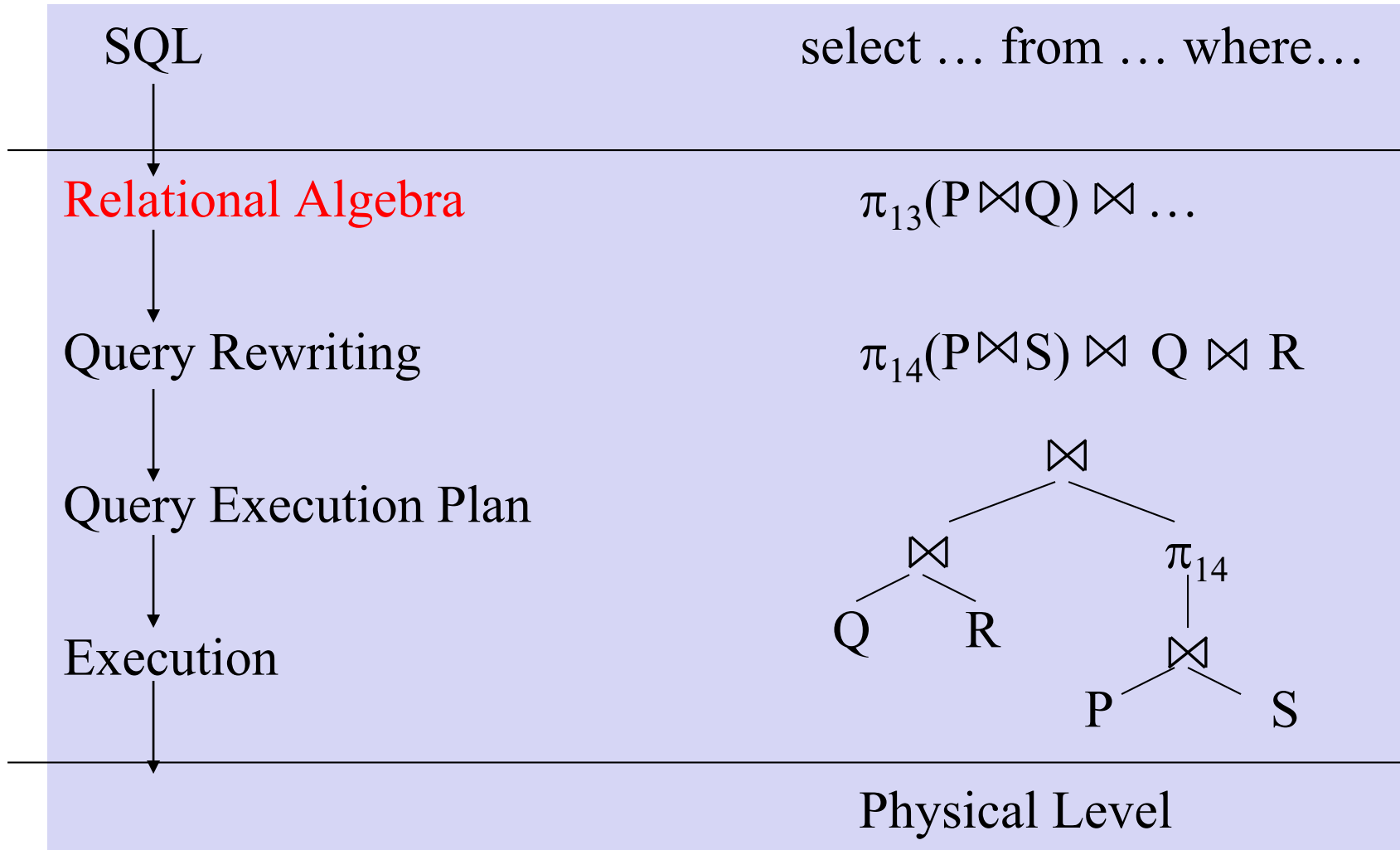


Journey of a Query



Relational Algebra

- Simple set of algebraic operations on relations

- We use set semantics (no duplicates) and no nulls
- There are extensions with bag semantics and nulls

Projection π

Selection σ

Join \bowtie

Union \cup

Difference $-$

Attribute renaming ρ

Relational Algebra

- Projection: eliminate some columns

$\pi_X(R)$

R table name

$X \subseteq \text{attributes}(R)$

– *Find titles of current movies:* $\pi_{\text{TITLE}}(\text{SCHEDULE})$

Another example:

R	A	B	C
	0	1	2
	0	2	2
	1	3	1
	0	1	3

$\pi_A(R) =$	A
	0
	1

$\pi_{AB}(R) =$	A	B
	0	1
	0	2
	1	3

No repetitions of tuples!

Relational Algebra

- Selection: select tuples satisfying condition of the form

att = value

att \neq value

att₁ = (\neq) att₂

other comparators possible

$\sigma_{\text{cond}}(\mathbf{R})$ where cond is a condition involving only attributes of R

e.g. $\sigma_{A=0}(\mathbf{R}) =$

A	B	C
0	1	2
0	2	2
0	1	3

$$\sigma_{B=C}(\mathbf{R}) =$$

A	B	C
0	2	2

$$\sigma_{A \neq 0}(\mathbf{R}) =$$

A	B	C
1	3	1

Relational Algebra

- Union, Difference

R, S: tables with the same attributes

$R \cup S$, $R - S$: union, difference of sets of tuples in R and S

Example:

r	<table><tr><th>A</th><th>B</th></tr><tr><td>α</td><td>1</td></tr><tr><td>α</td><td>2</td></tr><tr><td>β</td><td>1</td></tr></table>	A	B	α	1	α	2	β	1	s	<table><tr><th>A</th><th>B</th></tr><tr><td>α</td><td>2</td></tr><tr><td>β</td><td>3</td></tr></table>	A	B	α	2	β	3	$r \cup s$	<table><tr><th>A</th><th>B</th></tr><tr><td>α</td><td>1</td></tr><tr><td>α</td><td>2</td></tr><tr><td>β</td><td>1</td></tr><tr><td>β</td><td>3</td></tr></table>	A	B	α	1	α	2	β	1	β	3	$r - s$	<table><tr><th>A</th><th>B</th></tr><tr><td>α</td><td>1</td></tr><tr><td>β</td><td>1</td></tr></table>	A	B	α	1	β	1
A	B																																				
α	1																																				
α	2																																				
β	1																																				
A	B																																				
α	2																																				
β	3																																				
A	B																																				
α	1																																				
α	2																																				
β	1																																				
β	3																																				
A	B																																				
α	1																																				
β	1																																				

Join $R \bowtie S$: operation combining tuples from two tables
(same as **natural join** in SQL)

e.g.

R	A	B
---	---	---

S	B	C
---	---	---

$R \bowtie S$	A	B	C
---------------	---	---	---

- Note: more than one common attribute allowed

Relational Algebra

Examples:

R	A	B	S	B	C	$R \bowtie S =$	A	B	C
	0	1		1	2		0	1	2
	0	2		1	3		0	1	3
	5	3		2	2		0	2	2

Note that $\langle 5, 3 \rangle$ is “lost”.

If S is	S	C	then $R \bowtie S =$	A	B	C
		2		0	1	2
		3		0	1	3
				0	2	2
				0	2	3
				5	3	2
				5	3	3

Definition of Join ⋈

Let r and s be relations on schemas R and S respectively.

Then, $r \bowtie s$ is a relation with attributes $\text{att}(R) \cup \text{att}(S)$ obtained as follows:

- Consider each pair of tuples t_r from r and t_s from s .
- If t_r and t_s have **the same value** on each of the attributes in $\text{att}(R) \cap \text{att}(S)$, add a tuple t to the result, where
 - t has the same value as t_r on r
 - t has the same value as t_s on s

Note: if $R \cap S$ is empty, the join consists of all combinations of tuples from R and S , i.e. their cross-product

- Attribute renaming: change the name of an attribute

$$\delta_{A1 \rightarrow A2}(R)$$

“change A1 to A2 in relation R”

Example:

R	A	B
---	---	---

$\delta_{A \rightarrow C}(R)$	C	B
-------------------------------	---	---

A is renamed to C in R, content is unchanged.

Note: can rename several attributes at once.

Relational Algebra

- Basic set of operations:

$$\pi, \sigma, \cup, -, \bowtie, \delta$$

- Back to movie example queries:

1. Titles of currently playing movies:

$$\pi_{\text{TITLE}}(\text{schedule})$$

2. Titles of movies by Berto:

$$\pi_{\text{TITLE}}(\sigma_{\text{DIR}=\text{BERTO}}(\text{movie}))$$

3. Titles and directors of currently playing movies:


$$\pi_{\text{TITLE}, \text{DIR}}(\text{movie} \bowtie \text{schedule})$$

4. Find the pairs of actors acting together in some movie


$$\pi_{\text{actor1, actor2}} (\delta_{\text{actor} \rightarrow \text{actor1}} (\pi_{\text{title, actor}} (\text{movie})) \bowtie \delta_{\text{actor} \rightarrow \text{actor2}} (\pi_{\text{title, actor}} (\text{movie})))$$

5. Find the actors playing in every movie by Berto


$$\pi_{\text{actor}} (\text{movie}) - \pi_{\text{actor}} [(\pi_{\text{actor}} (\text{movie}) \bowtie \pi_{\text{title}} (\sigma_{\text{dir} = \text{BERTO}} (\text{movie}))) - \pi_{\text{actor, title}} (\text{movie})]$$



actor



title by Berto



actor acts in title

actors for which there is a movie by Berto in which they do not act

Other useful operations

- Intersection $R \cap S$
- Division (Quotient) $R \div S$

R	A	B

S	B

$R \div S: \{a \mid \langle a, b \rangle \in R \text{ for every } b \in S\}$

e.g.:

R	A	B
	0	α
	0	β
	1	α
	1	β
	1	γ
	2	α

S	B
	α
	β

$R \div S$	A
	0
	1

Another Division Example

Find the actors playing in every movie by Berto

$$\pi_{\text{TITLE, ACTOR}}(\text{movie}) \div \pi_{\text{TITLE}}(\sigma_{\text{DIR=BERTO}}(\text{movie}))$$

Division by multiple attributes

Relations r , s :

r	A	B	C	D	E
	α	a	α	a	1
	α	a	γ	a	1
	α	a	γ	b	1
	β	a	γ	a	1
	β	a	γ	b	3
	γ	a	γ	a	1
	γ	a	γ	b	1
	γ	a	β	b	1

s	D	E
	a	1
	b	1

$r \div s$:

A	B	C
α	a	γ
γ	a	γ

Relational Algebra

- Note:
 - π is like \exists “there exists”...
 - \div is like \forall “for all”...
- Expressing \div using other operators:

$$R \div S = \pi_A(R) - \pi_A((\pi_A(R) \bowtie S) - R)$$

$$\begin{array}{c|cc} R & A & B \\ \hline & & \end{array} \quad \begin{array}{c|c} S & B \\ \hline & \end{array}$$

Similar to: $\forall x \varphi(x) \equiv \neg \exists x \neg \varphi(x)$

Calculus Vs. Algebra

- Theorem: Calculus and Algebra are **equivalent**

Basic Correspondence

Algebra Operation

Calculus Operator



Example

- “Find theaters showing movies by Bertolucci”:
 - SQL:
 - SELECT s.theater
FROM schedule s, movie m
WHERE s.title = m.title AND m.director = ‘Berto’
 - tuple calculus:
 - $\{ t: \text{theater} \mid \exists s \in \text{schedule} \exists m \in \text{movie} [t(\text{theater}) = s(\text{theater}) \wedge s(\text{title}) = m(\text{title}) \wedge m(\text{director}) = \text{Berto}] \}$
 - relational algebra:

$$\pi_{\text{theater}} (\text{schedule} \bowtie \sigma_{\text{dir} = \text{Berto}} (\text{movie}))$$

Note: number of items in FROM clause = (number of joins + 1)

Examples

Beer drinker's database:

frequents	drinker	bar

serves	bar	beer

likes	drinker	beer

Find the drinkers who frequent some bars serving Coors

frequents	drinker	bar

serves	bar	beer

likes	drinker	beer

answer	drinker

Find the drinkers who frequent at least one bar serving a beer they like

frequents	drinker bar

serves	bar beer

likes	drinker beer

answer	drinker

Find the drinkers who frequent ONLY bars serving a beer they like

frequents	drinker	bar

serves	bar	beer

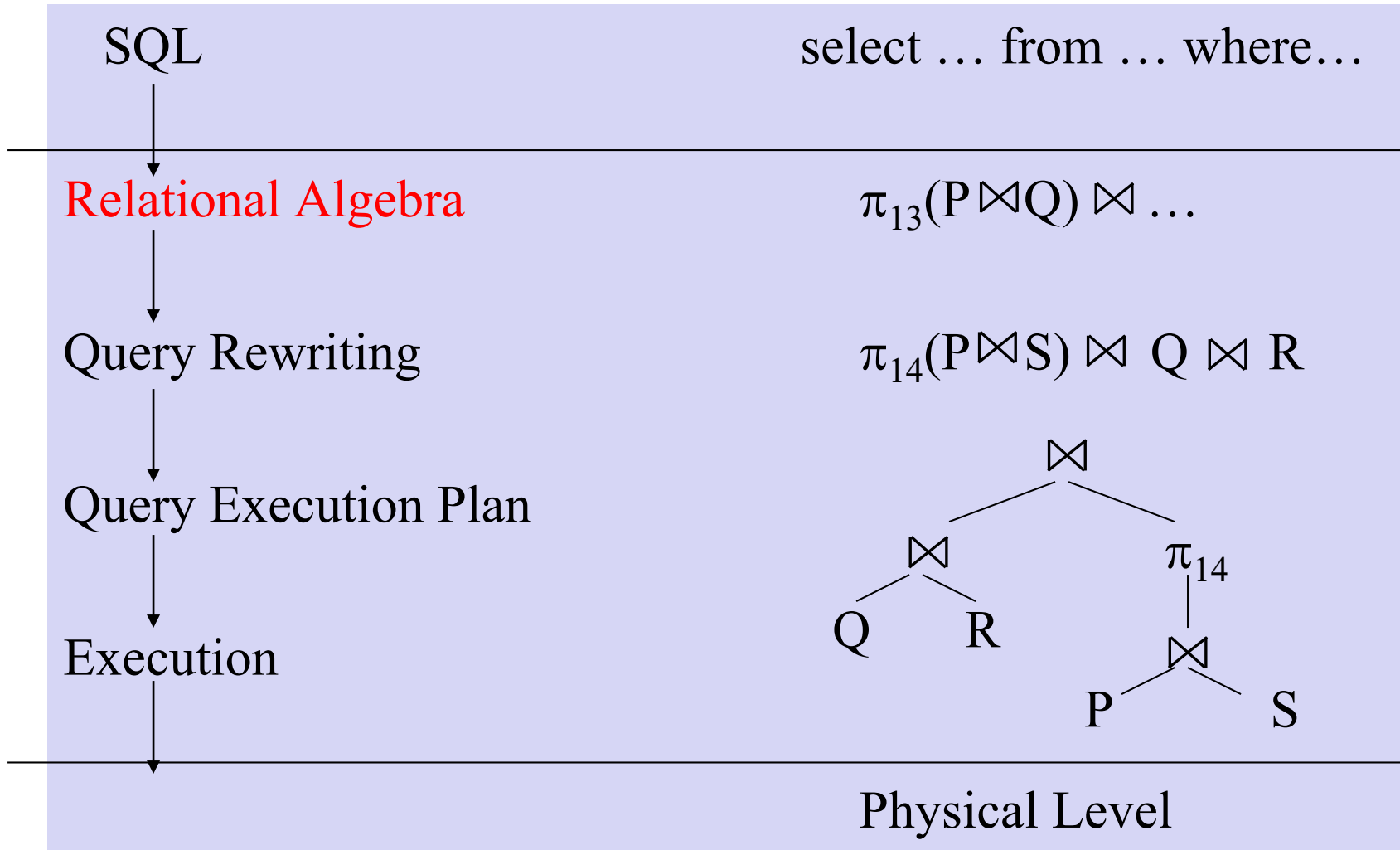
likes	drinker	beer

answer	drinker

Query Processing

- The query processor turns user queries and data modification commands into a sequence of operations (or algorithm) on the database
 - from high level queries to low level commands
- Decisions made by the query processor
 - Which of the algebraically equivalent forms of a query will lead to the most efficient algorithm?
 - For each algebraic operator what algorithm should we use to run the operator?
 - How should the operations pass data from one to the other? (eg, main memory buffers, disk buffers)

Journey of a Query



Query Rewriting

- Static Optimization (independent on current database)
- Simplification Rules:

Example: R: AB S: BC

$$R \bowtie R \longrightarrow R$$

$$R - R \longrightarrow \emptyset$$

$$\emptyset \bowtie R \longrightarrow \emptyset$$

$$\sigma_{A=a}(R \bowtie S) \longrightarrow \sigma_{A=a}(R) \bowtie S \quad \text{“pushing selection”}$$

$$\pi_{AB}(R \bowtie S) \longrightarrow R \bowtie \pi_B(S) \quad \text{“column elimination”}$$

$$\pi_A(\pi_{AB}(T)) \longrightarrow \pi_A(T) \quad \text{“cascading projections”}$$

Etc.

Example: pushing selections & column elimination

$$\pi_{\text{drinker}} \left(\sigma_{\text{beer} = \text{Bass}} \left(\text{frequents} \bowtie \text{serves} \right) \right)$$



$$\pi_{\text{drinker}} \left(\text{frequents} \bowtie \sigma_{\text{beer} = \text{Bass}} (\text{serves}) \right)$$



$$\pi_{\text{drinker}} \left(\text{frequents} \bowtie \pi_{\text{bar}} \left(\sigma_{\text{beer} = \text{Bass}} (\text{serves}) \right) \right)$$

Query rewriting

- Other techniques (sometimes applied on SQL query before translation to algebra)

Nested query decorrelation, view unfolding, common subqueries, etc

- Can be spectacularly effective on complex queries

Example: Bill McKenna's "monster query"
Performance improved by 95%

More examples

source: Bill McKenna, ParAccel

- Subquery decorrelation

P(A,B)
Q(C,D)

```
SELECT COUNT(*)  
FROM P  
WHERE A < (SELECT MAX(Q.C)  
FROM Q WHERE P.B=Q.D) ;
```



```
SELECT COUNT(distinct P.*)  
FROM P , Q  
WHERE B = D and A < C
```

More examples

source: Bill McKenna, ParAccel

- View unfolding

R(A,B,C)

```
CREATE VIEW V(D, E, F) AS  
SELECT A, B, SUM(C) FROM R  
GROUP BY A, B ;  
SELECT D, SUM(F) as Total  
FROM V  
GROUP BY D;
```



```
SELECT A as D, SUM(C) as Total  
FROM R  
GROUP BY A
```

More examples

source: Bill McKenna, ParAccel

- Rewriting can be tricky if duplicates matter

R(A,B)

```
SELECT t.A  
FROM (SELECT A, B FROM R  
GROUP BY A, B) AS t
```

Can B be eliminated from the SELECT for t?

A: yes B: no

Also from the GROUP BY?

A: yes B: no

- Later: rewriting SQL queries to minimize joins

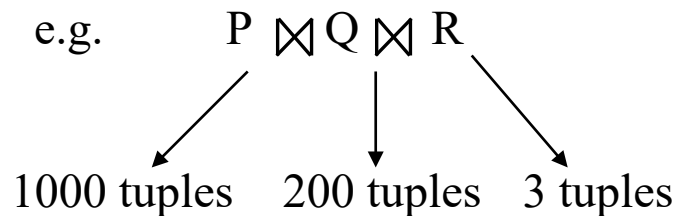
Query Evaluation Plan

- Additional decisions on evaluation of rewritten query
- No longer static: use partial information about database
- Example: For given algebra query, order of evaluation
- Some factors:

- Statistical info on data

(# tuples, selectivity, etc.)

e.g.

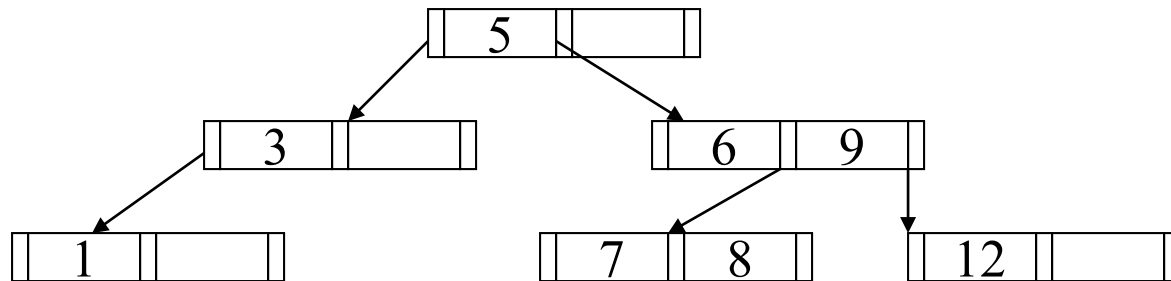


Is it always better to first join Q and R, then join with P ?

A: yes **B: no**

- Common subexpressions
 - Availability of indexes

- **Index:** auxiliary file providing fast access to physical location of record with given key value
- Search tree



- If tree is **balanced**, access time is $\log_p(n)$ for p-ary trees
- Special search trees: **B-trees, B⁺-trees**
always balanced

- Note: SQL allows specification of indexes
 - e.g. `CREATE INDEX(NAME_INDEX)
ON EMPLOYEE(NAME)`
- Indexes can also be created on-the-fly by the query processor
- New approach: **machine learning!**

Implementation of Individual Operations: \bowtie , σ , ...

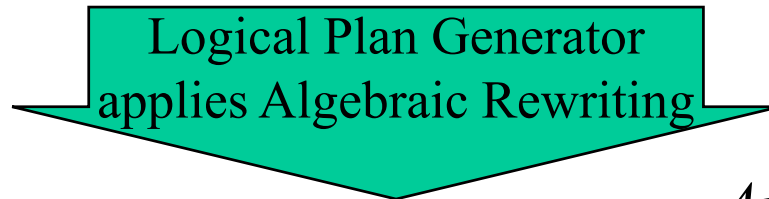
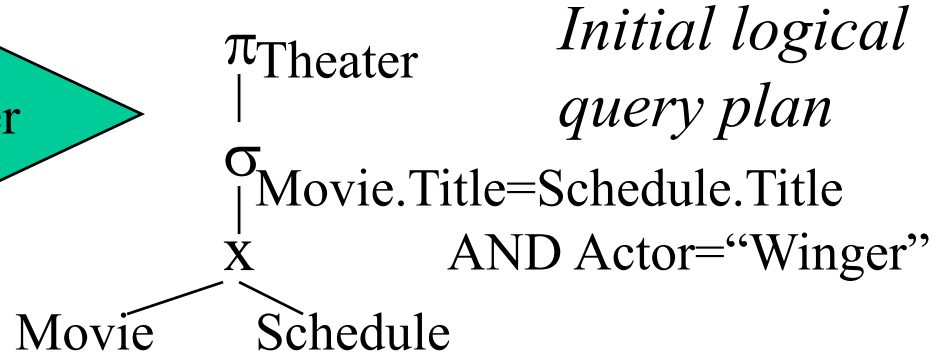
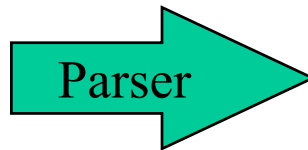
- Related to access method and file organization
- Unordered files: \bowtie takes $O(n^2)$
- Files ordered by some key field
 - Access by index: \bowtie takes $O(n \log n)$
- Ordered file (sequential) by name

Name	Sal	Name	Sal	...
Aaron		Abbott		

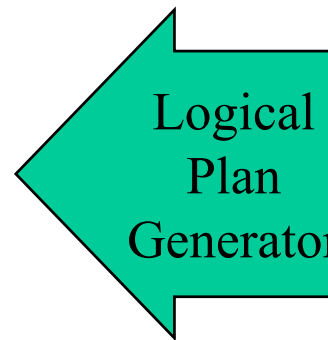
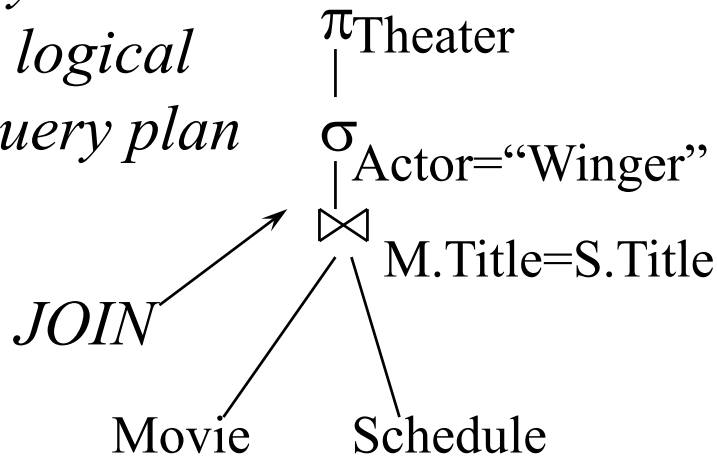
- e.g. R: AB S: BC; index on B
 - $R \bowtie S$ takes $O(n \log n)$

Query Processing and Optimization: The Journey of a Query

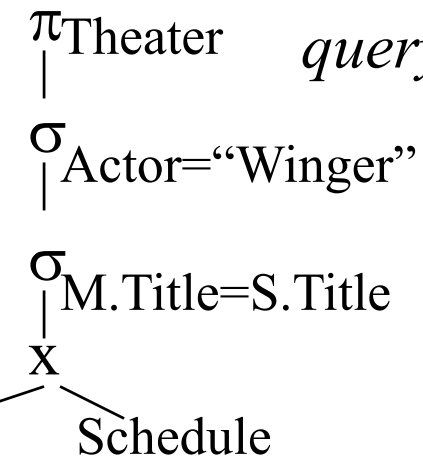
SELECT Theater
FROM Movie, Schedule
WHERE
Movie.Title=Schedule.Title
AND Actor="Winger"



And yet another logical query plan

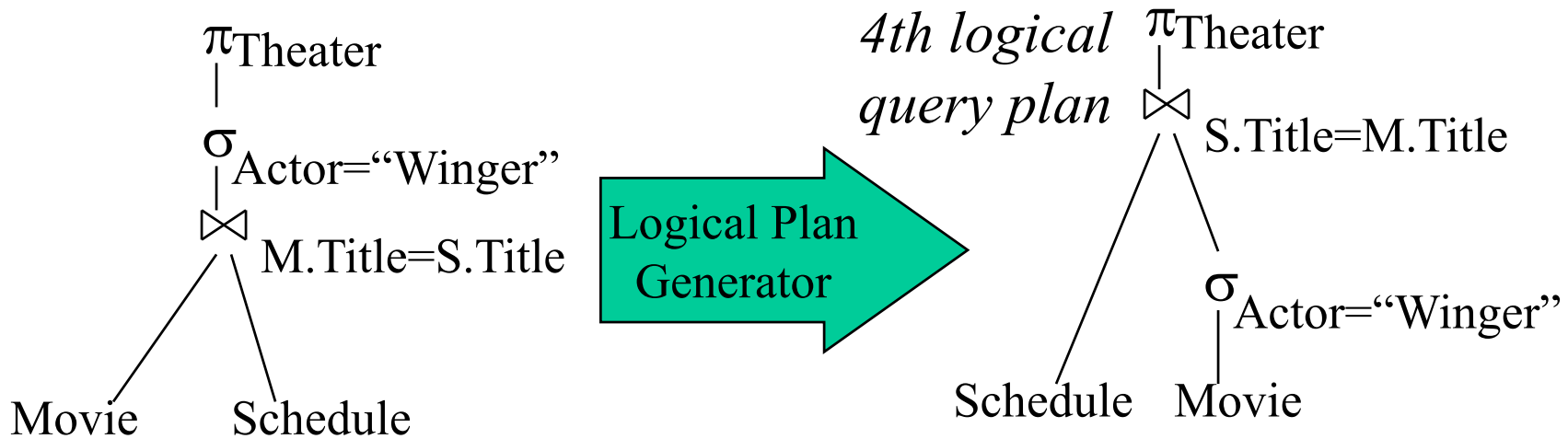


Another logical query plan

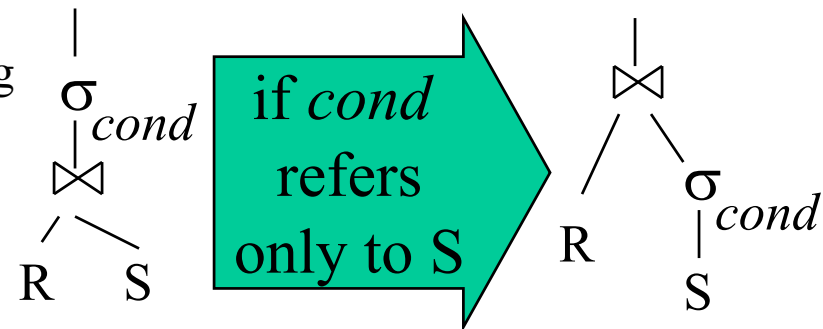


The Journey of a Query cont'd:

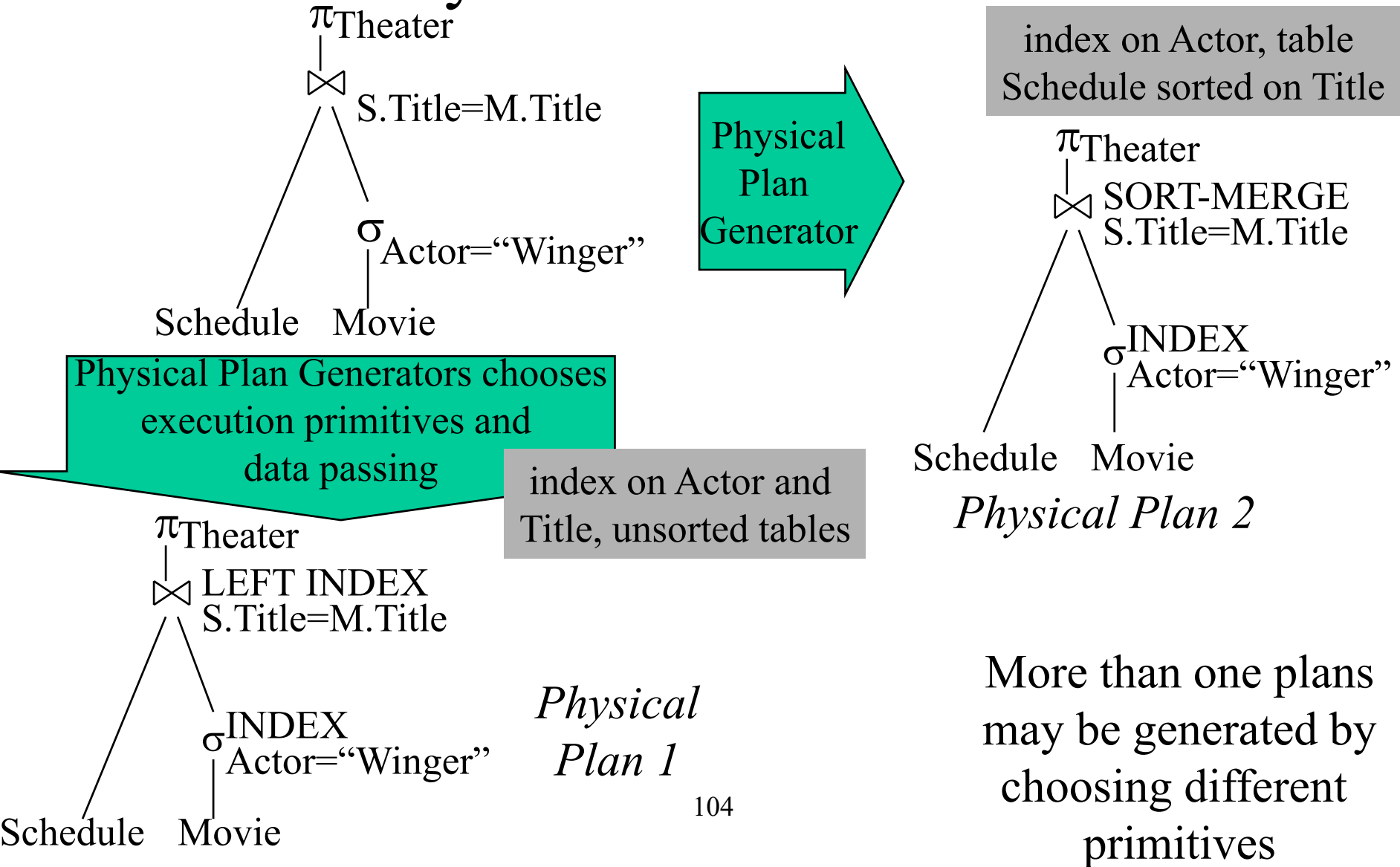
Summary of Logical Plan Generator



- 4 logical query plans created
- algebraic rewritings were used for producing the candidate logical query plans
- the last one is the winner (at least, cannot be a big loser)
- in general, multiple logical plans may “win” eventually



The Journey of a Query Continues at the Physical Plan Generator



Issues in Query Processing and Optimization

- Generate Plans
 - systematically transform expressions
 - employ execution primitives for computing relational algebra operations
- Estimate Cost of Generated Plans
 - Statistics, cost model
- “Smart” Search of the Space of Possible Plans
 - always do the “good” transformations (e.g. relational algebra optimization)
 - prune the space (e.g., System R)
- Often the above steps are mixed

Parallel processing

- theoretical potential for perfect scaling:

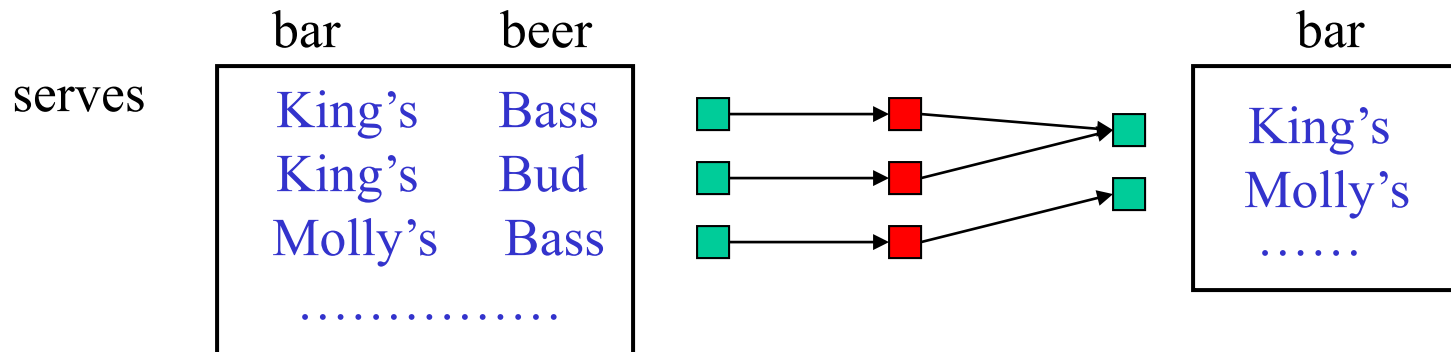
given sufficient resources, performance does not degrade as the database becomes larger

- cost: number of processors polynomial in the size of the database
- role of algebra: operations highlight parallelism

Each algebra operation can in principle be implemented
very efficiently in parallel

Example: projection

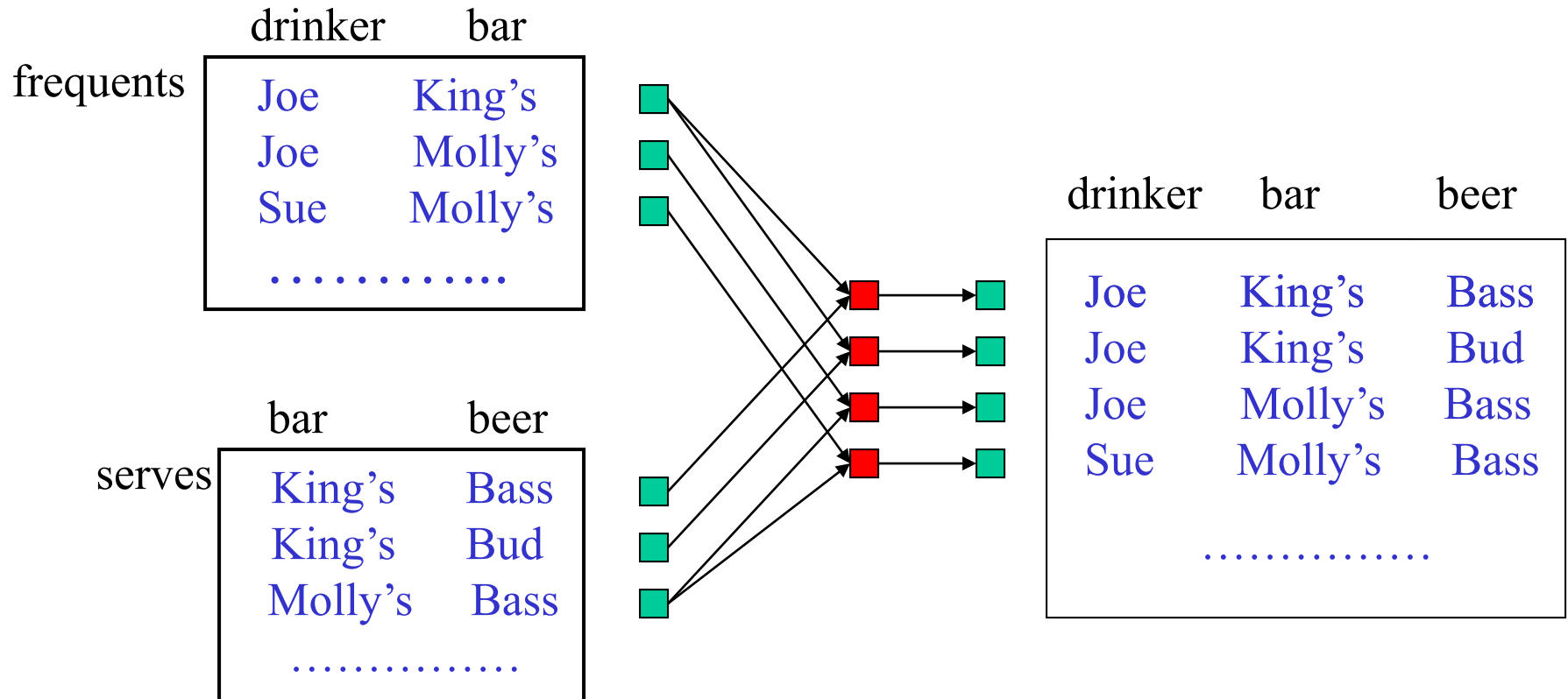
$\pi_{\text{bar}}(\text{serves})$



Constant parallel time!

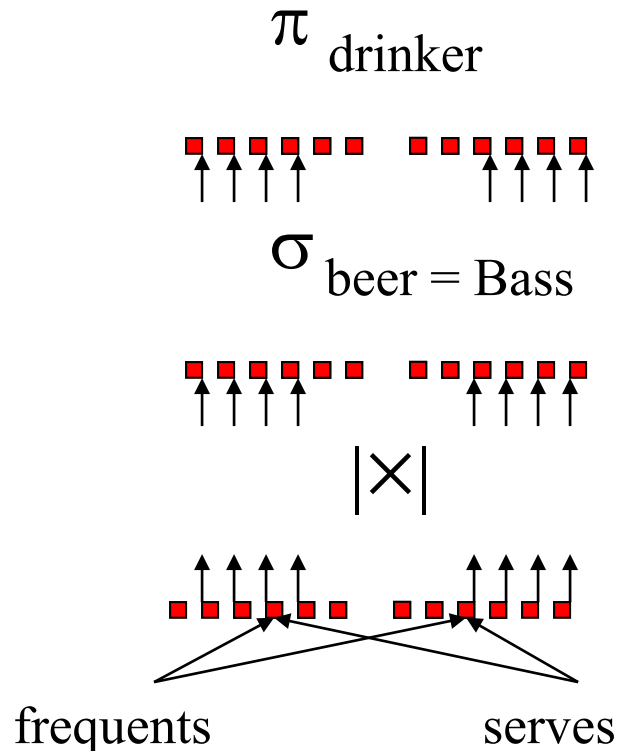
Another example: join

frequents \bowtie serves



Every relational algebra query takes **constant parallel time**!

$$\pi_{\text{drinker}} \left(\sigma_{\text{beer} = \text{Bass}} \left(\text{frequents} \bowtie \text{serves} \right) \right)$$



constant parallel time

From: Raghu Ramakrishnan

I am very happy to share with you the news that the Gamma Parallel Database project is being honored with the 2009 ACM Software Systems Award! The citation reads:

* **Gamma Parallel Database System** - the Software System Award for this prototype parallel relational database system, which was the first parallel database management system (DBMS) to publish results **demonstrating the ability to run the same query with the same performance on larger data sets by simply adding hardware nodes**. The Gamma project had a profound impact on the database field by demonstrating that scalable performance could be achieved without the use of specialized hardware. Developed by David J. DeWitt, Microsoft/University of Wisconsin-Madison (Emeritus); Robert Gerber of Microsoft; Murali Krishna of Hewlett Packard; Donovan A. Schneider of Yahoo!; Shahram Ghandeharizadeh of the University of Southern California; Goetz Graefe of Hewlett Packard; Michael Heytens of RGM Advisors; Hui-I Hsiao of IBM; Jeffrey F. Naughton of the University of Wisconsin-Madison; and Anoop Sharma of Hewlett Packard.

The Software System Award is given to an institution or individuals recognized for developing software systems that have had a lasting influence, reflected in contributions to concepts and/or commercial acceptance. Please join me in celebrating this recognition of seminal work in our community.

As a colleague from Wisconsin and Chair of SIGMOD, I'm doubly delighted! Raghu Ramakrishnan