David Eccles

INFO90002 Database Systems & Information Modelling

Week 8

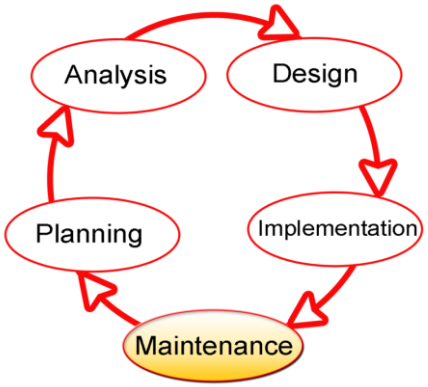
Database Administration

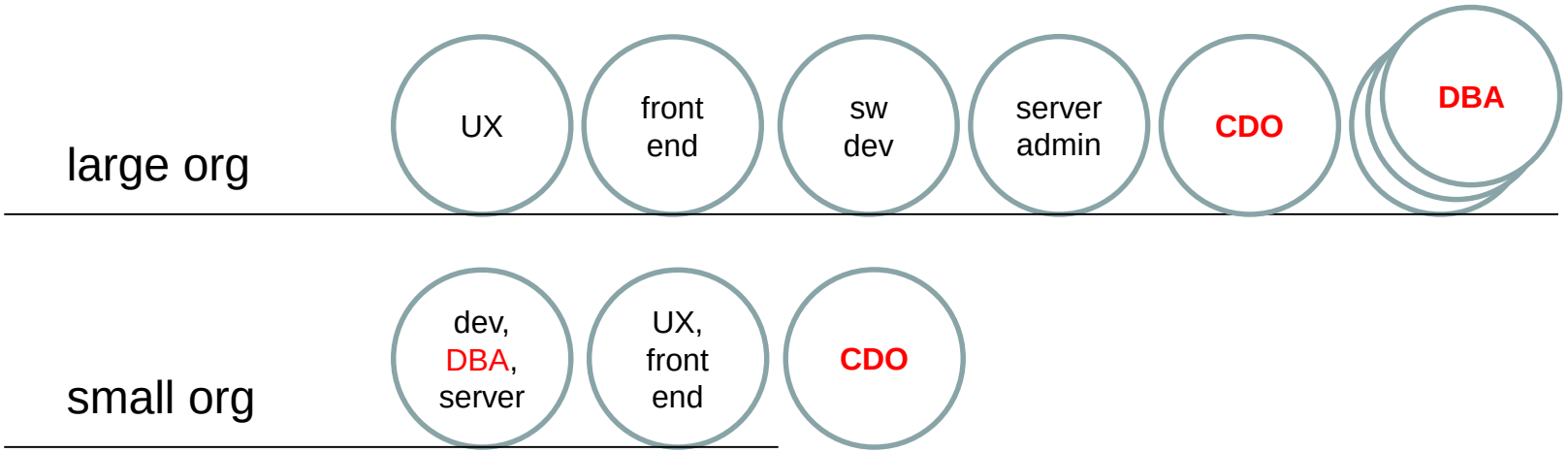
Today's Session…

- The Database 'Administrator' role
- The Data Manager role
- Architecture – Understanding the DBMS
  - concepts
- Performance improvement
  - concepts
  - common approaches e.g. indexes

The DBA role

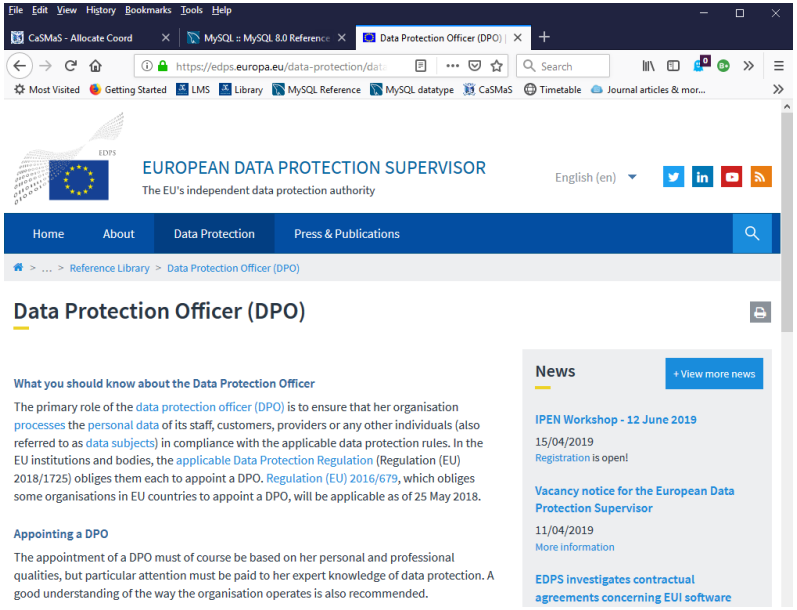
- primarily concerned with “maintenance” / “ops” phase
- but should be consulted during all phases of development
- “Database Administrator” or “DBA”
- often framed as a “job” or a “person”
- Large companies – many DBA’s
- Small company developer is the DBA
- DBA role can be made redundant by Cloud-based DBMS or “database as a service” DAAS (often IAAS or PAAS)





Data and Database Administration

- Data Administrator (CDO / DPO)
  - data policies, procedures and standards
  - planning
  - data conflict resolution
  - managing data
  - internal marketing & education
  - compliance
- Database Administrator (technical role)
  - analyze and design DB
  - select DBMS / tools / vendor
  - install and upgrade DBMS
  - tune DBMS performance
  - manage security, privacy, integrity
  - backup and recovery



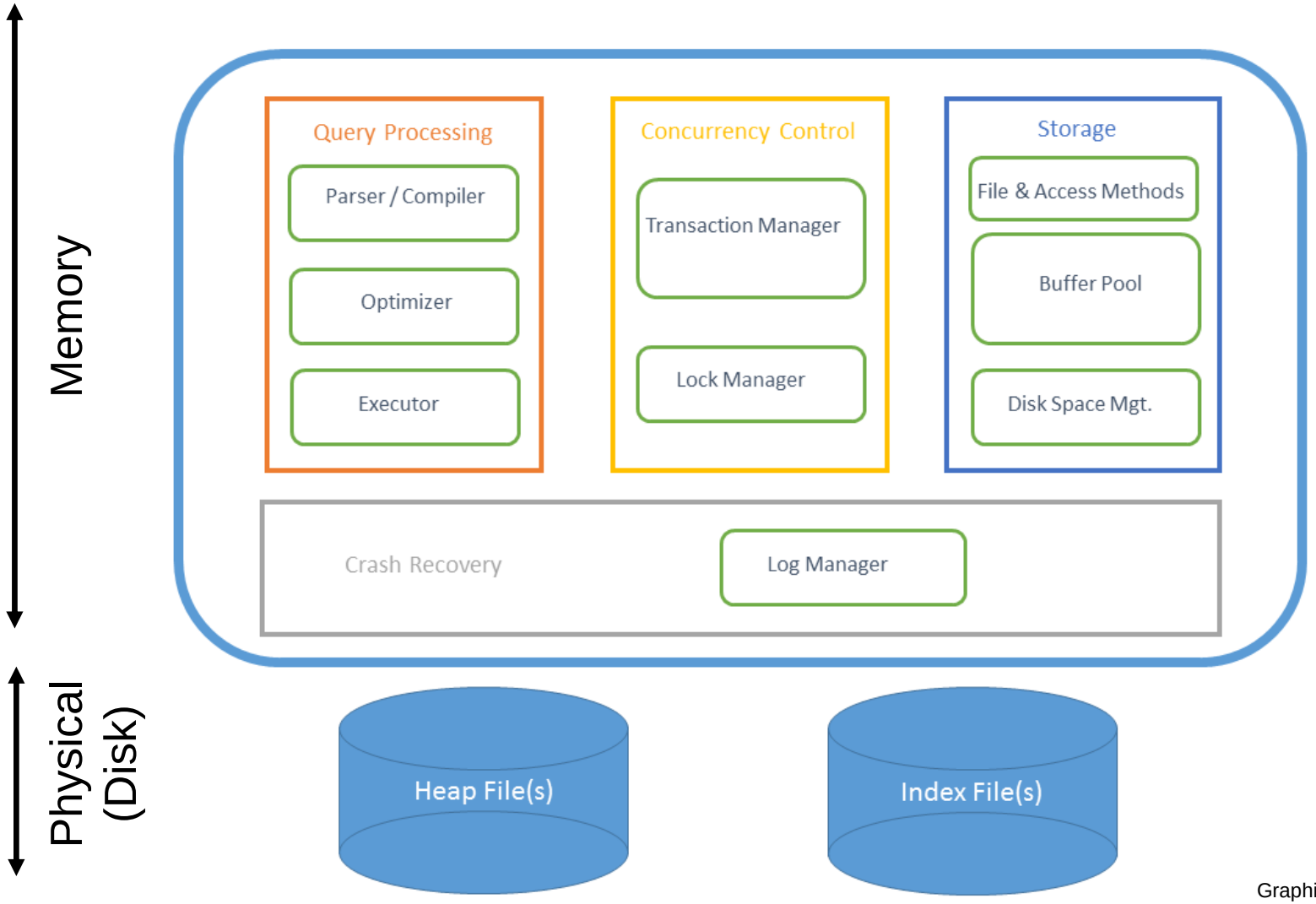
(Hoffer et al., chapter 11)

Architecture of a Database Management System (DBMS)

Database Systems Architecture

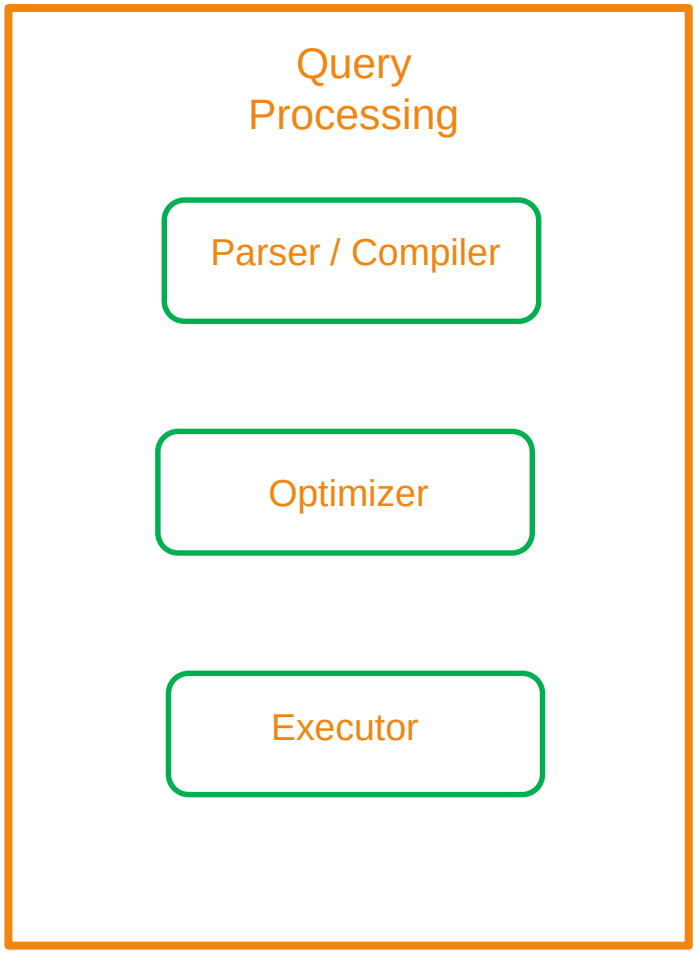
- A Database Management System (DBMS)
- Exists as one entity in two places
  - In Memory
  - Physically on disk
- Both places manage
  - Data (the reason we have the DBMS)
  - Performance (how it performs as it is used & grows)
  - Concurrency (manage high volumes of users)
  - Availability (assist in recovery and availability)
- One place is persistent the other transient
  - Disk representation is always present
  - Memory – transient – only exists when DBMS is running

DBMS Overview



Graphic Courtesy of
Dr Renata Borovica-Gajic

Query Processing



Parsing

- Syntax is correct & can “compile”
- DBMS User Permissions
- Resources (Data, Code, be able to Record Changes/Retrieve results)

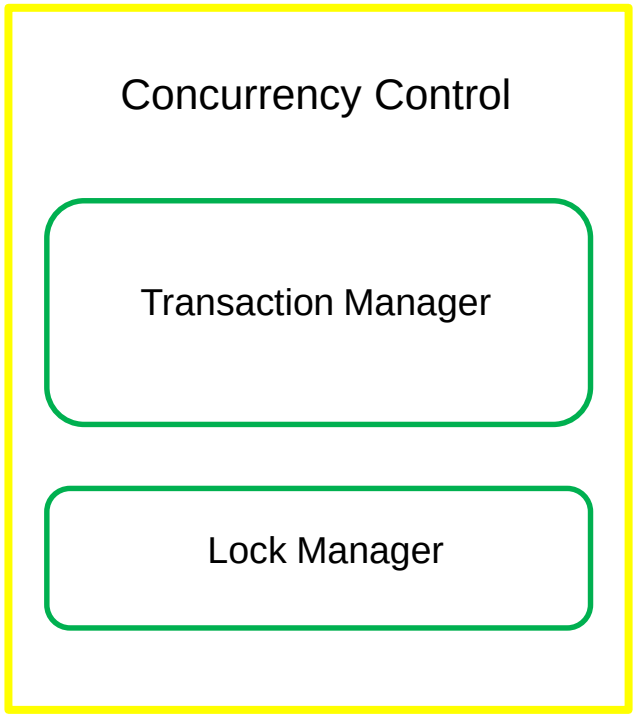
Optimizing

- Execution Plan & Execution Cost
- Evaluate indexes, table scans, hashing
- Eliminate worst, consider best options
- Lowest cost theoretically “best”

Execution

- Meet the ACID test,
- Atomic: All rows succeed or all fail
- Ensure resources are available
  - Data, Log changes, Memory, *Cursor* to do the work for the USER

Concurrency Control

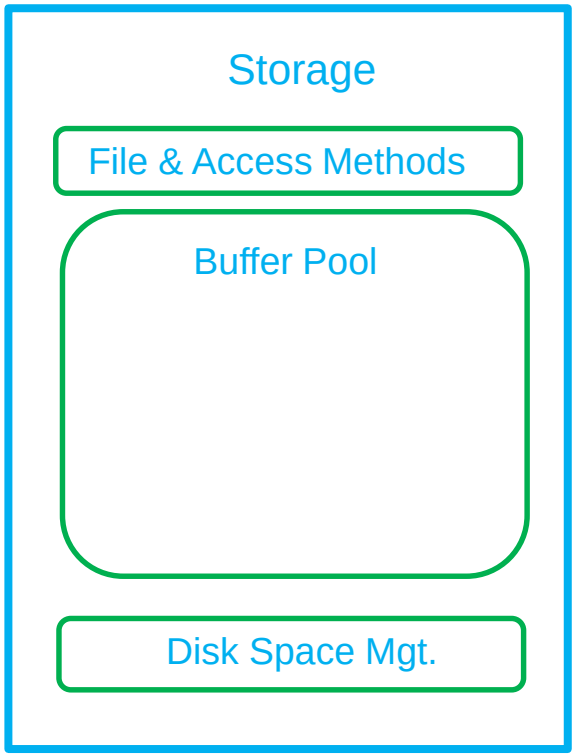


- Manages the work of the DBMS.
- Transaction Manager handles all aspects of the SQL transaction - which DBMS user wants WHAT resource
- Lock Manager is a list of what resources are locked and by which user at what level
- not only tables, indexes
  - buffers, cursor, memory addresses of resources

Concurrency Control

- Essential to manage large scalable DBMS
- Enables 1,000,000s of concurrent users
- Like a Traffic Policemen controlling the flow of traffic
  - Who can do what (allowed to do what they need to do)
  - Who has to wait (queue)
  - Who can travel through the intersection concurrently
    - Usually *readers* of data

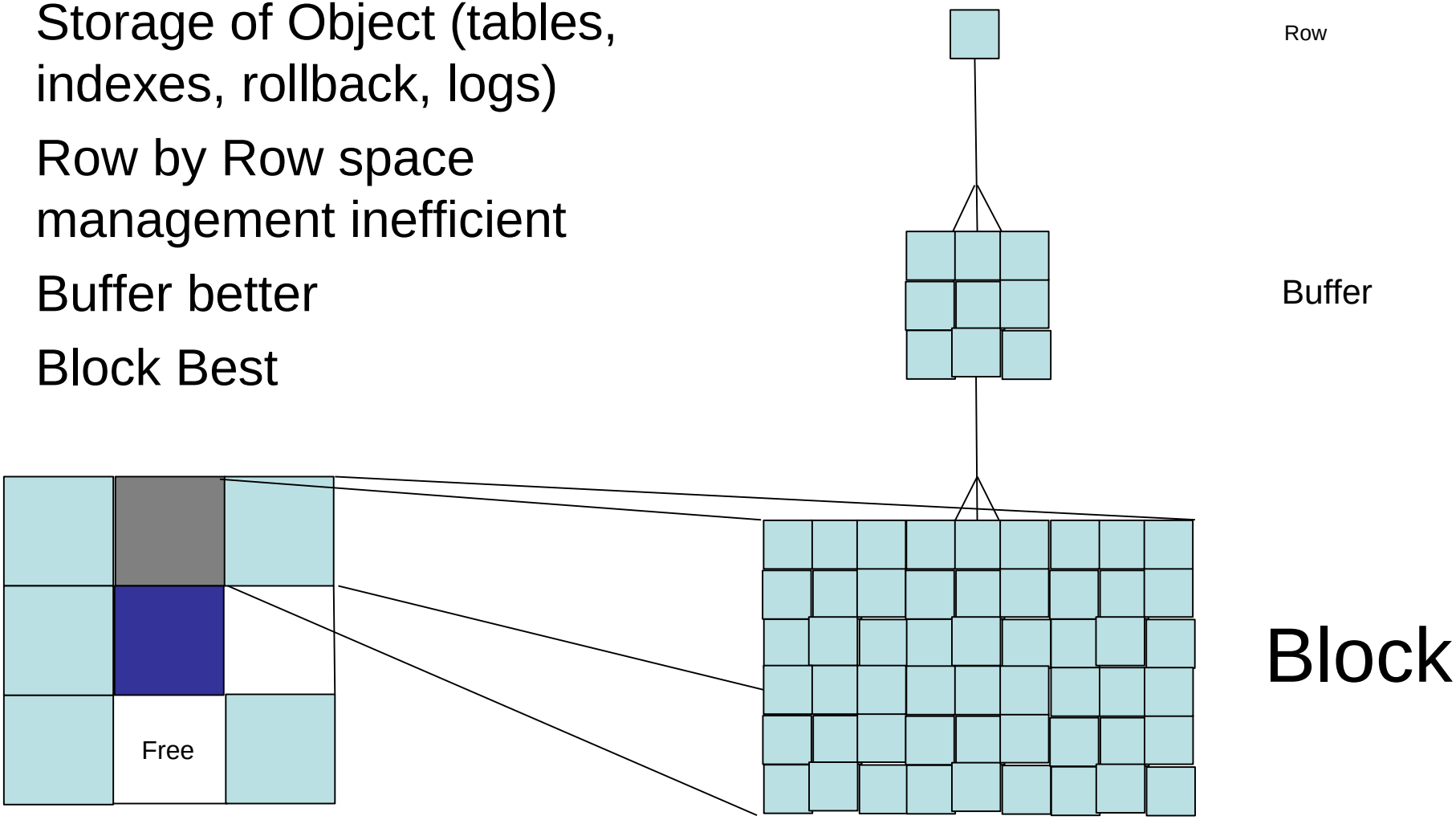
Storage



- File & Access Methods
  - Disk to Memory to Disk
  - Read a buffer or a block of buffers
- Buffer Pool
  - Data in memory
    - Row data
    - Index data
  - Organised
- Disk Space Management
  - How to organise growth of data on disk efficiently by writing efficiently.

Storage – Disk Space Management

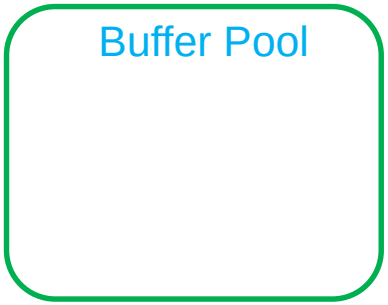
- Hierarchical Structure
- Storage of Object (tables, indexes, rollback, logs)
- Row by Row space management inefficient
- Buffer better
- Block Best



Buffer Pool

- Many Object Types (Tables, Indexes, Undo)
- Each buffer contains rows, b+tree leaf etc.
- Each buffer can have one of four status types:
  - Current
    - In use current committed version of data (row)
  - Active
    - Most recent change (may not be committed)

  COMMIT (Current: DepartmentID=9)
  - Stale
    - An old version of the data
  - Aged
    - Old and about to be removed from buffer pool



Log Manager

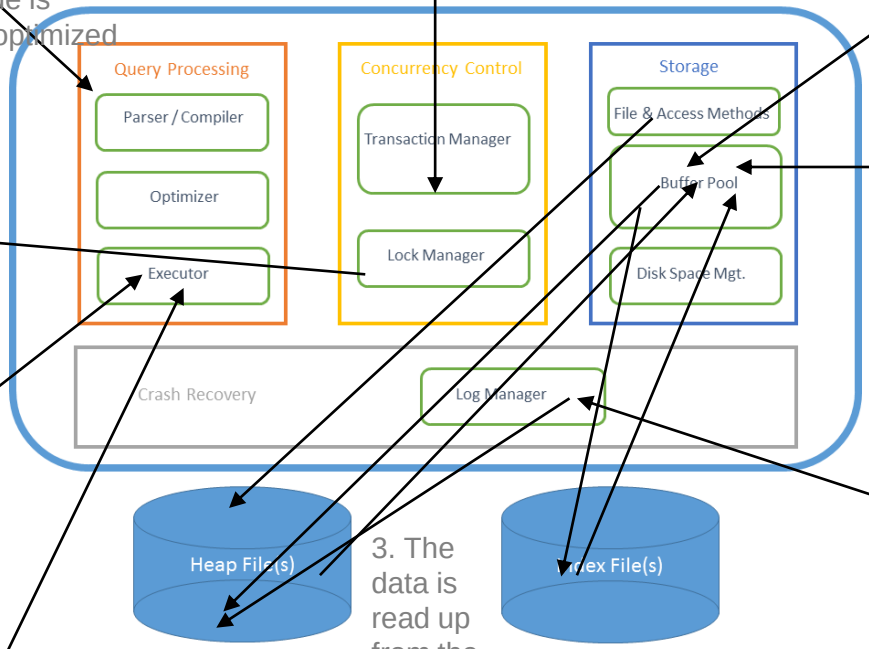
- Recovery
- Log Manager records ALL changes
  - Statement
  - Transaction
    - Statement
    - Rollback values
    - Before and After values
    - Timestamp begin
      - transaction, savepoint & commit timestamps
  - Database
    - Data Dictionary Changes

```
UPDATE employee
SET departmentid = 9
WHERE firstname = 'Nancy'
AND departmentid = 8;
```

1. The code is parsed & optimized

4. The Concurrency manager ensures all the tables and index data is in the buffer pool and ready to be used in the execution of the SQL.

6a In the buffer pool we record the before value the after value and the change to the value



2. We check if the table & index data (rows) are in the buffer pool. If it is we skip to 4. If not we go to 3.

5a When the rows & indexes are locked we execute the query

5b. The SQL is executed

3. The data is read up from the table & index on disk

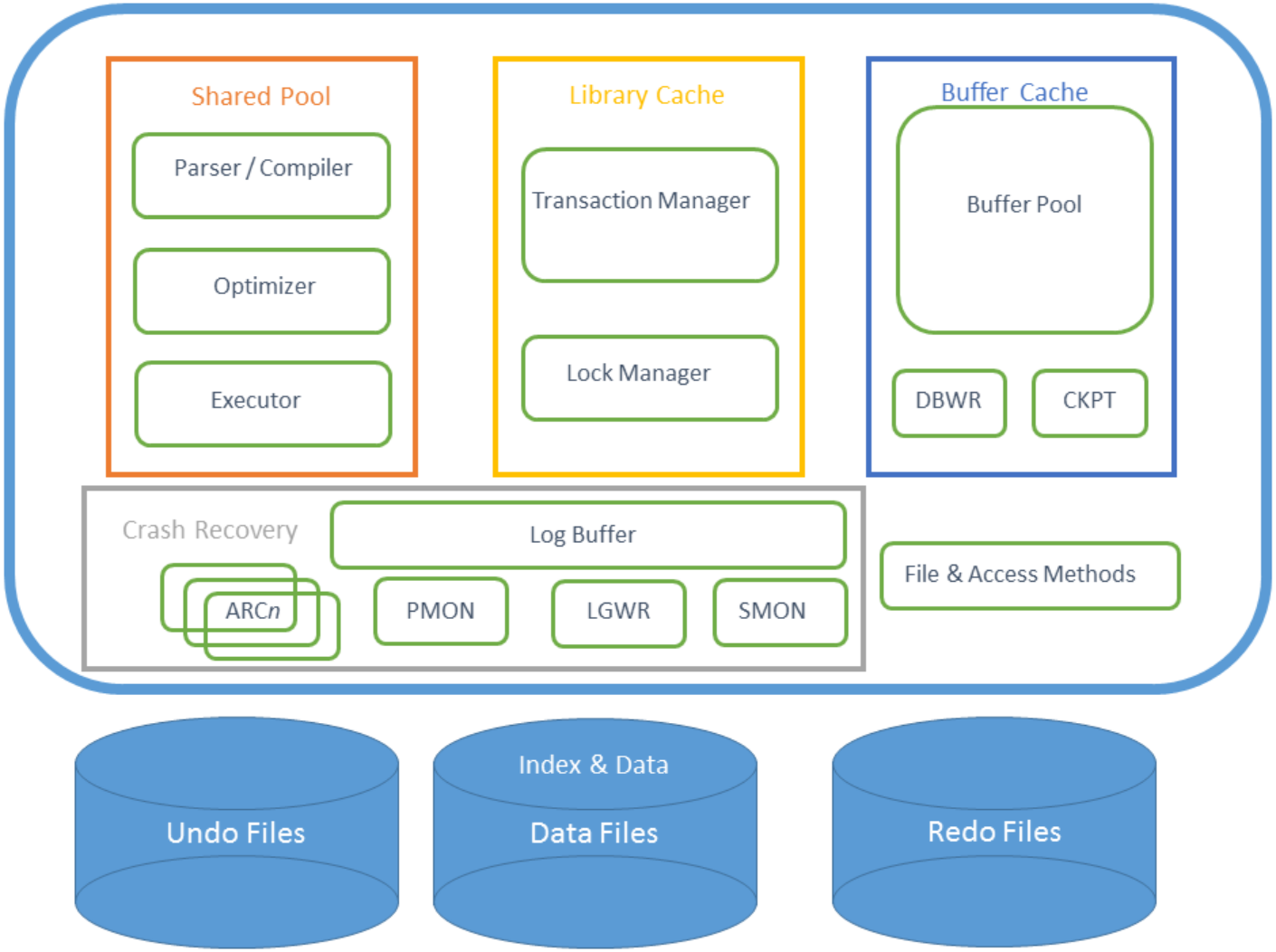
6b We record a timestamp and the before value, the after value and the change to the value for all transactions in the Log Manager

7. The user commits the change to make it permanent

8. The changed data and transaction log are written to disk

Graphic Courtesy of
Dr Renata Borovica-Gajic

How Oracle* DBMS looks



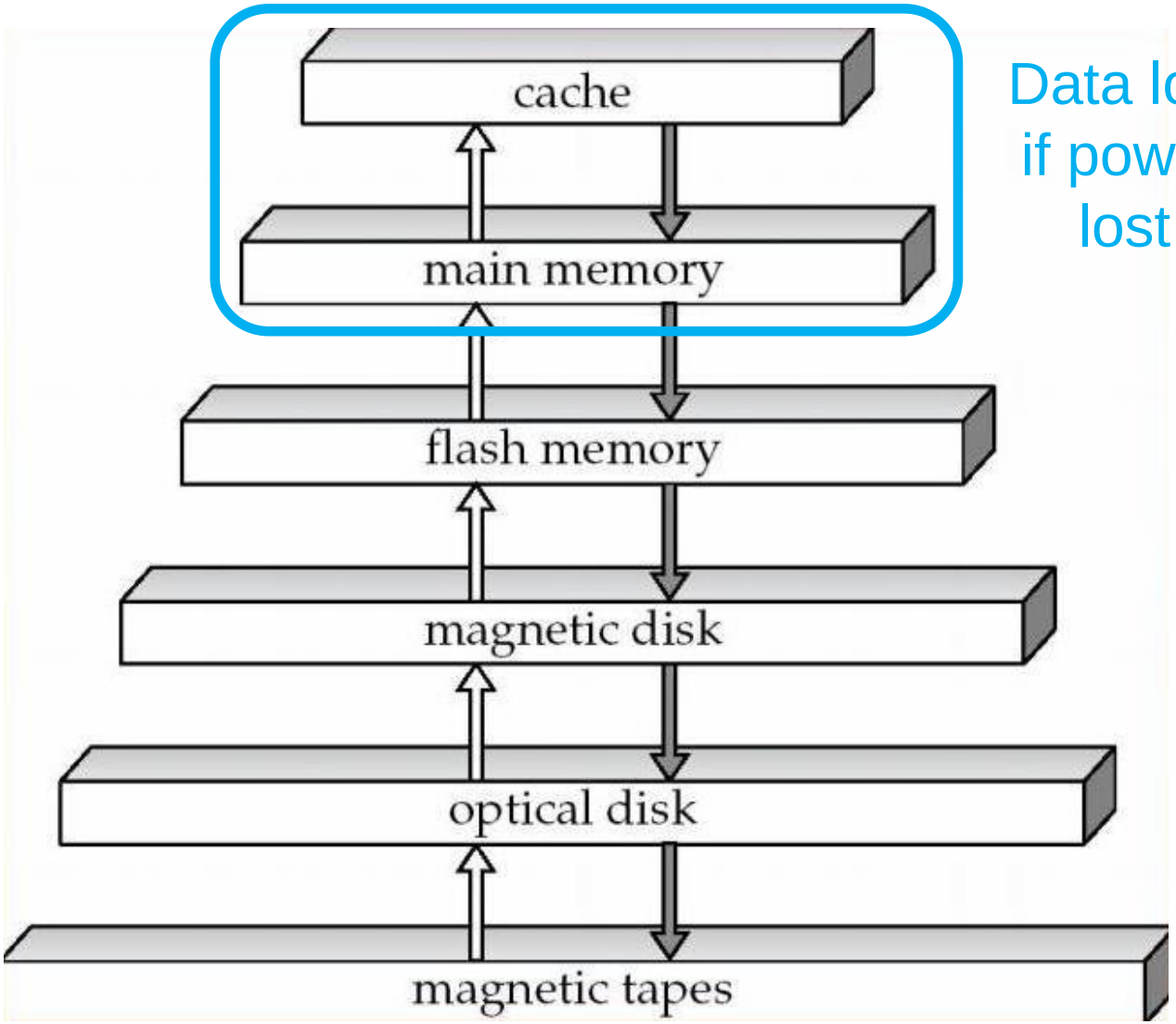
* This slide is not examinable

Database Performance

Storage media hierarchy

faster, more expensive, smaller capacity

slower, cheaper, *older*, bigger capacity



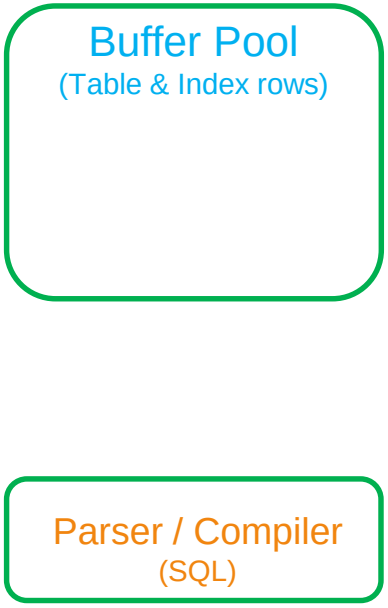
Data lost if power lost

What affects database performance?

- Caching data in memory, e.g. data buffers
- Placement of data files across disc drives
- Fast storage such as SSD
- Database replication and server clustering
- Use of indexes to speed up searches and joins
- Good choice of data types (especially PKs)
- Good program logic (no long running CRUD)
- Good query execution plans
- Good code (no deadlocks)

Caching Data in Memory

- Data and Code found in memory
- Avoids a read
- Reads are expensive
- Goal in to minimize reads (& writes)
  - Writes are necessary (recovery logs, changed data)
- “in memory databases”
  - all code all data loaded into memory on db start & stays until shutdown

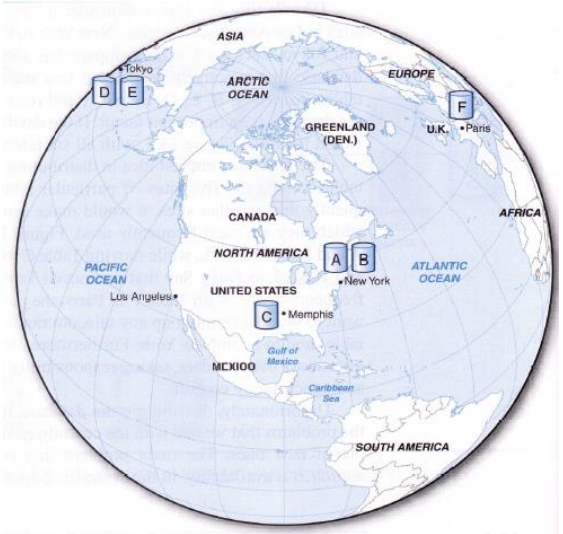


Data file location & Fast Disks (SSD)

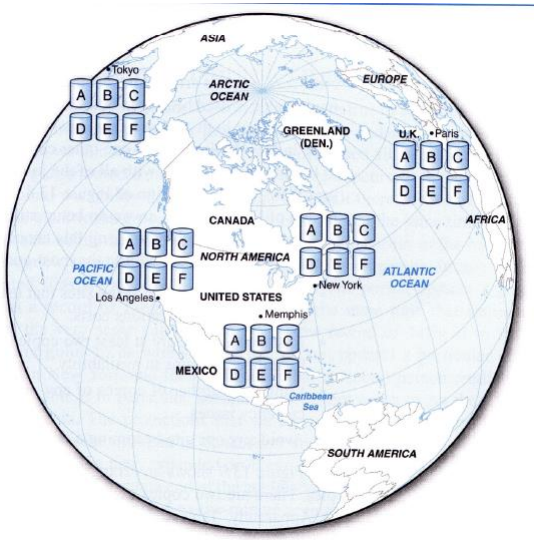
- Spread the files across the physical server
  - RAID (0, 0 +1, 5)
- We can't avoid writes
  - Spread files across many disks
    - Avoid contention
      - (many users competing for same resource)
  - Recovery Logs (always writing)
    - faster disk
- SSD (Solid State Drives)
  - No moving parts – nothing to break down
  - Faster I/O (Input & Output compared to other disk types)

Distribution & Replication

- Distributed data
  - Spreads the load
  - Data kept only where it is needed
  - Less work per physical server – faster response times



- Replicated Data
  - Spreads Load
  - Less work per physical server – faster response times



When to create indexes

- for each table, choose the columns you will index:
- column is *queried frequently* (used in Where clauses)
- columns that are used for *joins (PK to FK)*
- primary *keys* (automatic in most DBMS)
- foreign *keys* (automatic in MySQL)
- unique columns (automatic in most DBMS)
- large tables only - small tables do not require indexes
  - if you frequently retrieve less than about 15% of the rows
- wide range of values (good for regular indexes).
- small range of values (good for bitmap/hash indexes).

covered already in week 6

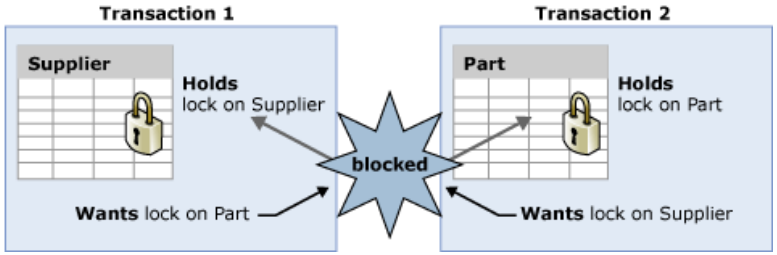
source: Oracle® Database Application Developer's Guide

Good Choices

- Good Data Types
  - INTEGERS for PK FK & PFK
- Good Program Logic & Code
  - Transaction design

```
BEGIN TRANSACTION
SELECT
UPDATE
UPDATE
UPDATE
COMMIT
```

  - Avoid Long Complex Transactions that never commit or savepoint
  - Avoid coding deadlocks!
  - Appropriate Locking strategy
    - Row b4 Buffer b4 Table b4 Database
    - Consider Lock Timeouts (if not automatic)



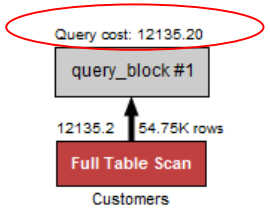
Deadlock Scenario

Good Execution Plans

- The best execution plan has the lowest “cost”
- Known as Cost Based Optimization (CBO)

```
SELECT cust_first_name, cust_last_name, cust_marital_status,
cust_city, cust_income_level
FROM Customers
WHERE cust_last_name = 'Parkburg'
AND cust_first_name = 'Peter'
AND cust_city = 'Trafford';
```

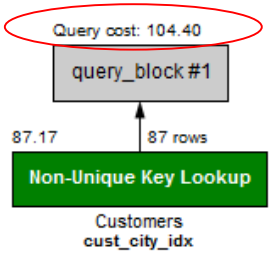
	cust_first_name	cust_last_name	cust_marital_status	cust_city	cust_income_level
	Peter	Parkburg	single	Trafford	H: 150.000 - 169.999



Index on where condition (cust_city) improves cost:

```
CREATE INDEX cust_city_idx
ON customers(cust_city);

SELECT cust_first_name, cust_last_name, cust_marital_status,
cust_city, cust_income_level
FROM Customers
WHERE cust_last_name = 'Parkburg'
AND cust_first_name = 'Peter'
AND cust_city = 'Trafford';
```



Performance Tuning

- Collect a baseline of performance statistics
  - (EG Long running update takes 43 seconds)
  - Retrieval of all rows in the largest table takes 14 seconds
  - Snapshot memory, usage at the same time every day every week.
- Initiate a SINGLE change
  - Allocate more memory to the database

  OR (NOT BOTH!)
  - Spread the files over different disks
- Retest your baseline metrics
  - Did performance improve or decline
- It is the change in the value not the absolute value that is important in Database Performance Tuning

David Eccles

INFO90002 Database Systems & Information Modelling

Week 8

Database Administration