Translating SQL to RA

*Database Systems: The Complete Book*

Ch 16, 16.1
The Evaluation Pipeline

.\[.sql\]

How does this work? (now)

Parsed Query

What does this look like? (last class)

Employee  Department

Data

How does this work? (later today?)

Results
A Basic SQL Query

SELECTION [DISTINCT] target-list

A list of attributes of relations in relation-list

FROM relation-list

A list of relation names

(possibly with a range-variable after each name)

WHERE condition


(optional) keyword indicating that the answer should not contain duplicates
SQL

- SQL is a language for querying relations
  - **SELECT** to access (query) data
    - Different features for different access patterns.
  - **INSERT INTO**, **DELETE FROM** to modify data
  - **CREATE TABLE**, **DROP TABLE**, **ALTER TABLE** to modify relations
Relational Algebra Trees

SELECT O.FirstName
FROM Officers O, Ships S
WHERE O.Ship = S.ID
    AND S.Name = 'Enterprise'

\[ \pi_{\text{FirstName}}(\text{Officers} \bowtie_{\text{Ship}=\text{ID}}(\sigma_{\text{Name}='Enterprise'}(\text{Ships}))) \]
Relational Algebra Trees

\[ \pi_{\text{FirstName}}(\text{Officers} \bowtie_{\text{Ship}=\text{ID}}(\sigma_{\text{Name}=\text{"Enterprise"}}(\text{Ships}))) \]
Relational Algebra Trees

\[ \pi_{\text{FirstName}}(\text{Officers} \bowtie_{\text{Ship} = \text{ID}} (\sigma_{\text{Name} = 'Enterprise'}(\text{Ships}))) \]
Syntax Trees in Java

```
Statement
|
Select
  [Body]
|
SelectBody
|
PlainSelect
  [SelectList, FromList, Where]
|
FromItem
|
Table
|
SubSelect
  [Body]
```

- **Member**: instance of
- **Interface**: class
Statement statement = parser.Statement();

if(statement instanceof Select) {
    Algebra raTree = parseTree((Select)statement);
    evaluate(raTree);
}
else if(statement instanceof CreateTable) {
    loadTableSchema((CreateTable)statement);
}

Syntax Trees in Java

What would a class hierarchy look like for Relational Algebra?
Syntax Trees in Java

Operator

\[ \pi \quad \sigma \quad \sqcup \quad \times \]

\[ [c] \quad [c] \quad [l,r] \quad [l,r] \]
SQL to RA

```
SELECT [DISTINCT] target
FROM source
WHERE cond1
GROUP BY ...
HAVING cond2
ORDER BY order
LIMIT lim
UNION nextselect
```
FROM Clause

FROM R, S, T, ...

What happens if I have a FROM-nested query?
FROM Clause

FROM R, (SELECT ...) S, T, ...

Selects are just relations!
FROM R JOIN S ON cond
FROM Clause

FROM R JOIN S ON cond
FROM Clause

FROM R NATURAL JOIN S
You need to be able to compute the schema of a RA operator
SQL to RA

```
SELECT [DISTINCT] target
FROM source
WHERE cond1
GROUP BY ...
HAVING cond2
ORDER BY order
LIMIT lim
UNION nextselect
```

```
\text{agg}\text{target}\ (\pi) \\
\text{cond2}\ (\sigma) \\
\text{order by} \\
\text{distinct} \\
\text{lim} \\
\text{nextselect}
```

```
\text{source}\ (\times, \bowtie) \\
\text{cond1}\ (\sigma) \\
\text{source}\ (\times, \bowtie)
```

19
SELECT (target) Clause

\[ \pi_{\text{targets}} \]

\[ \text{input} \]

\[ \begin{align*}
\text{SELECT} & \quad * \\
\text{no } \pi & \quad (\text{or target} = \text{schema(input)}) \\
\text{targets} & \quad = A, B, \ldots \\
\text{SELECT} & \quad R.*, S.* \\
\text{targets} & \quad = \text{schema(input)} \text{ from } R, S
\end{align*} \]

Schemas need both Table Alias & Attribute Name
(see Column class)
SQL to RA

```
SELECT [DISTINCT] target
FROM source
WHERE cond1
GROUP BY ...
HAVING cond2
ORDER BY order
LIMIT lim
UNION nextselect
```

```
U
```

```
distinct
```

```
order by
```

```
target (π)
```

```
cond2 (σ)
```

```
agg
```

```
cond1 (σ)
```

```
source (⋈, ×)
```

```
nextselect
```

```
lim
```
CREATE TABLE R(a int, b int)
CREATE TABLE S(b int, c int, d int)
CREATE TABLE T(c int, e int, f string)

SELECT R.*, T.f
FROM R, T, (  
    SELECT b, AVG(d) AS DAvg,
    SUM(c) AS CSum
    FROM S WHERE d > 10
    GROUP BY b HAVING DAvg < 20
) SAgg
WHERE R.b = SAgg.b AND T.c = SAgg.CSum
\[ \text{TR.a, R.b, T.f} \]
\[ \sigma \text{R.b = S} \land \text{T.c} = \text{S} \Rightarrow \text{Csum} \]
\[ \sigma \text{DAvg < 20} \]
\[ \sum \text{Csum: SUM}(C) \]
\[ \text{b DAvg: Avg(D)} \]
\[ S \]
… but that’s stupid!

That query will be slooooollllllooooww.
Translation is hard.

Don’t make your life harder.

Translate Dumb, Fix it in the Optimizer
Group Work

Write pseudocode translating from a non-aggregate SELECT … FROM … WHERE … to a relational algebra expression
Evaluating RA

*Database Systems: The Complete Book*

Ch 15,15.1-15.3
Project Outline

SQL Query -> Parser & Translator -> Relational Algebra

Here Lie Dragons?

Query Result
The Evaluation Pipeline

Parsed Query

- How does this work?

Data

Results

Employee  Department

.Parsed Query

.sql

Results

Data

How does this work?

(now)
Evaluation Strategies

- **Staged Evaluation**: Start at leaves, Evaluate each operator as one step.

- **Pull Model**: Tuple-at-a-time Iterator for each operator (also called *Volcano Operators*) reads from source iterator(s).

- **Push Model**: Thread-per operator reads from input buffer(s) and writes to output buffer.
Staged Evaluation

Officers \bowtie Ship=ID

\pi_{\text{FirstName}} (\sigma_{\text{Name}='Enterprise'} (\text{Ships} ))
Staged Evaluation

**Dependency**: Compute $\pi$

Compute the Output

$\pi_{\text{FirstName}}$

$\Join_{\text{Ship}=\text{ID}}$

$\sigma_{\text{Name}='\text{Enterprise}'}$

$\text{Officers}$

$\text{Ships}$
Staged Evaluation

\[ \pi \text{FirstName} \]

\[ \bowtie \text{Ship=ID} \]

\[ \sigma \text{Name='Enterprise'} \]

\[ \text{Officers} \]

\[ \text{Ships} \]

Compute the Output

**Dependency**: Compute $\pi$

**Dependency**: Compute $\bowtie$
Staged Evaluation

Compute the Output

**Dependency**: Compute \( \pi \)

**Dependency**: Compute \( \bowtie \)

**Dependency**: Compute \( \sigma \)

\[ \text{Officers} \]

\[ \pi \text{FirstName} \]

\[ \bowtie \text{Ship=ID} \]

\[ \sigma \text{Name='Enterprise'} \]

\[ \text{Ships} \]
Staged Evaluation

**Dependency**: Compute $\pi$

**Dependency**: Compute $\bowtie$

**Dependency**: Compute $\sigma$

**Dependency**: Load Ships

Compute the Output

<table>
<thead>
<tr>
<th>ID,</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1701,</td>
<td>Enterprise</td>
</tr>
<tr>
<td>[DS9,</td>
<td>Deep Space 9</td>
</tr>
<tr>
<td>[74656,</td>
<td>Voyager</td>
</tr>
<tr>
<td>[75633,</td>
<td>Defiant</td>
</tr>
</tbody>
</table>
Staged Evaluation

Compute the Output

**Dependency**: Compute \( \pi \)

**Dependency**: Compute \( \bowtie \)

**Dependency**: Compute \( \sigma \)

<table>
<thead>
<tr>
<th>ID, Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1701, Enterprise]</td>
</tr>
<tr>
<td>[DS9, Deep Space 9]</td>
</tr>
<tr>
<td>[74656, Voyager]</td>
</tr>
<tr>
<td>[75633, Defiant]</td>
</tr>
</tbody>
</table>
Staged Evaluation

Compute the Output

**Dependency**: Compute $\pi$

**Dependency**: Compute $\Join$

<table>
<thead>
<tr>
<th>ID, Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1701, Enterprise]</td>
</tr>
</tbody>
</table>
## Staged Evaluation

### Dependency: Compute $\pi$

### Dependency: Compute $\bowtie$

### Dependency: Load Officers

#### Compute the Output

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1701</td>
<td>Enterprise</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>FirstName</th>
<th>LastName</th>
<th>Rank</th>
<th>Ship</th>
</tr>
</thead>
<tbody>
<tr>
<td>1701</td>
<td>James</td>
<td>Kirk</td>
<td>4.0</td>
<td>1701</td>
</tr>
<tr>
<td>1701D</td>
<td>Jean Luc</td>
<td>Picard</td>
<td>4.0</td>
<td>1701D</td>
</tr>
<tr>
<td></td>
<td>Benjamin</td>
<td>Sisko</td>
<td>3.0</td>
<td>DS9</td>
</tr>
<tr>
<td>74656</td>
<td>Kathryn</td>
<td>Janeway</td>
<td>4.0</td>
<td>74656</td>
</tr>
<tr>
<td>75633</td>
<td>Nerys</td>
<td>Kira</td>
<td>2.5</td>
<td>75633</td>
</tr>
<tr>
<td></td>
<td>Spock</td>
<td>NULL</td>
<td>2.5</td>
<td>1701</td>
</tr>
<tr>
<td>1701D</td>
<td>William</td>
<td>Riker</td>
<td>2.5</td>
<td>1701D</td>
</tr>
<tr>
<td>75633</td>
<td>Nerys</td>
<td>Kira</td>
<td>2.5</td>
<td>DS9</td>
</tr>
<tr>
<td>74656</td>
<td>Chakotay</td>
<td>NULL</td>
<td>3.0</td>
<td>74656</td>
</tr>
</tbody>
</table>
Staged Evaluation

\[ \pi_{\text{FirstName}} \bowtie_{\text{Ship}=\text{ID}} \sigma_{\text{Name}='\text{Enterprise}'} \pi \bowtie \pi \]

Compute the Output

<table>
<thead>
<tr>
<th>ID, Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1701, Enterprise</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FirstName, LastName, Rank, Ship, ID, Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>James, Kirk, 4.0, 1701, Enterprise</td>
</tr>
<tr>
<td>Spock, NULL, 2.5, 1701, Enterprise</td>
</tr>
<tr>
<td>Jean Luc, Picard, 4.0, 1701D</td>
</tr>
<tr>
<td>Benjamin, Sisko, 3.0, DS9</td>
</tr>
<tr>
<td>Kathryn, Janeway, 4.5, 74656</td>
</tr>
<tr>
<td>Nerys, Kira, 2.5, 75633</td>
</tr>
<tr>
<td>Spock, NULL, 2.5, 1701</td>
</tr>
<tr>
<td>William, Riker, 2.5, 1701D</td>
</tr>
<tr>
<td>Nerys, Kira, 2.5, DS9</td>
</tr>
<tr>
<td>Chakotay, NULL, 3.0, 74656</td>
</tr>
</tbody>
</table>
Staged Evaluation

\[ \pi_{\text{FirstName}} (\bowtie_{\text{Ship}=\text{ID}} (\sigma_{\text{Name}='\text{Enterprise}'} (\text{Officers} \odot \text{Ships}))) \]

Compute the Output

**Dependency**: Compute \( \pi \)

<table>
<thead>
<tr>
<th>FirstName, LastName, Rank, Ship, ID, Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>James,       Kirk,        4.0, 1701, 1701, Enterprise</td>
</tr>
<tr>
<td>Spock,       NULL,        2.5, 1701, 1701, Enterprise</td>
</tr>
</tbody>
</table>
Staged Evaluation

Compute the Output

FirstName
[James]
[Spock]
Staged Evaluation

Can we do better?
Staged Evaluation

- **Expensive**: Lots of Bulk Copies
- **Cache Locality**: Repeated Scans over Full Tables
- **Memory Use**: Working Set is a Full Table (or more)

How do we do better?
The Memory Hierarchy and You

• We want to keep data as close to the CPU as possible
  • Faster memory == Smaller memory

• **Solution 1:** Minimize the **Working Set** Size!
  • (the memory used at any one time)

• **Solution 2:** Aggressively Batch & Reuse Data
Volcano Evaluation

Compute one tuple

**Diagram:**

```
πFirstName
∀Ship=ID

Officers ⊤ Name='Enterprise'
```

Ships
Volcano Evaluation

\[ \pi \text{FirstName} \quad \bowtie \quad \sigma \text{Name} = 'Enterprise' \quad \text{Ships} \]

Compute one tuple

**Dependency**: Next tuple from \( \pi \)
Volcano Evaluation

Compute one tuple

**Dependency**: Next tuple from \( \pi \)

**Dependency**: Next tuple from \( \bowtie \)

\[ \pi \text{FirstName} \]

\[ \bowtie \text{Ship=ID} \]

\[ \sigma \text{Name=‘Enterprise’} \]

\[ \text{Officers} \]

\[ \text{Ships} \]
Volcano Evaluation

Compute one tuple

**Dependency**: Next tuple from $\pi$

**Dependency**: Next tuple from $\bowtie$

**Dependency**: Next tuple from $\sigma$

$\pi$\text{FirstName} $\bowtie$ $\sigma$\text{Name='Enterprise'} $\bowtie$ $\pi$\text{Ship=ID}
Volcano Evaluation

Compute one tuple

**Dependency**: Next tuple from $\pi$

**Dependency**: Next tuple from $\bowtie$

**Dependency**: Next tuple from $\sigma$

**Dependency**: Tuple from $\text{Ships}$

ID, Name
[1701, Enterprise]
Volcano Evaluation

Compute one tuple

**Dependency**: Next tuple from $\pi$

**Dependency**: Next tuple from $\bowtie$

**Dependency**: Next tuple from $\sigma$

<table>
<thead>
<tr>
<th>ID, Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1701, Enterprise]</td>
</tr>
</tbody>
</table>
Volcano Evaluation

Compute one tuple

**Dependency**: Next tuple from π
**Dependency**: Next tuple from \( \bowtie \)

\[
\begin{align*}
\text{ID, Name} \\
[1701, \text{Enterprise}] 
\end{align*}
\]
Volcano Evaluation

Compute one tuple

**Dependency**: Next tuple from $\pi$

**Dependency**: Next tuple from $\bowtie$

**Dependency**: Tuple from $\text{Officers}$

```
ID, Name
[1701, Enterprise ]
```

```
FirstName, LastName, Rank, Ship
[James, Kirk, 4.0, 1701 ]
```
Volcano Evaluation

\[ \text{Officers } \bowtie_{\text{Name}=\text{Enterprise}} \pi_{\text{FirstName}} \]
Volcano Evaluation

Compute one tuple

**Dependency**: Next tuple from \( \pi \)

\[
\begin{align*}
\pi & \text{FirstName} \\
\bowtie & \text{Name} = \text{Enterprise} \\
\sigma & \text{Ship} = \text{ID} \\
\text{Officers} & \text{Name} = \text{Enterprise'} \\
\text{Ships} & \\
\end{align*}
\]

**Example Tuple**: [James, Kirk, 4.0, 1701, 1701, Enterprise]
Volcano Evaluation

Compute one tuple

\[
\pi_{\text{FirstName}}
\n\Join_{\text{Ship} = \text{ID}}
\n\sigma_{\text{Name} = 'Enterprise'}
\]

\text{Officers}

\text{Ships}

\text{FirstName}\n
\begin{array}{c}
\text{James}
\end{array}
void open() {
    // call open() on child iterators
    // prepare the iterator
}

Tuple getNext() {
    // read, process, and return a tuple
}

void close() {
    // clean-up the iterator
    // call close() on child iterators
}
GetNext()

Relation
Read One Line from File
Split Line into Fields
Parse Field Types
Return Tuple

What is the Working Set Size?
GetNext()

**Projection** (\(\pi\))

Read One Tuple

Compute Projected Attributes

Return Tuple

What is the Working Set Size?
GetNext()

**Selection (σ)**

- Read One Tuple
- Test Condition
  - Reject Tuple
  - Return Tuple

What is the Working Set Size?
GetNext()

Union (U)

Read One Tuple from R

R Empty?

Read One Tuple from S → Return Tuple

What is the Working Set Size?
GetNext()

Nested Loop Join/Cross ($\times$)

Read (and save) One Tuple from $R$

Read One Tuple from $S$

$S$ Empty?

Reset $S$ (Close(), Open())

Construct Joint Tuple: 
\[ < S > \circ < R > \]

What is the Working Set Size? 
but...

Return Tuple
Implementing: Joins

Solution 1 (Nested-Loop)

For Each (a in A) { For Each (b in B) { emit (a, b); }}

A

B
Implementing: Joins

Solution 2 (Block-Nested-Loop)
Implementing: Joins

Solution 2 (Block-Nested-Loop)

1) Partition into Blocks
2) NLJ on each pair of blocks