joins
prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
**Milestone1**: quite involved but easy on the algorithmic side
**Milestone2**: easy once you understand we just batching data
**Milestone3**: not easy, not easy (all concepts/tools needed)
**Milestone4**: better than M3 but still heavy on concepts
**Milestone5**: should be quick

**Testing server**: will run twice a day as of this week

**Remember**: limited chances of success if you try to do this alone

**Lab marathon**: once more in a couple of weeks
FINAL REPORT CONTAINS EXPERIMENTAL ANALYSIS

HOW TO DO EXPERIMENTS?

find out what matters,

say we want to test the select operator,

say we want to test the select operator,
to compare scan vs secondary index
ISOLATE PERFORMANCE AS BEST AS POSSIBLE

do not listen to youtube while you run experiments!
  close all apps, recreate the same environment every time
  create scripts for everything
examples for final evaluation

e.g., to test the select operator
examples for final evaluation

e.g., to test shared scans

throughput (q/s)

# of queries

OK

throughput (q/s)

# of queries

not OK
**Midterm1**: overall great performance!
If you did not score 90+
please consider joining more for OH!
If you did score 90+
please consider joining more for OH!

**Midterm2**: Nov 15 -> Nov 20?

all quizzes, all discussions, all “Read” readings
extra weekend OH will be announced
so far
so far
so far

database kernel

scan, binary search, tuple reconstruction, min, max, search b-tree, etc.

columns, rows, hybrids, trees

cpu
memory
disk

algorithms/operators

data
data
data
so far

- early/late tuple reconstruction, tuple-at-a-time, vectorized or bulk processing, intermediates format, pushing selects down, etc

- scan, binary search, tuple reconstruction, min, max, search b-tree, etc.

- columns, rows, hybrids, trees

- algorithms/operators

- cpu

- memory

- disk

- data

- database kernel
joins

(project=m4)
star schema

fact table
(id1, id2,...)

dimension table 1
(id1,...)

dimension table 2
(id2,...)

...
avoid duplicates - minimize update cost - but we have to do joins
snowflake schema
give me all students enrolled in cs165

```
select student.name from student, enrolled, course where
course.name="cs165" and enrolled.courseld=course.id and
student.id=enrolled.studentId
```
find all tuples where FK = PK

join: glue the data back together
find all tuples where FK=PK

join: glue the data back together

equi join
key, payload
1, d1, e1, f1
1, d2, e2, f2
2, d3, e3, f3
2, d4, e4, f4
2, d5, e5, f5
3, d6, e6, f6

join

key, payload
1, a1, b1, c1
2, a2, b2, c2
3, a3, b3, c3
4, a4, b4, c4
1, a5, b5, c5

join result
1, d1, e1, f1, a1, b1, c1
1, d1, e1, f1, a5, b5, c5
1, d2, e2, f2, a1, b1, c1
1, d2, e2, f2, a5, b5, c5
2, d3, e3, f3, a2, b2, c2
2, d4, e4, f4, a2, b2, c2
2, d5, e5, f5, a2, b2, c2
3, d6, e6, f6, a3, b3, c3
key, payload
1, d1, e1, f1
1, d2, e2, f2
2, d3, e3, f3
2, d4, e4, f4
2, d5, e5, f5
3, d6, e6, f6

join

key, payload
1, a1, b1, c1
2, a2, b2, c2
3, a3, b3, c3
4, a4, b4, c4
1, a5, b5, c5

join result
1, d1, e1, f1, a1, b1, c1
1, d1, e1, f1, a5, b5, c5
1, d2, e2, f2, a1, b1, c1
1, d2, e2, f2, a5, b5, c5
2, d3, e3, f3, a2, b2, c2
2, d4, e4, f4, a2, b2, c2
2, d5, e5, f5, a2, b2, c2
3, d6, e6, f6, a3, b3, c3
The figure illustrates a join operation in database management. The left and middle panels show the original datasets with keys and payloads. The right panel shows the join result.

### Left Panel
- Key: 1, 2, 3
- Payload: d1,e1,f1, d2,e2,f2, d3,e3,f3, d4,e4,f4, d5,e5,f5, d6,e6,f6

### Middle Panel
- Key: 1, 2, 3
- Payload: a1,b1,c1, a2,b2,c2, a3,b3,c3, a4,b4,c4, a5,b5,c5

### Right Panel
- Join Result:
  1. d1,e1,f1, a1,b1,c1
  1. d1,e1,f1, a5,b5,c5
  1. d2,e2,f2, a1,b1,c1
  1. d2,e2,f2, a5,b5,c5
  2. d3,e3,f3, a2,b2,c2
  2. d4,e4,f4, a2,b2,c2
  2. d5,e5,f5, a2,b2,c2
  3. d6,e6,f6, a3,b3,c3
key, payload

1, d1, e1, f1
1, d2, e2, f2
2, d3, e3, f3
2, d4, e4, f4
2, d5, e5, f5
3, d6, e6, f6

join

key, payload

1, a1, b1, c1
2, a2, b2, c2
3, a3, b3, c3
4, a4, b4, c4
1, a5, b5, c5

join result

1, d1, e1, f1, a1, b1, c1
1, d1, e1, f1, a5, b5, c5
1, d2, e2, f2, a1, b1, c1
1, d2, e2, f2, a5, b5, c5
2, d3, e3, f3, a2, b2, c2
2, d4, e4, f4, a2, b2, c2
2, d5, e5, f5, a2, b2, c2
3, d6, e6, f6, a3, b3, c3
key,payload
1,d1,e1,f1
1,d2,e2,f2
2,d3,e3,f3
2,d4,e4,f4
2,d5,e5,f5
3,d6,e6,f6

join

key,payload
1,a1,b1,c1
2,a2,b2,c2
3,a3,b3,c3
4,a4,b4,c4
1,a5,b5,c5

join result

1,d1,e1,f1,a1,b1,c1
1,d1,e1,f1,a5,b5,c5
1,d2,e2,f2,a1,b1,c1
1,d2,e2,f2,a5,b5,c5
2,d3,e3,f3,a2,b2,c2
2,d4,e4,f4,a2,b2,c2
2,d5,e5,f5,a2,b2,c2
3,d6,e6,f6,a3,b3,c3
join result

1,d1,e1,f1,a1,b1,c1
1,d1,e1,f1,a5,b5,c5
1,d2,e2,f2,a1,b1,c1
1,d2,e2,f2,a5,b5,c5
2,d3,e3,f3,a2,b2,c2
2,d4,e4,f4,a2,b2,c2
2,d5,e5,f5,a2,b2,c2
3,d6,e6,f6,a3,b3,c3
key, payload

1, d1, e1, f1
1, d2, e2, f2
2, d3, e3, f3
2, d4, e4, f4
2, d5, e5, f5
3, d6, e6, f6

join

key, payload

1, a1, b1, c1
2, a2, b2, c2
3, a3, b3, c3
4, a4, b4, c4
1, a5, b5, c5

join result

1, d1, e1, f1, a1, b1, c1
1, d1, e1, f1, a5, b5, c5
1, d2, e2, f2, a1, b1, c1
1, d2, e2, f2, a5, b5, c5
2, d3, e3, f3, a2, b2, c2
2, d4, e4, f4, a2, b2, c2
2, d5, e5, f5, a2, b2, c2
3, d6, e6, f6, a3, b3, c3
key, payload

1, d1, e1, f1
1, d2, e2, f2
2, d3, e3, f3
2, d4, e4, f4
2, d5, e5, f5
3, d6, e6, f6

join

key, payload

1, a1, b1, c1
2, a2, b2, c2
3, a3, b3, c3
4, a4, b4, c4
1, a5, b5, c5

join result

1, d1, e1, f1, a1, b1, c1
1, d1, e1, f1, a5, b5, c5
1, d2, e2, f2, a1, b1, c1
1, d2, e2, f2, a5, b5, c5
2, d3, e3, f3, a2, b2, c2
2, d4, e4, f4, a2, b2, c2
2, d5, e5, f5, a2, b2, c2
3, d6, e6, f6, a3, b3, c3
key, payload
1, d1, e1, f1
1, d2, e2, f2
2, d3, e3, f3
2, d4, e4, f4
2, d5, e5, f5
3, d6, e6, f6

join result
1, d1, e1, f1, a1, b1, c1
1, d1, e1, f1, a5, b5, c5
1, d2, e2, f2, a1, b1, c1
1, d2, e2, f2, a5, b5, c5
2, d3, e3, f3, a2, b2, c2
2, d4, e4, f4, a2, b2, c2
2, d5, e5, f5, a2, b2, c2
3, d6, e6, f6, a3, b3, c3

key, payload
1, a1, b1, c1
2, a2, b2, c2
3, a3, b3, c3
4, a4, b4, c4
1, a5, b5, c5

join
1, d1, e1, f1
1, d1, e1, f1
1, d2, e2, f2
2, d3, e3, f3
2, d4, e4, f4
2, d5, e5, f5
3, d6, e6, f6
inner join

left outer

right outer

\[ \text{v1, a1, b2} \]

\[ \text{v1, a1, b2} \]

\[ \text{v1, a1, b2} \]

\[ \text{v2, a2, null} \]

\[ \text{v3, null, b1} \]

\[ \text{v1, a1, b2} \]

\[ \text{v1, a1, b2} \]

\[ \text{v3, b1} \]

\[ \text{v1, b2} \]
```
SELECT student.name
FROM students, enrolled, courses
WHERE courses.name = "cs165"
AND enrolled.courseId = course.id
AND student.id = enrolled.studentId
```

```sql
SELECT courses.name = "cs165"
```

```
SELECT student.name
FROM students, enrolled, courses
WHERE courses.name = "cs165"
AND enrolled.courseId = course.id
AND student.id = enrolled.studentId
```

```
project student.name
```

join

student.id = enrolled.studentId
select student.name
from students, enrolled, courses
where courses.name="cs165"
and enrolled.courseId=course.id
and student.id=enrolled.studentId

pushing selects down
R(A,B,C,D) - S(A,E,F,G)

```sql
select max(R.D),min(S.G)
from R,S
where R.A=S.A and R.C<10 and S.F>30
```
### Initial Status

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th></th>
<th></th>
<th></th>
<th>S</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>3</td>
<td>12</td>
<td>12</td>
<td>17</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>34</td>
<td>34</td>
<td>49</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>56</td>
<td>75</td>
<td>53</td>
<td>58</td>
<td>62</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>45</td>
<td>23</td>
<td>99</td>
<td>44</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>49</td>
<td>78</td>
<td>64</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>58</td>
<td>65</td>
<td>37</td>
<td>78</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>97</td>
<td>33</td>
<td>53</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>41</td>
<td>75</td>
<td>21</td>
<td>61</td>
<td>81</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>42</td>
<td>29</td>
<td>32</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>55</td>
<td>0</td>
<td>50</td>
<td>23</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and 5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
Initial Status

<table>
<thead>
<tr>
<th>Relation R</th>
<th>Relation S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Rb</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
</tbody>
</table>

Query and Query Plan (MAL Algebra)

\[
\text{select sum}(R.a) \text{ from } R, S \text{ where } R.c = S.b \text{ and } 5 < R.a < 20 \text{ and } 40 < R.b < 50 \text{ and } 30 < S.a < 40
\]

1. \( \text{inter1} = \text{select}(Ra, 5, 20) \)
2. \( \text{inter2} = \text{reconstruct}(Rb, \text{inter1}) \)
3. \( \text{inter3} = \text{select}(\text{inter2}, 40, 50) \)
4. \( \text{join_input_R} = \text{reconstruct}(Rc, \text{inter3}) \)
5. \( \text{inter4} = \text{select}(Sa, 50, 65) \)
6. \( \text{inter5} = \text{reconstruct}(Sb, \text{inter4}) \)
7. \( \text{join_input_S} = \text{reverse}(\text{inter5}) \)
8. \( \text{join_res_R_S} = \text{join}(\text{join_input_R}, \text{join_input_S}) \)
9. \( \text{inter6} = \text{voidTail}(\text{join_res_R_S}) \)
10. \( \text{inter7} = \text{reconstruct}(Ra, \text{inter6}) \)
11. \( \text{result} = \text{sum}(\text{inter7}) \)
Initial Status

<table>
<thead>
<tr>
<th>Relation R</th>
<th>Relation S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Rb</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
</tbody>
</table>

Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and 5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
**Initial Status**

<table>
<thead>
<tr>
<th>Relation R</th>
<th>Relation S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Rb</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
</tbody>
</table>

**Query and Query Plan (MAL Algebra)**

\[
\text{select sum}(R.a) \text{ from } R, S \text{ where } R.c = S.b \text{ and } 5 < R.a < 20 \text{ and } 40 < R.b < 50 \text{ and } 30 < S.a < 40
\]

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)

**Select (Sa,50,65)**

<table>
<thead>
<tr>
<th>Sa</th>
<th>inter4</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>49</td>
<td>5</td>
</tr>
<tr>
<td>58</td>
<td>7</td>
</tr>
<tr>
<td>99</td>
<td>8</td>
</tr>
<tr>
<td>64</td>
<td>10</td>
</tr>
</tbody>
</table>

**Reverse (inter5)**

<table>
<thead>
<tr>
<th>inter5</th>
<th>join_input_S</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>29</td>
</tr>
<tr>
<td>7</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>81</td>
</tr>
<tr>
<td>10</td>
<td>23</td>
</tr>
</tbody>
</table>

**Reconstruct (Sb,inter4)**

<table>
<thead>
<tr>
<th>Sb</th>
<th>inter5</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>35</td>
<td>5</td>
</tr>
<tr>
<td>62</td>
<td>7</td>
</tr>
<tr>
<td>29</td>
<td>8</td>
</tr>
<tr>
<td>19</td>
<td>10</td>
</tr>
</tbody>
</table>

**Select (Sa,50,65)**

<table>
<thead>
<tr>
<th>Sa</th>
<th>inter4</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>49</td>
<td>5</td>
</tr>
<tr>
<td>58</td>
<td>7</td>
</tr>
<tr>
<td>99</td>
<td>8</td>
</tr>
<tr>
<td>64</td>
<td>10</td>
</tr>
</tbody>
</table>

**Reverse (inter5)**

<table>
<thead>
<tr>
<th>inter5</th>
<th>join_input_S</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>29</td>
</tr>
<tr>
<td>7</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>81</td>
</tr>
<tr>
<td>10</td>
<td>23</td>
</tr>
</tbody>
</table>

**Reconstruct (Sb,inter4)**

<table>
<thead>
<tr>
<th>Sb</th>
<th>inter5</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>35</td>
<td>5</td>
</tr>
<tr>
<td>62</td>
<td>7</td>
</tr>
<tr>
<td>29</td>
<td>8</td>
</tr>
<tr>
<td>19</td>
<td>10</td>
</tr>
</tbody>
</table>

**Join (join_input_R,join_input_S)**

<table>
<thead>
<tr>
<th>join_res_R_S</th>
<th>inter5</th>
<th>join_input_S</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>3</td>
<td>29</td>
</tr>
<tr>
<td>7</td>
<td>19</td>
<td>81</td>
</tr>
<tr>
<td>8</td>
<td>81</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>23</td>
<td>23</td>
</tr>
</tbody>
</table>

**Result**

result = sum(inter7)
<table>
<thead>
<tr>
<th>Ra</th>
<th>Rb</th>
<th>Rc</th>
<th>Sa</th>
<th>Sb</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12</td>
<td>12</td>
<td>17</td>
<td>11</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
<td>34</td>
<td>49</td>
<td>35</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
<td>53</td>
<td>58</td>
<td>62</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
<td>23</td>
<td>99</td>
<td>44</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
<td>78</td>
<td>64</td>
<td>29</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
<td>65</td>
<td>37</td>
<td>78</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
<td>33</td>
<td>53</td>
<td>19</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
<td>21</td>
<td>61</td>
<td>81</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
<td>29</td>
<td>32</td>
<td>26</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
<td>0</td>
<td>50</td>
<td>23</td>
</tr>
</tbody>
</table>

**Query and Query Plan (MAL Algebra)**

select sum(R.a) from R, S where R.c = S.b and 5 < R.a < 20 and 40 < R.b < 50 and 30 < S.a < 40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S
where R.c = S.b and
5 < R.a < 20 and 40 < R.b < 50 and 30 < S.a < 40

1. inter1 = select(Ra, 5, 20)
2. inter2 = reconstruct(Rb, inter1)
3. inter3 = select(inter2, 40, 50)
4. join_input_R = reconstruct(Rc, inter3)
5. inter4 = select(Sa, 50, 65)
6. inter5 = reconstruct(Sb, inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R, join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra, inter6)
11. result = sum(inter7)
Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and 5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
Initial Status

<table>
<thead>
<tr>
<th>Relation R</th>
<th>Relation S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Rb</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
</tbody>
</table>

Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and 5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
nested loops

for all tuples of one side check all tuples of the other side

new resL[]; new resR[]; k=0
for (i=0;i<L;i=i++)
  for (j=0;j<R;j++)
    if L[i]==R[j]
      resL[k]=i
      resR[k++]=j
outer(L) join inner(R)

say red fits in Level1
probe the red side for better data locality

Total footprint = L + R + res (bytes),
R.pages <= Level1.pages - 2

stream outer pages hold inner pages

Level1
Level2
res
what if not all data fits in main-memory?
what if not all data fits in L3 cache?
what if not all data fits in L2 cache?
what if not all data fits in L1 cache?
can we utilize >1 cores?
for every L.key = R.key pair
return [L.pos,R.pos]
data/results stored
one column-at-a-time

1) design a nested loops join algorithm and give its I/O cost
2) which column should be the inner and why?
3) describe optimizations to minimize Level1 misses

Quickly if there is time think of the following:
4) can we use sorting?
5) how would you use a b-tree to do a join?

R.size > Level1.size, L.size > Level1.size,
R.size + L.size << L2, Level1 block size = Level 2 block size
\( \text{lp} = \text{LeftEntriesThatFitInOnePage} \)
\( \text{rp} = \text{RightEntriesThatFitInOnePage} \)
\( L = \text{number of values in L column} \)
\( R = \text{number of values in R column} \)

```plaintext
new resL[]; new resR[]; k=0
for (i=0; i<L; i=i++)
  for (j=0; j<R; j++)
    if L[i] == R[j]
      resL[k]=i
      resR[k++]=j
```

\textbf{comp} \( O(L \times R) \)

\textbf{I/O} \( O(L/\text{lp} + L \times (R/\text{rp})) \)
A number of pages will still be in LLC!
\[ \text{lp}=\text{LeftEntriesThatFitInOnePage} \]
\[ \text{rp}=\text{RightEntriesThatFitInOnePage} \]
\[ \text{L}= \text{number of values in L column} \]
\[ \text{R}= \text{number of values in R column} \]

new resL[]; new resR[]; k=0

\textbf{for} (i=0;i<L;i=i++)
\hspace{1em} \textbf{for} (j=0;j<R;j++)
\hspace{2em} \textbf{if} L[i]==R[j]
\hspace{3em} \text{resL}[k]=i
\hspace{3em} \text{resR}[k++]=j

\textbf{comp} \ O(L \times R)
\textbf{I/O} \ O(L/lp+L \times (R/rp))
lp = LeftEntriesThatFitInOnePage
rp = RightEntriesThatFitInOnePage
L = number of values in L column
R = number of values in R column

New resL[]; new resR[]; k = 0

for (i = 0; i < L; i++)
  for (j = 0; j < R; j++)
    if L[i] == R[j]
      resL[k] = i
      resR[k++] = j

comp $O(L \times R)$

I/O $O(L/lp + L \times (R/rp))$

I/O with zig zag:
L/lp + R/rp, if $R/rp \leq LLC - 2$

L/lp + L \times (R/rp - (LLC - 2)), if $R/rp \leq 2 \times ( LLC - 2)$
new resL[], new resR[], k=0
for (i=0;i<L;i=i++)
    for (j=0;j<R;j++)
        if L[i]==R[j]
            resL[k]=i
            resR[k++]=j

comp \(O(L \times R)\)
I/O \(O(L/lp+L \times (R/rp))\)

new resL[], new resR[], k=0
for (i=0;i<L;i+=lp)
    for (j=0;j<R;j+=rp)
        for (r=i;r<i+lp;r++)
            for (m=j;m<j+rp;m++)
                if L[r]==R[m]
                    resL[k]=r
                    resR[k++]=m

comp \(O(L \times R)\)
I/O \(O(L/lp+(L/lp) \times (R/rp))\)
new resL[]; new resR[]; k=0
for (i=0; i<L; i=i++)
    for (j=0; j<R; j++)
        if L[i]==R[j]
            resL[k]=i
            resR[k++]=j
sort merge join

while left and right still have tuples
  if left.val < right.val left++
  else if left.val > right.val right++
  else
    add to result, left++, right++

+ handle duplicates on both sides!
new resL[]; new resR[]; k=0
for (i=0;i<L;i=i++)
    jk=R.btree.probe(L[i])
    if (jk!=null)
        resL[k]=i
        resR[k++]=jk.pos
Read: textbook: chapters 4, 14
joins

DATA SYSTEMS

prof. Stratos Idreos