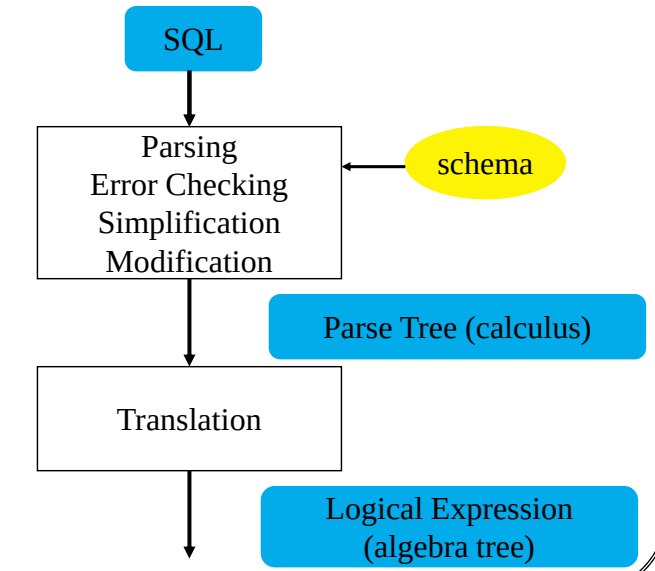
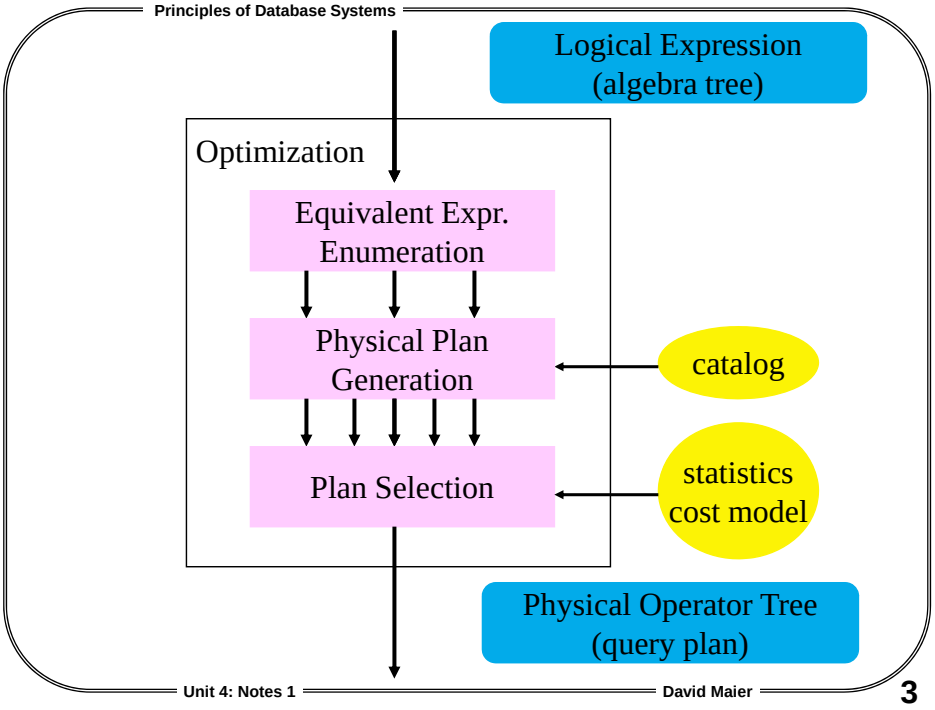
CS 589, Principles of Database Systems, Unit 3, Notes 1

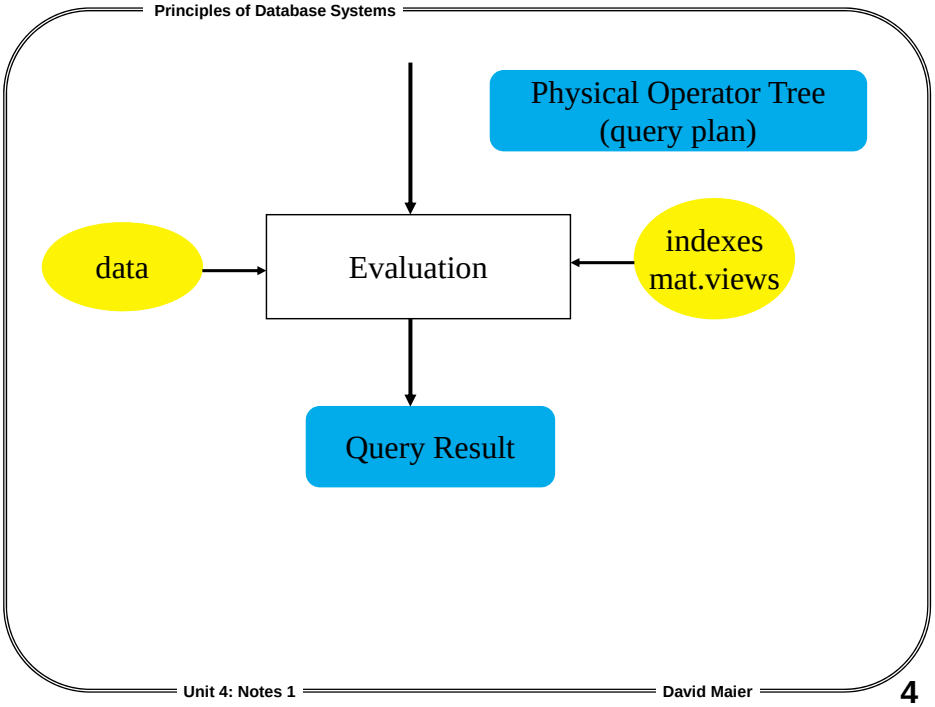
Unit 3
Relational Algebra and
Query Optimization

Steps in Query Processing



David Maier & Lois Delcambre





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
```

} select

Why?

Tableau Query Minimization

Reduce the number of joins in SPJ queries

$\sigma, \pi, \bowtie$

Query containment

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

$q(I) \subseteq p(I)$ $\quad p \equiv q$

Tableau as a Query

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

answer on r(R)

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

ρ is a *valuation*: maps variables to constants

(and constants to themselves)

Tableau Query Example: q = (T, w)

r	F	D	B	E
t1	176	6Oct	15	160
t2	185	6Oct	20	150
t3	192	6Oct	35	174
t4	176	7Oct	20	95
t5	176	8Oct	17	152

T	F	D	B	E
w1	f1	d1	20	e1
w2	f2	d1	b2	e2
w	f2		b2	

v	f1	d1	e1	f2	b2	e2
ρ1(v)	176	6Oct	160	185	20	150

So

ρ1(w1) = <176 6Oct 20 160> ∉ r

ρ1(w2) = <185 6Oct 20 150> = t2 ∈ r

Adds no tuple to q(r)

Other Valuations

v	f1	d1	e1	f2	b2	e2
ρ2(v)	185	6Oct	150	176	15	160

So

ρ2(w1) = <185 6Oct 20 150> = t2 ∈ r

ρ2(w2) = <176 6Oct 15 160> = t1 ∈ r

Adds ρ2(w)= <176 15> to q(r)

ρ3(w1) = <185 6Oct 20 150> = t2 ∈ r

ρ3(w2) = <192 6Oct 35 174> = t3 ∈ r

Adds ρ3(w)= <192 35> to q(r)

More Valuations

ρ4(w1) = <185 6Oct 20 150> = t2 ∈ r
ρ4(w2) = <185 6Oct 20 150> = t2 ∈ r
Adds ρ4(w)= <185 20> to q(r)

ρ5(w1) = <176 7Oct 20 95> = t4 ∈ r
ρ5(w2) = <176 7Oct 20 95> = t4 ∈ r
Adds ρ5(w)= <176 20> to q(r)

ρ6(w1) = <176 8Oct 20 > ∉ r
ρ6(w2) = <176 8Oct 17 152> = t5 ∈ r

Final Result

q(r)

F	B
176	15
185	20
192	35
176	20

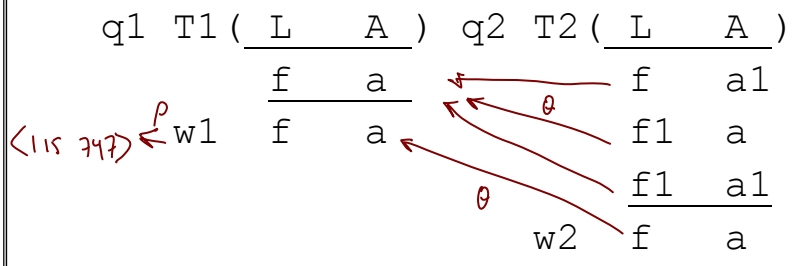
Substitution

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

Substitution θ: maps variables to variables or constants
(and constants to themselves)

Mapping Tableau Queries

Interested in substitution that maps one tableau to another



Want a substitution θ where

θ(T2) ⊆ T1
θ(w2) = w1

θ | f a1 f1 a
  | f a f a

Means q1 ⊆ q2

Example of Containment

Suppose t=<115 747> in q1(r)

Must be ρ, ρ(T1) ⊆ r and ρ(w1) = t

Want to show t in q2(r)

- Let ρ′=ρ°θ
- Have θ(T2) ⊆ T1 so
- ρ(θ(T2)) ⊆ ρ(T1) or ρ′(T2) ⊆ ρ(T1)
- Also know ρ(T1) ⊆ r
- So ρ′(T2) ⊆ ρ(T1) ⊆ r

What is ρ′(w2)?

- ρ′(w2) = ρ(θ(w2)) = ρ(w1) = t

So t in q2(r)!

Homomorphism Theorem

Given tableau queries q1 = <T1, w1> and q2 = <T2, w2>;

q1 ⊆ q2 if and only if there is a substitution θ such that T1 ⊇ θ(T2) and w1 = θ(w2).

- 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.

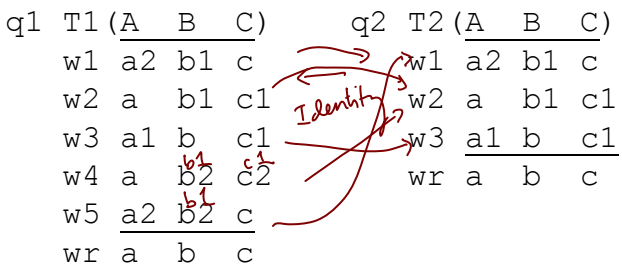
Given tableau query q, want a query q' with fewest rows such that $q \equiv q'$.

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

(Maybe 0 rows if already minimum.)

David Maier & Lois Delcambre

Example



- q2 ⊇ q1 (why?)
- q1 ⊇ q2 use the following substitution

v	a	a1	a2	b	b1	b2	c	c1	c2
θ(v)	a	a1	a2	b	b1	**b1**	c	c1	**c1**

- so q1 ≡ q2

Query Translation

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

Duplicates

multi-set

duplicates allowed

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
- π^+ 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(FLIGHT DATE BUS ECON)

FLIGHT	DATE	BUS	ECON
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

π^+_{BUS}(flights)(BUS)

BUS
10
15
10
20
15

π_{BUS}(flights)(BUS)

BUS
10
15
20

DE(π^+_{BUS}(flights)) = π_{BUS}(flights)

David Maier & Lois Delcambre

Group-by

Scalar: one row summary of whole relation: *G1*

Vector: one row per group: *G*

G1[F] - F is aggregate functions

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

```
G1[sum(B) as SB, sum(E) as SE](flights)
  SB   SE
  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;sum(B) as SB, sum(E) as SE](flights)
  L  SB  SE
 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)

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

in capacity & uses

uses(FLIGHT AIRCRAFT)

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

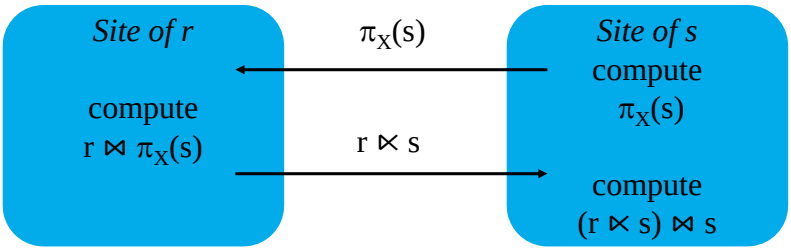
$r \ltimes s = \{t \mid t \text{ in } r \text{ and } \exists u \text{ in } s, t[X] = u[X]\}$

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

capacity $\ltimes$ uses

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

Semi-Join is Useful in Distributed Query Processing



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

Apply (Map, Do-all)

relational algebra

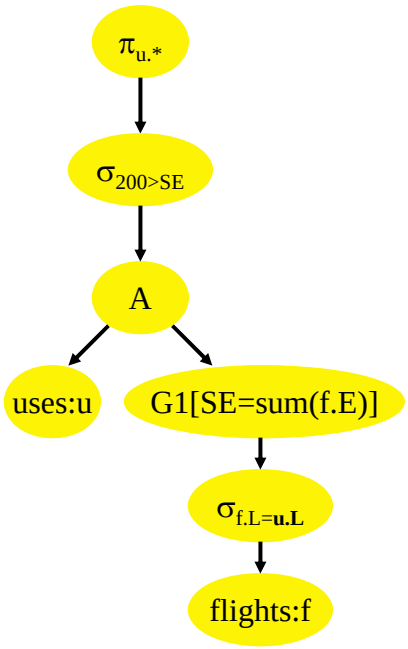
concat

$r\ A\ e(x) = \{t{+}{+}u \mid t \text{ in } r \text{ and } u \text{ in } e(t)\}$

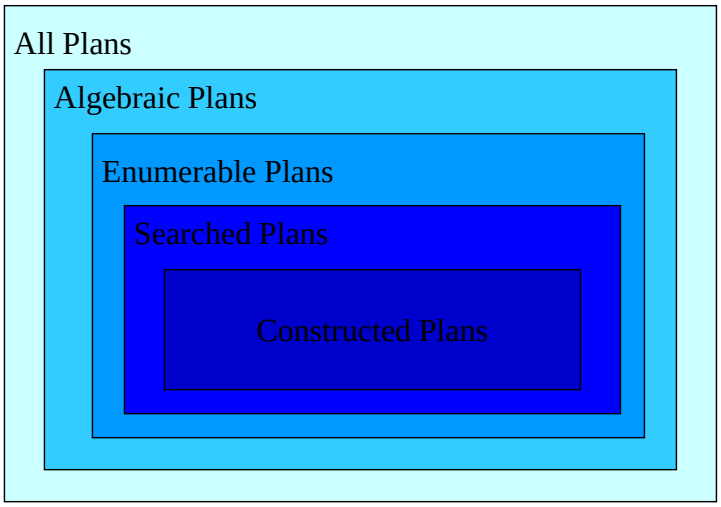
```
select u.*
from uses as u
where 200 >
   (select sum(f.E)
    from flights as f
    where u.L = f.L)
```

⟨A:1, B:3⟩ ++ ⟨C:5⟩

⟨A:1, B:3, C:5⟩



Optimization: Logical Expression Enumeration



David Maier & Lois Delcambre

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

Bellman's 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

David Maier & Lois Delcambre

Example Metrics

size(E ⋈ F) = size(E)*size(F)/10

cost(E ⋈ F) = size(E ⋈ F) + cost(E) + cost(F)

cost(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
```