UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

CS411 - Query Optimization

illinois.edu

Announcements

- HW3 is due today
- MP3 is coming
- Piazza poll results:
  - data mining
  - distributed databases

illinois.edu

Review

- What steps are involved in query processing?
- What is a parse tree?
- What is the role of the preprocessor?
- What is a logical query plan?
- What is a physical query plan?

Review

- Why do we care about rewriting our query plans?
- What are some heuristics for optimizing a query plan?
- Why do we care about estimating cost?

Cost Estimation

- We start with metadata (data dictionary)
  - B(R) number of blocks holding R
  - T(R) number of tuples in R
  - V(R,a) number of distinct values for attribute a in relation R
- Computed periodically offline
- Possible to compute better statistics
  - e.g. histograms on page 805

Projection Estimation

- Can be computed exactly
  - B(S)=(T(R)*size(newtuple))/blocksize
  - T(S)=T(R)
  - V(S,a)=V(R,a) (if a in S)

Selection Estimation

- For $S=\sigma_{a=c}(R)$
  - $T(S) \approx T(R)/V(R,A)$
- For $S=\sigma_{a<c}(R)$
  - $T(S) \approx T(R)/3$
- Treat AND as multiple nested selections
- Treat OR as independent (more on this later)
  - $T(S) \approx n(1-(1-m_1/n)(1-m_2/n))$

Example

- Given R(a,b,c) with:
  - T(R)=10,000
  - V(R,a)=50, V(R,b)=20, V(R,c)=100
- Let $S=\sigma_{a=5 \text{ AND } b<100}(R)$
  - $T(S) \approx (10{,}000/50)/3=67$

Selection with OR

- $S=\sigma_{c1\ OR\ c2}(R)$
  - Assume m_1 tuples satisfy c_1, m_2 satisfy c_2
  - Assume conditions are independent
  - $(1-m_1/n)$ fraction of tuples that don't satisfy c_1
  - $(1-m_2/n)$ fraction of tuples that don't satisfy c_2
  - $(1-m_1/n)(1-m_2/n)$ fraction that satisfy neither
  - $1-(1-m_1/n)(1-m_2/n)$ fraction that satisfy either
  - $T(S) \approx n(1-(1-m_1/n)(1-m_2/n))$

Example

- Given R(a,b,c) with:
  - T(R)=10,000
  - V(R,a)=50, V(R,b)=20, V(R,c)=100
- Let $S=\sigma_{a=5 \text{ OR } b<100}(R)$
  - Tuples satisfying a=5 ≈ 10k/50=200=m_1
  - Tuples satisfying b<100≈3333=m_2
  - T(S)≈(1-(1-200/10k)(1-3333/10k)=3466

illinois.edu

Join Estimation

- Arguably the most important cost
- We need to predict how many tuples will relate between R and S
  - Could be anywhere between 0 and $T(R)T(S)$
- We'll first work with natural join
  - $U=R(X, Y) \bowtie S(Y, Z)$
- Can easily generalize to other joins

Join Estimation

- Simplifying assumptions:
  1. If join attribute has more values in R than in S, ***all*** values in S occur in R
     - $V(R,Y) \geq V(S,Y)$
     - in other words, assume every tuple in S joins
  2. Non-join attributes do not lose values
     - $V(R \bowtie S,A) \approx V(R,A)$ for A not in Y
     - $V(R \bowtie S,B) \approx V(S,B)$ for B not in Y

illinois.edu

Join Estimation

- Probability a pair of tuples join: $1/\max(V(R,Y),V(S,Y))$
- Total number of pairs of tuples: $T(R)T(S)$
- $T(R \bowtie S) \approx T(R)T(S)/\max(V(R,Y),V(S,Y))$

Example

- $R(a,b,c) \bowtie S(b,d,e)$
  - $T(R)=1000$, $V(R,a)=100$, $V(R,b)=20$, $V(R,c)=200$
  - $T(S)=2000$, $V(S,b)=50$, $V(S,d)=100$, $V(S,e)=400$
- $T(R \bowtie S) \approx 1000*2000/\max(20,50)=40{,}000$

Other Operators

- Assume $T(S)<T(R)$
  - $T(R \cup S) \approx T(R)+T(S)/2$
  - $T(R \cap S) \approx T(S)/2$
  - $T(R-S) \approx T(R)-T(S)/2$
  - $T(\delta(R)) \approx \min(T(R)/2, V(R,a)V(R,b),\ldots V(R,z))$
  - $T(\gamma(R)) \approx \min(T(R)/2, V(R,a)V(R,b),\ldots V(R,z))$

Cost estimation

- Final result of cost estimation:
  - Ignore the input and the output
  - Sum up total number of I/Os
  - Proportional to total size of intermediate relations, so sum this up

Example

Given R(a,b) and S(b,c)

T(R)=5000, V(R,a)=50, V(R,b)=100

T(S)=2000, V(S,b)=200, V(S,c)=100

$U=\delta(\sigma_{a=10}(R \bowtie S))$

$$\left.\begin{array}{l} T(R \bowtie S)=5000*2000/200=50000 \\ T(\sigma_{a=10}(R \bowtie S))=50000/50=1000 \end{array}\right\} \text{cost=51,000}$$

$T(U)=\min(500,20000)=500$

Example

Given R(a,b) and S(b,c)

- T(R)=5000, V(R,a)=50, V(R,b)=100
- T(S)=2000, V(S,b)=200, V(S,c)=100

$U=\delta(\sigma_{a=10}(R))\bowtie\delta(S)$

- $T(\sigma_{a=10}(R))=100$, $T(\delta(\sigma_{a=10}(R)))=50$
- $T(\delta(S))=1000$
- $T(U)=1000*50/200=250$

Example

Given R(a,b) and S(b,c)

$T(R)=5000$, $V(R,a)=50$, $V(R,b)=100$

$T(S)=2000$, $V(S,b)=200$, $V(S,c)=100$

$U=\delta(\sigma_{a=10}(R))\bowtie\delta(S)$

$$\left.\begin{array}{l} T(\sigma_{a=10}(R))=100,\ T(\delta(\sigma_{a=10}(R)))=50 \\ T(\delta(S))=1000 \end{array}\right\} \text{cost=1,150}$$

$T(U)=1000*50/200=250$

Query Optimization

- We need three things:
  1. ~~Rewrite rules~~
  2. An optimization algorithm
  3. ~~A cost estimator~~

Naïve Approach

- Try all possible rewrites of our query
- Estimate the cost of each one
- Choose the one with the lowest cost

Naïve Approach

$\gamma_{a,b,avg(x)}(\sigma_C(R \bowtie S \bowtie T \bowtie U \bowtie V \bowtie W))$

- Number of binary trees with n leaves=(n-1) Catalan number= [2(n-1)]!/[(n!)(n-1)!]
- Number of arrangements of n leaves=n!
- Number of ways to reorder n joins=[2(n-1)]!/(n-1)!

Reordering JUST the joins in this problem: 30,240

For 10 joins: 17,643,225,600

Naïve Approach

- For realistic queries, naïve approach is impractical
  - Can't afford to enumerate all possible plans
  - Takes longer than the query itself!

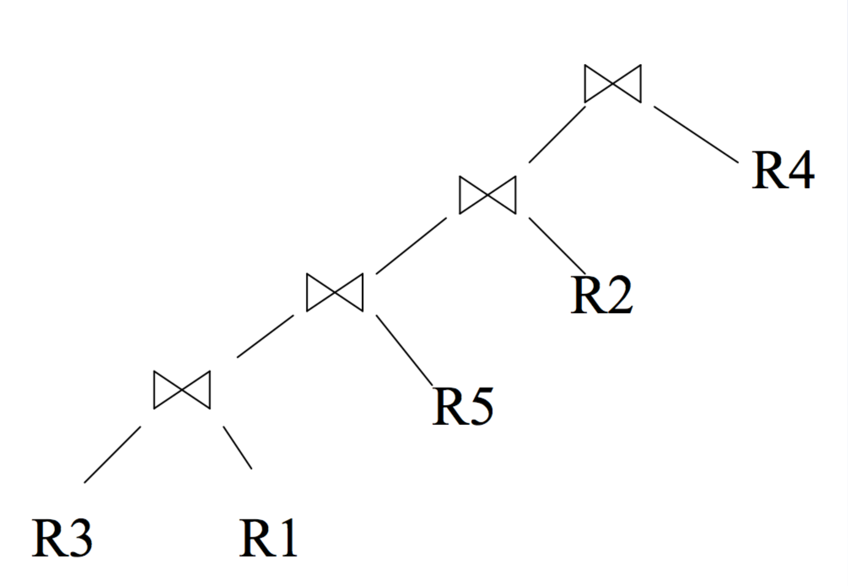
Search Strategies

- Optimization on a large search space
  - Greedy
  - Branch and Bound
  - Hill Climbing
  - Simulated annealing
  - Genetic algorithms
  - Stochastic optimization
  - … and many, many, many, many more

Join Order

- Choosing join order is very important
  - Why?
- We will focus entirely on this task:
  - given a series of n joins
  - assume we can compute a cost for each join
  - Find the best tree

Left Join Tree



- A tree topology with two advantages:
  1. Only n! different orderings possible
  2. Can be easily pipelined with our algorithms
- On examples/exams, only use left join trees

Dynamic Programming

- Main idea: in large combinatorial problems, many of the partial results are helpful in later steps
- Example:
  $\text{Cost}((R \bowtie S) \bowtie T)$ result is useful for computing
  $\text{Cost}((R \bowtie S) \bowtie T) \bowtie U)$
  $\text{Cost}(((R \bowtie S) \bowtie T) \bowtie V)$
  $\text{Cost}(((R \bowtie S) \bowtie T) \bowtie W)$

Dynamic Programming

- For each subset of $\{R_1,R_2,\ldots R_n\}$, compute best plan for that subset
  - Do this in increasing order of cardinality
  - Step 0: $\{R_1\},\{R_2\},\ldots\{R_n\}$
  - Step 1: $\{R_{1,}R_2\},\{R_{1,}R_3\},\ldots\{R_{n-1,}R_n\}$
  - …
  - Step n: $\{R_{1,}R_2,\ldots,R_n\}$

Dynamic Programming

- For each subset $S \subseteq \{R_1, R_2, \ldots R_n\}$
  - Compute size of T(S)
  - Find the best plan for S
  - Compute cost of S

Dynamic Programming

- Step 0: for each R_i
  - Size(R_i)=0
  - Plan(R_i)=R_i
  - Cost(R_i)=0

Dynamic Programming

- Step i: $S \subseteq \{R_1, R_2, \ldots R_n\}$ of cardinality i+1
  - For each pair of smaller subsets: S' and S'' such that S=S'∪S'', find cost of best plan for S'⋈S''
  - Size(S)=T(S)
  - Plan(S)=best plan
  - Cost(S)=cost of best plan

Dynamic Programming

- We already have best plan and cost for smaller subsets S' and S''
- Cost(S)= Cost(S')+Cost(S'')+Size(S')+Size(S'')
  - Size(A)=0 if A is a relation (it's input!)

Example

- $Cost(R \ltimes S) = Cost(R) + Cost(S) + 0 + 0 = 0$
- $Cost((R \ltimes S) \ltimes T) = Cost(R \ltimes S) + Cost(T) + Size(R \ltimes S) + Size(T) = Size(R \ltimes S)$

Example

- Given R(a,b),S(b,c),T(c,d), U(d,a)
  - T(R)=T(S)=T(T)=T(U)=1k
  - V(R,a)=100, V(R,b)=200
  - V(S,b)=100, V(S,c)=500
  - V(T,c)=20, V(T,d)=500
  - V(U,a)=50, V(U,d)=1k

Example

Subset	Size(S)	Cost	Plan
{R,S}			
{R,T}			
{R,U}			
{S,T}			
{S,U}			
{T,U}			
{R,S,T}			
{R,S,U}			
{R,T,U}			
{S,T,U}			
{R,S,T,U}			

REMEMBER!

- We're ONLY considering left join trees!
  - $((R \bowtie S) \bowtie T) \bowtie U$ is valid
  - $(R \bowtie S) \bowtie (T \bowtie U)$ is ignored!
  - It ***is*** possible to do dynamic programming that considers all possible trees, though
- $R \bowtie S$ and $S \bowtie R$ could have different costs
  - estimation could depend on the algorithm

Example

Subset	Size(S)	Cost	Plan
{R,S}			R⋈S
{R,T}			
{R,U}			
{S,T}			
{S,U}			
{T,U}			
{R,S,T}			
{R,S,U}			
{R,T,U}			
{S,T,U}			
{R,S,T,U}			

Example

Subset	Size(S)	Cost	Plan
{R,S}	T(R)T(S)/max(V(R,b),V(S,b))		R⋈S
{R,T}			
{R,U}			
{S,T}			
{S,U}			
{T,U}			
{R,S,T}			
{R,S,U}			
{R,T,U}			
{S,T,U}			
{R,S,T,U}			

Example

Subset	Size(S)	Cost	Plan
{R,S}	1k*1k/200		R⋈S
{R,T}			
{R,U}			
{S,T}			
{S,U}			
{T,U}			
{R,S,T}			
{R,S,U}			
{R,T,U}			
{S,T,U}			
{R,S,T,U}			

Example

Subset	Size(S)	Cost	Plan
{R,S}	5k	0	R⋈S
{R,T}			
{R,U}			
{S,T}			
{S,U}			
{T,U}			
{R,S,T}			
{R,S,U}			
{R,T,U}			
{S,T,U}			
{R,S,T,U}			

Example

Subset	Size(S)	Cost	Plan
{R,S}	5k	0	R⋈S
{R,T}	1,000k	0	R⋈T
{R,U}	10k	0	R⋈U
{S,T}	2k	0	S⋈T
{S,U}	1,000k	0	S⋈U
{T,U}	1k	0	T⋈U
{R,S,T}			
{R,S,U}			
{R,T,U}			
{S,T,U}			
{R,S,T,U}			

Example

Subset	Size(S)	Cost	Plan
{R,S}	5k	0	R⋈S
{R,T}	1,000k	0	R⋈T
{R,U}	10k	0	R⋈U
{S,T}	2k	0	S⋈T
{S,U}	1,000k	0	S⋈U
{T,U}	1k	0	T⋈U
{R,S,T}	(1k*1k*1k)/(500*200)=10k	5k+0	(R⋈S)⋈T
{R,S,U}			
{R,T,U}			
{S,T,U}			
{R,S,T,U}			

Example

Subset	Size(S)	Cost	Plan
{R,S}	5k	0	R⋈S
{R,T}	1,000k	0	R⋈T
{R,U}	10k	0	R⋈U
{S,T}	2k	0	S⋈T
{S,U}	1,000k	0	S⋈U
{T,U}	1k	0	T⋈U
{R,S,T}	(1k*1k*1k)/(500*200)=10k	1,000k+0	(R⋈T)⋈S
{R,S,U}			
{R,T,U}			
{S,T,U}			
{R,S,T,U}			

Example

Subset	Size(S)	Cost	Plan
{R,S}	5k	0	R⋈S
{R,T}	1,000k	0	R⋈T
{R,U}	10k	0	R⋈U
{S,T}	2k	0	S⋈T
{S,U}	1,000k	0	S⋈U
{T,U}	1k	0	T⋈U
{R,S,T}	(1k*1k*1k)/(500*200)=10k	2k+0	(S⋈T)⋈R
{R,S,U}			
{R,T,U}			
{S,T,U}			
{R,S,T,U}			

Example

Subset	Size(S)	Cost	Plan
{R,S}	5k	0	R⋈S
{R,T}	1,000k	0	R⋈T
{R,U}	10k	0	R⋈U
{S,T}	2k	0	S⋈T
{S,U}	1,000k	0	S⋈U
{T,U}	1k	0	T⋈U
{R,S,T}	10k	2k	(S⋈T)⋈R
{R,S,U}			
{R,T,U}			
{S,T,U}			
{R,S,T,U}			

Example

Subset	Size(S)	Cost	Plan
{R,S}	5k	0	R⋈S
{R,T}	1,000k	0	R⋈T
{R,U}	10k	0	R⋈U
{S,T}	2k	0	S⋈T
{S,U}	1,000k	0	S⋈U
{T,U}	1k	0	T⋈U
{R,S,T}	10k	2k	(S⋈T)⋈R
{R,S,U}	50k	5k	(R⋈S)⋈U
{R,T,U}	10k	1k	(T⋈U)⋈R
{S,T,U}	2k	1k	(T⋈U)⋈S
{R,S,T,U}			

Example

Subset	Size(S)	Cost	Plan
{R,S}	5k	0	R⋈S
{R,T}	1,000k	0	R⋈T
{R,U}	10k	0	R⋈U
{S,T}	2k	0	S⋈T
{S,U}	1,000k	0	S⋈U
{T,U}	1k	0	T⋈U
{R,S,T}	10k	2k	(S⋈T)⋈R
{R,S,U}	50k	5k	(R⋈S)⋈U
{R,T,U}	10k	1k	(T⋈U)⋈R
{S,T,U}	2k	1k	(T⋈U)⋈S
{R,S,T,U}	(1k*1k*1k*1k)/ (1k*500*200*100)=10k	10k+2k+0+0=12k	((S⋈T)⋈R)⋈U

Example

Subset	Size(S)	Cost	Plan
{R,S}	5k	0	R⋈S
{R,T}	1,000k	0	R⋈T
{R,U}	10k	0	R⋈U
{S,T}	2k	0	S⋈T
{S,U}	1,000k	0	S⋈U
{T,U}	1k	0	T⋈U
{R,S,T}	10k	2k	(S⋈T)⋈R
{R,S,U}	50k	5k	(R⋈S)⋈U
{R,T,U}	10k	1k	(T⋈U)⋈R
{S,T,U}	2k	1k	(T⋈U)⋈S
{R,S,T,U}	(1k*1k*1k*1k)/ (1k*500*200*100)=10k	50k+5k+0+0=55k	((R⋈S)⋈U)⋈T

Example

Subset	Size(S)	Cost	Plan
{R,S}	5k	0	R⋈S
{R,T}	1,000k	0	R⋈T
{R,U}	10k	0	R⋈U
{S,T}	2k	0	S⋈T
{S,U}	1,000k	0	S⋈U
{T,U}	1k	0	T⋈U
{R,S,T}	10k	2k	(S⋈T)⋈R
{R,S,U}	50k	5k	(R⋈S)⋈U
{R,T,U}	10k	1k	(T⋈U)⋈R
{S,T,U}	2k	1k	(T⋈U)⋈S
{R,S,T,U}	(1k*1k*1k*1k)/ (1k*500*200*100)=10k	10k+1k+0+0=11k	((T⋈U)⋈R)⋈S

Example

Subset	Size(S)	Cost	Plan
{R,S}	5k	0	R⋈S
{R,T}	1,000k	0	R⋈T
{R,U}	10k	0	R⋈U
{S,T}	2k	0	S⋈T
{S,U}	1,000k	0	S⋈U
{T,U}	1k	0	T⋈U
{R,S,T}	10k	2k	(S⋈T)⋈R
{R,S,U}	50k	5k	(R⋈S)⋈U
{R,T,U}	10k	1k	(T⋈U)⋈R
{S,T,U}	2k	1k	(T⋈U)⋈S
{R,S,T,U}	((1k*1k*1k*1k)/ (1k*500*200*100)=10k	2k+1k+0+0=3k	((T⋈U)⋈S)⋈R

Example

Subset	Size(S)	Cost	Plan
{R,S}	5k	0	R⋈S
{R,T}	1,000k	0	R⋈T
{R,U}	10k	0	R⋈U
{S,T}	2k	0	S⋈T
{S,U}	1,000k	0	S⋈U
{T,U}	1k	0	T⋈U
{R,S,T}	10k	2k	(S⋈T)⋈R
{R,S,U}	50k	5k	(R⋈S)⋈U
{R,T,U}	10k	1k	(T⋈U)⋈R
{S,T,U}	2k	1k	(T⋈U)⋈S
{R,S,T,U}	10k	3k	((T⋈U)⋈S)⋈R

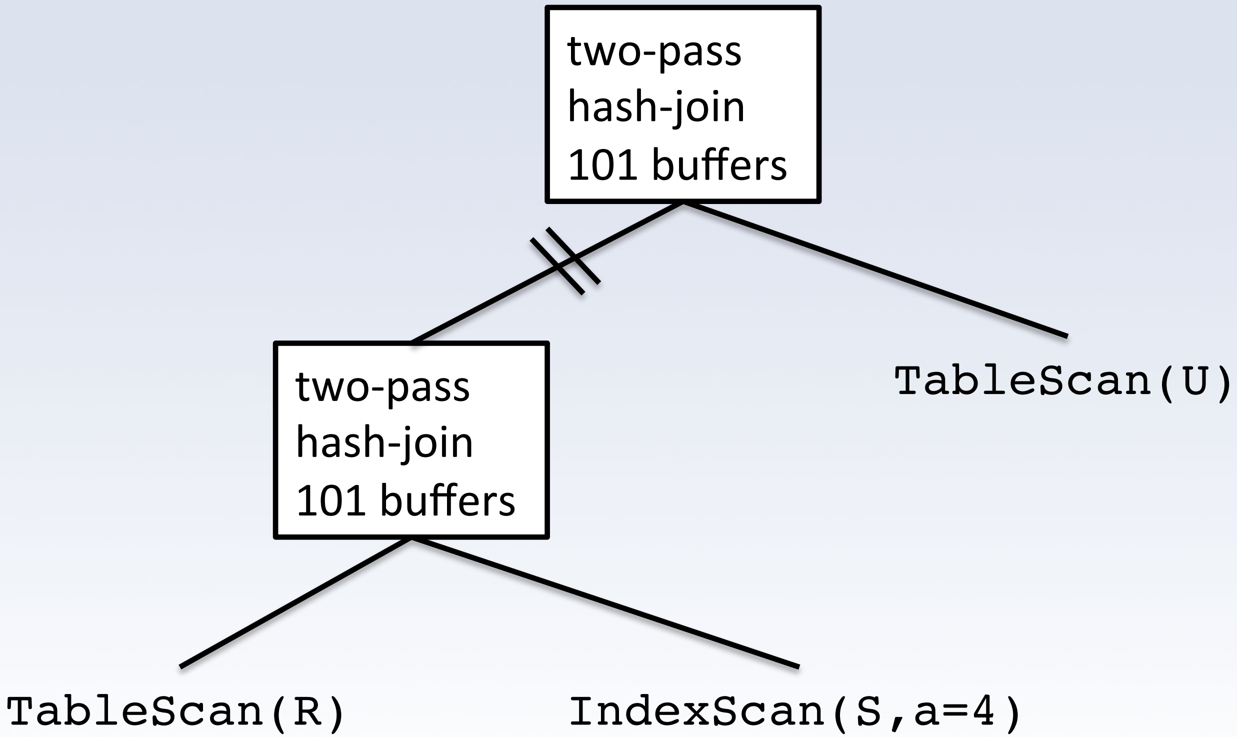
Physical Query Plan

- At each node of logical query plan:
  - Replace relational algebra operator with execution algorithms we discussed
    - How much memory is available?
    - Are relations indexed?
    - Are results sorted?
    - Should we pipeline or materialize intermediate results?

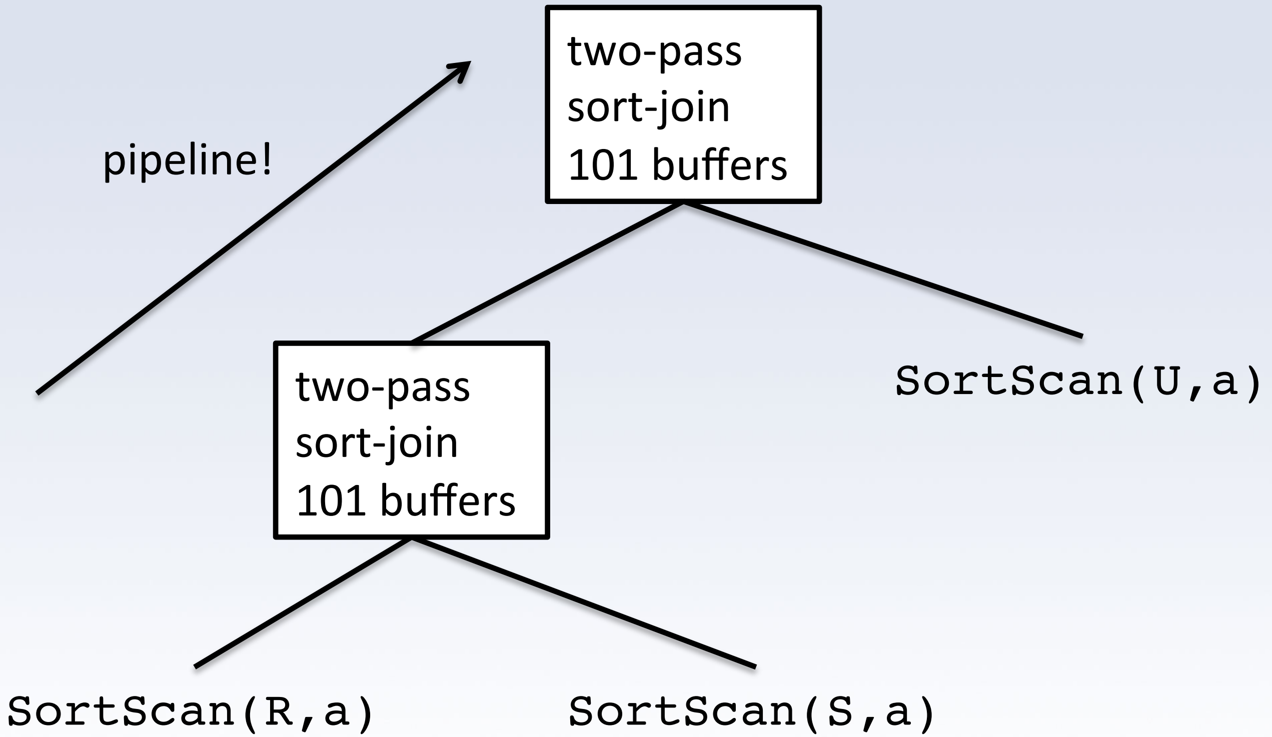
Pipelining vs. Materialization

- Materialization - write the results of the operation back to the disk
  - results in 2 intermediate disk I/Os per block
- Pipelining - keep results in memory buffers
  - Results of one iterator passed to the next

Physical Query Plan

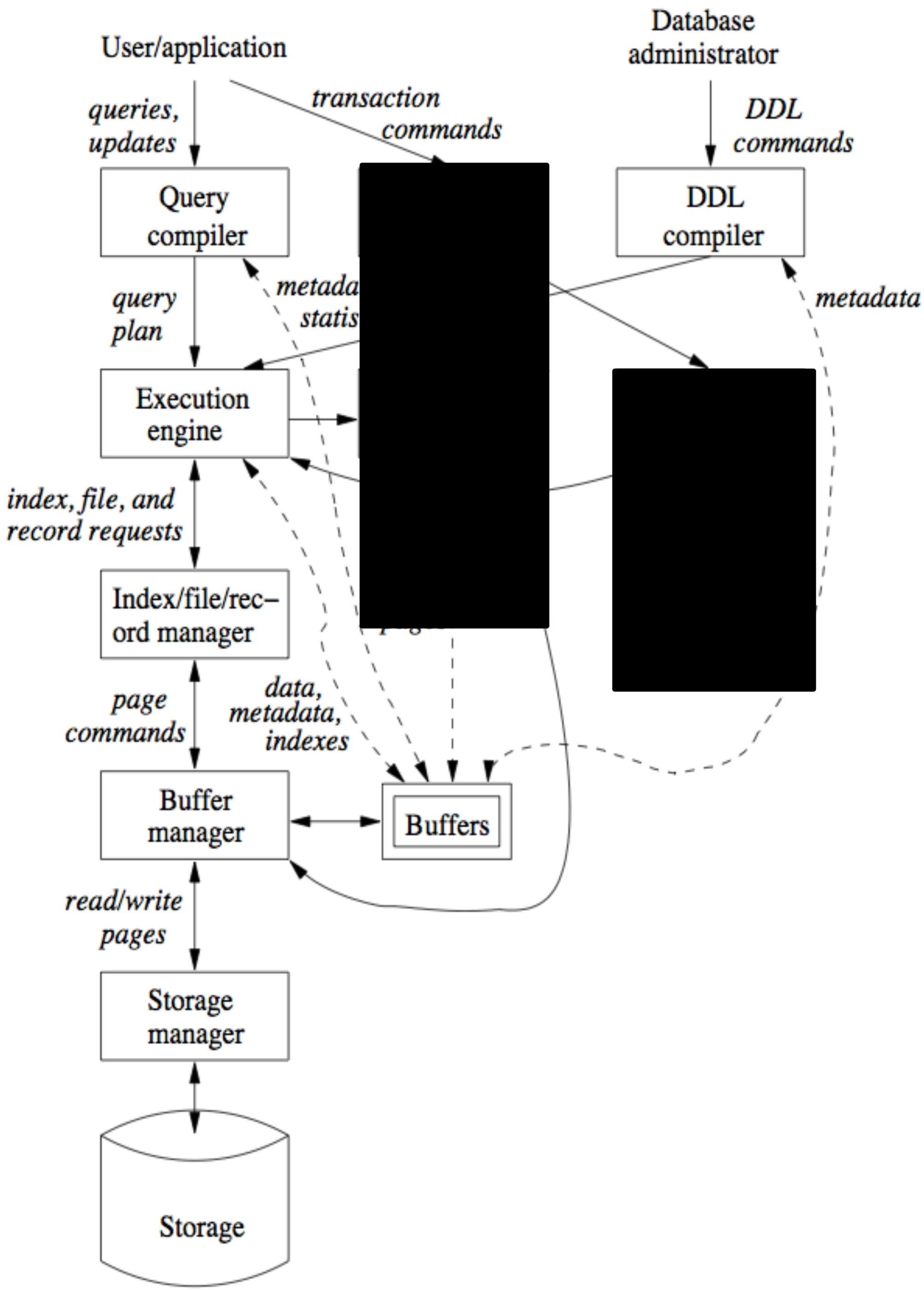


Physical Query Plan

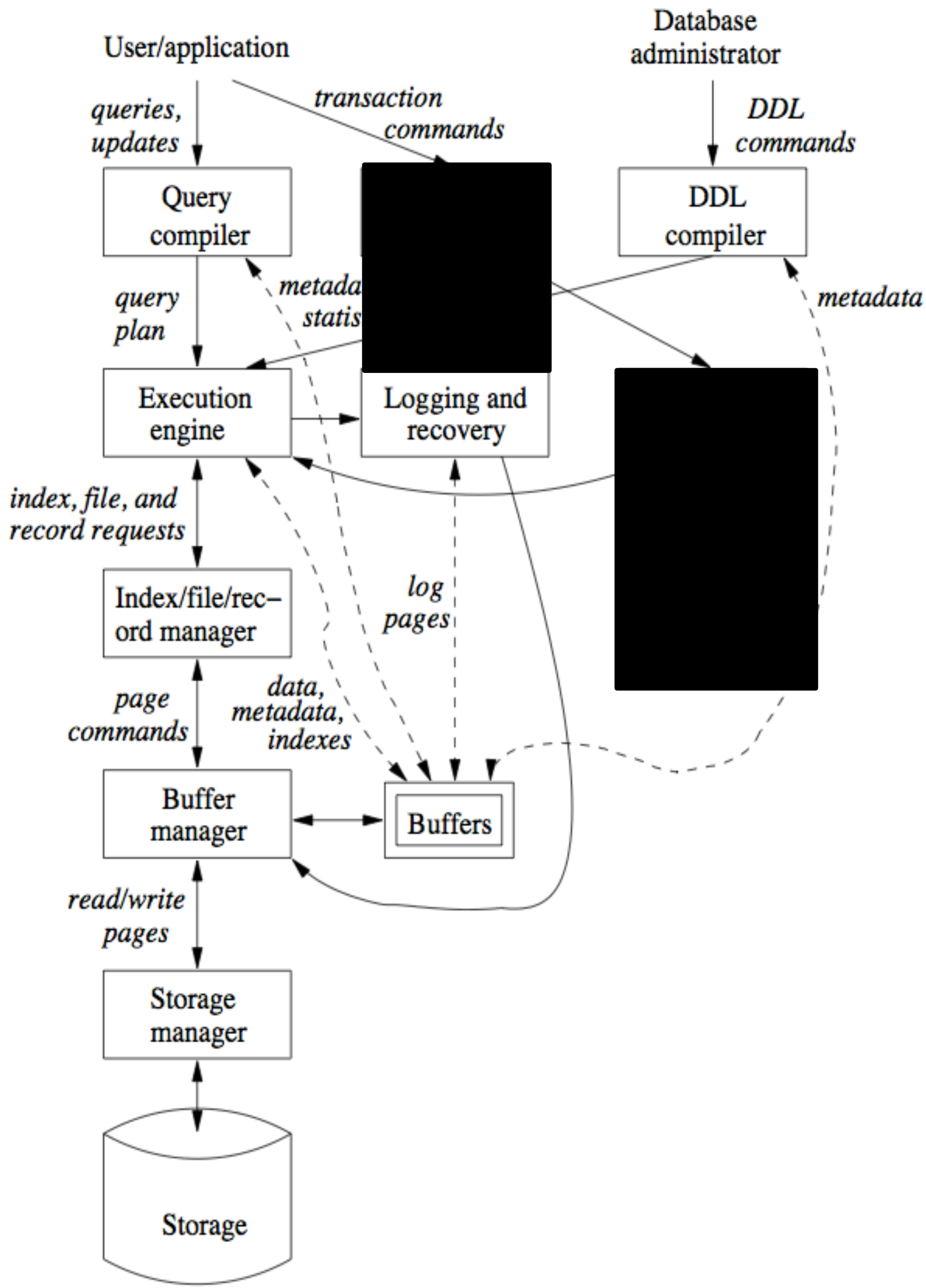


MP3

- We'll give you
  - a series of query plans
  - a max buffer size (amount of memory)
  - a choice of implementation algorithms
- Your job:
  - improve the query plans


User/application
Database administrator
queries, updates
transaction commands
DDL commands
Query compiler
[redacted]
DDL compiler
metada[redacted]
statis[redacted]
metadata
query plan
Execution engine
[redacted]
index, file, and record requests
Index/file/rec–ord manager
page commands
data, metadata, indexes
Buffer manager
Buffers
read/write pages
Storage manager
Storage

Next week...



illinois.edu