Database Management Systems
CS 457/657

Lectures 7: Relational Algebra (cont’d)
Logical Query Plan

SELECT city, count(*)
FROM sales
GROUP BY city
HAVING sum(price) > 100

T1, T2, T3 = temporary tables
Typical Plan for Block (1/2)

\[ \text{SELECT-PROJECT-JOIN Query} \]

\[ \pi \text{ fields} \]
\[ \sigma \text{ selection condition} \]
\[ \text{join condition} \]
\[ \text{join condition} \]
R
S
Typical Plan For Block (2/2)

\[
\begin{align*}
\text{having}_{\text{condition}} & \quad \gamma \text{ fields, sum/count/min/max}(\text{fields}) \\
\pi \text{ fields} & \quad \sigma \text{ selection condition} \\
\text{join condition} & \quad \ldots \quad \ldots
\end{align*}
\]
How about Subqueries?

```sql
SELECT  Q.sno
FROM    Supplier Q
WHERE   Q.sstate = 'WA'
        and not exists
            (SELECT *
             FROM  Supply P
             WHERE  P.sno = Q.sno
                    and P.price > 100)
```
How about Subqueries?

```
SELECT Q.sno
FROM Supplier Q
WHERE Q.sstate = 'WA'
  and not exists
  (SELECT *
   FROM Supply P
   WHERE P.sno = Q.sno
   and P.price > 100)
```

Correlation!
How about Subqueries?

```sql
SELECT  Q.sno
FROM    Supplier Q
WHERE   Q.sstate = 'WA'
        and not exists
        (SELECT *
         FROM  Supply P
         WHERE P.sno = Q.sno
         and P.price > 100)
```

De-Correlation

```sql
SELECT  Q.sno
FROM    Supplier Q
WHERE   Q.sstate = 'WA'
        and Q.sno not in
        (SELECT P.sno
         FROM  Supply P
         WHERE P.price > 100)
```
How about Subqueries?

```
(SELECT Q.sno
 FROM Supplier Q
 WHERE Q.sstate = 'WA')
EXCEPT
(SELECT P.sno
 FROM Supply P
 WHERE P.price > 100)

EXCEPT = set difference
```
How about Subqueries?

\[
\begin{align*}
\{ & \text{(SELECT } Q.\text{sno } \\
& \text{FROM Supplier } Q \\
& \text{WHERE } Q.\text{sstate = 'WA')} \\
& \text{EXCEPT} \\
& \text{(SELECT } P.\text{sno } \\
& \text{FROM Supply } P \\
& \text{WHERE } P.\text{price > 100)} \}
\end{align*}
\]

Finally…

\[
\begin{align*}
\pi_{\text{sno}} & \quad \pi_{\text{sno}} \\
\sigma_{\text{sstate='WA'}} & \quad \sigma_{\text{Price > 100}} \\
\text{Supplier} & \quad \text{Supply}
\end{align*}
\]
From Logical Plans to Physical Plans
Query Evaluation Steps Review

SQL query

Parse & Rewrite Query

Select Logical Plan

Select Physical Plan

Query Execution

Disk

Query optimization

Logical plan

Physical plan
Example

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
```

Give a relational algebra expression for this query
Relational Algebra

\[ \pi_{\text{name}}(\sigma_{\text{scity} = 'Seattle' \land \text{sstate} = 'WA' \land \text{pno} = 2} (\text{Supplier} \bowtie_{\text{sid} = \text{sid}} \text{Supply})) \]
Relational Algebra

SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
   and y.pno = 2
   and x.scity = 'Seattle'
   and x.sstate = 'WA'

Relational algebra expression is also called the "logical query plan"
Physical Query Plan 1

A physical query plan is a logical query plan annotated with physical implementation details.

SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
and y.pno = 2
and x.scity = 'Seattle'
and x.sstate = 'WA'
Physical Query Plan 2

\[
\begin{align*}
\text{SELECT} & \quad \text{sname} \\
\text{FROM} & \quad \text{Supplier } x, \text{ Supply } y \\
\text{WHERE} & \quad x.\text{sid} = y.\text{sid} \\
& \quad \text{and } y.\text{pno} = 2 \\
& \quad \text{and } x.\text{scity} = \text{‘Seattle’} \\
& \quad \text{and } x.\text{sstate} = \text{‘WA’}
\end{align*}
\]

Same logical query plan
Different physical plan

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Physical Query Plan 3

Supplier($sid$, sname, scity, sstate)
Supply($sid$, pno, quantity)

(On the fly) $\pi$ sname

(Sort-merge join) $\sigma$ scity = 'Seattle' $\land$ sstate = 'WA'

(Scan & write to T1) $\sigma$ pno = 2

(Scan & write to T2) $\sigma$ pno = 2

Different but equivalent logical query plan; different physical plan
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
Query Optimization Problem

• For each SQL query… many logical plans

• For each logical plan… many physical plans

• How do find a fast physical plan?
  – Will discuss in a few lectures
  – A lot more details, low-level system stuff
Query Execution
Pipelined Execution

• Tuples generated by an operator are immediately sent to the parent

• Benefits:
  – No operator synchronization issues
  – No need to buffer tuples between operators
  – Saves cost of writing intermediate data to disk
  – Saves cost of reading intermediate data from disk

• This approach is used whenever possible
Intermediate Tuple Materialization

• Tuples generated by an operator are written to disk an in intermediate table

• No direct benefit

• Necessary:
  – For certain operator implementations
  – When we don’t have enough memory
Intermediate Tuple Materialization

(On the fly)

(Sort-merge join)

(Scan: write to T1)

(Scan: write to T2)

\[ \pi_{\text{sname}} \]

\[ \sigma_{\text{sscity}=\text{Seattle} \land \text{sstate}=\text{WA}} \]

Suppliers
(File scan)

Supplies
(File scan)
Query Execution Bottom Line

- SQL query transformed into physical plan
  - Access path selection for each relation
    - Scan the relation or use an index (discussed later)
  - Implementation choice for each operator
    - Nested loop join, hash join, etc.
  - Scheduling decisions for operators
    - Pipelined execution or intermediate materialization
Where are we?

• We are very close to the internal details of a DBMS
  – System architecture, Storage management, Memory management

• Before digging into them, let’s spend some time on understanding how a DBMS user designs her databases
  – We already know users will use SQL to implement databases