CMU SCS

Carnegie Mellon Univ.
Dept. of Computer Science
15-415/615 - DB Applications

C. Faloutsos – A. Pavlo

Lecture#15: Query Optimization

CMU SCS

Administrivia

- HW5 is due **Wed Oct 28th**.
- Mid-terms are available for **viewing**
  - Marilyn Walgora's office (GHC 8120)
  - 9:00am-11:45am and 1:00pm-4:15pm
  - Bring your CMU id

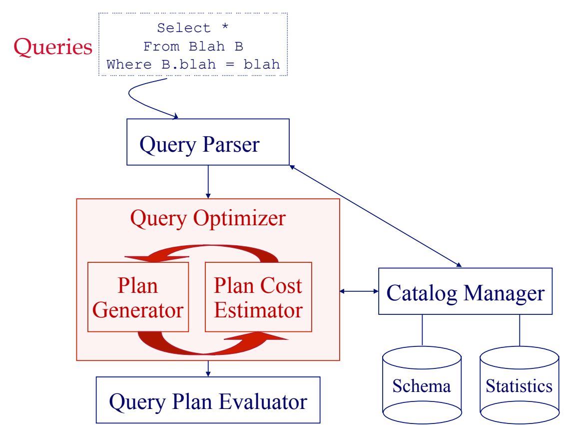
CMU SCS

Today's Class

- History & Background
- Relational Algebra Equivalences
- Plan Cost Estimation
- Plan Enumeration

CMU SCS

Cost-based Query Sub-System



CMU SCS

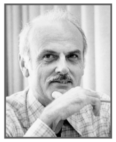
Query Optimization

- Remember that SQL is declarative.
  - User tells the DBMS *what* answer they want, not *how* to get the answer.
- There can be a big difference in performance based on plan is used:
  - See last week: 5.7 days vs. 45 seconds

Faloutsos/Pavlo CMU SCS 15-415/615 5

CMU SCS

1970s – Relational Model


Codd

- Ted Codd saw the maintenance overhead for IMS/Codasyl.
- Proposed database abstraction based on relations:
  - Store database in simple data structures.
  - Access it through high-level language.
  - Physical storage left up to implementation.

Faloutsos/Pavlo CMU SCS 15-415/615 6

CMU SCS

IBM System R

- Skunkworks project at IBM Research in San Jose to implement Codd's ideas.
- Had to figure out all of the things that we are discussing in this course themselves.
- IBM never commercialized System R.

Faloutsos/Pavlo CMU SCS 15-415/615 7

CMU SCS

IBM System R

- First implementation of a query optimizer.
- People argued that the DBMS could never choose a query plan better than what a human could write.
- A lot of the concepts from System R's optimizer are still used today.

Faloutsos/Pavlo CMU SCS 15-415/615 8

CMU SCS

Today's Class

- History & Background
- Relational Algebra Equivalences
- Plan Cost Estimation
- Plan Enumeration
- Nested Sub-queries

CMU SCS

Relational Algebra Equivalences

- A query can be expressed in different ways.
- The optimizer considers variations and choose the one with the lowest cost.
- How do we know whether two queries are equivalent?

CMU SCS

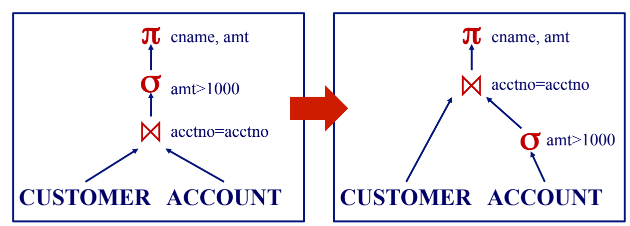
Relational Algebra Equivalences

- Two relational algebra expressions are ***equivalent*** if they generate the same set of tuples.

CMU SCS

Predicate Pushdown

```
SELECT cname, amt
  FROM customer, account
 WHERE customer.acctno =
       account.acctno
   AND account.amt > 1000
```



Relational Algebra Equivalences

```
SELECT cname, amt
 FROM customer, account
WHERE customer.acctno =
      account.acctno
  AND account.amt > 1000
```

$$\pi_{\text{cname, amt}}(\sigma_{\text{amt>1000}}(\text{customer} \bowtie \text{account}))$$

$$=$$

$$\pi_{\text{cname, amt}}(\text{customer} \bowtie (\sigma_{\text{amt>1000}}(\text{account}))$$

Relational Algebra Equivalences

- **Selections:**
  - Perform them early
  - Break a complex predicate, and push

$$\sigma_{p1\wedge p2\wedge \ldots pn}(R) = \sigma_{p1}(\sigma_{p2}(\ldots\sigma_{pn}(R))\ldots)$$

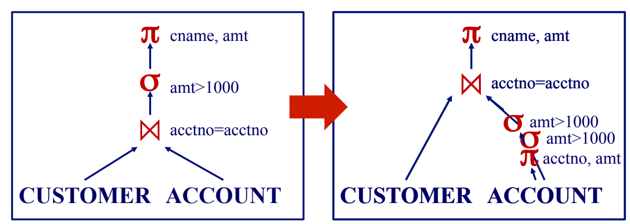
- Simplify a complex predicate
  - **(X=Y AND Y=3)** → **X=3 AND Y=3**

Relational Algebra Equivalences

- **Projections:**
  - Perform them early
    - Smaller tuples
    - Fewer tuples (if duplicates are eliminated)
  - Project out all attributes except the ones requested or required (e.g., joining attr.)

Projection Pushdown

```
SELECT cname, amt
 FROM customer, account
WHERE customer.acctno =
      account.acctno
  AND account.amt > 1000
```



CMU SCS

Relational Algebra Equivalences

- **Joins:**
  - Commutative, associative

$$R \bowtie S = S \bowtie R$$
$$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$$

- Q: How many different orderings are there for an *n*-way join?

Faloutsos/Pavlo CMU SCS 15-415/615 17

CMU SCS

Relational Algebra Equivalences

- **Joins:** How many different orderings are there for an n-way join?
- A: Catalan number ~ 4^n
  - Exhaustive enumeration: too slow.
- We'll see in a second how an optimizer limits the search space...

Faloutsos/Pavlo CMU SCS 15-415/615 18

CMU SCS

Today's Class

- History & Background
- Relational Algebra Equivalences
- Plan Cost Estimation
- Plan Enumeration

Faloutsos/Pavlo CMU SCS 15-415/615 19

CMU SCS

Cost Estimation

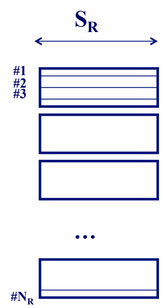
- How long will a query take?
  - **CPU**: Small cost; tough to estimate
  - **Disk**: # of block transfers
  - **Memory**: Amount of DRAM used
  - **Network**: # of messages
- How many tuples will qualify?
- What statistics do we need to keep?

Faloutsos/Pavlo CMU SCS 15-415/615 20

CMU SCS

Cost Estimation – Statistics

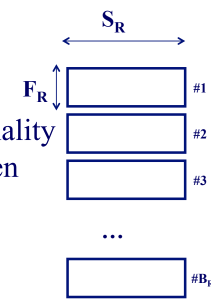
- For each relation **R** we keep:
  - $\mathbf{N_R}$ → # tuples
  - $\mathbf{S_R}$ → size of tuple in bytes
  - **V(A,R)** → # of distinct values of attribute ‘A’



CMU SCS

Derivable Statistics

- $\mathbf{F_R}$ → max# records/block
- $\mathbf{B_R}$ → # blocks
- **SC(A,R)** → selection cardinality avg# of records with A=given



CMU SCS

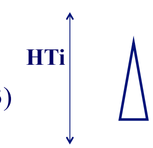
Derivable Statistics

- **SC(A,R)** → Selection Cardinality
  avg# of records with A=given
  → $\mathbf{N_R / V(A,R)}$
- Note that this assumes data uniformity
  - 10,000 students, 10 colleges – how many students in SCS?

CMU SCS

Additional Statistics

- For index **i**:
  - $\mathbf{F_i}$ → average fanout (~50-100)
  - $\mathbf{HT_i}$ → # levels of index **i** (~2-3)
    ~ log(#entries)/log($\mathbf{F_i}$)
  - $\mathbf{LB_i}$ # → blocks at leaf level



CMU SCS

Statistics

- Where do we store them?
- How often do we update them?

- Manual invocations:
  - Postgres: **ANALYZE**
  - MySQL: **ANALYZE TABLE**

CMU SCS

Selection Statistics

- We saw simple predicates (**name="Trump"**)
- How about more complex predicates, like
  - **salary > 10000**
  - **age=30 AND jobTitle="Costermonger"**
- What is their selectivity?

CMU SCS

Selections – Complex Predicates

- Selectivity **sel(P)** of predicate **P**:

  == fraction of tuples that qualify

  sel(P) = SC(P) / N_R

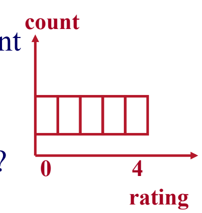
  # of tuples

  Selection Cardinality

CMU SCS

Selections – Complex Predicates

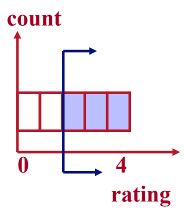
- Assume that **V(rating, SAILORS)** has 5 distinct values (i.e., 0 to 4).
- simple predicate **P**: A=constant
  - **sel(A=constant)** = **1/V(A,R)**
  - eg., **sel(rating='2')** = 1/5
- What if **V(A,R)** is unknown??



CMU SCS

Selections – Complex Predicates

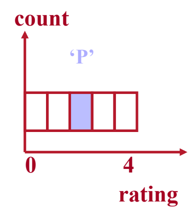
- Range Query: **sel(**rating >= ‘2’**)**
- **sel(**A>a) = **(A$_{max}$ – a) / (A$_{max}$ – A$_{min}$)**



CMU SCS

Selections – Complex Predicates

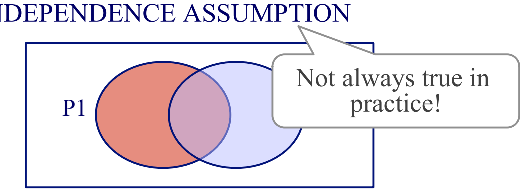
- Negation: **sel(**rating != ‘2’**)**
  - **sel(not** P) = **1 – sel(**P)
- Observation: selectivity ≈ probability



CMU SCS

Selections – Complex Predicates

- **Conjunction:**
  - **sel(**rating = ‘2’ **and** name LIKE ‘C%’**)**
  - **sel(**P1 ∧ P2**) = sel(**P1) **· sel(**P2**)**
  - INDEPENDENCE ASSUMPTION



CMU SCS

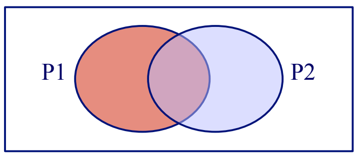
Selections – Complex Predicates

- **Disjunction**:
  - **sel(**rating = ‘2’ **or** name LIKE ‘C%’**)**
  - **sel(**P1 ∨ P2**)**
    = **sel(**P1) + **sel(**P2) – **sel(**P1 ∧ P2**)**
    = **sel(**P1) + **sel(**P2) – **sel(**P1) **· sel(**P2**)**
  - INDEPENDENCE ASSUMPTION, again

CMU SCS

Selections – Complex Predicates

- **Disjunction, in general**:
  - **sel**(P1 **or** P2 **or** … Pn) =
  - 1 - (1- **sel**(P1)) · (1 - **sel**(P2)) · … (1 - **sel**(Pn))



Faloutsos/Pavlo CMU SCS 15-415/615 33

CMU SCS

Joins

- Q: Given a join of **R** and **S**, what is the range of possible result sizes in #of tuples?

Faloutsos/Pavlo CMU SCS 15-415/615 34

CMU SCS

Result Size Estimation for Joins

- General case: $\mathbf{R_{cols}} \cap \mathbf{S_{cols}}$ = {A} where A is not a key for either table.
- *Hint: for a given tuple of R, how many tuples of S will it match?*

Faloutsos/Pavlo CMU SCS 15-415/615 35

CMU SCS

Result Size Estimation for Joins

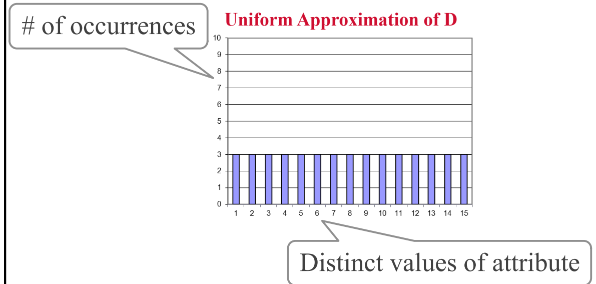
- General case: $\mathbf{R_{cols}} \cap \mathbf{S_{cols}}$ = {A} where A is not a key for either table.
  - Match each **R**-tuple with **S**-tuples: estSize ≈ $\mathbf{N_R \cdot N_S / V(A,S)}$
  - Symmetrically, for **S**: estSize ≈ $\mathbf{N_R \cdot N_S / V(A,R)}$
- Overall:
  - estSize ≈ $\mathbf{N_R \cdot N_S / max(\{V(A,S), V(A,R)\})}$

Faloutsos/Pavlo CMU SCS 15-415/615 36

CMU SCS

Cost Estimations

- Our formulas are nice but we assume that data values are uniformly distributed.

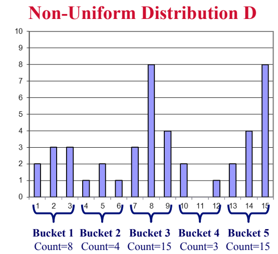


37

CMU SCS

Cost Estimations

- Our formulas are nice but we assume that data values are uniformly distributed.

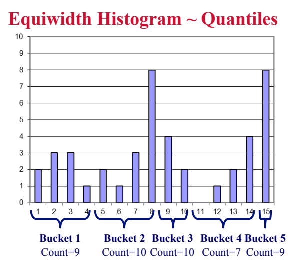


38

CMU SCS

Histograms w/ Quantiles

- Allows the DBMS to have leverage better statistics about the data.



CMU SCS

Today's Class

- History & Background
- Relational Algebra Equivalences
- Plan Cost Estimation
- Plan Enumeration

Faloutsos/Pavlo CMU SCS 15-415/615 40

CMU SCS

Query Optimization

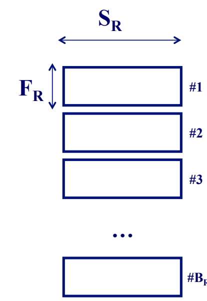
- Bring query in internal form into "canonical form" (syntactic q-opt)
- Generate alternative plans.
  - Single relation.
  - Multiple relations.
  - Nested sub-queries.
- Estimate cost for each plan.
- Pick the best one.

CMU SCS

Plan Generation

```
SELECT *
 FROM SAILORS
WHERE rating = 10
```

- What are our plan options?

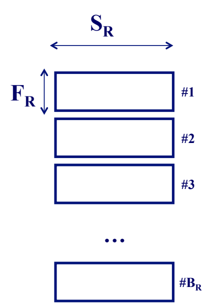


CMU SCS

Plan Generation

```
SELECT *
 FROM SAILORS
WHERE rating = 10
```

- Sequential Scan
- Binary Search
  - *if sorted & consecutive*
- Index Search
  - *if an index exists*

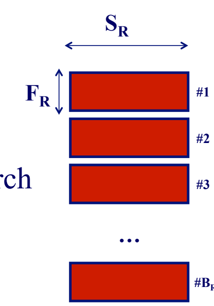


CMU SCS

Sequential Scan

```
SELECT *
 FROM SAILORS
WHERE rating = 10
```

- **B_R** (worst case)
- **B_R /2** (on average, if we search for primary key)

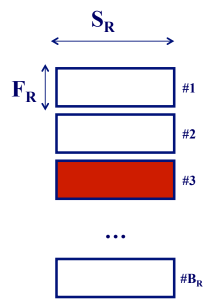


CMU SCS

Binary Search

```
SELECT *
 FROM SAILORS
WHERE rating = 10
```

- **~log(B_R) + SC(A,R)/ F_R**
- Extra blocks are ones that contain qualifying tuples



Faloutsos/Pavlo CMU SCS 15-415/615 45

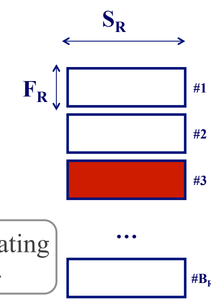
CMU SCS

Binary Search

```
SELECT *
 FROM SAILORS
WHERE rating = 10
```

- **~log(B_R) + SC(A,R)/ F_R**
- Extra blocks are ones that contain qualifying tuples

We showed that estimating this is non-trivial.



Faloutsos/Pavlo CMU SCS 15-415/615 46

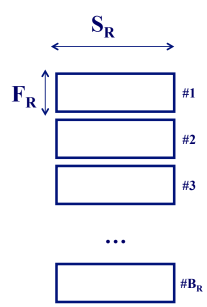
CMU SCS

Index Search

```
SELECT *
 FROM SAILORS
WHERE rating = 10
```

- Index Search:
  - levels of index + blocks w/ qual. tuples

Case#1: Primary Key
Case#2: Secondary key – clustering index
Case#3: Secondary key – non-clust. index



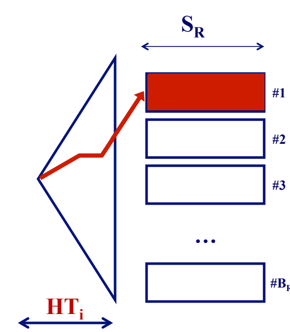
Faloutsos/Pavlo CMU SCS 15-415/615 47

CMU SCS

Index Search: Case #1

```
SELECT *
 FROM SAILORS
WHERE rating = 10
```

- Primary Key
  - **cost: HT_i + 1**



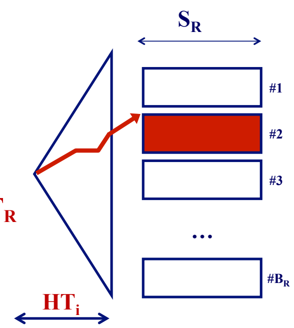
Faloutsos/Pavlo CMU SCS 15-415/615 48

CMU SCS

Index Search: Case #2

SELECT *
FROM SAILORS
WHERE rating = 10

- Secondary key with clustering index:
  - **cost: $HT_i + SC(A,R)/F_R$**

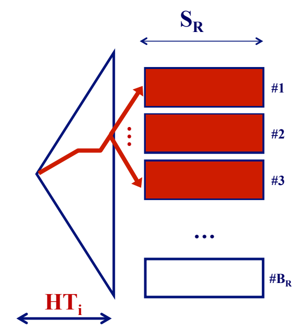


CMU SCS

Index Search: Case #3

SELECT *
FROM SAILORS
WHERE rating = 10

- Secondary key with non-clustering index:
  - **cost: $HT_i + SC(A,R)$**



CMU SCS

Query Optimization

- Bring query in internal form into "canonical form" (syntactic q-opt)
- Generate alternative plans.
  - Single relation.
  - Multiple relations.
  - Nested sub-queries.
- Estimate cost for each plan.
- Pick the best one.

CMU SCS

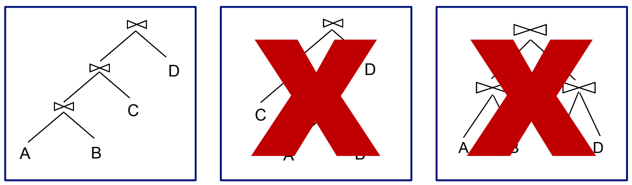
Queries over Multiple Relations

- As number of joins increases, number of alternative plans grows rapidly
  - We need to restrict search space.
- **Fundamental decision in System R:** only left-deep join trees are considered.

CMU SCS

Queries over Multiple Relations

- **Fundamental decision in System R:** only left-deep join trees are considered.



CMU SCS

Queries over Multiple Relations

- **Fundamental decision in System R:** only left-deep join trees are considered.
  - Allows for fully pipelined plans where intermediate results not written to temp files.
  - Not all left-deep trees are fully pipelined.

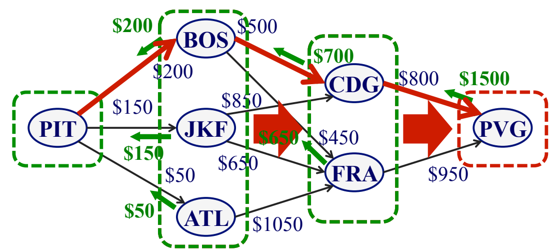
CMU SCS

Queries over Multiple Relations

- **Enumerate the orderings**
  - *Example: Left-deep tree #1, Left-deep tree #2…*
- **Enumerate the plans for each operator**
  - *Example: Hash, Sort-Merge, Nested Loop…*
- **Enumerate the access paths for each table**
  - *Example: Index #1, Index #2, Seq Scan…*
- Use **dynamic programming** to reduce the number of cost estimations.

CMU SCS

Dynamic Programming Example



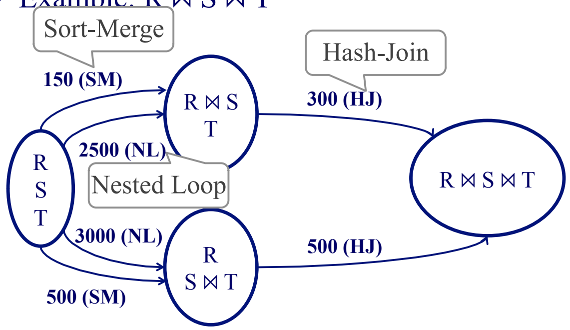
Compute the cheapest flight PIT -> PVG

Solution: Compute partial optimal, left-to-right

CMU SCS

Q-Opt + Dynamic Programming

- Example: R $\bowtie$ S $\bowtie$ T



CMU SCS

Q-Opt + Dynamic Programming

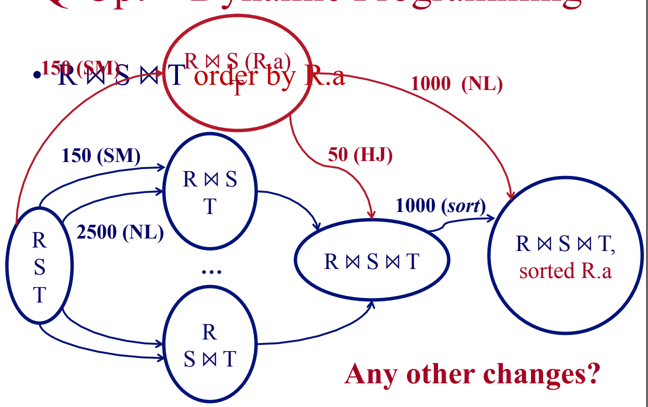
- How to plan a query where **R** is sorted on **R.a**?
- Consider the following query:

```
SELECT *
 FROM R, S, T
WHERE R.a = S.a AND S.b = T.b
ORDER BY R.a
```


CMU SCS

Q-Opt + Dynamic Programming

- R $\bowtie$ S $\bowtie$ T order by R.a



Any other changes?

CMU SCS

Candidate Plan Example

```
SELECT sname, bname, day
 FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
```

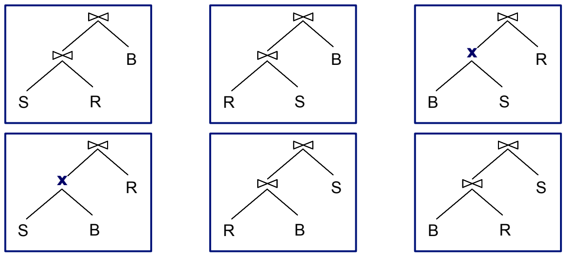
1. Enumerate relation orderings
2. Enumerate join algorithm choices
3. Enumerate access method choices

CMU SCS

Candidate Plans

```
SELECT sname, bname, day
 FROM Sailors S, Reserves R, Boats B
 WHERE S.sid = R.sid AND R.bid = B.bid
```

1. Enumerate relation orderings:



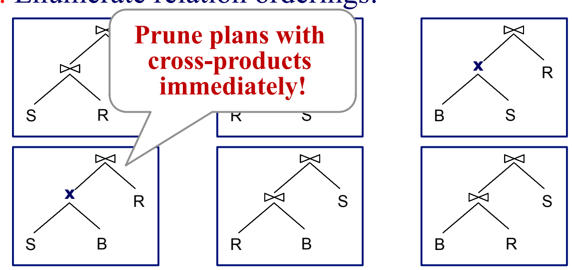
Faloutsos/Pavlo CMU SCS 15-415/615 61

CMU SCS

Candidate Plans

```
SELECT sname, bname, day
 FROM Sailors S, Reserves R, Boats B
 WHERE S.sid = R.sid AND R.bid = B.bid
```

1. Enumerate relation orderings:



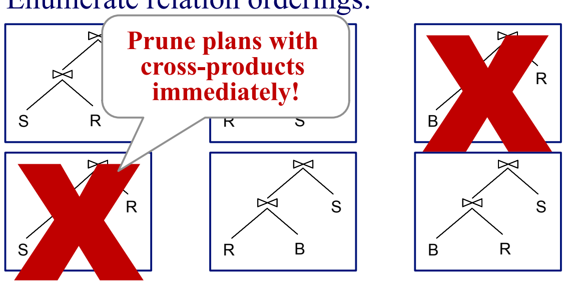
Faloutsos/Pavlo CMU SCS 15-415/615 62

CMU SCS

Candidate Plans

```
SELECT sname, bname, day
 FROM Sailors S, Reserves R, Boats B
 WHERE S.sid = R.sid AND R.bid = B.bid
```

1. Enumerate relation orderings:



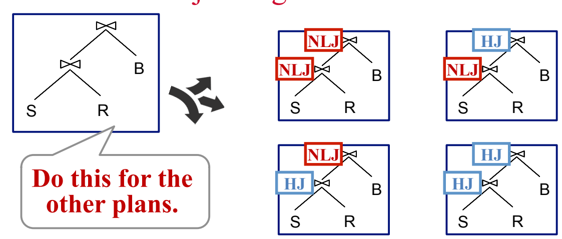
Faloutsos/Pavlo CMU SCS 15-415/615 63

CMU SCS

Candidate Plans

```
SELECT sname, bname, day
 FROM Sailors S, Reserves R, Boats B
 WHERE S.sid = R.sid AND R.bid = B.bid
```

2. Enumerate join algorithm choices:



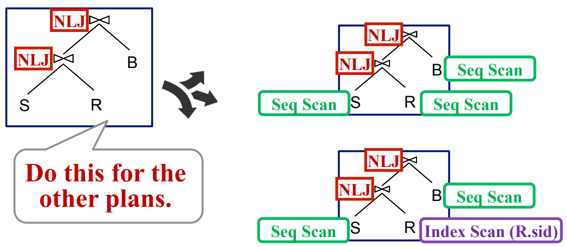
Faloutsos/Pavlo CMU SCS 15-415/615 64

CMU SCS

Candidate Plans

```
SELECT sname, bname, day
 FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
```

3. Enumerate access method choices:

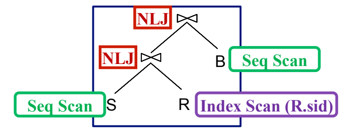


CMU SCS

Candidate Plans

```
SELECT sname, bname, day
 FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
```

4. Now we can estimate the cost of each plan.



CMU SCS

Query Optimization

- Bring query in internal form into “canonical form” (syntactic q-opt)
- Generate alternative plans.
  - Single relation.
  - Multiple relations.
  - Nested sub-queries.
- Estimate cost for each plan.
- Pick the best one.

CMU SCS

Nested Sub-Queries

- The DBMS treats nested sub-queries in the where clause as functions that take parameters and return a single value or set of values.
- Two Approaches:
  - Rewrite to de-correlate and/or flatten them
  - Decompose nested query and store result to temporary table

CMU SCS

Nested Sub-Queries: Rewrite

```
SELECT name FROM Sailors AS S
 WHERE EXISTS (
   SELECT * FROM Reserves AS R
    WHERE S.sid = R.sid
      AND R.day = "2015-10-18"
 )
```

```
SELECT name
  FROM Sailors AS S, Reserves AS R
 WHERE S.sid = R.sid
   AND R.day = "2015-10-18"
```


CMU SCS

Nested Sub-Queries: Decompose

```
SELECT S.sid, MIN(R.day)
  FROM Sailors S, Reserves R, Boats B
 WHERE S.sid = R.sid
   AND R.bid = B.bid
   AND B.color = 'red'
   AND S.rating = (SELECT MAX(S2.rating)
                     FROM Sailors S2)
 GROUP BY S.sid
HAVING COUNT(*) > 1
```

For each sailor with the highest rating (over all sailors) and at least two reservations for red boats, find the sailor id and the earliest date on which the sailor has a reservation for a red boat.

CMU SCS

Decomposing Queries

- For harder queries, the optimizer breaks up queries into blocks and then concentrates on one block at a time.
- Sub-queries are written to a temporary table that are discarded after the query finishes.

CMU SCS

Decomposing Queries

```
SELECT S.sid, MIN(R.day)
  FROM Sailors S, Reserves R, Boats B
 WHERE S.sid = R.sid
   AND R.bid = B.bid
   AND B.color = 'red'
   AND S.rating = (SELECT MAX(S2.rating)
                     FROM Sailors S2)
 GROUP BY S.sid
HAVING COUNT(*) > 1
```

Outer Block **Nested Block**

What Optimizers are Still Bad At

- Cardinality estimations are still hard.
- Problem Areas:
  - Prepared Statements
  - Correlated Columns
  - Plan Stability

More Information: Guy Lohman, *Is Query Optimization a "Solved" Problem?* SIGMOD Blog (April 2014), http://wp.sigmod.org/?p=1075

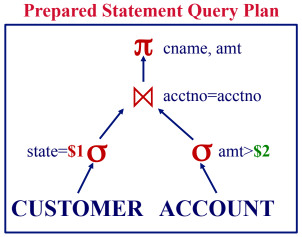
Prepared Statements

```
PREPARE myQuery (varchar, int) AS
SELECT cname, amt
 FROM customer AS C, account AS A
 WHERE C.acctno = A.acctno
  AND C.state = $1
  AND A.amt > $2
```

```
EXECUTE myQuery ('PA', 1000);
```


Prepared Statements

Prepared Statement Query Plan



- What should be the join order for **CUSTOMER** and **ACCOUNT**?

Prepared Statements

- **Solution #1** – Rerun optimizer each time the query is invoked.
- **Solution #2** – Generate multiple plans for different values of the parameters.
- **Solution #3** – Choose the average value for a parameter and use that for all invocations.

CMU SCS

Correlated Columns

- We showed how selectivities are modeled as probabilities on whether a predicate on any given row will be satisfied.
- We then multiply these individual selectivities together.

Faloutsos/Pavlo CMU SCS 15-415/615 77

CMU SCS

Correlated Columns

- Consider a database of automobiles:
  - # of Makes = 10, # of Models = 100
- And the following query:
  - **make="Honda" AND model="Accord"**
- With the independence and uniformity assumption, the selectivity is:
  - **1/10 * 1/100 = 0.001**
- But since only Honda makes Accords the real selectivity is **1/100 = 0.01**.

78

CMU SCS

Column Group Statistics

- Tell the DBMS that it should keep track of statistics for groups of columns together rather than just treating them all as independent variables.
- Only supported in commercial systems.

Faloutsos/Pavlo CMU SCS 15-415/615 79

CMU SCS

Plan Stability

- We want to deploy a new version of a DBMS but need to make sure that there are no performance regressions.
- What if 99% of the query plans are faster on the newer DBMS version, but 1% are slower?

Faloutsos/Pavlo CMU SCS 15-415/615 80

CMU SCS

Plan Stability

- **Solution #1 –** Allow tuning hints in plans.
- **Solution #2 –** Set the optimizer version number and migrate queries one-by-one to the new optimizer.
- **Solution #3 –** Save query plan from old version and provide it to the new DBMS.

CMU SCS

Query Optimizer Overview

- System R:
  - Break query in query blocks
  - Simple queries (ie., no joins): look at stats
  - n-way joins: left-deep join trees; ie., only one intermediate result at a time
    - *Pros: smaller search space; pipelining*
    - *Cons: may miss optimal*
  - 2-way joins: NL and sort-merge

CMU SCS

Conclusions

- Ideas to remember:
  - Syntactic q-opt – do selections early
  - Selectivity estimations (uniformity, indep.; histograms; join selectivity)
  - Hash join (nested loops; sort-merge)
  - Left-deep joins
  - Dynamic programming

CMU SCS

Next Class

- How to refine database schemas through normalization and functional dependencies to remove redundancies and prevent loss data.

Equivalence of Expressions

- Q: How to prove a transformation rule?
- Use relational tuple calculus to show that LHS = RHS:

$$\underbrace{\sigma_P(R1 \cup R2)}_{\textbf{LHS}} = \underbrace{\sigma_P(R1) \cup \sigma_P(R2)}_{\textbf{RHS}}$$

Equivalence of Expressions

$$\sigma_P(R1 \cup R2) = \sigma_P(R1) \cup \sigma_P(R2)$$

$$t \in LHS \Leftrightarrow$$
$$t \in (R1 \cup R2) \wedge P(t) \Leftrightarrow$$
$$(t \in R1 \vee t \in R2) \wedge P(t) \Leftrightarrow$$
$$(t \in R1 \wedge P(t)) \vee (t \in R2) \wedge P(t)) \Leftrightarrow$$

Equivalence of Expressions

$$\sigma_P(R1 \cup R2) = \sigma_P(R1) \cup \sigma_P(R2)$$

...

$$(t \in R1 \wedge P(t)) \vee (t \in R2) \wedge P(t)) \Leftrightarrow$$
$$(t \in \sigma_P(R1)) \vee (t \in \sigma_P(R2)) \Leftrightarrow$$
$$t \in \sigma_P(R1) \cup \sigma_P(R2) \Leftrightarrow$$
$$t \in RHS$$
$$QED$$

Equivalence of Expressions

- Q: How to disprove a rule?

$$\pi_A(R1 - R2) = \pi_A(R1) - \pi_A(R2)$$

R1

A	B
Trump	squirrels

R2

A	B
Trump	knifefights

A	B
Trump	squirrels

$\neq$ Ø