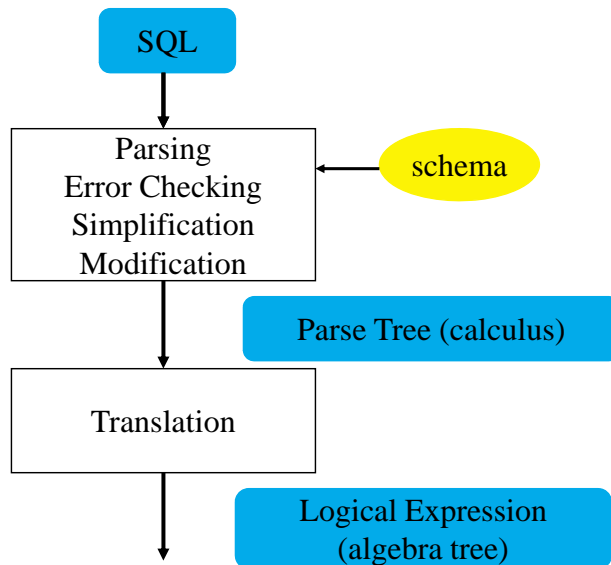
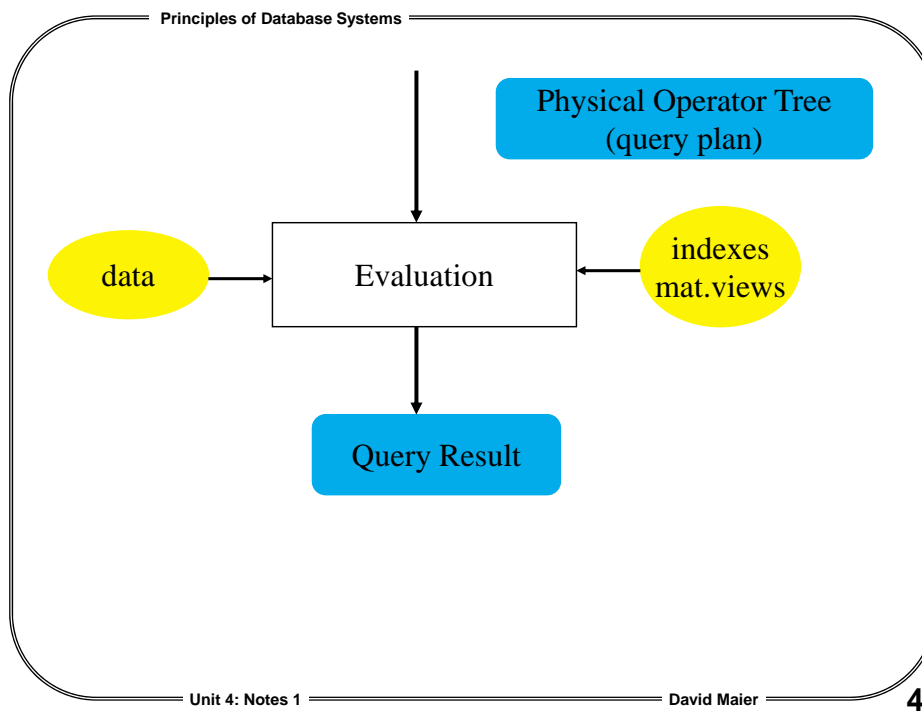
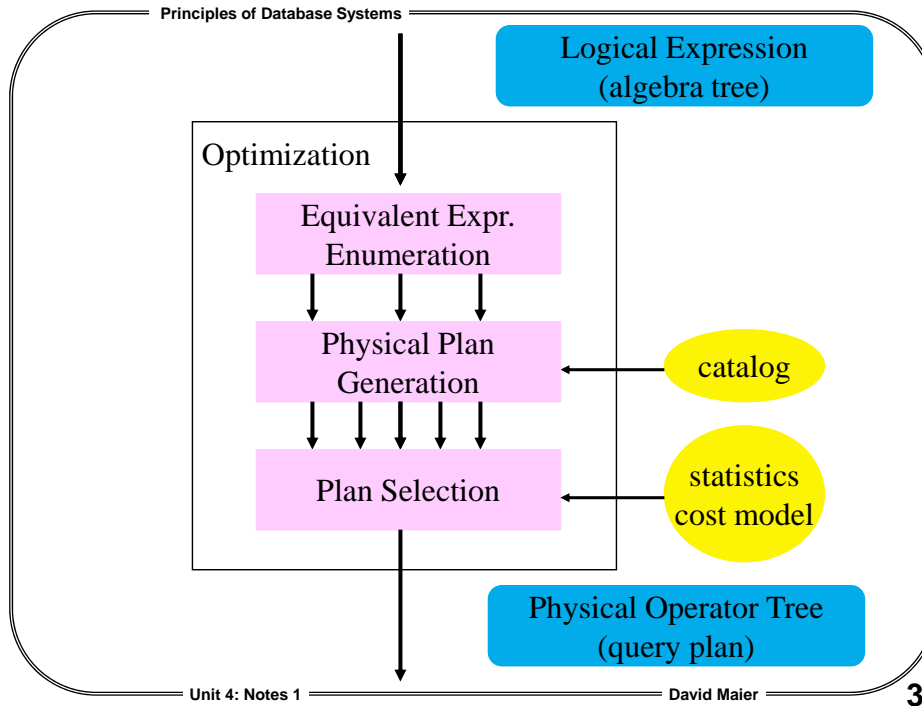


## Unit 3 Relational Algebra and Query Optimization

### Steps in Query Processing



# CS 589, Principles of Database Systems, Unit 3, Notes 1



## Topics

Will focus on role of algebra and algebraic equivalences in query optimization

## Issues

- bags versus sets
- additional operators (e.g., group-by)
- transformations
- logical versus physical ops.
- alternative interpretations of operators
  - size
  - cost

## Simplification

There is some work more easily done in a calculus form

Rewrite of query conditions

```
flights(FLIGHT DATE BUS ECON)
```

```
select f.L, f.D
```

```
from flights as f
```

```
where f.B > 60 and f.B > 40
```

## More Examples

```
select f.L, f.D
from flights as f
where (f.B ≥ 60 and f.E > f.B
and f.E < 30) or f.B < 40
```

```
select f.L, f.D
from flights as f
where f.B = f.E and f.E = 40
and f.B = 40
```

## Rewrite Doesn't Always "Simplify"

```
select f1.L, f2.L
from flights as f1, f2
where f1.B = f2.B
and f1.B = f2.E
```

Some query systems rewrite the condition to:

```
where f1.B = f2.B
and f1.B = f2.E
and f2.B = f2.E
```

Why?

## Tableau Query Minimization

Reduce the number of joins in SPJ queries

Query containment

Query  $p$  is *contained by* query  $q$ ,  $p \subseteq q$ , if for all database instances  $I$ ,  
 $p(I) \subseteq q(I)$

## Tableau as a Query

Have tableau  $T$  (possibly with constants) with *result row*  $w$

answer on  $r(R)$

$$\{\rho(w) \mid \rho(T) \subseteq r\}$$

$\rho$  is a *valuation*: maps variables to constants

(and constants to themselves)

### Tableau Query Example: $q = (T, w)$

r	(F	D	B	E)	T	(F	D	B	E)
t1	176	6Oct	15	160	w1	f1	d1	20	e1
t2	185	6Oct	20	150	w2	f2	d1	b2	e2
t3	192	6Oct	35	174	w	f2		b2	
t4	176	7Oct	20	95					
t5	176	8Oct	17	152					

v	f1	d1	e1	f2	b2	e2
$\rho_1(v)$	176	6Oct	160	185	20	150

So

$$\rho_1(w1) = \langle 176 \ 6Oct \ 20 \ 160 \rangle \notin r$$

$$\rho_1(w2) = \langle 185 \ 6Oct \ 20 \ 150 \rangle = t2 \in r$$

Adds no tuple to  $q(r)$

### Other Valuations

v	f1	d1	e1	f2	b2	e2
$\rho_2(v)$	185	6Oct	150	176	15	160

So

$$\rho_2(w1) = \langle 185 \ 6Oct \ 20 \ 150 \rangle = t2 \in r$$

$$\rho_2(w2) = \langle 176 \ 6Oct \ 15 \ 160 \rangle = t1 \in r$$

Adds  $\rho_2(w) = \langle 176 \ 15 \rangle$  to  $q(r)$

$$\rho_3(w1) = \langle 185 \ 6Oct \ 20 \ 150 \rangle = t2 \in r$$

$$\rho_3(w2) = \langle 192 \ 6Oct \ 35 \ 174 \rangle = t3 \in r$$

Adds  $\rho_3(w) = \langle 192 \ 35 \rangle$  to  $q(r)$

## More Valuations

$\rho_4(w_1) = \langle 185 \text{ 60Oct } 20 \text{ 150} \rangle = t_2 \in r$

$\rho_4(w_2) = \langle 185 \text{ 60Oct } 20 \text{ 150} \rangle = t_2 \in r$

Adds  $\rho_4(w) = \langle 185 \text{ 20} \rangle$  to  $q(r)$

$\rho_5(w_1) = \langle 176 \text{ 70Oct } 20 \text{ 95} \rangle = t_4 \in r$

$\rho_5(w_2) = \langle 176 \text{ 70Oct } 20 \text{ 95} \rangle = t_4 \in r$

Adds  $\rho_5(w) = \langle 176 \text{ 20} \rangle$  to  $q(r)$

$\rho_6(w_1) = ?$

$\rho_6(w_2) = \langle 176 \text{ 80Oct } 17 \text{ 152} \rangle = t_5 \in r$

## Final Result

$q(r)$  (F    B )

176 15

185 20

192 35

176 20

## Substitution

Need something like a valuation that goes from one tableau query to another.

*Substitution*  $\theta$ : maps variables to variables or constants  
(and constants to themselves)

## Mapping Tableau Queries

Interested in substitution that maps one tableau to another

$q_1$	$T_1$	$($	<u>L</u>	<u>A</u>	$)$	$q_2$	$T_2$	$($	<u>L</u>	<u>A</u>	$)$
			<u>f</u>	<u>a</u>					f	a1	
			w1	f	a				f1	a	
									<u>f1</u>	<u>a1</u>	
						w2			f	a	

Want a substitution  $\theta$  where

$$\theta(T_2) \subseteq T_1$$

$$\theta(w_2) = w_1$$

Means  $q_1 \subseteq q_2$

## Example of Containment

Suppose  $t = \langle 115 \ 747 \rangle$  in  $q_1(r)$

Must be  $\rho, \rho(T_1) \subseteq r$  and  $\rho(w_1) = t$

Want to show  $t$  in  $q_2(r)$

Let  $\rho' = \rho \circ \theta$

Have  $\theta(T_2) \subseteq T_1$  so

$\rho(\theta(T_2)) \subseteq \rho(T_1)$  or  $\rho'(T_2) \subseteq \rho(T_1)$

Also know  $\rho(T_1) \subseteq r$

So  $\rho'(T_2) \subseteq \rho(T_1) \subseteq r$

What is  $\rho'(w_2)$ ?

$\rho'(w_2) = \rho(\theta(w_2)) = \rho(w_1) = t$

So  $t$  in  $q_2(r)$ !

## Homomorphism Theorem

Given tableau queries  $q_1 = \langle T_1, w_1 \rangle$  and  $q_2 = \langle T_2, w_2 \rangle$ ;

$q_1 \subseteq q_2$  if and only if there is a substitution  $\theta$  such that  $T_1 \supseteq \theta(T_2)$  and  $w_1 = \theta(w_2)$ .

Such a substitution is called a *query homomorphism*.

## What's Containment Good For?

Containment itself isn't that interesting.

Equivalence:  $q1 \equiv q2$  if  $q1(I) = q2(I)$  for all instances  $I$ .

Observation:  $q1 \equiv q2$  if and only if  $q1 \subseteq q2$  and  $q2 \subseteq q1$ .

## Query Minimization

The real use of the Homomorphism Theorem is in finding a *minimum equivalent* query.

Given tableau query  $q$ , want a query  $q'$  with fewest rows such that  $q \equiv q'$ .

**Remarkable fact:** You can always get to a minimum equivalent query by removing rows.

(Maybe 0 rows if already minimum.)

### Example

$q_1$ T1 ( <u>A</u> B C) w1 a2 b1 c w2 a b1 c1 w3 a1 b c1 w4 a b2 c2 w5 <u>a2 b2 c</u> wr a b c	$q_2$ T2 (A B C) w1 a2 b1 c w2 a b1 c1 w3 <u>a1 b c1</u> wr a b c
-------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------

- $q_2 \supseteq q_1$  (why?)
- $q_1 \supseteq q_2$

$v$	a	a1	a2	b	b1	b2	c	c1	c2
$\theta(v)$	a	a1	a2	b	b1	<b>b1</b>	c	c1	<b>c1</b>

- so  $q_1 \equiv q_2$

### Query Translation

What are some features of SQL that are not easily expressed using the relational algebra operators introduced so far?

## Duplicates

Table is a *bag* of tuples, rather than a set.

Need operators that keep duplicates and an operator to remove them

- $\cup^+$  union-all: don't remove duplicates
- $\pi^+$  project-all: don't remove duplicates
- DE duplicate-elim: convert bag to set

Beware: Some authors assume duplicate-preserving ops without telling you.

## Example

flights	<u>FLIGHT</u>	<u>DATE</u>	<u>BUS</u>	<u>ECON</u>
	115	1 Nov	10	80
	115	2 Nov	15	70
	115	3 Nov	10	60
	116	1 Nov	20	75
	116	2 Nov	15	90

$\pi_{BUS}^+$ (flights)	<u>BUS</u>	$\pi_{BUS}$ (flights)	<u>BUS</u>
	10		10
	15		15
	10		20
	20		
	15		

$$DE(\pi_{BUS}^+(\text{flights})) = \pi_{BUS}(\text{flights})$$

## Group-by

Scalar: one row summary of whole relation:  $G_1$

Vector: one row per group:  $G$

$G_1[F]$  -  $F$  is aggregate functions

```
select sum(B) as SB, sum(E) as SE
from flights
```

$G_1[\text{sum}(B) \text{ as } SB, \text{sum}(E) \text{ as } SE](\text{flights})$

<u>SB</u>	<u>SE</u>
70	375

## Vector Group-by

$G[A; F]$  -  $A$  is grouping attributes,  $F$  is aggregate functions

```
select L, sum(B) as SB, sum(E)
as SE
```

```
from flights
```

```
group by L
```

$G[L; \text{sum}(B) \text{ as } SB, \text{sum}(E) \text{ as } SE](\text{flights})$

<u>L</u>	<u>SB</u>	<u>SE</u>
115	35	210
116	35	165

Other aggregates: count, avg, min, max

## Join Variants

Outer-join: used when you want to preserve all tuples from one input in the result (even if some don't connect with tuples in the other input)

- LOJ: Left outer-join, include all of left input
- ROJ: Right outer-join, include all of right input
- FOJ: Full outer-join, include all of both inputs

Need to pad with nulls

## Outer-join Example

capacity(AIRCRAFT MAXBUS MAXECON)

707	25	100
737	30	120
747	80	300
757	50	250

uses(FLIGHT AIRCRAFT)

115	707
116	737
80	747
81	737

## [Natural] Left Outer-Join

capacity LOJ[A=A] uses

(AIRCRAFT MAXBUS MAXECON FLIGHT)			
707	25	100	115
737	30	120	116
737	30	120	81
747	80	300	80
757	50	250	⊥

## Semi-Join

Find the tuples in one relation that join with those in another.

Have  $r(R)$ ,  $s(S)$ , common attributes  $X$

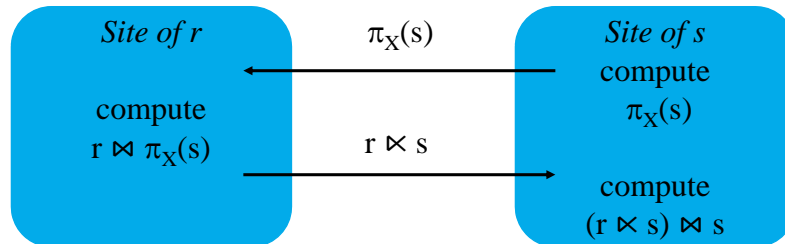
$$r \bowtie s = \{t \mid t \text{ in } r \text{ and } \exists u \text{ in } s, t[X] = u[X]\}$$

$$r \bowtie s = \pi_R(r \bowtie s)$$

capacity  $\bowtie$  uses

Note:  $r \bowtie s = r \bowtie \pi_X(s)$

## Semi-Join is Useful in Distributed Query Processing

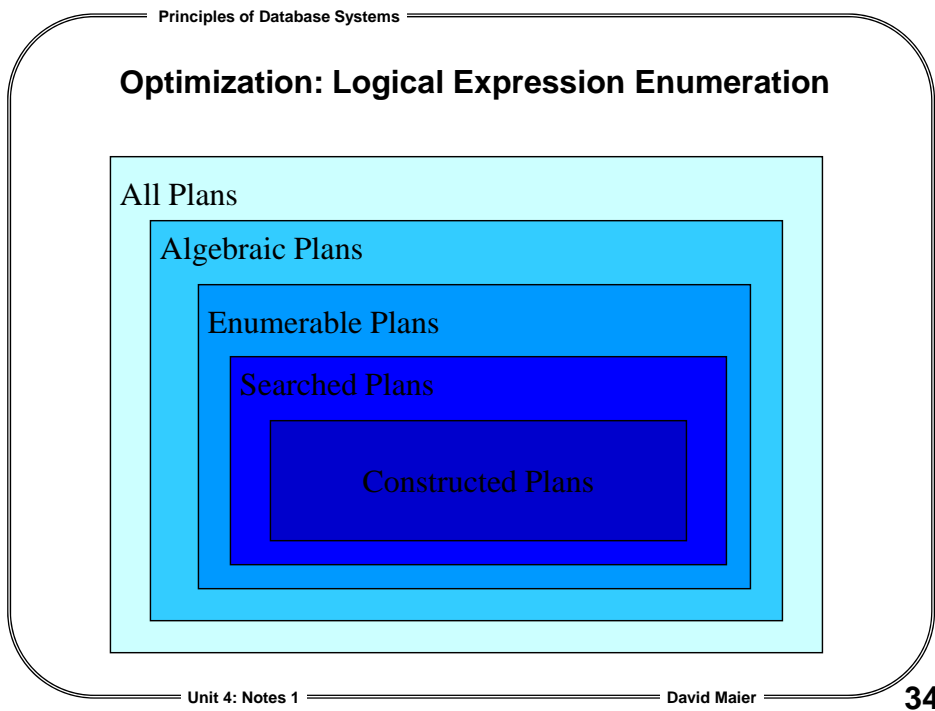
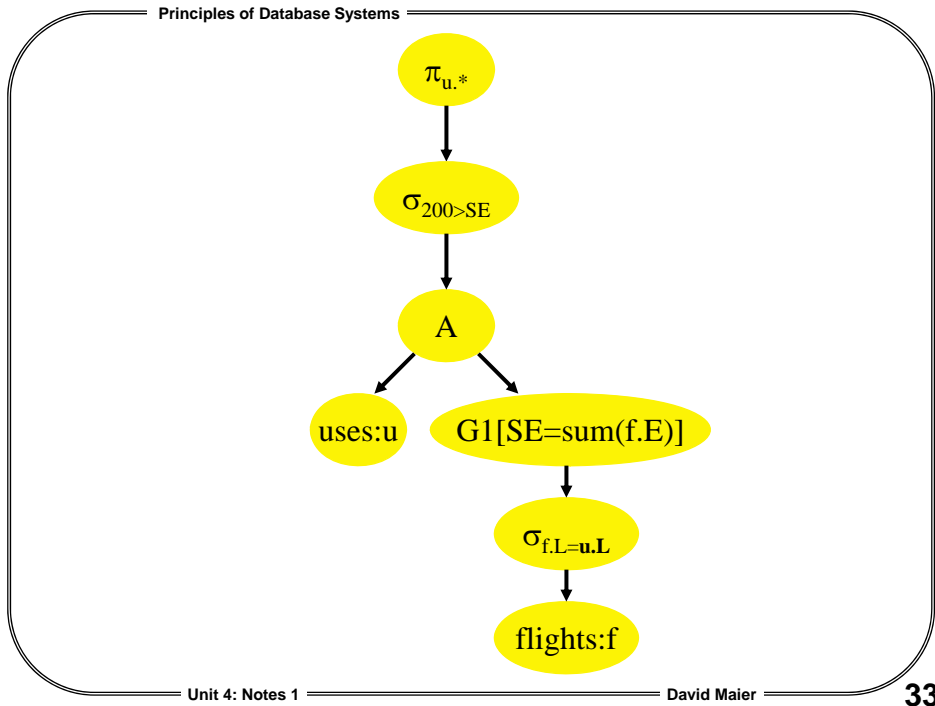


$$(r \bowtie s) \bowtie s = r \bowtie s$$

## Apply (Map, Do-all)

```

r A e(x) = {t++u | t in r and u in e(t)}
select u.*
from uses as u
where 200 >
  (select sum(f.E)
   from flights as f
   where u.L = f.L)
    
```



## Search

Search can be limited by

- *Plan shape*: no Cartesian products, bushy joins
- *Heuristics*: abandon subplans with cost over 80% of current best plan
- *Time*: stop when optimization time = estimated execution time

## Optimality Principle

Subplans of an optimal plan are themselves optimal

If  $((r * s) * u) * v$  is best for joining  $r, s, u$  &  $v$ ,  
then  $(r * s) * u$  is best for joining  $r, s$  &  $u$

Dynamic Programming for join ordering  
(System R)

- Find best order for every join of  $k-1$  or fewer tables
- Use those results to find best order for  $k$  tables

## Example Metrics

$$\text{size}(E * F) = \text{size}(E) * \text{size}(F) / 10$$

$$\text{cost}(E * F) = \text{size}(E * F) + \text{cost}(E) + \text{cost}(F)$$

$$\text{cost}(\text{scan}(r)) = 0$$

## 1-joins and 2-joins

1-joins relation: **size**  
cost

r:10	s:100	u:50	v:10
0	0	0	0

2-joins

(r, s):100	(r, u):50	(r, v):10
100	50	10

(s, u):500	(s, v):100	(u, v):50
500	100	50

## 3-joins

$(r, s, u) : 500$	$(r, s, v) : 100$
$(rs)u + 100 + 0 = 600$	$(rs)v + 100 + 0 = 200$
$* (ru)s + 50 + 0 = 550$	$* (rv)s + 10 + 0 = 110$
$(su)r + 500 + 0 = 1000$	$(sv)r + 100 + 0 = 200$
$(r, u, v) : 50$	$(s, u, v) : 500$
$(ru)v + 50 + 0 = 100$	$(su)v + 500 + 0 = 1000$
$* (rv)u + 10 + 0 = 60$	$(sv)u + 100 + 0 = 600$
$(uv)r + 50 + 0 = 100$	$* (uv)s + 50 + 0 = 550$

## 4-join

$(r, s, u, v) : 500$

$(rsu)v + 550 + 0 = 1050$	
$* (ruv)s + 60 + 0 = 560$	
$(rsv)u + 110 + 0 = 610$	
$(suv)r + 550 + 0 = 1050$	
$(rs)(uv) + 100 + 50 = 650$	
$(ru)(sv) + 50 + 100 = 650$	
$(rv)(su) + 10 + 500 = 1010$	