Database Management Systems
CS 457/657

Lectures 7: Relational Algebra (cont’d), ER Diagram
Where We Are

• Motivation for using a DBMS for managing data
• SQL, SQL, SQL
  – Declaring the schema for our data (CREATE TABLE)
  – Inserting data one row at a time or in bulk (INSERT/.import)
  – Modifying the schema and updating the data (ALTER/UPDATE)
  – Querying the data (SELECT)

• Next step: More knowledge of how DBMSs work
  – Client-server architecture
  – Relational algebra and query execution
Query Evaluation Steps

- Parse & Check Query
  - Check syntax, access control, table names, etc.
  - Translate query string into internal representation
  - Logical plan → physical plan

- Decide how best to answer query: query optimization

- Query Execution
  - Return Results

- SQL query
Set Difference

- $A = \{1, 2, 3\}, B = \{2, 4\}$
- $A \cap B = \{2\}$
- $A - B = \{1, 3\}$
- $A - (A - B) = \{1, 2, 3\} - \{1, 3\} = \{2\} = A \cap B$
- **WRONG:** $A - (A - B) \neq A - A + B = B...$
Logical Query Plan

\[
\text{SELECT city, count(*)}
\text{FROM sales}
\text{GROUP BY city}
\text{HAVING sum(price) > 100}
\]

\[
\text{T1, T2, T3 = temporary tables}
\text{sales(product, city, price)}
\]

\[
\begin{align*}
\Pi_{city, c} & : T3(city, c) \\
\sigma_{p > 100} & : T2(city, p, c) \\
\gamma_{city, sum(price) \rightarrow p, count(*) \rightarrow c} & : T1(city, p, c)
\end{align*}
\]
Typical Plan for Block (1/2)

\[ \text{SELECT-PROJECT-JOIN Query} \]
Typical Plan For Block (2/2)

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

Correlation!
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)

De-Correlation

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

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?

Finally…

(\textbf{SELECT} Q.sno
\textbf{FROM} Supplier Q
\textbf{WHERE} Q.sstate = 'WA')
\textbf{EXCEPT}
(\textbf{SELECT} P.sno
\textbf{FROM} Supply P
\textbf{WHERE} P.price > 100)

\hspace{1cm}

\begin{itemize}
\item Supplier(sno,sname,scity,sstate)
\item Part(pno,pname,psize,pcolor)
\item Supply(sno,pno,price)
\end{itemize}
From Logical Plans to Physical Plans
Query Evaluation Steps Review

1. Parse & Rewrite Query
2. Select Logical Plan
3. Select Physical Plan
4. Query Execution

Query optimization

SQL query

Logical plan

Physical plan

Disk
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

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

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

(On the fly)  \[
\pi_{\text{sname}}
\]
(On the fly)  \[
\sigma_{\text{scity}=\text{Seattle} \land \text{sstate}=\text{WA} \land \text{pno}=2}
\]
(Hash join)  \[
sid = sid
\]

Same 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’
**Physical Query Plan 3**

(On the fly) \[ \pi_{\text{sname}} \]

(Sort-merge join) \[ \sigma_{\text{scity}=\text{Seattle} \land \text{sstate}=\text{WA}} \]

(Scan & write to T1) \[ \sigma_{\text{pno}=2} \]

(Scan & write to T2) \[ \text{Select sname} \] From Supplier \( x \) , Supply \( y \) Where \( x.\text{sid} = y.\text{sid} \) and \( y.\text{pno} = 2 \) and \( x.\text{scity} = \text{'Seattle'} \) and \( x.\text{sstate} = \text{'WA'} \)

Different but equivalent logical query plan; different physical plan

Supplier(\( \text{sid, sname, scity, sstate} \))

Supply(\( \text{sid, pno, quantity} \))
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 and in an 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)

Suppliers

Supplies

\( \sigma_{\text{sscity='Seattle' } \land \text{sstate='WA')} \)

\( \sigma_{\text{pno=2}} \)

\( \pi_{\text{sname}} \)

Suppliers (File scan)

Supplies (File scan)

Scan: write to T1

On the fly

Sort-merge join

(Suppliers, Supplies)
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