Lecture 16: Query Processing and Optimization

CS 1555: Database Management Systems

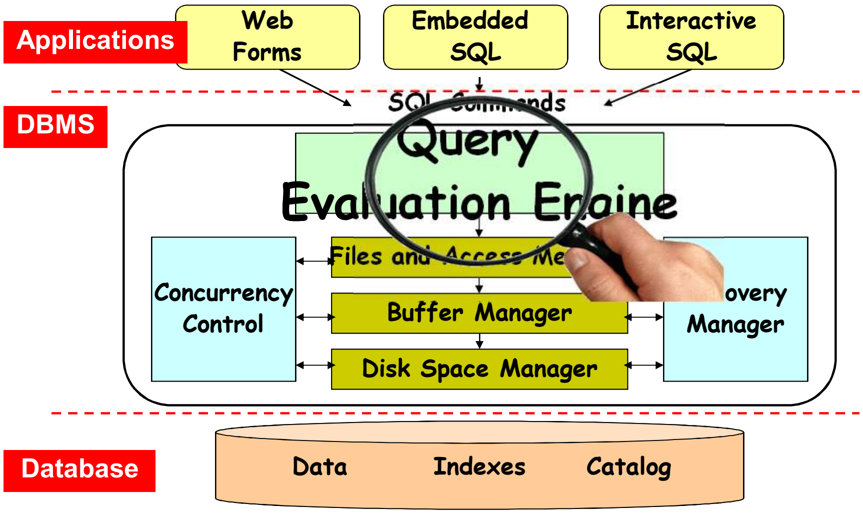
Constantinos Costa

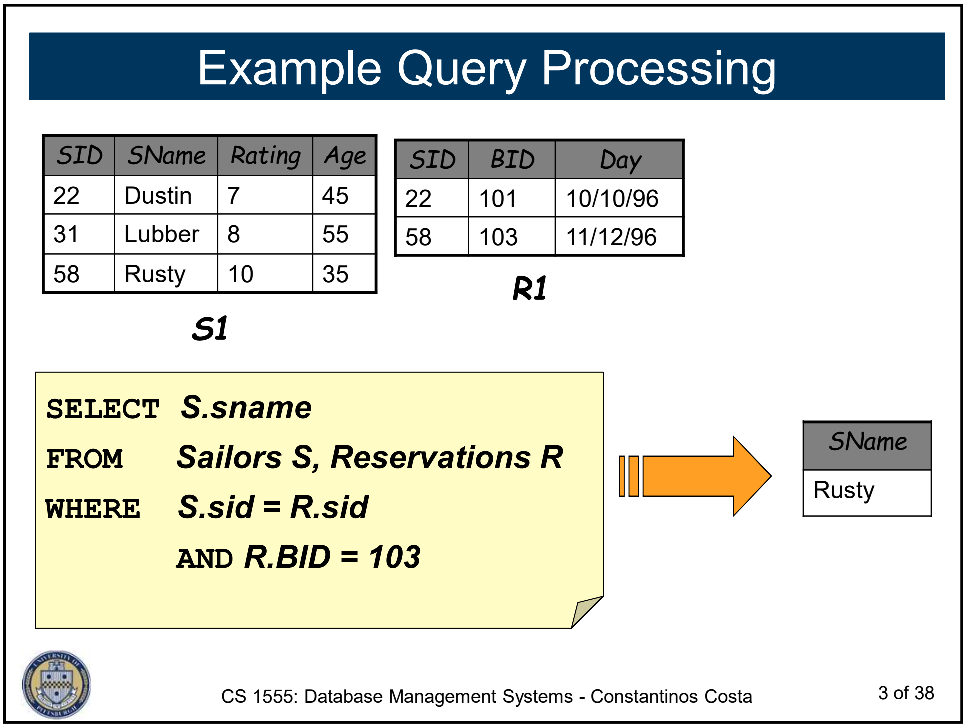
http://db.cs.pitt.edu/courses/cs1555/current.term/

March 21, 2019, 16:00-17:15
University of Pittsburgh, Pittsburgh, PA

Lectures based: P. Chrysanthis & N. Farnan Lectures

Database Management System (DBMS)





Example Query Processing

SID	SName	Rating	Age
22	Dustin	7	45
31	Lubber	8	55
58	Rusty	10	35

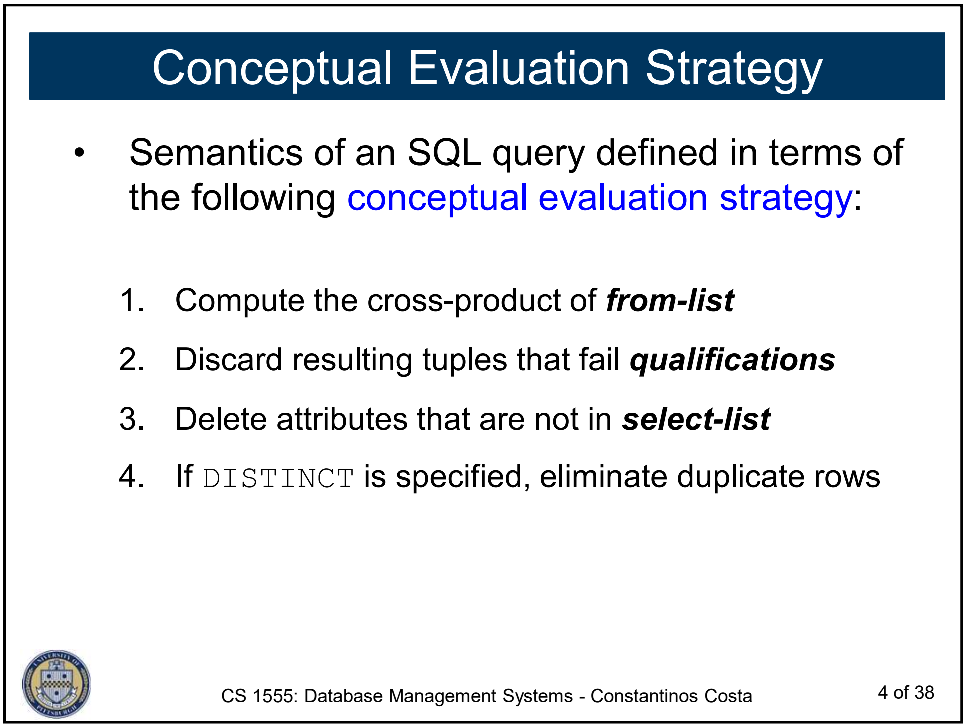
S1

SID	BID	Day
22	101	10/10/96
58	103	11/12/96

R1

```
SELECT  S.sname
FROM    Sailors S, Reservations R
WHERE   S.sid = R.sid
        AND R.BID = 103
```

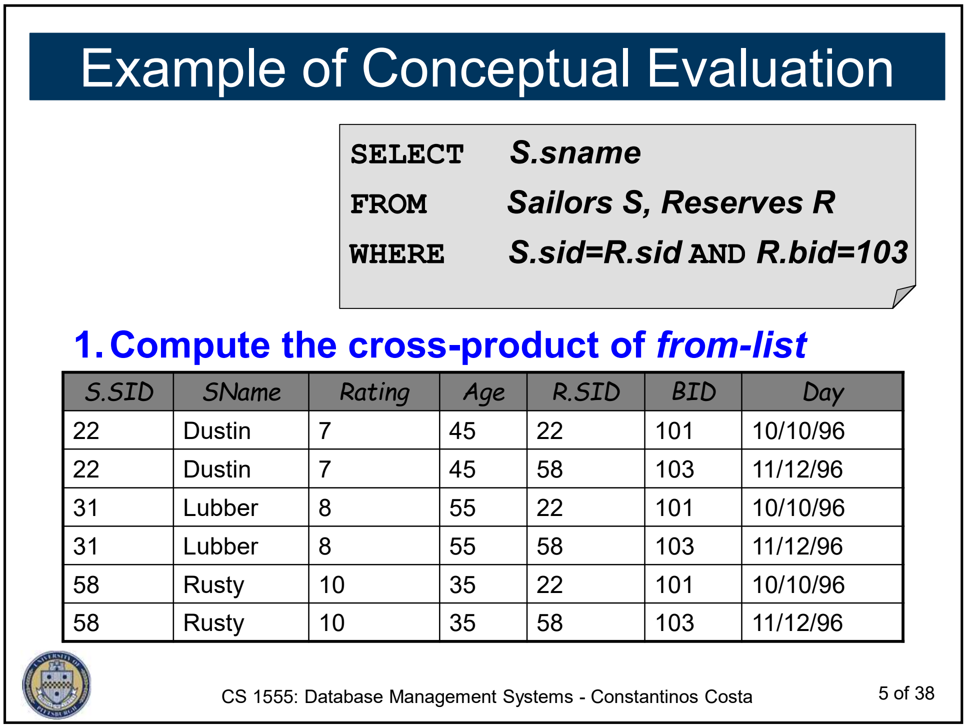
SName
Rusty



Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:

1. Compute the cross-product of ***from-list***
2. Discard resulting tuples that fail ***qualifications***
3. Delete attributes that are not in ***select-list***
4. If `DISTINCT` is specified, eliminate duplicate rows

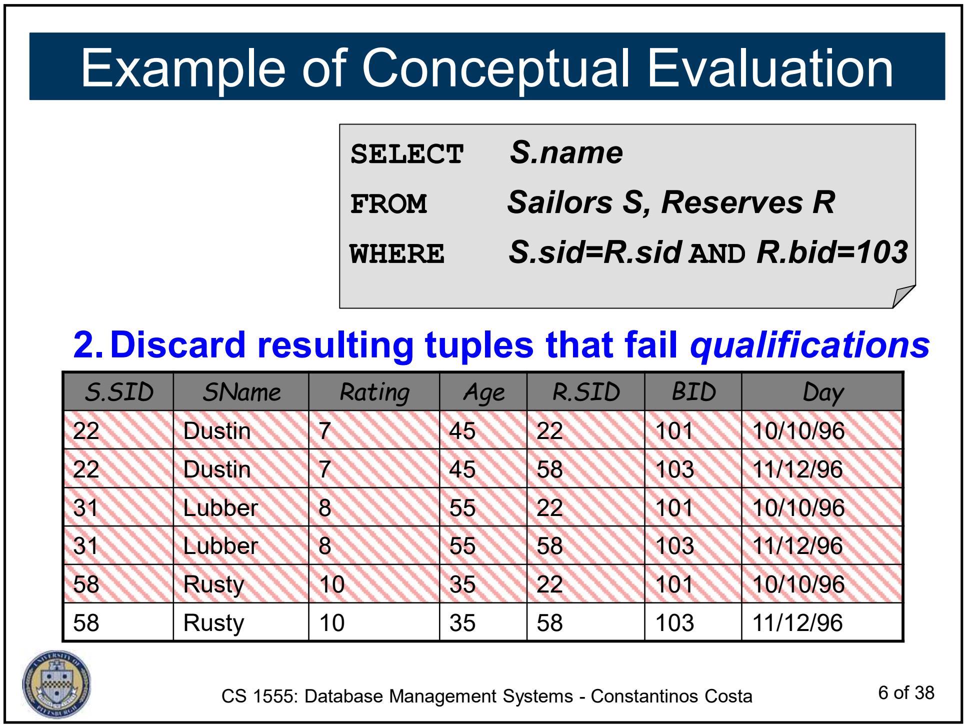


Example of Conceptual Evaluation

```
SELECT   S.sname
FROM     Sailors S, Reserves R
WHERE    S.sid=R.sid AND R.bid=103
```

1. **Compute the cross-product of *from-list***

S.SID	SName	Rating	Age	R.SID	BID	Day
22	Dustin	7	45	22	101	10/10/96
22	Dustin	7	45	58	103	11/12/96
31	Lubber	8	55	22	101	10/10/96
31	Lubber	8	55	58	103	11/12/96
58	Rusty	10	35	22	101	10/10/96
58	Rusty	10	35	58	103	11/12/96



Example of Conceptual Evaluation

```
SELECT   S.name
FROM     Sailors S, Reserves R
WHERE    S.sid=R.sid AND R.bid=103
```

2. **Discard resulting tuples that fail *qualifications***

S.SID	SName	Rating	Age	R.SID	BID	Day
22	Dustin	7	45	22	101	10/10/96
22	Dustin	7	45	58	103	11/12/96
31	Lubber	8	55	22	101	10/10/96
31	Lubber	8	55	58	103	11/12/96
58	Rusty	10	35	22	101	10/10/96
58	Rusty	10	35	58	103	11/12/96

Example of Conceptual Evaluation

```
SELECT    S.name
FROM      Sailors S, Reserves R
WHERE     S.sid=R.sid AND R.bid=103
```

3. Delete attributes that are not in *select-list*

S.SID	SName	Rating	Age	R.SID	BID	Day
22	Dustin	7	45	22	101	10/10/96
22	Dustin	7	45	58	103	11/12/96
31	Lubber	8	55	22	101	10/10/96
31	Lubber	8	55	58	103	11/12/96
58	Rusty	10	35	22	101	10/10/96
58	Rusty	10	35	58	103	11/12/96

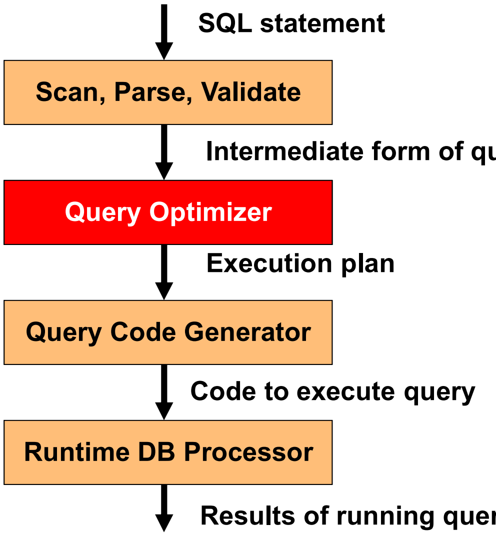
Conceptual Evaluation Strategy

- This strategy is probably the least efficient way to compute a query!

© Original Artist
Reproduction rights obtainable from
www.CartoonStock.com
SPEED
55
LIMIT
search ID: tprn62
"I don't think I can do it, but I'll try."

- An "**optimizer**" will find more efficient strategies to compute the same answers

Steps in Processing a Query


SQL statement

Scan, Parse, Validate

Intermediate form of query

Query Optimizer

Execution plan

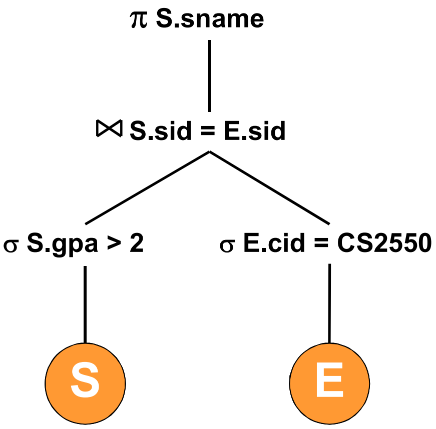
Query Code Generator

Code to execute query

Runtime DB Processor

Results of running query

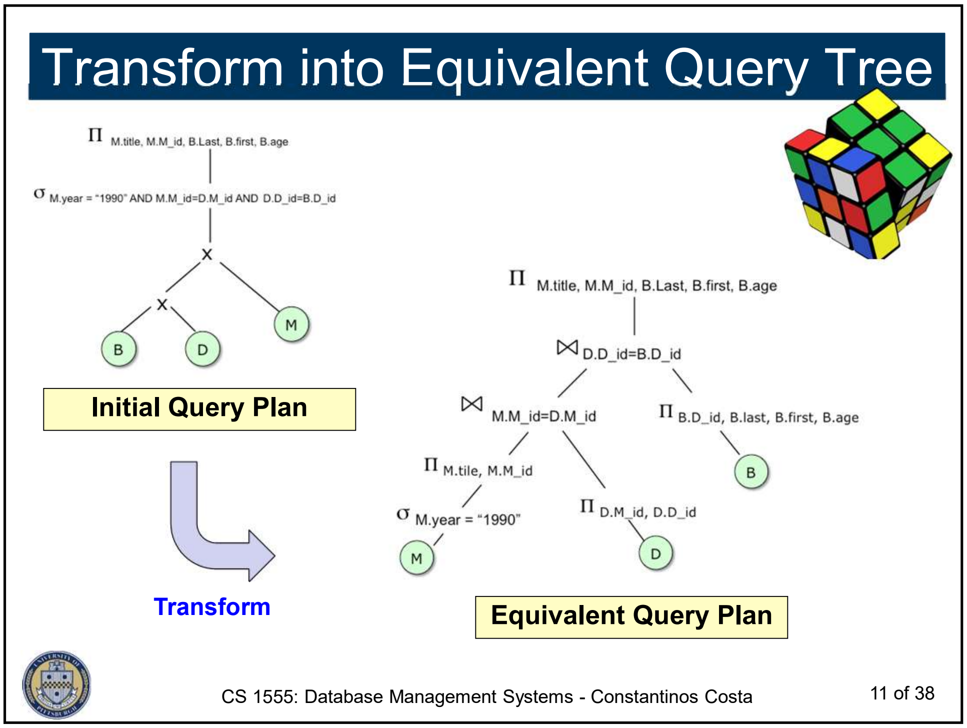
Query Tree



```
SELECT  S.sname
FROM    Student S, Enrolled E
WHERE   S.sid = E.sid
        AND S.gpa > 2
        AND E.cid = CS2550;
```

$$\pi_{sname}(\sigma_{\ gpa > 2}(S) \bowtie_{id=sid} \sigma_{\ cid=CS2550}(E))$$

Transform into Equivalent Query Tree


Π M.title, M.M_id, B.Last, B.first, B.age
σ M.year = "1990" AND M.M_id=D.M_id AND D.D_id=B.D_id
X
X
B
D
M
Π M.title, M.M_id, B.Last, B.first, B.age
⋈ D.D_id=B.D_id
⋈ M.M_id=D.M_id
Π B.D_id, B.last, B.first, B.age
B
Π M.tile, M.M_id
σ M.year = "1990"
M
Π D.M_id, D.D_id
D

Initial Query Plan

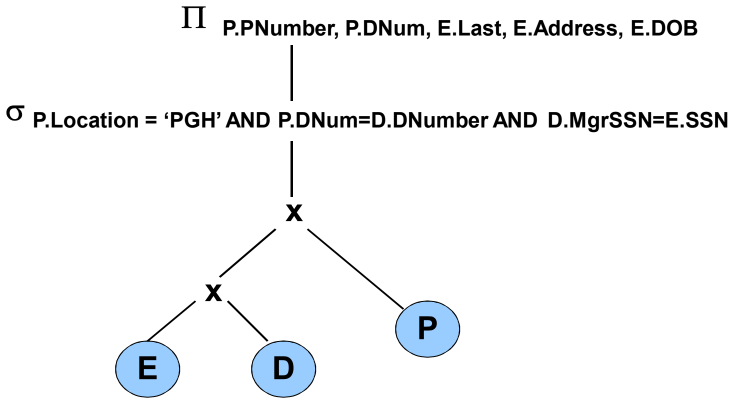
Transform

Equivalent Query Plan

Outline of algebraic optimization

1. Break up selections (with conjunctive conditions) into a cascade of selection operators
2. Push selection operators as far down in the tree as possible
3. Convert Cartesian products into joins
4. Rearrange leaf nodes to:
   - Execute first the most restrictive select operators
     - What is restrictive? (Fewest tuples or Smallest size)
   - Make sure we don’t have Cartesian products
5. Move projections as far down as possible
6. Identify subtrees that represent groups of operations which can be executed by single algorithm

Example of Query Tree



Π P.PNumber, P.DNum, E.Last, E.Address, E.DOB

σ P.Location = ‘PGH’ AND P.DNum=D.DNumber AND D.MgrSSN=E.SSN

x

x

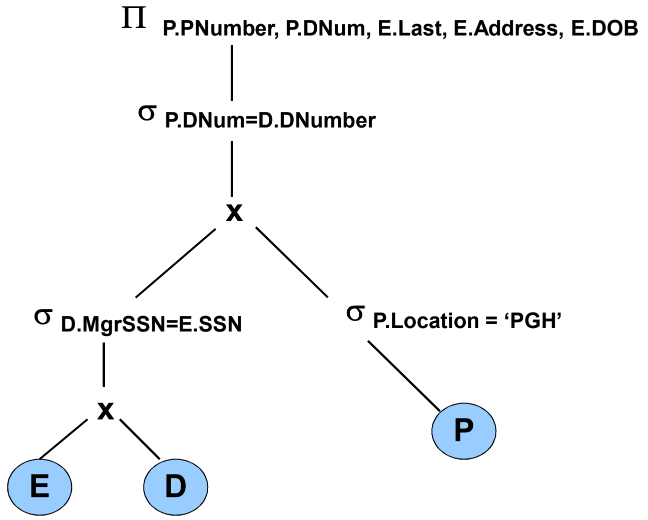
P

E

D

-
  ```
  Select P.PNumber, P.DNum, E.Last, E.Address, E.DOB
  From   Employee as E, Department as D, Project as P,
  Where  P.DNum = D.DNumber and D.MgrSSN = E.SSN and
         P.Location=‘PGH’
  ```

Example of Query Tree (Step 2)



Π P.PNumber, P.DNum, E.Last, E.Address, E.DOB

σ P.DNum=D.DNumber

x

σ D.MgrSSN=E.SSN

σ P.Location = ‘PGH’

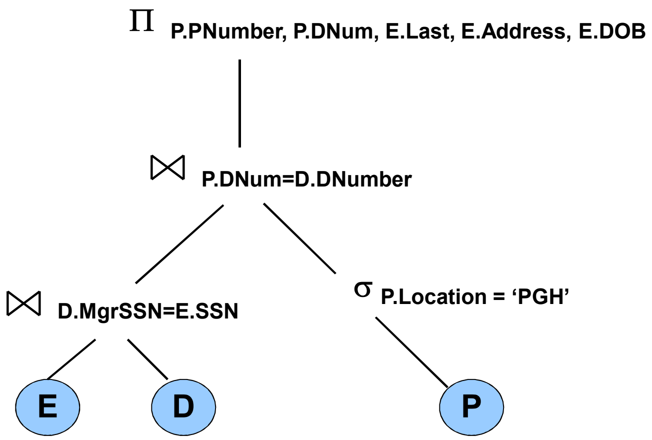
x

P

E

D

Example of Query Tree (Step 3)



Π P.PNumber, P.DNum, E.Last, E.Address, E.DOB

⋈ P.DNum=D.DNumber

⋈ D.MgrSSN=E.SSN

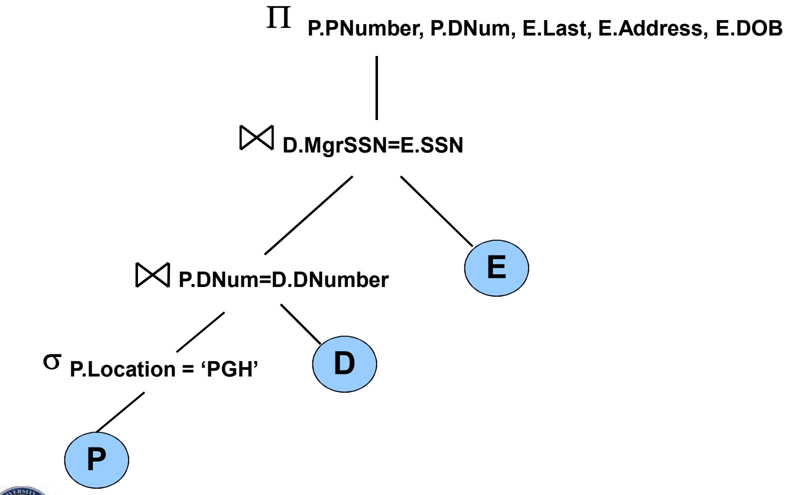
σ P.Location = ‘PGH’

E

D

P

Example of Query Tree (Step 4)



Π P.PNumber, P.DNum, E.Last, E.Address, E.DOB

⋈ D.MgrSSN=E.SSN

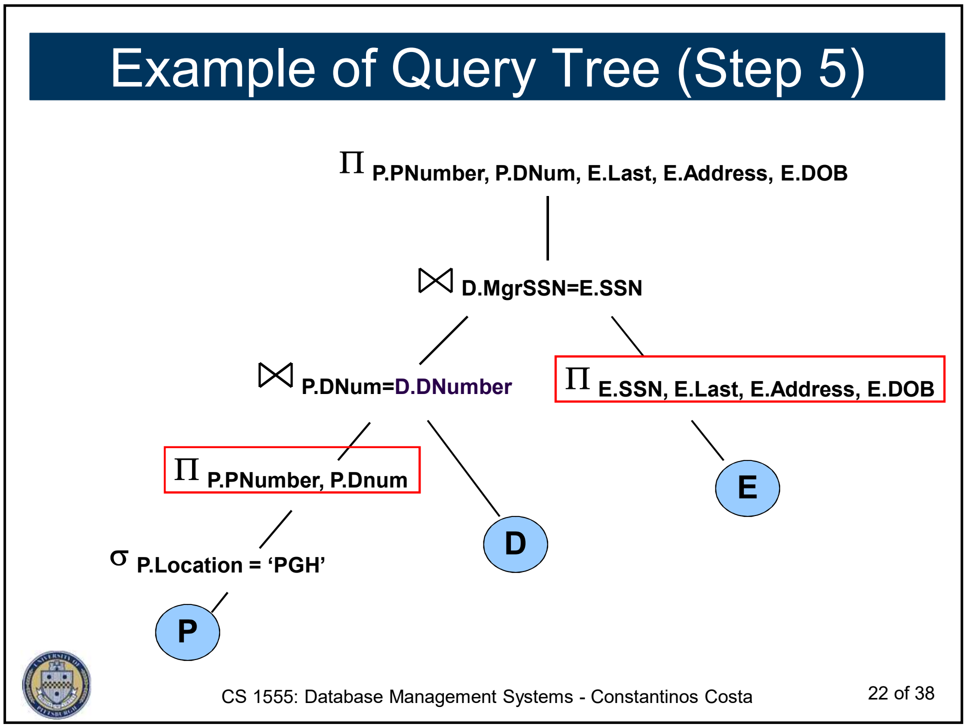
⋈ P.DNum=D.DNumber

E

σ P.Location = ‘PGH’

D

P



Example of Query Tree (Step 5)

Π P.PNumber, P.DNum, E.Last, E.Address, E.DOB

$\bowtie$ D.MgrSSN=E.SSN

$\bowtie$ P.DNum=D.DNumber

Π E.SSN, E.Last, E.Address, E.DOB

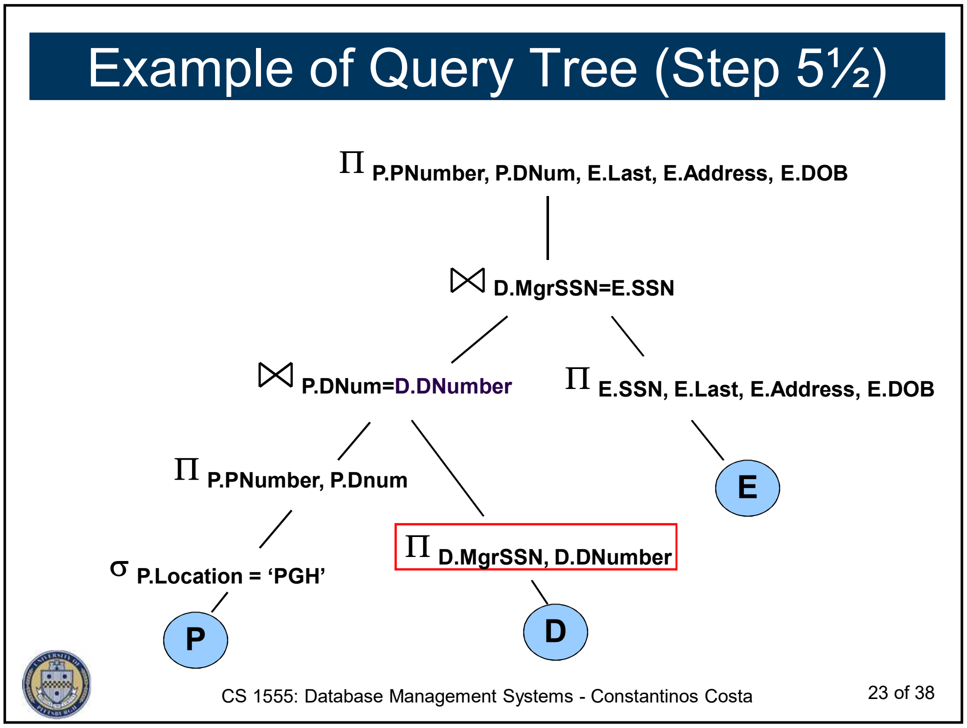
Π P.PNumber, P.Dnum

E

D

σ P.Location = ‘PGH’

P



Example of Query Tree (Step 5½)

Π P.PNumber, P.DNum, E.Last, E.Address, E.DOB

$\bowtie$ D.MgrSSN=E.SSN

$\bowtie$ P.DNum=D.DNumber

Π E.SSN, E.Last, E.Address, E.DOB

Π P.PNumber, P.Dnum

E

Π D.MgrSSN, D.DNumber

σ P.Location = ‘PGH’

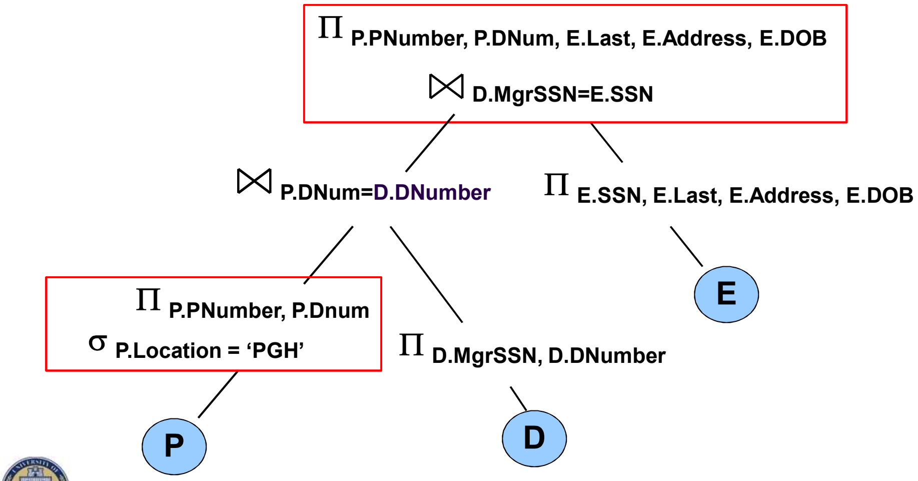
D

P

Query Execution Plans

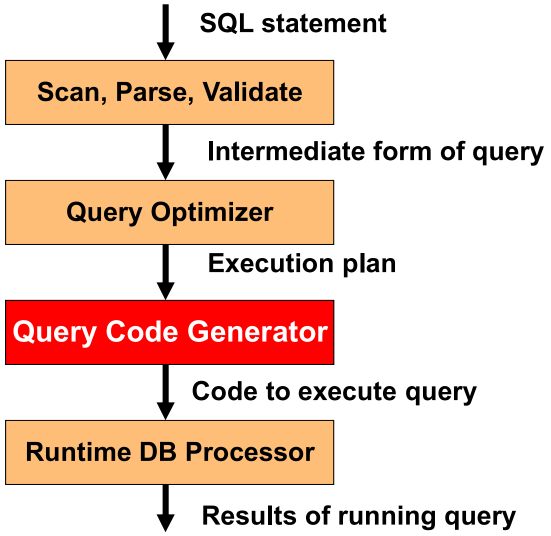
- Query tree annotated with algorithms and access methods
- Q: Is single-operator-at-a-time appropriate?
  - **Materialized evaluation or processing**
  - A: **NO**. Why?
  - A: We would need temporary relations (and extra disk space) to store intermediate results
- What is the alternative?
  - **Pipelining or Stream-based Processing**
  - Combine operators (and their execution) into a sequence

Example of Query Tree (Step 6)



Π P.PNumber, P.DNum, E.Last, E.Address, E.DOB
$\bowtie$ D.MgrSSN=E.SSN
$\bowtie$ P.DNum=D.DNumber
Π E.SSN, E.Last, E.Address, E.DOB
E
Π P.PNumber, P.Dnum
σ P.Location = ‘PGH’
Π D.MgrSSN, D.DNumber
P
D

Steps in Processing a Query



SQL statement

Scan, Parse, Validate

Intermediate form of query

Query Optimizer

Execution plan

Query Code Generator

Code to execute query

Runtime DB Processor

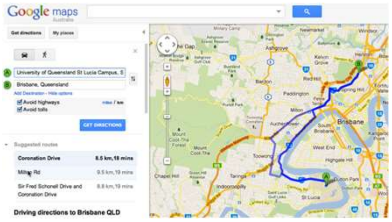
Results of running query

Operator Evaluation

- Several alternative algorithms are available for **implementing** each relational operator
  - No algorithm is universally superior
  - Several factors influence which algorithm performs best:
    - Size of the tables
    - Existing index and sort orders
    - Size of available memory
    - ….

How to implement Selection (σ)?

- Depends on:
  - **Type of Query:**
    1. Point query,
    2. Range query,
    3. Conjunction, or
    4. Disjunction



  - **Type of available Access Path:**
    1. Index,
    2. Sorted File, or
    3. None!

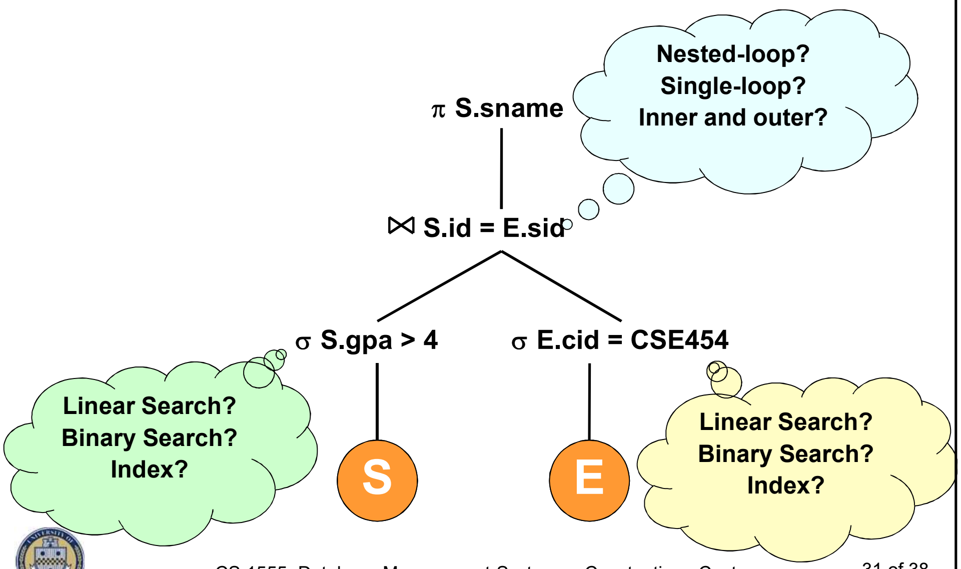
Selectivity

- The optimizer uses **selectivity** to choose!
- **Selectivity (S)**:
  - The **ratio** of: the number of records that satisfy a condition to the total number of records
  - Example: for $\sigma_{\text{type="Drama"}}$, **S** = 500K/2M = 0.25
  - ***S*** between 0.0 and 1.0
    - ***S* = 0:** no records satisfy the condition
    - ***S* = 1:** all records satisfy the condition
  - Selectivity estimates are stored in DBMS **Catalog**

Selectivity

- The actual distribution of selectivity is kept in the catalog in the form of a **histogram**
  - To get more accurate estimate of the number of records that satisfy a particular condition
    - Drama: 500,000
    - Comedy: 460,000
    - Documentary: 200,000
    - Musicals: 20,000
- Apply the condition with smallest estimate first

Putting it together



π S.sname

⋈ S.id = E.sid

σ S.gpa > 4

σ E.cid = CSE454

S

E

Nested-loop?
Single-loop?
Inner and outer?

Linear Search?
Binary Search?
Index?

Linear Search?
Binary Search?
Index?