CS186 Discussion 5

(Relational Algebra, Join Algorithms)

Matthew Deng

Relational Algebra

Relational Algebra

- Set of operators that map relations to relations
- Set semantics
  - Does not include duplicates
  - SQL has multiset semantics

Relational Operators

- Selection (σ) Selects a subset of rows (horizontal)
  - $\sigma_{age>20}(R)$
- Projection (π) Selects a subset of columns (vertical)
  - $\pi_{name,\ age}(R)$
- Cross-product ($\times$) Allows us to combine two relations.
  - $R \times S$
- Set-difference (—) Tuples in r1, but not in r2.
  - $R - S$
- Union (U) Tuples in r1 or in r2.
  - $R \cup S$

Compound Operators

- Intersection ($\cap$) Tuples in r1 and r2
  - $R \cap S = R - (R - S)$
- Join ($\bowtie$) joins r1 and r2 on common attributes
  - Compute $R \times S$
  - Select rows where attributes appearing in both relations have equal values
  - Project onto all unique attributes and one copy of each of the common ones

Relational Algebra Exercises

Relational Algebra Exercises

1. Why do we care about relational algebra? Why do you think it might be useful?

Relational Algebra Exercises

1. Why do we care about relational algebra? Why do you think it might be useful?

   Relational algebra are plans to execute queries.

   The many ways of writing the plans give the system room to design for optimizations.

   We will learn more about how to estimate the cost of the plan in the future.

Relational Algebra Exercises

2. How would you compare what you've seen in PySpark (transformations like map/reduce) to the SQL model (select project join)? Which one do you like using better? (open ended)

Relational Algebra Exercises

2. How would you compare what you've seen in PySpark (transformations like map/reduce) to the SQL model (select project join)? Which one do you like using better? (open ended)

In general, the SQL DBMSs were significantly faster and required less code to implement each task, but took longer to tune and load the data."

Relational Algebra Exercises

Consider the schema:

`Songs (song_id, song_name, album_id, weeks_in_top_40)`

`Artists(artist_id, artist_name, first_year_active)`

`Albums (album_id, album_name, artist_id, year_released, genre)`

Write relational algebra expressions for the following queries:

1. Find the name of the artists who have albums with a genre of either ‘pop’ or ‘rock’.

Relational Algebra Exercises

Consider the schema:

`Songs (song_id, song_name, album_id, weeks_in_top_40)`

`Artists(artist_id, artist_name, first_year_active)`

`Albums (album_id, album_name, artist_id, year_released, genre)`

Write relational algebra expressions for the following queries:

1. Find the name of the artists who have albums with a genre of either ‘pop’ or ‘rock’.

$$\pi_{artists.artist\_name}$$
$$(Artists \bowtie (\sigma_{albums.genre = \text{‘pop’} \vee albums.genre = \text{‘rock’}} Albums))$$

Relational Algebra Exercises

Consider the schema:

`Songs (song_id, song_name, album_id, weeks_in_top_40)`

`Artists(artist_id, artist_name, first_year_active)`

`Albums (album_id, album_name, artist_id, year_released, genre)`

Write relational algebra expressions for the following queries:

2. Find the name of the artists who have albums of genre ‘pop’ and ‘rock’.

Relational Algebra Exercises

Consider the schema:

```
Songs (song_id, song_name, album_id,
weeks_in_top_40)
Artists(artist_id, artist_name, first_year_active)
Albums (album_id, album_name, artist_id, year_released,
genre)
```

Write relational algebra expressions for the following queries:

2. Find the name of the artists who have albums of genre ‘pop’ and ‘rock’.

$$\pi_{artists.artist\_name}((\sigma_{albums.genre\ =\ ‘pop’}Albums) \bowtie Artists)\ \cap$$
$$\pi_{artists.artist\_name}((\sigma_{albums.genre\ =\ ‘rock’}Albums) \bowtie Artists)$$

Relational Algebra Exercises

Consider the schema:

`Songs (song_id, song_name, album_id, weeks_in_top_40)`

`Artists(artist_id, artist_name, first_year_active)`

`Albums (album_id, album_name, artist_id, year_released, genre)`

Write relational algebra expressions for the following queries:

3. Find the id of the artists who have albums of genre ‘pop’ or have spent over 10 weeks in the top 40.

Relational Algebra Exercises

Consider the schema:

`Songs (song_id, song_name, album_id, weeks_in_top_40)`

`Artists(artist_id, artist_name, first_year_active)`

`Albums (album_id, album_name, artist_id, year_released, genre)`

Write relational algebra expressions for the following queries:

3. Find the id of the artists who have albums of genre ‘pop’ or have spent over 10 weeks in the top 40.

$$\pi_{\text{albums.artist\_id}}(\sigma_{\text{albums.genre = 'pop'}}\text{Albums})) \cup$$

$$\pi_{\text{albums.artist\_id}}(\text{Albums} \bowtie (\sigma_{\text{songs.weeks\_in\_top\_40} > 10}\text{Songs}))$$

Relational Algebra Exercises

Consider the schema:

`Songs (song_id, song_name, album_id, weeks_in_top_40)`

`Artists(artist_id, artist_name, first_year_active)`

`Albums (album_id, album_name, artist_id, year_released, genre)`

Write relational algebra expressions for the following queries:

4. Find the names of all artists who do not have any albums.

Relational Algebra Exercises

Consider the schema:

`Songs (song_id, song_name, album_id, weeks_in_top_40)`

`Artists(artist_id, artist_name, first_year_active)`

`Albums (album_id, album_name, artist_id, year_released, genre)`

Write relational algebra expressions for the following queries:

4. Find the names of all artists who do not have any albums.

$$\pi_{artists.artist\_name}$$
$$(Artists \bowtie ((\pi_{artists.artist\_id} Artists) - (\pi_{albums.artist\_id} Albums)))$$

Join Algorithms

Cost Notation

- [R] = number of pages in Table R
- p_R = number of records per page of R
- |R| = number of records in R

$$|R| = p_R * [R]$$

Simple Nested Loop Join

```
for record r in R:
  for record s in S:
    if theta(r, s):
      add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + |R| * [S]$$

Simple Nested Loop Join

```
for record r in R:
  for record s in S:
    if theta(r, s):
      add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + |R| * [S]$$

Simple Nested Loop Join

```
for record r in R:
  for record s in S:
    if theta(r, s):
      add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + |R| * [S]$$

Page Nested Loop Join

```
for page p_r in R:
  for page p_s in S:
    for record r in p_r:
      for record s in p_s:
        if theta(r, s):
          add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + [R] * [S]$$

Page Nested Loop Join

```
for page p_r in R:
  for page p_s in S:
    for record r in p_r:
      for record s in p_s:
        if theta(r, s):
          add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + [R] * [S]$$

Page Nested Loop Join

```
for page p_r in R:
  for page p_s in S:
    for record r in p_r:
      for record s in p_s:
        if theta(r, s):
          add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

[R] + [R] * [S]

Block Nested Loop Join

```
for block b_r in R:
  for page p_s in S:
    for record r in b_r:
      for record s in p_s:
        if theta(r, s):
          add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + \lceil [R] / (B - 2) \rceil * [S]$$

B-2 pages in each block

Block Nested Loop Join

```
for block b_r in R:
  for page p_s in S:
    for record r in b_r:
      for record s in p_s:
        if theta(r, s):
          add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + \lceil [R] / (B - 2) \rceil * [S]$$

B-2 pages in each block

Block Nested Loop Join

```
for block b_r in R:
  for page p_s in S:
    for record r in b_r:
      for record s in p_s:
        if theta(r, s):
          add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + \lceil [R] / (B - 2) \rceil * [S]$$

B-2 pages in each block

Sort Merge Join

1. External Sort R
2. External Sort S
3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$4 * ([R] + [S]) + [R] + [S]$$
$$= 5 * ([R] + [S])$$

Sort Merge Join

1. External Sort R
2. External Sort S
3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$4 * ([R] + [S]) + [R] + [S]$$
$$= 5 * ([R] + [S])$$

Sort Merge Join

1. External Sort R
2. External Sort S
3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$4 * ([R] + [S]) + [R] + [S]$$
$$= 5 * ([R] + [S])$$

Sort Merge Join

1. External Sort R
2. External Sort S
3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$4 * ([R] + [S]) + [R] + [S]$$
$$= 5 * ([R] + [S])$$

Optimized Sort Merge Join

1. Internal Sort R

   (pass 0 only)

2. Internal Sort S

   (pass 0 only)

3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Optimized Sort Merge Join

1. Internal Sort R
   (pass 0 only)
2. Internal Sort S
   (pass 0 only)
3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Optimized Sort Merge Join

1. Internal Sort R
   (pass 0 only)
2. Internal Sort S
   (pass 0 only)
3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Optimized Sort Merge Join

1. Internal Sort R
   (pass 0 only)
2. Internal Sort S
   (pass 0 only)
3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$

$$= 3 * ([R] + [S])$$

Hash Join

1. Partition R
2. Partition S
3. Build Hash Table for R
4. Stream and Probe S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Hash Join

1. Partition R
2. Partition S
3. Build Hash Table for R
4. Stream and Probe S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Hash Join

1. Partition R
2. Partition S
3. Build Hash Table for R
4. Stream and Probe S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Hash Join

1. Partition R
2. Partition S
3. Build Hash Table for R
4. Stream and Probe S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Hash Join

1. Partition R
2. Partition S
3. Build Hash Table for R
4. Stream and Probe S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$

$$= 3 * ([R] + [S])$$

Join Algorithms Worksheet #1

Join Algorithms Exercises

1. Describe when you would want to use a block nested loop join, a sort-merge join and a hash join:

Join Algorithms Exercises

1. Describe when you would want to use a block nested loop join, a sort-merge join and a hash join:

Block Nested Loop Join:
- Cartesian Product
- Non-equality Predicate

Sort Merge Join:
- Sorting
- Skewed Data
- Limited Memory

Hash Join:
- Hashing
- Uneven Relations
- Hybrid Hashing

Join Algorithms Worksheet #2

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables R and S, where R contains [R] = 100 pages and S contains [S] = 50 pages, R holds $p_R = 100$ tuples per page and S holds $p_S = 50$ tuples per page.

How many disk reads are needed to perform a Simple Nested Loops join?

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables R and S, where R contains [R] = 100 pages and S contains [S] = 50 pages, R holds p_R = 100 tuples per page and S holds p_S = 50 tuples per page.

How many disk reads are needed to perform a Simple Nested Loops join?

Using S as the outer relation yields the lowest I/O count.

p_S*[R]*[S] + [S]

= 50*100*50 + 50

= 250050 I/O's

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables R and S, where R contains [R] = 100 pages and S contains [S] = 50 pages, R holds $p_R = 100$ tuples per page and S holds $p_S = 50$ tuples per page.

How about a block nested loops join?

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables R and S, where R contains [R] = 100 pages and S contains [S] = 50 pages, R holds p_R = 100 tuples per page and S holds p_S = 50 tuples per page.

How about a block nested loops join?

Using S as the outer relation yields the lowest I/O count.

$$[S] + \lceil [S] / (B - 2) \rceil * [R]$$

$$= 50 + \lceil 50/13 \rceil * 100$$

$$= 450 \text{ I/Os}$$

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables R and S, where R contains [R] = 100 pages and S contains [S] = 50 pages, R holds p_R = 100 tuples per page and S holds p_S = 50 tuples per page.

How about a Sort Merge Join? (Utilize the optimized version)

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables R and S, where R contains [R] = 100 pages and S contains [S] = 50 pages, R holds p_R = 100 tuples per page and S holds p_S = 50 tuples per page.

How about a Sort Merge Join? (Utilize the optimized version)

$$3 * ([R] + [S])$$
$$= 3 * 150$$
$$= 450 \text{ I/O's}$$

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables R and S, where R contains [R] = 100 pages and S contains [S] = 50 pages, R holds $p_R = 100$ tuples per page and S holds $p_S = 50$ tuples per page.

How about a Hash Join? (Assume no recursive partitioning and ignore output costs)

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables R and S, where R contains [R] = 100 pages and S contains [S] = 50 pages, R holds p_R = 100 tuples per page and S holds p_S = 50 tuples per page.

How about a Hash Join? (Assume no recursive partitioning and ignore output costs)

Partitioning Phase: 2([R]+[S])

Matching Phase: [R]+[S]

Total = 3([R] + [S]) = 3* 150 = 450 I/O's