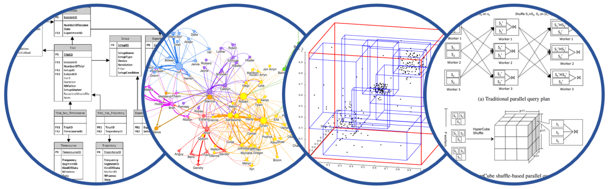


Database System Internals
Architecture

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

1

What we already know…

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

3

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

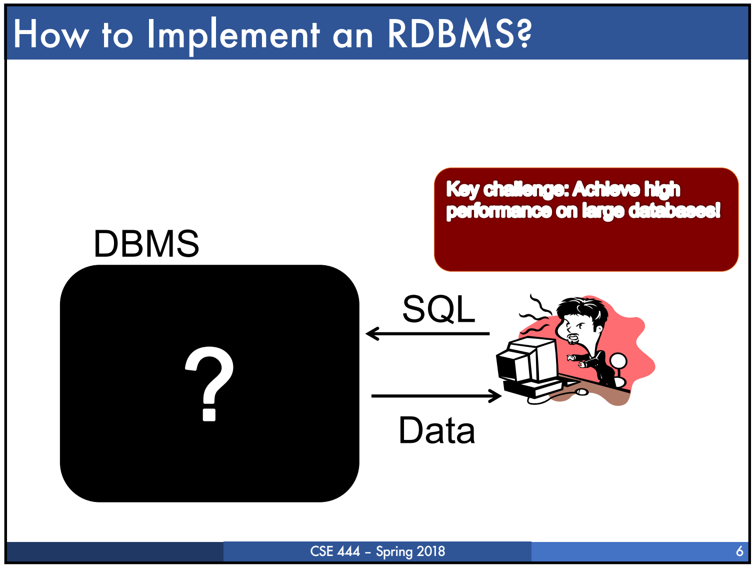
4

What we already know…

Relational Query Language:

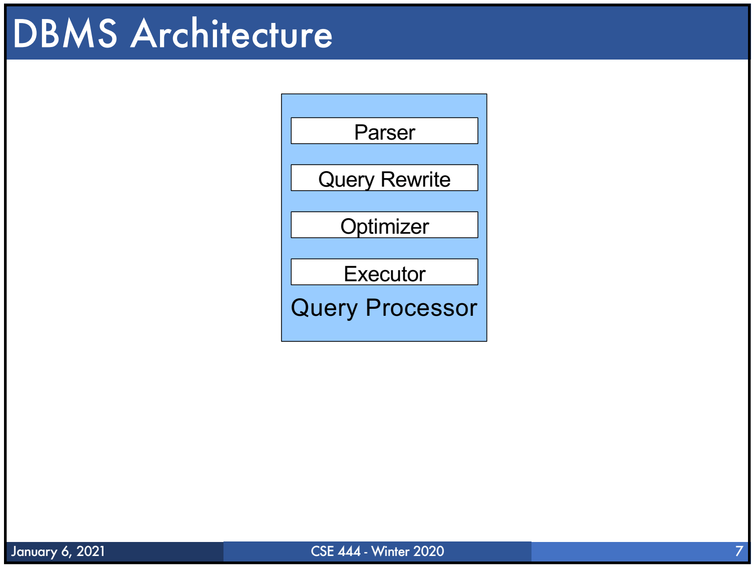
- Set-at-a-time: instead of tuple-at-a-time
- Declarative: user says what they want and not how to get it
- Query optimizer: from *what* to *how*

5



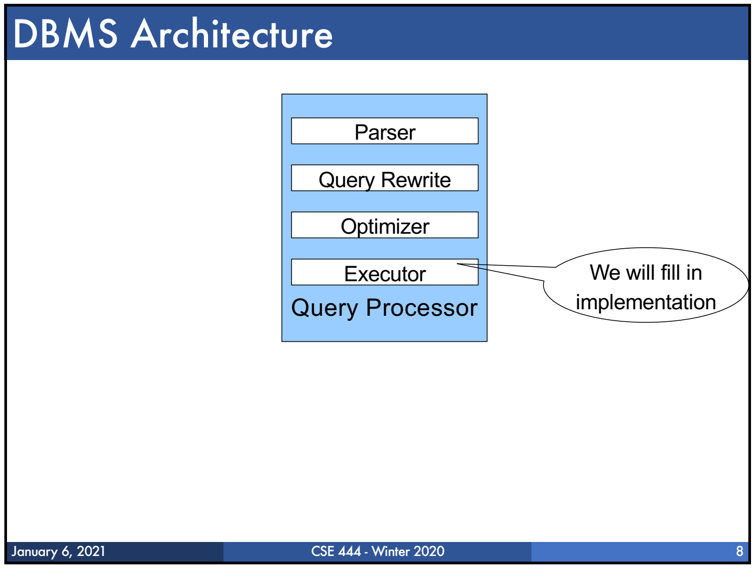
How to Implement an RDBMS?

6



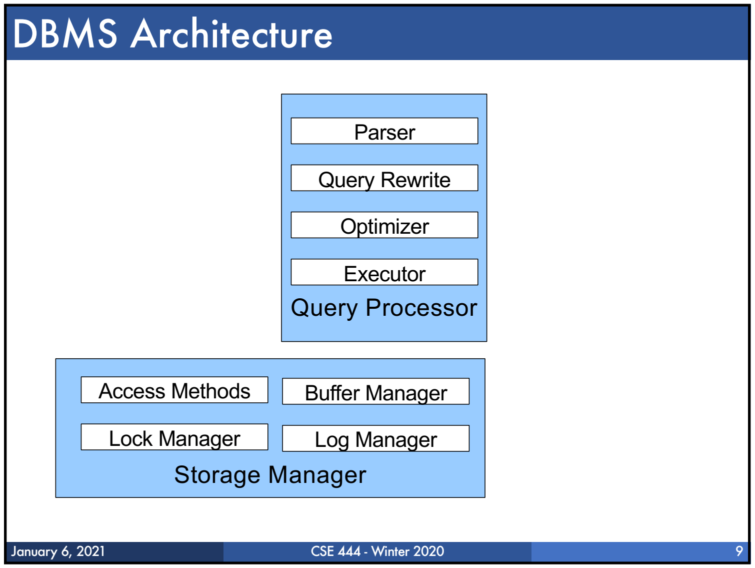
DBMS Architecture

7



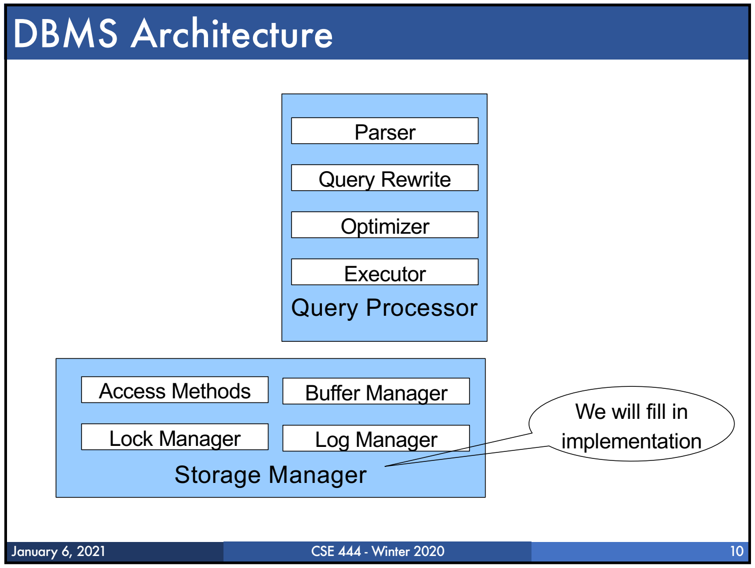
DBMS Architecture

8



DBMS Architecture

9



DBMS Architecture

Parser

Query Rewrite

Optimizer

Executor

Query Processor

Access Methods

Buffer Manager

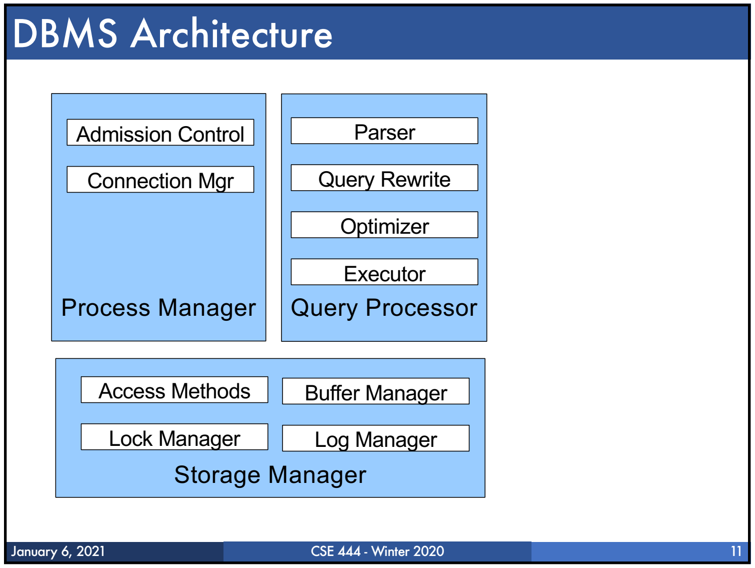
Lock Manager

Log Manager

Storage Manager

We will fill in implementation

10



DBMS Architecture

Admission Control

Connection Mgr

Process Manager

Parser

Query Rewrite

Optimizer

Executor

Query Processor

Access Methods

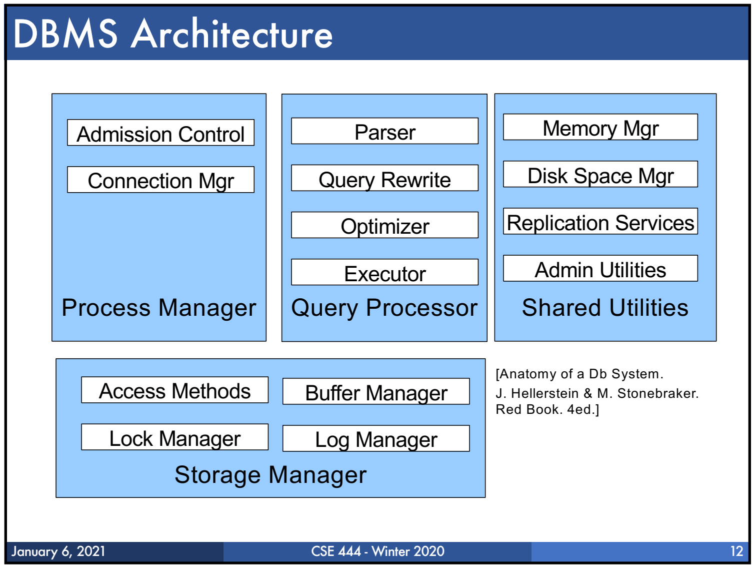
Buffer Manager

Lock Manager

Log Manager

Storage Manager

11



DBMS Architecture

Admission Control

Connection Mgr

Process Manager

Parser

Query Rewrite

Optimizer

Executor

Query Processor

Memory Mgr

Disk Space Mgr

Replication Services

Admin Utilities

Shared Utilities

Access Methods

Buffer Manager

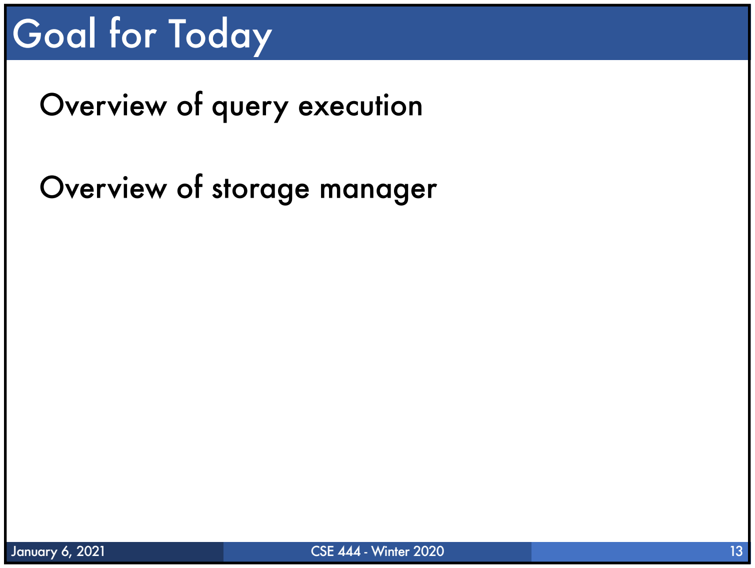
Lock Manager

Log Manager

Storage Manager

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

12

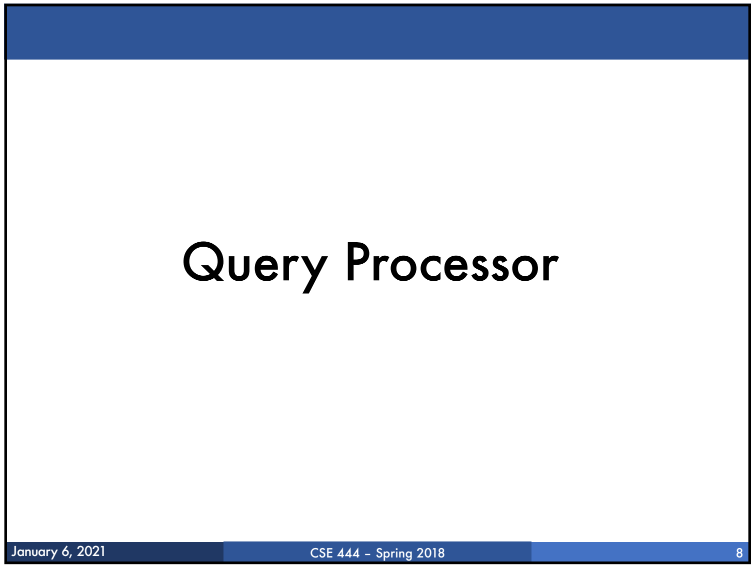


Goal for Today

Overview of query execution

Overview of storage manager

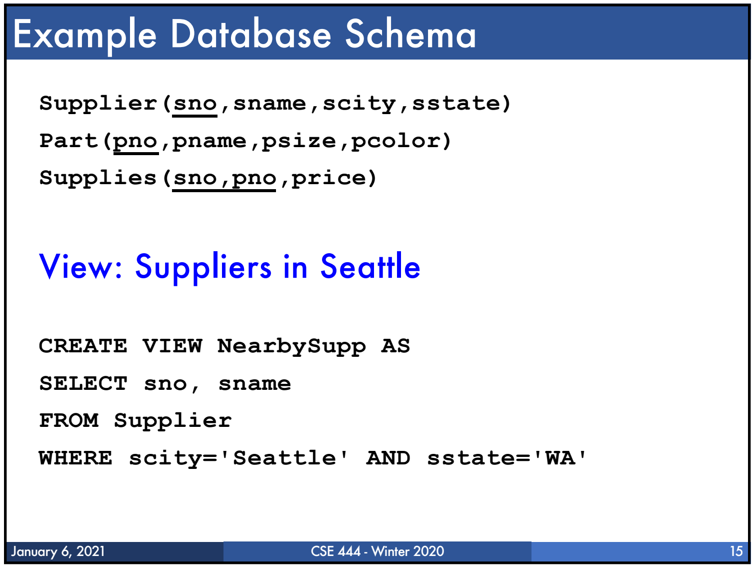
13



Query Processor

January 6, 2021 CSE 444 – Spring 2018 8

14



Example Database Schema

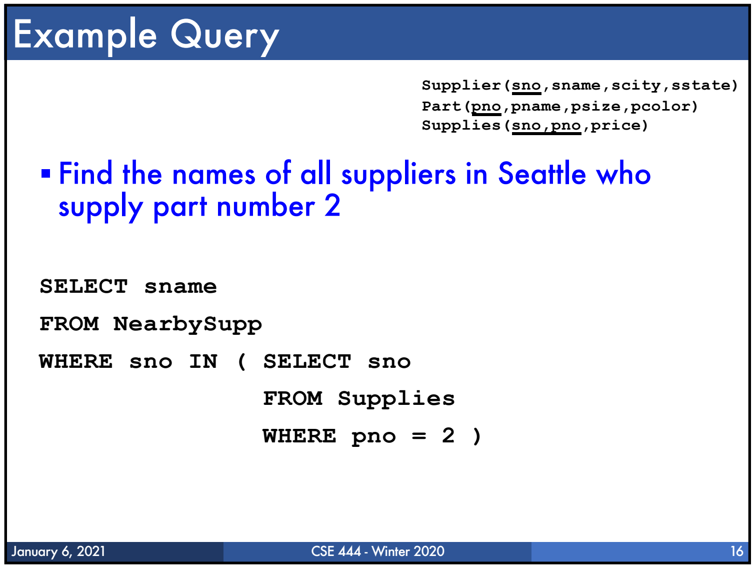
```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supplies(sno,pno,price)
```

View: Suppliers in Seattle

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

January 6, 2021 CSE 444 - Winter 2020 15

15



Example Query

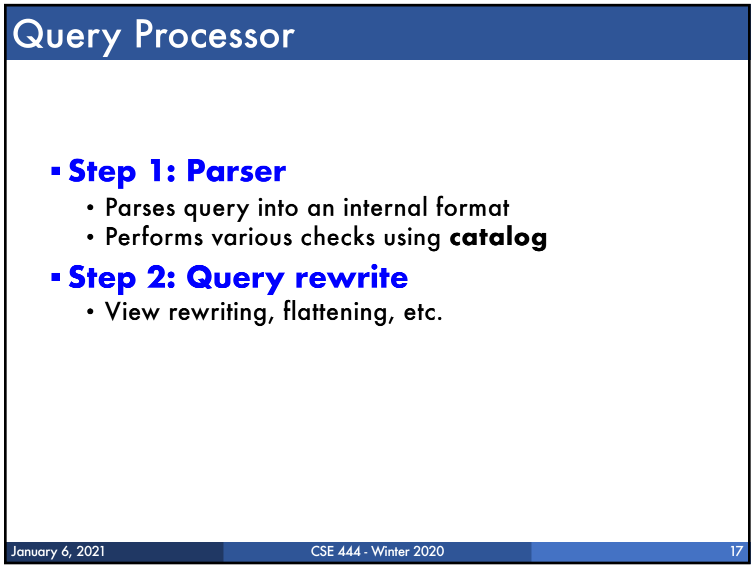
```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supplies(sno,pno,price)
```

- 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 )
```

January 6, 2021 CSE 444 - Winter 2020 16

16

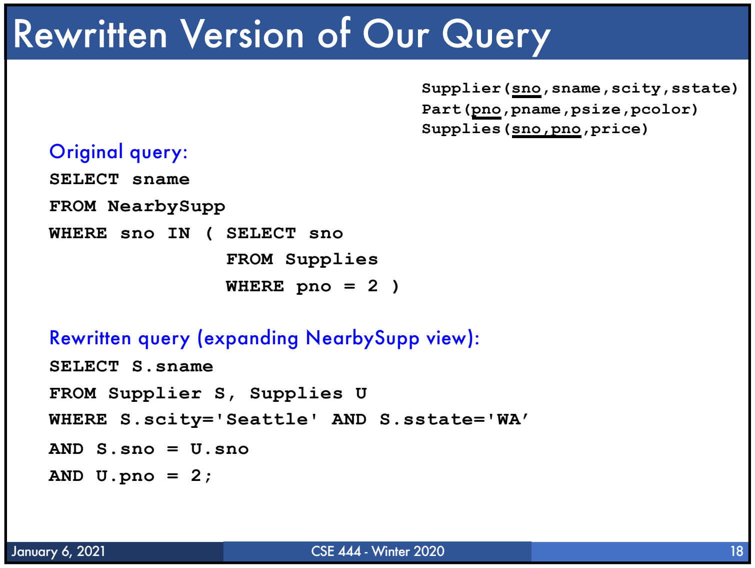


Query Processor

- **Step 1: Parser**
  - Parses query into an internal format
  - Performs various checks using **catalog**
- **Step 2: Query rewrite**
  - View rewriting, flattening, etc.

January 6, 2021 CSE 444 - Winter 2020 17

17



Rewritten Version of Our Query

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

Original query:

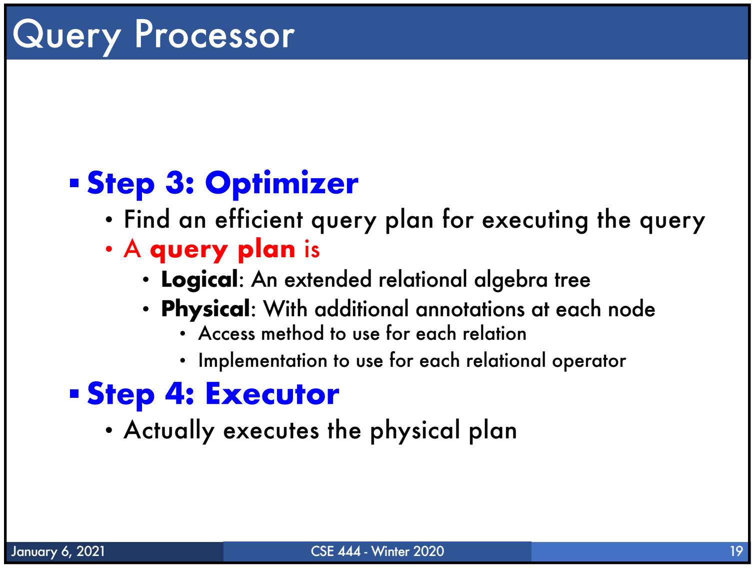
```
SELECT sname
FROM NearbySupp
WHERE sno IN ( SELECT sno
               FROM Supplies
               WHERE pno = 2 )
```

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;
```

January 6, 2021 CSE 444 - Winter 2020 18

18

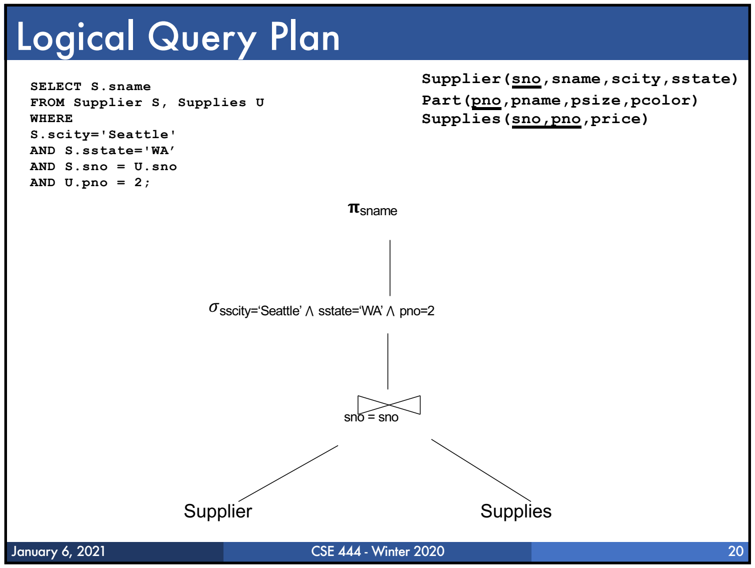


Query Processor

- **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

January 6, 2021 CSE 444 - Winter 2020 19

19



Logical Query Plan

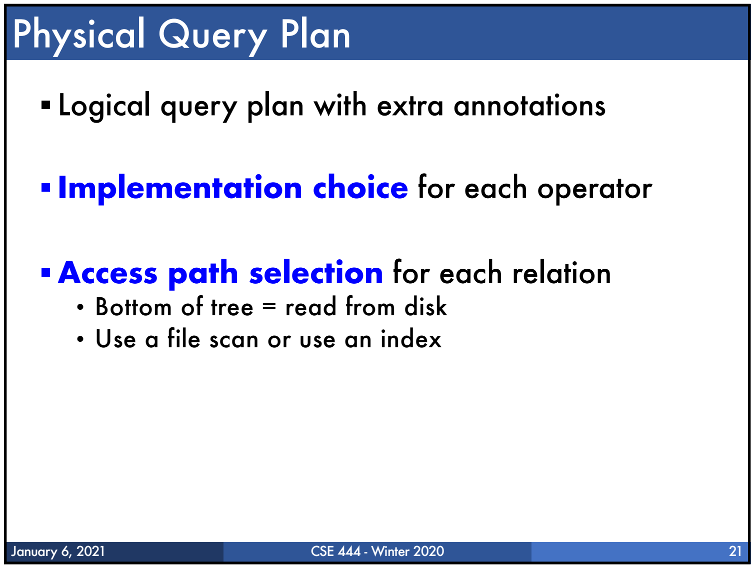
```
SELECT S.sname
FROM Supplier S, Supplies U
WHERE
S.scity='Seattle'
AND S.sstate='WA'
AND S.sno = U.sno
AND U.pno = 2;
```

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

π_{sname}
$\sigma_{\text{sscity='Seattle'} \land \text{sstate='WA'} \land \text{pno=2}}$
⋈ sno = sno
Supplier
Supplies

January 6, 2021 CSE 444 - Winter 2020 20

20



Physical Query Plan

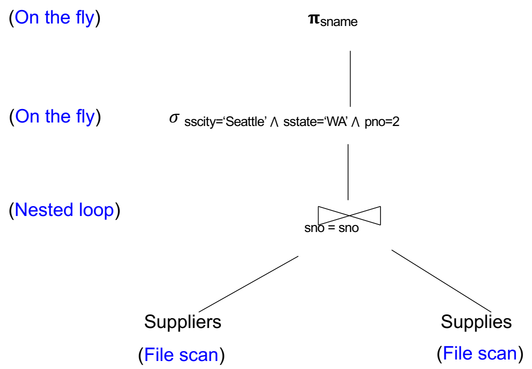
- Logical query plan with extra annotations
- **Implementation choice** for each operator
- **Access path selection** for each relation
  - Bottom of tree = read from disk
  - Use a file scan or use an index

January 6, 2021 CSE 444 - Winter 2020 21

21

Physical Query Plan

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



(On the fly) π_{sname}

(On the fly) $\sigma_{\text{sscity='Seattle'} \wedge \text{sstate='WA'} \wedge \text{pno=2}}$

(Nested loop) $\bowtie_{sno = sno}$

Suppliers (File scan)

Supplies (File scan)

22

Query Executor

23

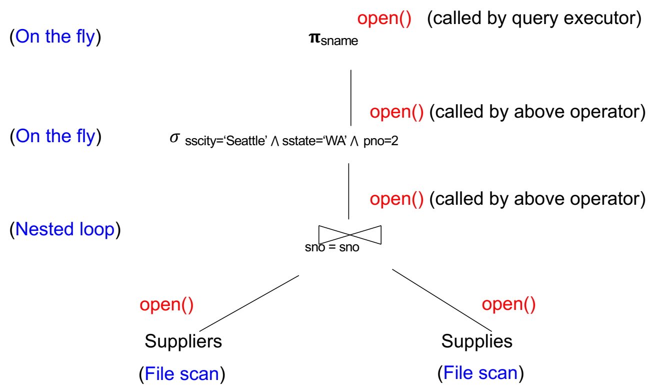
Iterator Interface

- Each **operator implements OpIterator.java**
- **open()**
  - Initializes operator state
  - Sets parameters such as selection predicate
- **next()**
  - **Returns a Tuple!**
  - 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

24

Query Execution

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



(On the fly) π_{sname} open() (called by query executor)

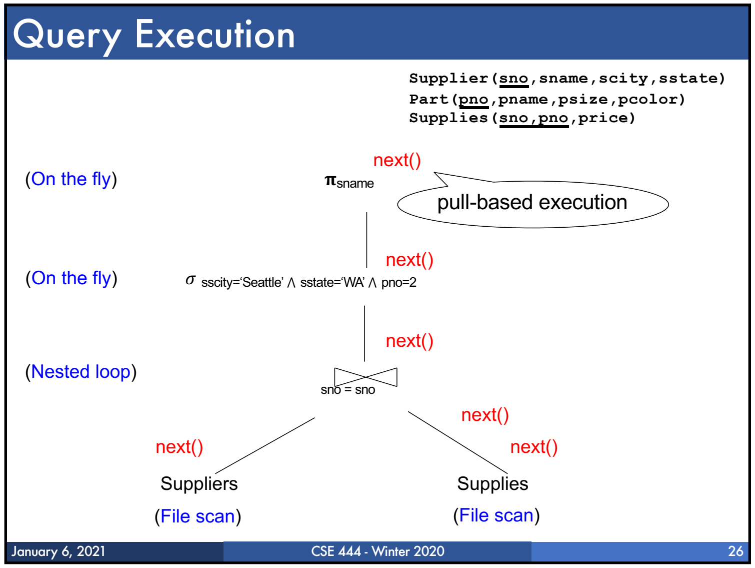
(On the fly) $\sigma_{\text{sscity='Seattle'} \wedge \text{sstate='WA'} \wedge \text{pno=2}}$ open() (called by above operator)

(Nested loop) $\bowtie_{sno = sno}$ open() (called by above operator)

open() Suppliers (File scan)

open() Supplies (File scan)

25



Query Execution

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

(On the fly) π_{sname} next()

pull-based execution

(On the fly) $\sigma_{sscity='Seattle' \wedge sstate='WA' \wedge pno=2}$ next()

(Nested loop) $\bowtie_{sno = sno}$ next()

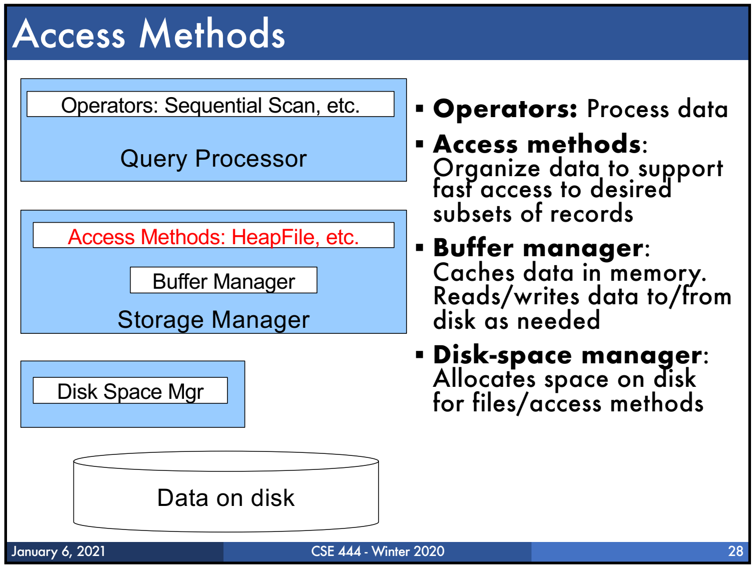
next() Suppliers (File scan)

next() next() Supplies (File scan)

26

Storage Manager

27



Access Methods

Operators: Sequential Scan, etc.

Query Processor

Access Methods: HeapFile, etc.

Buffer Manager

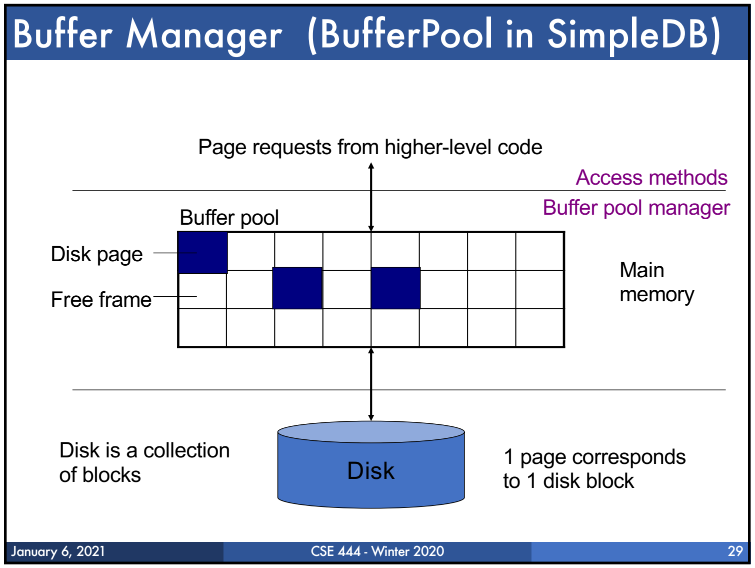
Storage Manager

Disk Space Mgr

Data on disk

- **Operators:** Process data
- **Access methods**: Organize data to support fast access to desired subsets of records
- **Buffer manager**: Caches data in memory. Reads/writes data to/from disk as needed
- **Disk-space manager**: Allocates space on disk for files/access methods

28



Buffer Manager (BufferPool in SimpleDB)

Page requests from higher-level code

Access methods

Buffer pool manager

Buffer pool

Disk page

Free frame

Main memory

Disk is a collection of blocks

Disk

1 page corresponds to 1 disk block

29

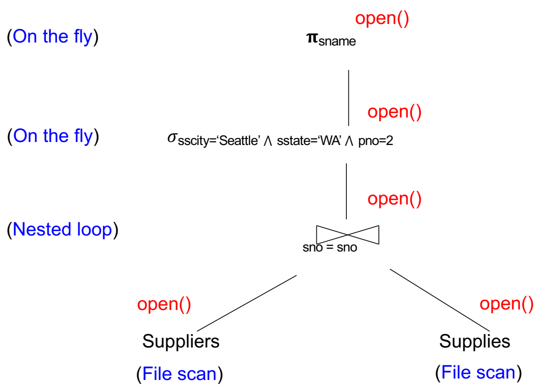
Buffer Manager

- Brings pages in from memory and caches them
- Eviction policies
  - Random page (ok for SimpleDB)
  - Least-recently used
  - The "clock" algorithm (see book)
- Keeps track of which **pages are dirty**
  - A dirty page has changes not reflected on disk
  - Implementation: Each page includes a dirty bit

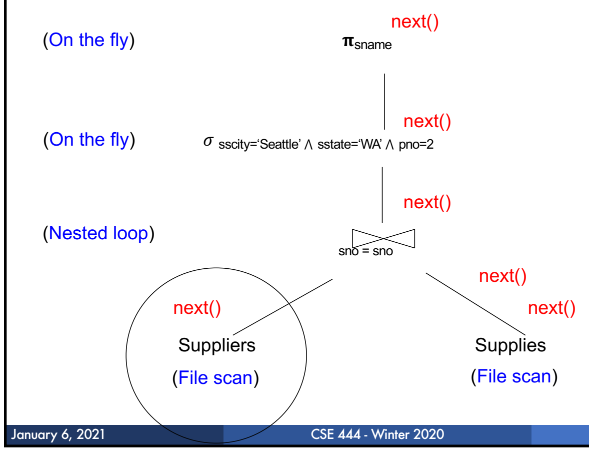
Access Methods

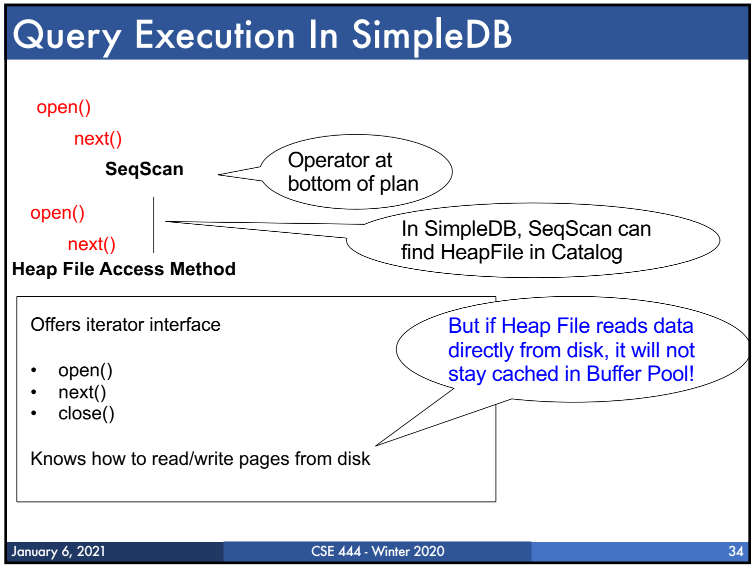
- A DBMS stores data on disk by breaking it into *pages*
  - A page is the size of a disk block.
  - A page is the unit of disk IO
- Buffer manager caches these pages in memory
- Access methods do the following:
  - They organize pages into collections called DB *files*
  - They organize data inside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

Query Execution



Query Execution





Query Execution In SimpleDB

open()
next()
SeqScan
Operator at bottom of plan
open()
next()
In SimpleDB, SeqScan can find HeapFile in Catalog
Heap File Access Method

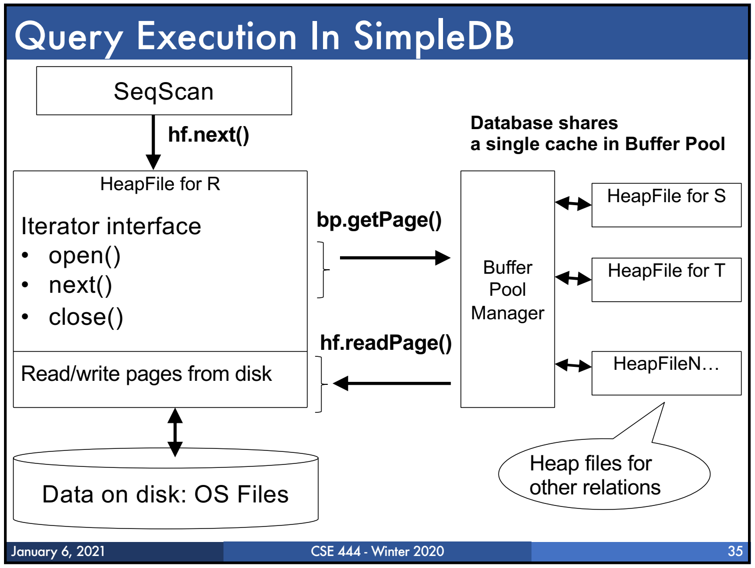
Offers iterator interface

- open()
- next()
- close()

Knows how to read/write pages from disk

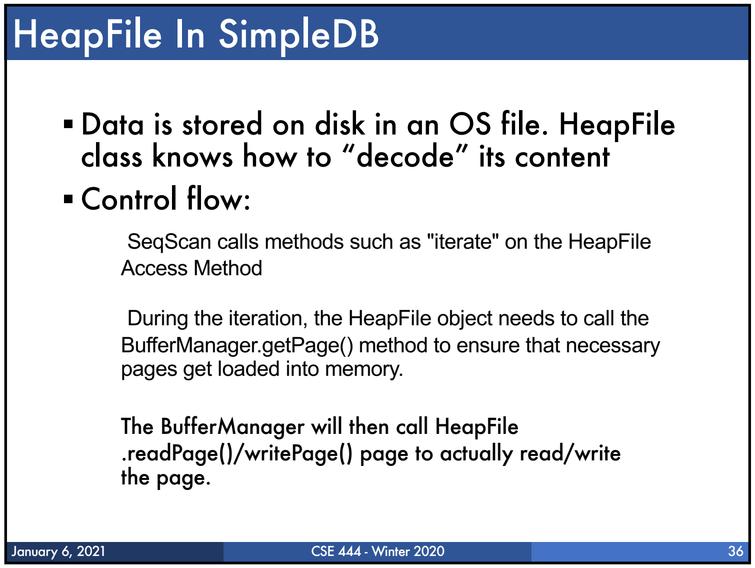
But if Heap File reads data directly from disk, it will not stay cached in Buffer Pool!

34



Query Execution In SimpleDB

35



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.

36