Introduction to Data Management

Relational Algebra

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

Announcements

- Midterm exam October 30th
- coming up in a few weeks
- one sheet of notes, start writing it now!
- In general – feedback appreciated!
- Lecture timing
- Content pace
- Feedback link: https://feedback.cs.washington.edu/

Outline

- Introduce relational algebra
- Look at some example RA from previous lectures
- Translating SQL ⇢ RA

What’s the Point of RA?

- SQL is a Declarative Language
  - “What to get” rather than “how to get it”
  - Easier to write a SQL query than write a whole Java program that will probably perform worse
- But computers are imperative/procedural
  - Computers only understand the “how”
- RDBMS

Starting point

FWGHOS™

SELECT ...
FROM ...
WHERE ...
GROUP BY ...
HAVING ...
ORDER BY ...

What’s the Point of RA?

- SQL is a Declarative Language
  - “What to get” rather than “how to get it”
  - Easier to write a SQL query than write a whole Java program that will probably perform worse
- But computers are imperative/procedural
  - Computers only understand the “how”
- RDBMS

History of RA

- Invented/Formalized by Ted Codd while working for IBM
- He realized we need a way to describe imperative programming on tables without knowing physical details
- IBM initially ignored his techniques
History of RA

- Invented/Formalized by Ted Codd while working for IBM
- He realized we need a way to describe imperative programming on tables without knowing physical details
- IBM initially ignored his techniques
- 10 years later he won the Turing Award

October 7, 2019

What’s the Point of RA?

- We need a language that reads more like instructions but still captures the fundamental operations of a query

October 7, 2019

Flashback to our first query

- Relational Algebra (RA) does the job
  - When processing your query, the RDBMS will actually store an RA tree (like a bunch of labeled nodes and pointers)
  - After some optimizations, the RA tree is converted into instructions (like a bunch of functions linked together)

October 7, 2019

Turing Awards in Data Management

- Charles Bachman, 1973
  - IDS and CODASYL

- Ted Codd, 1981
  - Relational model

- Jim Gray, 1998
  - Transaction processing

- Michael Stonebraker, 2014
  - INGRES and Postgres

October 7, 2019
Flashback to our first query

- Code has to boil down to instructions at some point
- Relational Database Management Systems (RDBMSs) use Relational Algebra (RA)

```
SELECT P.Name, P.UserID
FROM Payroll AS P
WHERE P.Job = 'TA';
```

RA Operators

- Symbols are mostly Greek letters like π
  - σ (sigma)
  - γ (gamma)
  - Π (pi) Projection
  - Δ (Delta) Projection
  - Σ (Sigma) Selection
  - Ξ (Xi) Cartesian Product
  - Ψ (Psi) Union
  - Φ (Phi) Intersection

Another example from before...

```
SELECT AVG(P.Salary)
FROM Payroll AS P, Regist AS R
WHERE P.UserID = R.UserID;
```

RA Operators

- These are all the operators you will see in this class
  - We'll profile these one at a time

Join

Cartesian Product

Union

Duplicate Elimination

Sort

Grouping & Aggregation

Projection

Difference
RA Operators

- For the curious...

- Right Outer Join
- Left Outer Join
- Full Outer Join

 Rename

 Left Outer Join

 Right Outer Join

 Full Outer Join

 §

 For the curious...

 Get ready for some math...

 Rename

 Left Outer Join

 Right Outer Join

 Full Outer Join

 §

 Get ready for some math...

 Rename

 Left Outer Join

 Right Outer Join

 Full Outer Join

 §

 Get ready for some math...

 Rename

 Left Outer Join

 Right Outer Join

 Full Outer Join

 §

 Get ready for some math...

 Rename

 Left Outer Join

 Right Outer Join

 Full Outer Join

 §
Sometimes RA can be written in-line

$$\sigma_{T,(A,B,C)}(T(A,B,C)) \rightarrow R(A,B,C)$$

$$\delta(T(A,B,C)) \rightarrow R(A,B,C)$$

$$T(A,B) \cup S(A,B) \rightarrow R(A,B)$$
RA Operators

- Difference
  - Binary operator (but direction matters)
  - Reads as (left input) − (right input)

\[ T(A, B) − S(A, B) \rightarrow R(A, B) \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A | B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>

Basic SQL to RA Conversion

- The general plan structure for a “flat” SQL query

```
γ Tables
σ ⋈ × ⋯
π ORDER BY
خلاف & aggregates
GROUP BY & aggregates
FROM & WHERE

SELECT
σ
itaire
y

```

English to SQL to RA Example

```sql
CREATE TABLE Payroll
(
UserID INT PRIMARY KEY,
Name VARCHAR(100),
Job VARCHAR(100),
Salary INT);

CREATE TABLE Regist
(
UserID INT REFERENCES Payroll,
Car VARCHAR(100));

SELECT DISTINCT P.Name
FROM Payroll AS P
JOIN Regist AS R
ON P.UserID = R.UserID
WHERE P.Job = 'TA'
GROUP BY P.UserID, P.Name
HAVING COUNT(*) > 1
ORDER BY COUNT(*);
```

Name all the TAs that drive multiple cars ordered by the number of cars they drive

```
SELECT DISTINCT P.Name
FROM Payroll AS P
JOIN Regist AS R
ON P.UserID = R.UserID
WHERE P.UserID = '12345'
AND P.Job = 'TA'
GROUP BY P.UserID, P.Name
HAVING COUNT(*) > 1
ORDER BY COUNT(*);
```

Name all the TAs that drive multiple cars ordered by the number of cars they drive

```
SELECT DISTINCT P.Name
FROM Payroll AS P
JOIN Regist AS R
ON P.UserID = R.UserID
WHERE P.UserID = '56789'
AND P.Job = 'TA'
GROUP BY P.UserID, P.Name
HAVING COUNT(*) > 1
ORDER BY COUNT(*);
```

Name all the TAs that drive multiple cars ordered by the number of cars they drive

```
SELECT DISTINCT P.Name
FROM Payroll AS P
JOIN Regist AS R
ON P.UserID = R.UserID
WHERE P.UserID = '98765'
AND P.Job = 'TA'
GROUP BY P.UserID, P.Name
HAVING COUNT(*) > 1
ORDER BY COUNT(*);
```
Name all the TAs that drive multiple cars ordered by the number of cars they drive.

```
SELECT DISTINCT P.Name
FROM Payroll AS P, Regist AS R
WHERE P.UserID = R.UserID
AND P.Job = 'TA'
GROUP BY P.UserID, P.Name
HAVING COUNT(*) > 1
ORDER BY COUNT(*)
```

Summary of RA

- **SQL** = a declarative language where we say what data we want to retrieve
- **RA** = an algebra where we say how we want to retrieve the data
- RDMS translates SQL to RA then optimizes for performance