Fall 2013

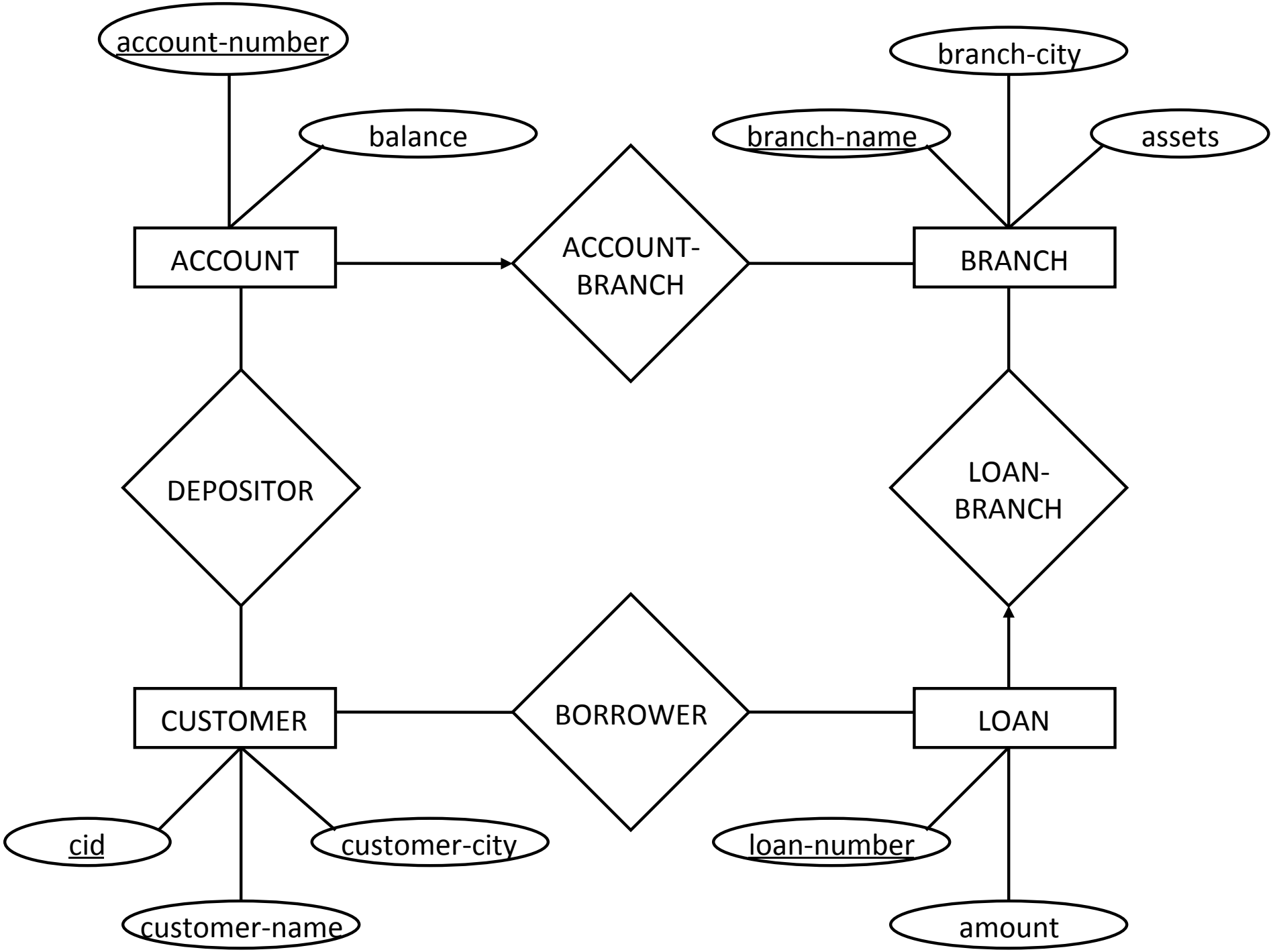
RELATIONAL ALGEBRA

[CH 4: SECTIONS 4.1 AND 4.2]

Relational Query Languages

- Allow manipulation and retrieval of data from a database
- Two "pure" types
  - Declarative: Tuple Relational Calculus (TRC), Domain Relational Calculus (DRC)
    - Describe what a user wants, rather than how to compute it.
  - Procedural : Relational Algebra (RA)
    - Operational, very useful for representing execution plans.
- Commercial Relational QL (SQL): borrow from both.
- 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.

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


account-number
balance
branch-city
branch-name
assets
ACCOUNT
ACCOUNT-
BRANCH
BRANCH
DEPOSITOR
LOAN-
BRANCH
CUSTOMER
BORROWER
LOAN
cid
customer-city
loan-number
customer-name
amount

The *account* relation (A)

bname	acct#	balance
Downtown	A-101	500
Mainus	A-215	700
Perryridge	A-102	400
Round Hill	A-305	350
Brighton	A-201	900
Redwood	A-222	700
Brighton	A-217	750

The *branch* relation (B)

bname	bcity	assets
Downtown	Brooklyn	9000000
Redwood	Palo Alto	2100000
Perryridge	Horseneck	1700000
Mianus	Horseneck	400000
Round Hill	Horseneck	8000000
Pownal	Bennington	300000
North Town	Rye	3700000
Brighton	Brooklyn	7100000

The *customer* relation (C)

cid	cname	ccity
001	Jones	Harrison
002	Smith	Rye
003	Hayes	Harrison
004	Curry	Rye
005	Lindsay	Pittsfield
006	Turner	Stamford
007	Williams	Princeton
008	Adams	Pittsfield
009	Jones	Palo Alto
010	Green	Woodside
011	Brooks	Brooklyn
012	Green	Stamford

The *loan* relation (L)

bname	loan#	amount
Downtown	L-17	1000
Redwood	L-23	2000
Perryridge	L-15	1500
Downtown	L-14	1500
Redwood	L-93	500
Round Hill	L-11	900
Perryridge	L-16	1300

The *depositor* relation (D)

cid	acct#
009	A-101
002	A-215
003	A-102
006	A-305
009	A-201
001	A-217
005	A-222

The *borrower* relation (Bw)

cid	loan#
001	L-17
002	L-23
003	L-15
009	L-14
004	L-93
002	L-11
007	L-17
008	L-16

Relational Algebra Preliminaries

- Query:
  - Input: Relational instances
  - Output: Relational instances!
  - Specified using the schemas.
    - May produce different results for different instances.
    - But schema of the result is fixed. Determined by QL constructs.
- Positional vs. named-field notation:
  - Positional notation easier for formal definitions, named-field notation more readable.
  - Both used in SQL
    - C.cname or
    - 2 (note this only works in the ORDER BY clause)

Relational Algebra

- Basic operations:
  - *Selection* (σ) Selects a subset of rows from relation.
  - *Projection* (π) Deletes unwanted columns from relation.
  - *Cross-product* ($\times$) 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.
- Operations can be composed, i.e. algebra is closed.

Projection

- Deletes attributes that are not in *projection list*.
- *Schema* of result?
- Duplicates?

$\pi_{bname,balance}(A)$

bname	balance
Downtown	500
Mainus	700
Perryridge	400
Round Hill	350
Brighton	900
Redwood	700
Brighton	750

$\pi_{bname}(B)$

OR $\pi_{1}(B)$

bname
Downtown
Mainus
Perryridge
Round Hill
Brighton
Redwood

$\pi_{bcity}(B)$

bcity
Palo Alto
Horseneck
Bennington
Rye
Brooklyn

Selection

- Selects rows that satisfy *selection condition*.
- Duplicates?
- *Schema* of result?

$$\sigma_{amount>1400}(L)$$

bname	loan#	amount
Redwood	L-23	2000
Perryridge	L-15	1500
Downtown	L-14	1500

$$\pi_{bname}(\sigma_{amount>1400}(L))$$

bname
Redwood
Perryridge
Downtown

Set Operations: Union (∪), Intersection(∩), Set-Difference (-)

- Input: two *union-compatible* relations:
  - Same number/types of fields. And in same order.
- Field names of result?
- Schema of result?

All customers with no loan

cid
001
002
003
004
005
006
007
008
009
010
011
012

-

cid
001
002
003
009
004
002
007
008

=

cid
005
006
010
011
012

$$\pi_{cid}(C) - \pi_{cid}(Bw)$$

Cross-Product

- Cartesian Product
- Result Schema
  - One field from both relations, names inherited if possible
- Both *L* and *A* have a field called bname.
  - These fields are un-named, use positions
  - Rename operator **ρ (C(1 → b1, 4 → b2)), L X A)**

b1	loan#	amount	b2	acct#	balance
Downtown	L-17	1000	Downtown	A-101	500
Downtown	L-17	1000	Mainus	A-215	700
Downtown	L-17	1000	Perryridge	A-102	400
Downtown	L-17	1000	Round Hill	A-305	350
⋮					

Joins

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

b1	loan#	amount	b2	acct#	balance
Redwood	L-93	500	Redwood	A-215	700
Redwood	L-93	500	Redwood	A-201	900
Redwood	L-93	500	Round Hill	A-222	700
Redwood	L-93	500	Perryridge	A-217	750

$$L \bowtie_{L.amount < A.balance} A$$

Rewrite using X and ☞ ?

$$\sigma_{L.amount < A.balance}(L \times A)$$

- Result Schema?
- Fewer tuples than cross-product, might be able to compute more efficiently
- Sometimes called a *theta-join*.

Joins

- *Equi-Join*: Special case of condition join where the condition *c* contains only ***equalities***

$$D \bowtie_{cid} Bw$$

cid	acct#	loan#
009	A-101	L-14
002	A-215	L-23
003	A-102	L-15
009	A-201	L-14
001	A-217	L-17

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

Division

- Not a primitive operator, but useful for queries like:

  *Find customers with accounts in **all** the branches in Brooklyn.*

- Let *A* have 2 fields, *x* and *y* ; *B* have only field *y* :
  - $A/B = \{x \mid \forall \ \langle y \rangle \ \in B \ [\ \exists \ \langle x, y \rangle \ \in A \ ]\}$
  - ***A/B* contains all *x* tuples (customers) such that for *every* *y* tuple (branches in Brooklyn) in *B*, there is an <*x,y*> tuple in *A***
- In general, x and y can be any lists of fields; y is the list of fields in B, and x $\bigcup$ y is the list of fields of A.

Examples of Division A/B

cid	bname
c1	b1
c1	b2
c1	b3
c1	b4
c2	b1
c2	b2
c3	b2
c4	b2
c4	b4

A

bname
b2

B1

bname
b2
b4

B2

bname
b1
b2
b4

B3

cid
c1
c2
c3
c4

A/B1

A/B2

A/B3

Expressing A/B Using Basic Operators

- Division is a useful shorthand, not essential op
  - Also true of joins
- Idea: For A/B, compute all x values that are not disqualified by some y value in B.
  - x value is disqualified if by attaching y value from B, we obtain an <x,y> tuple that is not in A

Can you express this operator using basic operators?

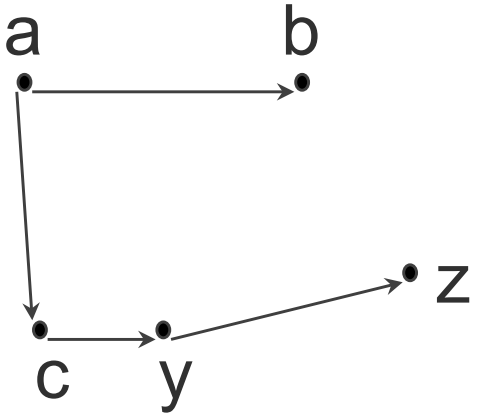
A/B: $\pi_x(A) -$ Disqualified x values

Expressive Power

- Codd's Theorem: Every RA query can be expressed as a safe query in TRC/DRC; the converse is also true.
- *Relational Completeness*: A "relationally complete" query language (e.g., SQL) can express every query that is expressible in relational algebra/calculus.
- Limitation of RA:
  - For any particular instance of Edges, there is an R.A. expression to compute transitive closure. (What is it?)
  - There's no R.A. for transitive closure of an arbitrary instance of Edges. (Why?)

Edges

From	*To*
a	b
a	c
c	y
y	z



Find names of sailors who've reserved boat #103

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

Solution 1:

Solution 2:

Solution 3:

Find names of sailors who've reserved a red boat

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

- Join relations?
  - Sailor, Reserves, Boats (for color)

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

A more efficient solution:

A query optimizer can find the most efficient solution!

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

- 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 and a green boat

1. Identify
   - sailors who've reserved red boats
   - sailors who've reserved green boats
2. Then find the intersection (*sid* is a key for Sailors):

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

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

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

Find the names of sailors who've reserved all boats

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

- Uses division; schemas of the input relations to / must be carefully chosen:

$$\rho(Tempsids, (\pi_{sid,bid}\text{Reserves}) / (\pi_{bid} Boats))$$

$$\pi_{sname}(Tempsids \bowtie Sailors)$$

- To find sailors who've reserved all '470' boats: