

# **Advanced Database Management Systems**

**Lecture 17**  
**Query Processing: Chapter 15**

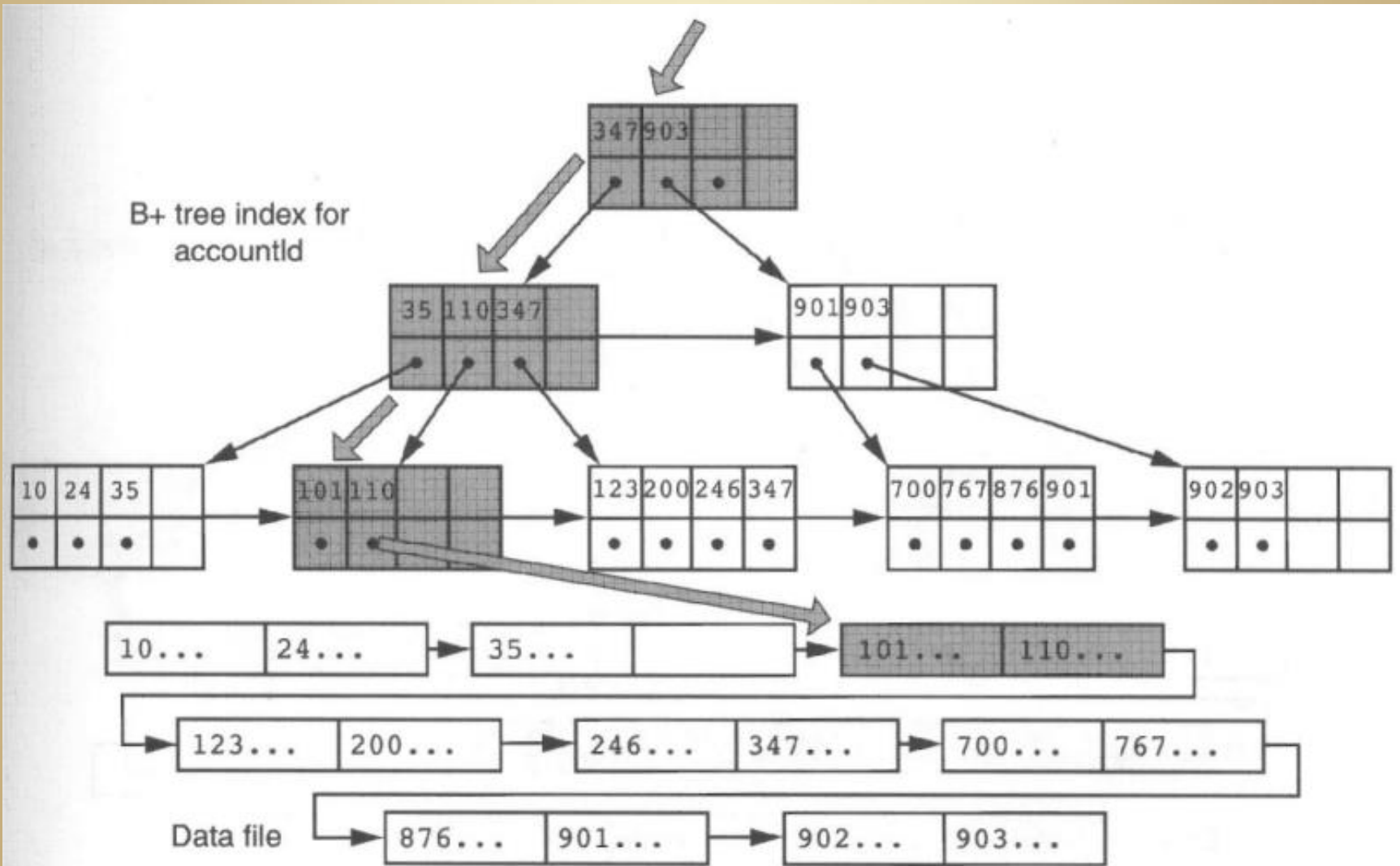
# Using Indexes for Queries

Database Instance and Available Indexes:

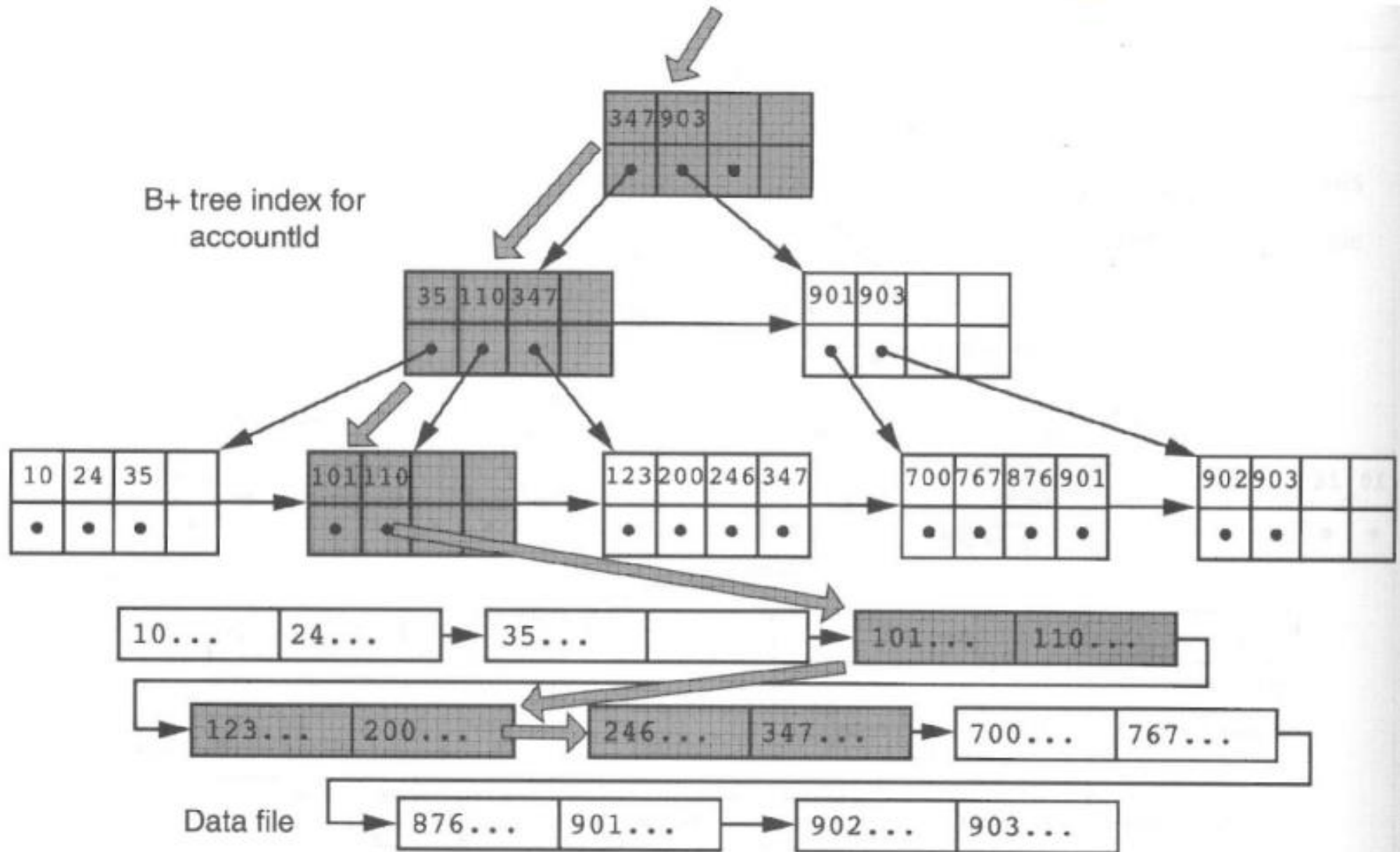
Table	Entries	Number of Entries per Block	Number of Blocks	Index Fields	Index Type	Keys per Node	Depth of B+ Tree
Customer	10,000	10	1,000	accountId	B+ tree and ordered sequential file	100	3
				lastName	B+ tree	50	3
				zipcode	hash	100	
Rental	1,000,000	100	10,000	accountId	B+ tree	100	3
				movieId	B+ tree	100	3
				date	B+ tree	100	2
Movie	10,000	20	500	movieId	B+ tree and ordered sequential file	100	3
				title	B tree	20	4
				genre	Hash	100	

Examples from: *Principles of Database Systems* by Greg Ricciardi, Addison Wesley, 2001

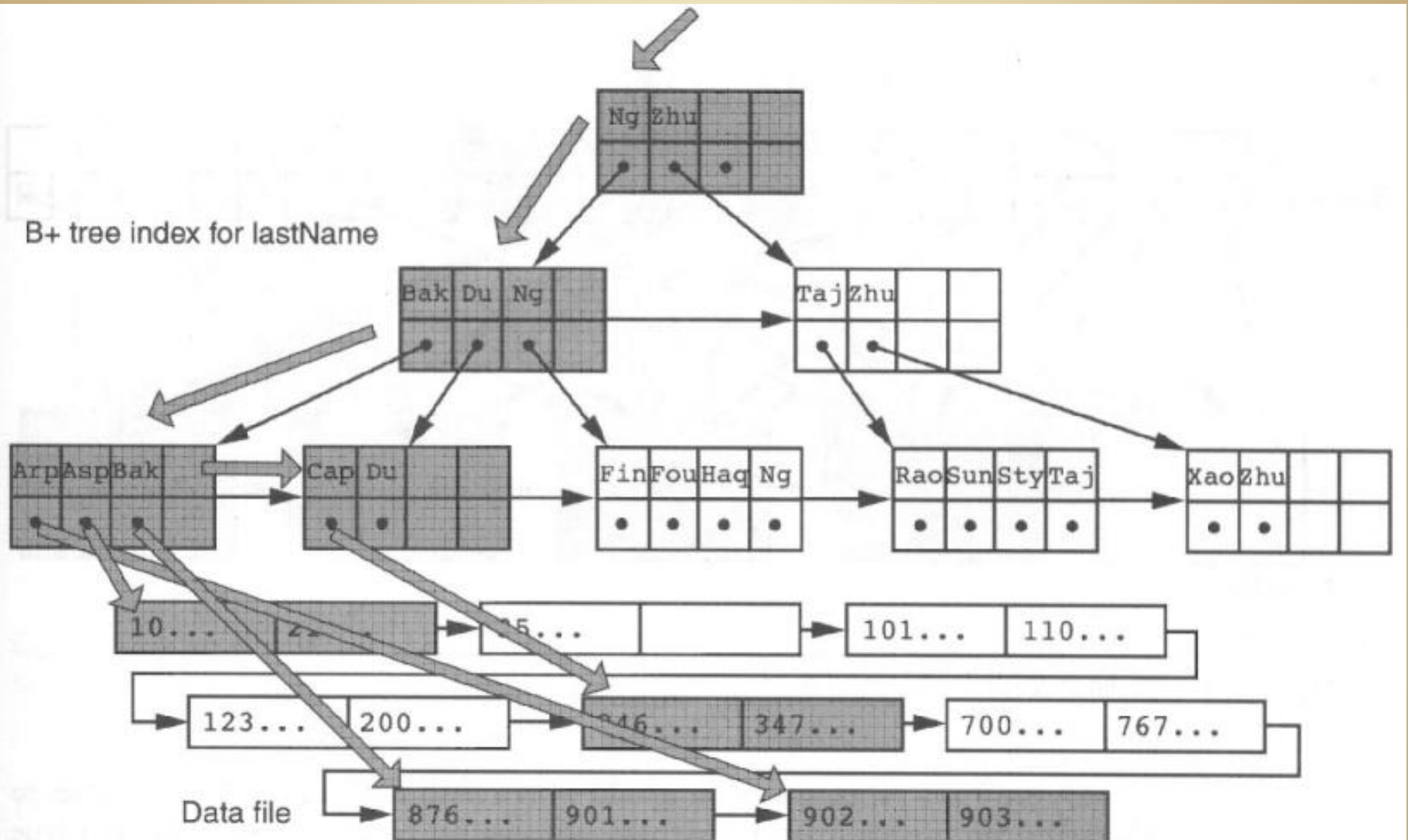
select \* from Customer  
where accountId = 101



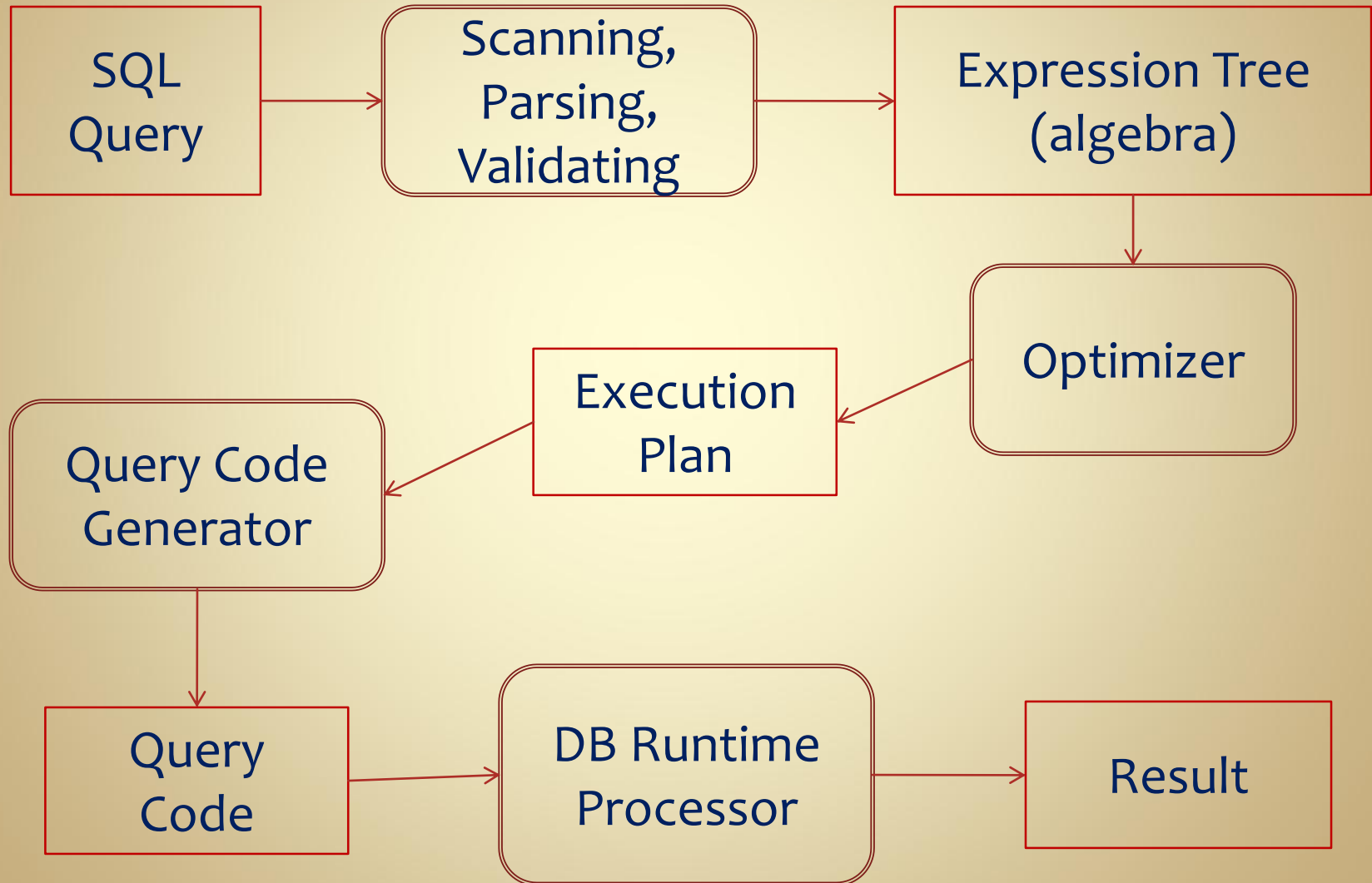
select \* from Customer  
where accountId >= 101 and accountId < 300



select \* from Customer where lastName < 'D'



# Query Processing



# Query Optimization

- Query optimization determines the most efficient (or sufficiently efficient) process for executing the query
- Optimization reorganizes an expression tree for a query using algebraic transformation rules
- Information used:
  - implementation techniques for algebra operators
  - algebraic transformation rules
  - heuristic rules
  - cost estimates



# Algebraic Transformations

- Commutative:  $\sigma \bowtie x \cup \cap$
- Associative:  $\bowtie x \cup \cap$
- Cascades of select:
  - $\sigma_{c1 \text{ AND } c2 \text{ AND } \dots \text{ AND } c_n}(R) \equiv \sigma_{c1}(\sigma_{c2}(\dots \sigma_{c_n}(R))\dots)$
  - selects can then be commuted
- Cascades of project:
  - $\pi_{L1}(\pi_{L2}(\dots \pi_{Ln}(R))\dots) \equiv \pi_{L1}(R)$
  - projects cannot be commuted



# Algebraic Transformations

- Commuting select and project:

- $\pi_{A_1, A_2, \dots, A_n}(\sigma_c(R)) \equiv \sigma_c(\pi_{A_1, A_2, \dots, A_n}(R))$

valid when selection condition only involves  
attributes in projection list

- Converting select/cross-product into join:

- $\sigma_c(R \times S) \equiv R \bowtie_c S$

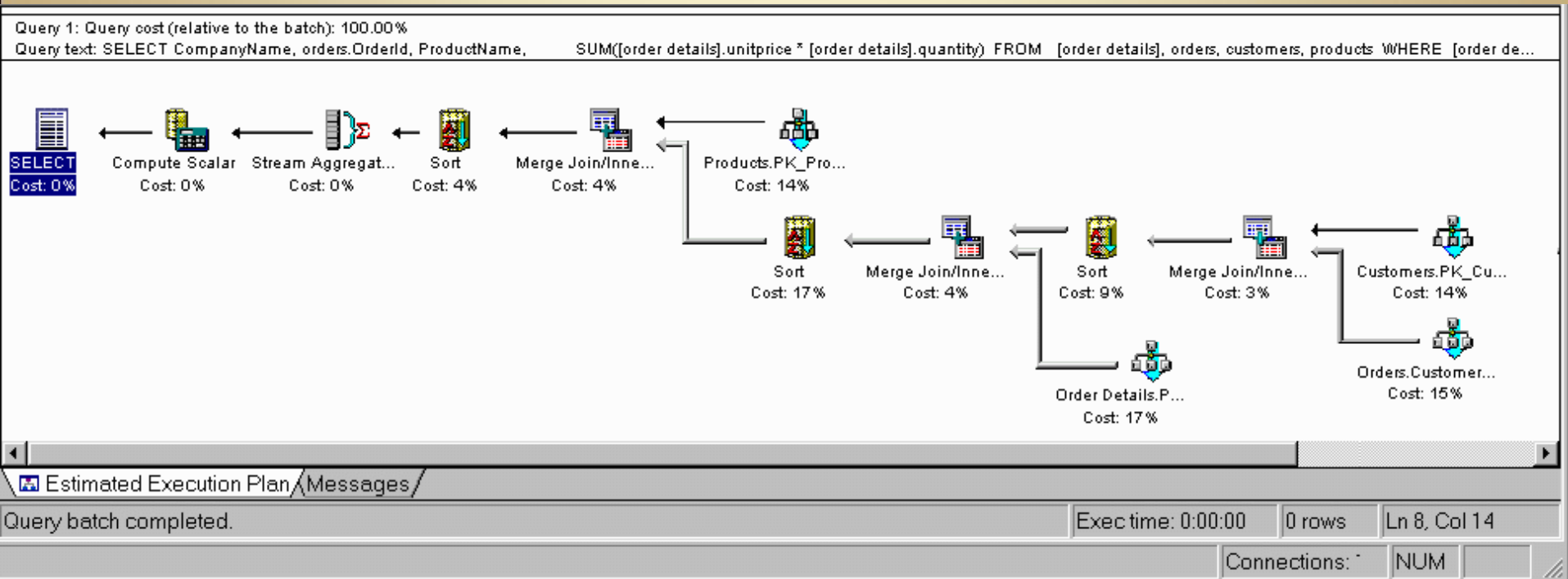
# Algebraic Transformations

- Commuting select and join or cross-product:
  - $\sigma_c(R \bowtie S) \equiv \sigma_c(R) \bowtie S$   
valid when selection condition involves only attributes in R
- Commuting project with join or cross-product:
  - $\pi_L(R \bowtie_c S) \equiv (\pi_{A_1, A_2, \dots, A_n}(R)) \bowtie_c (\pi_{B_1, B_2, \dots, B_n}(S))$   
where  $A_1, \dots, A_n \subseteq L$  are attributes of R,  
and  $B_1, \dots, B_n \subseteq L$  are attributes of S,  
and  $A_1, \dots, A_n, B_1, \dots, B_n$  are involved in C  
(slightly more complicated if C contains attributes not in L)

# Optimization Heuristics

- **Break up conjunctive select conditions**  
into a cascade of selects
  - more flexibility in moving selects
- **Push selects as early as possible,**  
using commutativity of select with other operators
- **Reorder sub-expressions such**  
that most restrictive selects are done first
- **Combine select/cross-product into join**
- **Push projects as early as possible**
  - keep only necessary attributes

# Example Query Plan



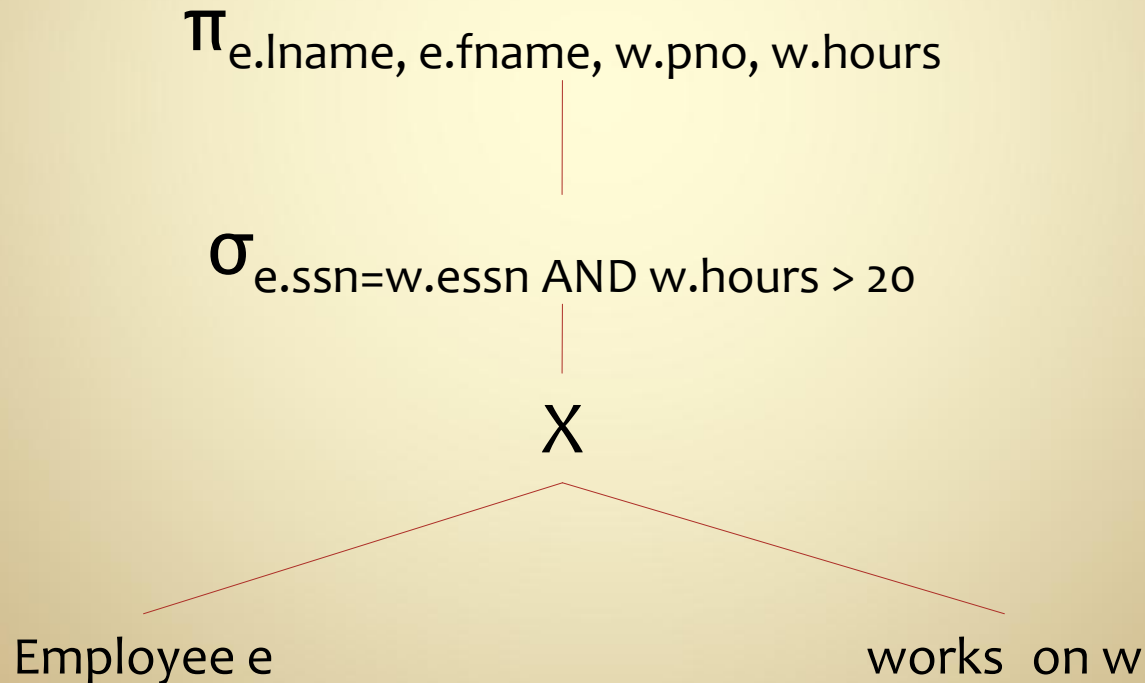
Microsoft SQL Server

# Optimization Processor

- Enumerate the query plans
  - Apply algebraic transformations & heuristics to get all reasonable trees
- Estimate the cost of each plan
  - Account for size of relations, available indexes, information about file layout, info about value distributions ...
- Choose the best (fastest) plan

# Example Query

```
select e.lname, e.fname, w.pno, w.hours  
from employee e, works_on w  
where e.ssn = w.essn and w.hours > 20;
```



# Example Query

$\pi_{e.lname, e.fname, w.pno, w.hours}$

$\sigma_{w.hours > 20}$

$\sigma_{e.ssn=w.essn}$

X

Employee e

works\_on w

Heuristic:  
Conjunctive  
select  $\rightarrow$  cascade  
of selects

Note: selects  
could commute



# Example Query

$\pi_{e.lname, e.fname, w.pno, w.hours}$

Heuristic:  
Combine select  
and cross  $\rightarrow$  join

$\sigma_{w.hours > 20}$

$\bowtie_{e.ssn=w.essn}$

Employee e

works\_on w

# Example Query

$\pi_{e.lname, e.fname, w.pno, w.hours}$

Heuristic:  
Push selects as  
early as possible

$\bowtie_{e.ssn=w.essn}$

Employee e

$\sigma_{w.hours > 20}$

works\_on w

# Example Query

$\pi_{e.lname, e.fname, w.pno, w.hours}$

$\pi_{e.ssn, e.lname, e.fname, w.essn, w.pno, w.hours}$

Heuristic:  
Push projects as  
early as possible

$\bowtie_{e.ssn=w.essn}$

First cascade  
projects to get  
attributes  
needed to push  
through join

Employee e

$\sigma_{w.hours > 20}$

works\_on w

# Example Query

$\pi_{e.lname, e.fname, w.pno, w.hours}$

$\bowtie_{e.ssn=w.essn}$

Heuristic:  
Push projects as  
early as possible

Push project  
through join

$\pi_{e.ssn, e.lname, e.fname}$

Employee e

$\pi_{w.essn, w.pno, w.hours}$

$\sigma_{w.hours > 20}$

works\_on w

# Example Query

$\pi_{e.lname, e.fname, w.pno, w.hours}$

$\bowtie_{e.ssn=w.essn}$

Heuristic:  
Push projects as  
early as possible

Push project  
through select

$\pi_{e.ssn, e.lname, e.fname}$

Employee e

$\sigma_{w.hours > 20}$

$\pi_{w.essn, w.pno, w.hours}$

works\_on w

# Example Query

- At this point we've generated nine different query plans for the same query
- Adding strategies/algorithms for implementing individual operations would add even more potential plans
- Next Step: Estimate the cost of executing each plan