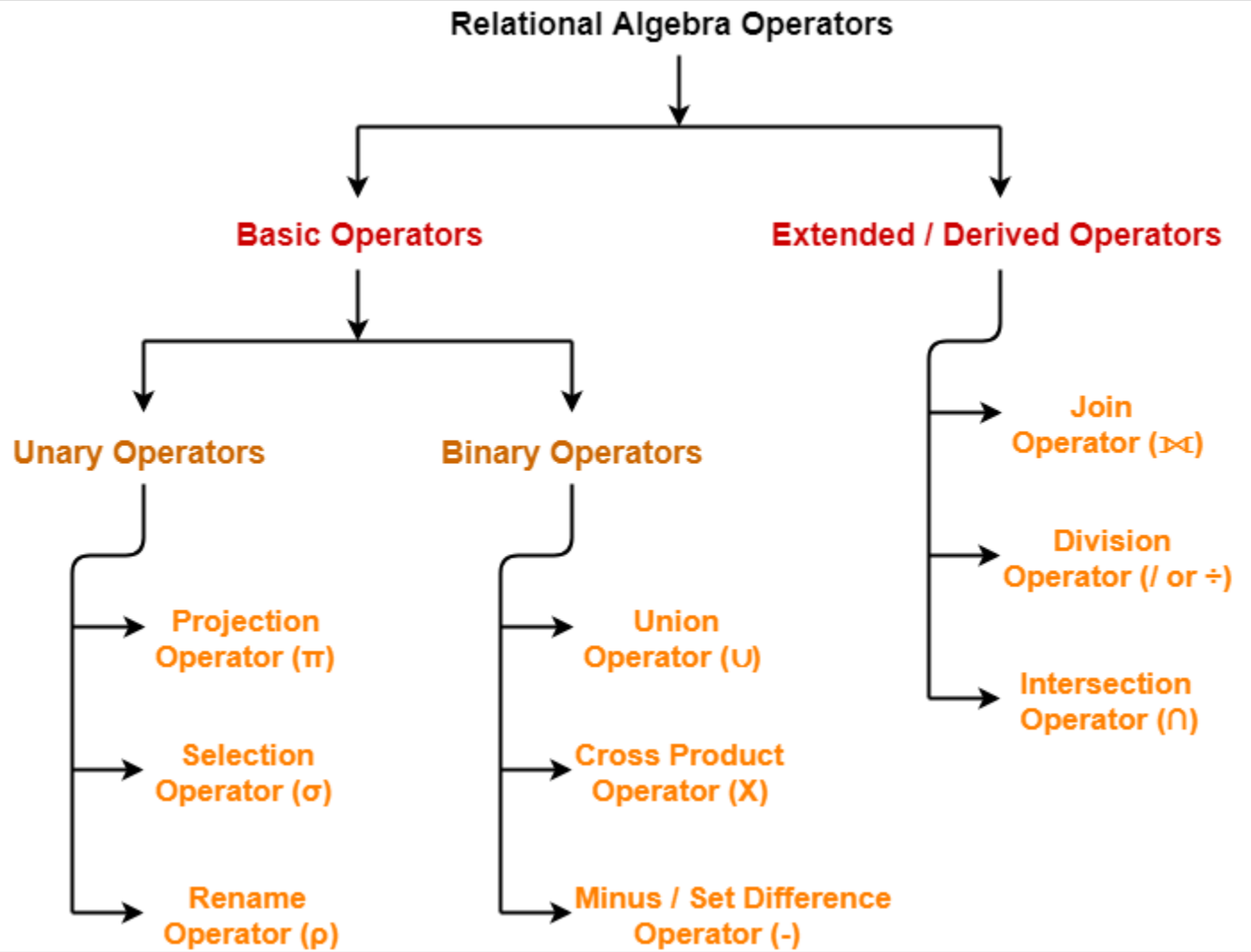
Dept. of Computer Science and Engineering
University of Rajshahi
www.ru.ac.bd

Dr. Shamim Ahmad


Relational Algebra Operators
Basic Operators
Extended / Derived Operators
Unary Operators
Binary Operators
Join Operator (⋈)
Division Operator (/ or ÷)
Intersection Operator (∩)
Projection Operator (π)
Selection Operator (σ)
Rename Operator (ρ)
Union Operator (∪)
Cross Product Operator (X)
Minus / Set Difference Operator (-)

Select and Project Operation

SELECT (σ)
The SELECT operation is used for selecting a **subset of the tuples (Horizontal subset ?)** according to a given selection condition.
Select operator selects tuples that satisfy a given predicate

Projection(π)
The projection eliminates all attributes of the input relation but those mentioned in the projection list. The projection method defines a relation that contains values for a list all attributes (**vertical subset ?) of Relation.**
.

Source: https://www.guru99.com/relational-algebra-dbms.html#2

Roll	Name	Department	Fees	Team
1	Bikash	CSE	22000	A
2	Josh	CSE	34000	A
3	Kevin	ECE	36000	C
4	Ben	ECE	56000	D

Select all the student of Team A : $\sigma_{Team = 'A'}(Student)$

Roll	Name	Department	Fees	Team
1	Bikash	CSE	22000	A
2	Josh	CSE	34000	A

Roll	Name	Department	Fees	Team
1	Bikash	CSE	22000	A
2	Josh	CSE	34000	A
3	Kevin	ECE	36000	C
4	Ben	ECE	56000	D

Select all the students of department ECE whose fees is greater then equal to 10000 and belongs to Team other than A.

$$\sigma_{\text{Fees >= 10000}}(\sigma_{\text{Class != 'A'}}\ (\text{Student}))$$

Roll	Name	Department	Fees	Team
3	Kevin	ECE	36000	C
4	Ben	ECE	56000	D

Important points about Select operation :

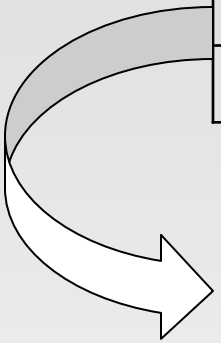
Select operator is Unary, means it it applied to single relation only. Selection operation **is commutative** that is,

$\sigma_{c1}(\sigma_{c2}(R)) = \sigma_{c2}(\sigma_{c1}(R))$

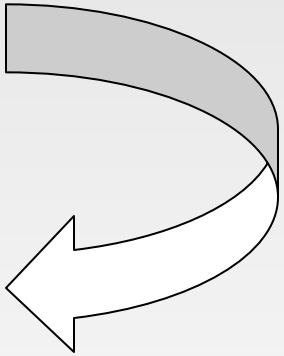
The degree (number of attributes) of resulting relation from a Selection operation is same as the degree of the Relation given. The cardinality (number of tuples) of resulting relation from a Selection operation is,

$0 <= \sigma_{c}(R) <= |R|$

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active



$\Pi_{\text{CustomerName, Status}}$ (Customers)



CustomerName	Status
Google	Active
Amazon	Active
Apple	Inactive
Alibaba	Active

Source: https://www.geeksforgeeks.org/

Class	Dept	Position
1st	CSE	Dr. Asif
1st	CSE	Dr. Kushal
2nd	CSE	Dr. Qawsar
2nd	EEE	Dr. Qawsar
3rd	EEE	Dr. Sangeeta
3rd	EEE	Dr. Asif`
1st	ICE	Dr. Sangeeta
1st	ISE	Dr. Kushal

$\pi_{Class,\ Dept}(Faculty)$?

$\pi_{Class,\ position}(Faculty)$?

$\pi_{Dept,\ position}(Faculty)$?

$\pi_{\text{Attribute List 1}}(\pi_{\text{Attribute List2}}(R)) = \pi_{\text{Attribute List 2}}(\pi_{\text{Attribute List1}}(R))$
?

$\pi_{\text{Attribute List 1}}(\pi_{\text{Attribute List2}}(R)) \mathrel{!}= \pi_{\text{Attribute List 2}}(\pi_{\text{Attribute List1}}(R))$

The Project operation is not commutative

$$\pi_{\text{Attribute List 1}}(\pi_{\text{Attribute List2}}(R)) \mathrel{!}= \pi_{\text{Attribute List 2}}(\pi_{\text{Attribute List1}}(R))$$

$$\pi_{\text{Attribute List 1}}(\pi_{\text{Attribute List2}}(R))$$

???

The following expression is valid only if
Attribute List 1 is a subset of Attribute List 2.

$$\pi_{\text{Attribute List 1}}(\pi_{\text{Attribute List2}}(R))$$

Any Alternate way?

The following expression is valid only if **Attribute List 1 is a subset of Attribute List 2.**

$$\pi_{\text{Attribute List 1}}(\pi_{\text{Attribute List2}}(R))$$

Any Alternate way?

$$\pi_{\text{Attribute List 1}}(\pi_{\text{Attribute List2}}(R)) = \pi_{\text{Attribute List 1}}(R)$$

Cartesian-Product Operation

- Notation $r \times s$
- Defined as:

$$r \times s = \{t\, q \mid t \in r \textbf{ and } q \in s\}$$

- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \varnothing$).
- If attributes of *r(R)* and *s(S)* are not disjoint, then renaming must be used.

If *r1(R1) and r2(R2)*
then r1 × r2 is a relation whose schema is the concatenation of *R1 and R2.*
Relation R contains a ***ll tuples t*** *for which* **there is**
a tuple *t1 in r1* *and*
a tuple t2 in r2*,*
for which
t[R1] = t1[R1] and t[R2] = t2[R2].

Cartesian Product Operation in Relational Algebra

the CROSS PRODUCT of two relation
A(R1, R2, R3, …, Rp) with degree p,
B(S1, S2, S3, …, Sn) with degree n,

relation C(R1, R2, R3, …, Rp, S1, S2, S3, …, Sn) with degree p + n attributes

Cartesian-Product Operation – Example

- 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

If r(R) and s(S)
then r × s is a relation whose schema is the concatenation of R and S.
Relation Q contains all tuples t for which there is
a tuple t1 in r and
a tuple t2 in s,
for which
t[R] = t1[R] and t[S] = t2[S].

Composition of Operations

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

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

Example Queries

The *loan* relation

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

$\sigma_{branch\text{-}name = \text{“Perryridge”} \wedge amount > 1200}(loan)$

loan-number	*branch-name*	*amount*
L-15	Perryridge	1500
L-16	Perryridge	1300

Example Queries

- Find all loans of over $1200

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

The *loan* relation

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

Example Queries

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

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

The *loan* Relation

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

The Union Operation

Query
To find the names of all **bank customers** who have **either** an account **or** a loan **or** both.

The *borrower* relation

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

The *depositor* Relation

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

The *customer* Relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

$\Pi_{customer\text{-}name}$ (*borrower*)

$\Pi_{customer\text{-}name}$ (*depositor*)

Example Queries

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

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

The *borrower* relation

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

The *depositor* Relation

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

customer-name
Adams
Curry
Hayes
Jackson
Jones
Smith
Williams
Lindsay
Johnson
Turner

Set Difference Operation

Query

Find all **customer**s of the bank who have an account **but** not a loan

The *borrower* relation

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

The *depositor* Relation

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

The *customer* Relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

$\Pi_{customer\text{-}name}(borrower)$

$\Pi_{customer\text{-}name}(depositor)$

Set Difference Operation

Query

Find all **customer**s of the bank who have an **account but** not a **loan**

The expression *r* – *s* produces a relation containing **those tuples in *r* but not in *s*.**

$\Pi_{customer\text{-}name}(depositor) - \Pi_{customer\text{-}name}(borrower)$

The *borrower* relation

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

The *depositor* Relation

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

customer-name
Johnson
Lindsay
Turner

Set Difference Operation

Query

Find all **customer**s of the bank who have an **loan but** not a **account**

The expression *r* – *s* produces a relation containing **those tuples in *r* but not in *s*.**

$\Pi_{customer\text{-}name}(borrower) - \Pi_{customer\text{-}name}(depositor)$

The *borrower* relation

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

The *depositor* Relation

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

Example Queries

- Find the names of all **customers** who have a **loan** at the Perryridge branch.

The *borrower* relation

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

The *loan* relation

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

The *depositor* Relation

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

The *customer* Relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

Example Queries

- Find the names of all **customers** who have a **loan** at the Perryridge branch.

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

The *borrower* relation

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

The *loan* relation

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

customer-name	*borrower. loan-number*	*loan. loan-number*	*branch-name*	*amount*
Adams	L-16	L-11	Round Hill	900
Adams	L-16	L-14	Downtown	1500
Adams	L-16	L-15	Perryridge	1500
Adams	L-16	L-16	Perryridge	1300
Adams	L-16	L-17	Downtown	1000
Adams	L-16	L-23	Redwood	2000
Adams	L-16	L-93	Mianus	500
Curry	L-93	L-11	Round Hill	900
Curry	L-93	L-14	Downtown	1500
Curry	L-93	L-15	Perryridge	1500
Curry	L-93	L-16	Perryridge	1300
Curry	L-93	L-17	Downtown	1000
Curry	L-93	L-23	Redwood	2000
Curry	L-93	L-93	Mianus	500
Hayes	L-15	L-11		900
Hayes	L-15	L-14		1500
Hayes	L-15	L-15		1500
Hayes	L-15	L-16		1300
Hayes	L-15	L-17		1000
Hayes	L-15	L-23		2000
Hayes	L-15	L-93		500
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
Smith	L-23	L-11	Round Hill	900
Smith	L-23	L-14	Downtown	1500
Smith	L-23	L-15	Perryridge	1500
Smith	L-23	L-16	Perryridge	1300
Smith	L-23	L-17	Downtown	1000
Smith	L-23	L-23	Redwood	2000
Smith	L-23	L-93	Mianus	500
Williams	L-17	L-11	Round Hill	900
Williams	L-17	L-14	Downtown	1500
Williams	L-17	L-15	Perryridge	1500
Williams	L-17	L-16	Perryridge	1300
Williams	L-17	L-17	Downtown	1000
Williams	L-17	L-23	Redwood	2000
Williams	L-17	L-93	Mianus	500

Figure 3.14 Result of *borrower* × *loan*.

customer-name	borrower. loan-number	loan. loan-number	branch-name	amount
Adams	L-16	L-11	Round Hill	900
Adams	L-16	L-14	Downtown	1500
Adams	L-16	L-15	Perryridge	1500
Adams	L-16	L-16	Perryridge	1300
Adams	L-16	L-17	Downtown	1000
Adams	L-16	L-23	Redwood	2000
Adams	L-16	L-93	Mianus	500
Curry	L-93	L-11	Round Hill	900
Curry	L-93	L-14	Downtown	1500
Curry	L-93	L-15	Perryridge	1500
Curry	L-93	L-16	Perryridge	1300
Curry	L-93	L-17	Downtown	1000
Curry	L-93	L-23	Redwood	2000
Curry	L-93	L-93	Mianus	500
Hayes	L-15	L-11		900
Hayes	L-15	L-14		1500
Hayes	L-15	L-15		1500
Hayes	L-15	L-16		1300
Hayes	L-15	L-17		1000
Hayes	L-15	L-23		2000
Hayes	L-15	L-93		500
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
Smith	L-23	L-11	Round Hill	900
Smith	L-23	L-14	Downtown	1500
Smith	L-23	L-15	Perryridge	1500
Smith	L-23	L-16	Perryridge	1300
Smith	L-23	L-17	Downtown	1000
Smith	L-23	L-23	Redwood	2000
Smith	L-23	L-93	Mianus	500
Williams	L-17	L-11	Round Hill	900
Williams	L-17	L-14	Downtown	1500
Williams	L-17	L-15	Perryridge	1500
Williams	L-17	L-16	Perryridge	1300
Williams	L-17	L-17	Downtown	1000
Williams	L-17	L-23	Redwood	2000
Williams	L-17	L-93	Mianus	500

Figure 3.14 Result of borrower × loan.

customer-name	borrower. loan-number	loan. loan-number	branch-name	amount
Adams	L-16	L-11	Round Hill	900
Adams	L-16	L-14	Downtown	1500
Adams	L-16	L-15	Perryridge	1500
Adams	L-16	L-16	Perryridge	1300
Adams	L-16	L-17	Downtown	1000
Adams	L-16	L-23	Redwood	2000
Adams	L-16	L-93	Mianus	500
Curry	L-93	L-11	Round Hill	900
Curry	L-93	L-14	Downtown	1500
Curry	L-93	L-15	Perryridge	1500
Curry	L-93	L-16	Perryridge	1300
Curry	L-93	L-17	Downtown	1000
Curry	L-93	L-23	Redwood	2000
Curry	L-93	L-93	Mianus	500
Hayes	L-15	L-11		900
Hayes	L-15	L-14		1500
Hayes	L-15	L-15		1500
Hayes	L-15	L-16		1300
Hayes	L-15	L-17		1000
Hayes	L-15	L-23		2000
Hayes	L-15	L-93		500
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
Smith	L-23	L-11	Round Hill	900
Smith	L-23	L-14	Downtown	1500
Smith	L-23	L-15	Perryridge	1500
Smith	L-23	L-16	Perryridge	1300
Smith	L-23	L-17	Downtown	1000
Smith	L-23	L-23	Redwood	2000
Smith	L-23	L-93	Mianus	500
Williams	L-17	L-11	Round Hill	900
Williams	L-17	L-14	Downtown	1500
Williams	L-17	L-15	Perryridge	1500
Williams	L-17	L-16	Perryridge	1300
Williams	L-17	L-17	Downtown	1000
Williams	L-17	L-23	Redwood	2000
Williams	L-17	L-93	Mianus	500

Figure 3.14 Result of borrower × loan.

$$\sigma_{branch\text{-}name = \text{“Perryridge”}}(borrower \times loan)$$

customer-name	borrower. loan-number	loan. loan-number	branch-name	amount
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

Example Queries

- Find the names of all customers who have a loan at the Perryridge branch.

$$\sigma_{borrower.loan\text{-}number = loan.loan\text{-}number}(\sigma_{branch\text{-}name = \text{“Perryridge”}}(borrower \times loan))$$

To have only customer-name, we do a projection

$$\Pi_{customer\text{-}name}(\sigma_{borrower.loan\text{-}number = loan.loan\text{-}number}(\sigma_{branch\text{-}name = \text{“Perryridge”}}(borrower \times loan)))$$

customer-name
Adams
Hayes

Result of

$\Pi_{customer\text{-}name}$

$(\sigma_{borrower.loan\text{-}number = loan.loan\text{-}number}$

$(\sigma_{branch\text{-}name = \text{“Perryridge”}} (borrower \times loan))).$

Example Queries

- Find the names of all customers who have a loan at the Perryridge branch.

$$\Pi_{customer\text{-}name}(\sigma_{borrower.loan\text{-}number = loan.loan\text{-}number}(\sigma_{branch\text{-}name = \text{“Perryridge”}}(borrower \times loan))).$$

OR ??

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

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us 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

$$\rho_{x(A_1,A_2,\ldots,A_n)}(E)$$

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

Example-1:
Query to rename the relation **Student** as **MaleStudent** and the attributes of **Student – RollNo, SName** as (**Eno, Name**).

$$\rho_{\text{MaleStudent(Eno, Name)}}\ \pi_{\text{RollNo, SName}}(\sigma_{\text{Condition}}(\text{Student}))$$

Eno	Name
2600	Ronny
2655	Raja

Example:
Query to rename the attributes Name, Age of table Department to A,B.

$$\rho_{(A, B)} (Department)$$

Example:
Query to rename the table name Project to Pro and its attributes to P, Q, R.

$$\rho_{Pro(P, Q, R)} (Project)$$

Example:
Query to rename the first attribute of the table Employee with attributes A, B, C to P.

$$\rho_{(P, B, C)} (Employee)$$

Query:

Find the **largest account** balance in the bank.

The *account* relation

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_no	Brance_name	Balance
101	Rajshahi	700
104	Bogra	450
112	Rajshahi	900
130	Khulna	600
140	Dhaka	350

Account_no	Brance_name	Balance	Account_no	Brance_name	Balance
101	Rajshahi	700	101	Rajshahi	700
101	Rajshahi	700	104	Bogra	450
101	Rajshahi	700	112	Rajshahi	900
101	Rajshahi	700	130	Khulna	600
101	Rajshahi	700	140	Dhaka	350
104	Bogra	450	101	Rajshahi	700
104	Bogra	450	104	Bogra	450
104	Bogra	450	112	Rajshahi	900
104	Bogra	450	130	Khulna	600
104	Bogra	450	140	Dhaka	350
112	Rajshahi	900	101	Rajshahi	700
112	Rajshahi	900	104	Bogra	450
112	Rajshahi	900	112	Rajshahi	900
112	Rajshahi	900	130	Khulna	600
112	Rajshahi	900	140	Dhaka	350
130	Khulna	600	101	Rajshahi	700
130	Khulna	600	104	Bogra	450
130	Khulna	600	112	Rajshahi	900
130	Khulna	600	130	Khulna	600
130	Khulna	600	140	Dhaka	350
140	Dhaka	350	101	Rajshahi	700
140	Dhaka	350	104	Bogra	450
140	Dhaka	350	112	Rajshahi	900
140	Dhaka	350	130	Khulna	600
140	Dhaka	350	140	Dhaka	350

$\sigma_{t1.balance < t2.balance}$

Account_no	Brance_name	Balance	Account_no	Brance_name	Balance
101	Rajshahi	700	101	Rajshahi	700
101	Rajshahi	700	104	Bogra	450
101	Rajshahi	700	112	Rajshahi	900
101	Rajshahi	700	130	Khulna	600
101	Rajshahi	700	140	Dhaka	350
104	Bogra	450	101	Rajshahi	700
104	Bogra	450	104	Bogra	450
104	Bogra	450	112	Rajshahi	900
104	Bogra	450	130	Khulna	600
104	Bogra	450	140	Dhaka	350
112	Rajshahi	900	101	Rajshahi	700
112	Rajshahi	900	104	Bogra	450
112	Rajshahi	900	112	Rajshahi	900
112	Rajshahi	900	130	Khulna	600
112	Rajshahi	900	140	Dhaka	350
130	Khulna	600	101	Rajshahi	700
130	Khulna	600	104	Bogra	450
130	Khulna	600	112	Rajshahi	900
130	Khulna	600	130	Khulna	600
130	Khulna	600	140	Dhaka	350
140	Dhaka	350	101	Rajshahi	700
140	Dhaka	350	104	Bogra	450
140	Dhaka	350	112	Rajshahi	900
140	Dhaka	350	130	Khulna	600
140	Dhaka	350	140	Dhaka	350

$\Pi_{T1.balance}$ $\Pi_{T2.balance}$

Account_no	Brance_name	Balance	Account_no	Brance_name	Balance
101	Rajshahi	700	101	Rajshahi	700
101	Rajshahi	700	104	Bogra	450
101	Rajshahi	700	112	Rajshahi	900
101	Rajshahi	700	130	Khulna	600
101	Rajshahi	700	140	Dhaka	350
104	Bogra	450	101	Rajshahi	700
104	Bogra	450	104	Bogra	450
104	Bogra	450	112	Rajshahi	900
104	Bogra	450	130	Khulna	600
104	Bogra	450	140	Dhaka	350
112	Rajshahi	900	101	Rajshahi	700
112	Rajshahi	900	104	Bogra	450
112	Rajshahi	900	112	Rajshahi	900
112	Rajshahi	900	130	Khulna	600
112	Rajshahi	900	140	Dhaka	350
130	Khulna	600	101	Rajshahi	700
130	Khulna	600	104	Bogra	450
130	Khulna	600	112	Rajshahi	900
130	Khulna	600	130	Khulna	600
130	Khulna	600	140	Dhaka	350
140	Dhaka	350	101	Rajshahi	700
140	Dhaka	350	104	Bogra	450
140	Dhaka	350	112	Rajshahi	900
140	Dhaka	350	130	Khulna	600
140	Dhaka	350	140	Dhaka	350

Query:

Find the largest account balance in the bank.

(1) compute first a **temporary relation** consisting of those balances that are ***not*** **the largest**

(2) take the set difference between the relation $\Pi_{balance}(account)$ and the temporary relation just computed, to obtain

(1) compute first a **temporary relation** consisting of those balances that are ***not*** **the largest**

$$\Pi_{account.balance}(\sigma_{account.balance < d.balance}(account \times \rho_{d}(account)))$$

balance
500
400
700
750
350

Query:
Find the largest account balance in the bank.

(2) take the set difference between
the relation $\Pi_{balance}(account)$ and the temporary relation just computed, to obtain

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

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

balance
500
400
700
750
350

balance
900

???

$$\Pi_{account.balance}\left(\sigma_{account.balance > d.balance}\left(account \times \rho_{d}(account)\right)\right)$$

Account_no	Brance_name	Balance	Account_no	Brance_name	Balance
101	Rajshahi	700	101	Rajshahi	700
101	Rajshahi	700	104	Bogra	450
101	Rajshahi	700	112	Rajshahi	900
101	Rajshahi	700	130	Khulna	600
101	Rajshahi	700	140	Dhaka	350
104	Bogra	450	101	Rajshahi	700
104	Bogra	450	104	Bogra	450
104	Bogra	450	112	Rajshahi	900
104	Bogra	450	130	Khulna	600
104	Bogra	450	140	Dhaka	350
112	Rajshahi	900	101	Rajshahi	700
112	Rajshahi	900	104	Bogra	450
112	Rajshahi	900	112	Rajshahi	900
112	Rajshahi	900	130	Khulna	600
112	Rajshahi	900	140	Dhaka	350
130	Khulna	600	101	Rajshahi	700
130	Khulna	600	104	Bogra	450
130	Khulna	600	112	Rajshahi	900
130	Khulna	600	130	Khulna	600
130	Khulna	600	140	Dhaka	350
140	Dhaka	350	101	Rajshahi	700
140	Dhaka	350	104	Bogra	450
140	Dhaka	350	112	Rajshahi	900
140	Dhaka	350	130	Khulna	600
140	Dhaka	350	140	Dhaka	350

???

$$\Pi_{account.balance}(\sigma_{account.balance > d.balance}(account \times \rho_{d}(account)))$$

$\Pi_{T1.balance}$ $\Pi_{T2.balance}$

Account_no	Brance_name	Balance	Account_no	Brance_name	Balance
101	Rajshahi	700	101	Rajshahi	700
101	Rajshahi	700	104	Bogra	450
101	Rajshahi	700	112	Rajshahi	900
101	Rajshahi	700	130	Khulna	600
101	Rajshahi	700	140	Dhaka	350
104	Bogra	450	101	Rajshahi	700
104	Bogra	450	104	Bogra	450
104	Bogra	450	112	Rajshahi	900
104	Bogra	450	130	Khulna	600
104	Bogra	450	140	Dhaka	350
112	Rajshahi	900	101	Rajshahi	700
112	Rajshahi	900	104	Bogra	450
112	Rajshahi	900	112	Rajshahi	900
112	Rajshahi	900	130	Khulna	600
112	Rajshahi	900	140	Dhaka	350
130	Khulna	600	101	Rajshahi	700
130	Khulna	600	104	Bogra	450
130	Khulna	600	112	Rajshahi	900
130	Khulna	600	130	Khulna	600
130	Khulna	600	140	Dhaka	350
140	Dhaka	350	101	Rajshahi	700
140	Dhaka	350	104	Bogra	450
140	Dhaka	350	112	Rajshahi	900
140	Dhaka	350	130	Khulna	600
140	Dhaka	350	140	Dhaka	350

Example Queries

- 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.

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

$$(\sigma_{borrower.loan\_number = loan.loan\_number}(\text{borrower x loan}))) -$$
$$\Pi_{customer\_name}(\text{depositor})$$

Formal Definition

- A basic expression in the relational algebra consists of either one of the following:
  - A relation in the database
  - A constant relation
- Let ***E_1* and *E_2* be relational-algebra expressions**; the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p (E_1)$, P is a predicate on attributes in E_1
  - $\Pi_s(E_1)$, S is a list consisting of some of the attributes in E_1
  - $\rho_x (E_1)$, x is the new name for the result of E_1

Additional Operations

We define additional operations that do not add any power to the relational algebra, but that simplify common queries.

- Set intersection
- Natural join
- Division
- Assignment

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$
- Assume:
  - r, s have the *same arity*
  - attributes of r and s are compatible
- Note: $\boldsymbol{r \cap s = r - (r - s)}$

Set-Intersection Operation – Example

- Relation *r, s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

Query

Find all customers who have both a loan and an account.

The *account* relation

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

The *depositor* Relation

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

The *borrower* relation

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

The *customer* Relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

Query

Find all customers who have both a loan and an account.

The *borrower* relation

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

The *depositor* Relation

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

customer-name
Hayes
Jones
Smith

$\Pi_{customer\text{-}name}(borrower)$

$\Pi_{customer\text{-}name}(depositor)$

$\Pi customer\text{-}name\ (borrower) \cap \Pi customer\text{-}name\ (depositor)$

? = $\Pi customer\text{-}name\ (\mathbf{borrower}) -$

$(\Pi customer\text{-}name\ (\mathbf{borrower}) - \Pi customer\text{-}name\ (\mathbf{depositor}))$

Query

Find all customers who have both a loan and an account.

The *borrower* relation

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

The *depositor* Relation

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

customer-name
Hayes
Jones
Smith

$\Pi_{customer\text{-}name}(borrower)$

$\Pi_{customer\text{-}name}(depositor)$

$\Pi_{customer\text{-}name}(borrower) \cap \Pi_{customer\text{-}name}(depositor)$

? = $\Pi_{customer\text{-}name}(\mathbf{depositor}) -$

$(\Pi_{customer\text{-}name}(\mathbf{depositor}) - \Pi_{customer\text{-}name}(\mathbf{borrower}))$

Natural-Join Operation

- Notation: r ⋈ s
- Let *r* and *s* be relations on schemas *R* and *S* respectively.
  Then, r ⋈ s is a relation on schema $R \cup S$ 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 $R \cap S$, 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*
- Example:

  $R = (A, B, C, D)$

  $S = (E, B, D)$
  - Result schema = (A, ***B***, C, ***D,*** E)
  - $r \bowtie s$ is defined as:

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

Natural Join Operation – Example

- Relations r, s:

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	∈

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

Query:
Find the names of **all customers** who have **a loan** at the bank, along with the **loan number** and the **loan amount**

The *borrower* relation

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

The *loan* relation

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

The *depositor* Relation

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

The *customer* Relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

Query:
Find the names of **all customers** who have **a loan** at the bank, along with the **loan number** and the **loan amount**

$$\Pi_{customer\text{-}name,\ loan.loan\text{-}number,\ amount}\left(\sigma_{borrower.loan\text{-}number = loan.loan\text{-}number}\left(borrower \times loan\right)\right)$$

- Cartesian product of its two arguments:
- Selection forcing equality on those attributes that appear in both relation schemas,
- Finally removes duplicate attributes

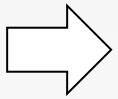
$R = (A, B, C, D)$

$S = (E, B, D)$

- Result schema = (A, ***B***, C, ***D***, E)
- r s is defined as:

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

Let $r(R)$ and $s(S)$ be relations without any attributes in common; that is, $R \cap S = \emptyset$.



$$r \bowtie_\theta s = \sigma_\theta(r \times s)$$

Query:
Find the names of **all customers** who have **a loan** at the bank, along with the **loan number** and the **loan amount**

Π*customer-name, loan.loan-number, amount*
(σ*borrower .loan-number =loan.loan-number* (*borrower × loan*))

The *borrower* relation

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

The *loan* relation

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

customer-name	*loan-number*	*amount*
Adams	L-16	1300
Curry	L-93	500
Hayes	L-15	1500
Jackson	L-14	1500
Jones	L-17	1000
Smith	L-23	2000
Smith	L-11	900
Williams	L-17	1000

$$\Pi_{customer\text{-}name,\ loan\text{-}number,\ amount}\ (borrower \bowtie loan)$$

Query:

Find the **names of all branches** with **customers** who have an account in the bank and who live in **Harrison.**

The *borrower* relation

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

The *loan* relation

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

The *branch* relation

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

The *depositor* Relation

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

The *customer* Relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

The *account* relation

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

Query:
Find the **names of all branches** with **customers** who have an account in the bank and who live in **Harrison.**

$$\Pi_{branch\text{-}name}\left(\sigma_{customer\text{-}city = \text{“Harrison”}}\left(customer \bowtie account \bowtie depositor\right)\right)$$

The *depositor* Relation

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

The *customer* Relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

The *account* relation

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

Query:

Find the **names of all branches** with **customers** who have an account in the bank and who live in **Harrison.**

- ☐ $(customer \bowtie account) \bowtie depositor$
- ☐ $customer \bowtie (account \bowtie depositor)$

The *account* relation

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

The *depositor* Relation

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

The *customer* Relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

A-101	Downtown	500

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

Query:

Find the **names of all branches** with **customers** who have an account in the bank and who live in **Harrison.**

The *account* relation

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

The *depositor* Relation

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

The *customer* Relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

A-217	Brighton	750	

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

Query:

Find the **names of all branches** with **customers** who have an account in the bank and who live in **Harrison.**

The *account* relation

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

The *depositor* Relation

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

The *customer* Relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

			Hayes	A-102
			Johnson	A-101
			Johnson	A-201
A-217	Brighton	750	Jones	A-217
			Lindsay	A-222
			Smith	A-215
			Turner	A-305

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

Division Operation

- 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
  - $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S = (B_1, \ldots, B_n)$

  The result of r ÷ 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

r ÷ s is used when we wish to express queries with “all”

Ex.
“Which persons have a loyal customer's card at ALL the clothing boutiques in town X?”
“Which persons have a bank account at ALL the banks in the country?”
“Which students are registered on ALL the courses given by Soini?”
“Which students are registered on ALL the courses that are taught in period 1?”
“Which boys are registered on those courses that are taken by ALL the girls?”
“Which girls are registered on ALL the courses taken by student nr. 40101?”

“**IF** there is a clothing boutique in town X **THEN** the person has a loyal customer's card there”,
“**IF** Soini gives a course **THEN** the student is registered on it” or
“**IF** 40101 is registered on a course **THEN** the girl is registered on the same course”.

Query:
"Which girls are registered on ALL the courses taken by student nr. 40101?"

Suppose that 40101 takes three courses: G555, 456306 and 456302.
To get e a part of the answer **the girl in** question
must be registered on ALL of these three.
(In addition to that she may be registered on other courses not taken by 40101, these will not affect the result.)

matr	namn	up	kön	kurskod
40112	Brita	DT	K	456306
40112	Brita	DT	K	456302
40113	Ann-Helen	DT	K	456304
40113	Ann-Helen	DT	K	456306
40113	Ann-Helen	DT	K	456302
40128	Siru	DT	K	G555
40128	Siru	DT	K	456306
40128	Siru	DT	K	456302
40240	Sara	IS	K	456304
40240	Sara	IS	K	456306
40240	Sara	IS	K	456302

matr	kurskod
40101	G555
40101	456306
40101	456302

Query:
“Which girls are registered on ALL the courses taken by student nr. 40101?”

$$\Pi_{\text{matr, namn, kurskod}}(\sigma_{\text{kön = 'K'}}(\text{student}) \,|x|\, \text{kursanmälan}) \div \Pi_{\text{kurskod}}(\sigma_{\text{matr = 40101}}(\text{kursanmälan}))$$

matr	namn	up	kön	kurskod
40112	Brita	DT	K	456306
40112	Brita	DT	K	456302
40113	Ann-Helen	DT	K	456304
40113	Ann-Helen	DT	K	456306
40113	Ann-Helen	DT	K	456302
40128	Siru	DT	K	G555
40128	Siru	DT	K	456306
40128	Siru	DT	K	456302
40240	Sara	IS	K	456304
40240	Sara	IS	K	456306
40240	Sara	IS	K	456302

matr	kurskod
40101	G555
40101	456306
40101	456302

Example 3: for X1 of r for all Y1 and Y2 of s ???

r

X1	Y1	Y2	X2
A	P	3	1
A	P	4	1
B	Q	2	6
A	P	3	1
C	Z	1	2
A	P	1	1
B	Q	4	1
A	P	2	2
A	Q	2	4
A	P	5	1

s

Y1	Y2
P	1
P	2
Q	2

Example 3: for X1 of r for all Y1 and Y2 of s ???

r

X1	Y1	Y2
A	P	3
A	P	4
B	Q	2
A	P	3
C	Z	1
A	P	1
B	Q	4
A	P	2
A	Q	2
A	P	5

s

Y1	Y2
P	1
P	2
Q	2

Example 3: for X1 and X2 of r for all Y1 and Y2 of s ???

r

X1	Y1	Y2	X2	X3
A	P	3	1	?
A	P	4	1	?
B	Q	2	6	?
A	P	3	1	?
C	Z	1	2	?
A	P	1	1	?
B	Q	4	1	?
A	P	2	2	?
A	Q	2	4	?
A	P	5	1	?

s

Y1	Y2
P	1
P	2
Q	2

Which boys name and roll are registered on **ALL courses** that are taken by the girls?”

B.Name	Roll	Cell	Course ID
X	201	01711	CSE101
X	201	01711	CSE 102
X	202	01711	CSE 103
Y	202	01911	CSE 102
Y	202	01911	CSE 101
Z	203	01811	CSE 104
Z	203	01811	CSE 101
X	204	01811	CSE 104

G.Name	Roll	Cell	Course ID
A	101	01711	CSE101
A	101	01711	CSE 102
A	101	01711	CSE 103
B	102	01911	CSE 102
B	102	01911	CSE 101
C	103	01811	CSE 104
C	103	01811	CSE 101
C	103	01811	CSE 102

Which boys name and roll are registered on **ALL courses** that are taken by the girls?”

B.Name	Roll	Cell	Course ID
X	201	01711	CSE101
X	201	01711	CSE 102
X	201	01711	CSE 103
Y	202	01911	CSE 102
Y	202	01911	CSE 101
Z	203	01811	CSE 104
Z	203	01811	CSE 101
X	201	01811	CSE 104

G.Name	Roll	Cell	Course ID
A	101	01711	CSE101
A	101	01711	CSE 102
A	101	01711	CSE 103
B	102	01911	CSE 102
B	102	01911	CSE 101
C	103	01811	CSE 104
C	103	01811	CSE 101
C	103	01811	CSE 102

Which boys name and roll are registered on **ALL courses** that are taken by the girls?”

B.Name	Roll	Course ID
X	201	CSE101
X	201	CSE 102
X	201	CSE 103
Y	202	CSE 102
Y	202	CSE 101
Z	203	CSE 104
Z	203	CSE 101
X	201	CSE 104

Course ID
CSE101
CSE 102
CSE 103
CSE 104

Which boys name and roll are registered on **ALL courses** that are taken by the girls?”

B.Name	Roll	Course ID
X	201	CSE101
X	201	CSE 102
X	201	CSE 103
Y	202	CSE 102
Y	202	CSE 101
Z	203	CSE 104
Z	203	CSE 101
X	201	CSE 104

Course ID
CSE101
CSE 102
CSE 103
CSE 104

Which boys name and roll are registered on those courses that are taken by ALL the girls?”

B.Name	Roll	Cell	Course ID
X	201	01711	CSE101
X	201	01711	CSE 102
Y	202	01711	CSE 103
Y	202	01911	CSE 102
Y	202	01911	CSE 101
Z	203	01811	CSE 104
Z	203	01811	CSE 101
W	204	01811	CSE 102

G.Name	Roll	Cell	Course ID
A	101	01711	CSE101
A	101	01711	CSE 102
A	101	01711	CSE 103
B	102	01911	CSE 102
B	102	01911	CSE 101
C	103	01811	CSE 104
C	103	01811	CSE 101
C	103	01811	CSE 102

Which boys name and roll are registered on those courses that are taken by ALL the girls?”

B.Name	Roll	Cell	Course ID
X	201	01711	CSE101
X	201	01711	CSE 102
Y	202	01711	CSE 103
Y	202	01911	CSE 102
Y	202	01911	CSE 101
Z	203	01811	CSE 104
Z	203	01811	CSE 101
W	204	01811	CSE 102

G.Name	Roll	Cell	Course ID
A	101	01711	CSE101
A	101	01711	CSE 102
A	101	01711	CSE 103
B	102	01911	CSE 102
B	102	01911	CSE 101
C	103	01811	CSE 104
C	103	01811	CSE 101
C	103	01811	CSE 102

Which boys **name** and roll are registered on those **courses** that are taken by ALL the girls?”

B.Name	Roll	Cell	Course ID
X	201	01711	CSE101
X	201	01711	CSE 102
Y	202	01711	CSE 103
Y	202	01911	CSE 102
Y	202	01911	CSE 101
Z	203	01811	CSE 104
Z	203	01811	CSE 101
W	204	01811	CSE 102

G.Name	Roll	Cell	Course ID
A	101	01711	CSE101
A	101	01711	CSE 102
A	101	01711	CSE 103
B	102	01911	CSE 102
B	102	01911	CSE 101
C	103	01811	CSE 104
C	103	01811	CSE 101
C	103	01811	CSE 102

Which boys **name** and roll are registered on those **courses** that are taken by ALL the girls?”

G.Name/Roll??? Primary key

B.Name	Roll	Course ID
X	201	CSE101
X	201	CSE 102
Y	202	CSE 103
Y	202	CSE 102
Y	202	CSE 101
Z	203	CSE 104
Z	203	CSE 101
W	204	CSE 102

G.Name	Course ID
A	CSE101
A	CSE 102
A	CSE 103
B	CSE 102
B	CSE 101
C	CSE 104
C	CSE 101
C	CSE 102

Which boys **name** and roll are registered on those **courses** that are taken by ALL the girls?”

G.Name/Roll??? Primary key

B.Name	Roll	Course ID
X	201	CSE101
X	201	CSE 102
Y	202	CSE 103
Y	202	CSE 102
Y	202	CSE 101
Z	203	CSE 104
Z	203	CSE 101
W	204	CSE 102

G.Name	Course ID
A	CSE101
A	CSE 102
A	CSE 103
B	CSE 102
B	CSE 101
C	CSE 104
C	CSE 101
C	CSE 102

G.Name	Course ID
A	CSE101
A	CSE 102
A	CSE 103
B	CSE 102
B	CSE 101
C	CSE 104
C	CSE 101
C	CSE 102

Which boys **name** and roll are registered on those **courses** that are taken by ALL the girls?”

B.Name	Roll	Course ID
X	201	CSE101
X	201	CSE 102
Y	202	CSE 103
Y	202	CSE 102
Y	202	CSE 101
Z	203	CSE 104
Z	203	CSE 101
W	204	CSE 102

G.Name	Course ID
A	CSE101
A	CSE 102
A	CSE 103
B	CSE 102
B	CSE 101
C	CSE 104
C	CSE 101
C	CSE 102

Course ID	G.Name
CSE101	A
CSE101	B
CSE101	C
CSE 102	A
CSE 102	B
CSE 102	C
CSE 103	A
CSE 104	C

Which boys **name** and roll are registered on those **courses** that are taken by ALL the girls?”

B.Name	Roll	Course ID
X	201	CSE101
X	201	CSE 102
Y	202	CSE 103
Y	202	CSE 102
Y	202	CSE 101
Z	203	CSE 104
Z	203	CSE 101
W	204	CSE 102

Course ID	G.Name
CSE101	A
CSE101	B
CSE101	C
CSE 102	A
CSE 102	B
CSE 102	C
CSE 103	A
CSE 104	C

G.Name
A
B
C

Which boys **name** and roll are registered on those **courses** that are taken by ALL the girls?”

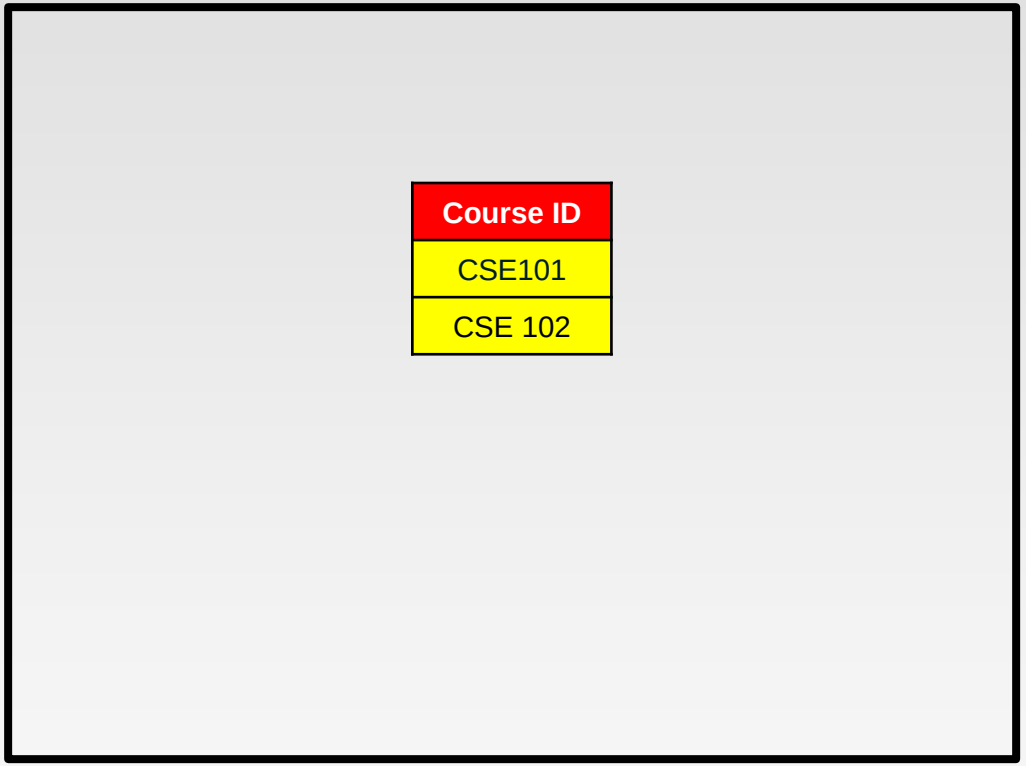
B.Name	Roll	Course ID
X	201	CSE101
X	201	CSE 102
Y	202	CSE 103
Y	202	CSE 102
Y	202	CSE 101
Z	203	CSE 104
Z	203	CSE 101
W	204	CSE 102

Course ID
CSE101
CSE 102

Which boys **name** and roll are registered on those **courses** that are taken by ALL the girls?”

G.Name/Roll??? Primary key

B.Name	Roll	Course ID
X	201	CSE101
X	201	CSE 102
Y	202	CSE 103
Y	202	CSE 102
Y	202	CSE 101
Z	203	CSE 104
Z	203	CSE 101
W	204	CSE 102



Which boys **name** and roll are registered on those **courses** that are taken by ALL the girls?”

B.Name	Roll	Course ID
X	201	CSE101
X	201	CSE 102
Y	202	CSE 103
Y	202	CSE 102
Y	202	CSE 101
Z	203	CSE 104
Z	203	CSE 101
W	204	CSE 102

B.Name	Roll
X	201
Y	202

Division Operation

Query 1

All **customers** who have an **account** at ***all*** **the branches located in Brooklyn.**

The *borrower* relation

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

The *loan* relation

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

The *branch* relation

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

The *depositor* Relation

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

The *customer* Relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

The *account* relation

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

Division Operation

Query 1

All **customers** who have an **account** at ***all*** **the branches located in Brooklyn.**

Step 1:
obtain all branches in Brooklyn

$r1 = \Pi_{branch\text{-}name}(\sigma_{branch\text{-}city = \text{“Brooklyn”}}(branch))$

The *branch* relation

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-name
Brighton
Downtown

Division Operation

Query 1

All **customers** who have an **account** at ***all*** **the** **branches** **located in Brooklyn.**

Step 2:
find all (*customer-name, branch-name*) pairs
for which **the customer** has an
account at a **branch**

$$r_2 = \Pi_{customer\text{-}name,\ branch\text{-}name}\ (depositor \bowtie account)$$

The *depositor* Relation

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

The *account* relation

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

customer-name	*branch-name*
Hayes	Perryridge
Johnson	Downtown
Johnson	Brighton
Jones	Brighton
Lindsay	Redwood
Smith	Mianus
Turner	Round Hill

Division Operation

To find customers who appear in r_2 with *every* branch name in r_1.

The operation that provides exactly those customers is the divide operation.
We formulate the query by writing

Step 3:
To find **customers** who appear **in r_2 with *every* branch name in r_1.**
The operation that provides exactly those customers is the divide operation.

$$\Pi_{customer\text{-}name,\ branch\text{-}name}\ (depositor \bowtie account) \div \Pi_{branch\text{-}name}\ (\sigma_{branch\text{-}city = \text{“Brooklyn”}}\ (branch))$$

customer-name	*branch-name*
Hayes	Perryridge
Johnson	Downtown
Johnson	Brighton
Jones	Brighton
Lindsay	Redwood
Smith	Mianus
Turner	Round Hill

branch-name
Brighton
Downtown

Division Operation

let ***r(R)*** and ***s(S)*** be relations,
let ***S ⊆ R***;

(that is, every attribute of schema *S* is also in schema *R*)

The relation ***r ÷ s*** is a relation on schema ***R – S***

(that is, on the schema containing all attributes of schema *R* that are not in schema *S*).

A tuple ***t* is in *r ÷ s*** if and only if both of two conditions hold:

1. *t* is in $\Pi_{R-S}(r)$

2. For every tuple t_s in *s*, there is a tuple t_r in *r* satisfying both of the following:

a. $t_r[S] = t_s[S]$

b. $t_r[R - S] = t$

B.Name	Roll	Course ID
X	201	CSE101
X	201	CSE 102
X	201	CSE 103
Y	202	CSE 102
Y	202	CSE 101
Z	203	CSE 104
Z	203	CSE 101
X	201	CSE 104

Course ID
CSE101
CSE 102
CSE 103
CSE 104

a. t_r**[Course ID] =** t_s**[Course ID]**

b. **b.** t_r***[B.Name, Roll] = t***

Division Operation

Let $r(R)$ and $s(S)$ be given, with $S \subseteq R$:

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$$

A tuple t is in $r \div s$ if and only if both of two conditions hold:

1. t is in $\Pi_{R-S}(r)$

2. For every tuple t_s in s, there is a tuple t_r in r satisfying both of the following:

a. $t_r[S] = t_s[S]$

b. $t_r[R - S] = t$

$$\Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$$

$\Pi_{R-S,S}(r)$ merely reorders the attributes of r.`

$\Pi_{R-S}(r) \times s$. This relation is on schema R
Pairs every tuple in $\Pi_{R-S}(r)$ with every tuple in s.

$(\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r)$ (gives us those pairs of tuples
from $\Pi_{R-S}(r)$ and s that do not appear in r.

$\Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$ If a tuple t_j is in here;
Then there is some tuple t_s in s that does not
combine with tuple t_j to form a tuple in r.

Thus, tj holds a value for attributes $R - S$ that does not appear in $r \div s$. It is these values that we eliminate from $\Pi_{R-S}(r)$.

Division Operation – Example

- Relations *r, s*:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
$\in$	6
$\in$	1
β	2

r

B
1
2

s

- $r \div s$:

A
α
β

Another Division Example

- Relations *r, s*:

A	*B*	*C*	*D*	*E*
α	a	α	a	*1*
α	a	γ	a	*1*
α	a	γ	b	*1*
β	a	γ	a	*1*
β	a	γ	b	*3*
γ	a	γ	a	*1*
γ	a	γ	b	*1*
γ	a	β	b	*1*

r

D	*E*
a	*1*
b	*1*

s

- $r \div s$:

A	*B*	*C*
α	a	γ
γ	a	γ

Example 1: for X1 and X2 of r for all Y1 and Y2 of s

r

X1	Y1	Y2	X2
A	P	3	1
B	Q	2	1
A	P	3	1
C	P	1	2
A	P	1	1
B	Q	2	1
A	P	2	1
C	P	2	2
A	Q	2	1

s

Y1	Y2
P	1
P	2
Q	2

Example 1

r

X1	Y1	Y2	X2
A	P	3	1
B	Q	2	1
A	P	3	1
C	P	1	2
A	P	1	1
B	Q	2	1
A	P	2	1
C	P	2	2
A	Q	2	1

R

X1	Y1	Y2	X2

S

Y1	Y2

R-S

X1	X2

$\Pi_{R-S,S}(r)$

=

$\Pi_{X1,X2,Y1,Y2}(r)$

$\Pi_{R-S,S}(r)$ merely reorders the attributes of *r*.`

Example 1

r

X1	Y1	Y2	X2
A	P	3	1
B	Q	2	1
A	P	3	1
C	P	1	2
A	P	1	1
B	Q	2	1
A	P	2	1
C	P	2	2
A	Q	2	1

$\Pi_{R-S,S}(r) = \Pi_{X1,X2,Y1,Y2}(r)$

X1	X2	Y1	Y2
A	1	P	3
B	1	Q	2
A	1	P	3
C	2	P	1
A	1	P	1
B	1	Q	2
A	1	P	2
C	2	P	2
A	1	Q	2

$\Pi_{R-S,S}(r)$ merely reorders the attributes of *r*.`

Example 1

$$\Pi_{R-S}(r) = \Pi_{X1,X2}(r)$$

X1	X2
A	1
B	1
A	1
C	2
A	1
B	1
A	1
C	2
A	1

X1	X2
A	1
B	1
C	2

S

Y1	Y2
P	1
P	2
Q	2

Example 1

$\Pi_{R-S}(r) = \Pi_{X1,X2}(r)$

X1	X2
A	1
B	1
C	2

s

Y1	Y2
P	1
P	2
Q	2

$\Pi_{R-S}(r) \times s$

X1	X2	Y1	Y2
A	1	P	1
A	1	P	2
A	1	Q	2
B	1	P	1
B	1	P	2
B	1	Q	2
C	2	P	1
C	2	P	2
C	2	Q	2

$\Pi_{R-S}(r) \times s$. This relation is on schema R

Pairs every tuple in $\Pi_{R-S}(r)$ with every tuple in s.

Example 1

$\Pi_{R-S}(r) \times s$

X1	X2	Y1	Y2
A	1	P	1
A	1	P	2
A	1	Q	2
B	1	P	1
B	1	P	2
B	1	Q	2
C	2	P	1
C	2	P	2
C	2	Q	2

$\Pi_{R-S,S}(r) = \Pi_{X1,X2,Y1,Y2}(r)$

X1	X2	Y1	Y2
A	1	P	3
B	1	Q	2
A	1	P	3
C	2	P	1
A	1	P	1
B	1	Q	2
A	1	P	2
C	2	P	2
A	1	Q	2

$\Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$

$(\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r)$ (gives us those pairs of tuples from $\Pi_{R-S}(r)$ and s that do not appear in r.

$\Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$ If a tuple t_j is in here; hen there is some tuple t_s in s that does not combine with tuple t_j to form a tuple in r.

X1	X2
B	1
C	2

Example 1

$\Pi_{R-S}(r) = \Pi_{X1,X2}(r)$

X1	X2
A	1
B	1
C	2

$\Pi_{R-S}((\Pi_{R-S}(r) \times S) - \Pi_{R-S,S}(r))$

X1	X2
B	1
C	2

$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$

X1	X2
A	1

Example 1

r

X1	Y1	Y2	X2
A	P	3	1
B	**Q**	**2**	1
A	P	3	1
C	**P**	**1**	2
A	**P**	**1**	1
B	**Q**	**2**	1
A	**P**	**2**	1
C	**P**	**2**	2
A	**Q**	**2**	1

S

Y1	Y2
P	1
P	2
Q	2

Example 2: for X1 and X2 of r for all Y1 and Y2 of s

r

X1	Y1	Y2	X2
A	P	3	1
A	P	4	1
B	Q	2	6
A	P	3	1
C	Z	1	2
A	P	1	1
B	Q	4	1
A	P	2	2
A	Q	2	4
A	P	5	1

s

Y1	Y2
P	1
P	2
Q	2

Example 2

r

X1	Y1	Y2	X2
A	P	3	1
A	P	4	1
B	Q	2	6
A	P	3	1
C	Z	1	2
A	P	1	1
B	Q	4	1
A	P	2	2
A	Q	2	4
A	P	5	1

R

X1	Y1	Y2	X2

S

Y1	Y2

R-S

X1	X2

$\Pi_{R-S,S}(r)$

=

$\Pi_{X1,X2,Y1,Y2}(r)$

Example 2

r

X1	Y1	Y2	X2
A	P	3	1
A	P	4	1
B	Q	2	6
A	P	3	1
C	Z	1	2
A	P	1	1
B	Q	4	1
A	P	2	2
A	Q	2	4
A	P	5	1

$\Pi_{R-S,S}(r) = \Pi_{X1,X2,Y1,Y2}(r)$

X1	X2	Y1	Y2
A	1	P	3
A	1	P	4
B	6	Q	2
A	1	P	3
C	2	Z	1
A	1	P	1
B	1	Q	4
A	2	P	2
A	4	Q	2
A	1	P	5

??? $\Pi_{R-S}(r) = \Pi_{X1,X2}(r)$

Example 2

$\Pi_{R-S}(r) = \Pi_{X1,X2}(r)$

X1	X2
A	1
A	1
B	6
A	1
C	2
A	1
B	1
A	2
A	4
A	1

X1	X2
A	1
A	2
A	4
B	1
B	6
C	2

S

Y1	Y2
P	1
P	2
Q	2

??? $\Pi_{R-S}(r) \times s$

Example 2

$\Pi_{R-S}(r) = \Pi_{X1,X2}(r)$

s

X1	X2
A	1
A	2
A	4
B	1
B	6
C	2

Y1	Y2
P	1
P	2
Q	2

X1	X2	Y1	Y2
A	1	P	1
A	1	P	2
A	1	Q	2
A	2	P	1
A	2	P	2
A	2	Q	2
A	4	P	1
A	4	P	2
A	4	Q	2
B	1	P	1
B	1	P	2
B	1	Q	2
B	6	P	1
B	6	P	2
B	6	Q	2
C	2	P	1
C	2	P	2
C	2	Q	2

$\Pi_{R-S}(r) \times s$

??? $\Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$

X1	X2	Y1	Y2
A	1	P	1
A	1	P	2
A	1	Q	2
A	2	P	1
A	2	P	2
A	2	Q	2
A	4	P	1
A	4	P	2
A	4	Q	2
B	1	P	1
B	1	P	2
B	1	Q	2
B	6	P	1
B	6	P	2
B	6	Q	2
C	2	P	1
C	2	P	2
C	2	Q	2

$\Pi_{R-S}(r) \times s$

$\Pi_{R-S,S}(r) = \Pi_{X1,X2,Y1,Y2}(r)$

X1	X2	Y1	Y2
A	1	P	3
A	1	P	4
B	6	Q	2
A	1	P	3
C	2	Z	1
A	1	P	1
B	1	Q	4
A	2	P	2
A	4	Q	2
A	1	P	5

X1	X2	Y1	Y2
A	1	P	1
A	1	P	2
A	1	Q	2
A	2	P	1
A	2	P	2
A	2	Q	2
A	4	P	1
A	4	P	2
A	4	Q	2
B	1	P	1
B	1	P	2
B	1	Q	2
B	6	P	1
B	6	P	2
B	6	Q	2
C	2	P	1
C	2	P	2
C	2	Q	2

$\Pi_{R-S}(r) \times s$

X1	X2
A	1
A	2
A	4
B	1
B	6
C	2

$\Pi_{R-S,S}(r) = \Pi_{X1,X2,Y1,Y2}(r)$

X1	X2	Y1	Y2
A	1	P	3
A	1	P	4
B	6	Q	2
A	1	P	3
C	2	Z	1
A	1	P	1
B	1	Q	4
A	2	P	2
A	4	Q	2
A	1	P	5

$\Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$

Example 2

$\Pi_{R-S}(r) = \Pi_{X1,X2}(r)$

X1	X2
A	1
A	2
A	4
B	1
B	6
C	2

$\Pi_{R-S}((\Pi_{R-S}(r) \times S) - \Pi_{R-S,S}(r))$

X1	X2
A	1
A	2
A	4
B	1
B	6
C	2

$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$

X1	X2

Example 2

r

X1	Y1	Y2	X2
A	P	3	1
A	P	4	1
B	Q	2	6
A	P	3	1
C	Z	1	2
A	P	1	1
B	Q	4	1
A	P	2	2
A	Q	2	4
A	P	5	1

s

Y1	Y2
P	1
P	2
Q	2

r ÷ s is used
when we wish to express
queries with “all”

Ex.
“Which persons have a loyal customer's card at ALL the clothing boutiques in town X?”
“Which persons have a bank account at ALL the banks in the country?”
“Which students are registered on ALL the courses given by Soini?”
“Which students are registered on ALL the courses that are taught in period 1?”
“Which boys are registered on those courses that are taken by ALL the girls?”
“Which girls are registered on ALL the courses taken by student nr. 40101?”

Assignment Operation

- The assignment operation ($\leftarrow$) 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.
- Example: Write $r \div s$ as

$$temp1 \leftarrow \Pi_{R\text{-}S}(r)$$
$$temp2 \leftarrow \Pi_{R\text{-}S}((temp1 \times s) - \Pi_{R\text{-}S,S}(r))$$
$$result = temp1 - temp2$$

  - The result to the right of the $\leftarrow$ is assigned to the relation variable on the left of the $\leftarrow$.
  - May use variable in subsequent expressions.

Bank Example Queries

- Find the names of all customers who have a loan and an account at bank.

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

- Find the name of all customers who have a loan at the bank and the loan amount

$$\Pi_{customer\_name,\ loan\_number,\ amount}(borrower \bowtie loan)$$

Bank Example Queries

- Find all customers who have an account from at least the “Downtown” and the Uptown” branches.
  - Query 1

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Downtown”}}(depositor \bowtie account)) \cap$$

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Uptown”}}(depositor \bowtie account))$$

  - Query 2

$$\Pi_{customer\_name,\ branch\_name}(depositor \bowtie account) \div \rho_{temp(branch\_name)}(\{(\text{“}Downtown\text{”}), (\text{“}Uptown\text{”})\})$$

Note that Query 2 uses a constant relation.

Bank Example Queries

- Find all customers who have an account at all branches located in Brooklyn city.

$$\prod_{customer\_name,\ branch\_name} (depositor \bowtie account)$$
$$\div \prod_{branch\_name} (\sigma_{branch\_city = \text{“Brooklyn”}} (branch))$$

Extended Relational-Algebra-Operations

- Generalized Projection
- Aggregate Functions
- Outer Join

Generalized Projection

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

$$\Pi_{F_1, F_2, \ldots, F_n}(E)$$

- E is any relational-algebra expression
- Each of $F_1, F_2, \ldots, F_n$ are are arithmetic expressions involving constants and attributes in the schema of E.
- Given relation *credit_info(customer_name, limit, credit_balance),* find how much more each person can spend:

$$\Pi_{customer\_name,\ limit - credit\_balance}\ (credit\_info)$$

Aggregate Functions and Operations

- **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 operation** in relational algebra

$$G_1,G_2,\ldots,G_n\ \mathcal{G}_{F_1(A_1),F_2(A_2,\ldots,F_n(A_n)}(E)$$

  E is any relational-algebra expression

  - G_1, G_2 …, G_n is a list of attributes on which to group (can be empty)
  - Each F_i is an aggregate function
  - Each A_i is an attribute name

Aggregate Operation – Example

- Relation *r*:

A	*B*	*C*
α	α	7
α	β	7
β	β	3
β	β	10

- $\mathcal{G}_{\mathbf{sum(c)}}(r)$

sum(*c*)
27

Aggregate Operation – Example

- Relation *account* grouped by *branch-name*:

branch_name	*account_number*	*balance*
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

$$branch\_name\ \mathcal{G}_{\mathbf{sum}(balance)}\ (account)$$

branch_name	**sum**(*balance*)
Perryridge	1300
Brighton	1500
Redwood	700

Aggregate Functions (Cont.)

- Result of aggregation does not have a name
  - Can use rename operation to give it a name
  - For convenience, we permit renaming as part of aggregate operation

$$branch\_name\ \mathcal{G}\ \textbf{sum}(balance)\ \textbf{as}\ sum\_balance\ (account)$$

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 (roughly speaking) **false** by definition.
    - We shall study precise meaning of comparisons with nulls later

Outer Join – Example

- Relation *loan*

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

- Relation *borrower*

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

Outer Join – Example

- Join

  loan $\bowtie$ *borrower*

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

- 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*

Outer Join – Example

- 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

Null Values

- It is possible for tuples to have a null value, denoted by *null*, for some of their attributes
- *null* signifies an unknown value or that a value does not exist.
- The result of any arithmetic expression involving *null* is *null.*
- Aggregate functions simply ignore null values (as in SQL)
- For duplicate elimination and grouping, null is treated like any other value, and two nulls are assumed to be the same (as in SQL)

Null Values

- Comparisons with null values return the special truth value: *unknown*
  -
- **Three-valued logic using the truth value** *unknown*:
  - OR: (*unknown* **or** *true*) = *true*,
    (*unknown* **or** *false*) = *unknown*
    (*unknown* **or** *unknown*) = *unknown*
  - AND: (*true* **and** *unknown*) = *unknown,*
    (*false* **and** *unknown*) = *false,*
    (*unknown* **and** *unknown*) = *unknown*
  - NOT: (**not** *unknown*) = *unknown*
  - In SQL "*P* **is unknown**" evaluates to true if predicate *P* evaluates to *unknown*
- Result of select predicate is treated as *false* if it evaluates to *unknown*

Modification of the Database

- The content of the database may be modified using the following operations:
  - Deletion
  - Insertion
  - Updating
- All these operations are expressed using the assignment operator.

Deletion

- A delete request is expressed similarly to a query, except instead of displaying tuples to the user, the selected tuples are removed from the database.
- Can delete only whole tuples; cannot delete values on only particular attributes
- A deletion is expressed in relational algebra by:

$$r \leftarrow r - E$$

where r is a relation and E is a relational algebra query.

Deletion Examples

- Delete all account records in the Perryridge branch.

$$account \leftarrow account - \sigma_{branch\_name = \text{“Perryridge”}}(account)$$

- Delete all loan records with amount in the range of 0 to 50

$$loan \leftarrow loan - \sigma_{amount \geq 0 \text{ and } amount \leq 50}(loan)$$

Insertion

- To insert data into a relation, we either:
  - specify a tuple to be inserted
  - write a query whose result is a set of tuples to be inserted
- in relational algebra, an insertion is expressed by:

$$r \leftarrow r \cup E$$

  where r is a relation and E is a relational algebra expression.
- The insertion of a single tuple is expressed by letting E be a constant relation containing one tuple.

Insertion Examples

- Insert information in the database specifying that Smith has $1200 in account A-973 at the Perryridge branch.

$$account \leftarrow account \cup \{(\text{“A-973”}, \text{“Perryridge”}, 1200)\}$$

$$depositor \leftarrow depositor \cup \{(\text{“Smith”}, \text{“A-973”})\}$$

⋈

Updating

- A mechanism to change a value in a tuple without charging *all* values in the tuple
- Use the generalized projection operator to do this task

$$r \leftarrow \prod_{F_1, F_2, \ldots, F_l} (r)$$

- Each F_i is either
  - the I^{th} attribute of r, if the I^{th} attribute is not updated, or,
  - if the attribute is to be updated F_i is an expression, involving only constants and the attributes of r, which gives the new value for the attribute

Update Examples

- Make interest payments by increasing all balances by 5 percent.

$$account \leftarrow \prod_{account\_number,\ branch\_name,\ balance * 1.05} (account)$$

Tuple Relational Calculus (TRC) in DBMS

- Tuple Relational Calculus is a **non-procedural query language** unlike relational algebra.
- Tuple Calculus provides only the description of the query but it does not provide the methods to solve it.
- Thus, it explains what to do but not how to do.

Source: https://www.geeksforgeeks.org/

Tuple Calculus

A query is expressed as

{t| P(t)} , Where
t = resulting tuples,
P(t) = known as Predicate
These are the conditions that are used to fetch t

Thus, it generates set of all tuples t
such that Predicate P(t) is true for t.

P(t) may have various conditions logically combined with
OR (∨), AND (∧), NOT(¬).`

Table-1: Customer

Customer name	Street	City
Saurabh	A7	Patiala
Mehak	B6	Jalandhar
Sumiti	D9	Ludhiana
Ria	A5	Patiala

Table-2: Branch

Branch name	Branch city
ABC	Patiala
DEF	Ludhiana
GHI	Jalandhar

Table-3: Account

Account number	Branch name	Balance
1111	ABC	50000
1112	DEF	10000
1113	GHI	9000
1114	ABC	7000

Table-4: Loan

Loan number	Branch name	Amount
L33	ABC	10000
L35	DEF	15000
L49	GHI	9000
L98	DEF	65000

Table-5: Borrower

Customer name	Loan number
Saurabh	L33
Mehak	L49
Ria	L98

Table-6: Depositor

Customer name	Account number
Saurabh	1111
Mehak	1113
Sumiti	1114

Queries-1

: Find the loan number, branch, amount of loans of greater than or equal to 10000 amount.

$\{t \mid t \in loan \wedge t[amount] >= 10000\}$

Loan number	Branch name	Amount
L33	ABC	10000
L35	DEF	15000
L98	DEF	65000

Queries-2

Find the loan number for each loan of an amount greater or equal to 10000.

$\{t \mid \exists s \in loan(t[loan\ number] = s[loan\ number] \wedge s[amount] >= 10000)\}$

Loan number
L33
L35
L98

Queries-3

Find the names of all customers who have a loan and an account at the bank.

$\{t \mid \exists\ s \in borrower(\ t[customer\text{-}name] = s[customer\text{-}name]) \wedge \exists\ u \in depositor(\ t[customer\text{-}name] = u[customer\text{-}name])$

Customer name
Saurabh
Mehak

Queries-4

Find the names of all customers having a loan at the “ABC” branch.

$\{t \mid \exists\ s \in borrower(t[customer\text{-}name] = s[customer\text{-}name] \wedge \exists\ u \in loan(u[branch\text{-}name] = \text{“ABC”} \wedge u[loan\text{-}number] = s[loan\text{-}number]))\}$

Customer name
Saurabh

Domain Relational Calculus in DBMS

- **Domain Relational Calculus** is a non-procedural query language equivalent in power to Tuple Relational Calculus.

- Domain Relational Calculus provides only the description of the query but it does not provide the methods to solve it. In Domain Relational Calculus, a query is expressed as,

Query-1:
Find the loan number, branch, amount of loans of greater than or equal to 100 amount.

$$\{\prec l, b, a\succ \mid \prec l, b, a\succ \in \text{loan} \wedge (a \geq 100)\}$$

Loan number	Branch name	Amount
L01	Main	200
L03	Main	150

Query-2:
Find the loan number for each loan of an amount greater or equal to 150.

$\{\langle l \rangle \mid \exists\ b, a\ (\langle l, b, a \rangle \in loan \wedge (a \geq 150)\}$

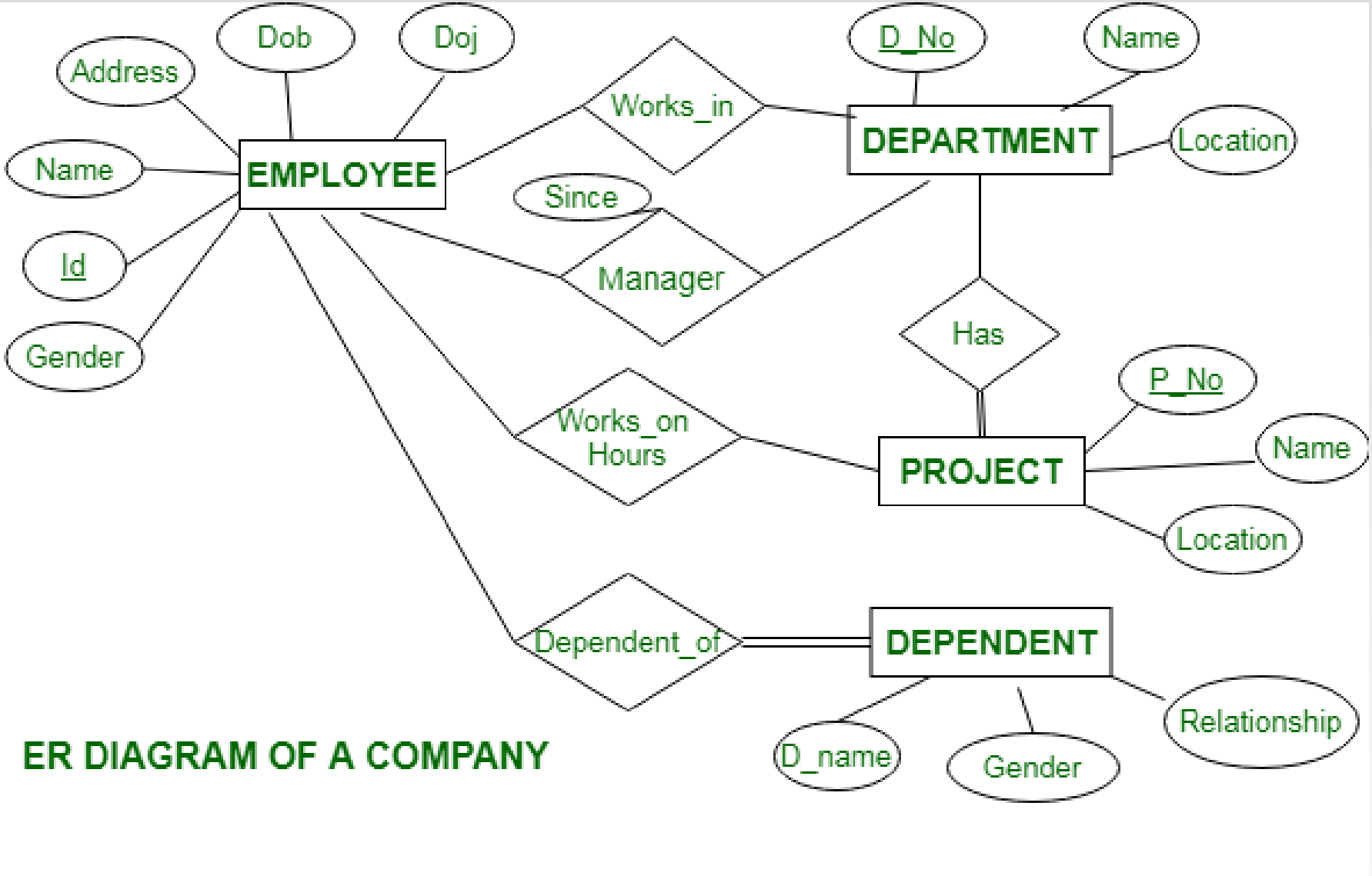
Loan number
L01
L03

Query-3

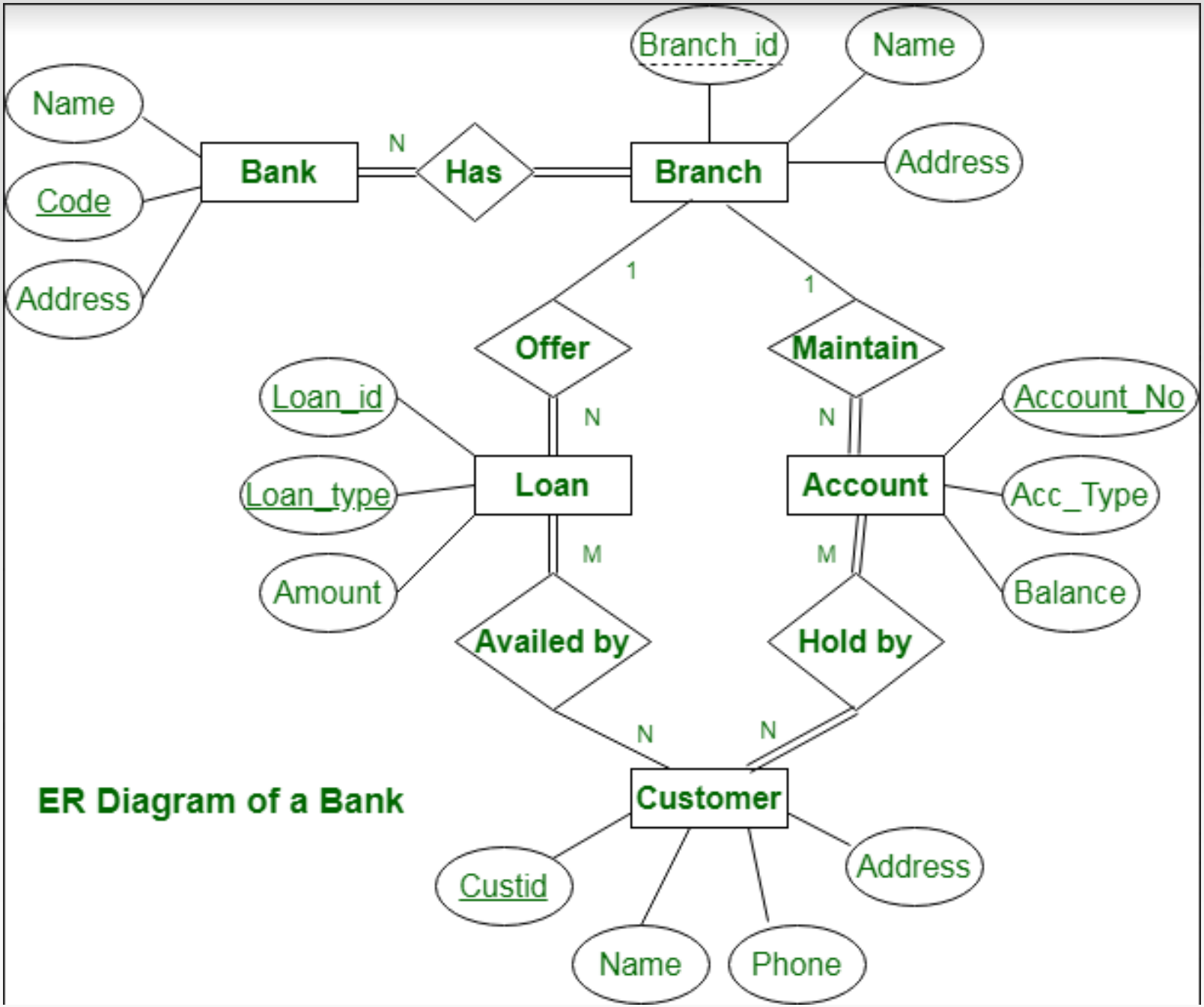
Find the names of all customers having a loan at the “Main” branch and find the loan amount .

$\{\prec c, a\succ \mid \exists\ l\ (\prec c, l\succ \in \text{borrower} \wedge \exists\ b\ (\prec l, b, a\succ \in \text{loan} \wedge (b = \text{“Main”})))$

Customer Name	Amount
Ritu	200
Debomit	60
Soumya	150



ER DIAGRAM OF A COMPANY



ER Diagram of a Bank