

# CSE 444: Database Internals

## Lecture 3 DBMS Architecture

CSE 444 – Winter 2018

1

## Announcements

- [Should be well on your way to finishing part 1](#)
- [Tuple, TupleDesc,](#)
- [Quiz 1 and 2 merge, 3 and 4 merge. Same work](#)
- Lab 1 part 1 due tonight at 11pm
  - Turn in using script in local repo: `./turnInLab.sh lab1-part1`
  - Remember to confirm that the tag has been applied in GitLab!
- HW1 is due on Friday at 11pm
  - Turn in by uploading to GitLab (will post instructions online) or submit a paper copy in class or office hours on the due date.
  - Helps you think about Lab 1 before implementing it... but don't wait until Wednesday to continue on Lab 1!!!
- 544M first reading assignment due on Friday
- Lab 1 is due next Wednesday (1/17) at 11pm

CSE 444 – Winter 2018

3

## 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!
- Try to save late days for later in the quarter
- But no late days for final lab

CSE 444 – Winter 2018

4

## What we already know...

- **Database** = collection of related files
- **DBMS** = program that manages the database

CSE 444 – Winter 2018

5

## What we already know...

- **Data models:** relational, semi-structured (XML), graph (RDF), key-value pairs
- **Relational model:** defines only the logical model, and does not define a physical storage of the data

CSE 444 – Winter 2018

6

## What we already know...

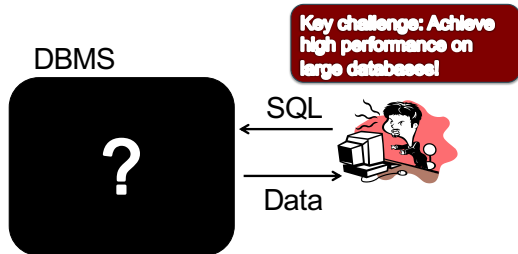
### Relational Query Language:

- **Set-at-a-time:** instead of tuple-at-a-time
- **Declarative:** user says what they want and not how to get it
- **Query optimizer:** from *what* to *how*

CSE 444 – Winter 2018

7

## How to Implement a Relational DBMS?



CSE 444 – Winter 2018

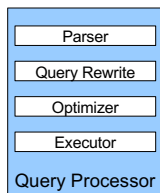
8

## DBMS Architecture

CSE 444 – Winter 2018

9

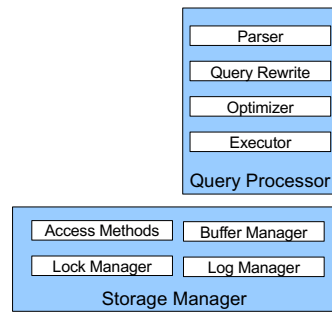
## DBMS Architecture



CSE 444 – Winter 2018

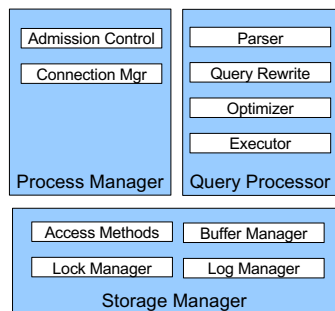
10

## DBMS Architecture



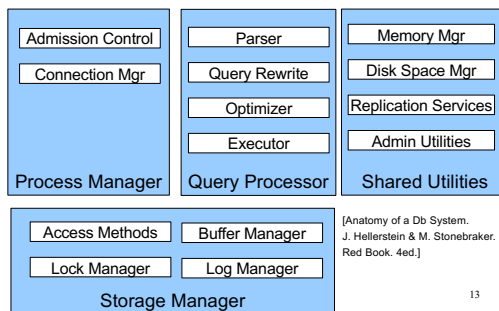
11

## DBMS Architecture



12

## DBMS Architecture



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

13

## Goal for Today

- Overview of query execution
- Overview of storage manager

## Query Processor

## Example Database Schema

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

### View: Suppliers in Seattle

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

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

## 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**
- Step 2: Query rewrite**
  - View rewriting, flattening, etc.

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

## Rewritten Version of Our Query

### Original query:

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

### Rewritten query (expanding NearbySupp view):

```
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;
```

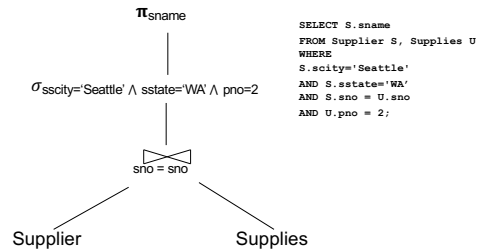
## Query Processor

- **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

CSE 444 – Winter 2018

20

## Logical Query Plan



CSE 444 – Winter 2018

21

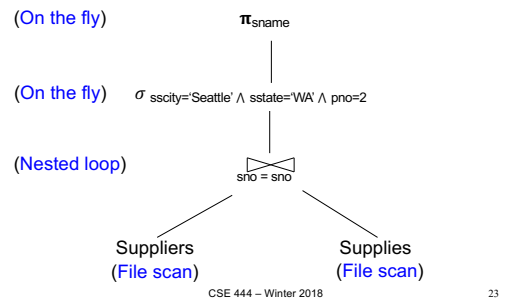
## Physical Query Plan

- 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

CSE 444 – Winter 2018

22

## Physical Query Plan



CSE 444 – Winter 2018

23

## Query Executor

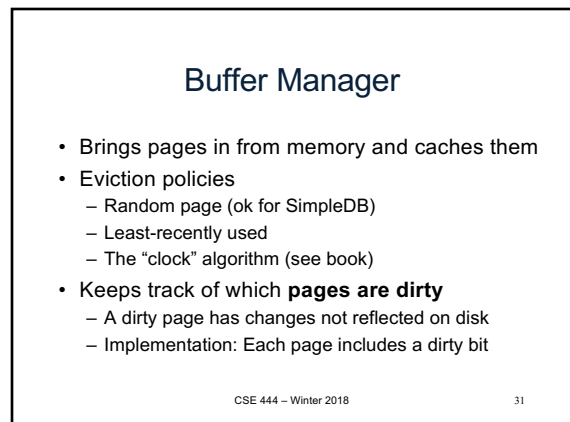
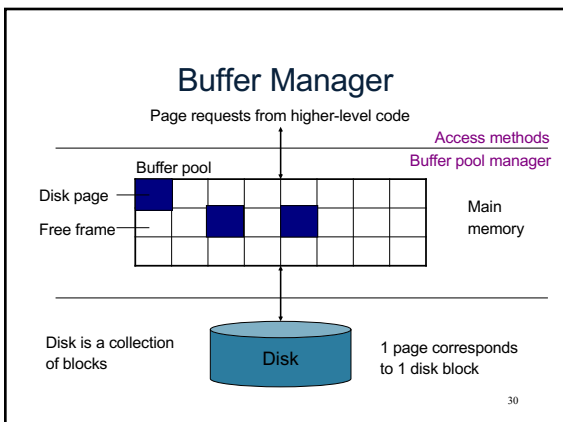
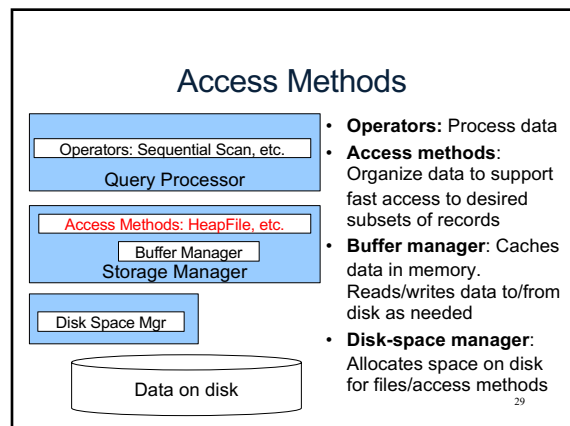
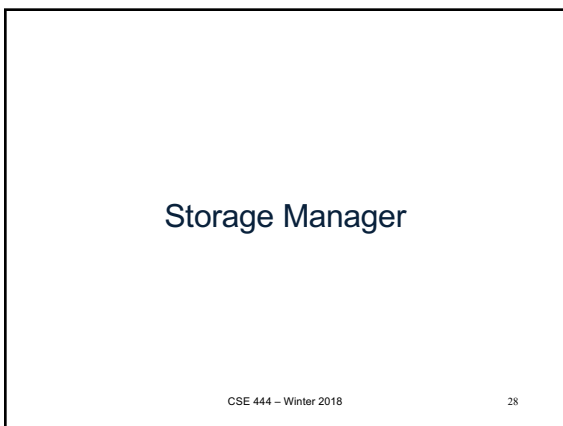
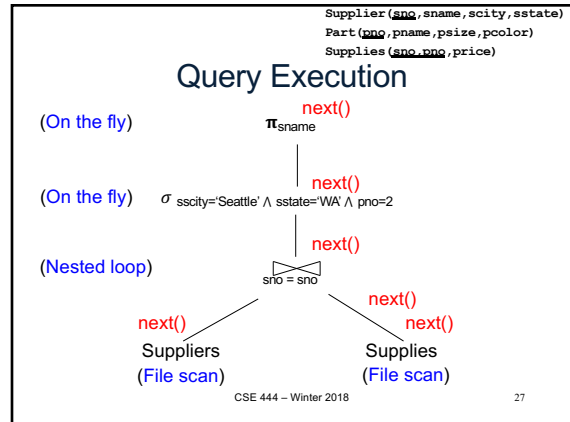
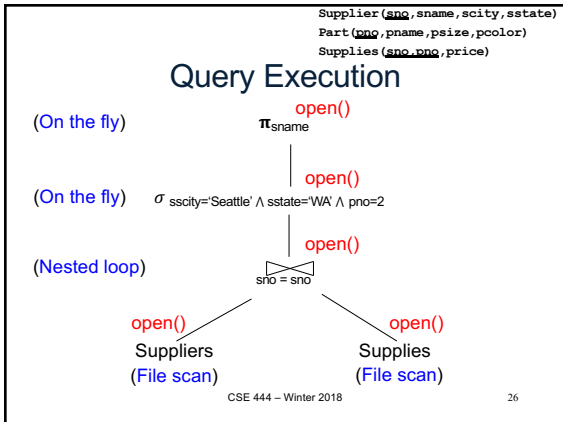
CSE 444 – Winter 2018

24

## Iterator Interface

- Each **operator implements this interface**
- **open()**
  - Initializes operator state
  - Sets parameters such as selection predicate
- **next()**
  - Operator invokes next() recursively on its inputs
  - Performs processing and produces an output tuple
- **close():** clean-up state
- Operators also have reference to their **child** operator in the query plan

25



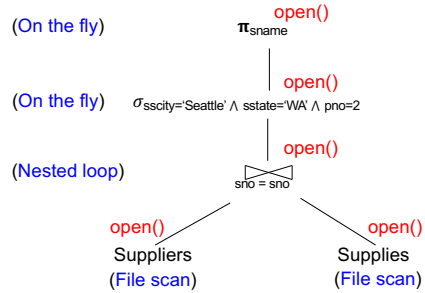
## Access Methods

- A DBMS stores data on disk by breaking it into *pages*
  - A page is the size of a disk block.
  - A page is the unit of disk IO
- Buffer manager caches these pages in memory
- Access methods do the following:
  - They organize pages into collections called *DB files*
  - They organize data inside pages
  - They provide an API for operators to access data in these files
- Discussion:
  - OS vs DBMS files
  - OS vs DBMS buffer manager

CSE 444 – Winter 2018

32

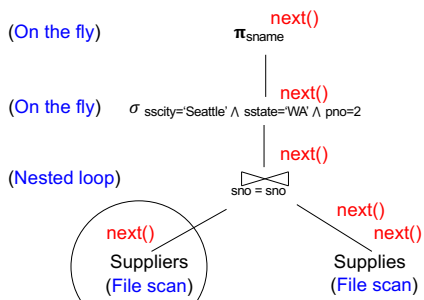
## Query Execution How it all Fits Together



CSE 444 – Winter 2018

33

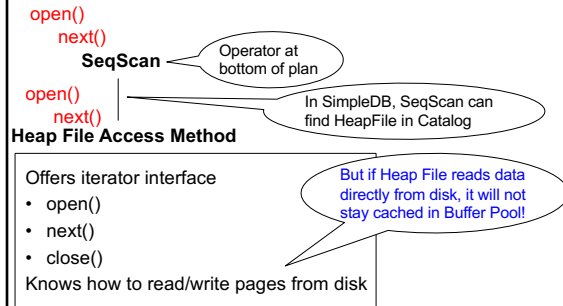
## Query Execution How it all Fits Together



CSE 444 – Winter 2018

34

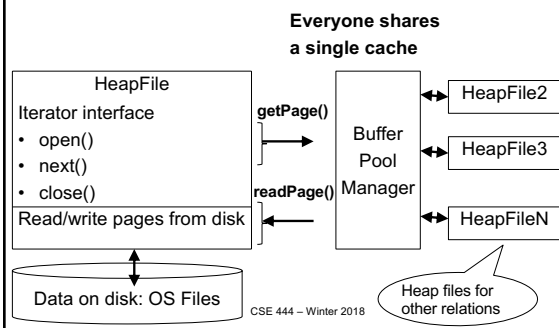
## Query Execution In SimpleDB



CSE 444 – Winter 2018

35

## Query Execution In SimpleDB



CSE 444 – Winter 2018

## HeapFile In SimpleDB

- Data is stored on disk in an OS file. HeapFile class knows how to “decode” its content
- Control flow:
  - SeqScan calls methods such as “iterate” on the HeapFile Access Method
  - During the iteration, the HeapFile object needs to call the BufferManager.getPage() method to ensure that necessary pages get loaded into memory.
  - The BufferManager will then call HeapFile.readPage()/writePage() page to actually read/write the page.

CSE 444 – Winter 2018

37