CptS 415 Big Data

Relational Algebra

Srini Badri

Acknowledgements: Tinghui Wang

Relational Query Languages

- Query Languages (QLs):
  - Allow manipulation and retrieval of data from a database
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic
  - Allows for much optimization
- Query Languages != Programming Languages!
  - QLs not expected to be "Turing complete"
  - QLs not intended to be used for complex calculations
  - QLs support easy, efficient access to large datasets

Formal Relational Query Languages

- Two mathematical Query Languages form the basis for “real” languages (e.g. SQL), and for implementation
- Relational Algebra
  - More operational (procedural), very useful for representing execution plans
- Relational Calculus
  - Let users describe what they want, rather than how to compute it
  - Non-operational, Declarative

Preliminary

- A query is applied to relation instances, and the result of a query is also a relation instance
- Schemas of input relations of a query are fixed
- The schema of the result of a given query is also fixed, determined by definition of query language constructs.

Relational Algebra

- Basic Operations:
  - Selection σ: Selects a subset of rows from a relation
  - Projection π: Deletes unwanted columns from a relation
  - Cross-Product $\times$: Combines two relations
  - Set-Difference $-$: Tuples in R_1 but not in R_2
  - Union $\cup$: Tuples in R_1 and R_2
- Additional operations
  - Intersection, Join, Division, Renaming: Optional, but very useful
- Since each operation returns a relation, operations can be composed
  - The algebra is “closed”

Example Instances

- Sailors and Reserves relations
- Columns:
  - Sid: Sailor ID
  - Bid: Boat ID
- We will use positional or named field notations

S_1

sid	sname	rating	age
22	dustin	7	45
31	luber	8	55.5
58	rusty	10	35

S_2

sid	sname	rating	age
28	yuppy	9	35
31	luber	8	55.5
44	guppy	5	35
58	rusty	10	35

R_1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Selection σ

- Select rows that satisfy selection condition
- Schema of result is identical to schema of (only) input relation
- Result relation can be the input for another relational operation

$$\sigma_{rating>8}(S_2)$$

sid	sname	rating	age
28	yuppy	9	35
58	rusty	10	35

$$\pi_{sname,rating}\left(\sigma_{rating>8}(S_2)\right)$$

sname	rating
yuppy	9
rusty	10

Projection π

- Deletes attributes that are not in projection list
- Schema of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation
- Project operator has to eliminate duplicates!
  - Why?
  - What are the consequences?

S_2

sid	sname	rating	age
28	yuppy	9	35
31	luber	8	55.5
44	guppy	5	35
58	rusty	10	35

$\pi_{sname,\ rating}(S_2)$

sname	rating
yuppy	9
luber	8
guppy	5
rusty	10

$\pi_{age}(S_2)$

age
35
55.5

Union, Intersection, Set-Difference

- All of these operations take two input relations, which must be union-compatible
  - Same number of fields
  - “Corresponding” fields have the same type
- What is the output schema?

S_1

sid	sname	rating	age
22	dustin	7	45
31	luber	8	55.5
58	rusty	10	35

S_2

sid	sname	rating	age
28	yuppy	9	35
31	luber	8	55.5
44	guppy	5	35
58	rusty	10	35

$S_1 - S_2$

sid	sname	rating	age
22	dustin	7	45

$S_1 \cap S_2$

sid	sname	rating	age
31	luber	8	55.5
58	rusty	10	35

Cross-Product $\times$

- Each row of S_1 is paired with each row of R_1
- Result schema has one field per field of S_1 and R_1, with field names "inherited" if possible.
  - Conflict: Both S_1 and R_1 have a field called sid

$S_1 \times R_1$

sid	sname	rating	age	sid	bid	day
22	dustin	7	45	22	101	10/10/96
22	dustin	7	45	58	103	11/12/96
31	luber	8	55.5	22	101	10/10/96
31	luber	8	55.5	58	103	11/12/96
58	rusty	10	35	22	101	10/10/96
58	rusty	10	35	58	103	11/12/96

S_1

sid	sname	rating	age
22	dustin	7	45
31	luber	8	55.5
58	rusty	10	35

R_1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Renaming ρ

- Renaming the attributes in the schema

$$\rho\left(C(1 \to sid1, 5 \to sid2),\ S_1 \times R_1\right)$$

sid1	sname	rating	age	sid2	bid	day
22	dustin	7	45	22	101	10/10/96
22	dustin	7	45	58	103	11/12/96
31	luber	8	55.5	22	101	10/10/96
31	luber	8	55.5	58	103	11/12/96
58	rusty	10	35	22	101	10/10/96
58	rusty	10	35	58	103	11/12/96

Join $\bowtie$

- Conditional Join: $R_1 \bowtie_c S_1 = \sigma_c\left(R_1 \times S_1\right)$
- Result schema same as that of cross-product
- Fewer tuples than cross-product.
  - Filters tuples not satisfying the join condition
- Sometimes called a theta-join

S_1

sid	sname	rating	age
22	dustin	7	45
31	luber	8	55.5
58	rusty	10	35

R_1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

$R_1 \bowtie_{S_1.sid < R_1.sid} S_1$

(sid)	sname	rating	age	sid	bid	day
22	dustin	7	45	58	103	11/12/96
31	luber	8	55.5	58	103	11/12/96

Join (Cont'd)

- Inner Join:
  - Specifies a join between two relations with an explicit join clause
- Left Outer Join
  - Specifies a join between two relations with an explicit join clause
  - preserving unmatched rows from the first table
- Right Outer Join
  - Specifies a join between two tables with an explicit join clause
  - preserving unmatched rows from the second table

Equi-Join $\bowtie_c$

- A special case of condition join where the condition only contains only equalities
- Result schema similar to cross-product, but only one copy of fields for which equality is specified
- Natual Join: Equi-Join on all common fields

S_1

sid	sname	rating	age
22	dustin	7	45
31	luber	8	55.5
58	rusty	10	35

$$\pi_{sid,\ \ldots,age,bid,\ldots}\left(R_1 \bowtie_{sid} S_1\right)$$

sid	sname	rating	age	bid	day
22	dustin	7	45	101	10/10/96
58	rusty	10	35	103	11/12/96

R_1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Division /

- Not supported as a primitive operator, but useful for expressing queries like
  - Find sailors who have reserved all boats
- Precondition: in A/B the attributes in B must be included in the schema for A, also, the results has attributes $A - B$
  - SALES(supId, prodId)
  - PRODUCTS(prodId)
  - Relations SALES and PRODUCTS must be built using projections
  - SALES/PRODUCTS: the ids of the suppliers supplying all products

Example of Division

A

sid	pid
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B_1

pid
p2

B_1

pid
p2
p4

B_1

pid
p1
p2
p4

A / B_1

sid
s1
s2
s3
s4

A / B_2

sid
s1
s4

A / B_3

sid
s1

Expressing A/B Using Basic Operators

- Devision is not an essential operator, i.e. composite
  - Also true of joins, but joins are so common that almost all systems implement joins.
  - Division is NOT implemented in SQL
- Idea: For SALES/PRODUCTS, compute all products such that there exists at least one supplier not supplying it
  - X value is disqualified if by attaching y value from B, we obtain an xy tuple that is not in C

$$C = \pi_{sid}\Big(\big(\pi_{sid}(Sales) \times Products\big) - Sales\Big)$$

$$Sales / Products = \pi_{sid}(Sales) - C$$

Find names of sailors who have reserved boat #103

- Solution 1:

$$\pi_{sname}\left(\left(\sigma_{bid=103}(R_1)\right) \bowtie Sailors\right)$$

- Solution 2:

$$\rho\left(Temp1, \left(\sigma_{bid=103}(R_1)\right)\right)$$

$$\rho(Temp2, Temp1 \bowtie S_1)$$

$$\pi_{sname}(Temp2)$$

- Solution 3:

$$\pi_{sname}\left(\sigma_{bid=103}(R_1 \bowtie S_1)\right)$$

- Which one is fastest?
- Optimization: push-down selection

Find names of sailors who've reserved a red boat

- Information about boat color only available in Boats; so need an extra join:

$$\pi_{sname}\left(\left(\sigma_{color='red'}(B)\right) \bowtie R_1 \bowtie S_1\right)$$

- A more efficient solution

$$\pi_{sname}\left(\pi_{sid}\left(\left(\pi_{bid}\sigma_{color='red'}(B)\right) \bowtie R_1\right) \bowtie S_1\right)$$

A query optimizer can find this, given the first solution

Find sailors who've reserved a red or a green boat

- Can identify all red or green boats, then find sailors who've reserved one of these boats:

$$\pi_{sname}\left(\pi_{bid}\left(\sigma_{color='red' \vee color='green'}(B)\right) \bowtie R \bowtie S\right)$$

- Or, use union operator

$$\pi_{sname}\left(\pi_{bid}\left(\sigma_{color='red'}(B) \cup \sigma_{color='green'}(B)\right) \bowtie R \bowtie S\right)$$

- What if $\vee$ is replaced by $\wedge$?

Find sailors who've reserved a red <u>and</u> a green boat

- Previous approach won't work!
- Must identify
  - sailors who have reserved red boats
  - sailors who have reserved green boats
- Then Find intersection

$$\rho\left(S_{red},\ \pi_{sid}\left(\sigma_{color='red'}(B) \bowtie R\right)\right)$$

$$\rho\left(S_{green},\ \pi_{sid}\left(\sigma_{color='green'}(B) \bowtie R\right)\right)$$

$$\pi_{sname}\left(\left(S_{red} \cap S_{green}\right) \bowtie S\right)$$

Find the names of sailors who've reserved all boats

- Uses division
  - Schemas of the input relations to division must be carefully chosen!

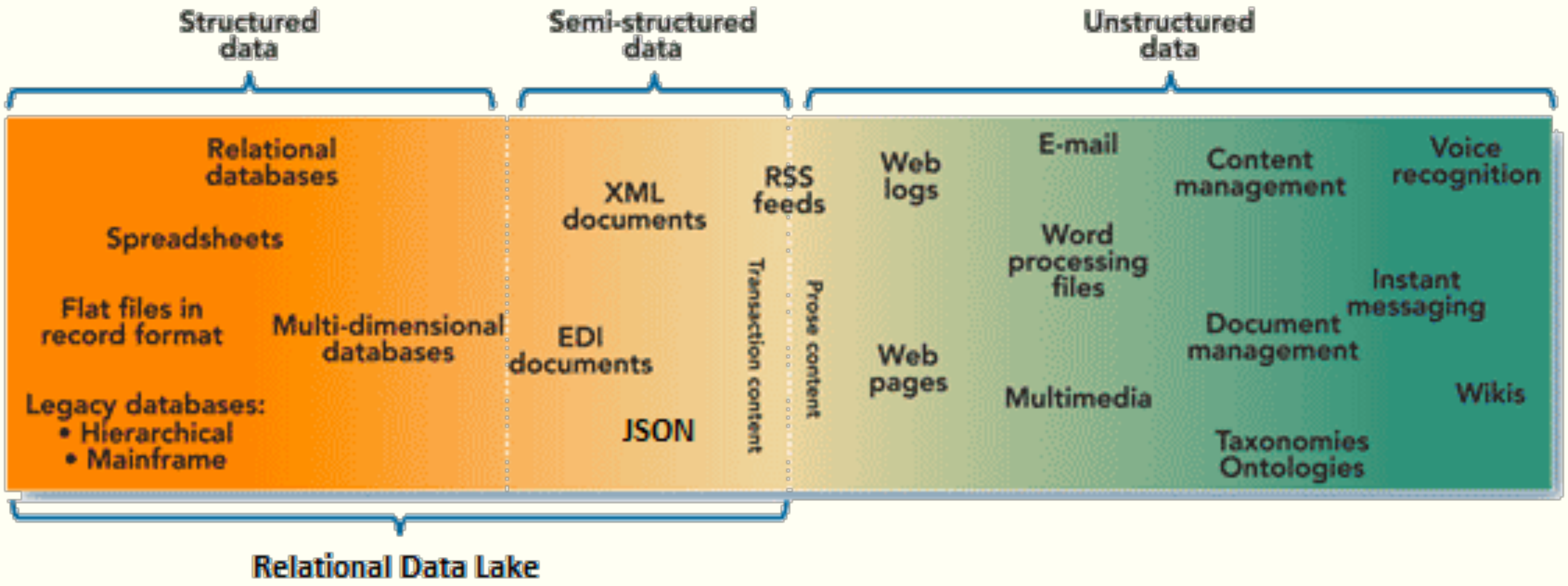
$$\rho\left(S_{sids},\ \left(\pi_{sid,\ bid}(R)/\pi_{bid}(B)\right)\right)$$

$$\pi_{sname}\left(S_{sids} \bowtie S\right)$$

- To find sailors who've reserved all 'interlake' boats:

$$\ldots/\pi_{bid}\left(\sigma_{bname='interlake'}(B)\right)$$

Summary



Structured data
Semi-structured data
Unstructured data
Relational databases
Spreadsheets
Flat files in record format
Multi-dimensional databases
Legacy databases:
• Hierarchical
• Mainframe
XML documents
EDI documents
JSON
RSS feeds
Transaction content
Prose content
Web logs
Web pages
E-mail
Word processing files
Multimedia
Content management
Document management
Taxonomies Ontologies
Voice recognition
Instant messaging
Wikis
Relational Data Lake