Lecture 3: Additional, Extended Relational Algebra Operators and Modification Operations

CSX3006 DATABASE SYSTEMS

ITX3006 DATABASE MANAGEMENT SYSTEMS

Outline

- Additional Operators
- Extended Operators
- Modification Operations

Relational Algebra Operators - 1

- **Fundamental Operators**
  - select: σ
  - project: $\prod$
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ

Relational Algebra Operators - 2

- **Additional Operators**
  - **set intersection: $\cap$**
  - **natural join: $\bowtie$**
  - **division: $\div$**
  - **assignment: $\leftarrow$**
- **Extended Operators**
  - Generalized Project
  - Aggregate Functions
  - Outer Joins
- **Modification Operations**
  - Deletion, insertion and updating of tuples

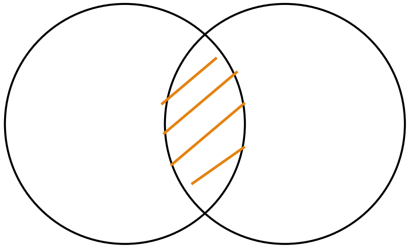
Additional Operations

Goal: simplify common queries

- **Set intersection**
- **Natural join**
  - More general form of join known as **theta join** and **equijoin**
- **Division**
- **Assignment**

Set-Intersection Operation - 1

- Notation: $r \cap s$
  - Binary Operator
  - The usual Set Intersect Operation
  - Produces a new relation containing tuples that are present in **both** *r* and *s*
  - $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$
- Assume:
  - *r*, *s* have the *same* ***arity***; (same number of attributes)
  - Domains of the attributes of *r* and *s* are compatible



- Note: $r \cap s = r - (r - s)$

Set-Intersection Operation - 2

- $\Pi_{customer\_name}$ *(depositor)* $\cap$ $\Pi_{customer\_name}$ *(borrower)*
  - What does the above relation algebra expression find?

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

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

Set-Intersection Operation - 3

- $\Pi_{customer\_name}$ *(depositor)* $\cap$ $\Pi_{customer\_name}$ *(borrower)*
  - What does the above relation algebra expression find?

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

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

customer_name
Hayes
Jones
Smith

Set-Intersection Operation – Example

- Relation *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

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

Natural-Join Operation - 1

- Notation: $r \bowtie s$
- Combine a Cartesian product **and** a selection

- Example 1: find the *names* of all customers who have a loan at the bank, along with the *loan number*, *branch* and the *loan amount*.

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 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

loan relation

Natural-Join Operation - 2

- Find the *names* of all customers who have a loan at the bank, along with the *loan number, branch* and the *loan amount*.

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 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

loan relation

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

- Result relation's schema:
  - (borrower.customer_name, borrower.loan_number, loan.loan_number, loan.branch_name, loan.amount)

- Find the names of all customers who have a loan at the bank, along with the loan number, branch and the loan amount.

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

borrower.customer_name	borrower.loan_number	loan.loan_number	loan.branch_name	loan.amount
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**	Round Hill	900
Smith	**L-23**	**L-23**	Redwood	2000
Williams	**L-17**	**L-17**	Downtown	1000

- borrower $\bowtie$ loan

customer_name	**loan_number**	branch_name	amount
Adams	**L-16**	Perryridge	1300
Curry	**L-93**	Mianus	500
Hayes	**L-15**	Perryridge	1500
Jackson	**L-14**	Downtown	1500
Jones	**L-17**	Downtown	1000
Smith	**L-11**	Round Hill	900
Smith	**L-23**	Redwood	2000
Williams	**L-17**	Downtown	1000

How to Generate Result of $\bowtie$? - 1

- Let *r* and *s* be relations on schemas *R* and *S,* respectively.
- 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 the **<u>common attributes</u>** in *R* and *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*

How to Generate Result of $\bowtie$? - 2

- Let *r* and *s* be relations on schemas *R* and *S,* respectively.
- 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 the **common attributes** in *R* and *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*

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 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

loan relation

Natural-Join Operation: Example 2

- 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	$\in$

s

- r ⋈ s

Natural-Join Operation: Example 2

- Relations r, s:

r

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

s

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	$\in$

- 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

Natural-Join Operation: Example 3

- Find the **name** of all customers who have an account with the bank, along with his/her **account number** and the **balance** of the account.

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

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

Natural-Join Operation: Example 3

- Find the **name** of all customers who have an account with the bank, along with his/her **account number** and the **balance** of the account.

***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

***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

Step 1: depositor $\bowtie$ account

Result of Step 1

customer_name	account_number	branch_name	balance
Johnson	A-101	Downtown	500
Hayes	A-102	Perryridge	400
Johnson	A-201	Brighton	900
Smith	A-215	Mianus	700
Jones	A-217	Brighton	750
Lindsay	A-222	Redwood	700
Turner	A-305	Round Hill	350

Natural-Join Operation: Example 3

- Find the **name** of all customers who have an account with the bank, along with his/her **account number** and the **balance** of the account.

  - Step 2: $\Pi_{\text{customer\_name, account\_number, balance}}$ (depositor ⋈ account)

Result of Step 1

customer_name	account_number	Branch_name	balance
Johnson	A-101	Downtown	500
Hayes	A-102	Perryridge	400
Johnson	A-201	Brighton	900
Smith	A-215	Mianus	700
Jones	A-217	Brighton	750
Lindsay	A-222	Redwood	700
Turner	A-305	Round Hill	350

Result of Step 2

customer_name	account_number	balance
Johnson	A-101	500
Hayes	A-102	400
Johnson	A-201	900
Smith	A-215	700
Jones	A-217	750
Lindsay	A-222	700
Turner	A-305	350

Natural-Join Operation: Example 4

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

***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

***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

***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

Natural-Join Operation: Example 4

- Find the **names of all branches** with customers who <u>have an **account** in the bank</u> and who <u>live in the **city** of Harrison</u>

***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

***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

***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

- Check retrieved fields to **identify associated relations**

Natural-Join Operation: Example 4

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

***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

***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

***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

- **Join relations** to obtain needed attributes
- (customer ⋈ depositor ⋈ account)

- (customer ⋈ depositor ⋈ account)

customer_name	customer_street	customer_city	account_number	branch_name	balance
Hayes	Main	Harrison	A-102	Perryridge	400
Johnson	Alma	Palo Alto	A-101	Downtown	500
Johnson	Alma	Palo Alto	A-201	Brighton	900
Jones	Main	Harrison	A-217	Brighton	750
Lindsay	Park	Pittsfield	A-222	Redwood	700
Smith	North	Rye	A-215	Mianus	700
Turner	Putnam	Stamford	A-305	Round Hill	350

Natural-Join Operation: Example 4

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

customer_ name	customer_ street	customer _city	account_ number	branch_name	balance
Hayes	Main	Harrison	A-102	Perryridge	400
Johnson	Alma	Palo Alto	A-101	Downtown	500
Johnson	Alma	Palo Alto	A-201	Brighton	900
Jones	Main	Harrison	A-217	Brighton	750
Lindsay	Park	Pittsfield	A-222	Redwood	700
Smith	North	Rye	A-215	Mianus	700
Turner	Putnam	Stamford	A-305	Round Hill	350

- **Select only records that satisfied the condition.**
- $\sigma_{customer\_city\ =\ \text{“Harrison”}}\ (\text{customer} \bowtie \text{depositor} \bowtie \text{account})$

- $\sigma_{customer\_city\ =\ \text{“Harrison”}}$ (customer ⋈ depositor ⋈ account)

customer_ name	customer_ street	customer _city	account_ number	branch_name	balance
Hayes	Main	Harrison	A-102	Perryridge	400
Jones	Main	Harrison	A-217	Brighton	750

Natural-Join Operation: Example 4

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

customer_ name	customer_ street	customer _city	account_ number	branch_name	balance
Hayes	Main	Harrison	A-102	Perryridge	400
Jones	Main	Harrison	A-217	Brighton	750

- **Project only needed attributes.**

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

Branch_name
Perryridge
Brighton

Natural-Join Operation: Example 4

- Find the names of all branches with customers who have an account in the bank and who live in the city of Harrison
  - There are many ways to write the relational algebra expression to produce the same result.
  - E.g.,
- $\Pi_{branch\_name}\ (\sigma_{customer\_city\ =\ \text{“Harrison”}}\ (customer \bowtie depositor \bowtie account))$
- $\Pi_{branch\_name}\ (\sigma_{customer\_city\ =\ \text{“Harrison”}}\ ((customer \bowtie depositor) \bowtie account))$
- $\Pi_{branch\_name}\ (\sigma_{customer\_city\ =\ \text{“Harrison”}}\ (customer \bowtie (depositor \bowtie account)))$

Branch_name
Perryridge
Brighton

Natural-Join Operation: Example 5

- Find all customers who have **both** a loan and an account at the bank

***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

***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

Natural-Join Operation: Example 5

- Find all customers who have **both** a loan and an account at the bank

  - $\Pi_{customer\_name}$ (depositor $\bowtie$ borrower)

  or

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

customer_name
Hayes
Jones
Smith

Natural-Join Operation: Example 6

- Find all account numbers managed by any of branches in the city of Horseneck.

***branch* relation**

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

***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

Natural-Join Operation: Example 6

- Find all account numbers managed by any of branches in the city of Horseneck.

- $\Pi_{account\_number}\ (\ \sigma_{branch\_city\ =\ \text{“Horseneck”}}(branch) \bowtie account\ )$

***branch* relation**

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

***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_number
A-102
A-215
A-305

Theta Join Operator

- **Theta Join (Condition Join):** More general form of join operation
  - $r1 \bowtie_{p} r2$ is equivalent to $\sigma_{p}(r1 \times r2)$

    where p is a formula in propositional calculus consisting of terms connected by :

    $\wedge$ (and), $\vee$ (or), $\neg$ (not)

    Each term is one of: <attribute> *op* <attribute> or <constant>

    where *op* is one of: $\boxed{=,\ \neq,\ >,\ \geq,\ <,\ \leq}$

Theta Join Operator – Example 1

- Find the name of all customers who have an account with the bank, along with his/her account number and the balance of the account.

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

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

Theta Join Operator – Example 1

- Find the name of all customers who have an account with the bank, along with his/her account number and the balance of the account.

***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

***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

Theta Join Operator – Example 1

- Find the name of all customers who have an account with the bank, along with his/her account number and the balance of the account.
  - $\Pi_{\text{customer\_name, depositor.account\_number, balance}}$ ($\sigma_{\text{depositor.account\_number = account.account\_number}}$ (depositor x account))
  - $\Pi_{\text{customer\_name, account\_number, balance}}$ (depositor $\underbrace{\bowtie}_{}$ account))

    Natural Join
    (Implicitly use common attributes to join)
  - $\Pi_{\text{customer\_name, depositor.account\_number, balance}}$ (depositor $\underbrace{\bowtie_{\text{depositor.account\_number = account.account\_number}}}_{}$ account))

    Theta Join (must specify a join condition)

theta join	natural join
any attribute in common are **repeated** e.g.) account_number	duplicate attributes are **removed**

Theta Join Operator – Example 2

- Find the customer names having loans and the loan amounts if the value of loan is more than 1000.

***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

***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

Theta Join Operator – Example 2

- Find the **customer names** having loans and the loan amounts if the value of loan is more than 1000.

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

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

Theta Join Operator – Example 2

- Find the customer names having loans and the loan amounts if the value of loan is more than 1000.

$$\Pi_{\text{customer\_name, amount}} ($$

$$\text{borrower} \bowtie_{\text{borrower.loan\_number = loan.loan\_number} \wedge \text{amount} > 1000} \text{loan})$$

***borrower* relation**

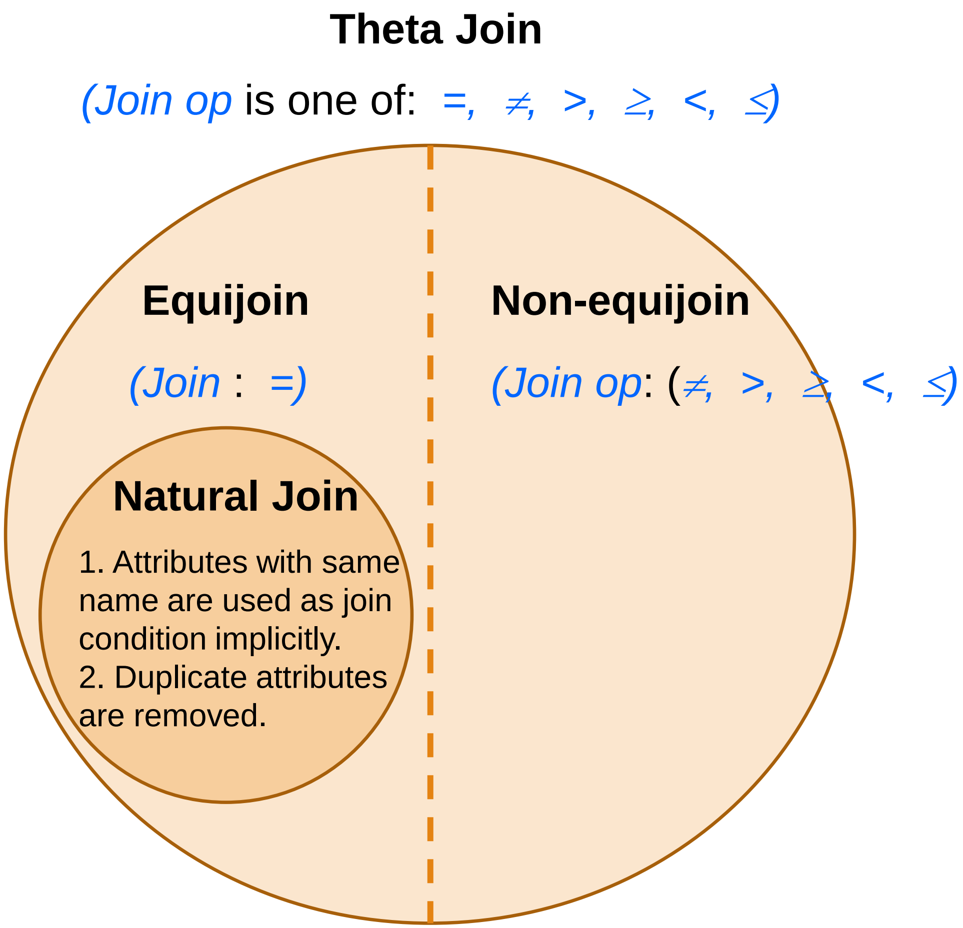
customer_name	*loan_number*
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

***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	**amount**
Adams	1300
Hayes	1500
Jackson	1500
Smith	2000

Theta Join, Natural join and equijoin



Division Operation

- Notation: $r \div s$
  - Suited to queries that include the phrase "**for all**".

Division Operation – Example 1

- 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

"Retrieve any value in A that relates to all values in B. "

Division Operation – Example 2

- 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

"Retrieve any combination of A,B,C that relates to all values in D,E. "

Division Operation – Example 3

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

***branch* relation**

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

***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

***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

Division Operation – Example 3

- Find the **names of customers** who have an account at all the branches located in the city of Brooklyn.

branch **relation**

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

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

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

Division Operation – Example 3

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

1. Get all branches in the city of Brooklyn (r1)
2. Find the names of all customers and branches where the customers have their accounts (r2)
3. Find customers who appear in r2 **with every branch name in r1** ($r2 \div r1$)

Division Operation – Example 3

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

1. Get all branches in the city of Brooklyn

$$r1 \leftarrow \Pi_{branch\_name}((\sigma_{branch\_city\ =\ \text{“Brooklyn”}}(branch))$$

***branch* relation**

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

r1

branch_name
Brighton
Downtown

Division Operation – Example 3

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

2. Find the names of all customers and branches where the customers have their accounts

$$r2 \leftarrow \Pi_{customer\_name,\ branch\_name}\ (depositor \bowtie account)$$

***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

***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

r2

customer_ name	branch_name
Hayes	Perryridge
Johnson	Brighton
Johnson	Downtown
Jones	Brighton
Lindsay	Redwood
Smith	Mianus
Turner	Round Hill

Division Operation – Example 1

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

3. Find customers who appear in r2 **with every branch name** in r1

   (r2 ÷ r1) :

$$\Pi_{customer\_name,\ branch\_name}\ (depositor \bowtie account)\ \div$$

$$\Pi_{branch\_name}\ ((\sigma_{branch\_city\ =\ \text{“Brooklyn”}}\ (branch))$$

r2

customer_ name	branch_name
Hayes	Perryridge
Johnson	**Brighton**
Johnson	**Downtown**
Jones	Brighton
Lindsay	Redwood
Smith	Mianus
Turner	Round Hill

r1

branch_name
Brighton
Downtown

r2 ÷ r1

customer_name
Johnson

Division Operation

- Notation $r \div s$
- 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 $r \div s$ is a relation on schema $R - S = (A_1, \ldots, A_m)$

$$r \div s = \{ t \mid t \in \prod_{R-S}(r) \wedge \forall u \in s \, (\boldsymbol{tu} \in r) \}$$

where ***tu*** means the **concatenation** of the tuple ***t*** and the tuple ***u***

to produce a single tuple ***tu***

Division Operation – Example 4

- Find all customers who have an account from ***at least*** the "Downtown" ***and*** the "Uptown" branches.
    - Must have at least one account in "Downtown" branch and at least one account in "Uptown" branch; but may or may not have accounts in other branches as well

Division Operation – Example 4

- Find all customers who have an account from ***at least*** the "Downtown" ***and*** the "Uptown" branches.
  - Must have **at least one** account in "Downtown" branch **and at least one** account in "Uptown" branch; but may or may not have accounts in other branches as well

- Answer 1

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

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

Division Operation – Example 4

- Find all customers who have an account from ***at least*** the "Downtown" ***and*** the "Uptown" branches.
  - Must have at least one account in "Downtown" branch and at least one account in "Uptown" branch; but may or may not have accounts in other branches as well

- Answer 2

$$\Pi_{customer\_name,\ branch\_name}(depositor \bowtie account) \div \rho_{temp(branch\_name)}\left(\underbrace{\{(\text{"Downtown"}),\ (\text{"Uptown"})\}}_{\textbf{constant relation}}\right)$$

A Constant Relation

- Fixed set of tuples
- E.g.,
  - { (1, 2), (1, 3), (2, 3) }
  - { ("Downtown"), ("Uptown") }

- Use ρ to rename the relation (and attributes)
  - $\rho_{temp(branch\_name)}$ ({ ("Downtown"), ("Uptown") })

Assignment Operation

- The assignment operator (←) 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 Operation – Example 1

(Revised Example 3 of Division Operator)

- Find the names of all customers who have an account at all the branches located in the city of Brooklyn. [2]

[3]

***branch* relation**

branch_name	*branch_city*	*assets*
Brighton [1]	**Brooklyn**	7100000
Downtown	**Brooklyn**	9000000
Mianus	Horseneck	400000
Mprtj Tpwm	Rye	3700000
Perryridge	Horseneck	1700000
Pownal	Bennington	300000
Redwood	Palo Alto	2100000
Round Hill	Horseneck	8000000

***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

***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

Assignment Operation – Example 1

(Revised Example 3 of Division Operator)

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

[1] $temp1 \leftarrow \Pi_{branch\_name}((\sigma_{branch\_city\ =\ \text{“Brooklyn”}}\ (branch))$

[2] $temp2 \leftarrow \Pi_{customer\_name,\ branch\_name}\ (depositor \bowtie account)$

[3] $result \leftarrow temp2 \div temp1$

customer_name
Johnson

Assignment Operation – Example 2

- Find the names of customers who have one or more bank accounts in branches that are NOT in the same city as the customer is living in.

[2] [3] [1]

***branch* relation**

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

***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

***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* relation**

customer_name	*customer_street*	***customer_city***
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
…		

Assignment Operation – Example 2

- Find the names of customers who have one or more bank accounts in branches that are NOT in the same city as the customer is living in.

[1] cust_info $\leftarrow \Pi_{customer\_name,\ account\_number,\ customer\_city}$ (depositor $\bowtie$ customer)

[2] account_info $\leftarrow \Pi_{account\_number,\ branch\_city}$ (account $\bowtie$ branch)

/* Note that the natural join of account and branch's implied join condition is

account.branch_name = branch.branch_name*/

[3] result $\leftarrow \sigma_{branch\_city\ \neq\ customer\_city}$ (cust_info $\bowtie$ account_info)

Relational Algebra Operators

- **Additional Operators**
  - set intersection: $\cap$
  - natural join: $\bowtie$
  - division: $\div$
  - assignment: $\leftarrow$
- **Extended Operators**
  - **Generalized Project**
  - **Aggregate Functions**
  - **Outer Joins**
- **Modification Operations**
  - Deletion, insertion and updating of tuples

Generalized Projection

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

$$\Pi_{F1,\ F2,\ldots,\ Fn}\ (E)$$

  - E is any relational-algebra expression
  - Each of F_1, F_2, …, F_n is ***arithmetic expression*** involving **constants and attributes** in the schema of E.

More Ref: http://www.engineering-bachelors-degree.com/database-software/uncategorized/extended-relational-algebra-operations/

Generalized Projection Example

- Given relation *credit_info(customer_name, limit, credit_balance),*
  - find *how much more each person can spend*

customer_name	*limit*	*credit_balance*
Curry	2000	1750
Hayes	1500	1500
Jones	6000	700
Smith	2000	400

credit_info relation

customer_name	**credit_available**
Curry	250
Hayes	0
Jones	5300
Smith	1600

Generalized Projection Example

- Given relation *credit_info (customer_name, limit, credit_balance),*
  - find *how much more each person can spend*

customer_name	limit	credit_balance
Curry	2000	1750
Hayes	1500	1500
Jones	6000	700
Smith	2000	400

credit_info relation

customer_name	credit_available
Curry	250
Hayes	0
Jones	5300
Smith	1600

- Answer 1: $\Pi_{customer\_name,\ \mathbf{limit - credit\_balance\ as\ credit\_available}}$ (credit_info)
- Answer 2: $\rho_{credit\_report(customer\_name,\ credit\_available}$ ($\Pi_{customer\_name,\ limit - credit\_balance}$ (credit_info))

Aggregate Functions and Grouping - 1

- **Aggregation function** ($\mathcal{G}$) 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

  $\mathcal{G}$ pronounce as 'calligraphic G'

  - **Definition**

$$\underbrace{G_1 G_2, \ldots, G_n}_{\text{grouping attributes}} \mathcal{G} \underbrace{F_1(A_1), F_2(A_2), \ldots F_m(A_m)}_{\text{aggregate functions}} (E)$$

aggregated attributes

relational-algebra expression

Aggregate Functions and Grouping - 2

- **Definition**

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

  - E is any relational-algebra expression
  - G_1, G_2 …, G_n is a list of attributes on which to group (***can be empty***) *(grouping attribute)*
    - When it is empty, every tuple is made into ***a single group***
  - F_i is an aggregate function
  - A_i is an attribute name on which the aggregation is made *(aggregated attribute)*

Aggregate Operation Example - 1

- Relation *r*:

a	b	c
α	α	7
α	β	4
β	β	3
β	β	10

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

sum(c)
24

- $\mathcal{G}_{\textbf{avg(c)}}(\mathrm{r})$

avg(c)
6

- $\mathcal{G}_{\textbf{max(c)}}(\mathrm{r})$

max(c)
10

- $\mathcal{G}_{\textbf{min(c)}}(\mathrm{r})$

min(c)
3

Aggregate Operation Example - 2

- Relation *r*:

a	b	c
α	α	7
α	β	4
β	β	3
β	β	10

aggregate functions

- $\mathcal{G}_{\textbf{count(c)}}(r)$
- $\mathcal{G}_{\textbf{count(b)}}(r)$
- $\mathcal{G}_{\textbf{count-distinct(b)}}(r)$
- $\mathcal{G}_{\textbf{count-distinct(a)}}(r)$

aggregated attributes

count(c)
4

count(b)
4

count-distinct(b)
2

count-distinct(a)
2

The Effect of the Grouping Attribute G_i

- *The tuples* in the result of expression *E* are *partitioned into groups* such that
  1. All tuples **in a group** have the **same values** for $G_1, G_2, \ldots, G_n$.
  2. Tuples **in different groups** have **different values** for $G_1, G_2, \ldots, G_n$.

Aggregate Operation Example - 3

- Relation *r*:

a	b	c
α	α	7
α	β	4
β	β	3
β	β	10

Grouped by b

Grouped by a

- ${}_{b}\mathcal{G}_{\text{sum(c)}}(r)$
- ${}_{a}\mathcal{G}_{\text{sum(c)}}(r)$

grouping attributes

b	sum(c)
α	7
β	17

a	sum(c)
α	11
β	13

Aggregate Operation Example - 4

- Relation *r*:

a	b	c
α	α	7
α	β	4
β	β	3
β	β	10

Grouped by a,b

Grouped by a

- $_{a}\mathcal{G}_{\text{sum(c),max(c),min(c)}}(r)$

a	sum(c)	max(c)	min(c)
α	11	7	4
β	13	10	3

- $_{a,b}\mathcal{G}_{\text{sum(c),max(c),min(c)}}(r)$

a	b	sum(c)	max(c)	min(c)
α	α	7	7	7
α	β	4	4	4
β	β	13	10	3

Aggregate Operation Example - 5

- Find the **total balance** of all the accounts **at each branch location**

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

account relation

Aggregate Operation Example - 5

- Find the **total balance** of all the accounts **at each branch location**

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

account relation

Solution:

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

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

Aggregate Operation Example - 6

- Find the **total balance** of all the accounts **at each branch location**

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

account relation

Also rename aggregated attribute:

- $\rho_{report\ (branch\_name,\ branch\_total)}\ (\ _{branch\_name}\mathcal{G}_{\ \mathrm{sum}(balance)}\ (account)\ )$

or

- $_{branch\_name}\mathcal{G}_{\ \mathrm{sum}(balance)\ \mathrm{as}\ branch\_total}\ (account)$

branch_name	**branch_total**
Perryridge	1300
Brighton	1500
Redwood	700

Aggregate Operation Example - 7

Find the branches with **highest *average*** account balance.

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

account relation

Aggregate Operation Example - 7

Find the branches with **highest *average*** account balance.

- temp1 ← $_{branch\_name}\ \mathcal{G}\ _{\text{avg}(balance)\text{ as branch\_average}}\ (account)$

branch_name	**branch_average**
Perryridge	650
Brighton	750
Redwood	700

- temp2 ← $\mathcal{G}\ _{\text{max(branch\_average) as highest\_average}}\ (temp1)$

highest_average
750

- $\Pi_{\text{branch\_name}}(\sigma_{\text{branch\_average = highest\_average}}(\text{temp1 x temp2}))$

branch_name
Brighton

Outer Join - 1

- **Avoids "loss of information"** by

  "**adding tuples** from one relation that **does not match join condition** to be included in the output relation"

  - **Left Outer Join** ⟕ : includes **all tuples in the left** hand relation and only those matching tuples from the right hand relation
  - **Right Outer Join** ⟖ : includes **all tuples in the right** hand relation and only those matching tuples from the left hand relation
  - **Full Outer Join** ⟗ : includes **all tuples in the left** hand relation **and in the right** hand relation

Outer Join - 2

- When generating output relation, missing information is filled with null
  - *null* signifies that the value is **unknown** or **does not exist**
  - **All comparisons involving *null*** are **false** by definition.

Outer Join – Motivation

Name	Age	Food
Jenny	33	Burger
Donna	22	Pizza
Roy	21	Steak ?
Sara	34	Pasta

member relation

Food	Day
Pizza	Monday
Burger	Tuesday
Salad	Wednesday
Pasta	Thursday
Tacos	Friday

menu relation

What's the result of member $\bowtie$ menu ? (Natural Join)

Name	Age	Food	Day
Jenny	33	Burger	Tuesday
Donna	22	Pizza	Monday
Sara	34	Pasta	Thursday

What information have we lost as the result of the natural join?

Left Outer Join – Example

Name	Age	Food
Jenny	33	Burger
Donna	22	Pizza
Roy	21	Steak
Sara	34	Pasta

member relation

Food	Day
Pizza	Monday
Burger	Tuesday
Salad	Wednesday
Pasta	Thursday
Tacos	Friday

menu relation

- Left Outer Join

  member ⟕ menu

Name	Age	Food	day
Jenny	33	Burger	Tuesday
Donna	22	Pizza	Monday
Sara	34	Pasta	Thursday
Roy	21	Steak	**null**

Right Outer Join - Example

Name	Age	Food
Jenny	33	Burger
Donna	22	Pizza
Roy	21	Steak
Sara	34	Pasta

member relation

Food	Day
Pizza	Monday
Burger	Tuesday
Salad	Wednesday
Pasta	Thursday
Tacos	Friday

menu relation

- Right Outer Join

  member ⟖ menu

Name	Age	Food	day
Jenny	33	Burger	Tuesday
Donna	22	Pizza	Monday
Sara	34	Pasta	Thursday
null	**null**	Salad	Wednesday
null	**null**	Tacos	Friday

Full Outer Join – Example

Name	Age	Food
Jenny	33	Burger
Donna	22	Pizza
Roy	21	Steak
Sara	34	Pasta

member relation

Food	Day
Pizza	Monday
Burger	Tuesday
Salad	Wednesday
Pasta	Thursday
Tacos	Friday

menu relation

- Full Outer Join

member ⟗ menu

Name	Age	Food	day
Jenny	33	Burger	Tuesday
Donna	22	Pizza	Monday
Sara	34	Pasta	Thursday
Roy	21	Steak	**null**
null	**null**	Salad	Wednesday
null	**null**	Tacos	Friday

Effect of null in Arithmetic Operations and Predicate Logic

- The **result** of any **arithmetic expression** (+, -, *, /) **involving *null*** is ***null.***

  null + 3 = null   *null / null = null*

- The **result** of any **comparisons** (=, !=, <, <=, >, >=) **involving null** is **<u>unknown.</u>**

  (null < 500) → unknown   (null = null) → unknown

- **Three-valued logic** using the truth value unknown:
  - **OR**:

a	b	a OR b
Unknown	True	True
Unknown	False	Unknown
Unknown	Unknown	Unknown

  AND:

a	b	a AND b
Unknown	True	Unknown
Unknown	False	False
Unknown	Unknown	Unknown

  - **NOT***:* (not *unknown*) = *unknown*

How Do Relational Operations Deal with Null Values? - 1

- Select: $\sigma_P(E)$
  - If P returns **unknown** → t is **NOT ADDED** TO THE RESULT.

Select Operation with Null Values - Example

Name	Age	Food
Jenny	33	**null**
Donna	**null**	Pizza
Roy	21	Steak
Sara	34	**null**

member relation

- $\sigma_{age\ >\ 25}$ (member)

Name	Age	Food
Jenny	33	**null**
Sara	34	**null**

- $\sigma_{age\ >\ 25\ \vee\ food\ =\ \text{“Pizza”}}$ (member)

Name	Age	Food
Jenny	33	**null**
Donna	null	Pizza
Sara	34	**null**

How Do Relational Operations Deal with Null Values? - 2

- Natural Join:
  - If at least one of the two tuples have a null value in a common attribute → the tuples do not match.

Natural Join Operation with Null Values - Example

Name	Age	**Food**
Jenny	33	**null**
Donna	null	Pizza
Roy	21	Steak
Sara	34	**null**

member relation

Food	Day
Pizza	Monday
Burger	Tuesday
Salad	Wednesday
Pasta	Thursday
null	Friday

menu relation

- member ⋈ menu

Name	Age	Food	Day
Donna	**null**	Pizza	Monday

How Do Relational Operations Deal with Null Values? - 3

- Projection:
  - If two tuple in the projection result are **exactly the same** and **both have nulls in the same fields** → They are treated as **duplicates**.

Project Operation with Null Values - Example

Name	Age	Food
Jenny	33	**null**
Donna	null	Pizza
Roy	21	Steak
Sara	34	**null**

member relation

- Π_{food} (member)

Food
null
Pizza
Steak

How Do Relational Operations Deal with Null Values? - 2

- Union, intersection, difference, Generalized projection:
  - Same as projection
- Aggregate functions $_{G_1 G_2, \dots, G_n} \mathcal{G}_{F_1(A_1), F_2(A_2), \dots F_m(A_m)}(E)$
  - Null in grouping attributes (G_i):
    - If two tuples are the same on all G_i → they are in the same group (even if some of their attribute values are null.)
  - Null in aggregated attribute (A_j)
    - Delete null values at the outset, before applying aggregation.
      - If the resultant multiset is empty, the aggregate result is null.

Aggregate functions Involving null Examples

Name	Age	Food
Jenny	33	null
Donna	**null**	Pizza
Roy	21	Steak
Sara	34	null

member relation

- $\mathcal{G}_{\textbf{sum}(\text{age})}$ (member)

sum(age)
88

Aggregate functions Involving null Examples

Name	Age	**Food**
Jenny	33	null
Donna	null	Pizza
Roy	21	Steak
Sara	34	null

grouped by Food

member relation

- $_{\textbf{food}}\, g\, _{\textbf{sum}(\text{age})}$ (member)

Food	sum(age)
null	67
Pizza	null
Steak	21

Aggregate functions Involving null Examples

Name	Age	Food
Jenny	33	null
Donna	null	Pizza
Roy	21	Steak
Sara	34	null

grouped by Food

member relation

- $_{\textbf{food}}\,\mathcal{G}\,_{\textbf{count}(\text{age})}\,(\text{member})$

Food	count(age)
null	2
Pizza	**0**
Steak	1

Relational Algebra Operators

- **Additional Operators**
  - set intersection: $\cap$
  - natural join:
  - division: $\div$
  - assignment: $\leftarrow$
- **Extended Operators**
  - Generalized Project
  - Aggregate Functions
  - Outer Joins
- **Modification Operations**
  - **Deletion, insertion and updating of tuples**

Deletion

- Similar to a query,
  - BUT instead of displaying those tuples, they are removed from DB
- Delete "the whole tuple"
- 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 Example - 1

- Delete all account records in the Perryridge branch.

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

- Any problem?

***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

***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

Deletion Example - 2

- Delete all account records in the Perryridge branch.

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

- Any problem?
  - Referential Integrity
    - depositor.account_number is a foreign key referring to the account.account_number

Deletion Example - 3

- Delete all account records in the Perryridge branch.

Better solution:

$r1 \leftarrow \sigma_{\ branch\_name\ =\ \text{“Perryridge”}}(account)$

$account \leftarrow account - r1$

$r2 \leftarrow \Pi_{\ customer\_name,\ account\_number}(depositor \bowtie r1)$

$depositor \leftarrow depositor - r2$

Deletion Example - 5

- Delete all accounts at branches located in the city of Brooklyn.
  - realize that the branches still remain, but the accounts in the branches are removed

***branch* relation**

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

***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

***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

Deletion Example - 6

- Delete all accounts at branches located in the city of Brooklyn.
  - realize that the branches still remain, but the accounts in the branches are removed

$$r_1 \leftarrow \sigma_{\textit{branch\_city} = \text{“Brooklyn”}} (\textit{account} \bowtie \textit{branch})$$

$$r_2 \leftarrow \Pi_{\textit{account\_number}, \textit{branch\_name}, \textit{balance}} (r_1)$$

$$r_3 \leftarrow \Pi_{\textit{customer\_name}, \textit{account\_number}} (r_2 \bowtie \text{depositor})$$

$$\textit{account} \leftarrow \text{account} - r_2$$

$$\textit{depositor} \leftarrow \text{depositor} - r_3$$

Insertion

- Tuples inserted must be "compatible" to the schema of the relation being inserted
  - Same arity (same number of attributes)
  - Same domain for corresponding attributes
- 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.

Insertion Example - 1

- Insert information in the database specifying that Smith has $1200 in account A-973 at the Perryridge branch. (**Assume Smith is an existing customer**)

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

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

Insertion Example - 2

- Insert information in the database specifying that Smith has $1200 in account A-973 at the Perryridge branch. (**Assume Smith is an existing customer**)

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

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

Insertion Example - 3

- Provide as a gift for all loan customers in the Perryridge branch, a $200 savings account. Let the loan number serve as the account number for the new savings account.

***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

***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

***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

***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

Insertion Example - 4

- Provide as a gift for **all loan customers** in the **Perryridge** branch, **a $200 savings account**. Let the loan number serve as the account number for the new savings account.

$$r_1 \leftarrow (\sigma_{branch\_name\ =\ \text{“Perryridge”}}(borrower \bowtie \text{loan}))$$

$$account \leftarrow account \cup \Pi_{loan\_number,\ branch\_name,\ 200}(r_1)$$

$$\text{depositor} \leftarrow depositor \cup \Pi_{customer\_name,\ \mathbf{loan\_number}}(r_1)$$

Updating

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

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

  - if the *i* th attribute is **not updated**
    - F_i is the ***JUST an* attribute** of *r* or,
  - if the *i* th 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 Example - 1

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

***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

Update Example - 2

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

$$\textbf{\textit{account}} \leftarrow \Pi_{\,account\_number,\ branch\_name,\ balance\ *\ 1.05}\ (\textbf{\textit{account}})$$

Update Example - 3

- Pay all accounts with balances over $10,000 6 percent interest and pay all others 5 percent

***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

Update Example - 4

- Pay all accounts with balances over $10,000 6 percent interest and pay all others 5 percent

$$account \leftarrow \Pi_{account\_number,\ branch\_name,\ balance\ *\ 1.06}(\sigma_{BAL\ >\ 10000}(account)) \cup \Pi_{account\_number,\ branch\_name,\ balance\ *\ 1.05}(\sigma_{BAL\ \leq\ 10000}(account))$$

Note about Updating

- Make sure that the query expression specifying the updates cover all the tuples (and only the tuples) in the relation being updated.
  - If less, then result in deletion of certain tuples
  - If more, then result in insertion of extra tuples

Practice 3

1. Retrieve customers' name, branch name and balance of an account whose balance is between 500 and 700 inclusive.
2. Retrieve all branch information that has assets more than the asset at the branch "Round Hill".
3. Retrieve customers' name whose loan account in both "Round Hill" and "Redwood".
4. Retrieve customers' name whose account either in "Downtown" or "Mianus" or both.
5. Retrieve customers' name, account number and balance of customers who have joined account.
6. Retrieve customers' name and account number who have more than one account.

Practice 3 (Cont.)

7. Retrieve highest total assets of all branches that are located in the same city.
8. Retrieve average balance of all customers who lives in “Harrison” and “Stamford”
9. Retrieve the number of customers who have more than one account.
10. Retrieve the number of accounts that have more than one account holder.