Database Management Systems

Dr. Sadhana Jha
Computer Science and Information Systems Department
BITS, Pilani

BITS Pilani
Pilani Campus

innovate achieve lead

BITS Pilani
Pilani Campus

Operations on Relations: Relational Algebra

Query Language

- Language in which the user requests information from the database.
- Procedural or non-procedural
- Instructs the system to perform a sequence of operations on database to compute desired result. E.g. Relational algebra
- User describes the desired information without giving specific procedure for obtaining the information. E.g. tuple relational calculus and domain relational calculus.

Relational Query Languages

- Procedural languages:
  - Relational algebra (RA)
  - Tuple relational calculus (TRC)
  - Domain relational calculus (DRC)
- The above 3 languages are equivalent in computing power
- Procedural relational query language defines a set of operations that operate on tables and outputs table
- Provides a set of operations that take one or more relations as input and return a relation as output.
  - SQL is based on relational algebra. Adds more to it.
  - consists of 6 basic operations

Basic Operations in Relational Algebra

- select
- project
- set-union
- set-difference
- cartesian product
- rename

Select Operation

- Notation: $\sigma_p(r)$, where *p* is called the selection predicate and *r* is the relation name
- Defined as:
  - $\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$
  - where p is a formula in propositional calculus consisting of terms connected by : ∧ (and), ∨ (or), ¬ (not)
  - Each term is one of: <attribute_name> op <attribute_value>/<attribute_name>/constant, where op is one of: =, ≠, >, ≥, <, ≤

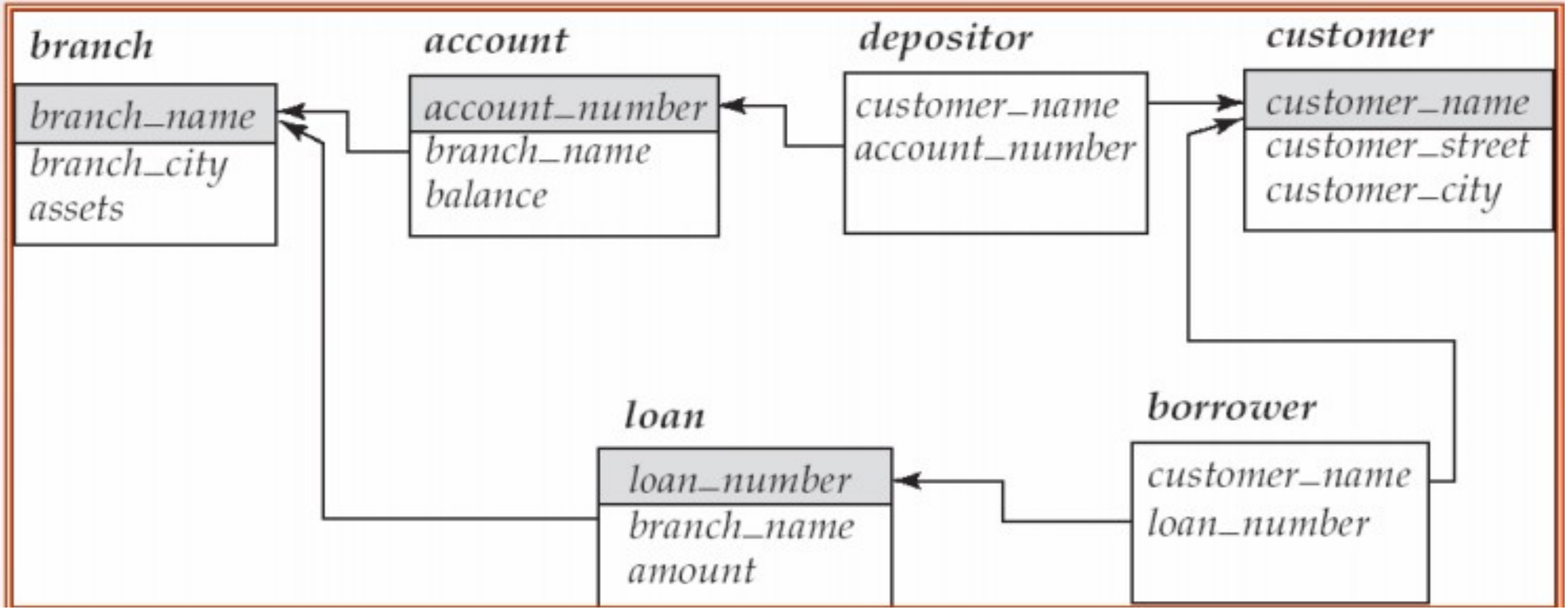
Example of Select Operation

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10



Q: Select the details of all the accounts opened in "Pilani" branch of bank B.

$$\sigma_{\text{branch\_name = "Pilani"}}(\text{account})$$

Q: Select the details of all the loans approved in "Pilani" branch and the amount is greater then 50lacs.

$$\sigma_{\text{branch\_name = "Pilani"} \wedge \text{amount} > 5000000}(\text{account})$$

Project Operation

- Notation: $\Pi_{A1, A2, \ldots, An}(r)$, where $A1, A2, \ldots, An$ are attribute names and *r* is a relation name
- Duplicate rows removed from result, since relations are sets
- Used for eliminating extra information

Example of Projection Operation

- Relation *r*:

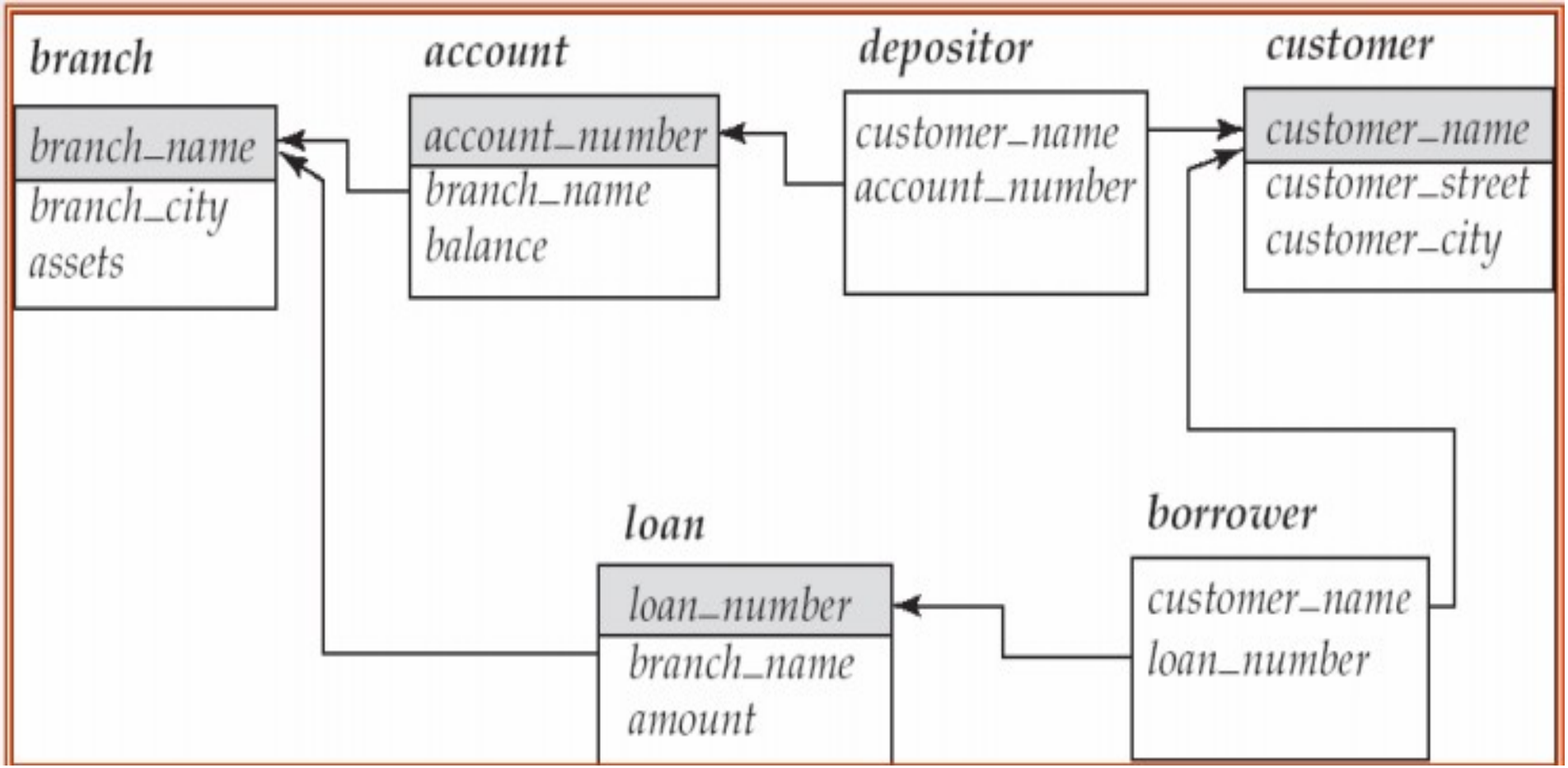
A	*B*	*C*
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}(r)$

A	*C*
α	1
α	1
β	1
β	2

=

A	*C*
α	1
β	1
β	2



Q: View account_number and balance of all the customers of the bank.

$$\Pi_{\text{account\_number, balance}}\ (\text{account})$$

Union Operation

- Notation: $r \cup s$
- Defined as: $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$
- For $r \cup s$ to be valid, *r* and *s* needs to be union compatible, i.e.:
  - r, s must have the same arity (same number of attributes)
  - The attribute domains must be compatible (example: 2nd column of r deals with the same type of values as does the 2nd column of s)

Example of Union Operation

- Relations *r, s:*

A	*B*
α	1
α	2
β	1

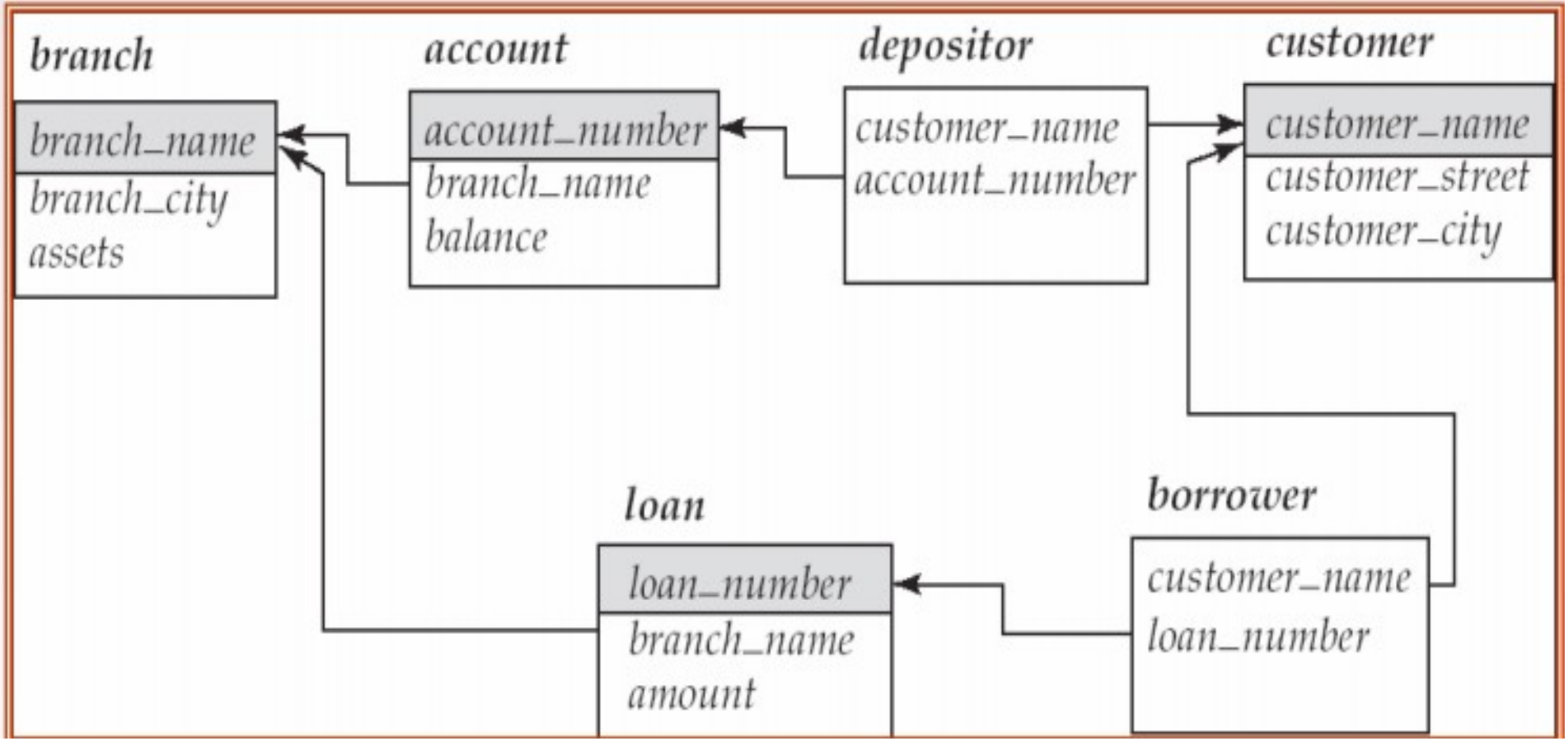
r

A	*B*
α	2
β	3

s

- **r ∪ s:**

A	*B*
α	1
α	2
β	1
β	3



Q: Find all customers with either an account or a loan

A: $\Pi_{customer\_name}$ (depositor) ∪ $\Pi_{customer\_name}$ (borrower)

Set Difference Operation

- Notation: **r – s**
- Defined as: **r – s = {t | t ∈ r and t ∉ s}**
- For **r – s** to be valid, *r* and *s* needs to be union compatible.

A	B
α	1
α	2
β	1

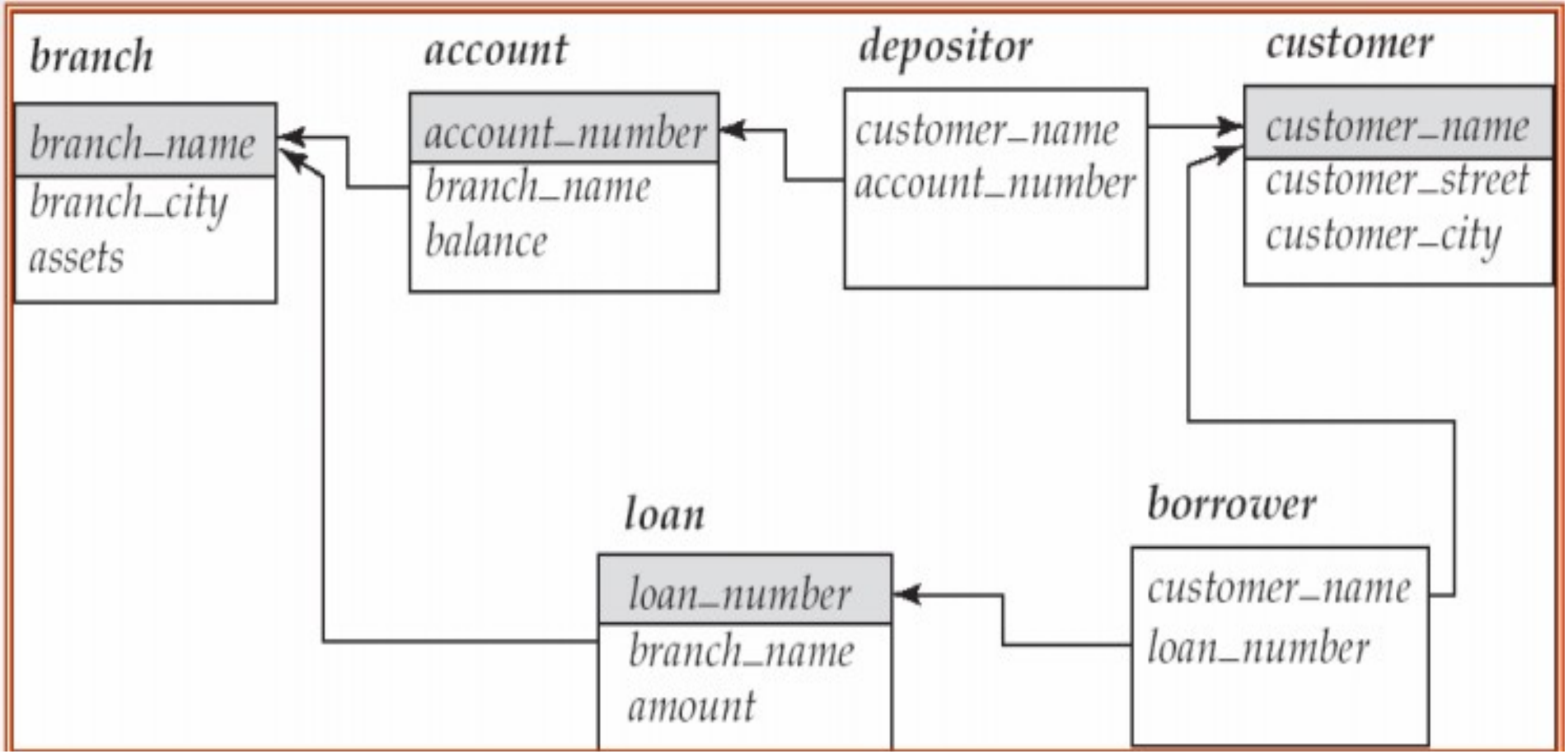
r

A	B
α	2
β	3

s

A	B
α	1
β	1

r-s



Q: Find the branches of the bank B which have not sanctioned any loan yet

$$\Pi_{branch\_name}(branch) - \Pi_{branch\_name}(loan)$$

Cartesian-product

- Notation: **r x s**
- Defined as: **r x s = {t q | t ∈ r and q ∈ s}**
- Assumes that attributes of r(R) and s(S) are disjoint. (That is, R ∩ S = ∅)
- If attributes of r(R) and s(S) are not disjoint, then renaming is done

Cartesian-product

- **Relations *r*, *s*:**

A	*B*
α	1
β	2

r

C	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

- ***r* x *s*:**

A	*B*	*C*	*D*	*E*
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Cartesian-product – naming issue

- Relations *r*, *s*:

A	*B*
α	1
β	2

r

B	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	*r.B*	*s.B*	*D*	*E*
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Composition of Operations

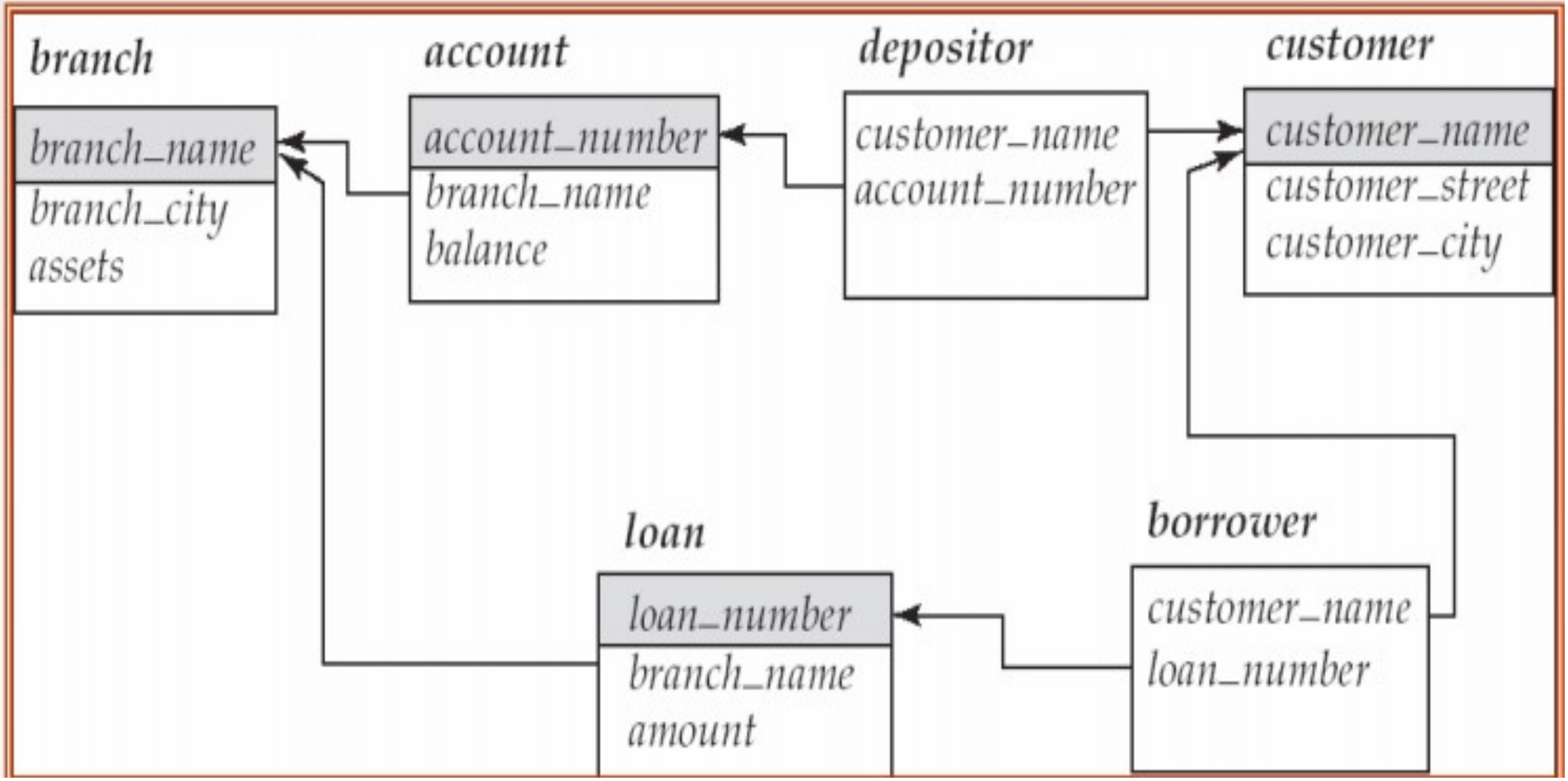
- Building expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$

r X s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

$\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b



Q: Find all the account number associated with the branch "Pilani" of bank B

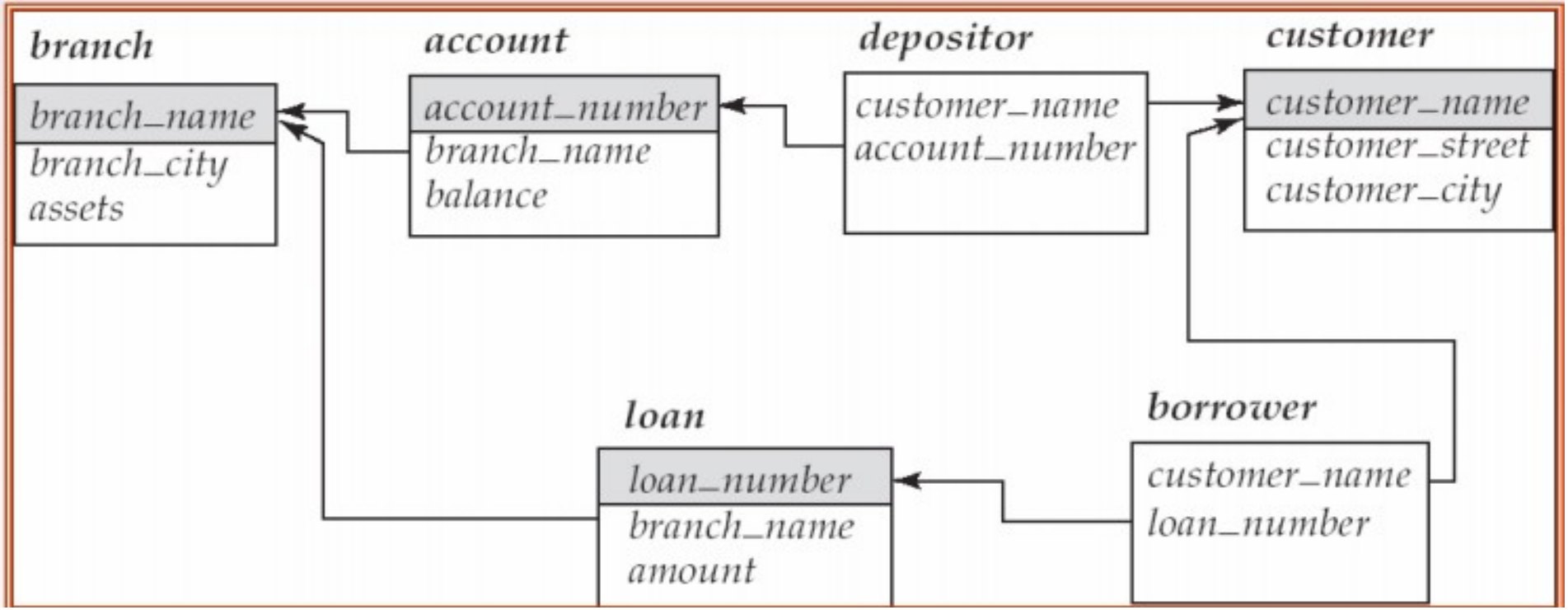
$$\Pi_{\text{account\_number}}(\sigma_{\text{branch\_name = "Pilani"}}(\text{account}))$$

Rename Operation

- Allows to name, and therefore to refer to, the results of relational algebra expressions
- Allows to refer to a relation by more than one name
- Example: $\rho_x(E)$ returns the expression E under the name X
- If a relational-algebra expression E has arity n, then

$$\rho_{x(A1 , A2 , \ldots, An)}(E)$$

returns the result of expression E under the name X, and with the attributes renamed to A_1 , A_2 , …., A_n .



Q: Find all the account number associated with the branch "Pilani" of bank B. Rename the account_number column of the result to AN.

$\rho_{x(AN)}(\Pi_{account\_number}(\sigma_{branch\_name = \text{“Pilani”}}(account))$

Formal Definition

- A basic expression in RA consists of either one of the following:
  - A relation in the database
  - A constant relation
- If E1 and E2 are two RA expressions, then the following are all relational-algebra expressions:
  - **E1 ∪ E2**, **E1 – E2**, **E1 X E2**,
  - $\sigma_p(E1)$, P is a predicate on attributes in E1,
  - $\Pi_s(E1)$, S is a list consisting of some of the attributes in E1,
  - $\rho_x(E1)$, x is the new name for the result of E1

Additional RA Operators

They do not add power to the RA, but are used to simplify RA queries:

- Set intersection
- Natural join
- Division
- Assignment

Set Intersection Operation

- Notation: **r ∩ s**
- Defined as: **r ∩ s = {t | t ∈ r and t ∈ s}**
- **r ∩ s = r – (r – s)**
- For r ∩ s to be valid:
  - r, s must have the same arity (same number of attributes)
  - The attribute domains must be compatible (example: 2nd column of r deals with the same type of values as does the 2nd column of s)

Set Intersection Operation

- **r**

A	B
α	1
α	2
β	1

- **s**

A	B
α	2
β	3

r ∩ s

A	B
α	2

Natural Join

- Notation: ⋈ s
- Let r and s be relations on schemas R and S respectively. Then ⋈ r s is a relation on schema R ∪ S obtained as follows:
  - Consider each pair of tuples t_r from r and t_s from s
  - If t_r and t_s have the same value on each of the attributes in R ∩ S, add a tuple t to the result, where
    - t has the same value as t_r on r
    - t has the same value as t_s on s

Natural Join

- Example:
- R = (A, B, C, D)
- S = (C, D, E)
- Resulting Schema = (A, B, C, D, E)
- $r \bowtie s = \prod_{R \cup S} (\sigma_{r.C = s.C \wedge r.D = s.D} (r \times s))$

- **Natural join:** Projection on **R ∪ S** and selection on values of **R ∩ S**

Natural Join Example

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

r ⋈ s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

$\Pi_{A, r.B, C, r.D, E}(\sigma_{r.B = s.B \wedge r.D = s.D}(r \times s))$

Division

- Notation: $r \div s$
- Suited to queries that include the phrase "for all"
- Let r and s be relations on schemas R and S respectively where $R \subseteq S$ and
  - **$R = (A1, \ldots, Am , B1, \ldots, Bn)$**
  - **$S = (B1, \ldots, Bn)$**
  - The result of $r \div s$ is a relation on schema **$R - S = (A_1, \ldots, A_m)$**
  - **$r \div s$** $= \{ t \mid t \in \prod_{R-S} (r) \wedge \forall u \in s (tu \in r) \}$

    Where tu means the concatenation of tuples t and u to produce a single tuple

Division Example

A	B	C
α	1	X
β	2	Y
π	3	Z
β	1	W
α	2	Y
β	2	Y
π	1	X

r

÷

B	C
1	X
2	Y

S

=

A
α

Assignment Operator

- Notation: ←
- Provides a convenient way to express complex queries:
  - Write query as a sequential program consisting of
    - a series of assignments
    - followed by an expression whose value is displayed as a result of the query.
  - Assignment must always be made to a temporary relation variable

Assignment Operator

- Find the names of all customers who have a loan at the **“Perryridge”** branch but **do not** have an account at any branch of the bank

A. $\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Perryridge”}}(\sigma_{borrower.loan\_number = loan.loan\_number}(\textbf{borrower x loan}))) - \Pi_{customer\_name}(\textbf{depositor})$

Assignment Operator

temp1 ← $\sigma_{\text{borrower.loan\_number} = \text{loan.loan\_number}}$(**borrower x loan**)

temp2 ← $\sigma_{\text{branch\_name} = \text{“Perryridge”}}$ (**temp1**)

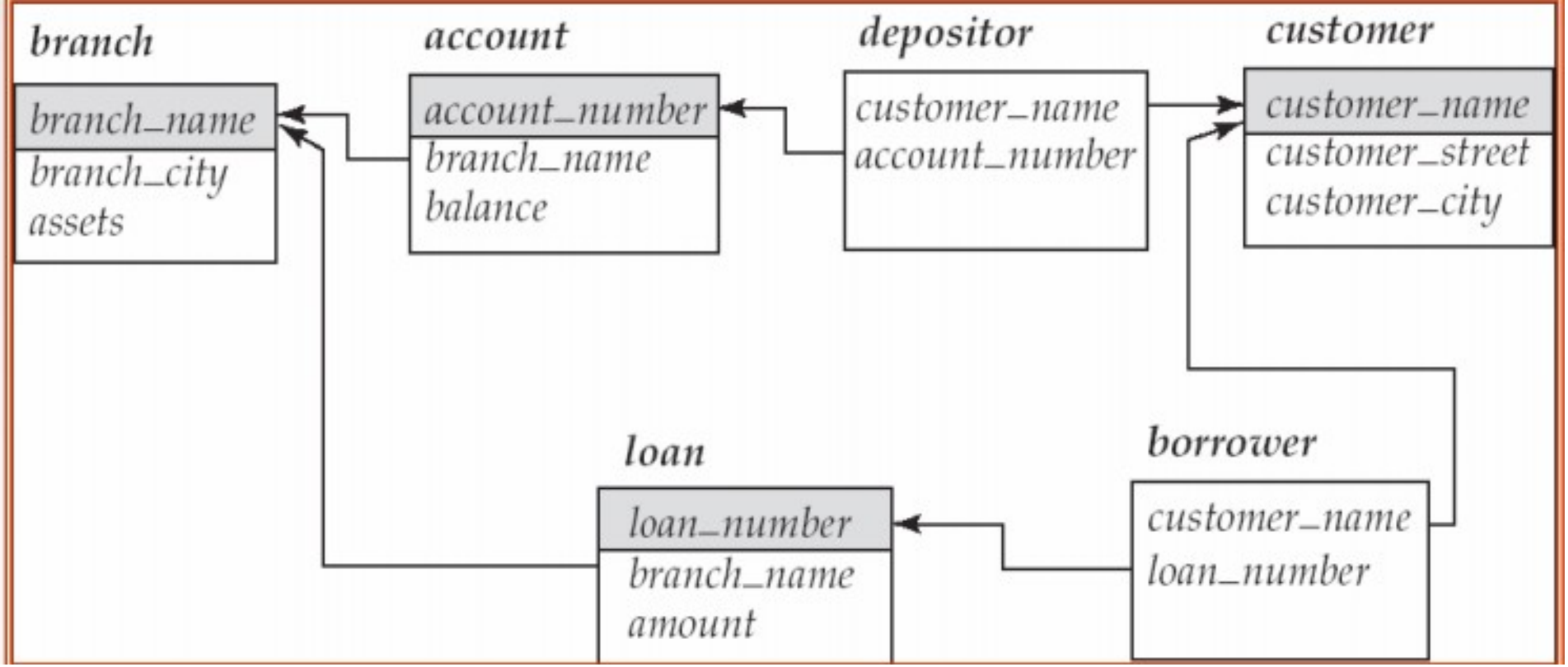
temp3 ← $\Pi_{\text{customer\_name}}$ (**temp2**)

temp4 ← $\Pi_{\text{customer\_name}}$(**depositor**)

result ← **temp3** – **temp4**

- The result to the right of the ← is assigned to the relation variable on the left of the ←
  - May use variable in subsequent expressions

Example Queries



Q1: Find the names of the customers who have a loan at the "Perryridge" branch

A: $\Pi_{\text{customer\_name}}$ ($\sigma_{\text{branch\_name}=\text{"Perryridge"}}$ ($\sigma_{\text{borrower.loan\_number} = \text{loan.loan\_number}}$(**borrower x loan**)))

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Borrower

loan_number	branch_name	amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

loan

CN	B.LN	L.LN	BN	Am
Adams	L-16	L-11	RoundHill	900
Adams	L-16	L-14	Downtown	1500
Adams.	L-16	L-15	Perryridge	1500
..	..	..	.	.
..	.	.	.	.
Adams	L-16	L-93	Mianus	500
Curry	L-93	L-11	RoundHill	900
Curry	L-93	L-14	Downtown	1500
Curry	L-93	L-15	Perryridge	1500
..	..	..	.	.
..	.	.	.	.
Curry	L-93	L-93	Mianus	500
..	L-16	L-14	Downtown	1500
..	L-16	L-15	Perryridge	1500
..	..	..	.	.
..	.	.	.	.
Williams	L-17	L-11	RoundHill	900
Williams	L-17	L-14	Downtown	1500
..	..	..	..	
.	..	..	..	
Williams	L-17	L-93	Mianus	500

Borrower X loan

(56 records)

Example Query-1

$\sigma_{borrower.loan\_number = loan.loan\_number}$ (borrower x loan)))

CN	*B.LN*	*L.LN*	*BN*	*Am*
Adams	L-16	L-16	Perryridge	1300
Curry	L-93	L-93	Mianus	500
Hayes	L-15	L-15	Perryridge	1500
Jackson	L-14	L-14	Downtown	1500
Jones	L-17	L-17	Downtown	1000
Smith	L-11	L-11	Roundhill	900
Smith	L-23	L-23	Redwood	2000
Williams	L-17	L-17	Downtown	1000

$\Pi_{customer\_name}$ ($\sigma_{branch\_name=\text{“Perryridge”}}$)

Customer Name
Adams
Hayes

Example Query-1

The other possible relational algebra query to **Find the names of the customers who have a loan at the "Perryridge" branch** can be:

$$\Pi_{\text{customer\_name}}(\sigma_{\text{borrower.loan\_number} = \text{loan.loan\_number}}(\text{borrower} \times \sigma_{\text{branch\_name}=\text{"Perryridge"}}(\text{loan})))$$

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Borrower

loan_number	branch_name	amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

loan

CN	B.LN	L.LN	BN	Am
Adams	L-16	L-15	Perryridge	1500
Adams	L-16	L-16	Perryridge	1300
Curry	L-93	L-15	Perryridge	1500
Curry	L-93	L-16	Perryridge	1300
Hayes	L-15	L-15	Perryridge	1500
Hayes	L-15	L-16	Perryridge	1300
Jackson	L-14	L-15	Perryridge	1500
Jackson	L-14	L-16	Perryridge	1300
Jones	L-17	L-15	Perryridge	1500
Jones	L-17	L-16	Perryridge	1300
Smith	L-11	L-15	Perryridge	1500
Smith	L-11	L-16	Perryridge	1300
Smith	L-23	L-15	Perryridge	1500
Smith	L-23	L-16	Perryridge	1300
Williams	L-17	L-15	Perryridge	1500
Williams	L-17	L-16	Perryridge	1300

(borrower x $\sigma_{branch\_name="Perryridge"}$ loan)

Example Query-1

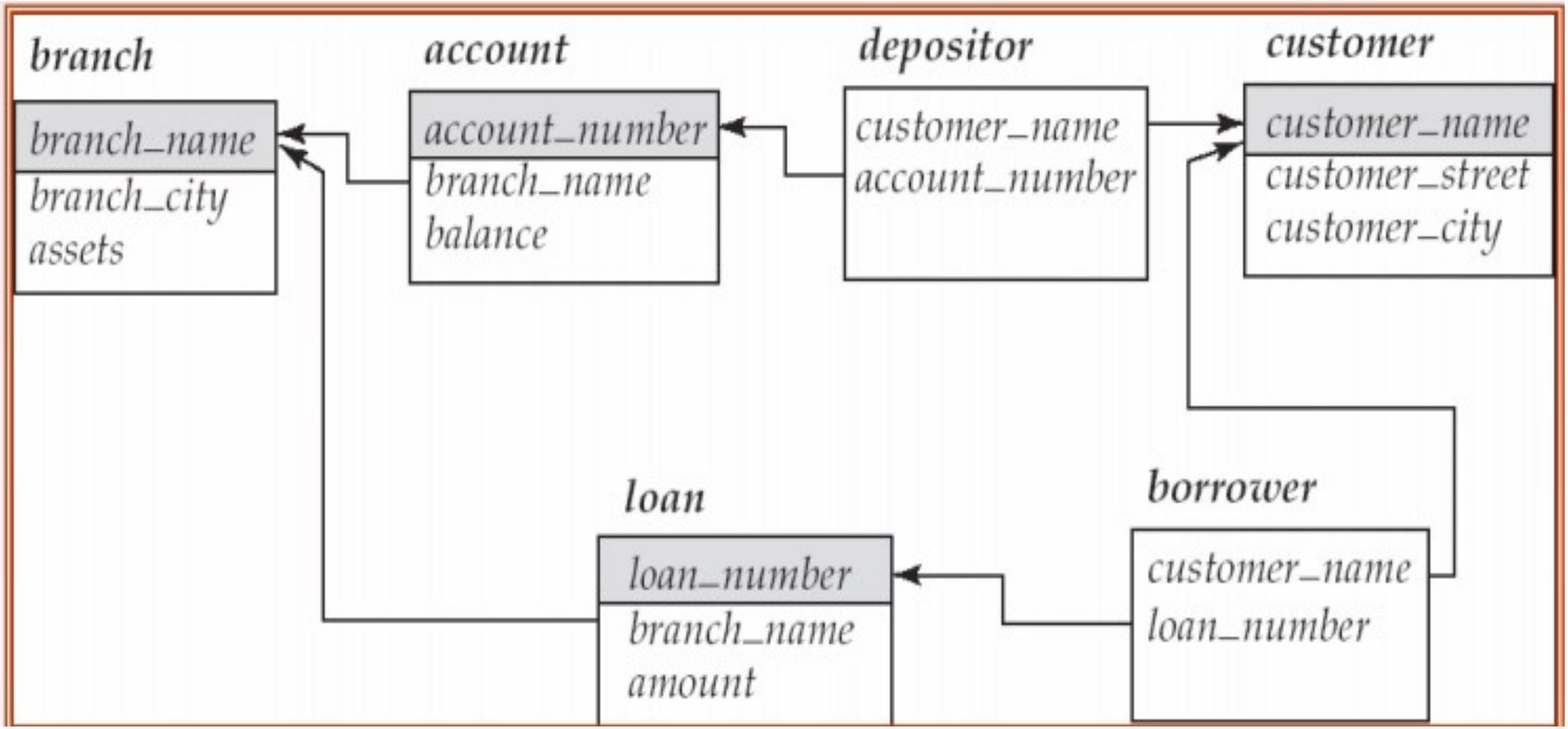
$\sigma_{borrower.loan\_number = loan.loan\_number}$ (borrower x $\sigma_{branch\_name=\text{“Perryridge”}}$ loan)

CN	*B.LN*	*L.LN*	*BN*	*Am*
Adams	L-16	L-16	Perryridge	1300
Hayes	L-15	L-15	Perryridge	1500

$\Pi_{customer\_name}(\sigma_{borrower.loan\_number = loan.loan\_number}$ (borrower x $\sigma_{branch\_name=\text{“Perryridge”}}$ loan))

Customer Name
Adams
Hayes

Example Query-2



Q: Find the names of all customers who have a loan but **do not** have an account at any branch of the bank

Example Query2

customer_name	*loan_number*
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Borrower

customer_name	*account_number*
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

Depositor

loan_number	*branch_name*	*amount*
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

Loan

Example Query-2

Customer Name
Adams
Curry
Hayes
Jackson
Jones
Smith
Williams

temp1

—

Customer Name
Hayes
Johnson
Jones
Lindsay
Smith
Turner

temp2

=

Customer Name
Adams
Curry
Jackson

result

temp1 ← $\Pi_{customer\_name}$ (**borrower** ⋈ **loan**)

temp2 ← $\Pi_{customer\_name}$(**depositor**)

result ← temp1 – temp2

Example Query-3

Q: Find the largest account balance

- Strategy:
  - Find those balances that are not the largest
  - Rename account relation as *d* so that we can compare each account balance with all others
  - Find account balances that were not found in the earlier step by set difference operation

A: $\Pi_{balance}(account) - \Pi_{account.balance}(\sigma_{account.balance < d.balance}(account \times \rho_d(account)))$

Example Query-4

Q: Find the names of the customers who have account at all branches located in Brooklyn city.

- Strategy:
  - Find all the branches of the bank located in the Brooklyn city
  - Find customer name along with all the branches in which they have account
  - Perform the division operation

customer_name	account_number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

depositor

account_number	branch_name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

account

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

branch

Example Query - 4

A: $\Pi_{customer\_name,\ branch\_name}$ (deposit ⋈ account) ÷ ρ_{branch_name} ($\sigma_{branch\_city = \text{“Brooklyn”}}$ (branch))

Customer_name	branch_name
Hayes	Perry Ridge
Johnson	Downtown
Johnson	Brighton
Jones	Brighton
Lindsay	Redwood
Smith	Mianus
Turner	Round Hill

÷

branch_name
Brighton
Downtown

Π_{branch_name} ($\sigma_{branch\_city = \text{“Brooklyn”}}$ (branch))

$\Pi_{customer\_name,\ branch\_name}$ (deposit ⋈ account)

Practice Problems

- customer (cid, name)
- transaction (tid, cid, pid)
- product (pid, pname)

Q: Write an RA query to find the name of all customers who have bought both bread and butter in the same transaction

Practice problems

Which information does the following RA expression extract from the following database schema:

loan(loan_number, branch_name, amount)

deposit(customer_name, account_number)

account(account_number, branch_name, balance)

Q: $\Pi_{\text{deposit.balance}}(\sigma_{\text{deposit.balance} < \text{ d.balance}}$ (deposit X ρ_{d} (account)))

Extended Relational Algebra Operator

- **Generalized projection**
- **Aggregate functions**
- **Outer Join**

Generalized projection

- **Extends the projection operation by allowing arithmetic functions to be used in the projection list**

- **Example:**

credit_info(customer_name, limit, credit_balance) Pi (C_Name, Limit-Credit_Balance) credit_info

Q: Find how much more each person can

Aggregate Functions

Aggregation function: Takes a collection of values and returns a single value as a result.

- **avg:** average value
- **min:** minimum value
- **max:** maximum value
- **sum:** sum of values
- **count:** number of values

Aggregate Functions

- **Aggregate operation**: In relational algebra E

$$_{G1, G2 \ldots, G}g_{F1(A1),F2(A2),\ldots.Fn(An)}(E)$$

- E is any relational-algebra expression
  - G1, G2 ..., Gn is a list of attributes on which to group (can be empty)
  - Each Fi is an aggregate function
  - Each Ai is an attribute name

Aggregate Functions

account_number	*branch_name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

g sum(balance) (account)

Sum(balance)
4300

account

Aggregate Functions

account_number	*branch_name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

account

branch_name **g** sum(balance)

(account)

branch_name	**Sum(balance)**
Downtown	500
Perryridge	400
Brighton	1650
Mianus	700
Redwood	700
Round Hill	350

Aggregate Functions

- Result of aggregation does not have a name
  - Can use rename operation to give it a name

$$\rho_{\text{branch\_name}}\, g_{\text{sum(balance)}\ \text{as sum\_balance}}\ (\text{account})$$

account_number	branch_name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

account

Sum(balance)
4300

sum_balance

Outer Join

- An extension of the join operation that avoids loss of information
- Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join
- Uses null values:
  - null signifies that the value is unknown or does not exist
  - All comparisons involving null are false by definition
  - We shall study precise meaning of comparisons with nulls later

Outer Join

loan_number	branch_name	amount
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

loan

customer_name	loan_number
Jones	L-170
Smith	L-230
Hayes	L-155

borrower

loan_number	branch_name	amount	customer_name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

loan ⋈ borrower

Left Outer Join

loan ⟕ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*

Right Outer Join

loan ⟖ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	*null*	*null*	Hayes

Full Outer Join

loan ⟗ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*
L-155	*null*	*null*	Hayes