# CSE 444: Database Internals

Lecture 3
DBMS Architecture



## Upcoming Deadlines

- Lab 1 Part 1 is due today
  - Go through logistics of getting started
  - Start to make some small changes to the code
- HW1 is due on Wednesday
  - Closely related to Lab 1
  - Helps you think about Lab 1 before implementing it… but don't wait until Wednesday to finish Lab 1!!!
- 544M first reading assignment due on Monday
- Lab 1 is due next Friday
  - A lot more work than part 1



## Late Days

- 4 late days total – At most 2 per lab or homework
- Can use in 24 hour chunks at any time
- NO OTHER EXTENSIONS!



## What we already know…

- A DBMS helps companies, organizations, and individuals to manage their data
- By providing capabilities to easily
  - Describe the data (database schema)
  - Load the data
  - Query the data
  - Update the data
  - Etc.
- For same reason, DBMS simplifies development of applications that need to operate on data



## What we already know…

- In 344, we learned about different data models: relational and semi-structured (XML)
- Relational model was proposed in 1970
- Most commonly used model today
- We reviewed the relational model in lecture 2



## Benefits of relational model

- Helps provide physical data independence
  - Can change data organization on disk for performance *without affecting applications*
  - Thanks to set-at-a-time query language
    - Relational algebra
- Helps provide logical data independence
  - Because data represented with simple structures
  - Can change schema *without affecting applications*
  - Thanks to views and simple data structure

Two important data mgmt princples

## Benefits of Declarative Query Languages

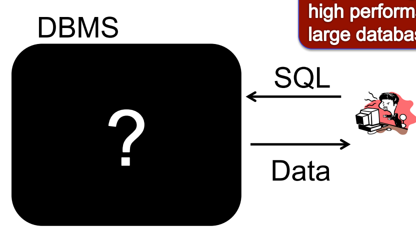
Another data mgmt princple

- User say what they want and not how to get it
  - Easier and faster for users
  - Need not be a programmer to write a SQL query
- The system can then worry about how to efficiently answer the queries
  - Enables *query optimization*



## How to Implement a Relational DBMS?

Key challenge: Achieve high performance on large databases!





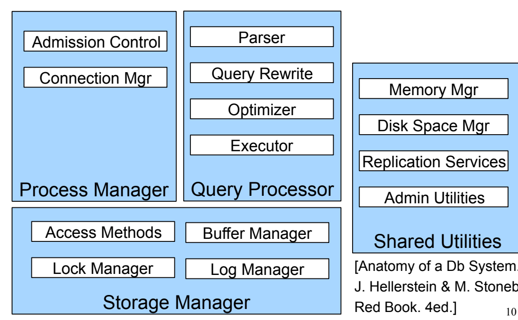

## Goal for Today

Overview of DBMS architecture

Overview of query execution



## DBMS Architecture




[Anatomy of a Db System. J. Hellerstein & M. Stonebraker. Red Book. 4ed.]



## Query Processor



## Example Database Schema

```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,price)
```

View: Suppliers in Seattle

```
CREATE VIEW NearbySupp AS
SELECT sno, sname
FROM Supplier
WHERE scity='Seattle' AND sstate='WA'
```

# Example Query

- Find the names of all suppliers in Seattle who supply part number 2

```
SELECT sname FROM NearbySupp
WHERE sno IN ( SELECT sno
               FROM Supplies
               WHERE pno = 2 )
```



# Query Processor

- **Step 1: Parser**
  - Parses query into an internal format
  - Performs various checks using catalog
    - Correctness, authorization, integrity constraints
    - Typically, catalog is stored in the form of set of relations
- **Step 2: Query rewrite**
  - View rewriting, flattening, etc.



# Rewritten Version of Our Query

Original query:

```
SELECT sname
FROM NearbySupp
WHERE sno IN ( SELECT sno
               FROM Supplies
               WHERE pno = 2 )
```

Rewritten query:

```
SELECT S.sname
FROM Supplier S, Supplies U
WHERE S.scity='Seattle' AND S.sstate='WA'
AND S.sno = U.sno
AND U.pno = 2;
```

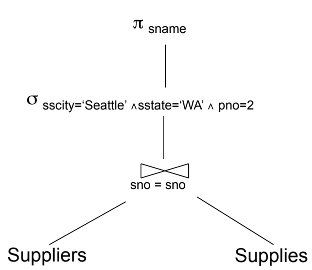


# Query Processor

- **Step 3: Optimizer**
  - Find an efficient query plan for executing the query
  - A **query plan** is
    - **Logical**: An extended relational algebra tree
    - **Physical**: With additional annotations at each node
      - Access method to use for each relation
      - Implementation to use for each relational operator
- **Step 4: Executor**
  - Actually executes the physical plan



# Logical Query Plan

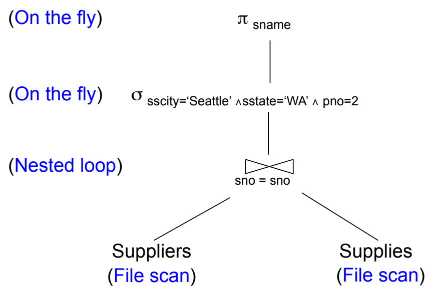





# Physical Query Plan

- Logical query plan with extra annotations
- **Access path selection** for each relation
  - Use a file scan or use an index
- **Implementation choice** for each operator
- **Scheduling decisions** for operators

## Physical Query Plan



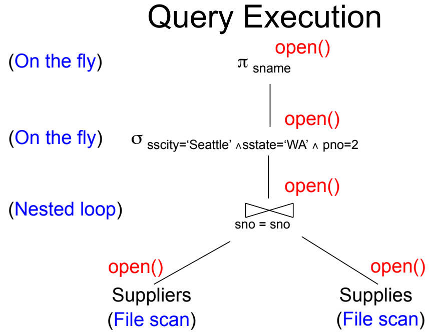



## Iterator Interface

- Each **operator implements this interface**
- **open()**
  - Initializes operator state
  - Sets parameters such as selection condition
- **next()**
  - Operator invokes next() recursively on its inputs
  - Performs processing and produces an output tuple
- **close()**: clean-up state

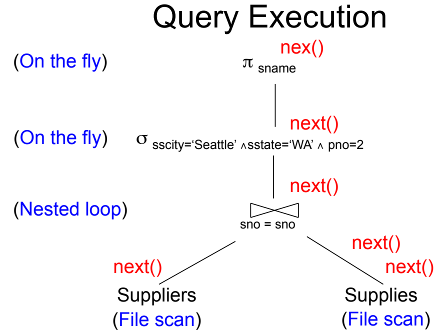


## Query Execution






## Query Execution






## Storage Manager



## Storage Manager

- **Buffer Manager**
  - Caches data in memory
  - Reduces the number of disk IO operations
  - Care is needed to support ACID transactions!
- **Access Methods**
  - Organize relation data on disk
  - Files ("heap files") and indexes
- **Log and Lock Managers**
  - Necessary to support transactions

# Process Manager



# Process Manager

- **Connection Manager**
  - Process per user or thread per user?
  - Various variants exist, partly for historical reasons
- **Admission Control**
  - To avoid thrashing
  - And provide “graceful degradation” under load
  - Second level of admission control: before running a query



# Shared Utilities



# Shared Utilities

- **Memory Manager**
  - Manages memory used by various components: internal operator state, query optimizer, etc.
  - Note: Buffer manager holds only *data*
- **Disk Space Manager**
  - Two basic deployment alternatives:
    - Use “raw” disk device interface directly
    - Use OS files
  - DB file abstraction on top of disk or OS file abstraction



# Shared Utilities

- **Replication Services**
  - For increased fault-tolerance
  - Or for increased performance
- **Admin Utilities**
  - Collecting statistics about data for optimizer
  - Re-organize data on disk, build indexes, etc.
  - Backup or export database