Relational Algebra

Relational Query Languages

- Query languages: allow manipulation and retrieval of data from a database
- Relational model supports simple, powerful QLs:
  - Strong formal foundation
  - Allows for much optimization
- Query languages are NOT programming languages
  - QLs not expected to be "turing complete"
  - QLs not intended to be used for complex calculations
  - QLs support easy, efficient and sophisticated access to large data sets

Formal Relational Query Languages

- Mathematical query languages form the basis for "real" languages (e.g. SQL) and for implementation:
  - **Relational Algebra**:
    - Operational - a query is a sequence of operations on data
    - Very useful for representing execution plans, i.e., to describe how a SQL query is executed internally
    - NOTE: the operational semantics given by relational algebra is one possible way of how a query might be executed, but DBS can implement a query in various different ways
  - **Relational Calculus:**
    - descriptive - a query describes how the data to be retrieved looks like

- Understanding Relational Algebra is key to understanding SQL, PigLatin, OQL, Xquery,….

Example Relations

Used in all Database courses But really BAD

Skaters

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

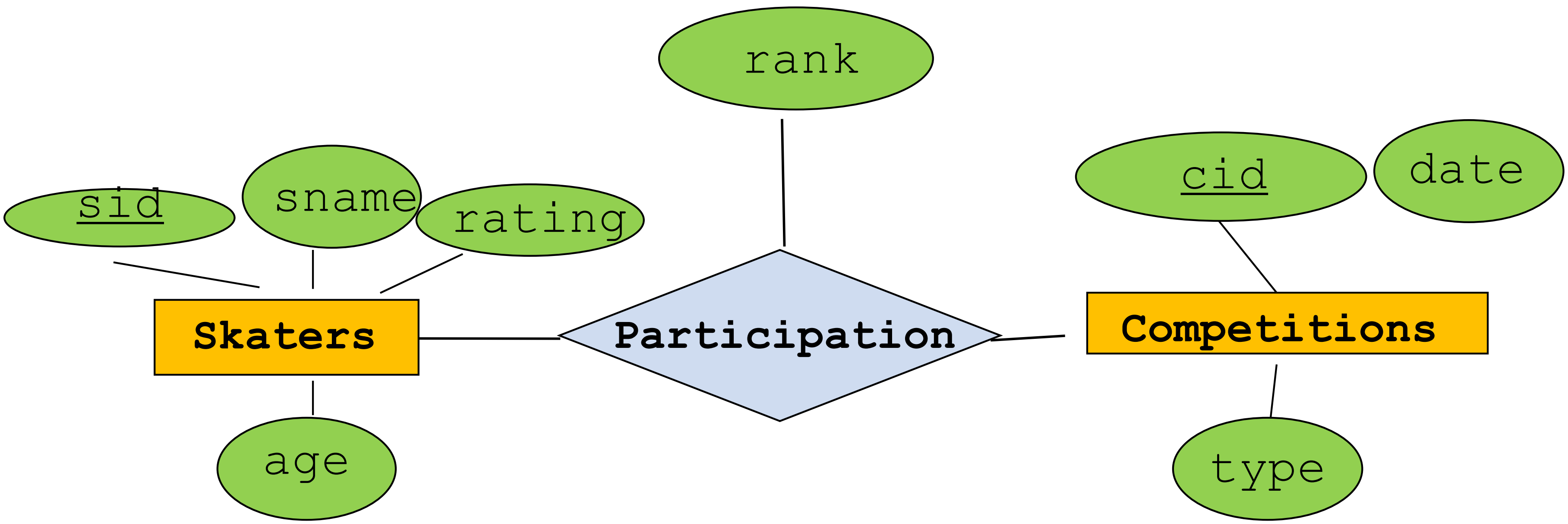
Participates

sid	cid	rank
31	101	2
58	103	7
58	101	7
58	104	1

Competitions

cid	date	type
101	12/13/2015	local
103	01/12/2016	regional
104	01/20/2016	local

Comes from… E/R



Relational Algebra: Basics

- RA consists of a set of basic *operators*
  - *Input:* one or two relations
    - schema of each relation is known
    - instance can be arbitrary
  - *Output:* a relation
    - schema of output relation depends on operator and input relations
- Relational algebra is closed:
  - since each operation has input relation(s), and returns a relation, operations can be composed
- Does not assume special primary key attributes in the relation
- Assumes a relation is a set
  - No two tuples have the same values in all attributes

Relational Algebra: Operations

- Single relation as input
  - **Selection σ:** Selects a subset of tuples from a relation
  - **Projection $\prod$:** projects to a subset of attributes from a relation
  - **Renaming ρ:** of relations or attributes; useful when combining several operators
- Two relations as input
  - **Cross Product X:** Combines two relations
  - **Join $\bowtie$:** Combination of Cross product and selection
  - **(Division):** not covered in class
- Set operators with two relations as input
  - **Intersection ($\cap$)**
  - **Union ($\cup$)**
  - **Set Difference –:** Tuples that are in the first but not the second relation

Projection: $\Pi_L(R_{in})$

- Returns the subset of the attributes of the input relation R_{in} that are in the projection list L
- Schema of result relation contains exactly the attributes of the projection list, with the same attribute names as in R_{in}

Skaters

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

$\Pi_{\texttt{sname,rating}}(\texttt{Skaters})$

sname	rating
yuppy	9
debby	7
conny	5
lilly	10

Projection: $\prod_L(R_{in})$

- Possible Operational Semantics:
  - Imagine a tuple variable iterating over all tuples in the relation
  - for each tuple: extract the projected attributes and output the reduced tuple in result relation
  - eliminate duplicates
    - Note: real systems typically do NOT eliminate duplicates unless the user explicitly asks for it; eliminating duplicates is a very costly operation!

Skaters

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

$\prod_{age}$ (**Skaters**)

age
15
10
13

Selection: $\sigma_C(R_{in})$

- *Selection:* $\sigma_C(R_{in})$
  - Schema of result relation identical to schema of R_{in}
  - Returns the subset of the rows of the input relation R_{in} that fulfill the condition C
  - Condition C involves the attributes of R_{in}
  - No duplicates (obviously)
- Operational Semantics
  - Imagine a tuple variable ranging over all tuples in the relation
  - for each tuple: check whether condition C is satisfied. If so, output the tuple into the result relation

Skaters

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

$\sigma_{\texttt{rating > 8}}$ **(Skaters)**

sid	sname	rating	age
28	yuppy	9	15
58	lilly	10	13

Operator Composition

- result relation of one operation can be input for another relational algebra operation

$$\Pi_{\texttt{sname,rating}}(\sigma_{\texttt{rating > 8}}(\texttt{Skaters}))$$

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

sid	sname	rating	age
28	yuppy	9	15
58	lilly	10	13

sname	rating
yuppy	9
lilly	10

- Operational Semantics:
  - Option 1:
    - Stepwise one operator at a time:
    - build intermediate temporary relations

Operator Composition

- result relation of one operation can be input for another relational algebra operation

$$\Pi_{\texttt{sname,rating}} (\sigma_{\texttt{rating > 8}}(\texttt{Skaters}))$$

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

sname	rating
yuppy	9
lilly	10

- Operational Semantics:
  - Option 2:
    - Consecutive operators on the fly: one scan through the relation
    - Not always possible

Union, Intersection, Set-Difference

- Notation:
  - $R_{in1} \cup R_{in2}$ (Union),
  - $R_{in1} \cap R_{in2}$ (Intersection),
  - R_{in1} - R_{in2} (Difference),
- Usual operations on sets
- R_{in1} and R_{in2} must be set-compatible,
  - same number of attributes
  - corresponding attributes must have the same type
  - no need for same name
- Result schema
  - same as the schema of the input relations
  - possibly renamed attributes

Example Tables

Skaters (Table of DB of the Glacier Club)

<u>sid</u>	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

OurSkaters (Table of DB of the Icy Club)

<u>id</u>	name	rating	age
28	yuppy	9	15
25	debby	7	10
27	willy	8	8

Union

Skaters

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

OurSkaters

id	name	rating	age
28	yuppy	9	15
22	debby	7	10
27	willy	8	8

Skaters ∪ OurSkaters

		rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13
22	debby	7	10
27	willy	8	8

Intersection

Skaters

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

OurSkaters

id	name	rating	age
28	yuppy	9	15
25	debby	7	10
27	willy	8	8

Skaters ∩ OurSkaters

		rating	age
28	yuppy	9	15

Difference

Skaters

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

OurSkaters

id	name	rating	age
28	yuppy	9	15
25	debby	7	10
27	willy	8	8

Skaters - OurSkaters

		rating	age
31	debby	7	10
22	conny	5	10
58	lilly	10	13

Concatenation of operators

Skaters

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

OurSkaters

id	name	rating	age
28	yuppy	9	15
25	debby	7	10
27	willy	8	8

$\Pi_{\texttt{sname, rating, age}}$ **(Skaters)** $\cap$
$\Pi_{\texttt{name, rating, age}}$ **(OurSkaters)**

	rating	age
yuppy	9	15
debby	7	10

Operational Semantics

- Take your favorite set-operator algorithm discussed in COMP 250/251, MATH 240
- Possible implementation for Intersection
  - For each tuple in first relation
    - For each tuple in second relation
      - If tuples are equal: output
- Possible implementation for Difference
  - For each tuple in first relation
    - For each tuple in second
      - If tuples are equal: exit loop / no output
    - If no early exit: output
- Union

Relational Algebra Quiz

Cross-Product

- *Cross-Product*: R_{in1} **X** R_{in2}
- Each row of R_{in1} is paired with each row of R_{in2}
- Result schema
  - one attribute per attribute of R_{in1}
  - one attribute per attribute R_{in2}
  - field names inherited
    - if both have attribute with same name: prefix with relation name
- Operational semantics
  - Consider a tuple variable t1 for first relation;
  - Consider a tuple variable t2 for second relation

for each assignment of t1

for each assignment of t2

combine all attribute values of t1 and t2 and output them as one tuple into result relation; prefix attribute names with relation name

Cross-Product

Skaters

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10

Participates

sid	cid	rank
31	101	2
58	103	7

Skaters X Participates

S.sid	sname	rating	age	P.sid	cid	rank
28	yuppy	9	15	31	101	2
28	yuppy	9	15	58	103	7
31	debby	7	10	31	101	2
31	debby	7	10	58	103	7
22	conny	5	10	31	101	2
22	conny	5	10	58	103	7

Joins

Cross-Product +

Selection with attributes from both relations

It's what makes an advanced query language

-- the way to relate tables

Joins

- ***Condition Join (Theta-Join):*** $R_{out} = R_{in1} \bowtie_{\mathbf{c}} R_{in2} = \sigma_C(R_{in1} \mathbf{X} R_{in2})$
- Result schema the same as for cross-product

Skaters

<u>sid</u>	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

OurSkaters

<u>id</u>	name	rating	age
28	yuppy	9	15
25	debby	7	10
27	willy	8	8

$\texttt{Skaters} \bowtie_{\texttt{Skaters.rating > OurSkaters.rating}} \texttt{OurSkaters}$

Joins

- *Condition Join (Theta-Join)*: $R_{out} = R_{in1} \bowtie_{c} R_{in2} = \sigma_C(R_{in1} \times R_{in2})$
- Result schema the same as for cross-product

Skaters

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

OurSkaters

id	name	rating	age
28	yuppy	9	15
25	debby	7	10
27	willy	8	8

sid	sname	Skaters .rating	Skaters. age	id	name	OurSkaters. rating	OurSkaters. age
28	yuppy	9	15	25	debby	7	10
28	yuppy	9	15	27	willy	8	8
58	lilly	10	13	28	yuppy	9	15
58	lilly	10	13	25	debby	7	10
58	lilly	10	13	27	willy	8	8

Operational Semantics

Consider a tuple variable t1 for first relation;
Consider a tuple variable t2 for second relation

for each assignment of t1

 for each assignment of t2

  if condition C is true, combine all attribute values of t1 and t2 and output them as one tuple into result relation; prefix attribute names with relation name

Equi-Join

- ***Equi-Join:*** $R_{out} = R_{in1} \bowtie_{Rin1.a1\ =\ Rin2.b1\ \wedge\ \ldots\ Rin1.an\ =\ Rin2.bn}\ R_{in2}$
- A special case of condition join where the condition C contains only *equalities*.
- Result schema similar to cross-product,
  - only *one copy of attributes* for which equality is specified

Equi-Join

Skaters

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	7	10
58	lilly	10	13

OurSkaters

id	name	rating	age
28	yuppy	9	15
25	debby	7	10
27	willy	8	8

$\texttt{Skaters} \bowtie_{\texttt{Skaters.rating = OurSkaters.rating}} \texttt{OurSkaters}$

sid	sname	rating	Skaters. age	id	name	OurSkaters. age
28	yuppy	9	15	28	yuppy	15
31	debby	7	10	25	debby	10
22	conny	7	10	25	debby	10

There is only one rating attribute in the output relation.

Natural Join

- ***Natural Join***: Equijoin on all common attributes, i.e., on all attributes with the same name
  - *Attributes do not need to be indicated* in index of join symbol

Skaters

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	7	10
58	lilly	10	13

Participates

sid	cid	rank
31	101	2
22	103	7
31	103	1
…		

Skaters $\bowtie$ Participates

sid	sname	rating	age	cid	rank
31	debby	7	10	101	2
31	debby	7	10	103	1
22	conny	5	10	103	7

Renaming

- ***Renaming :*** $\rho(R_{out}(B1,\ldots Bn), R_{in}(A1, \ldots An))$
  - Produces a relation identical to R_{in}
  - Output relation is named R_{out}
  - Attributes A1, … An of R_{in} renamed to B1, … Bn

ρ(Temp, Skaters),
ρ(Temp1(sid1,rating1), Skaters(sid,rating)))

Examples (discussed in class)

- Relations
  - Skaters(sid,sname,rating,age)
  - Participates(sid,cid,day)
  - Competition(cid,date,type)
- Queries
  - Find names of skaters who have participated in competition #103 (three solutions)
  - Find names of skaters who have participated in a local competition (2 solutions)
  - Find sids of skaters who have participated in a local or regional competition (1 solution)
  - Find name of skaters who have participated in a local or regional competition
  - Find sids of skaters who have participated in a local and regional competition (2 solutions)

– Find names of skaters who have participated in competition #101 (two solutions)

S

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

P

sid	cid	rank
31	101	2
58	103	7
58	101	7
58	104	1

C

cid	date	type
101	12/13/2014	local
103	01/12/2015	regional
104	01/20/2015	local

$$\Pi_{sname}(\sigma_{cid=101}(P) \bowtie S)$$

$$\Pi_{sname}(\sigma_{cid=101}(S \bowtie P))$$

– Find names of skaters who have participated in a local competition (3 solutions)

S

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

P

sid	cid	rank
31	101	2
58	103	7
58	101	7
58	104	1

C

cid	date	type
101	12/13/2014	local
103	01/12/2015	regional
104	01/20/2015	local

$$\Pi_{sname}(\sigma_{type='local'}(C \bowtie P \bowtie S))$$

$$\Pi_{sname}(\sigma_{type='local'}(C) \bowtie P \bowtie S)$$

$$\rho(Temp, \sigma_{type='local'}(C) \bowtie P)$$
$$\Pi_{sname}(Temp \bowtie S)$$

– Find sids of skaters who have participated in a local or regional competition (2 solutions)

S

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

P

sid	cid	rank
31	101	2
58	103	7
58	101	7
58	104	1

C

cid	date	type
101	12/13/2014	local
103	01/12/2015	regional
104	01/20/2015	local

$$\Pi_{sid}(\sigma_{ctype='local' \vee ctype='regional'}(C) \bowtie P)$$

$$\Pi_{sid}(\sigma_{ctype='local'}(C) \bowtie P) \cup \Pi_{sid}(\sigma_{ctype='regional'}(C) \bowtie P)$$

– Find names of skaters who have participated in a local or regional competition (1 solution)

S

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

P

sid	cid	rank
31	101	2
58	103	7
58	101	7
58	104	1

C

cid	date	type
101	12/13/2014	local
103	01/12/2015	regional
104	01/20/2015	local

$$\rho(Temp, \sigma_{ctype='local' \vee ctype='regional'}(C) \bowtie P)$$
$$\Pi_{sname}(Temp \bowtie S)$$

– Find sids of skaters who have participated in a local AND a regional competition (2 solution)

S

sid	sname	rating	age
28	yuppy	9	15
31	debby	7	10
22	conny	5	10
58	lilly	10	13

P

sid	cid	rank
31	101	2
58	103	7
58	101	7
58	104	1

C

cid	date	type
101	12/13/2014	local
103	01/12/2015	regional
104	01/20/2015	local

$\rho(Locals, \Pi_{sid}(\sigma_{ctype='local'}(C) \bowtie P))$

$\rho(Regionals, \Pi_{sid}(\sigma_{ctype='regional'}(C) \bowtie P))$

$Locals \cap Regionals$

WRONG !!

$\rho(Temp, \sigma_{ctype='local' \wedge ctype='regional'}(C) \bowtie P)$

Some rules and definitions

- Equivalence: Let R, S, T be relations; C, C1, C2 conditions; L projection lists of the relations R and S
  - Commutativity:
    - $\prod_L(\sigma_C(R)) = \sigma_C(\prod_L(R))$
      - But only if C only considers attributes of L
    - $R1 \bowtie R2 = R2 \bowtie R1$
  - Associativity:
    - $R1 \bowtie (R2 \bowtie R3) = (R1 \bowtie R2) \bowtie R3$
  - Idempotence:
    - $\prod_{L2} (\prod_{L1}(R)) = \prod_{L2} (R)$
      - Only if $L2 \subseteq L1$
    - $\sigma_{C2}(\sigma_{C1}(R)) = \sigma_{C1 \wedge C2}(R))$

Summary

- The relational model has rigorously defined query languages that are simple and powerful
- Relational algebra is operational; useful as internal representation for query evaluation plans
- Several ways of expressing a given query; a query optimizer should choose the most efficient version.
- *<u>Relational Completeness</u>* of a query language: A query language (e.g., SQL) can express every query that is expressible in relational algebra