Early Materialization

\[ \Pi_{B,C} \]
\[ \sigma_{B=5} \]
\[ \sigma_{A=8} \]

T

\[ \Pi_{B,C} \]
\[ \sigma_{B=5} \]
\[ \sigma_{A=8} \]

\[ \sigma_{B=5} \]
\[ \sigma_{A=8} \]

\[ \times_{R_{\text{ID}}} \]

A \mid B \mid C

T.A \rightarrow T.B \rightarrow T.C

SELECT B,C
FROM T
WHERE A=8 AND B=5
(Partially) Late Materialization

\[ \Pi_{B,C} \]
\[ \sigma_{B=5} \]
\[ \sigma_{A=8} \]
\[ T \]

\[ \Pi_{B,C} \frac{\text{potentially all results}}{\sigma_{A=8}} \frac{\sigma_{B=5}}{\Pi_{RID}} \frac{\Pi_{B,RID}}{\sigma_{A=5}} \frac{T.A}{\sigma_{B=5}} \frac{T.B}{\Pi_{RID}} \frac{\Pi_{C}}{\sigma_{C}} \frac{\sigma_{C}}{\Pi_{RID}} \]

\[
\text{SELECT } B, C \\
\text{FROM } T \\
\text{WHERE } A=8 \text{ AND } B=5
\]
(Really) Late Materialization

\[ \prod_{B,C} \]
\[ \sigma_{B=5} \]
\[ \sigma_{A=8} \]
\[ T \]

\[ \prod_{B,C} \]
\[ \cap_{RID} \]
\[ A=8 \land B=5 \]
\[ \prod_{RID} \]
\[ \cap_{RID} \]
\[ \sigma_{A=8} \]
\[ \sigma_{B=5} \]
\[ T.A \]
\[ T.B \]
\[ T.C \]

`SELECT B,C
FROM T
WHERE A=8 AND B=5`
Projection vs “Anti-Projection”

When to narrow tuples?

\[ \Pi_{A,B} \]

\[ \bigcup_c \]

When to widen tuples?
Example: Tuple Reconstruction in Column Stores

```
SELECT * FROM T
```

`query Column Store` -> Column Store

```
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

? -> Row-wise output

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
Tuple 0
Tuple 1
Tuple 2
```
Implementing Early Materialization

```
SELECT B, C
FROM T
WHERE A = 8 AND B = 5
```

![Diagram showing the query request, intermediate representation, and output](image)

- **Query Request**: `SELECT B, C FROM T WHERE A = 8 AND B = 5`
- **Intermediate Representation**:
  - Table:
    | A | B | C |
    |---|---|---|
    | 3 | 2 | 7 |
    | 8 | 5 | 3 |
    | 8 | 2 | 9 |

- **(Early) Materialization**:
- **Output**:
  - Table:
    | B | C |
    |---|---|
    | 5 | 3 |
Implementing Late Materialization (1)

SELECT B, C
FROM T
WHERE A = 8 AND B = 5

query request

selection on A and B

column store

input columns

bitvector marking qualifying entries

conjunction bitvector

\[ \sigma_{A=8} \]
\[ \sigma_{B=5} \]

\[ \sigma_{A=8 \text{ AND } B=5} \]
SELECT B, C
FROM T
WHERE A=8 AND B=5
Joins using Early Materialization

Input to join has been materialized already!

```
SELECT T.B, S.C
FROM T, S
WHERE T.A = S.A
```
Joins using Late Materialization

\[ \text{SELECT T.B, S.C} \]
\[ \text{FROM T, S} \]
\[ \text{WHERE T.A = S.A} \]
Early Materialization

Advantages:

no re-access of columns necessary
Early Materialization

Advantages:

no re-access of columns necessary

easier planning

Disadvantages:

possible generation of wide intermediate results
Late Materialization

Advantages:

- constructing tuples only when necessary
- slightly more complex planning (actually a disadvantage)

Diagram:

- Table R: \( A, B, C \)
- Table S: \( D, E, F \)
- Table R: \( A, B, C \)
- Table S: \( D, E, F \)

Tuple join: \( \rightarrow \)
Late Materialization

Advantages:

- constructing tuples only when necessary
- slightly more complex planning

Disadvantages:

- re-access of columns possible
<table>
<thead>
<tr>
<th>Feature</th>
<th>Early Materialization</th>
<th>Late Materialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selectivity</td>
<td>Low (many entries selected)</td>
<td>High (few entries selected)</td>
</tr>
<tr>
<td>Compression</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Aggregation</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>