ICS 321 Spring 2012
The Relational Model of Data (i)

Asst. Prof. Lipyeow Lim
Information & Computer Science Department
University of Hawaii at Manoa

Data Models

A data model is a collection of concepts for describing data

- Structure of the data.
  - More of a *conceptual model* rather than a *physical data model*. Eg. Arrays, objects in C/C++
- Operations on the data
  - *Queries* and *modifications* only
- Constraints on the data
  - Limitations on the data. Eg. Data type etc.

Examples: the relational model and the semi-structured model (XML)

The Relational Model

- *Relational database:* a set of *relations*
- A *relation* is made up of 2 parts:
  - *Instance* : a *table,* with rows and columns. #Rows = *cardinality,* #fields = *degree / arity.*
  - *Schema* : specifies name of relation, plus name and domain/type of each column or attribute.
    - E.G. Students(sid: string, name: string, login: string, age: integer, gpa: real).
- Can think of a relation as a *set* of rows or *tuples* (i.e., all rows are distinct).

Example Instance of Students Relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Cardinality = 3, degree=5, all rows distinct
- Do all columns in a relation instance have to be distinct?

Relational Query Languages

- The relational model supports simple, powerful *querying* of data.
- Queries are written declaratively in SQL, and the DBMS finds an efficient execution plan.
- Query Languages **!=** programming languages!
- Two mathematical query languages
  - *Relational Algebra*: More operational, useful for representing query execution plans.
  - *Relational Calculus*: More declarative

Preliminaries

- A query takes *relation instances* as input and outputs a relation instance.
- Positional vs. named-field notation:
  - Named-field notation more readable.
  - Both used in SQL
  - Field names in query results are `inherited’ from input relations
- “Sailors” and “Reserves” relations for our examples.

R1

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

S1

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

S2

sid	sname	rating	age
28	Yuppy	9	35.0
31	Lubber	8	55.5
44	Guppy	5	35.0
58	Rusty	10	35.0

Relational Algebra

- Basic operations:
  - *Selection* (σ) Selects a subset of rows from relation.
  - *Projection* (π) Deletes unwanted columns from relation.
  - *Cross-product* (×) Allows us to combine two relations.
  - *Set-difference* (−) Tuples in reln. 1, but not in reln. 2.
  - *Union* (U) Tuples in reln. 1 and in reln. 2.
- Additional operations:
  - Intersection, *join*, division, renaming: Not essential, but (very!) useful.
- Since each operation returns a relation, operations can be *composed*! (Algebra is “closed”.)

Projection

- 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.
- Projection operator has to eliminate *duplicates*! (Why??)
- Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it. (Why not?)

$\pi_{sname, rating}(S2)$

sname	rating
Yuppy	9
Lubber	8
Guppy	5
Rusty	10

$\pi_{age}(S2)$

age
35.0
55.5
35.0
35.0

Selection

- Selects rows that satisfy *selection condition*.
- No duplicates in result! (Why?)
- *Schema* of result identical to schema of (only) input relation.
- *Result* relation can be the *input* for another relational algebra operation! (*Operator composition.*)

$\sigma_{rating > 8}(S2)$

sid	sname	rating	age
28	Yuppy	9	35.0
~~31~~	~~Lubber~~	~~8~~	~~55.5~~
~~44~~	~~Guppy~~	~~5~~	~~35.0~~
58	Rusty	10	35.0

$\pi_{sname, rating}(\sigma_{rating>8}(S2))$

~~sid~~	sname	rating	~~age~~
~~28~~	Yuppy	9	~~35.0~~
~~31~~	~~Lubber~~	~~8~~	~~55.5~~
~~44~~	~~Guppy~~	~~5~~	~~35.0~~
~~58~~	Rusty	10	~~35.0~~

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 schema of result?

S1

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

S2

sid	sname	rating	age
28	Yuppy	9	35.0
31	Lubber	8	55.5
44	Guppy	5	35.0
58	Rusty	10	35.0

S1 U S2

sid	sname	rating	age
22	Dustin	7	45.0
28	Yuppy	9	35.0
31	Lubber	8	55.5
44	Guppy	5	35.0
58	Rusty	10	35.0

Intersection & Set-Difference

S1 ∩ S2

sid	sname	rating	age
31	Lubber	8	55.5
58	Rusty	10	35.0

S1 − S2

sid	sname	rating	age
22	Dustin	7	45.0

S1

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

S2

sid	sname	rating	age
28	Yuppy	9	35.0
31	Lubber	8	55.5
44	Guppy	5	35.0
58	Rusty	10	35.0

Cross-Product

- Consider the cross product of S1 with R1
- Each row of S1 is paired with each row of R1.
- *Result schema* has one field per field of S1 and R1, with field names `inherited' if possible.
  – *Conflict*: Both S1 and R1 have a field called *sid*.
  – Rename to sid1 and sid2

R1

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

S1

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

S1 × R1

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	Lubber	8	55.5	22	101	10/10/96
31	Lubber	8	55.5	58	103	11/12/96
58	Rusty	10	35.0	22	101	10/10/96
58	Rusty	10	35.0	58	103	11/12/96

Renaming

- The expression:

  ρ (C (1 → sid1, 5 → sid2), S1 × R1)
- Renames the result of the cross product of S1 and R1 to "C"
- Renames column 1 to sid1 and column 5 to sid2

ρ (C (1 → sid1, 5 → sid2), S1 × R1)

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	Lubber	8	55.5	22	101	10/10/96
31	Lubber	8	55.5	58	103	11/12/96
58	Rusty	10	35.0	22	101	10/10/96
58	Rusty	10	35.0	58	103	11/12/96

Joins

- *Condition Join*: $R \bowtie_c S = \sigma_c(R \times S)$
- *Result schema* same as that of cross-product.
- Fewer tuples than cross-product, might be able to compute more efficiently
- Sometimes called a *theta-join*.

$$S1 \bowtie_{S1.sid < R1.sid} R1$$

sid	sname	rating	age	sid	bid	day
22	Dustin	7	45	58	103	11/12/96
31	Lubber	8	55.5	58	103	11/12/96

Equi-Joins & Natural Joins

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

$$S1 \bowtie_{sid} R1$$

sid	sname	rating	age	bid	day
22	Dustin	7	45	101	10/10/96
58	Rusty	10	35.0	103	11/12/96

Find names of sailors who've reserved boat #103

Solution 1: $\pi_{sname}((\sigma_{bid=103} \text{Reserves}) \bowtie Sailors)$

Solution 2: $\rho\ (Temp1,\ \sigma_{bid=103} \text{Reserves})$

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

$\pi_{sname}(Temp2)$

Solution 3: $\pi_{sname}(\sigma_{bid=103}(\text{Reserves} \bowtie Sailors))$

Find names of sailors who've reserved a red boat

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

$$\pi_{sname}((\sigma_{color='red'} Boats) \bowtie \mathrm{Re}serves \bowtie Sailors)$$

- A more efficient solution:

$$\pi_{sname}(\pi_{sid}((\pi_{bid}\sigma_{color='red'} Boats) \bowtie \mathrm{Re}s) \bowtie Sailors)$$

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:

$$\rho\ (Tempboats, (\sigma_{color='red' \vee color='green'}\ Boats))$$

$$\pi_{sname}(Tempboats \bowtie \mathrm{Re}serves \bowtie Sailors)$$

- Can also define Tempboats using union! (How?)
- What happens if $\vee$ is replaced by $\wedge$ in this query?

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

- Previous approach won't work! Must identify sailors who've reserved red boats, sailors who've reserved green boats, then find the intersection (note that *sid* is a key for Sailors):

$$\rho\ (Tempred, \pi_{sid}((\sigma_{color='red'} Boats) \bowtie \text{Reserves}))$$

$$\rho\ (Tempgreen, \pi_{sid}((\sigma_{color='green'} Boats) \bowtie \text{Reserves}))$$

$$\pi_{sname}((Tempred \cap Tempgreen) \bowtie Sailors)$$

Summary

- The Relational Data Model
- Two theoretical relational query languages: relational algebra & relational calculus
- Relational Algebra (RA) operators: selection, projection, cross-product, set difference, union, intersection, join, division, renaming
- Operators are closed and can be composed
- RA is more operational and could be used as internal representation for query evaluation plans.
- For the same query, the RA expression is not unique.
- Query optimizer can choose the most efficient version.