

Database System Internals Architecture

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

January 10, 2020 CSE 444 - Winter 2020 1

1

Announcements

- Room temperature being looked into
- Lab 1 part 1 is due on Monday at 11pm
 - Lab 1 in full is due on January 17th
 - "git pull upstream master" before building
 - Remember to git commit and git push often!
 - In Thursday section we will introduce the SimpleDB repo and structure
- HW1 is due next week on Friday
 - Print out PDF and hand in completed version
- 544M first paper review is also due next week
 - Can hand in the report to me in class
 - Deadlines are flexible for graduate readings

January 10, 2020 CSE 444 - Winter 2020 2

2

What we already know...

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

January 10, 2020 CSE 444 - Spring 2018 3

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

January 10, 2020 CSE 444 - Spring 2018 4

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*

January 10, 2020 CSE 444 - Spring 2018 5

5


Relational DBMS?

?

DBMS

SQL

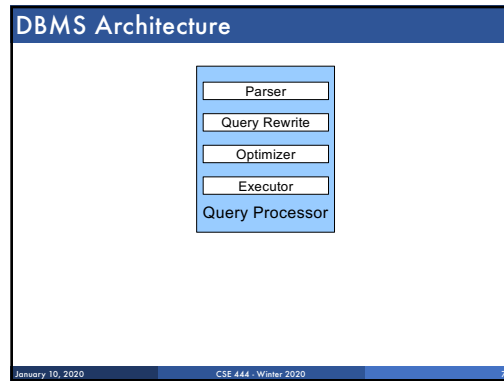
Data



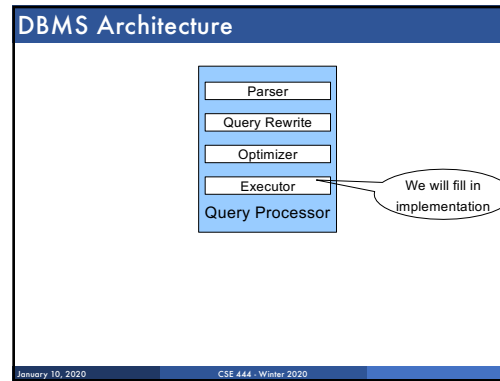
Key challenge: Achieve high performance on large databases!

January 10, 2020 CSE 444 - Spring 2018 6

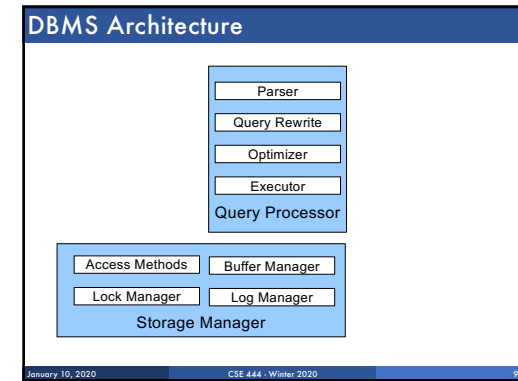
6



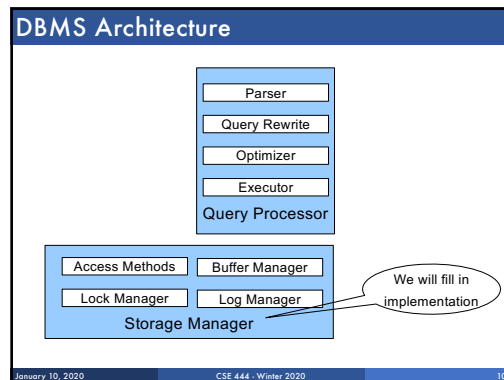
7



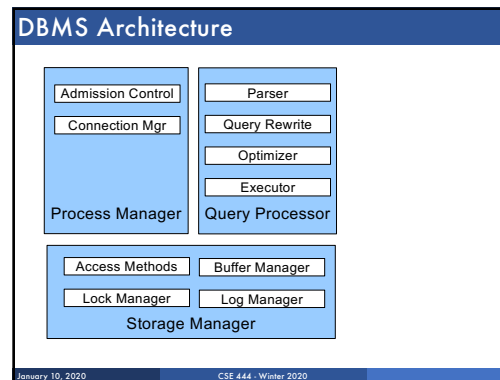
8



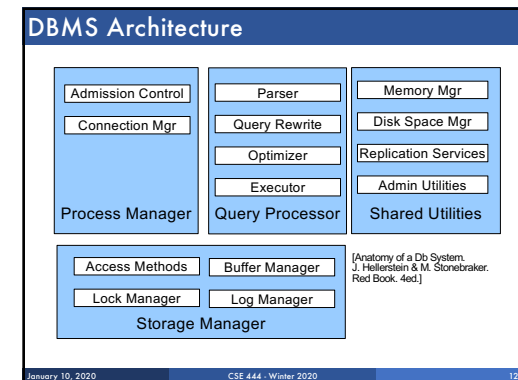
9



10



11



12

Goal for Today

- Overview of query execution
- Overview of storage manager

January 10, 2020 CSE 444 - Winter 2020 13

13

Query Processor

January 10, 2020 CSE 444 - Spring 2018 14

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 10, 2020 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 10, 2020 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 10, 2020 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 10, 2020 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 10, 2020 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)

The diagram shows a join operator (two triangles pointing to each other) with 'sno = sno' as the join predicate. It has two inputs: 'Supplier' and 'Supplies'. Above the join is a selection operator (a vertical line with a horizontal bar) with the predicate 'σ scity='Seattle' ∧ sstate='WA' ∧ pno=2'. The output of the selection is labeled 'Tename'.

January 10, 2020 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 10, 2020 CSE 444 - Winter 2020 21

21

Physical Query Plan

(On the fly)

(On the fly)

(Nested loop)

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

The diagram is identical to the Logical Query Plan, but with annotations: 'Suppliers (File scan)' under the 'Supplier' input, 'Supplies (File scan)' under the 'Supplies' input, and '(Nested loop)' next to the join operator. The selection operator is annotated with '(On the fly)' and the output 'Tename' is also annotated with '(On the fly)'.

January 10, 2020 CSE 444 - Winter 2020 22

22

Query Executor

January 10, 2020 CSE 444 - Winter 2020 23

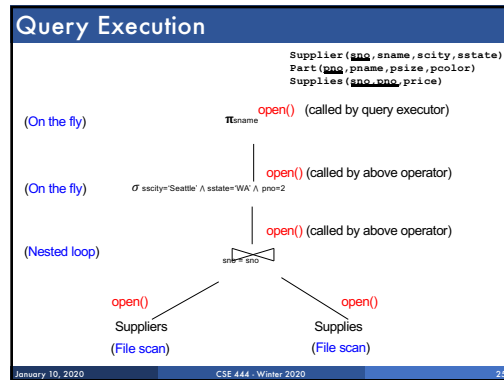
23

Iterator Interface

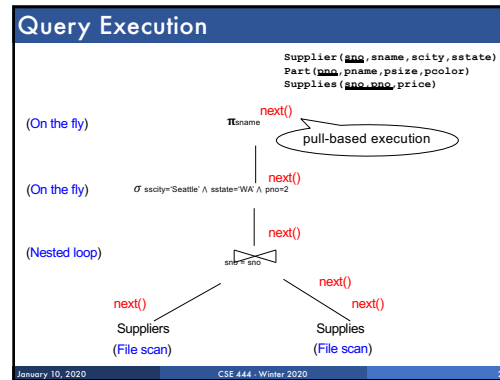
- Each operator implements **Oplerator.java**
- **open()**
 - Initializes operator state
 - Sets parameters such as selection predicate
- **next()**
 - **Returns a Tuple!**
 - Operator invokes next() recursively on its inputs
 - Performs processing and produces an output tuple
- **close():** clean-up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

January 10, 2020 CSE 444 - Winter 2020 24

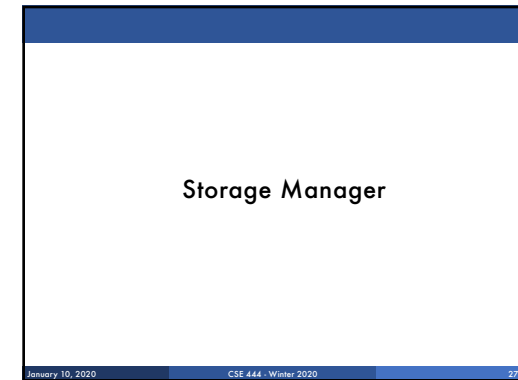
24



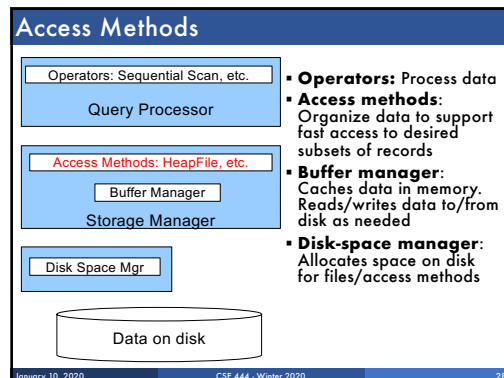
25



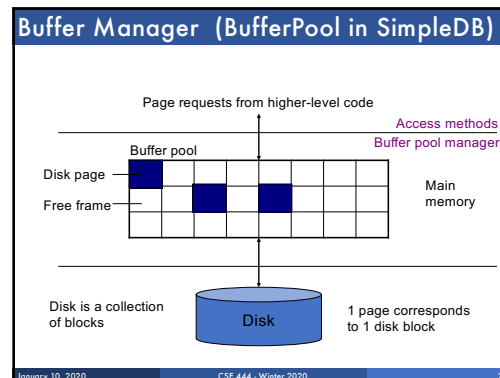
26



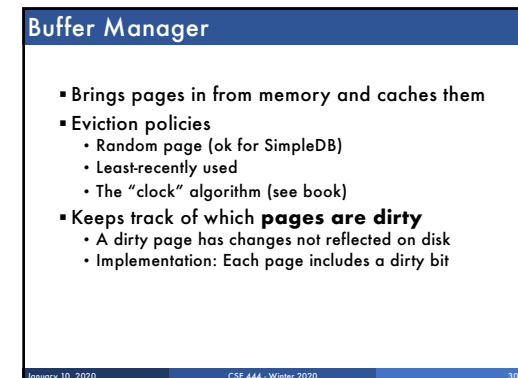
27



28



29



30

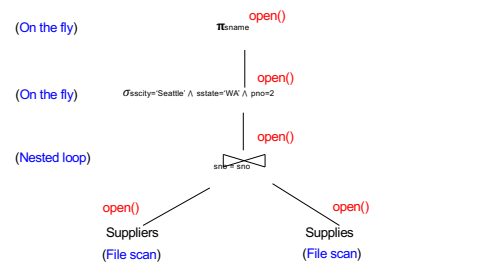
Access Methods

- 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

January 10, 2020 CSE 444 - Winter 2020 31

31

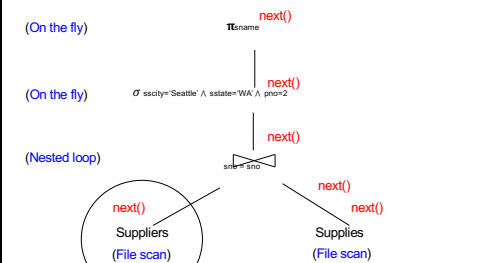
Query Execution



January 10, 2020 CSE 444 - Winter 2020 32

32

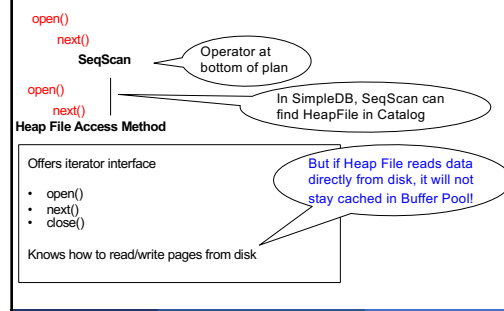
Query Execution



January 10, 2020 CSE 444 - Winter 2020 33

33

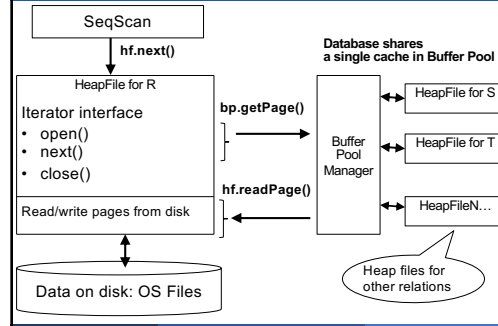
Query Execution In SimpleDB



January 10, 2020 CSE 444 - Winter 2020 34

34

Query Execution In SimpleDB



January 10, 2020 CSE 444 - Winter 2020 35

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.

January 10, 2020 CSE 444 - Winter 2020 36

36