CSCC43H: Introduction to Databases

Lecture 3

Wael Aboulsaadat

Acknowledgment: these slides are partially based on Prof. Garcia-Molina & Prof. Ullman slides accompanying the course's textbook.

Database Management System (DBMS)

- A collection of programs that enable:
  - ➔Defining (describing the structure),
  - – Populating by data (Constructing),
  - – Manipulating (querying, updating),
  - – Preserving consistency,
  - – Protecting from misuse,
  - – Recovering from failure, and
  - – Concurrent using

  of a database.

Relational Query Languages

- *Query languages:* 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 data sets.

Formal Relational Query Languages

Two mathematical Query Languages form the basis for "real" languages (e.g. SQL), and for implementation:

Relational Algebra: More operational, very useful for representing execution plans.

Relational Calculus: Lets users describe what they want, rather than how to compute it. (Non-procedural, *declarative*.)

- ***Understanding Algebra & Calculus is key to understanding SQL, query processing!***

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 over any legal instance)
  - The schema for the *result* of a given query is also fixed. It is determined by the definitions of the query language constructs.
- Positional vs. named-field notation:
  - Positional notation easier for formal definitions, named-field notation more readable.
  - Both used in SQL
    - Though positional notation is not encouraged

Relational Algebra: 5 Basic Operations

- *Selection* (σ) Selects a subset of *rows* from relation (horizontal).
- *Projection* (π) Retains only wanted *columns* from relation (vertical).
- *Cross-product* ($\times$) Allows us to combine two relations.
- *Set-difference* (—) Tuples in r1, but not in r2.
- *Union* ($\cup$) Tuples in r1 or in r2.

Since each operation returns a relation, operations can be *composed!* (Algebra is “closed”.)

Example Instances

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Sailors1 (S1)

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Reserves1 (R1)

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

Sailors2 (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

Projection (π)

- Examples: $\pi_{age}(S2)$ $\pi_{sname,rating}(S2)$
- Retains only attributes that are in the "*projection list*".
- *Schema* of result:
  - exactly the fields in the projection list, with the same names that they had in the input relation.
- Projection operator has to *eliminate duplicates* (How do they arise? Why remove them?)
  - Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it. (Why not?)

Projection (π)

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

S2

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

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

age
35.0
55.5

$\pi_{age}(S2)$

Selection (σ)

- Selects rows that satisfy *selection condition*.
- Result is a relation.

  Schema of result is same as that of the input relation.
- Do we need to do duplicate elimination?

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	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 and Set-Difference

- Both of these operations take two input relations, which must be ***union-compatible***:
  - Same number of fields.
  - `Corresponding' fields have the same type.

Union

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

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

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

$S1 \cup S2$

Set Difference

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
22	dustin	7	45.0

S1 – 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

S2

sid	sname	rating	age
28	yuppy	9	35.0
44	guppy	5	35.0

S2 – S1

Cross-Product

- S1 × R1: Each row of S1 paired with each row of R1.
- Q: How many rows in the result?
- *Result schema* has one field per field of S1 and R1, with field names `inherited' if possible.
  - *May have a naming conflict*: Both S1 and R1 have a field with the same name.
  - In this case, can use the *renaming operator*:

field-count → newname

Cross Product Example

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

R1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

S1 X R1 =

(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

Compound Operator: Intersection

- In addition to the 5 basic operators, there are several additional "Compound Operators"
  - These add no computational power to the language, but are useful shorthands.
  - Can be expressed solely with the basic ops.

- Intersection takes two input relations, which must be *union-compatible*.

- Q: How to express it using basic operators?

$$R \cap S = R - (R - S)$$

Intersection

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

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

S2

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap S2$

Compound Operator: Join

- Joins are compound operators involving cross product, selection, and (sometimes) projection.

- Most common type of join is a "*natural join*" (often just called "join"). $R \bowtie S$ conceptually is:
  - Compute $R \times S$
  - Select rows where attributes that appear in both relations have equal values
  - Project all unique attributes and one copy of each of the common ones.

- Note: Usually done much more efficiently than this.

Natural Join Example

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

R1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

S1 $\bowtie$ R1 =

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

Other Types of Joins

- *Condition Join (or "theta-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.
- May have fewer tuples than cross-product.
- *Equi-Join*: Special case: condition *c* contains only conjunction of *equalities*.

Divide

- Database Data

REGISTRATION-HISTORY : Table

Sem	Dept	C_no	Sec	Student-id	Grade
2001-1	MGT	459	1	525-99-2121	B
2001-1	MGT	459	1	585-11-2222	A
2002-2	MGT	331	1	220-00-1111	C
2002-2	MGT	331	1	525-99-2121	C
2002-2	MGT	331	2	585-11-2222	B
2002-2	MGT	337	2	220-00-1111	A
2002-2	MGT	337	2	525-99-2121	B

PREREQUISITE : Table

Crs_Dept	Crs_Crs_no	Prereq_Dept	Prereq_C
MGT	460	MGT	331
MGT	460	MGT	337
MGT	460	MGT	459

- Projected Tables

COMPLETED-CLASSES : Table

Student-id	Dept	Crs_no
220-00-1111	MGT	337
220-00-1111	MGT	331
525-99-2121	MGT	331
525-99-2121	MGT	337
525-99-2121	MGT	459
585-11-2222	MGT	459
585-11-2222	MGT	331

MGT460-PREREQS : Table

Prereq_Dept	Prereq_Crs_no
MGT	331
MGT	337
MGT	459

- COMPLETED-CLASSES DIVIDEBY MGT460-PREREQS

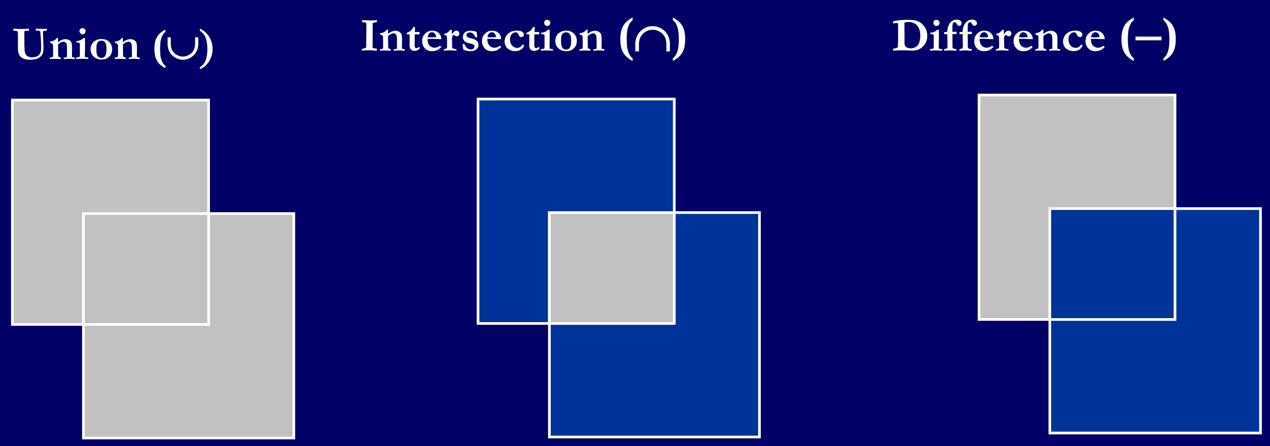
RA-DIVIDE : Select Query

Student-id
525-99-2121

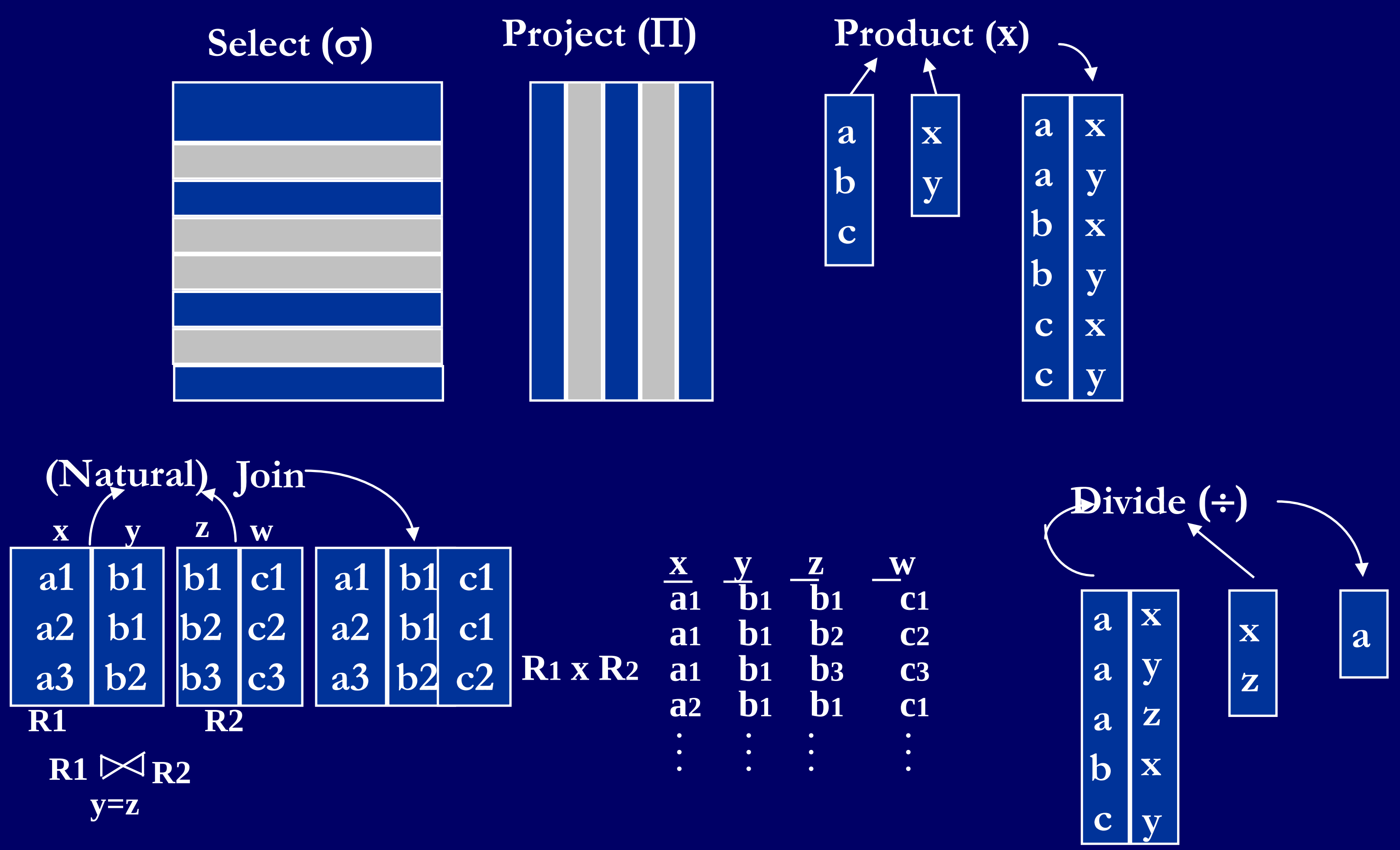
Summary

- Relational Algebra: a small set of operators mapping relations to relations
  - Operational, in the sense that you specify the explicit order of operations
  - A *closed* set of operators! Can mix and match.
- Basic ops include: σ, π, $\times$, $\cup$, —
- Important compound ops: $\cap$, $\bowtie$

Relational Operators

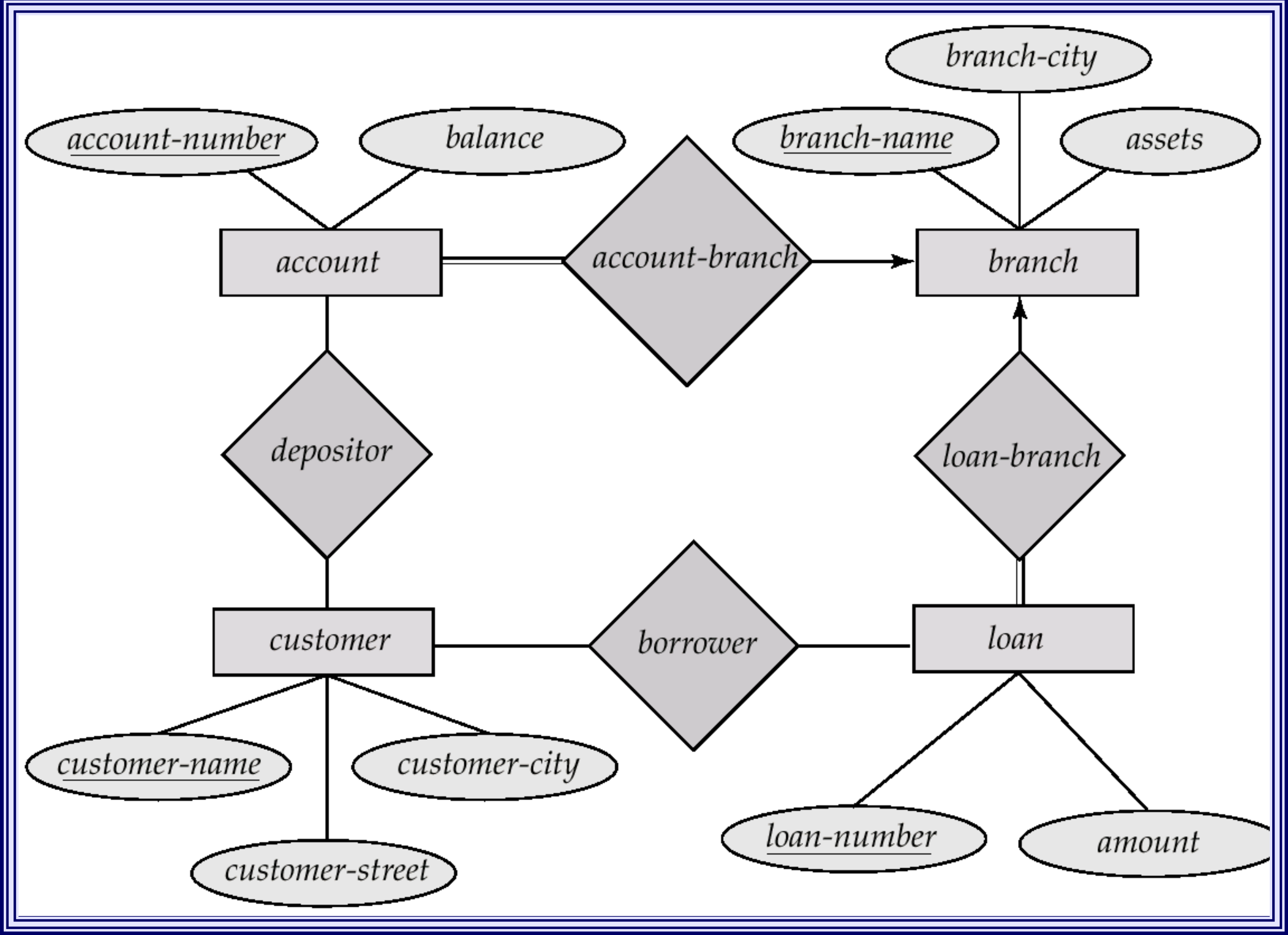


Relational Operators (cont.)



Banking Example

E-R Diagram for the Banking Enterprise



Banking Example

branch (branch_name, branch_city, assets)

customer (customer_name, customer_street, customer_city)

account (account_number, branch_name, balance)

loan (loan_number, branch_name, amount)

depositor (customer_name, account_number)

borrower (customer_name, loan_number)

Example Queries

- Find all loans of over $1200

$$\sigma_{amount > 1200}\ (loan)$$

- Find the loan number for each loan of an amount greater than $1200

$$\Pi_{loan\_number}\ (\sigma_{amount > 1200}\ (loan))$$

- Find the names of all customers who have a loan, an account, or both, from the bank

$$\Pi_{customer\_name}\ (borrower) \cup \Pi_{customer\_name}\ (depositor)$$