Relational Algebra Equivalencies

*Database Systems: The Complete Book*
Ch. 16.2-16.3
Implementing: Joins

Solution 1 (Nested-Loop)

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

Solution 1 (Nested-Loop)

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

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

Solution 2 (Block-Nested-Loop)

1) Partition into Blocks
Implementing: Joins

Solution 2 (Block-Nested-Loop)

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

Solution 3 (Sort-Merge Join)

Keep iterating on the set with the lowest value.
When you hit two that match, emit, then iterate both

\[
\begin{array}{c}
1 \\
2 \\
3 \\
5 \\
\end{array} \quad \begin{array}{c}
1 \\
4 \\
5 \\
6 \\
\end{array}
\]

A \quad B
Implementing: Joins

Solution 3 (Sort-Merge Join)

Keep iterating on the set with the lowest value. When you hit two that match, emit, then iterate both

A

1
2
3
5

B

1
4
5
6

5
Implementing: Joins

Solution 3 (Sort-Merge Join)

Keep iterating on the set with the lowest value. When you hit two that match, emit, then iterate both

A

B
Implementing: Joins

Solution 3 (Sort-Merge Join)

Keep iterating on the set with the lowest value. When you hit two that match, emit, then iterate both

A

1
2
3
5

B

1
4
5
6

5
Implementing: Joins

Solution 3 (Sort-Merge Join)

Keep iterating on the set with the lowest value. When you hit two that match, emit, then iterate both

A

1
2
3
5

B

1
4
5
6
Implementing: Joins

Solution 3 (Sort-Merge Join)

Keep iterating on the set with the lowest value. When you hit two that match, emit, then iterate both

A  
1  
2  
3  
5  

B  
1  
4  
5  
6  

Done!
Implementing: Joins

Solution 4 (External Hash)
Implementing: Joins

Solution 4 (External Hash)

1) Build a hash table on both relations
Implementing: Joins

Solution 4 (External Hash)

1) Build a hash table on both relations

2) In-Memory Nested-Loop Join on each hash bucket

(subdivide buckets using a different hash fn if needed)
Implementing: Joins

Solution 5 (Grace/Hybrid Hash)

Keep the hash table in memory

A

\[
\begin{array}{c}
1 \\
2 \\
3 \\
5
\end{array}
\]

(Origins)

\[
\begin{array}{c}
1 \\
4 \\
5 \\
6
\end{array}
\]

B

(Essentially a more efficient nested loop join)
Implementing: Joins

Solution 5 (Grace/Hybrid Hash)

Keep the hash table in memory

(Essentially a more efficient nested loop join)
Implementing: Joins

Solution 5 (Grace/Hybrid Hash)

Keep the hash table in memory

(Essentially a more efficient nested loop join)
Implementing: Joins

Solution 6 (Index-Nested-Loop)

Like nested-loop, but use an index to make the inner loop much faster!
Implementing: Joins

Solution 6 (Index-Nested-Loop)

Like nested-loop, but use an index to make the inner loop much faster!
Implementing: Joins

Solution 6 (Index-Nested-Loop)

Like nested-loop, but use an index to make the inner loop much faster!
Implementing: Joins

Solution 6 (Index-Nested-Loop)

Like nested-loop, but use an index to make the inner loop much faster!
Implementing: Joins

Solution 6 (Index-Nested-Loop)

Like nested-loop, but use an index to make the inner loop much faster!
Implementing: Joins

Solution 6 (Index-Nested-Loop)

Like nested-loop, but use an index to make the inner loop much faster!
What are the tradeoffs of each algorithm?

What properties do we care about?  How do the algorithms compare?
## Implementing: Joins

### Tradeoffs

<table>
<thead>
<tr>
<th>Method</th>
<th>Pipelined?</th>
<th>Memory Requirements?</th>
<th>Predicate Limitation?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested Loop</td>
<td>1/2</td>
<td>1 Table</td>
<td>No</td>
</tr>
<tr>
<td>Block-Nested Loop</td>
<td>No</td>
<td>2 ‘Blocks’</td>
<td>No</td>
</tr>
<tr>
<td>Index-Nested Loop</td>
<td>1/2</td>
<td>1 Tuple (+Index)</td>
<td>Single Comparison</td>
</tr>
<tr>
<td>Sort-Merge</td>
<td>If Data Sorted</td>
<td>Same as reqs. of Sorting Inputs</td>
<td>Equality Only</td>
</tr>
<tr>
<td>Hash</td>
<td>No</td>
<td>Max of 1 Page per Bucket and All Pages in Any Bucket</td>
<td>Equality Only</td>
</tr>
<tr>
<td>Grace Hash</td>
<td>1/2</td>
<td>Hash Table</td>
<td>Equality Only</td>
</tr>
</tbody>
</table>
```
Select

CreateTable

Saved

Schema

π σ x

R S

Iterator

CSV

PLAYERS.dat

(Output)
```
Select

CreateTable

Saved Schema

Optimizer

Iterator

PLAYERS.dat

(Output)
Equivalent Expressions

They look the same, but one is good, one is evil

(No Beard) ≠ (Beard)

(Leonard Nimoy) = (Zachary Quinto)

Two different expressions of the “same” character
Query Optimization

If X and Y are equivalent and Y is better...

... then replace all Xs with Ys
Equivalent Expressions

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; A &gt;</td>
<td>&lt; A, B &gt;</td>
</tr>
<tr>
<td>&lt; 1 &gt;</td>
<td>&lt; 2, 4 &gt;</td>
</tr>
<tr>
<td>&lt; 2 &gt;</td>
<td>&lt; 3, 5 &gt;</td>
</tr>
<tr>
<td></td>
<td>&lt; 3, 6 &gt;</td>
</tr>
</tbody>
</table>
Equivalent Expressions

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; A &gt;</td>
<td>&lt; A, B &gt;</td>
</tr>
<tr>
<td>&lt; 1 &gt;</td>
<td>&lt; 2, 4 &gt;</td>
</tr>
<tr>
<td>&lt; 2 &gt;</td>
<td>&lt; 3, 5 &gt;</td>
</tr>
<tr>
<td>&lt; 2 &gt;</td>
<td>&lt; 3, 6 &gt;</td>
</tr>
</tbody>
</table>

Is $R = S$ ?
Equivalent Expressions

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt; A &gt;)</td>
<td>(&lt; A, B &gt;)</td>
</tr>
<tr>
<td>(&lt; 1 &gt;)</td>
<td>(&lt; 2, 4 &gt;)</td>
</tr>
<tr>
<td>(&lt; 2 &gt;)</td>
<td>(&lt; 3, 5 &gt;)</td>
</tr>
<tr>
<td>(&lt; 2 &gt;)</td>
<td>(&lt; 3, 6 &gt;)</td>
</tr>
</tbody>
</table>

Is $R = S$?

Is $R = \pi_A(S)$?
Equivalent Expressions

R

< A >
< 1 >
< 2 >
< 2 >

S

< A, B >
< 2, 4 >
< 3, 5 >
< 3, 6 >

Is \( R = S \) ?

Is \( R = \pi_A(S) \) ?

Is \( R = \pi_{A \leftarrow (A-1)}(S) \) ?
Equivalent Expressions

<table>
<thead>
<tr>
<th>$R$</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt; A &gt;$</td>
<td>$&lt; A, B &gt;$</td>
</tr>
<tr>
<td>$&lt; 1 &gt;$</td>
<td>$&lt; 2, 4 &gt;$</td>
</tr>
<tr>
<td>$&lt; 2 &gt;$</td>
<td>$&lt; 3, 5 &gt;$</td>
</tr>
<tr>
<td>$&lt; 2 &gt;$</td>
<td>$&lt; 3, 6 &gt;$</td>
</tr>
</tbody>
</table>

Is $R = S$ ?

Is $R = \pi_A(S)$ ?

Is $R = \pi_{A\leftarrow(A-1)}(S)$ ?

Is $\pi_{A\leftarrow(A+1)}(R) = \pi_A(S)$ ?
Equivalent Expressions

Two expressions are equivalent if they produce the same output.
Equivalent Expressions

Two expressions are equivalent if they produce the same output

but...
Equivalent Expressions

\[
\begin{align*}
&\langle A \rangle & \langle A \rangle & \langle A \rangle \\
&\langle 1 \rangle & \langle 2 \rangle & \langle 1 \rangle \\
&\langle 2 \rangle & \langle 1 \rangle & \langle 2 \rangle \\
&\langle 2 \rangle & \langle 2 \rangle &
\end{align*}
\]

Equivalence under…
- **Bag Semantics**: The same tuples (order-independent)
- **Set Semantics**: The same set of tuples (count-independent)
- **List Semantics**: The same tuples (order matters)
Equivalent Expressions

\[ \langle A \rangle \quad \langle A \rangle \quad \langle A \rangle \]
\[ \langle 1 \rangle \quad \langle 2 \rangle \quad \langle 1 \rangle \]
\[ \langle 2 \rangle \quad \langle 1 \rangle \quad \langle 2 \rangle \]
\[ \langle 2 \rangle \quad \langle 2 \rangle \quad \]

Equivalence under...

- **Bag Semantics**: The same tuples (order-independent)
- **Set Semantics**: The same set of tuples (count-independent)
- **List Semantics**: The same tuples (order matters)
RA Equivalencies

**Selection**

\[
\sigma_{c_1 \land c_2}(R) \equiv \sigma_{c_1}(\sigma_{c_2}(R)) \quad \text{(Decomposable)}
\]

\[
\sigma_{c_1 \lor c_2}(R) \equiv \delta(\sigma_{c_1}(R) \cup \sigma_{c_2}(R)) \quad \text{(Decomposable)}
\]

\[
\sigma_{c_1}(\sigma_{c_2}(R)) \equiv \sigma_{c_2}(\sigma_{c_1}(R)) \quad \text{(Commutative)}
\]

**Projection**

\[
\pi_a(R) \equiv \pi_a(\pi_{a \cup b}(R)) \quad \text{(Idempotent)}
\]

**Cross Product (and Join)**

\[
R \times (S \times T) \equiv (R \times S) \times T \quad \text{(Associative)}
\]

\[
(R \times S) \equiv (S \times R) \quad \text{(Commutative)}
\]

**Try It:** Show that \( R \times (S \times T) \equiv T \times (R \times S) \)
Selection and Projection

\[ \pi_a(\sigma_c(R)) \equiv \sigma_c(\pi_a(R)) \]

Selection commutes with Projection (but only if attribute set \( a \) and condition \( c \) are compatible)

\( a \) must include all columns referenced by \( c \)

Show that

\[ \pi_a(\sigma_c(R)) \equiv \pi_a(\sigma_c(\pi_{a \cup \text{cols}(c)}(R))) \]
Selection and Projection

\[ \pi_a(\sigma_c(R)) \equiv \sigma_c(\pi_a(R)) \]

Selection commutes with Projection (but only if attribute set \( \mathbf{a} \) and condition \( \mathbf{c} \) are compatible)

\( \mathbf{a} \) must include all columns referenced by \( \mathbf{c} \)

Show that

\[ \pi_a(\sigma_c(R)) \equiv \pi_a(\sigma_c(\pi_{a \cup \text{cols}(c)}(R))) \]

When is this rewrite a good idea?
Join

\[ \sigma_c(R \times S) \equiv R \bowtie_c S \]

Selection combines with Cross Product to form a Join as per the definition of Join (Note: This only helps if we have a join algorithm for conditions like \( c \))

Show that

\[ \sigma_{(R.B=S.B) \land (R.A>3)}(R \times S) \equiv \sigma_{(R.A>3)}(R \bowtie_{(R.B=S.B)} S) \]
Join

\[ \sigma_c(R \times S) \equiv R \bowtie_c S \]

Selection combines with Cross Product to form a Join as per the definition of Join.
(Note: This only helps if we have a join algorithm for conditions like \( c \).)

Show that

\[ \sigma_{(R.B=S.B) \land (R.A>3)}(R \times S) \equiv \sigma_{(R.A>3)}(R \bowtie_{(R.B=S.B)} S) \]

When is this rewrite a good idea?
Selection and Cross Product

\[ \sigma_c(R \times S) \equiv (\sigma_c(R) \times S) \]

Selection commutes with Cross Product
(but only if condition \( c \) references attributes of \( R \) exclusively)

Show that

\[ \sigma_{(R.B=S.B) \land (R.A>3)}(R \times S) \equiv \sigma_{(R.A>3)}(R) \Join_{(R.B=S.B)} S \]
Selection and Cross Product

\[ \sigma_c(R \times S) \equiv (\sigma_c(R) \times S) \]

Selection **commutes** with Cross Product (but only if condition \( c \) references attributes of \( R \) exclusively)

**Show that**

\[ \sigma_{(R.B=S.B) \land (R.A>3)}(R \times S) \equiv \sigma_{(R.A>3)}(R) \bowtie_{(R.B=S.B)} S \]

When is this rewrite a good idea?
Projection and Cross Product

\[ \pi_a(R \times S) \equiv (\pi_{a_1}(R)) \times (\pi_{a_2}(S)) \]

Projection **commutes** (distributes) over Cross Product (where \(a_1\) and \(a_2\) are the attributes in \(a\) from \(R\) and \(S\) respectively)

**Show that**

\[ \pi_a(R \bowtie_c S) \equiv (\pi_{a_1}(R)) \bowtie_c (\pi_{a_2}(S)) \]

(under what condition)

How can we work around this limitation?

\[ \pi_a((\pi_{a_1} \cup (\text{cols}(c) \cap \text{cols}(R))(R)) \bowtie_c (\pi_{a_2} \cup (\text{cols}(c) \cap \text{cols}(S))(S))) \]
Projection and Cross Product

\[ \pi_a(R \times S) \equiv (\pi_{a_1}(R)) \times (\pi_{a_2}(S)) \]

Projection **commutes** (distributes) over Cross Product (where \(a_1\) and \(a_2\) are the attributes in \(a\) from \(R\) and \(S\) respectively)

**Show that**

\[ \pi_a(R \bowtie_c S) \equiv (\pi_{a_1}(R)) \bowtie_c (\pi_{a_2}(S)) \]

(under what condition)

**How can we work around this limitation?**

\[ \pi_a((\pi_{a_1 \cup (\text{cols}(c) \cap \text{cols}(R))}(R)) \bowtie_c (\pi_{a_2 \cup (\text{cols}(c) \cap \text{cols}(S))}(S))) \]

**When is this rewrite a good idea?**
RA Equivalencies

Union and Intersections are **Commutative** and **Associative**

Selection and Projection both commute with both Union and Intersection
RA Equivalencies

Union and Intersections are **Commutative** and **Associative**

Selection and Projection both commute with both Union and Intersection

When is this rewrite a good idea?
Example

```
SELECT R.A, T.E
FROM R, S, T
WHERE R.B = S.B
  AND S.C < 5
  AND S.D = T.D
```
Example

\[ \pi_{R.A,T.E} \]

\[ \sigma(R.B=S.B) \land (S.C<5) \land (S.D=T.D) \]

\[ X \]

\[ \pi_{R.A,T.E} \]

\[ \sigma(S.D=T.D) \]

\[ \sigma(R.B=S.B) \land (S.C<5) \]

\[ R \]

\[ S \]

\[ T \]
Example

\[ \pi_{R,A,T,E} \]

\[ \sigma(R.B=S.B) \land (S.C < 5) \land (S.D = T.D) \]

\[ \pi_{R,A,T,E} \]

\[ \sigma(S.D = T.D) \]

\[ \sigma(R.B = S.B) \land (S.C < 5) \]
Example

\[ \pi_{R.A,T.E} \]

\[ \sigma(R.B=S.B) \land (S.C < 5) \land (S.D = T.D) \]

\[ \pi_{R.A,T.E} \]

\[ \sigma(S.D = T.D) \]

\[ \sigma(R.B = S.B) \land (S.C < 5) \]
Example

\[ \pi_{R.A,T.E} \]
\[ \sigma(S.D=T.D) \]
\[ \sigma(R.B=S.B) \land (S.C<5) \]

\[ \pi_{R.A,T.E} \]
\[ \sigma(S.D=T.D) \]
\[ \sigma(R.B=S.B) \land (S.C<5) \]

T
Example

\[ \pi_{R,A,T,E} \]
\[ \sigma(S.D=T.D) \]
\[ \sigma(R.B=S.B) \land (S.C<5) \]

\[ \pi_{R,A,T,E} \]
\[ \sigma(S.D=T.D) \]
\[ \sigma(R.B=S.B) \land (S.C<5) \]
Example

$\pi_{R.A,T.E}$

$\sigma(S.D=T.D)$

$\sigma(R.B=S.B) \land (S.C<5)$

$\pi_{R.A,T.E}$

$\sigma(S.D=T.D)$

$\sigma(R.B=S.B) \land (S.C<5)$

T

R

S
Example

\[ \pi_{R.A,T.E} \]
\[ \sigma(S.D=T.D) \]
\[ X \]
\[ \sigma(R.B=S.B) \land (S.C<5) \] T
\[ X \]
\[ R \quad S \]

\[ \pi_{R.A,T.E} \]
\[ \sigma(R.B=S.B) \land (S.C<5) \] T
\[ X \]
\[ R \quad S \]
Example

\[ \sigma(S.D = T.D) \]

\[ \pi_{R.A,T.E} \]

\[ \sigma(R.B = S.B) \land (S.C < 5) \]

\[ \pi_{R.A,T.E} \]

\[ \nabla(S.D = T.D) \]

\[ \sigma(R.B = S.B) \land (S.C < 5) \]

\[ T \]

\[ X \]

\[ R \]

\[ S \]
Example

\[ \pi_{R.A,T.E} \]
\[ \sigma(S.D=T.D) \]
\[ X \]
\[ \sigma(R.B=S.B) \land (S.C<5) \]
\[ T \]
\[ X \]
\[ R \]
\[ S \]
Example

\[
\begin{align*}
\pi_{R.A,T.E} & \quad \bigtriangleup(S.D=T.D) \\
\sigma(R.B=S.B) \land (S.C<5) & \quad T \\
X & \\
R & \quad S
\end{align*}
\]

\[
\begin{align*}
\pi_{R.A,T.E} & \quad \bigtriangleup(S.D=T.D) \\
\sigma(R.B=S.B) & \quad T \\
X & \\
R & \quad S
\end{align*}
\]
Example

\[\pi_{R.A,T.E} \bowtie (S.D=T.D) \sigma(R.B=S.B) \land (S.C<5) \]

\[\pi_{R.A,T.E} \bowtie (S.D=T.D) \sigma(R.B=S.B) \sigma(S.C<5)\]
Example
Example
Example
Example

\[ \pi_{R.A,T.E} \]
\[ \bowtie (S.D=T.D) \]
\[ \sigma(R.B=S.B) \]
\[ \sigma(S.C<5) \]
\[ X \]
\[ R \quad S \]

\[ \pi_{R.A,T.E} \]
\[ \bowtie (S.D=T.D) \]
\[ \sigma(R.B=S.B) \]
\[ X \]
\[ R \quad \sigma(S.C<5) \]
\[ S \]
Example
Example

\[ \pi_{R.A,T.E} \]

\[ \bowtie(S.D=T.D) \]

\[ \sigma_{(R.B=S.B)} \]

\[
\begin{array}{c}
R \\
\sigma_{(S.C<5)} \\
S
\end{array}
\]

\[ T \]

\[ \pi_{R.A,T.E} \]

\[ \bowtie(S.D=T.D) \]

\[ \sigma_{(R.B=S.B)} \]

\[
\begin{array}{c}
R \\
\sigma_{(S.C<5)} \\
S
\end{array}
\]

\[ T \]
Example

\[ \pi_{R.A,T.E} \]
\[ \bowtie(S.D=T.D) \]
\[ \sigma(R.B=S.B) \]
\[ \times \]
\[ R \]
\[ \sigma(S.C<5) \]
\[ S \]

\[ \pi_{R.A,T.E} \]
\[ \bowtie(S.D=T.D) \]
\[ \times \]
\[ R \]
\[ \sigma(S.C<5) \]
\[ S \]
Final Plan

\[ \pi_{R.A, T.E} \]
\[ \bowtie(S.D = T.D) \]
\[ \bowtie(R.B = S.B) \]
\[ R \bowtriangledown \sigma(S.C < 5) \]
\[ S \]

```
SELECT R.A, T.E
FROM R, S, T
WHERE R.B = S.B
AND S.C < 5
AND S.D = T.D
```
Translate Dumb, Optimize Later

Find Patterns \( \text{Select} (\text{Cross}(R,S)) \) ...
... and Replace \( \text{Join}(R,S) \)
RA Equivalencies

$(R \otimes S) \otimes T \quad vs \quad R \otimes (S \otimes T)$
RA Equivalencies

$$(R \bowtie S) \bowtie T \quad \text{vs} \quad R \bowtie (S \bowtie T)$$

Which form is better?