UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

CS411 - Query Parsing/Optimization

illinois.edu

Announcements

- HW3 is due Friday
- MP3 is coming
  - Related to query optimization
  - Won't involve programming
- Piazza poll for advanced topics

Review

- What is the “buffer pool”?
- What is a “table scan”?
- What is the main idea of the two pass sort based algorithms?

Review

- What is the main idea of the two pass hash based algorithms?
- How was “hybrid hash join” different?

One-Pass/NBLJ Summary

Operator	M required	I/O Cost
σ, π	1	B
δ, γ	B	B
$\cup, \cap, -, \bowtie, \times$	min(B(R),B(S))	B(R)+B(S)
$\bowtie$	$M \geq 2$	B(R)B(S)/M

Sorting-Based Summary

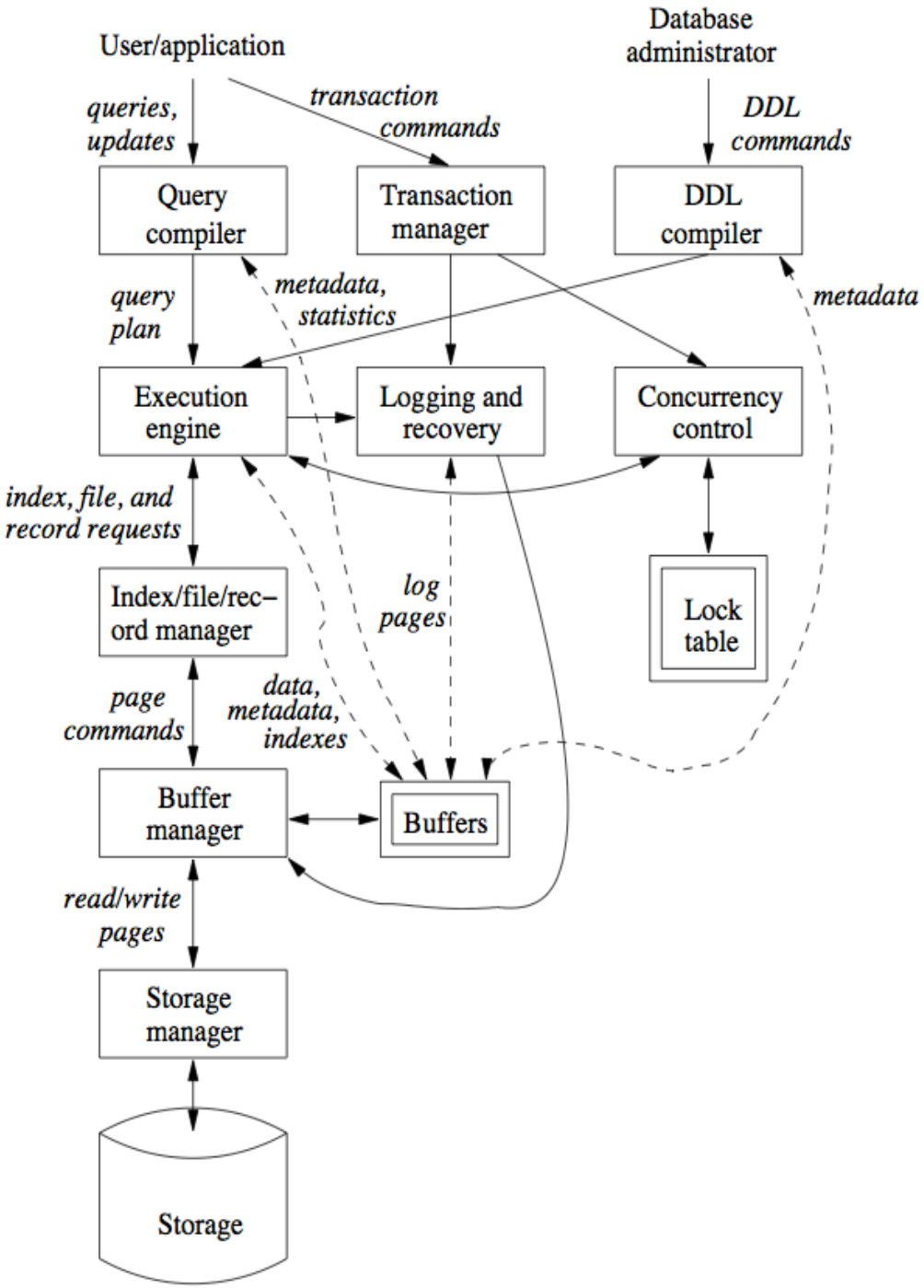
Operator	M required	I/O Cost
δ, γ	√B	3B
∪ , ∩ , -	√(B(R)+B(S))	3(B(R)+B(S))
⋈	√(max(B(R),B(S))	5(B(R)+B(S))
⋈	√(B(R)+B(S))	3(B(R)+B(S))

2 Pass Hash-Based Summary

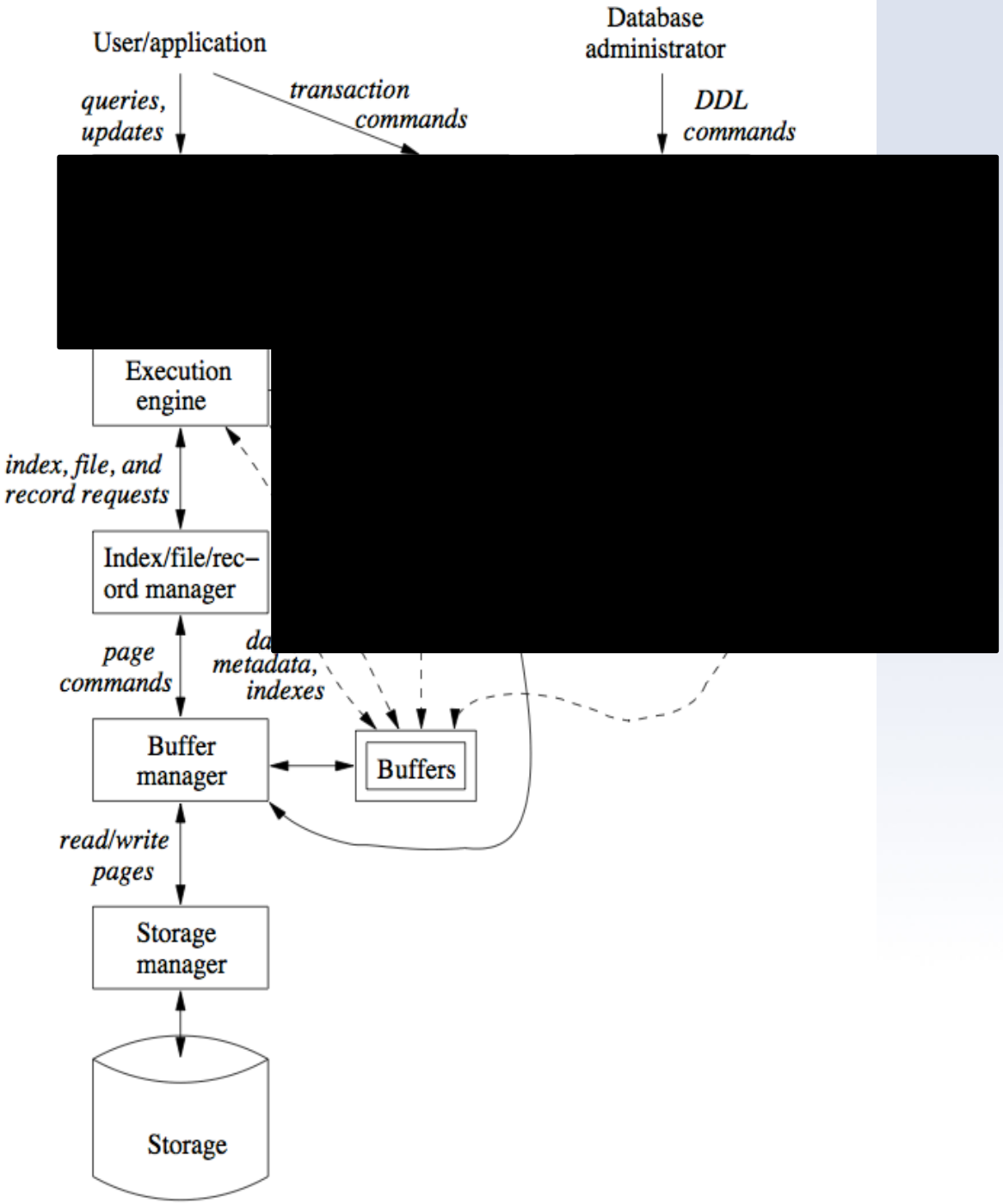
Operator	M required	I/O Cost
δ, γ	√B	3B
∪ , ∩ , -	√(B(S))	3(B(R)+B(S))
⋈	√(B(S))	3(B(R)+B(S))
⋈	√(B(S))	(3-2M/B(S))x (B(R)+B(S))

Practical Example

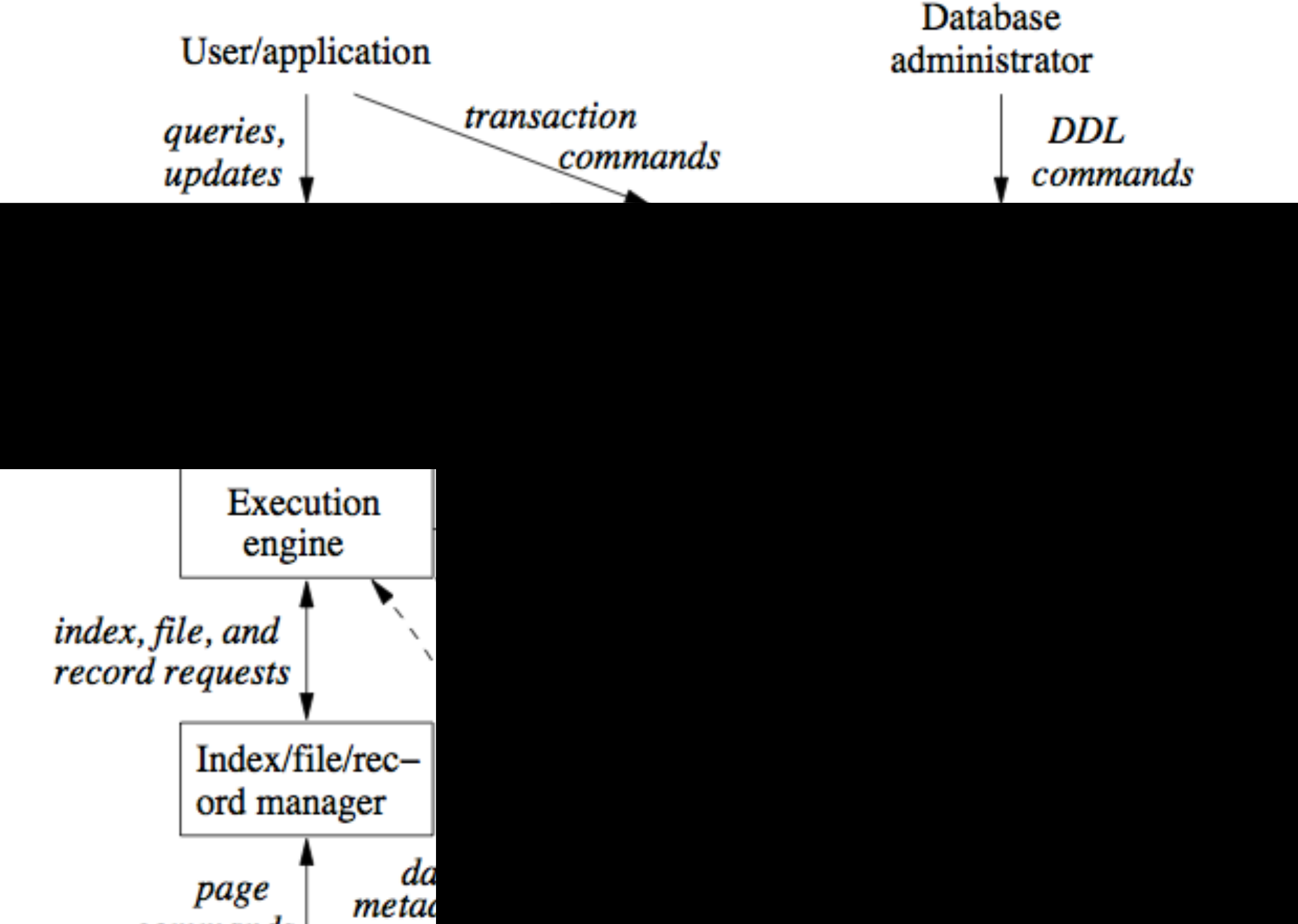
- On a system with 8GB=2^{33} of RAM
  - we can sort/join/deduplicate 2^{66}
  - bigger than an exabyte of data!
- On a system with 256GB=2^{38}
  - we can sort/join/deduplicate 2^{76} bytes of data
  - almost a yottabyte!

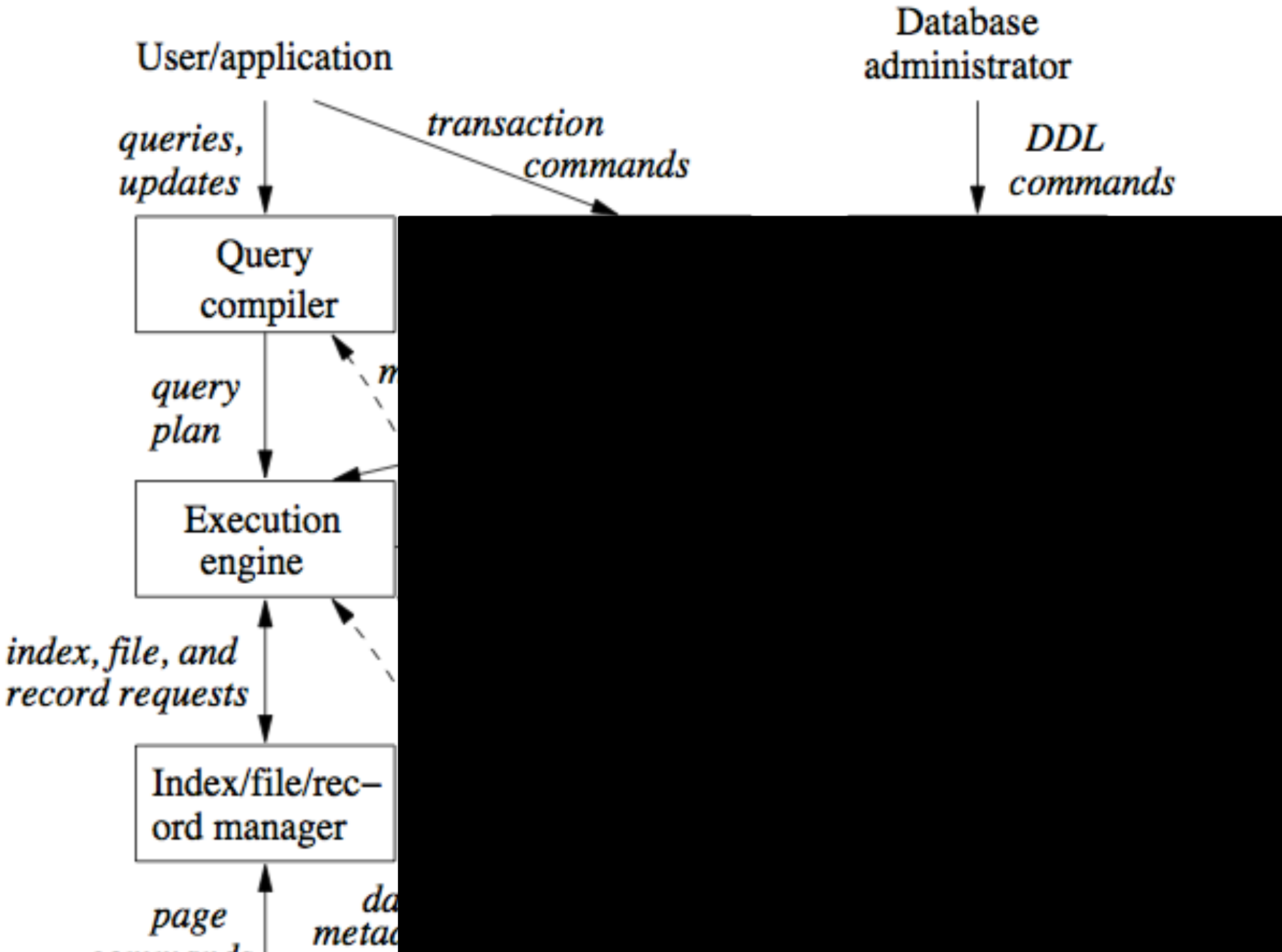


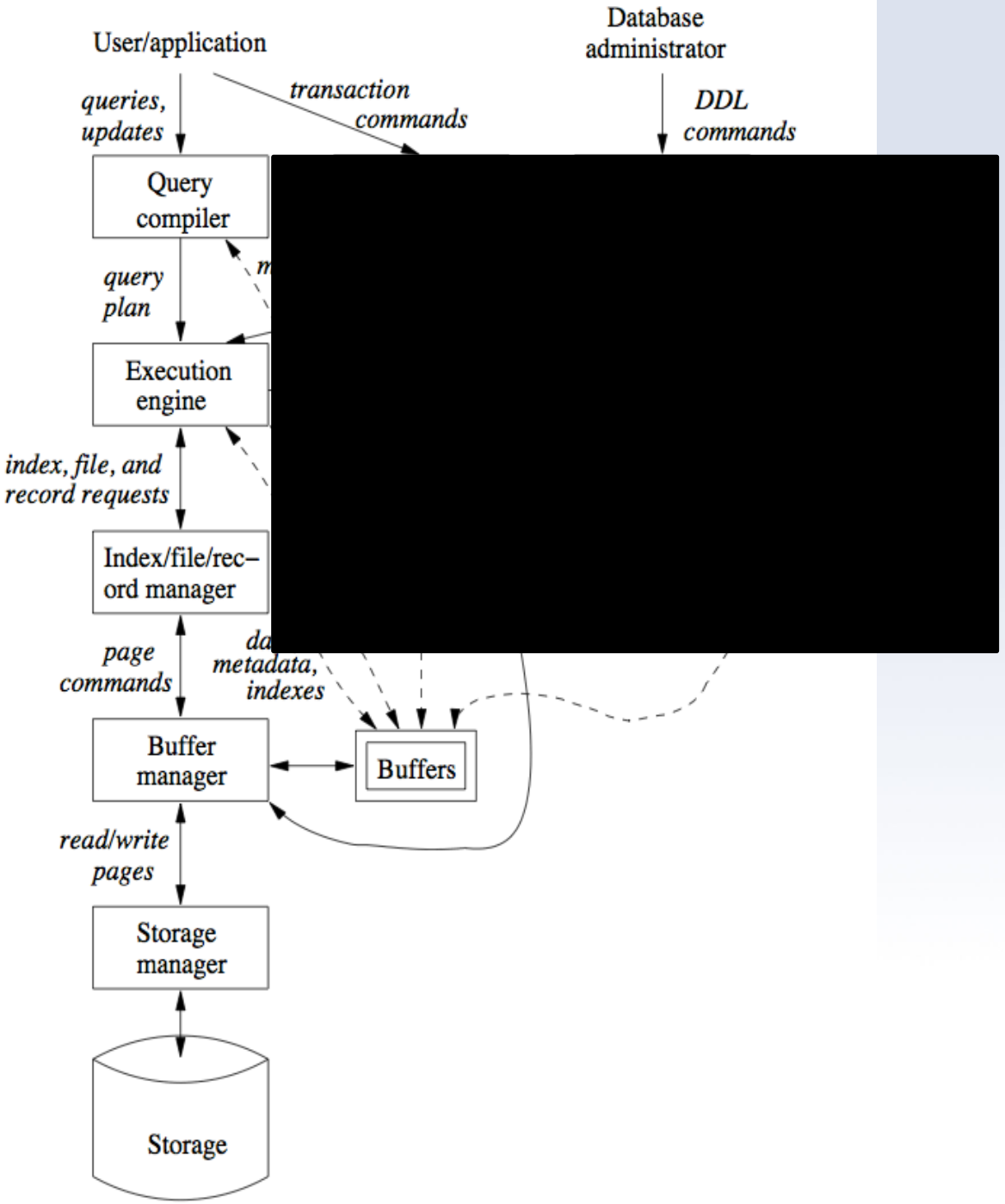
illinois.edu



illinois.edu


User/application
Database administrator
queries, updates
transaction commands
DDL commands
Execution engine
index, file, and record requests
Index/file/rec–ord manager
page


User/application
Database administrator
queries, updates
transaction commands
DDL commands
Query compiler
query plan
Execution engine
index, file, and record requests
Index/file/rec–ord manager
page



illinois.edu

Query Processing

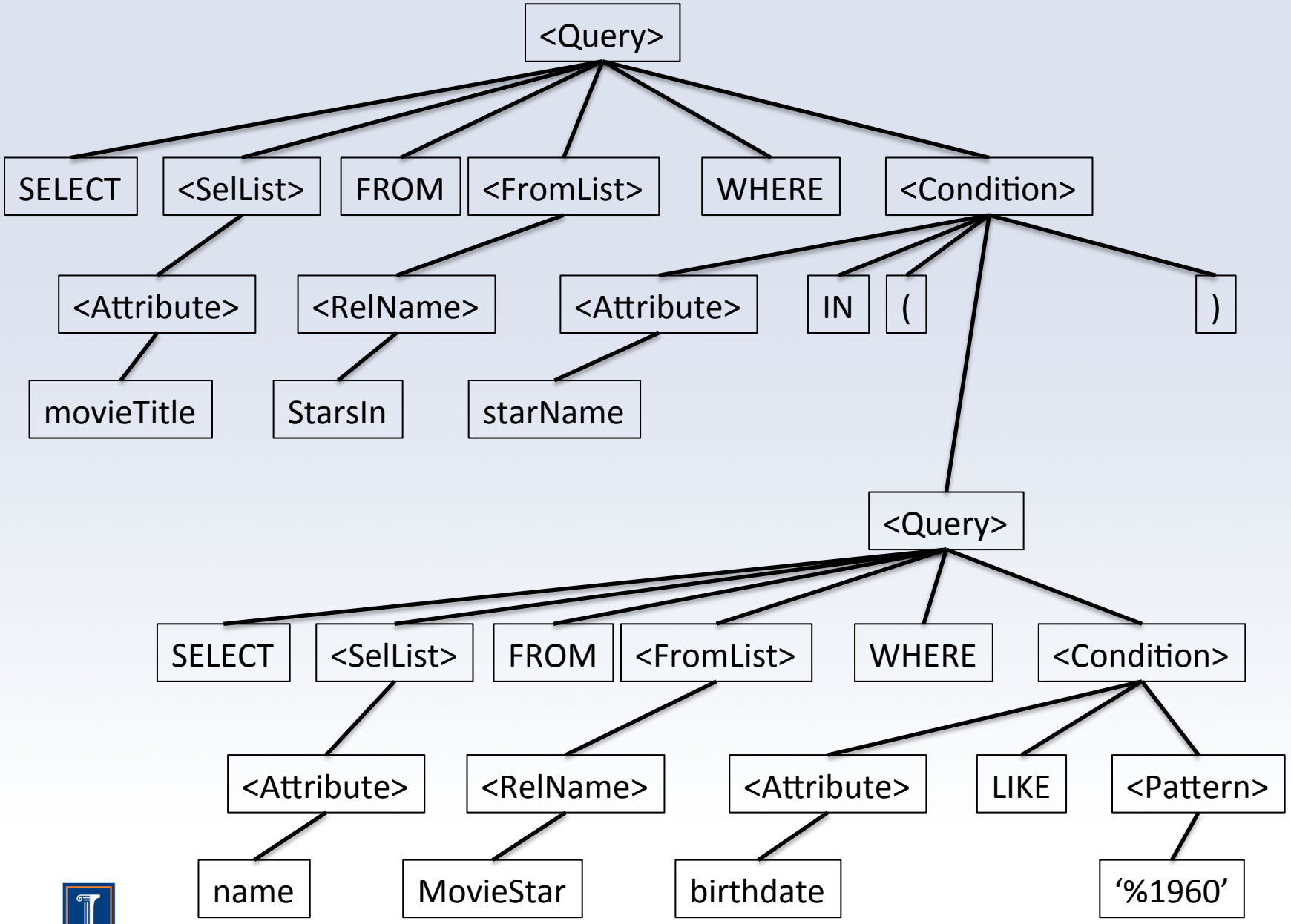
- Can be divided into three steps:
  1. Parse the query into a parse tree
  2. Transform parse tree into logical query plan
  3. Transform logical query plan into physical query plan

Parsing

- Result represented as a tree
- Nodes are either:
  - ***atoms***: keywords, names, attributes, constants, parentheses, operators
  - ***syntactic categories***: names of subparts of a query
    - represented like this: <Category>

Parsing

```
SELECT movieTitle
FROM StarsIn
WHERE starName IN(
   SELECT name
   FROM MovieStar
   WHERE birthdate LIKE '%1960'
);
```


<Query>
SELECT
<SelList>
FROM
<FromList>
WHERE
<Condition>
<Attribute>
<RelName>
<Attribute>
IN
(
)
movieTitle
StarsIn
starName
<Query>
SELECT
<SelList>
FROM
<FromList>
WHERE
<Condition>
<Attribute>
<RelName>
<Attribute>
LIKE
<Pattern>
name
MovieStar
birthdate
'%1960'

Rules

- Rules:
  - consist of atoms and categories
  - allow us to substitute one thing for another
  - Example:

```
<Query>::= SELECT <SelList> FROM
<FromList> WHERE <Condition>
```

Partial Grammar for SQL

```
<Query>::= SELECT <SelList> FROM
   <FromList> WHERE <Condition>

<SelList>::=<Attribute>, <SelList>
<SelList>::=<Attribute>

<FromList>::=<Relation>, <FromList>
<FromList>::=<Relation>
```

Partial Grammar for SQL

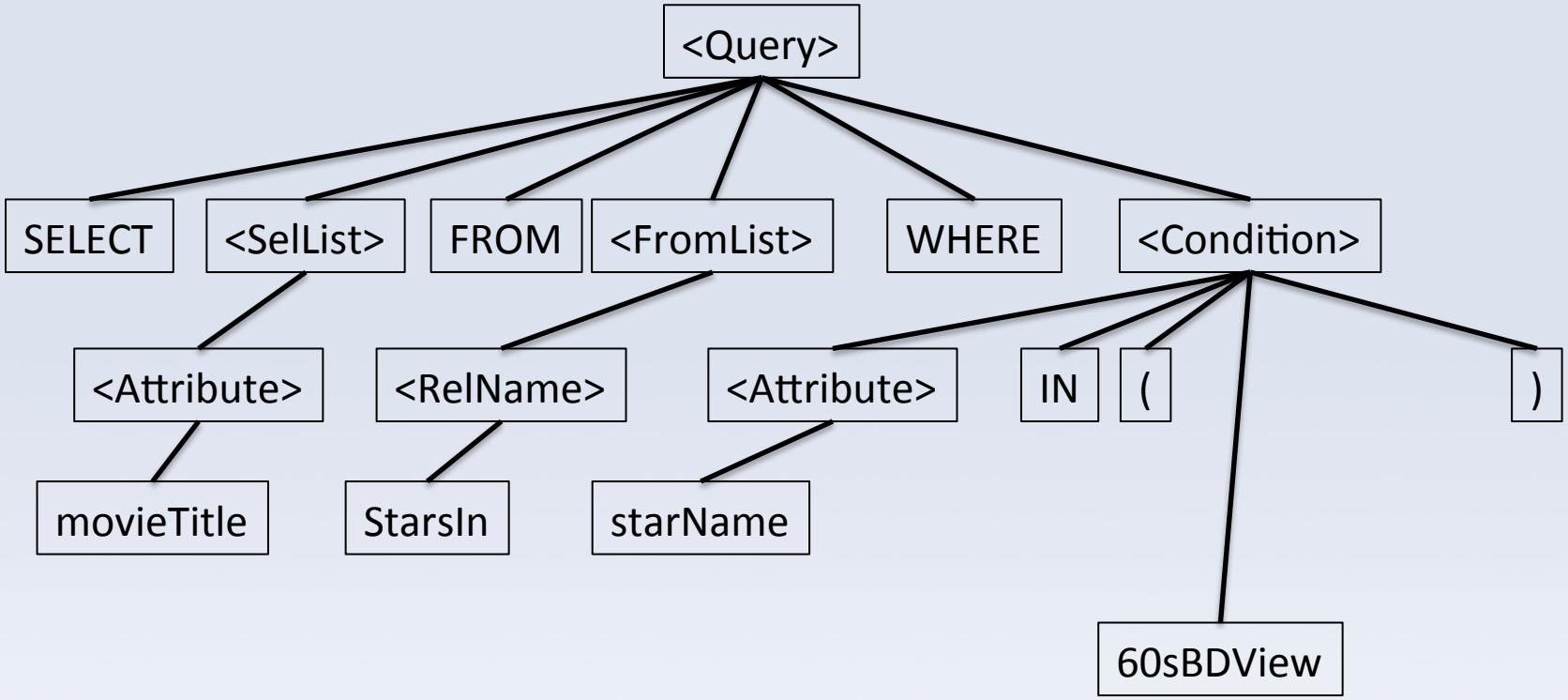
```
<Query>::= SELECT <SelList> FROM
   <FromList> WHERE <Condition>

<Condition>::= <Attribute>=<Attribute>
<Condition>::= <Attribute> LIKE <Pattern>
<Condition>::= <Condition> AND <Condition>
<Condition>::= <Attribute> IN (<Query>)
```

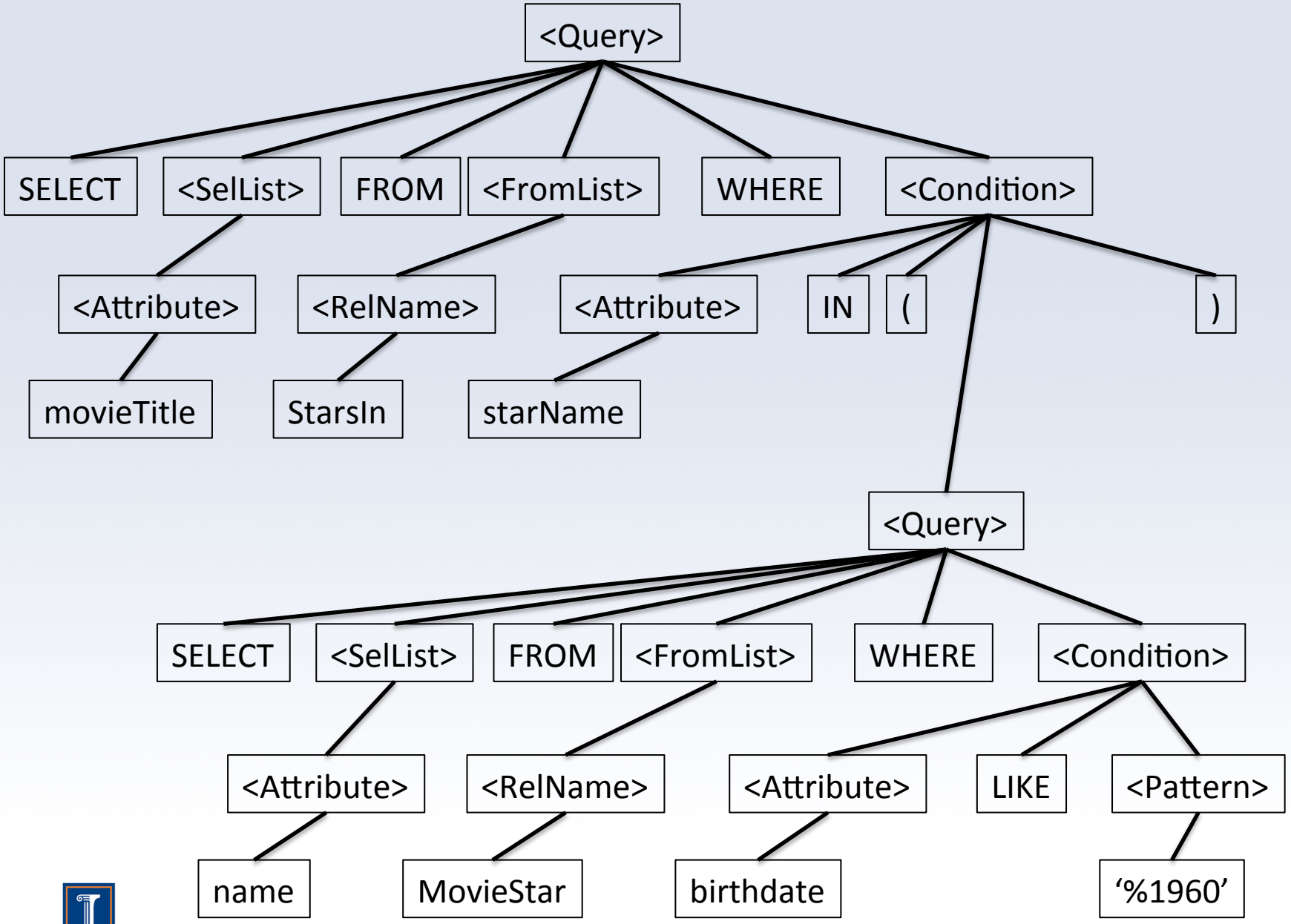
Look familiar?

- CS373 or CS421?
- It's a CFG in BNF
- Entire grammar available here:
  - http://savage.net.au/SQL/
- We can find the parse tree using CYK, Earley parsing, etc.

Preprocessor

- Parser:
  - checks syntax while building tree
- Preprocessor checks ***semantics***
  - Are relations in our database schema?
  - Are the attributes correct for the schema?
  - Are the types correct?
- Also substitutes subqueries for views


<Query>
SELECT
<SelList>
FROM
<FromList>
WHERE
<Condition>
<Attribute>
<RelName>
<Attribute>
IN
(
)
movieTitle
StarsIn
starName
60sBDView


<Query>
SELECT
<SelList>
FROM
<FromList>
WHERE
<Condition>
<Attribute>
<RelName>
<Attribute>
IN
(
)
movieTitle
StarsIn
starName
<Query>
SELECT
<SelList>
FROM
<FromList>
WHERE
<Condition>
<Attribute>
<RelName>
<Attribute>
LIKE
<Pattern>
name
MovieStar
birthdate
'%1960'

Preprocessor

- You wrote a preprocessor in MP1
  - "is valid" methods checked semantics of the input query

Parse Tree to Logical Query Plan

- All of our algorithms for query execution were based on relational algebra
- Need to convert ***parse tree*** to relational algebra-based ***query plan***

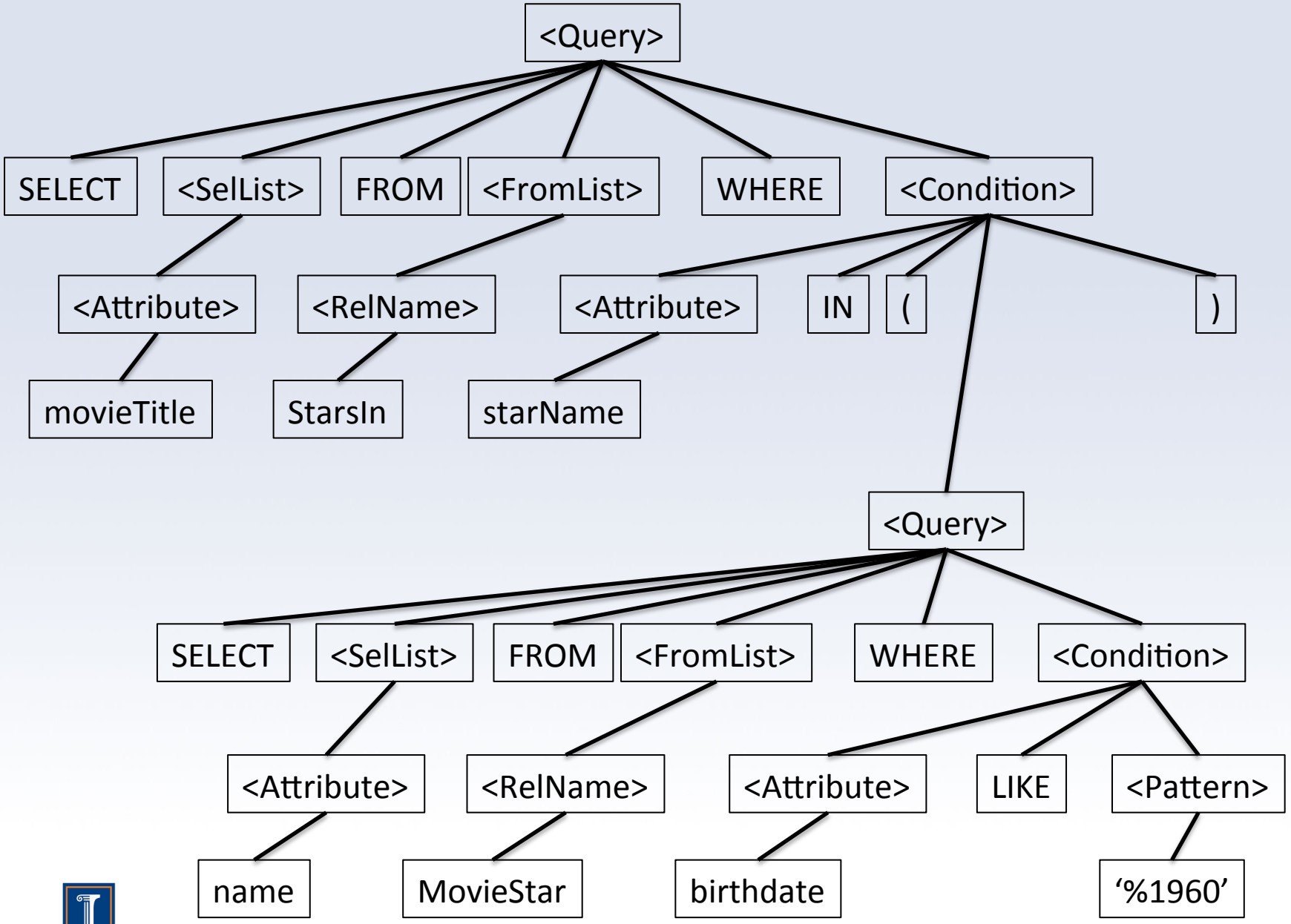
Parse Tree to Logical Query Plan

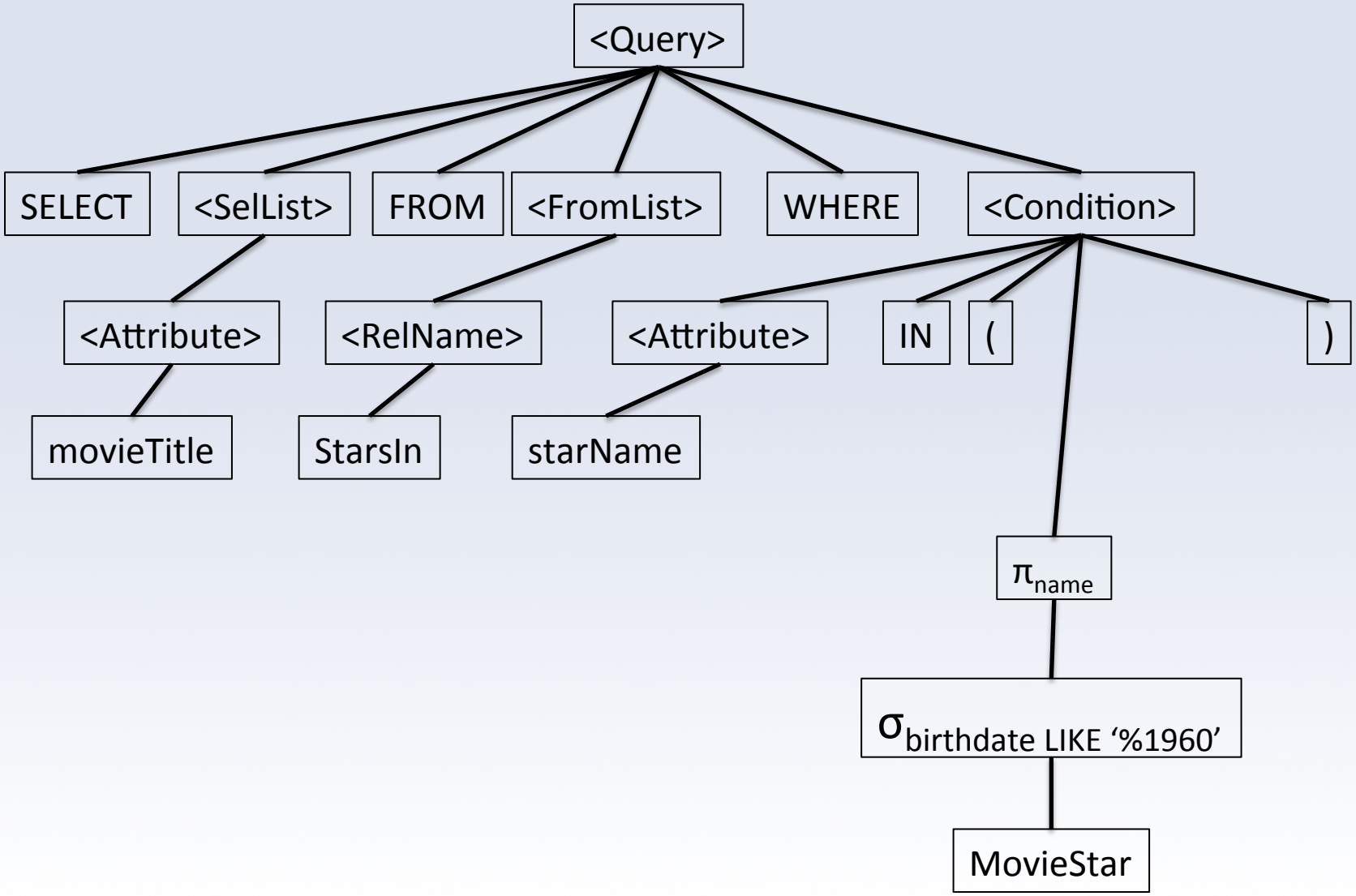
- Replacing <Query> (no subqueries)
  1. take product of all relations in <FromList>
  2. σ_C where C is <Condition>
  3. π_L where L is the attributes in <SelList>

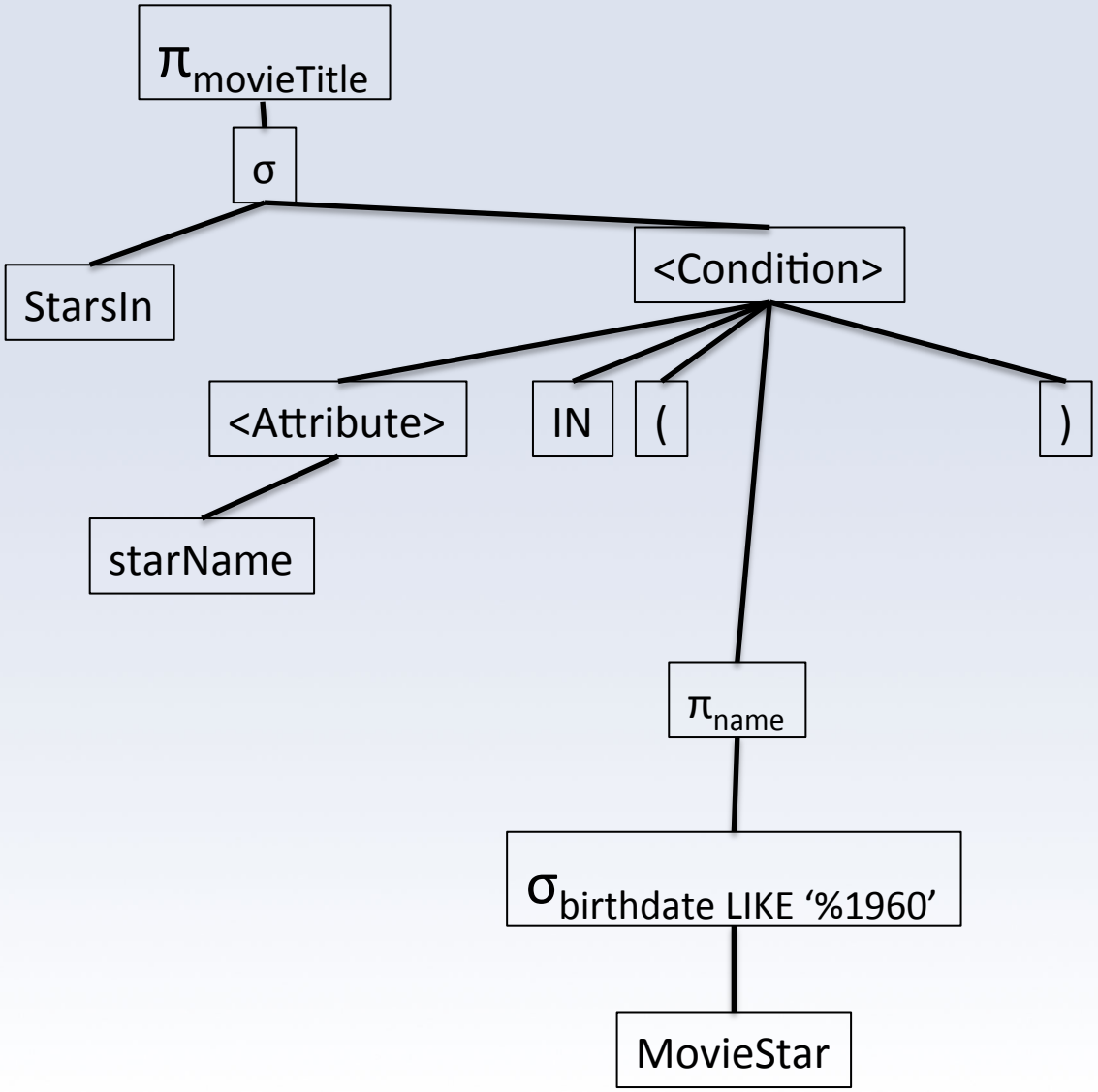
Example

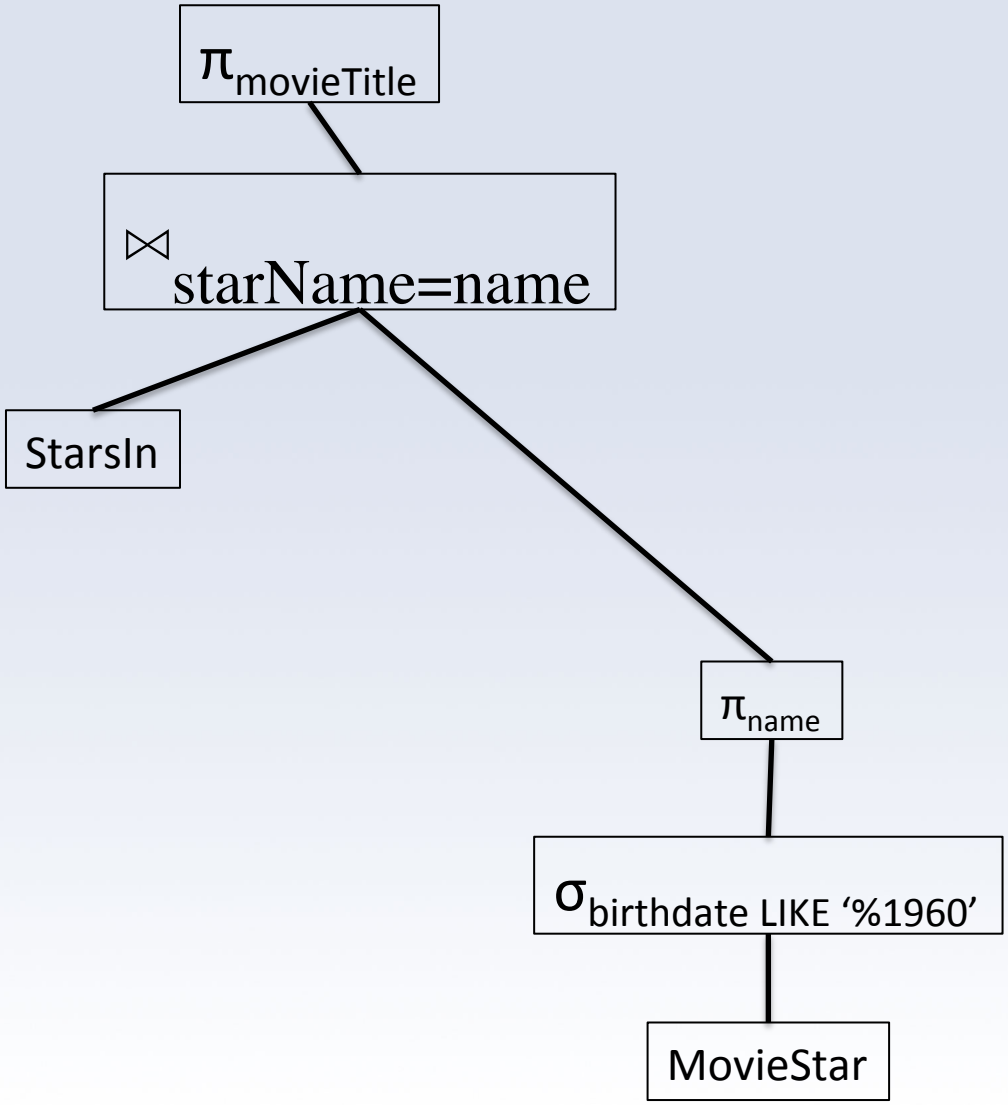
```
SELECT a,b,c
FROM X,Y,Z
WHERE d<e AND f=g
```

$$\pi_{a,b,c}(\sigma_{d<e \text{ AND } f=g}(X \times Y \times Z))$$


<Query>
SELECT
<SelList>
FROM
<FromList>
WHERE
<Condition>
<Attribute>
<RelName>
<Attribute>
IN
(
)
movieTitle
StarsIn
starName
<Query>
SELECT
<SelList>
FROM
<FromList>
WHERE
<Condition>
<Attribute>
<RelName>
<Attribute>
LIKE
<Pattern>
name
MovieStar
birthdate
'%1960'


<Query>
SELECT
<SelList>
FROM
<FromList>
WHERE
<Condition>
<Attribute>
<RelName>
<Attribute>
IN
(
)
movieTitle
StarsIn
starName
π_{name}
$\sigma_{birthdate\ LIKE\ '\%1960'}$
MovieStar


πmovieTitle
σ
StarsIn
<Condition>
<Attribute>
IN
(
)
starName
πname
σbirthdate LIKE '%1960'
MovieStar


πmovieTitle
⋈starName=name
StarsIn
πname
σbirthdate LIKE '%1960'
MovieStar

Parse Tree to Logical Query Plan

- Don't worry about algorithms for:
  - building parse trees
  - converting to query plans
- Subtle hint:
  - You should be able to convert SQL to relational algebra on the final
  - Think about methods for converting subqueries like IN, ALL, etc.

Refining Logical Query Plan

- One query can be expressed many ways
- Some more efficient than others
- Try to pick the best

Query Optimization

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

Rewrite Rules

- Algebraic laws for relational algebra
- Similar to $2(x+y)=2x+2y$

Commutative/Associative Laws

$$R \times S = S \times R, (R \times S) \times T = R \times (S \times T)$$
$$R \bowtie S = S \bowtie R, (R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$$
$$R \cup S = S \cup R, (R \cup S) \cup T = R \cup (S \cup T)$$
$$R \cap S = S \cap R, (R \cap S) \cap T = R \cap (S \cap T)$$
$$R \bowtie (S \cup T) = (R \bowtie S) \cup (R \bowtie T)$$

Selection Laws

$$\sigma_{C_1 \text{ AND } C_2}(R) = \sigma_{C_1}\left(\sigma_{C_2}(R)\right) = \sigma_{C_2}\left(\sigma_{C_1}(R)\right)$$

$$\sigma_{C_1 \text{ OR } C_2}(R) = \sigma_{C_1}(R) \cup \sigma_{C_2}(R)$$

$$\sigma_C(R \times S) = \sigma_C(R) \times \sigma_C(S) = \sigma_C(R) \times S^*$$

$$\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie \sigma_C(S) = \sigma_C(R) \bowtie S^*$$

$$\sigma_C(R \cap S) = \sigma_C(R) \cap \sigma_C(S) = \sigma_C(R) \cap S$$

$$\sigma_C(R \cup S) = \sigma_C(R) \cup \sigma_C(S)$$

$$\sigma_C(R - S) = \sigma_C(R) - \sigma_C(S) = \sigma_C(R) - S$$

*Valid only when C exclusively involves attributes of R

Example

$$R(A, B, C, D), S(E, F, G)$$

$$\sigma_{F=3}(R \bowtie_{D=E} S)$$

Example

$$R(A, B, C, D), S(E, F, G)$$

$$\sigma_{A=5 \text{ AND } G=9}(R \bowtie_{D=E} S)$$

Projection Laws

- Too many caveats to list
- Easier to summarize:
  - Can push a projection inside any operator that doesn't need the attributes projected out
- Example

$$\pi_{a+b \to x}(R \bowtie S) = \pi_{a+b \to x}(\pi_{a,c}(R) \bowtie \pi_{b,c}(S))$$

Join and Product Laws

$$R \bowtie_C S = \sigma_C (R \times S)$$

$$R \bowtie S = \pi_L (\sigma_C (R \times S))$$

Duplicate Elimination Laws

$$\delta(R \times S) = \delta(R) \times \delta(S)$$
$$\delta(R \bowtie S) = \delta(R) \bowtie \delta(S)$$
$$\delta(\sigma_C(R)) = \sigma_C(\delta(R))$$

Grouping and Aggregation

$$\delta(\gamma_L(R)) = \gamma_L(R)$$

$$\gamma_L(R) = \gamma_L(\pi_M(R))^*$$

$$\gamma_L(R) = \gamma_L(\delta(R))^*$$

*Valid only if $L \subseteq M$

*Valid only for MAX and MIN

Example

$$\gamma_{movieYear,MAX(birthdate)}(\sigma_{name=starName}(MovieStar \times StarsIn))$$

Query Optimization

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

General Heuristics

- Some general heuristics for query optimization:
  1. Push down selections
  2. Push down/add projections
  3. Push/add duplicate eliminations when needed or remove when not needed
  4. Turn products into equijoins

Can we do better?

- Physical properties of data helpful
  - How big are the relations involved?
  - Are they sorted?
  - Are they indexed?
- Related to the physical plan, but we won't see physical plans until Friday

Cost Estimation

- We will find ways of estimating the ***cost*** of a query
- Has to be based on the properties of the underlying relation
- We can't know ***exact*** cost without executing the query, so only an estimation

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

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)

Example

- Given R(a,b,c) where T(R)=10,000
  - a,b are 4 byte integers, c is 100 byte string
  - record header=12 bytes
  - block header=24 bytes
  - block size=1024
- Total record size=120 bytes
- Total records per block=8
- B(R)=1,250

Example

- $S=\pi_{a+b \rightarrow x,c}(R)$
- Total record size=116 bytes
- Total records per block=8
- T(S)=10,000
- B(S)=1,250

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
  - $(1-m_1/n)(1-m_2/n)$ fraction that satisfy both
  - $T(S)\approx n(1-(1-m_1/n)(1-m_2/n))$

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

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

Join Estimation

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

Example

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

Other Operators

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

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

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

$T(R \bowtie S)=5000*2000/100=10000$

$T(\sigma_{a=10}(R \bowtie S))=10000/50=200$

T(U)=min(100,5000)=100

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

Query Optimization

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

Next Time…

- We'll see:
  - how we use these estimates to find a good plan
    - order of joins is especially important
  - how to represent and choose a physical plan
    - other factors to consider (e.g. indexes, sorts)
    - pipelining vs. materialization