Updated 2/5/2023

Topic 1: Data models and query languages
Unit 3: Relational Algebra (RA)
Lecture 8

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp23)

https://northeastern-datalab.github.io/cs7240/sp23/

2/3/2023

Pre-class conversations

- Online today: please use your cameras!
- Last class summary
- Please do look through the slides between lectures
- It is time to start to hand in your first scribes (some ideas today)
- Project discussions (in 2 weeks: Fri 2/17: project ideas)

- today:
  - End of Algebra / Codd's theorem
  - Datalog (recursion), and a new tool: Souffle

Algebra and the connection to logic and queries

- Algebra
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RA and safe RC (Codd's theorem)
  - RA → RC
  - RC → RA

DRC → RA: Intuition

Proof (Sketch):

- Show first that for every relational database schema **S**, there is a relational algebra expression **E** such that for every database instance **D**, we have that ADom(**D**) = **E**(**D**).
- Use the above fact and induction on the construction of RC formulas to obtain a translation of RC under the active domain interpretation to RA.

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases

DRC → RA: Intuition

$E(A,B)$

- In this translation, the most interesting part is the simulation of the universal quantifier ∀ in relational algebra

  uses the logical equivalence: $\forall y. \phi \equiv$?

DRC → RA: Intuition

E(A,B)

- In this translation, the most interesting part is the simulation of the universal quantifier ∀ in relational algebra

  uses the logical equivalence: $\forall y.\, \phi \equiv \neg\exists y.\, \neg\phi$

- As an illustration, consider: $\forall y.\, E(x, y) \equiv$?

DRC → RA: Intuition

$E(A,B)$

- In this translation, the most interesting part is the simulation of the universal quantifier ∀ in relational algebra

  uses the logical equivalence: $\forall y.\,\phi \equiv \neg\exists y.\,\neg\phi$

- As an illustration, consider: $\forall y.\,E(x,y) \equiv \neg\exists y.\,\neg E(x,y)$

  and recall: $\mathrm{ADom}(D) =$?

DRC → RA: Intuition

E(A,B)

- In this translation, the most interesting part is the simulation of the universal quantifier ∀ in relational algebra

  uses the logical equivalence: $\forall y. \phi \equiv \neg\exists y. \neg\phi$

- As an illustration, consider: $\forall y. E(x, y) \equiv \neg\exists y. \neg E(x, y)$

  and recall: $\boxed{ADom(D) = \pi_A(E) \cup \pi_B(E)}$

DRC formula ϕ	RA expression for ϕ^{adom}
$\neg E(x, y)$	
$\exists y. \neg E(x, y)$	?
$\neg\exists y. \neg E(x, y)$	

DRC → RA: Intuition

$E(A,B)$

- In this translation, the most interesting part is the simulation of the universal quantifier ∀ in relational algebra

  uses the logical equivalence: $\forall y.\, \phi \equiv \neg\exists y.\, \neg\phi$

- As an illustration, consider: $\forall y.\, E(x, y) \equiv \neg\exists y.\, \neg E(x, y)$

  and recall: $\boxed{ADom(D) = \pi_A(E) \cup \pi_B(E)}$

DRC formula ϕ	RA expression for ϕ^{adom}
$\neg E(x, y)$	$(ADom(D) \times ADom(D)) - E$
$\exists y.\, \neg E(x, y)$	π_{-B} $\pi_A[(ADom(D) \times ADom(D)) - E]$
$\neg\exists y.\, \neg E(x, y)$	$ADom(D) - \pi_A[(ADom(D) \times ADom(D)) - E]$

Entire Story in One Slide (repeated slide)

1. RC = FOL over DB
2. RC can express “bad queries” that depend not only on the DB, but also on the domain from which values are taken (domain dependence)
3. We cannot test whether an RC query is “good,” but we can use a ”good” subset of RC that captures all “good” queries (safety)
4. “Good” RC and RA can express the same queries! (equivalence = Codd's theorem)

Discussion

- What is the monotone fragment of RA ?

?

- What are the safe queries in RA ?

?

- Where do we use RA (applications) ?

?

Source: Dan Suciu, CSE 554, 2011.

Discussion

- What is the monotone fragment of RA ?
  - Basic except difference (−): $\cup$, σ, π, $\bowtie$
- What are the safe queries in RA ?

?

- Where do we use RA (applications) ?

?

Discussion

$A(x) - B(x)$

$\neg B(x)$

- What is the monotone fragment of RA ?
  - Basic except difference (−): ∪, σ, π, ⋈
- What are the safe queries in RA ?
  - All RA queries are safe
- Where do we use RA (applications) ?

?

Discussion

- What is the monotone fragment of RA ?
  - Basic except difference ($-$): $\cup$, σ, π, $\bowtie$
- What are the safe queries in RA ?
  - All RA queries are safe
- Where do we use RA (applications) ?
  - Translating SQL (from WHAT to HOW)
  - Directly as query languages (e.g. Pig-Latin)

See next pages

Source: Dan Suciu, CSE 554, 2011.

EXAMPLE 1. *Suppose we have a table* **urls: (url, category, pagerank)**. *The following is a simple SQL query that finds, for each sufficiently large category, the average pagerank of high-pagerank urls in that category.*

```
SELECT category, AVG(pagerank)
FROM urls WHERE pagerank > 0.2
GROUP BY category HAVING COUNT(*) > 10^6
```

An equivalent Pig Latin program is the following. (Pig Latin is described in detail in Section 3; a detailed understanding of the language is not required to follow this example.)

```
good_urls = FILTER urls BY pagerank > 0.2;
groups = GROUP good_urls BY category;
big_groups = FILTER groups BY COUNT(good_urls)>10^6;
output = FOREACH big_groups GENERATE
            category, AVG(good_urls.pagerank);
```

EXAMPLE 1. *Suppose we have a table* **urls: (url, category, pagerank)**. *The following is a simple SQL query that finds, for each sufficiently large category, the average pagerank of high-pagerank urls in that category.*

```
SELECT category, AVG(pagerank)
FROM urls WHERE pagerank > 0.2
GROUP BY category HAVING COUNT(*) > 10^6
```

An equivalent Pig Latin program is the following. (Pig Latin is described in detail in Section 3; a detailed understanding of the language is not required to follow this example.)

```
good_urls = FILTER urls BY pagerank > 0.2;
groups = GROUP good_urls BY category;
big_groups = FILTER groups BY COUNT(good_urls)>10^6;
output = FOREACH big_groups GENERATE
                category, AVG(good_urls.pagerank);
```

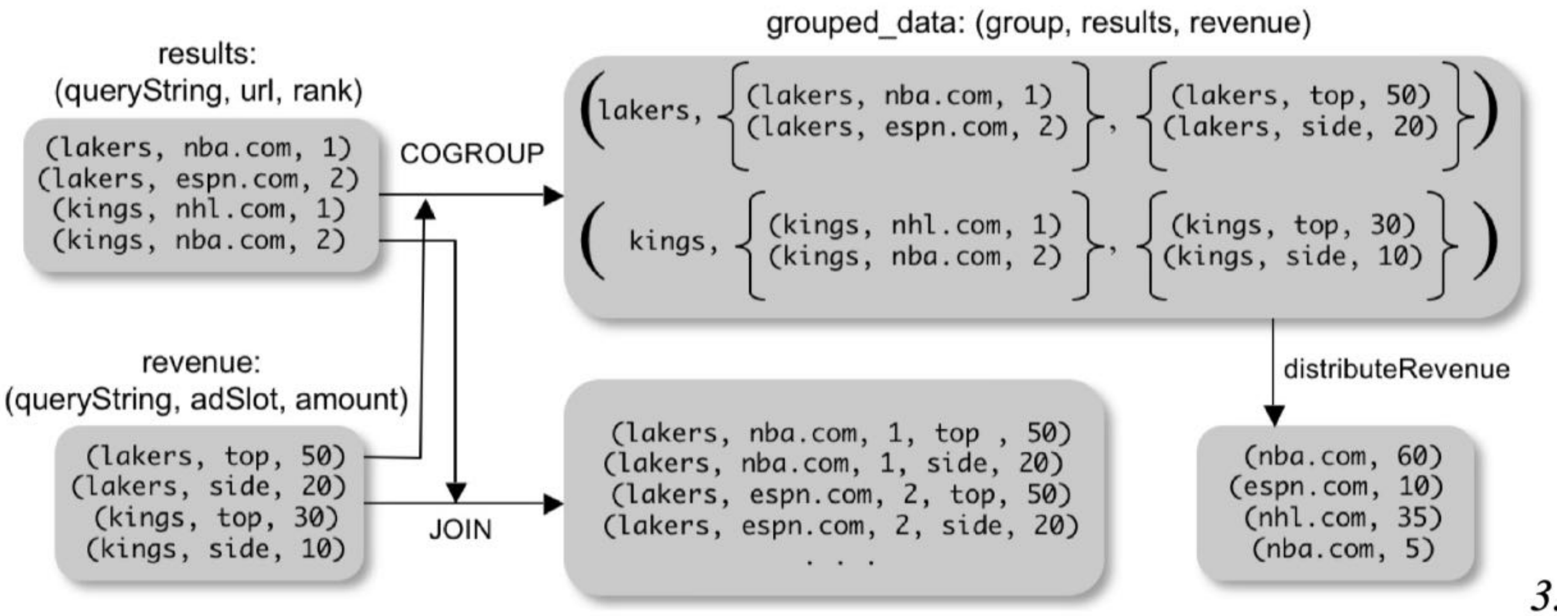

Figure 2: COGROUP versus JOIN.

```
grouped_data  =  COGROUP results BY queryString,
                         revenue BY queryString;
```

3.5.2 JOIN *in Pig Latin*

Not all users need the flexibility offered by COGROUP. In many cases, all that is required is a regular equi-join. Thus, Pig Latin provides a JOIN keyword for equi-joins. For example,

```
join_result  =  JOIN results BY queryString,
                revenue BY queryString;
```

It is easy to verify that JOIN is only a syntactic shortcut for COGROUP followed by flattening. The above join command is equivalent to:

```
   temp_var  =  COGROUP results BY queryString,
                revenue BY queryString;
join_result  =  FOREACH temp_var GENERATE
                FLATTEN(results), FLATTEN(revenue);
```