Relational Model

and

Relational Algebra

The Relational Data Model

By far, the most significant invention in the history of DBMS

– E.F. Codd, 1970

– Completely revolutionized the field– Before it, network and hierarchical model: difficult to use and pose queries

– IBM System R: started in 1973 and completed around 1980. Later replaced by IBM DB2 system

• Simplicity at the logical level: a system optimizer takes care of performance.

•Dominant in commercial world: IBM, Oracle, and Microsoft are the main players

• A rich field of research conributions: E.g., Turing Awards E.F. Codd in 1981, Jim Gray in 1998, Mike Stonebraker in 2015.

Relational Model

- Example: Student(sid, name, addr, age, GPA)

sid	name	addr	age	GPA
301	John	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

- All data is represented as relations (= tables)
- Each relation has a set of attributes (= columns)
- Each relation contains of a set of tuples (= rows)
- Each attribute has a domain (= type)
- – Only atomic types

Student(sid, name, addr, age, GPA)

sid	name	addr	age	GPA
301	John	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

Class(dept, cnum, sec, unit, title, instructor)

dept	cnum	sec	unit	title	instructor
CS	112	01	03	Modeling	Dick Muntz
CS	143	01	04	DB Systems	Carlo Zaniolo
EE	143	01	03	Signal	Dick Muntz
ME	183	02	05	Mechanics	Susan Tracey

Enroll(sid, dept, cnum, sec)

sid	dept	cnum	sec
301	CS	112	01
301	CS	143	01
303	EE	143	01
303	CS	112	01
401	CS	112	01

Examples of Queries

- Q1: Find the titles and instructors of all CS courses
- Q2: All student names and GPAs who live on Wilshire
- Q3: Find all student names who take CS classes.
- Q4: All names of students and instructors
- Q5: Find the names of pairs of students who live on the same addr
- Q6: Find students taking at least two classes
- Q7: Find all student names who take **all** CS classes (offered this quarter)
- Q8: Find the names of students who is taking no CS classes (offered this quarter)

Relational Databases

- Schema: defines the tables and data types for each table. Data Definition Language (DDL)
- The Query Language. Two flavors:
  1. Relational Algebra: used as the basis of the actual execution model.
  2. Relational Calculus languages, logic-oriented and more declarative. Structured query language (SQL)

Relational Databases (cont).

- Schema: defines the tables and data types for each table. Data Definition Language (DDL)
- SQL and other DB languages are often called Data Manipulation Languages, because besides query can be used to request the following DB changes:
  1. Insert into a table,
  2. Delete from a table
  3. Update the records in a table

  ◆Closure Properties: by applying queries or updat on tables we get back tables: e.g. Relation → algebra → Relation

Relational Algebra

Module 3, Lecture 1 from

Database Management Systems,
R. Ramakrishnan et al.

Preliminaries

- A query is applied to *relation instances*, and the result of a query is also a relation instance.
  - *Schemas* of input relations for a query are fixed (but query will run regardless of instance!)
  - The schema for the *result* of a given query is also fixed! Determined by definition of query language constructs.
- Positional vs. named-field notation:
  - Positional notation easier for formal definitions, named-field notation more readable.
  - Both used in SQL

Example Instances

- "Sailors" and "Reserves" relations for our examples.
- We'll use named field notation, assume that names of fields in query results are `inherited' from names of fields in query input relations.
- Relations are sets:

-- order of tuples is irrelevant

-- repetitions are irrelevants and can be ignored (thus they can be eliminated)

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* Selects a subset of rows from relation. σ
  - *Projection* Deletes unwanted columns from relation. π
  - *Cross-product* Allows us to combine two relations. $\times$
  - *Set-difference* Tuples in reln. 1, but not in reln. 2. $-$
  - *Union* Tuples in reln. 1 and in reln. 2. $\cup$
- 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 dont do duplicate elimination unless the user explicitly asks for it. (Why not?)

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

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

age
35.0
55.5

$$\pi_{age}(S2)$$

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.*)

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

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

sname	rating
yuppy	9
rusty	10

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

Union, Intersection, Set-Difference

- All of these operations take two input relations, which must be *union-compatible*:
  - Same number of columns
  - `Corresponding' columns have the same type.
- What is the *schema* of result?

$$S1 \cup S2$$

$$S1 \cap S2$$

$$S1 - S2$$

Union of Relations

$S1$

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

$S1 \cup S2$

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

Intersection and Set-Difference

$S1$

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

$S1 \cap S2$

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

Cross-Product

- 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*.

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	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 operator*: $\rho\,(C(1 \to sid1, 5 \to sid2),\ S1 \times R1)$

Equi-Joins

- the condition contains only ***equalities:***

$$S1 \bowtie_{S1.sid=R1.sid} R1 \quad \text{stands for :} \quad \sigma_{sid1=sid2}(S1 \times R1)$$

$$S1 \bowtie R1 \quad \text{stands for :} \quad \pi_{sid}(\sigma_{S1.sid=R1.sid}(S1 \times R1))$$

and returns

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

- *Result schema* similar to cross-product, but only one copy of fields for which equality is specified.
- *Natural Join*: Equijoin on *all* common fields.

General Joins

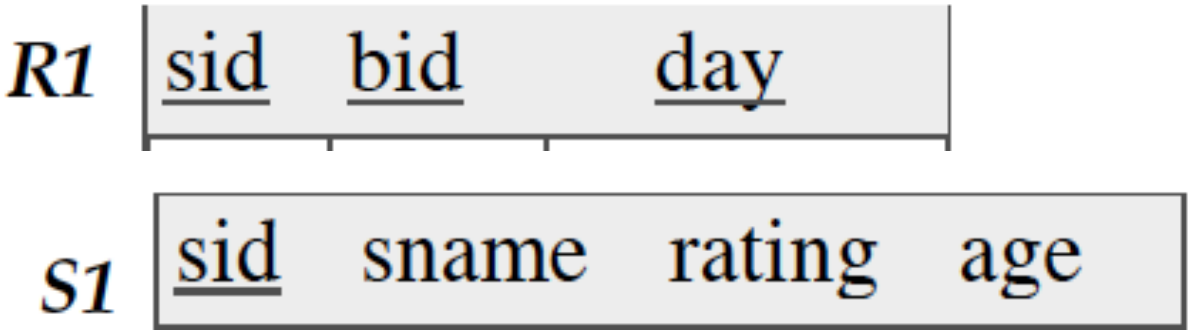
- *Condition Join*: $R \bowtie_c S = \sigma_c (R \times S)$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

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

- *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*.

Query: Find names of sailors who have reserved boat #103 and when?



- Solution 1: $\pi_{sname,day}((\sigma_{bid=103} R1) \bowtie S1)$

Student(sid, name, addr, age, GPA)

sid	name	addr	age	GPA
301	John	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

Class(dept, cnum, sec, unit, title, instructor)

dept	cnum	sec	unit	title	instructor
CS	112	01	03	Modeling	Dick Muntz
CS	143	01	04	DB Systems	Carlo Zaniolo
EE	143	01	03	Signal	Dick Muntz
ME	183	02	05	Mechanics	Susan Tracey

Enroll(sid, dept, cnum, sec)

sid	dept	cnum	sec
301	CS	112	01
301	CS	143	01
303	EE	143	01
303	CS	112	01
401	CS	112	01

Queries

- Q1: Find the titles and instructors of all CS courses
- Q2: All student names and GPAs who live on Wilshire
- Q3: Find all student names who take CS classes.
- Q4: All names of students and instructors
- Q5: Find the names of pairs of students who live on the same addr
- Q6: Find students taking at least two classes
- Q7: Find all student names who take **all** CS classes (offered this quarter)
- Q8: Find the names of students who is taking no CS classes (offered this quarter)

Student(sid, name, addr, age, GPA)

Class(dept, cnum, sec, unit, title, instructor)

Enroll(sid, dept, cnum, sec)

Q3: Find all names of students taking some (i.e., one or more) CS class.

$$\pi_{name}(\pi_{sid}(\sigma_{dept='cs'} Enroll)) \bowtie Student)$$

$$\pi_{name}((\sigma_{dept='cs'} Enroll) \bowtie Student)$$

Q5: Find the names of pairs of students who live on the same addr

Q6: Find students who do not live at "301 Wilshire"

Q8: Find the names of students who are taking no CS classes (offered this quarter)

*Q7: Find all students who take **all** CS classes (offered this quarter)*

Student(sid, name, addr, age, GPA)

sid	name	addr	age	GPA
301	John	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

Class(dept, cnum, sec, unit, title, instructor)

dept	cnum	sec	unit	title	instructor
CS	112	01	03	Modeling	Dick Muntz
CS	143	01	04	DB Systems	Carlo Zaniolo
EE	143	01	03	Signal	Dick Muntz
ME	183	02	05	Mechanics	Susan Tracey

Enroll(sid, dept, cnum, sec)

sid	dept	cnum	sec
301	CS	112	01
301	CS	143	01
303	EE	143	01
303	CS	112	01
401	CS	112	01

Q6: Find students who do not live at "301 Wilshire

Q7: Find the names of students who are taking no CS classes (offered this quarter)

Q8: Find all students who take ***all*** *S classes (offered this quarter)*

Division

Start with a table R where every students takes all CS courses

R:

sid	dept	cnum	sec
301	CS	112	01
301	CS	143	01
303	CS	112	01
303	CS	143	01
401	CS	112	01
401	CS	143	01
208	CS	112	01
208	CS	143	01

Enroll

sid	dept	cnum	sec
301	CS	112	01
301	CS	143	01
303	EE	143	01
303	CS	112	01
401	CS	112	01

R – Enroll

sid	dept	cnum	sec
303	CS	143	01
401	CS	143	01
208	CS	112	01
208	CS	143	01

Division

- π_{sid}(R – Enroll):

sid
303
401
208

- π_{sid}(Student):

sid
301
303
401
208

- π_{sid}(Student) - π_{sid}(R – Enroll):

sid
301