CSE 444: Database Internals

Lecture 3
DBMS Architecture

Upcoming Deadlines

- Lab 1 Part 1 is due today at 11pm
  - Go through logistics of getting started
  - Start to make some small changes to the code
- HW1 is due on Wednesday at 11pm
  - Closely related to Lab 1
  - You need lecture 4 to finish the homework
  - 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 Monday at 11pm
- Lab 1 is due next Friday at 11pm
  - 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!
- Try to save late days for later in the quarter
- But no late days for final project

What we already know…

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

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

What we already know…

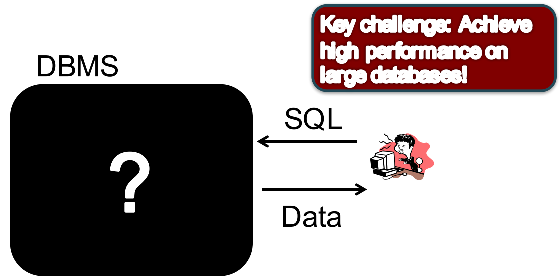
Relational Query Language:

- 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*

Benefits of relational model

- Physical data independence
  – Can change physical data organization on disk for performance *without affecting applications*
  – Thanks to logical data model and set-at-a-time query language
- Logical data independence
  – Can change logical schema *without affecting applications*
  – Thanks to views and query rewriting

How to Implement a Relational DBMS?



Goal for Today

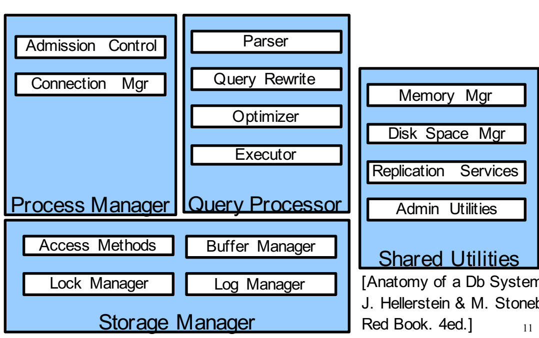
Overview of DBMS architecture

Overview of query execution

DBMS Architecture

(on the white board)

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'
```



```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(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**
    - Correctness, authorization, integrity constraints
    - Typically, catalog is stored in the form of set of relations
- **Step 2: Query rewrite**
  - View rewriting, flattening, etc.


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

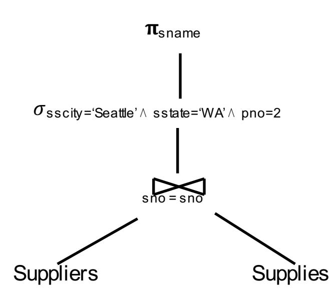
```
SELECT S.sname
FROM Supplier S, Supplies U
WHERE S.scity='Seattle' AND S.sstate='WA'
AND S.sno = U.sno
AND U.pno = 2;
```


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


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

Logical Query Plan

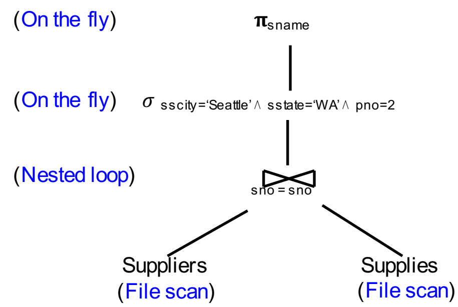


Physical Query Plan

- 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

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

Physical Query Plan



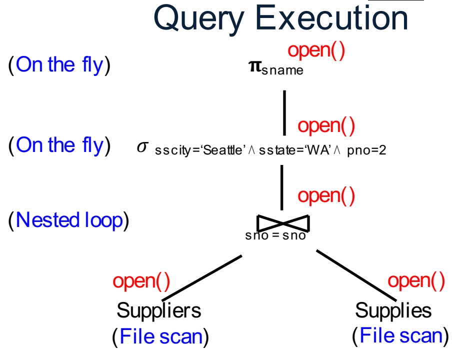
Query Executor

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

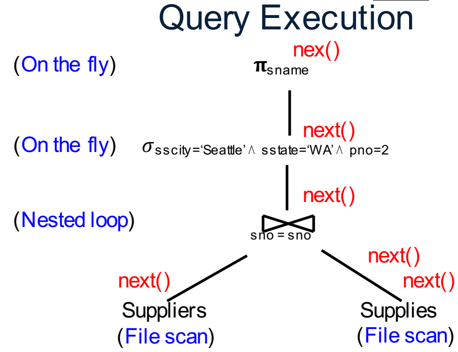
`Supplier(sno,sname,scity,sstate)`
`Part(pno,pname,psize,pcolor)`
`Supply(sno,pno,price)`

Query Execution



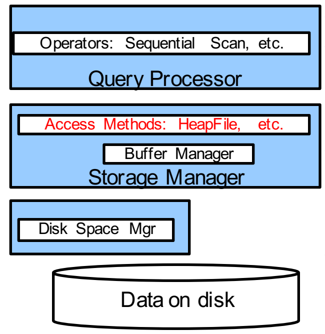
`Supplier(sno,sname,scity,sstate)`
`Part(pno,pname,psize,pcolor)`
`Supply(sno,pno,price)`

Query Execution



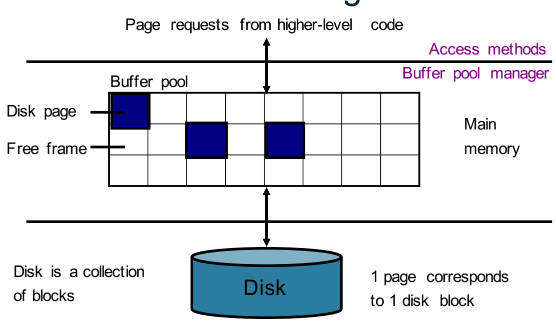
Storage Manager

Access Methods



- **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

Buffer Manager



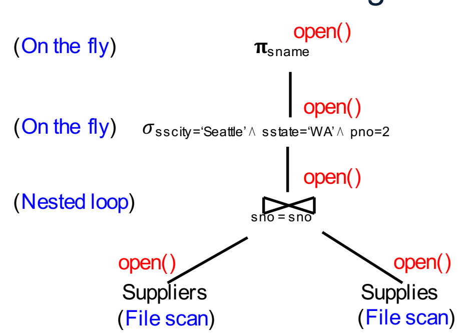
Buffer Manager

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

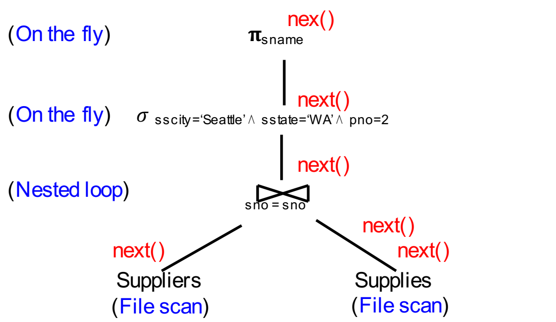
Access Methods

- 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

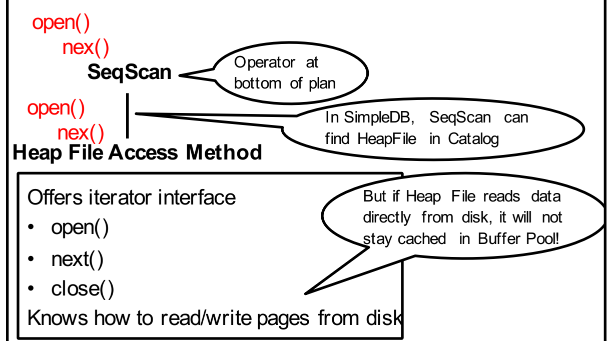
Query Execution
How it all Fits Together



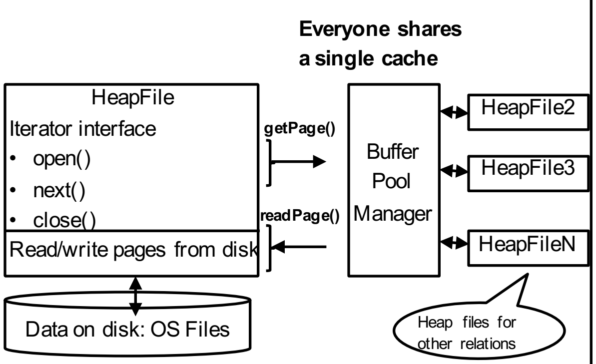
Query Execution
How it all Fits Together



Query Execution In SimpleDB



Query Execution In SimpleDB



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 DbFile Access Method
  - During the iteration, the DbFile object needs to call the BufferManager.getPage() method to ensure that necessary pages get loaded into memory.
  - The BufferManager will then call DbFile.read()/write() page to actually read/write the page.