Database Management Systems
CS 457/657

Lectures 7: Relational Algebra (cont’d), ER Diagram
Logistics

• HW1
  – Input: standard input…
  – Programming language: C#... Object C?
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

1. SQL query
2. Parse & Check Query
   - Translate query string into internal representation
   - Check syntax, access control, table names, etc.
3. Decide how best to answer query: query optimization
4. Query Execution
5. Return Results

Logical plan $\rightarrow$ physical plan

Query Evaluation

CS 457 - Spring 2018
Set Difference

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

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

T1, T2, T3 = temporary tables

sales(product, city, price)

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

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

\[\pi \text{ fields}\]

\[\sigma \text{ selection condition}\]

\[\text{JOIN condition}\]

\[R, S\]
Typical Plan For Block (2/2)

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

\[
\text{(SELECT } Q\text{.sno} \\
\text{FROM } \text{Supplier } Q \\
\text{WHERE } Q\text{.sstate} = 'WA') \\
\text{EXCEPT} \\
\text{(SELECT } P\text{.sno} \\
\text{FROM } \text{Supply } P \\
\text{WHERE } P\text{.price} > 100) \\
\text{EXCEPT} = \text{set difference}
\]

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…

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

Disk

Query optimization

Logical plan

Physical plan

SQL query

CS 457 - Spring 2018
Example

```sql
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

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

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

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

(On the fly)  \( \pi_{sname} \)

(On the fly)  \( \sigma_{sity='Seattle' \land sstate='WA' \land pno=2} \)

(Hash join)  \( \sigma_{sid = sid} \)

Supplier (File scan)
Supply (File scan)
Supplier\((sid, sname, scity, sstate)\)
Supply\((sid, pno, quantity)\)

**Physical Query Plan 3**

(On the fly)

\(\pi_{sname}\)

(Sort-merge join)

\(sid = sid\)

(Scan & write to T1)

(a) \(\sigma_{scity = 'Seattle' \land sstate = 'WA'}\)

Supplier (File scan)

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

Supply (File scan)

(Scan & write to T2)

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)

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

\[ \sigma_{\text{pno=2}} \]

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

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
Database Management Systems
CS 457/657

Lectures 7: ER Diagram
Database Design

• Why do we need it?
  – Need a way to model real world entities in terms of relations
  – Not easy to go from real-world entities to a database schema

• Consider issues such as:
  – What entities to model
  – How entities are related
  – What constraints exist in the domain
  – How to achieve good designs

• Several formalisms exists
  – We discuss E/R diagrams
Database Design Process

- **Conceptual Model:**
  - • name
  - • product
  - • makes
  - • company
  - • price

- **Relational Model:**
  - Tables + constraints
  - And also functional dep.

- **Normalization:**
  - Eliminates anomalies
  - • Conceptual Schema

- **Physical storage details**
  - • Physical Schema
Entity / Relationship Diagrams

- Entity set = a class
  - An entity = an object

- Attribute

- Relationship

- Product

- City

- Makes
Keys in E/R Diagrams

• Every entity set must have a key

- Product
  - name
  - price
What is a Relation?

• A mathematical definition:
  – if A, B are sets, then a relation R is a subset of \( A \times B \)

• \( A=\{1,2,3\}, \quad B=\{a,b,c,d\} \)
  \( A \times B = \{(1,a),(1,b), \ldots, (3,d)\} \)
  \( R = \{(1,a), (1,c), (3,b)\} \)

• makes is a subset of Product × Company:
Multiplicity of E/R Relations

- **one-one:**
  - (name, id)

- **many-one**
  - (employee, company)

- **many-many**
  - (book, author)
What does this say?
Multi-way Relationships

• How do we model a purchase relationship between buyers, products and stores?

- Person
- Purchase
- Product
- Store

• Can still model as a mathematical set (Q. how ?)

• A. As a set of triples \( \subseteq \text{Person} \times \text{Product} \times \text{Store} \)
Q: What does the arrow mean?

A: A given person buys a given product from at most one store.

[Arrow pointing to E means that if we select one entity from each of the other entity sets in the relationship, those entities are related to at most one entity in E]
Q: What does the arrow mean?

A: A given person buys a given product from at most one store AND every store sells to every person at most one product.
Converting Multi-way Relationships to Binary

- **Purchase**
- **StoreOf**
- **BuyerOf**
- **ProductOf**
- **date**

- **Product**
- **Store**
- **Person**

**Arrows go in which direction?**
Converting Multi-way Relationships to Binary

- Purchase
- StoreOf
- BuyerOf
- date
- ProductOf

• Purchase
• StoreOf
• BuyerOf
• date
• ProductOf

• Product
• Store
• Person

Make sure you understand why!