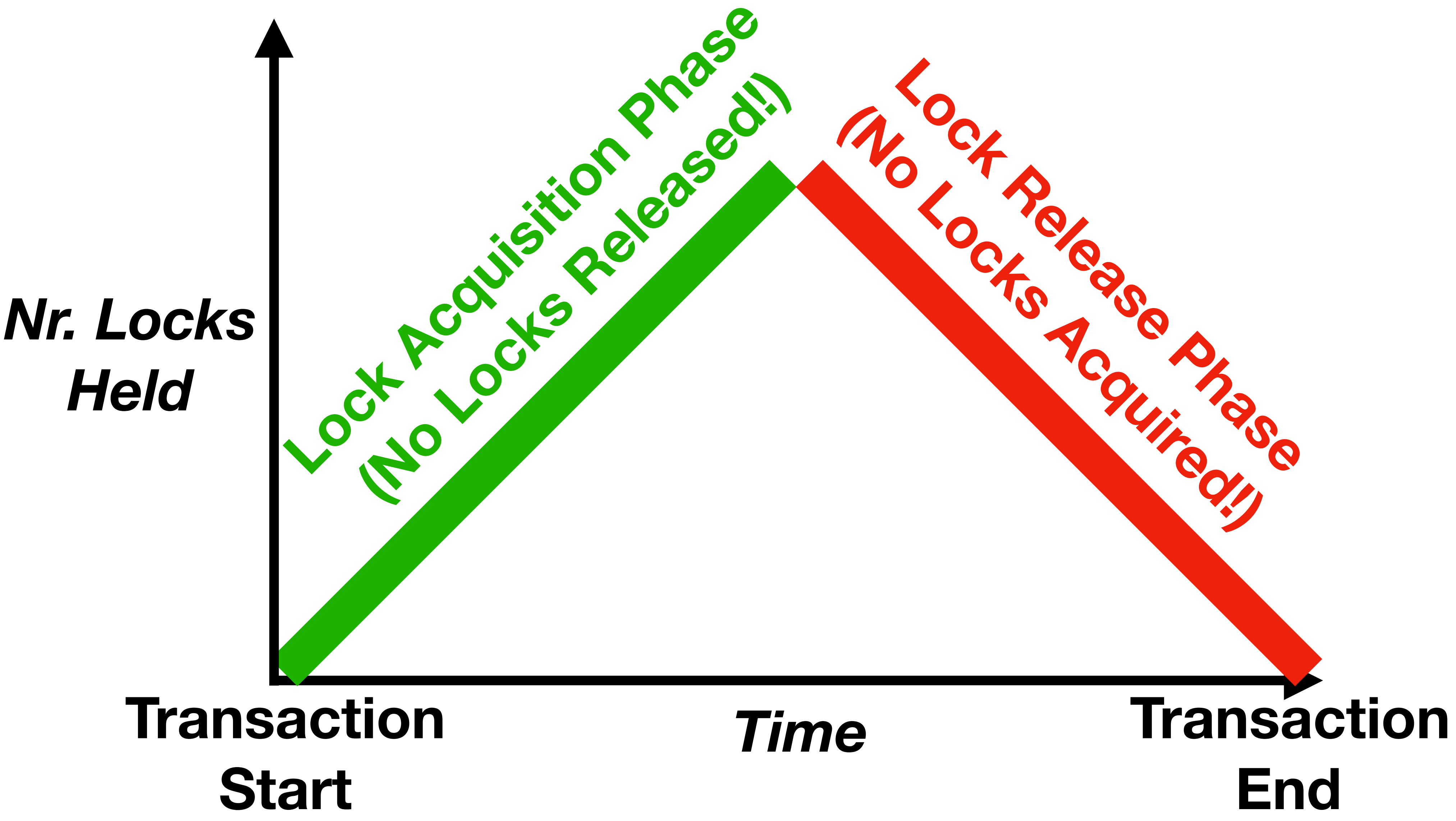
Overview of Classes of Schedules



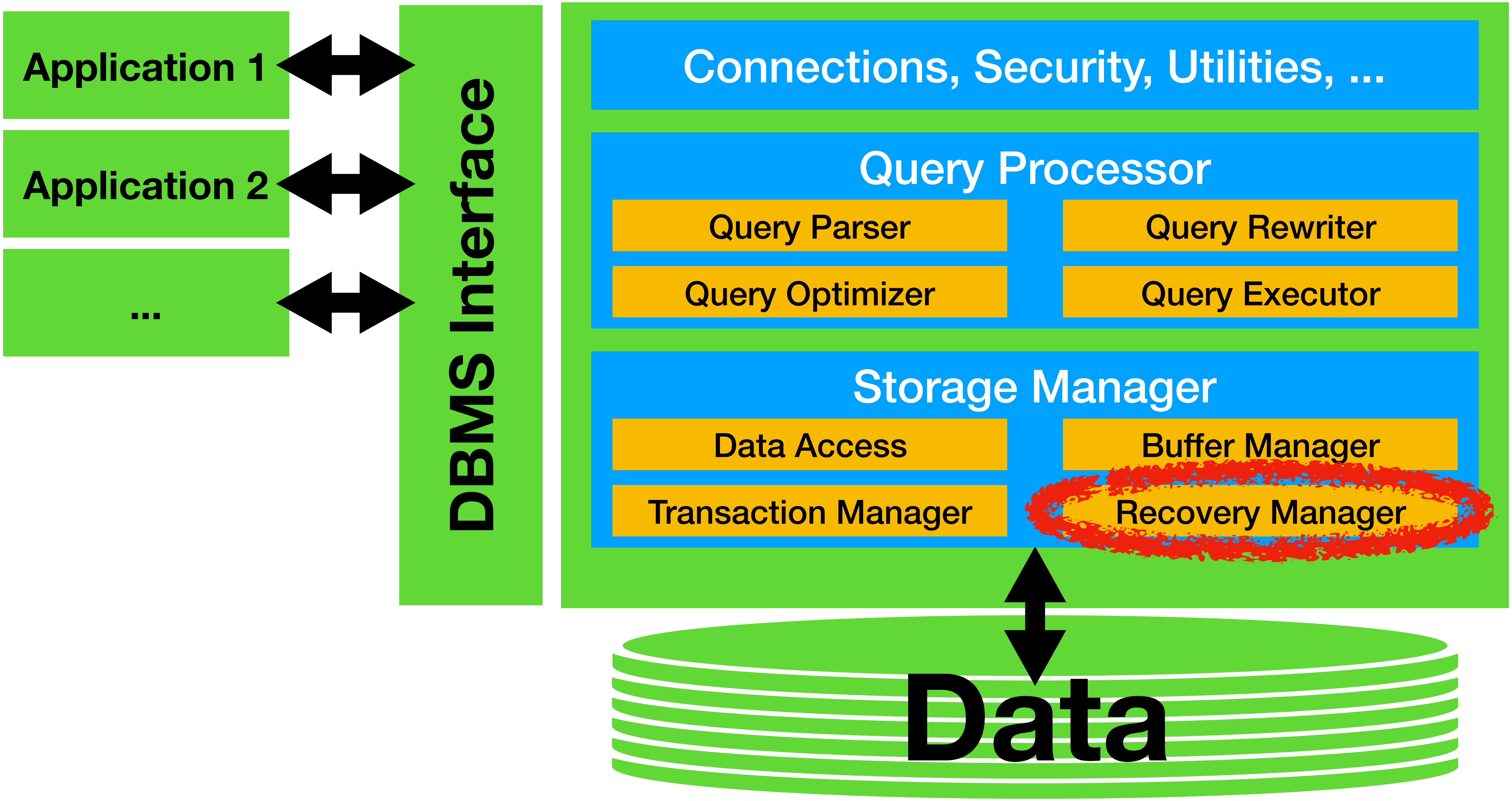
Anomalies!

All Schedules

The Two Phases of 2PL



Database Management Systems (DBMS)



Application 1

Application 2

...

DBMS Interface

Connections, Security, Utilities, ...

Query Processor

Query Parser

Query Rewriter

Query Optimizer

Query Executor

Storage Manager

Data Access

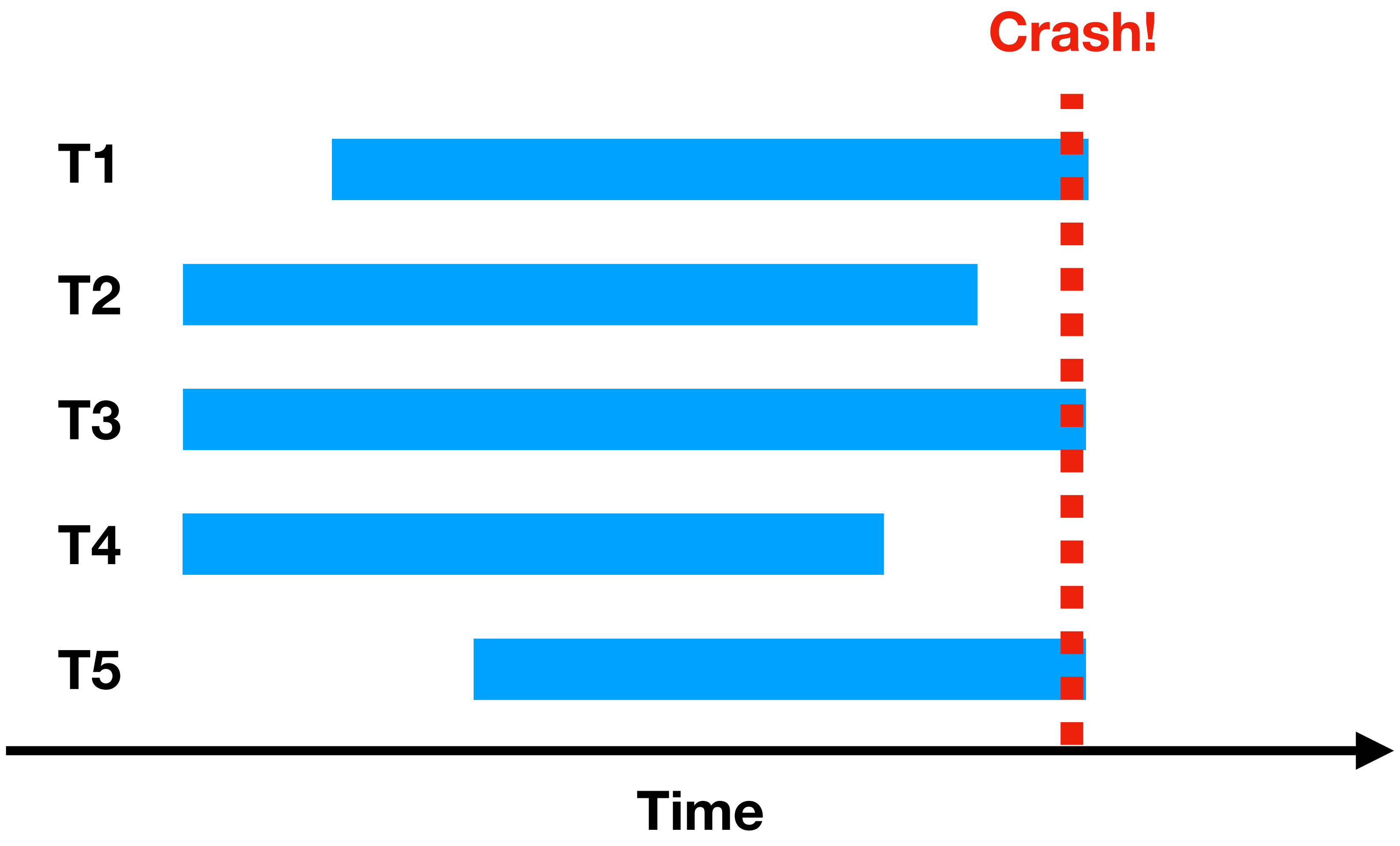
Buffer Manager

Transaction Manager

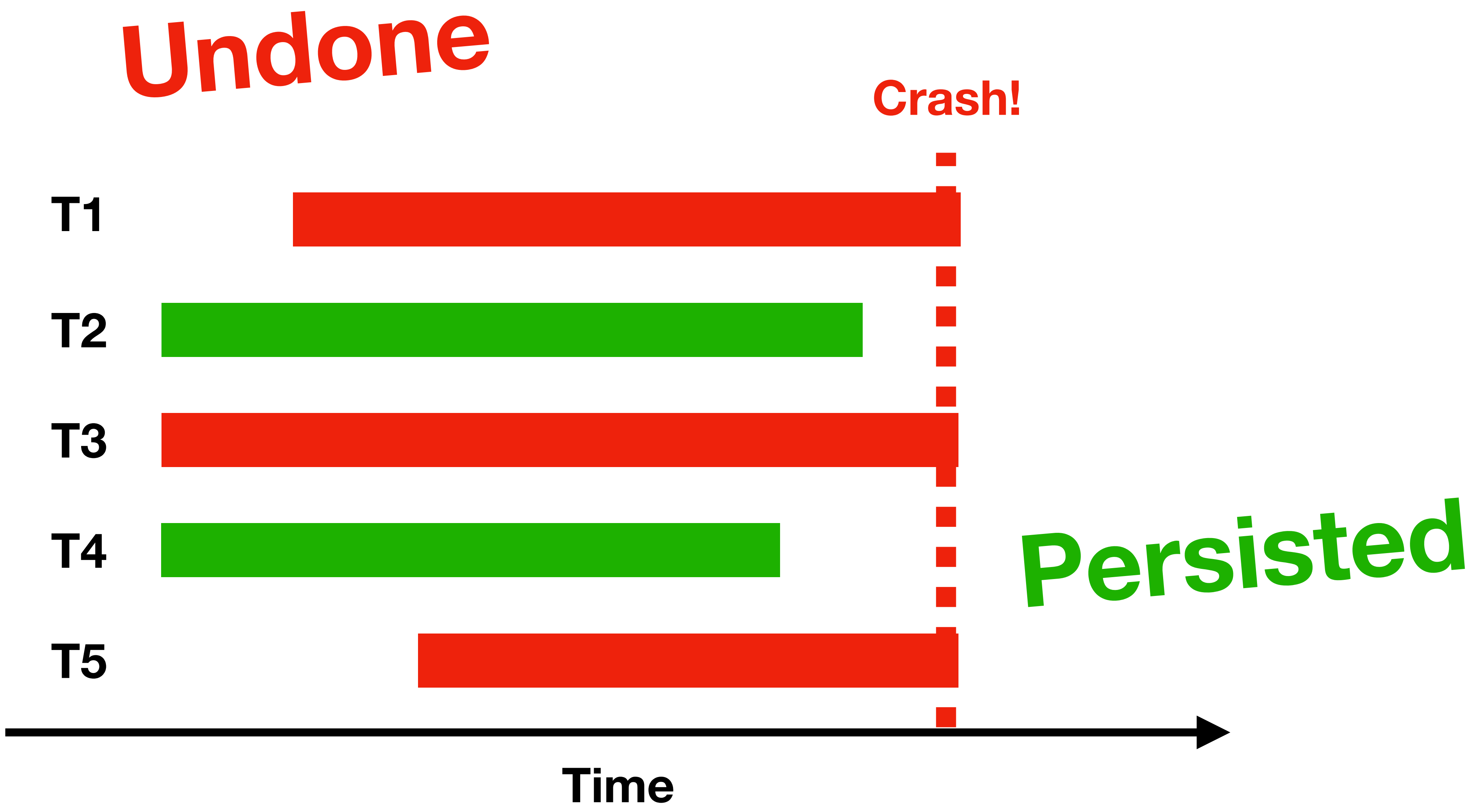
Recovery Manager

Data

Desired Behavior



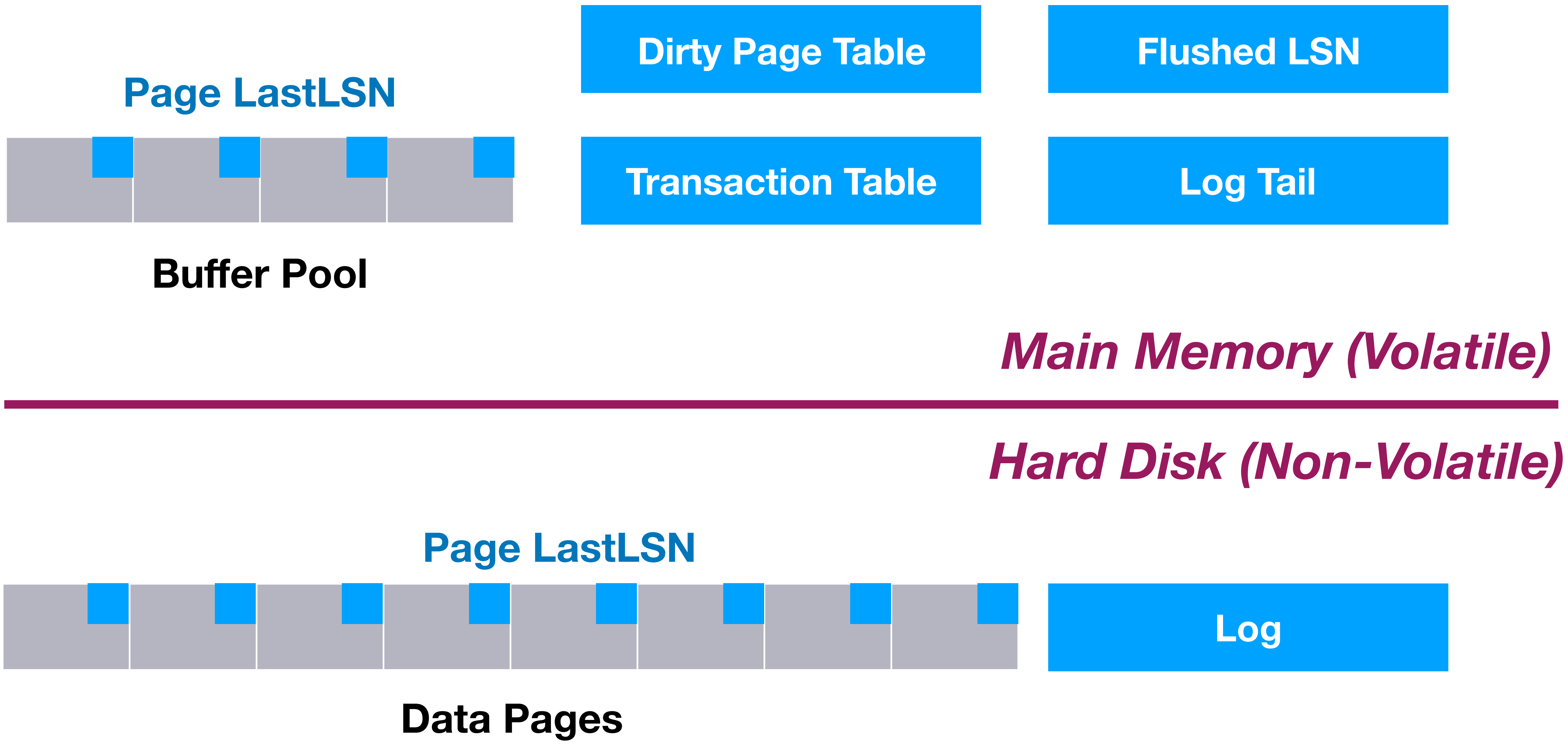
Desired Behavior



ARIES Algorithm Overview

- One of the most popular **recovery algorithms**
- Uses **write-ahead logging** at run time
- Executes **multiple phases** after a crash:
    - **Analysis**: determine transactions to undo/redo via log
    - **Redo**: get back to state directly before the crash
    - **Undo**: undo effects of aborted transactions

ARIES Data Structures



Page LastLSN

Buffer Pool

Dirty Page Table

Flushed LSN

Transaction Table

Log Tail

Main Memory (Volatile)

Hard Disk (Non-Volatile)

Page LastLSN

Data Pages

Log

?