CSE 541: Database Systems I

Overview of Database Systems

Slides Credit: Tianzheng Wang @ SFU

Database System = Database + DBMS

Database (DB): data organized in some way

- Model real-world applications
- Tables/entities/relations (e.g., students, courses)
- Relationships (e.g., a student is taking CMPT 454)
- A relational DB is a collection of tables

Database Management System (DBMS): software package manages databases

- Facilitate access to databases
- Accesses = queries and updates = reads and writes

Why not use files?

Side by Side: Files vs. DBMS

Using files:

- Application handles data access directly
  - Buffering
  - Writing data to disk
  - Optimizing performance
  - Protecting data from inconsistencies (concurrency issues)
  - Security and access control
- Special application code for different queries, and HW/SW platforms
- Custom recovery code

Using DBMS:

- Application calls DBMS for data access
- DBMS handles issues related to data access
  - Data independence
  - Buffering, performance optimization, consistency, security…
- Easier application development
  - Universal interface (e.g., SQL)
  - Same code for different platforms
- No need to worry about recovery

Relational Model

Attribute name

Table: Cities

City	Population
State College	41,992
Pittsburg	302,407
Philadelphia	1,581,000

Row (record/tuple)

Column

- A table consists of rows (aka records, tuples) and columns
- A row consists of fields, each column has a unique name
- A column includes all fields under a specific attribute
- Tables are instances of *schemas*

Schema

- DBMS enforces schemas defined by database designer
- Schema is “metadata” that defines what a table looks like:
  - Column definitions: name and data type of each column
  - Constraints: rules that define a valid table, enforced by DBMS

```
CREATE TABLE checklists (
    todo_id INT AUTO_INCREMENT,
    task_id INT,
    todo VARCHAR(255) NOT NULL,
    is_completed TINYINT NOT NULL DEFAULT 0,
    CHECK(task_id > 0),
    PRIMARY KEY (todo_id , task_id),
    FOREIGN KEY (task_id) REFERENCES tasks(task_id)
    ON UPDATE RESTRICT ON DELETE CASCADE);
```

Operations in RDBMS

- SQL = DDL + DML.

- DDL: Data Definition Language
  - Define/ALTER Structure of data.
  - Meta info change. (Database, Table, Schema, etc.)

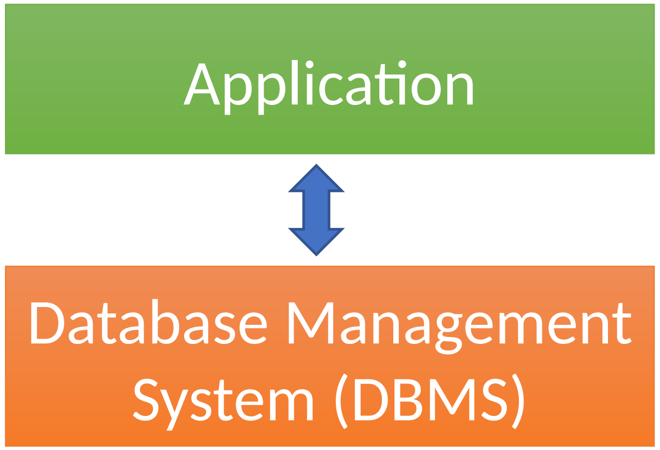
- DML: Data Manipulation Language
  - CRUD on actual data.

```
UPDATE Cities
SET Population = 100000
WHERE City = State College
```

```
SELECT * FROM Cities
WHERE Population < 2000000
```

Data Independence

- Application (i.e., user of DBMS) views data under the relational model
- DBMS uses physical, in-memory or on-disk representations of data



Conceptual view:
Tables, rows, columns…

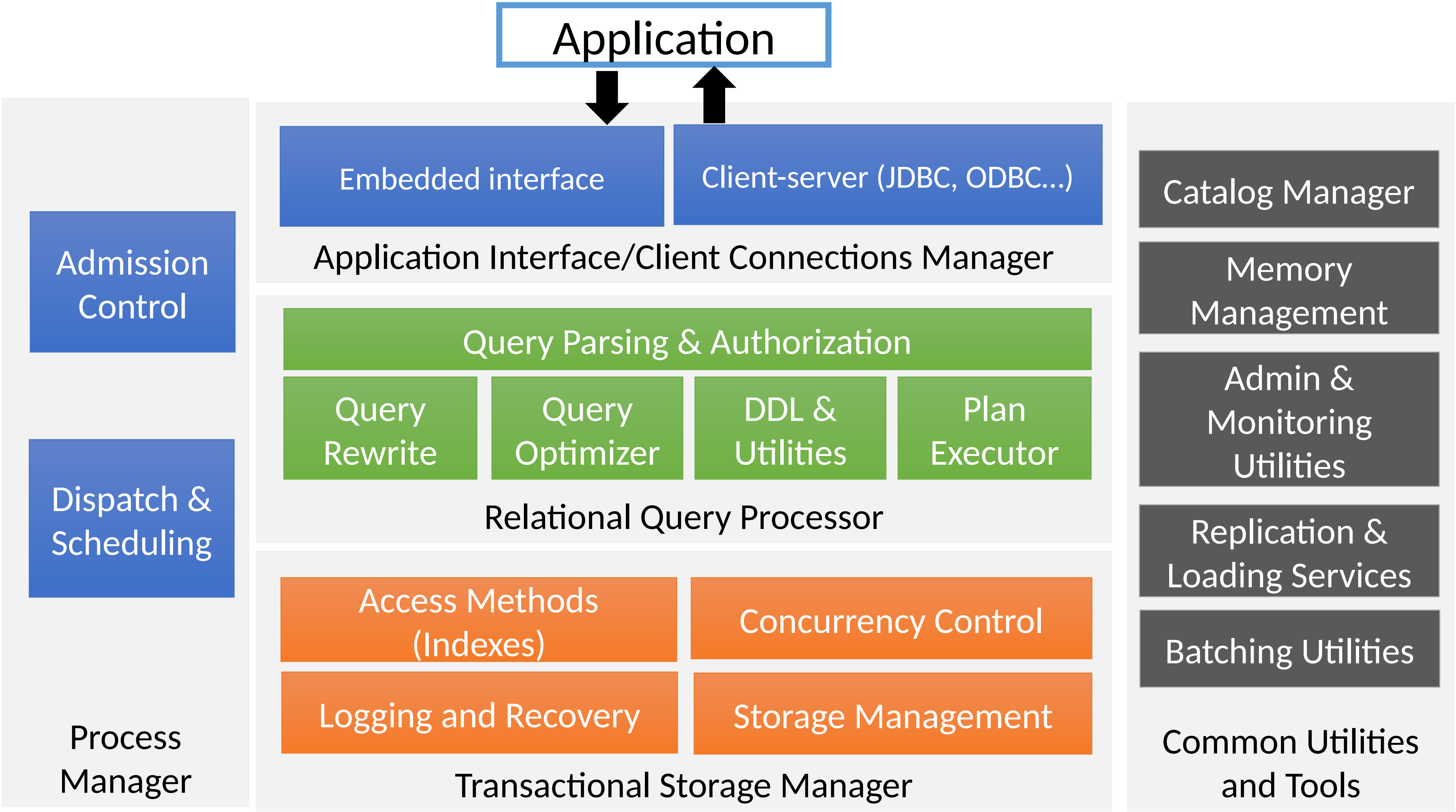
Maps conceptual view to *physical* view:
Bytes of data in memory or storage

- Regardless of implementation details, DBMS maintains the same relational conceptual view for applications
- **Logical** vs. **Physical** Data Independence

DBMS Functionality

- Interface to applications
- Manage SQL commands
  - **Admission control**: when to process the query?
  - Process management: assign thread to run the command
- Parse the command and generate a **query plan**
- Execute the query plan
  - **Storage management** and **transaction management** are involved.
  - Access data records while **maintaining constraints**
- Return results

DBMS Components



Running Example

- Q: *Which cities have at least 300k population?*


```
SELECT City
FROM Cities
WHERE Population
>=
      300k
```

Table: Cities

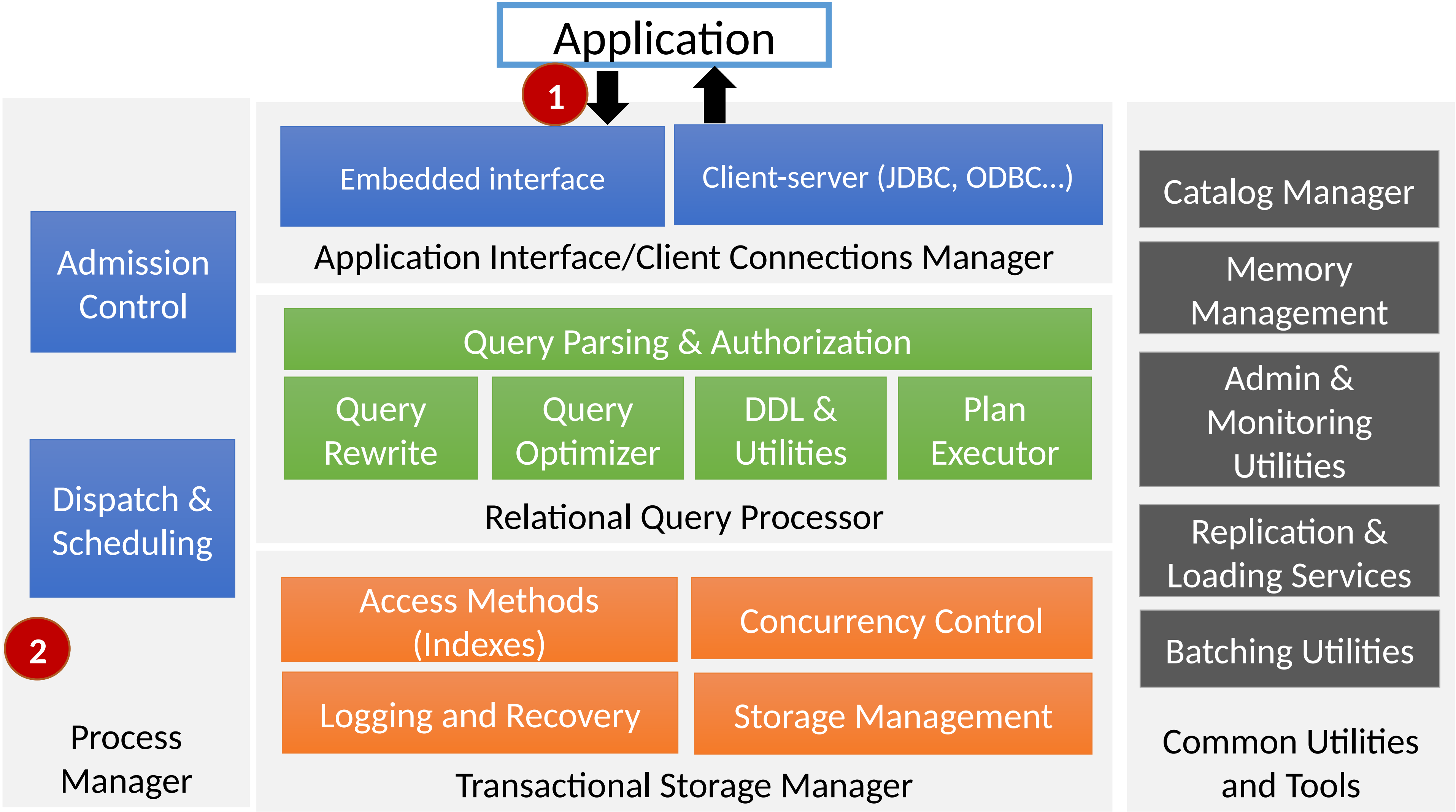
City	Population
State College	41,992
Pittsburg	302,407
Philadelphia	1,581,000

How does the query reach DBMS?

- Depends on architecture:

1. Embedded or
2. Client-server

DBMS Components

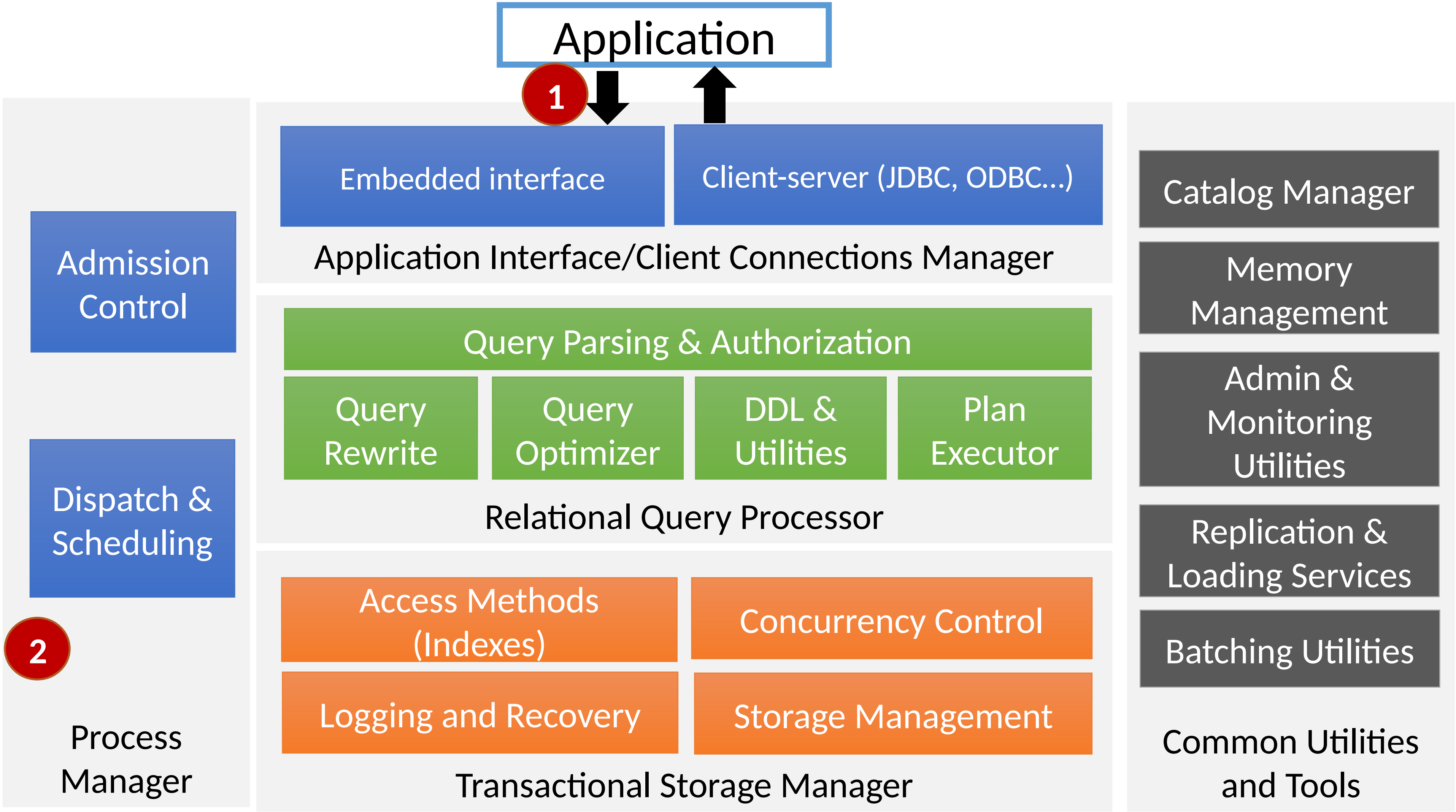


Application-DBMS Interface

Two Major Alternatives:

- Embedded
  - Application uses DBMS functionality via **direct API function calls**
  - Examples: SQLite, RocksDB, etc.
- Tiered client-server
  - Application establishes a connection with the DBMS's client communications manager, **via JDBC/ODBC/etc**.
  - Connect to some server(s) that connect to the database server
  - Examples: MySQL, Postgres, Oracle, etc.

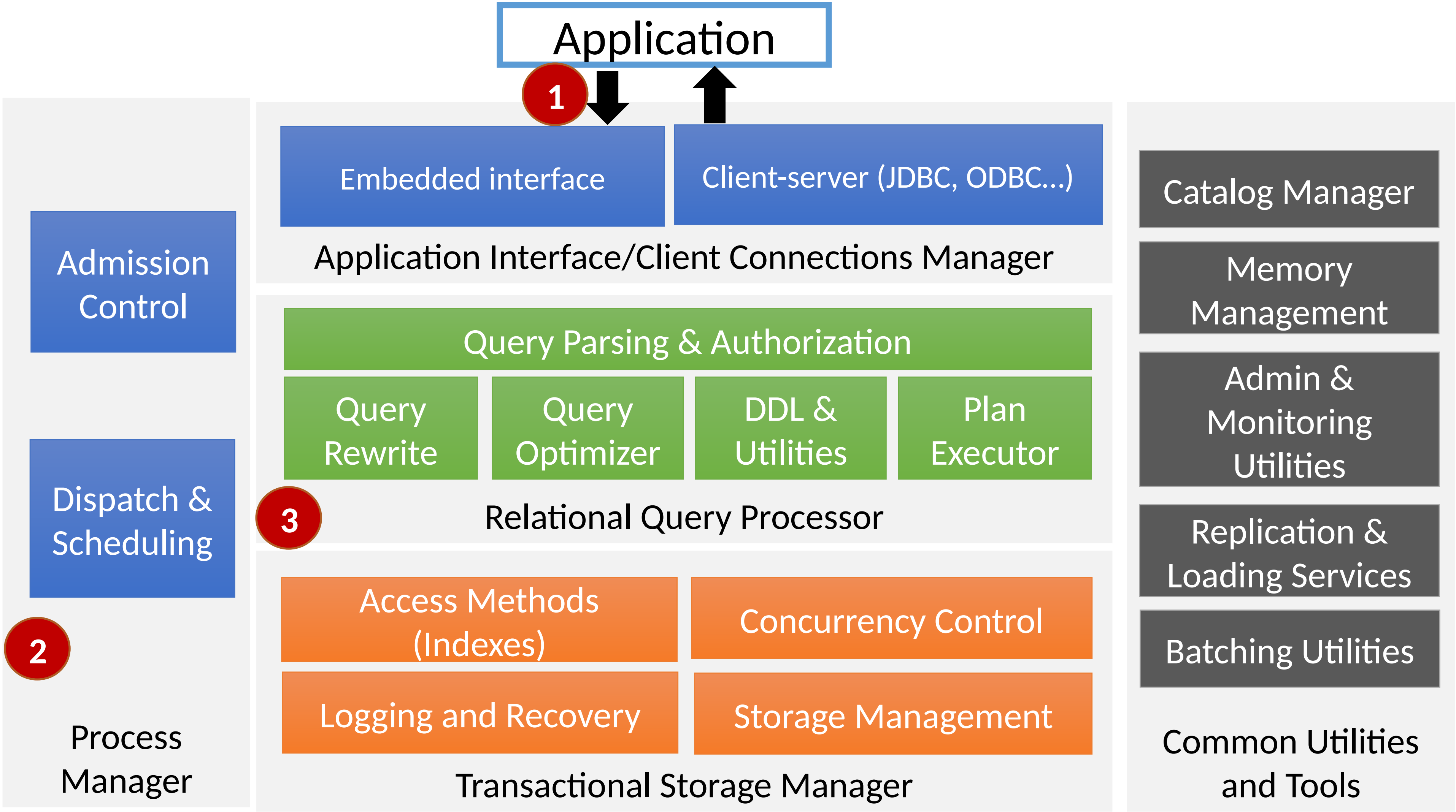
DBMS Components



Process Manager

- Queries (usually in the form of SQL commands) arrive through application-DBMS the connection/API calls
- DBMS must assign a **thread of computation** to handle the commands/transactions
  - I.e., a thread that runs on a CPU core
- Admission control
  - The DBMS and server have a limited amount of resources
  - Must not accept new work unless there is enough resource to handle it to avoid thrashing, allow graceful degradation
    - At peak utilization, throughput remains high, latency increases
  - Done in the DBMS or another tier (e.g., app server)

DBMS Components



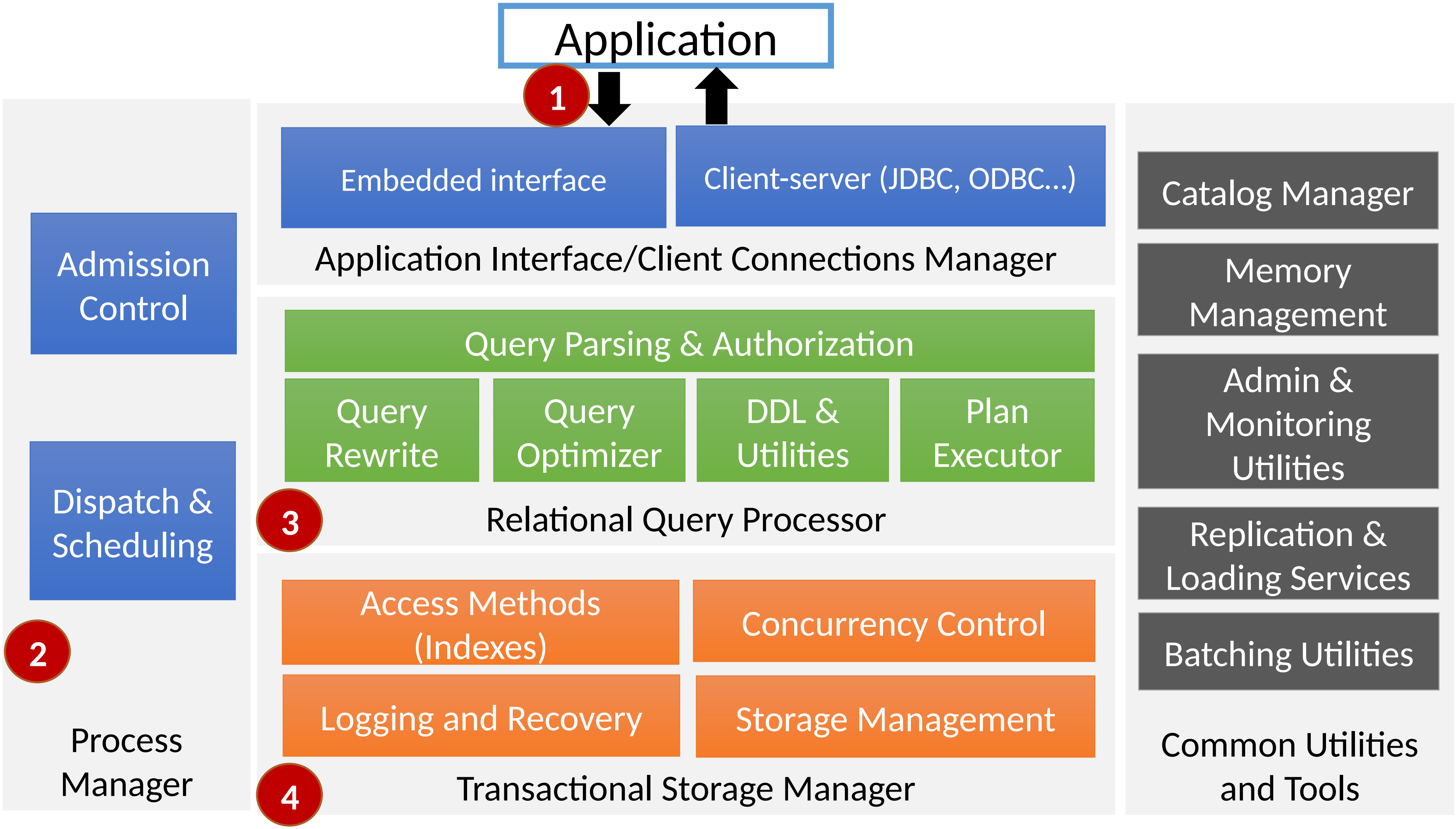
(Relational) Query Processor

- Check the user is authorized to run the query
- **Planner**: Generate a **query plan**
- Plan executor executes the query plan

- **Optimizer**: Choose the best (logical/physical) plan to execute.

- **Executor**: Consists of a series of general-purpose “operators” that can be used to execute any query, e.g., join, sort, select
- Operators will call into the transactional storage manager for the actual data access

DBMS Components



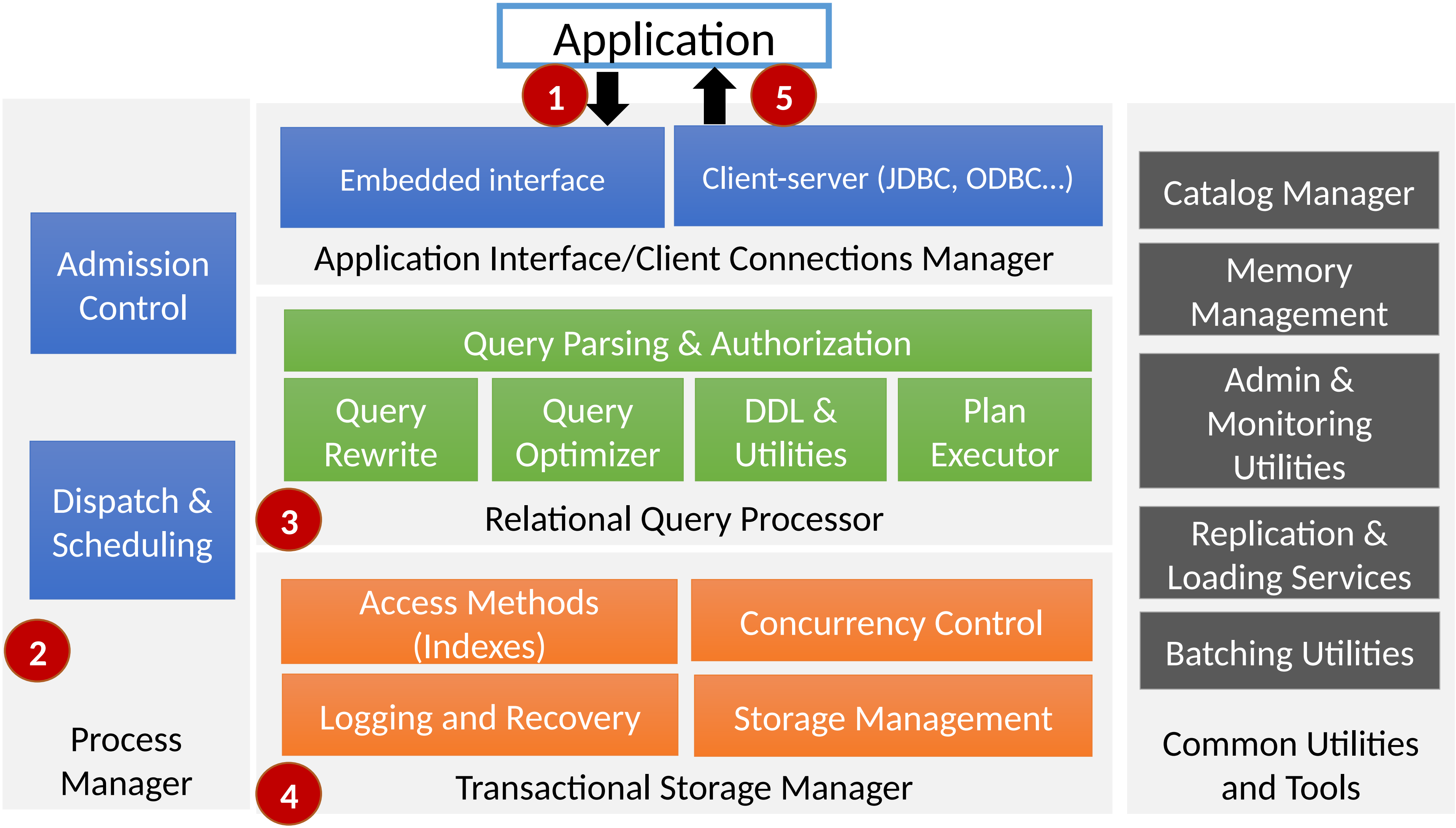
Transactional Storage Manager

- Manage all data accesses and manipulations
- Efficient ways to **read** and **modify** data
  - Indexes, tables
  - Buffering and judicious use of storage devices
- Guarantee data integrity
  - Guard against data corruption upon failures and crashes
  - Coordinate concurrent accesses from multiple threads

Key component for OLTP

We will spend a fair amount of time on this.

DBMS Components



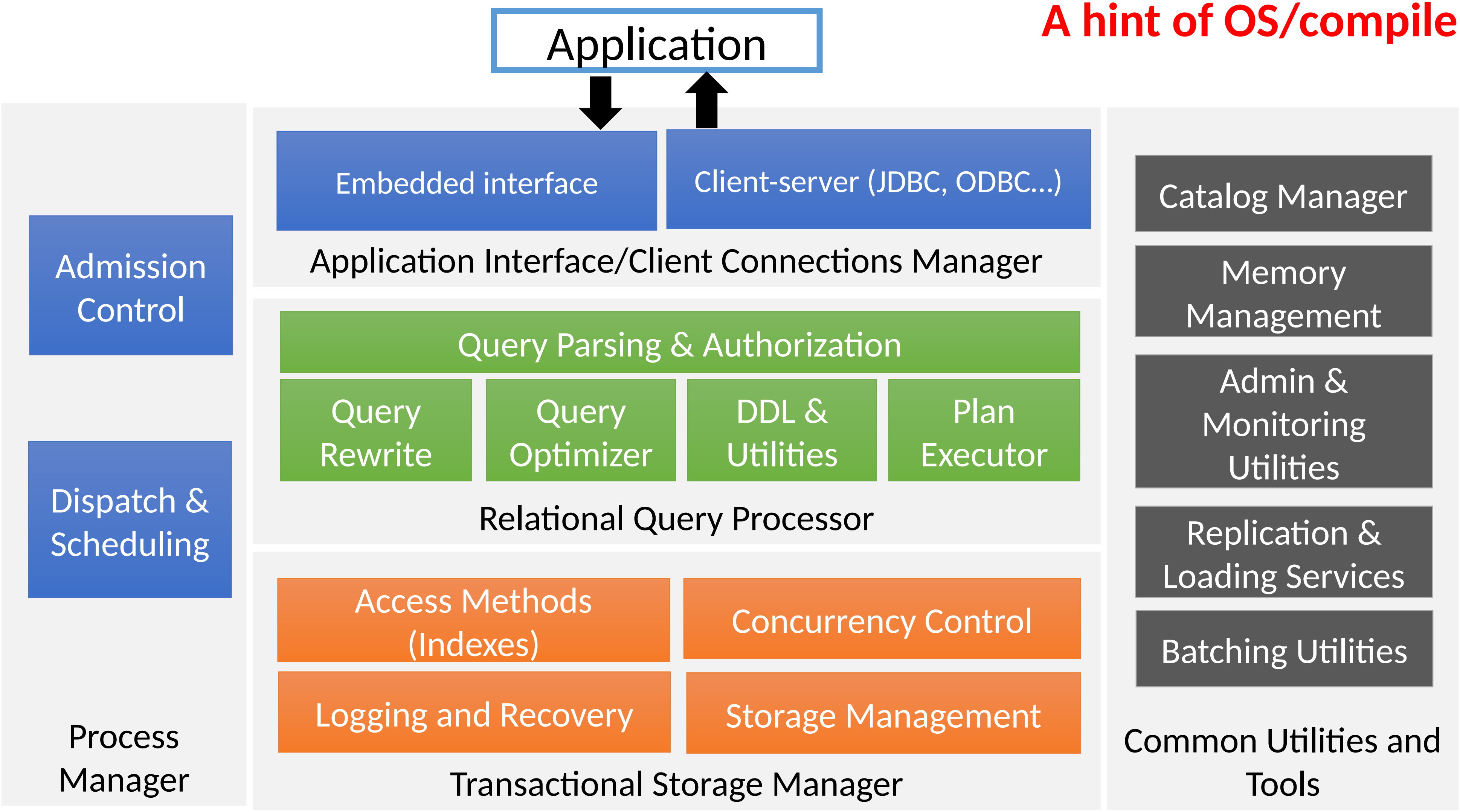
Last Step: Results back to Application

- “Unwinding” the stack of previous activities

- Access method returns control to Query Processor
- Query Processor generates result data records and place them in a buffer for Connection Manager/API Interface
- Connection manager/API Interface ships results back to the caller

DBMS Components

A hint of OS/compiler?



Takeaways…

- DBMS maps conceptual and physical views
  - Applications use the relational mode
  - DBMS uses its own physical representation
  - Optimizations can happen in the physical layer without affecting conceptual, application semantics
- Building a DBMS requires a full-stack effort
  - Understanding and utilizing hardware features
    - Sometimes (very rare) propose our own, new hardware
  - Know low-level software primitives well (features, traps, bugs…)
    - Sometimes propose our own, new ones
  - Designing suitable DBMS components