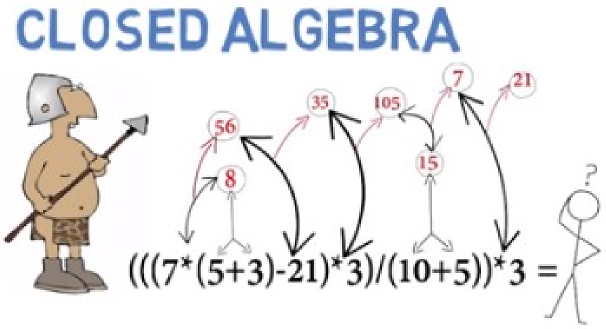
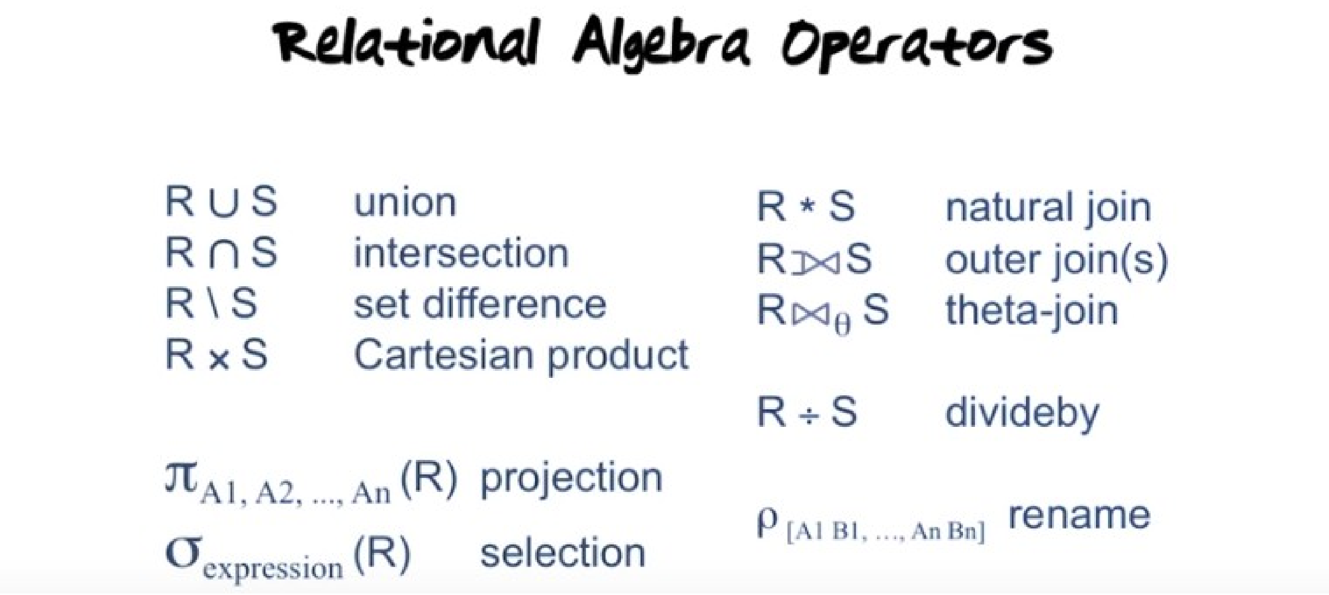


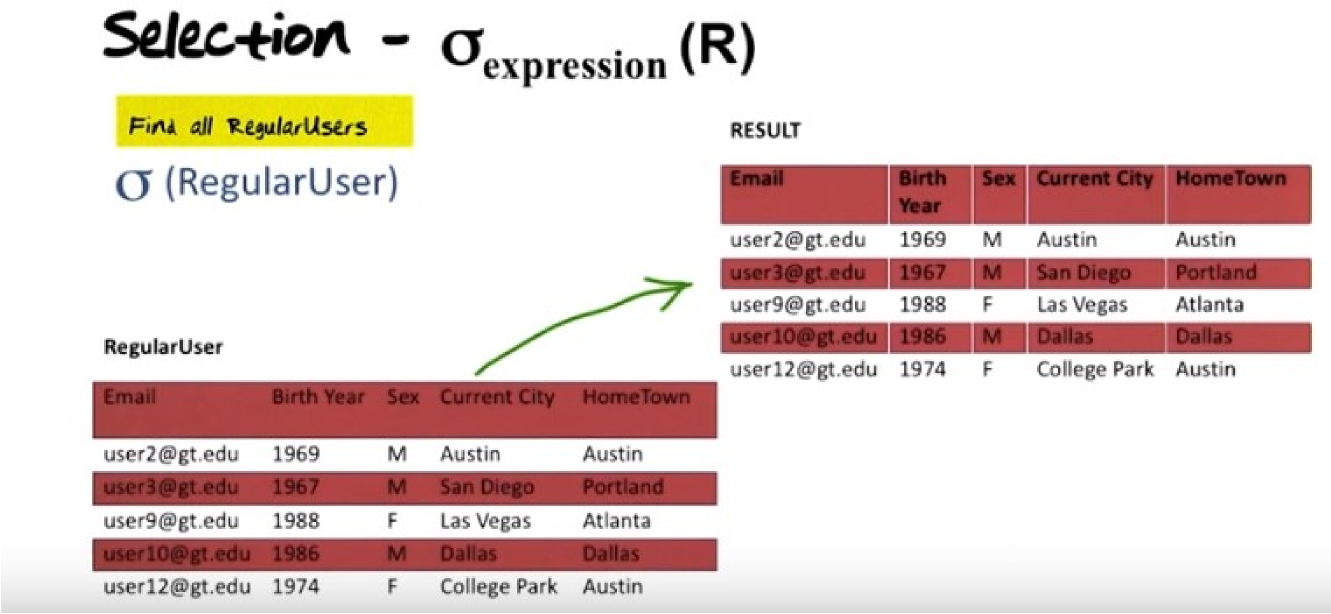
The above figure: the algebra is said to be closed because each time you do an operation on these rational number, you get a rational number back, the same type as the operands you started out with. That's what a closed algebra is.



Relational Algebra Operators

$R \cup S$	union	$R * S$	natural join
$R \cap S$	intersection	R⟗S	outer join(s)
$R \setminus S$	set difference	$R \bowtie_{\theta} S$	theta-join
$R \times S$	Cartesian product		
		$R \div S$	divideby
$\pi_{A1, A2, ..., An}(R)$	projection	$\rho_{[A1\ B1, ..., An\ Bn]}$	rename
$\sigma_{expression}(R)$	selection		

Projection operator: you can think of as eliminating columns from a result. The selection operator: you can think of as eliminating rows. 符號的記憶: π 即 pi, 跟 projection 首字母相同; σ 即 sigma, 跟 selection 首字母相同.



Selection - $\sigma_{\text{expression}}$ (R)

Find all RegularUsers

σ (RegularUser)

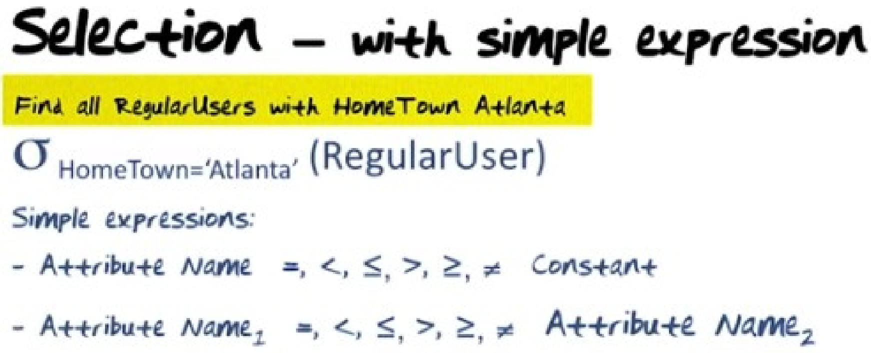
RESULT

Email	Birth Year	Sex	Current City	HomeTown
user2@gt.edu	1969	M	Austin	Austin
user3@gt.edu	1967	M	San Diego	Portland
user9@gt.edu	1988	F	Las Vegas	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas
user12@gt.edu	1974	F	College Park	Austin

RegularUser

Email	Birth Year	Sex	Current City	HomeTown
user2@gt.edu	1969	M	Austin	Austin
user3@gt.edu	1967	M	San Diego	Portland
user9@gt.edu	1988	F	Las Vegas	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas
user12@gt.edu	1974	F	College Park	Austin

上圖: σ_expression (R) 中的 expression 即選擇的要求, 若無 expression, 則表示所有的都選, 比如上圖中的例子.



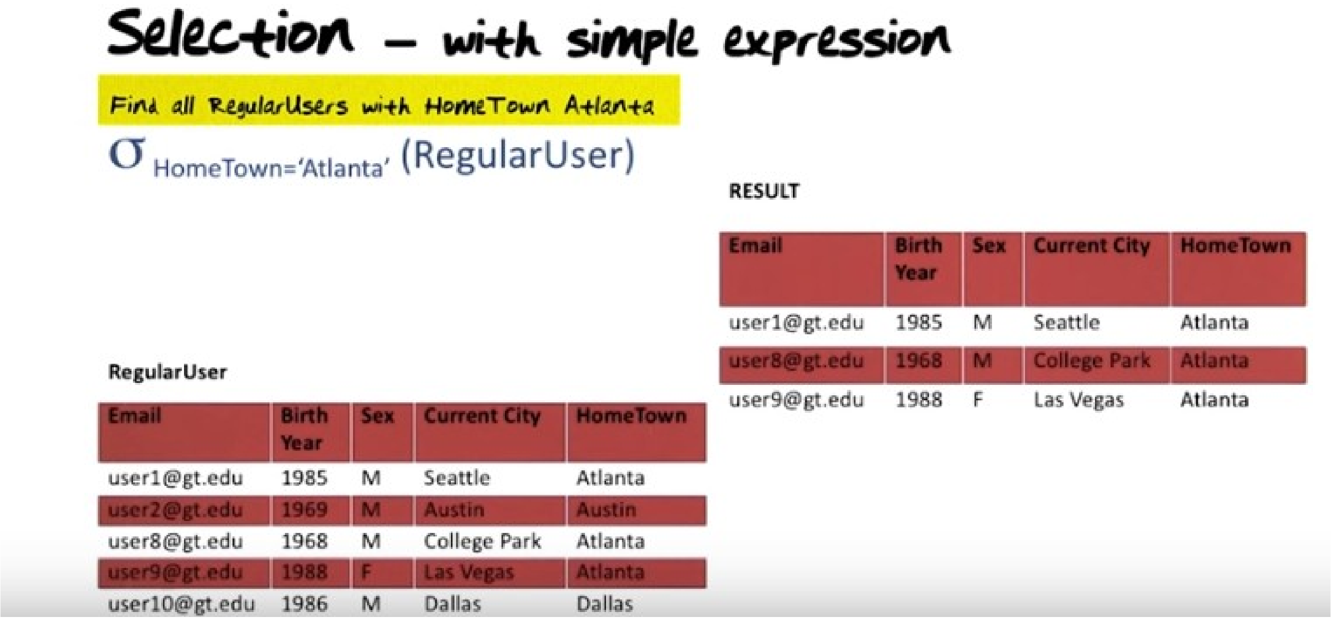
Selection – with simple expression

Find all RegularUsers with HomeTown Atlanta

$\sigma_{\text{HomeTown='Atlanta'}}$ (RegularUser)

Simple expressions:

- Attribute Name =, <, ≤, >, ≥, ≠ Constant
- Attribute Name_1 =, <, ≤, >, ≥, ≠ Attribute Name_2



Selection – with simple expression

Find all RegularUsers with HomeTown Atlanta

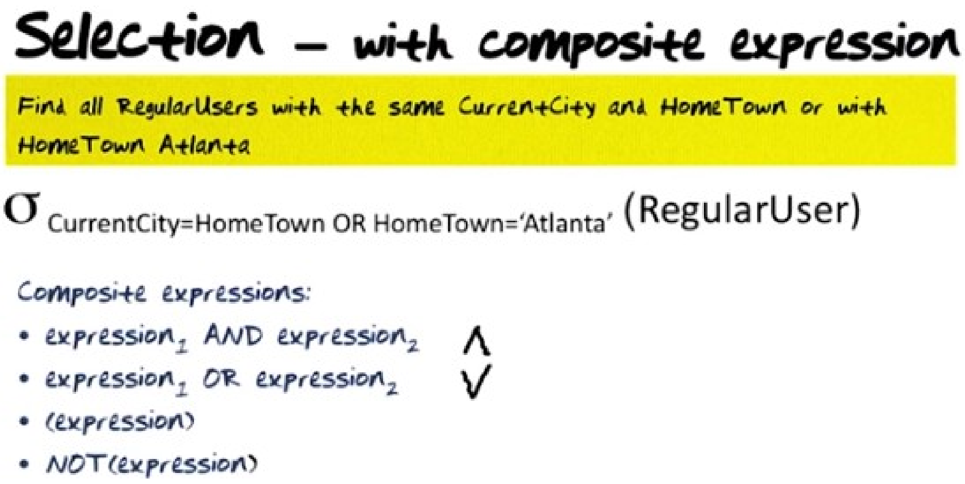
$\sigma_{\text{HomeTown='Atlanta'}}(\text{RegularUser})$

RESULT

Email	Birth Year	Sex	Current City	HomeTown
user1@gt.edu	1985	M	Seattle	Atlanta
user8@gt.edu	1968	M	College Park	Atlanta
user9@gt.edu	1988	F	Las Vegas	Atlanta

RegularUser

Email	Birth Year	Sex	Current City	HomeTown
user1@gt.edu	1985	M	Seattle	Atlanta
user2@gt.edu	1969	M	Austin	Austin
user8@gt.edu	1968	M	College Park	Atlanta
user9@gt.edu	1988	F	Las Vegas	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas



Selection – with composite expression

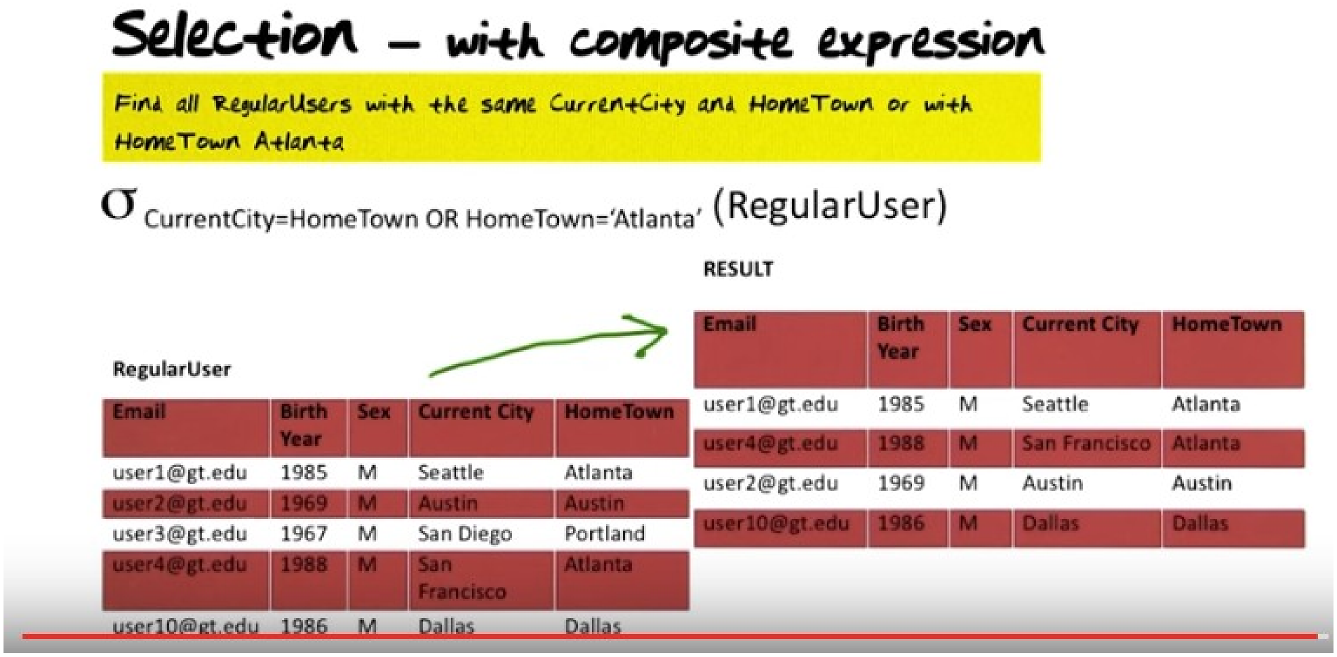
Find all RegularUsers with the same CurrentCity and HomeTown or with HomeTown Atlanta

$\sigma_{\text{CurrentCity=HomeTown OR HomeTown='Atlanta'}}(\text{RegularUser})$

Composite expressions:

- expression_1 AND expression_2 $\wedge$
- expression_1 OR expression_2 $\vee$
- (expression)
- NOT(expression)

^叫做 conjunction, V 叫做 disjunction. (expression)表示判斷 expression 是否為 true. Think of Java.



Selection – with composite expression

Find all RegularUsers with the same CurrentCity and HomeTown or with HomeTown Atlanta

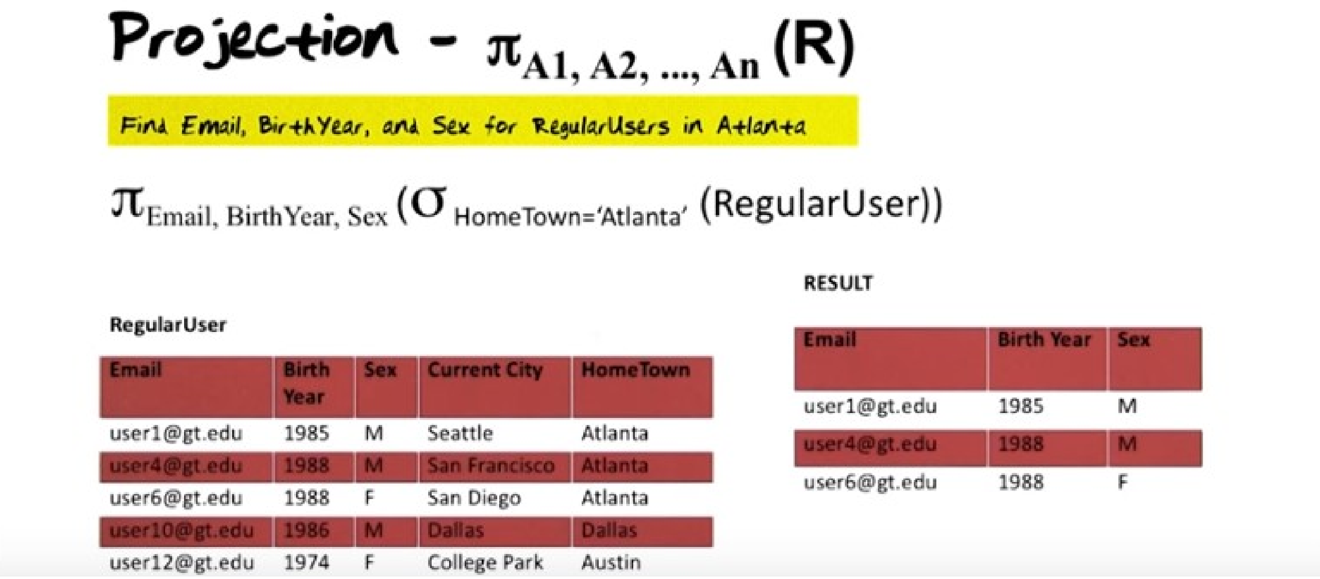
$\sigma_{\text{CurrentCity=HomeTown OR HomeTown='Atlanta'}}$ (RegularUser)

RegularUser

Email	Birth Year	Sex	Current City	HomeTown
user1@gt.edu	1985	M	Seattle	Atlanta
user2@gt.edu	1969	M	Austin	Austin
user3@gt.edu	1967	M	San Diego	Portland
user4@gt.edu	1988	M	San Francisco	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas

RESULT

Email	Birth Year	Sex	Current City	HomeTown
user1@gt.edu	1985	M	Seattle	Atlanta
user4@gt.edu	1988	M	San Francisco	Atlanta
user2@gt.edu	1969	M	Austin	Austin
user10@gt.edu	1986	M	Dallas	Dallas



Projection - $\pi_{A1, A2, ..., An}$ (R)

Find Email, BirthYear, and Sex for RegularUsers in Atlanta

$\pi_{\text{Email, BirthYear, Sex}}(\sigma_{\text{HomeTown='Atlanta'}}$ (RegularUser))

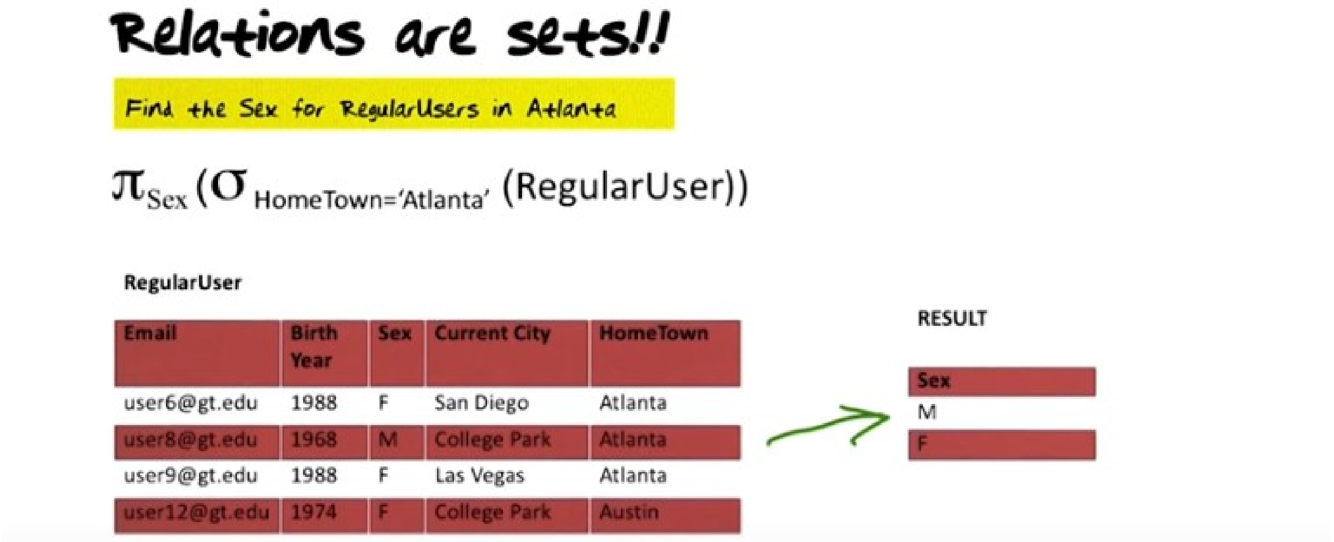
RegularUser

Email	Birth Year	Sex	Current City	HomeTown
user1@gt.edu	1985	M	Seattle	Atlanta
user4@gt.edu	1988	M	San Francisco	Atlanta
user6@gt.edu	1988	F	San Diego	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas
user12@gt.edu	1974	F	College Park	Austin

RESULT

Email	Birth Year	Sex
user1@gt.edu	1985	M
user4@gt.edu	1988	M
user6@gt.edu	1988	F

The above figure: what I talked about earlier that relational algebra is a closed query language, because what actually happends here is that this query here is done in two steps when we formulated. First, we take RegularUser and do a selection from it with the selecction condition, and then from the result of that which we know is a relation. We then do the projection on that. So this is a composite query using two operators and this can only be done when your algebra is closed.



Relations are sets!!

Find the Sex for RegularUsers in Atlanta

$\pi_{Sex}(\sigma_{HomeTown='Atlanta'}(RegularUser))$

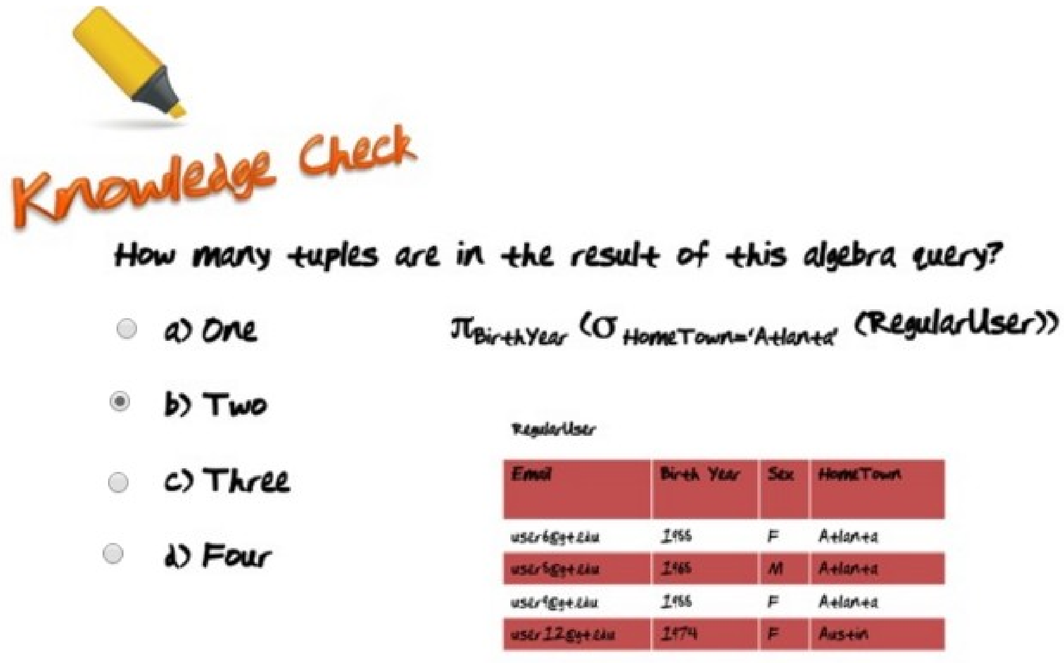
RegularUser

Email	Birth Year	Sex	Current City	HomeTown
user6@gt.edu	1988	F	San Diego	Atlanta
user8@gt.edu	1968	M	College Park	Atlanta
user9@gt.edu	1988	F	Las Vegas	Atlanta
user12@gt.edu	1974	F	College Park	Austin

RESULT

Sex
M
F

The above figure: RegularUser 中有兩個 F 滿足要求(Hometown='Atlanta'), 但 RESULT 中只有一個 F. 這是因為 relations are sets (think of Java 中的 Set, 沒有重復的元素). And again, relation algebra is a closed query language. You start with a relation, you end with a relation. Relation is a set, therefore you only have two rows: male and female. 由此可知, σ 和 π 弄出來的都是 relation.



Knowledge Check

How many tuples are in the result of this algebra query?

○ a) One

● b) Two

○ c) Three

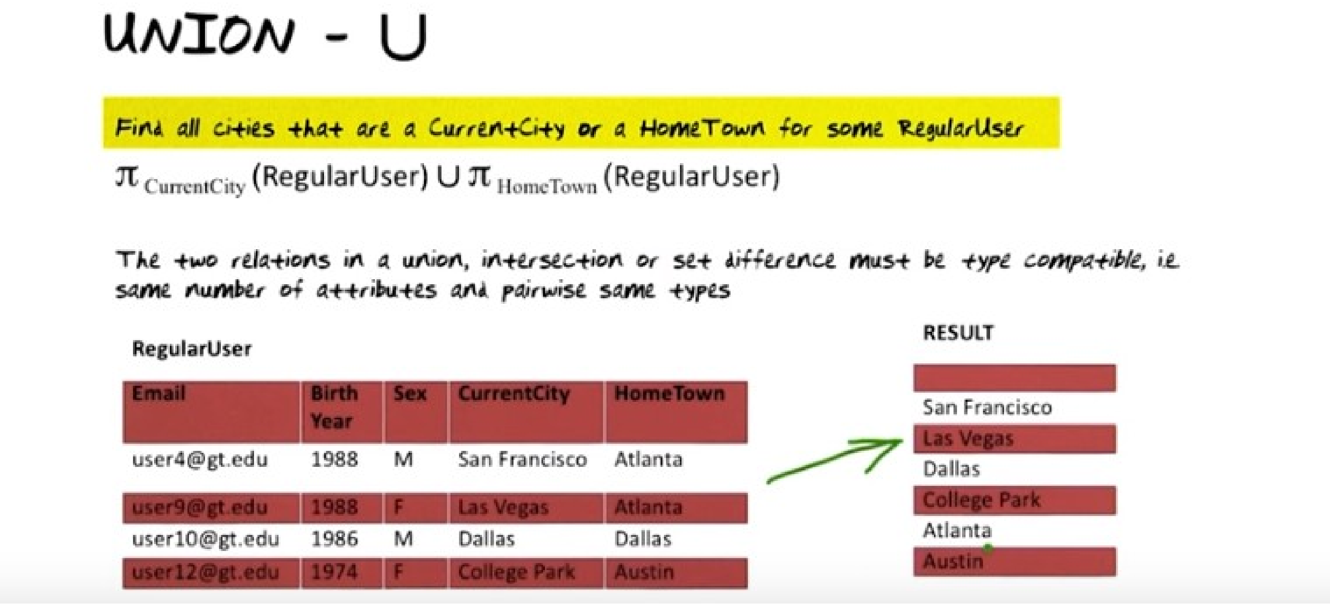
○ d) Four

$\pi_{BirthYear}(\sigma_{HomeTown='Atlanta'}(RegularUser))$

RegularUser

Email	Birth Year	Sex	HomeTown
user6@gt.edu	1966	F	Atlanta
user8@gt.edu	1966	M	Atlanta
user9@gt.edu	1966	F	Atlanta
user12@gt.edu	1974	F	Austin

The above figure: the Birth Year in each row are: 1966, 1966, 1966, 1974.



UNION - U

Find all cities that are a CurrentCity or a HomeTown for some RegularUser

$\pi_{\text{CurrentCity}}$ (RegularUser) ∪ π_{HomeTown} (RegularUser)

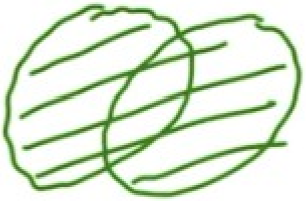
The two relations in a union, intersection or set difference must be type compatible, ie same number of attributes and pairwise same types

RegularUser

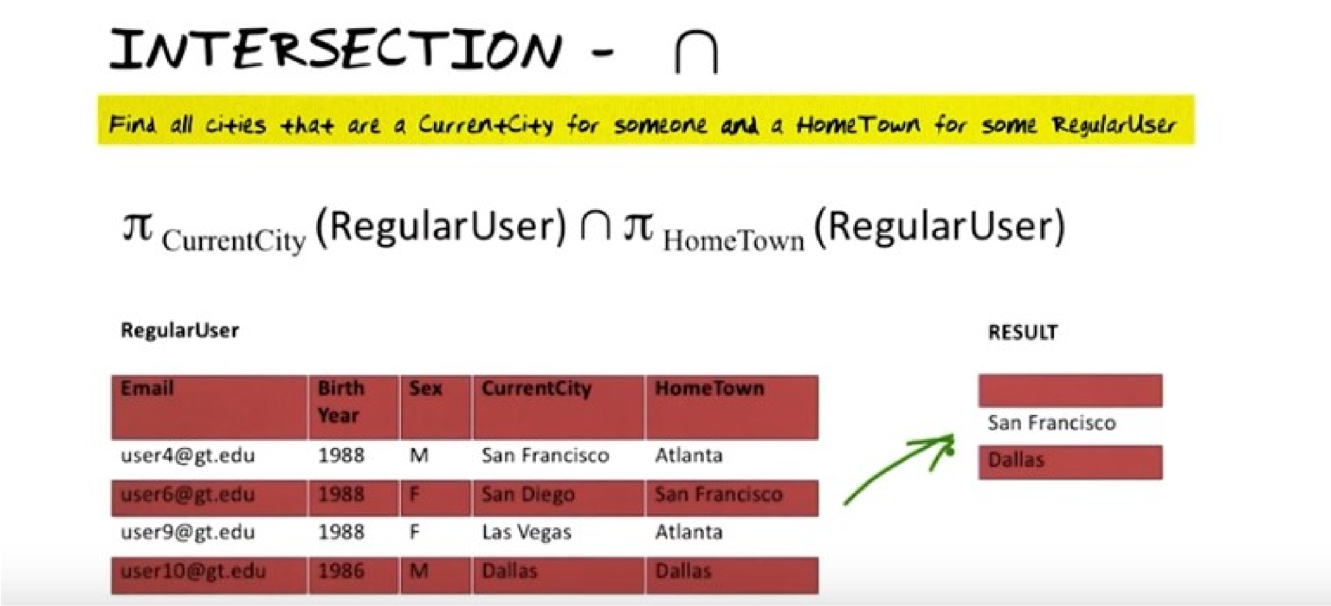
Email	Birth Year	Sex	CurrentCity	HomeTown
user4@gt.edu	1988	M	San Francisco	Atlanta
user9@gt.edu	1988	F	Las Vegas	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas
user12@gt.edu	1974	F	College Park	Austin

RESULT

San Francisco
Las Vegas
Dallas
College Park
Atlanta
Austin



上圖中那句 type compatible 的意思是指: U 兩邊的 relations 要有相同數量的 attributes. 注意上例中, U 兩邊的 relations 是 projection 之後的, 它們都只有一個 attribute(即 CurrentCity 或 HomeTown). 另外, pairwise same types 的意思是指 U 兩邊的 relations 的 attributes 配對後的 type 要相同，如 CurrentCity 和 HomeTown 都是 varchar 50.



INTERSECTION - ∩

Find all cities that are a CurrentCity for someone and a HomeTown for some RegularUser

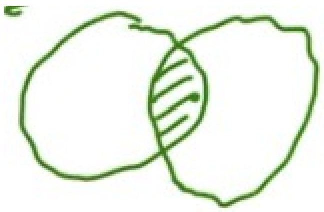
$$\pi_{\text{CurrentCity}}(\text{RegularUser}) \cap \pi_{\text{HomeTown}}(\text{RegularUser})$$

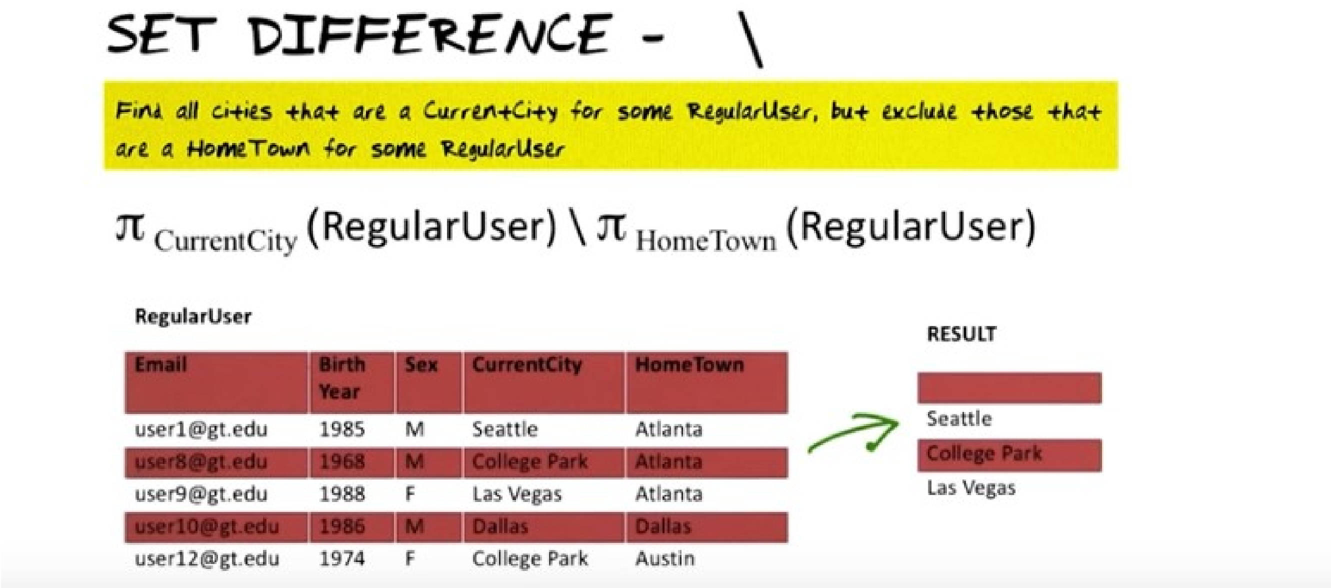
RegularUser

Email	Birth Year	Sex	CurrentCity	HomeTown
user4@gt.edu	1988	M	San Francisco	Atlanta
user6@gt.edu	1988	F	San Diego	San Francisco
user9@gt.edu	1988	F	Las Vegas	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas

RESULT

San Francisco
Dallas





SET DIFFERENCE - \

Find all cities that are a CurrentCity for some RegularUser, but exclude those that are a HomeTown for some RegularUser

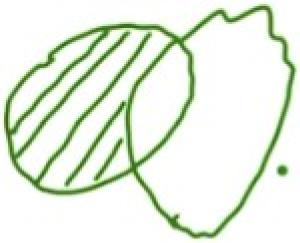
$$\pi_{\text{CurrentCity}}(\text{RegularUser}) \setminus \pi_{\text{HomeTown}}(\text{RegularUser})$$

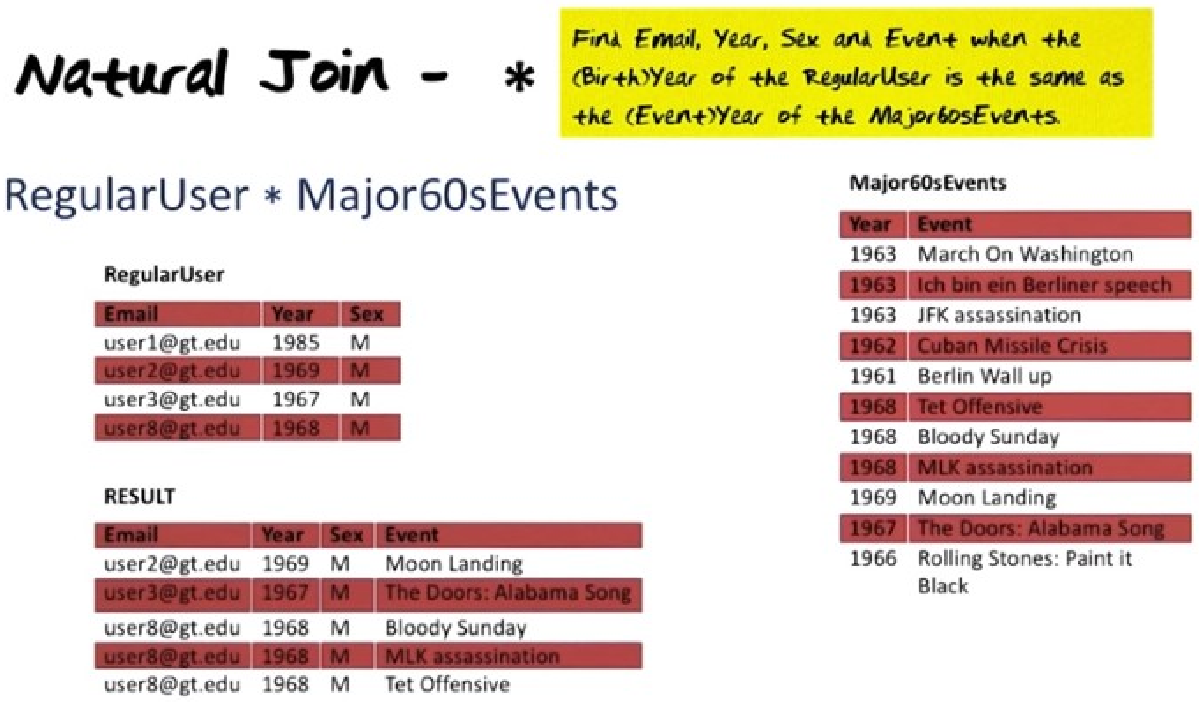
RegularUser

Email	Birth Year	Sex	CurrentCity	HomeTown
user1@gt.edu	1985	M	Seattle	Atlanta
user8@gt.edu	1968	M	College Park	Atlanta
user9@gt.edu	1988	F	Las Vegas	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas
user12@gt.edu	1974	F	College Park	Austin

RESULT

Seattle
College Park
Las Vegas





Natural Join - *

Find Email, Year, Sex and Event when the (Birth)Year of the RegularUser is the same as the (Event)Year of the Major60sEvents.

RegularUser * Major60sEvents

RegularUser

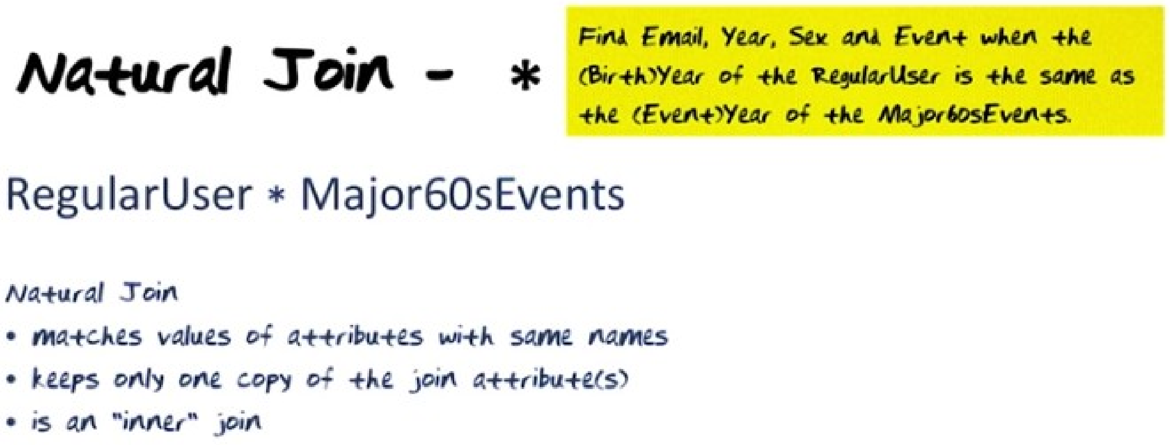
Email	Year	Sex
user1@gt.edu	1985	M
user2@gt.edu	1969	M
user3@gt.edu	1967	M
user8@gt.edu	1968	M

RESULT

Email	Year	Sex	Event
user2@gt.edu	1969	M	Moon Landing
user3@gt.edu	1967	M	The Doors: Alabama Song
user8@gt.edu	1968	M	Bloody Sunday
user8@gt.edu	1968	M	MLK assassination
user8@gt.edu	1968	M	Tet Offensive

Major60sEvents

Year	Event
1963	March On Washington
1963	Ich bin ein Berliner speech
1963	JFK assassination
1962	Cuban Missile Crisis
1961	Berlin Wall up
1968	Tet Offensive
1968	Bloody Sunday
1968	MLK assassination
1969	Moon Landing
1967	The Doors: Alabama Song
1966	Rolling Stones: Paint it Black



Natural Join - *

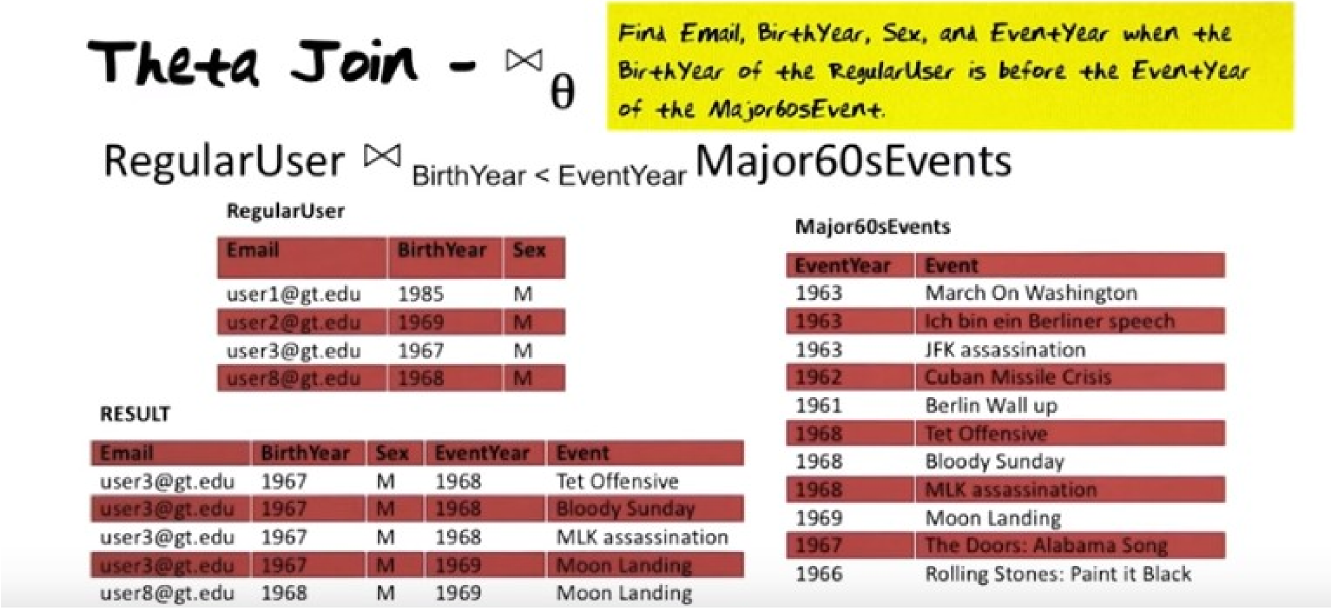
Find Email, Year, Sex and Event when the (Birth)Year of the RegularUser is the same as the (Event)Year of the Major60sEvents.

RegularUser * Major60sEvents

Natural Join

- matches values of attributes with same names
- keeps only one copy of the join attribute(s)
- is an "inner" join

The above figure: it's called an inner join because only the tuples that actually appear in the relation and match will appear in the result. Those that had no matches will not appear in the result. “Keeps only one copy of the join attributes”沒解釋. 注意上上圖中 RESULT 中沒有 user1, 但有三個 user8.



Theta Join - $\bowtie_\theta$

Find Email, BirthYear, Sex, and EventYear when the BirthYear of the RegularUser is before the EventYear of the Major60sEvent.

RegularUser $\bowtie_{\text{BirthYear < EventYear}}$ Major60sEvents

RegularUser

Email	BirthYear	Sex
user1@gt.edu	1985	M
user2@gt.edu	1969	M
user3@gt.edu	1967	M
user8@gt.edu	1968	M

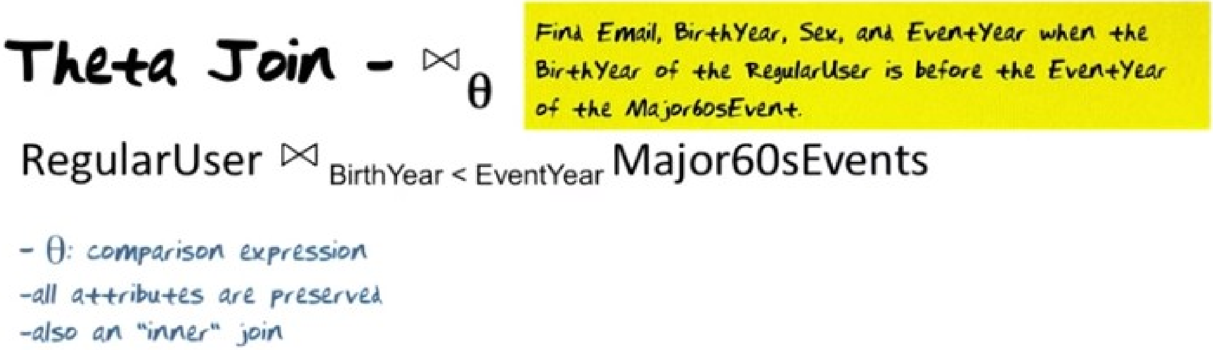
Major60sEvents

EventYear	Event
1963	March On Washington
1963	Ich bin ein Berliner speech
1963	JFK assassination
1962	Cuban Missile Crisis
1961	Berlin Wall up
1968	Tet Offensive
1968	Bloody Sunday
1968	MLK assassination
1969	Moon Landing
1967	The Doors: Alabama Song
1966	Rolling Stones: Paint it Black

RESULT

Email	BirthYear	Sex	EventYear	Event
user3@gt.edu	1967	M	1968	Tet Offensive
user3@gt.edu	1967	M	1968	Bloody Sunday
user3@gt.edu	1967	M	1968	MLK assassination
user3@gt.edu	1967	M	1969	Moon Landing
user8@gt.edu	1968	M	1969	Moon Landing

In contrast to the natural join, in this case the two join attributes have different names. With the natural join, they had the same name. Also in contrast to the natural join, both of the join attributes will appear in the result: BirthYear appears and EventYear appears.

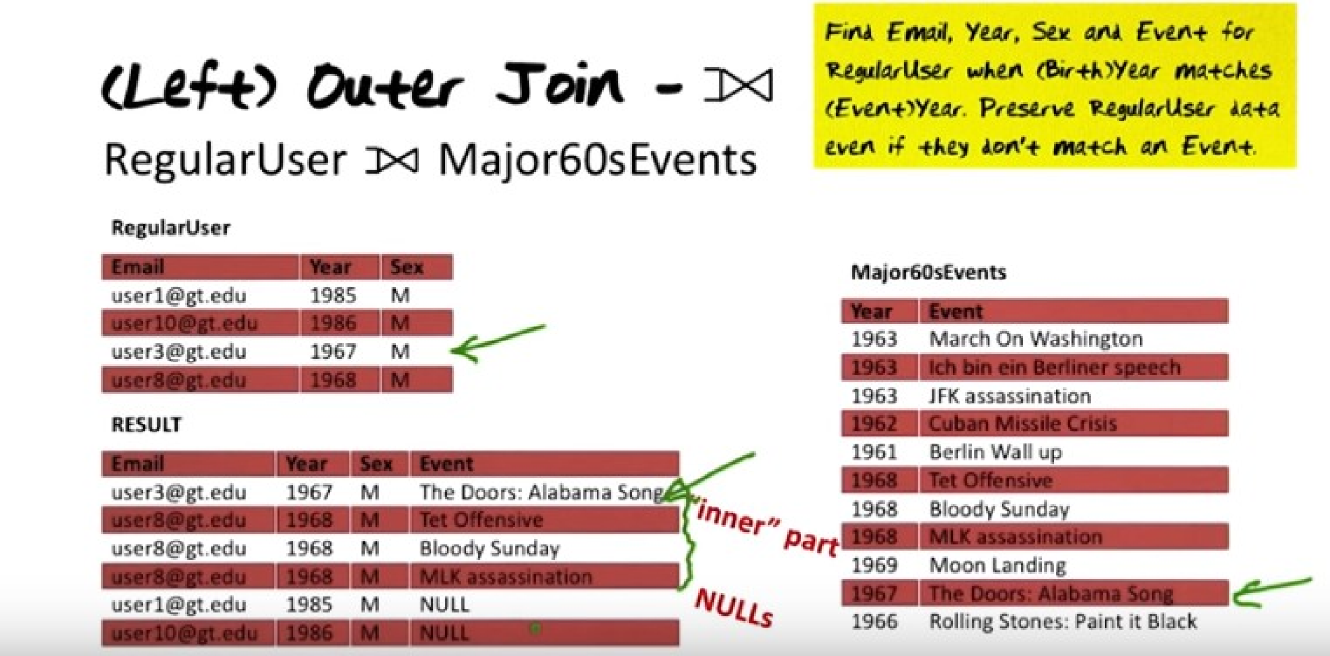


Theta Join - $\bowtie_\theta$

Find Email, BirthYear, Sex, and EventYear when the BirthYear of the RegularUser is before the EventYear of the Major60sEvent.

RegularUser $\bowtie_{\text{BirthYear < EventYear}}$ Major60sEvents

- θ: comparison expression
-all attributes are preserved
-also an "inner" join



(Left) Outer Join - ⟕

RegularUser ⟕ Major60sEvents

Find Email, Year, Sex and Event for RegularUser when (Birth)Year matches (Event)Year. Preserve RegularUser data even if they don't match an Event.

RegularUser

Email	Year	Sex
user1@gt.edu	1985	M
user10@gt.edu	1986	M
user3@gt.edu	1967	M
user8@gt.edu	1968	M

RESULT

Email	Year	Sex	Event
user3@gt.edu	1967	M	The Doors: Alabama Song
user8@gt.edu	1968	M	Tet Offensive
user8@gt.edu	1968	M	Bloody Sunday
user8@gt.edu	1968	M	MLK assassination
user1@gt.edu	1985	M	NULL
user10@gt.edu	1986	M	NULL

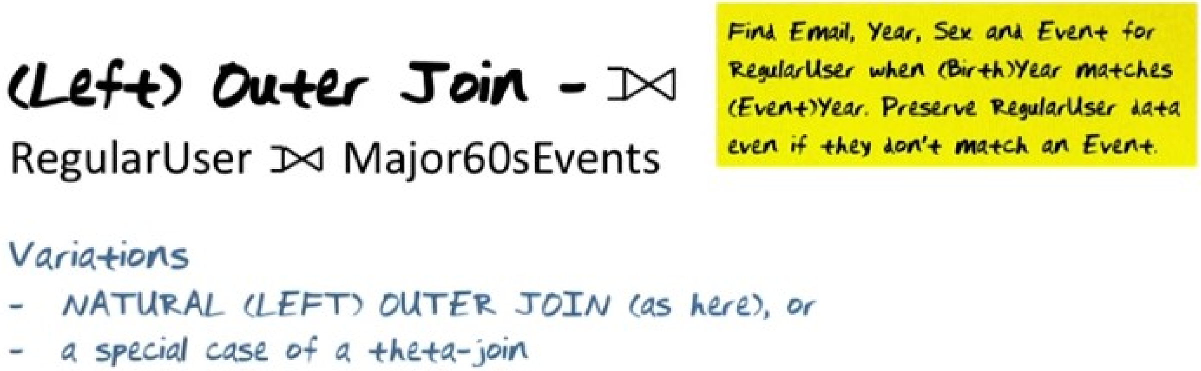
"inner" part

NULLs

Major60sEvents

Year	Event
1963	March On Washington
1963	Ich bin ein Berliner speech
1963	JFK assassination
1962	Cuban Missile Crisis
1961	Berlin Wall up
1968	Tet Offensive
1968	Bloody Sunday
1968	MLK assassination
1969	Moon Landing
1967	The Doors: Alabama Song
1966	Rolling Stones: Paint it Black

The above figure: RESULT 中的 user1 和 user10 是 outer part. 注意是只將 left operand(即 RegularUser)中的沒 match 到的弄到 outer part 中去了, right operand 中的沒管, 所以叫 left outer join (M.O.)



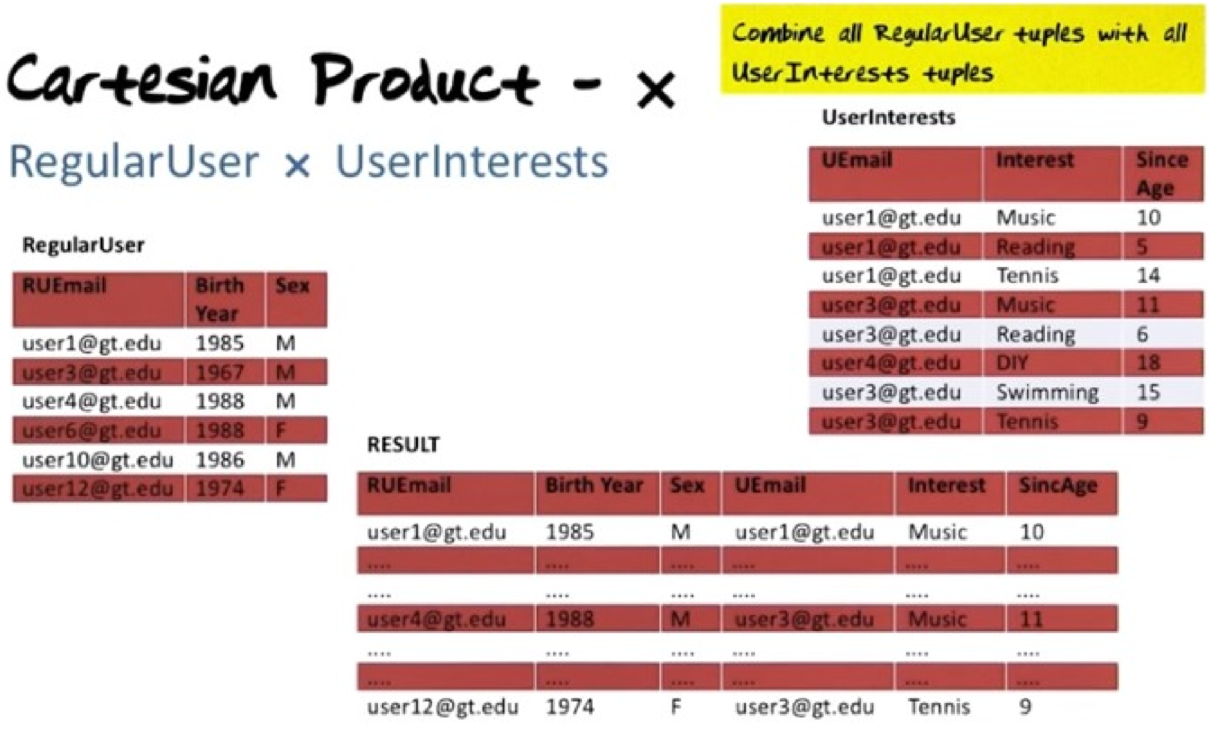
(Left) Outer Join - ⟕

RegularUser ⟕ Major60sEvents

Find Email, Year, Sex and Event for RegularUser when (Birth)Year matches (Event)Year. Preserve RegularUser data even if they don't match an Event.

Variations

- NATURAL (LEFT) OUTER JOIN (as here), or
- a special case of a theta-join



Cartesian Product - ×

Combine all RegularUser tuples with all UserInterests tuples

RegularUser × UserInterests

RegularUser

RUEmail	Birth Year	Sex
user1@gt.edu	1985	M
user3@gt.edu	1967	M
user4@gt.edu	1988	M
user6@gt.edu	1988	F
user10@gt.edu	1986	M
user12@gt.edu	1974	F

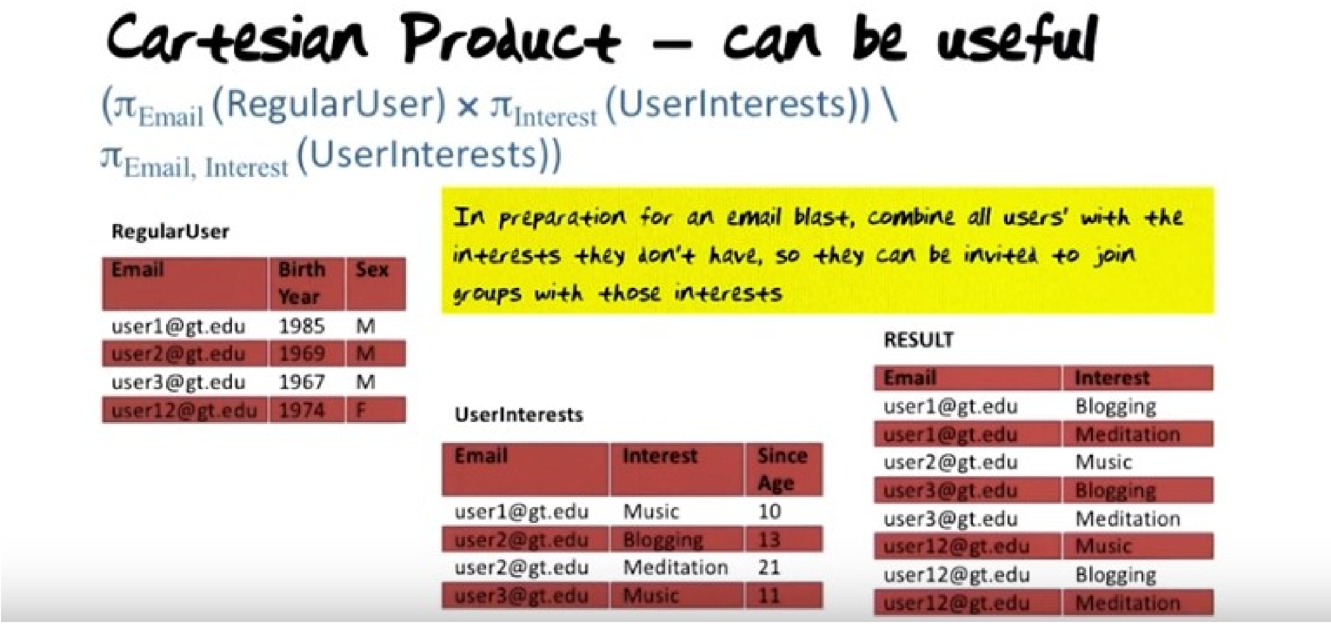
UserInterests

UEmail	Interest	Since Age
user1@gt.edu	Music	10
user1@gt.edu	Reading	5
user1@gt.edu	Tennis	14
user3@gt.edu	Music	11
user3@gt.edu	Reading	6
user4@gt.edu	DIY	18
user3@gt.edu	Swimming	15
user3@gt.edu	Tennis	9

RESULT

RUEmail	Birth Year	Sex	UEmail	Interest	SincAge
user1@gt.edu	1985	M	user1@gt.edu	Music	10
....	....	....	....	....	....
....	....	....	....	....	....
user4@gt.edu	1988	M	user3@gt.edu	Music	11
....	....	....	....	....	....
....	....	....	....	....	....
user12@gt.edu	1974	F	user3@gt.edu	Tennis	9

上圖: Cartesian Product 就相當於群論中的直乘. RegularUser 中的每一行都跟 UserInterest 中的所有行組成一個新行.



Cartesian Product – can be useful

$(\pi_{\text{Email}}(\text{RegularUser}) \times \pi_{\text{Interest}}(\text{UserInterests})) \setminus \pi_{\text{Email, Interest}}(\text{UserInterests}))$

In preparation for an email blast, combine all users' with the interests they don't have, so they can be invited to join groups with those interests

RegularUser

Email	Birth Year	Sex
user1@gt.edu	1985	M
user2@gt.edu	1969	M
user3@gt.edu	1967	M
user12@gt.edu	1974	F

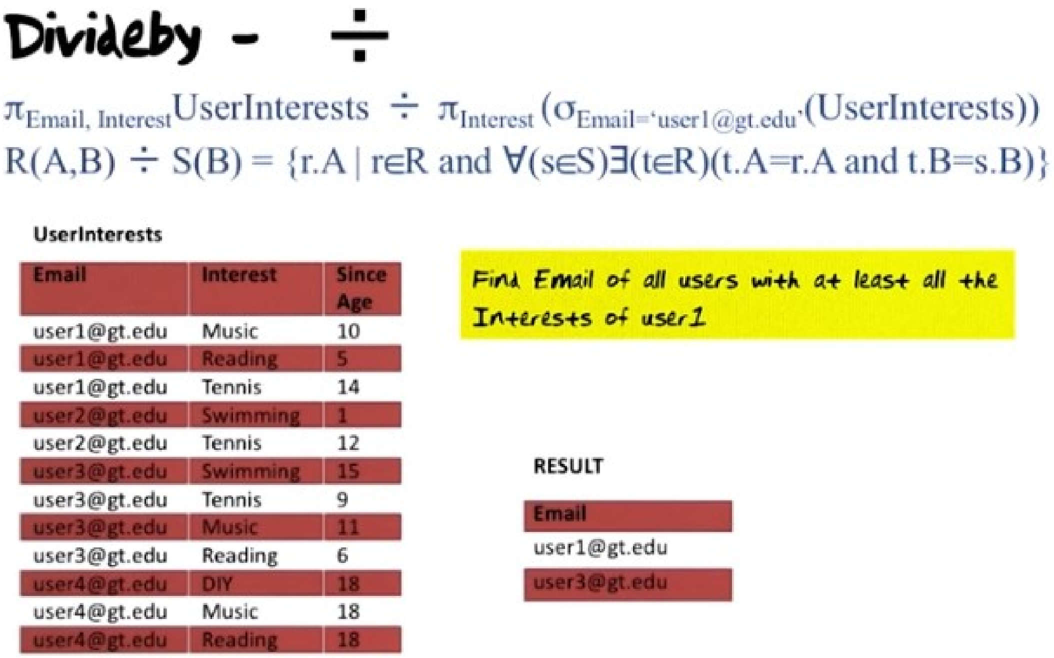
UserInterests

Email	Interest	Since Age
user1@gt.edu	Music	10
user2@gt.edu	Blogging	13
user2@gt.edu	Meditation	21
user3@gt.edu	Music	11

RESULT

Email	Interest
user1@gt.edu	Blogging
user1@gt.edu	Meditation
user2@gt.edu	Music
user3@gt.edu	Blogging
user3@gt.edu	Meditation
user12@gt.edu	Music
user12@gt.edu	Blogging
user12@gt.edu	Meditation

上圖: blast: 一陣風,爆炸,此處應該是群發郵件的意思. π_Email, Interest(UserInterests)意思是在 UserInterest 中將 Email 和 Interest 之外的所有列都刪掉, 即只留下 Email 和 Interest 兩列. Notice that the two operands in the substraction are type compatible in this case here, because you have Email × Interest, and you substract from that email and interest.



Divideby - ÷

$\pi_{Email,\ Interest}UserInterests \div \pi_{Interest}(\sigma_{Email='user1@gt.edu'}(UserInterests))$

$R(A,B) \div S(B) = \{r.A \mid r\in R \text{ and } \forall(s\in S)\exists(t\in R)(t.A=r.A \text{ and } t.B=s.B)\}$

UserInterests

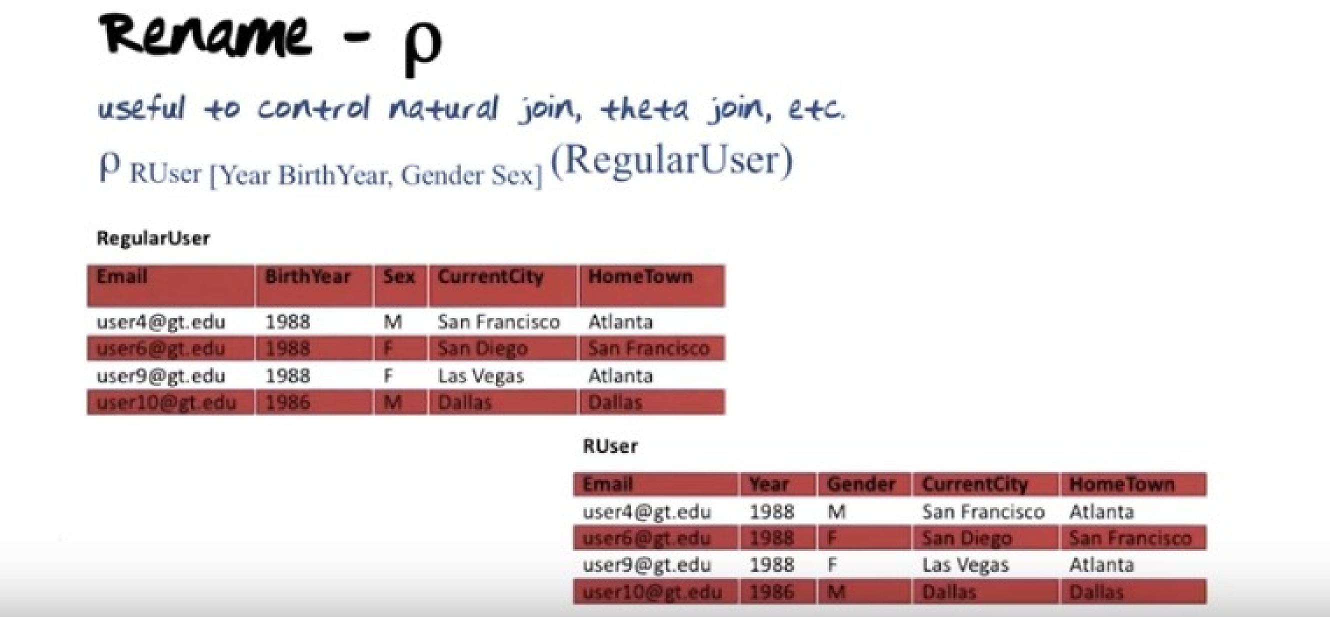
Email	Interest	Since Age
user1@gt.edu	Music	10
user1@gt.edu	Reading	5
user1@gt.edu	Tennis	14
user2@gt.edu	Swimming	1
user2@gt.edu	Tennis	12
user3@gt.edu	Swimming	15
user3@gt.edu	Tennis	9
user3@gt.edu	Music	11
user3@gt.edu	Reading	6
user4@gt.edu	DIY	18
user4@gt.edu	Music	18
user4@gt.edu	Reading	18

Find Email of all users with at least all the Interests of user1

RESULT

Email
user1@gt.edu
user3@gt.edu

÷右邊的 operand 括號中總是右邊 operand 的第二個 attribute(即 B)



Rename - ρ

useful to control natural join, theta join, etc.

$\rho_{RUser\ [Year\ BirthYear,\ Gender\ Sex]}(RegularUser)$

RegularUser

Email	BirthYear	Sex	CurrentCity	HomeTown
user4@gt.edu	1988	M	San Francisco	Atlanta
user6@gt.edu	1988	F	San Diego	San Francisco
user9@gt.edu	1988	F	Las Vegas	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas

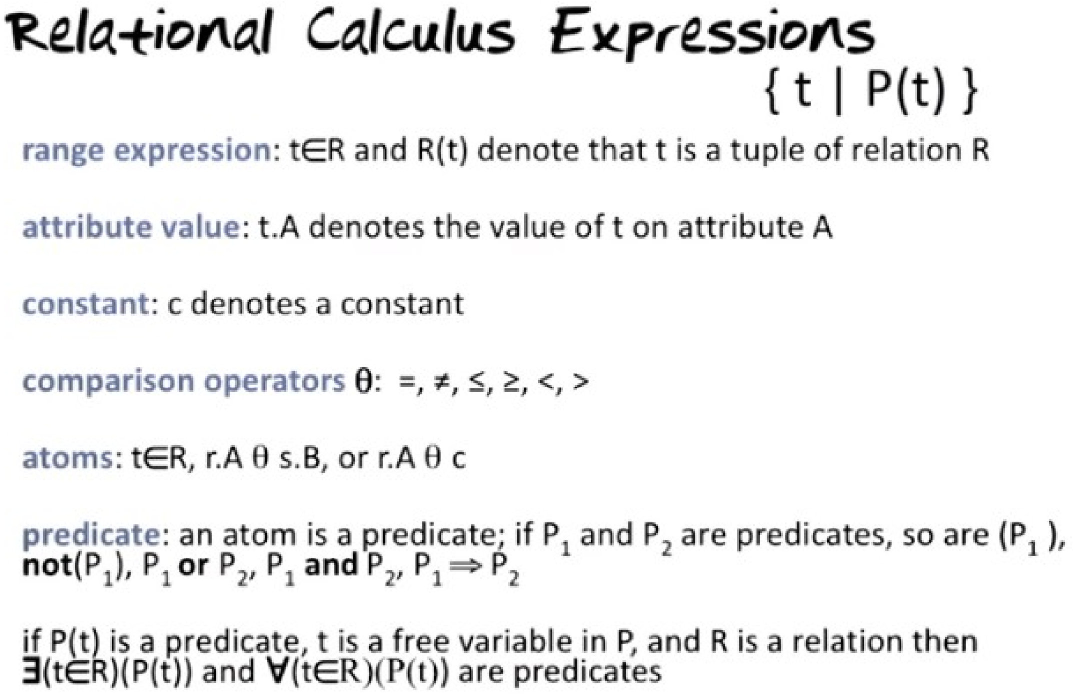
RUser

Email	Year	Gender	CurrentCity	HomeTown
user4@gt.edu	1988	M	San Francisco	Atlanta
user6@gt.edu	1988	F	San Diego	San Francisco
user9@gt.edu	1988	F	Las Vegas	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas

上圖中的 ρ 那個式子的意思即將 BirthYear 改名為 Year, Sex 改名為 Gender, 改後的表的名字也從 RegularUser 改為 RUser.

Relational Calculus

The above figure (沒錯, 上圖就只有 Relational Calculus 兩個單詞): the two fundamental formalisms for relational database query languages: relational algebra and relational calculus (不只是 calculus, 而是 relational calculus). To distinguish between the two, relational algebra is procedural in nature, it's operator based and basically what you do is you say take this relation, …. , so you are basically describing step by step what you're supposed to do to get to the result. Calculus, in comparison, is declared to be in nature. Instead of saying what to do step by step, you simply describe the result that you want. So it may appear that it's a higher-level language with more expressive power. As it turns out, the algebra and calculus were defined in such a manner that with respet to data retrieval horsepower, they can actually be shown to be equivalent. SQL is mostly based on tuple calculus.



Relational Calculus Expressions

$\{ t \mid P(t) \}$

range expression: $t \in R$ and $R(t)$ denote that t is a tuple of relation R

attribute value: t.A denotes the value of t on attribute A

constant: c denotes a constant

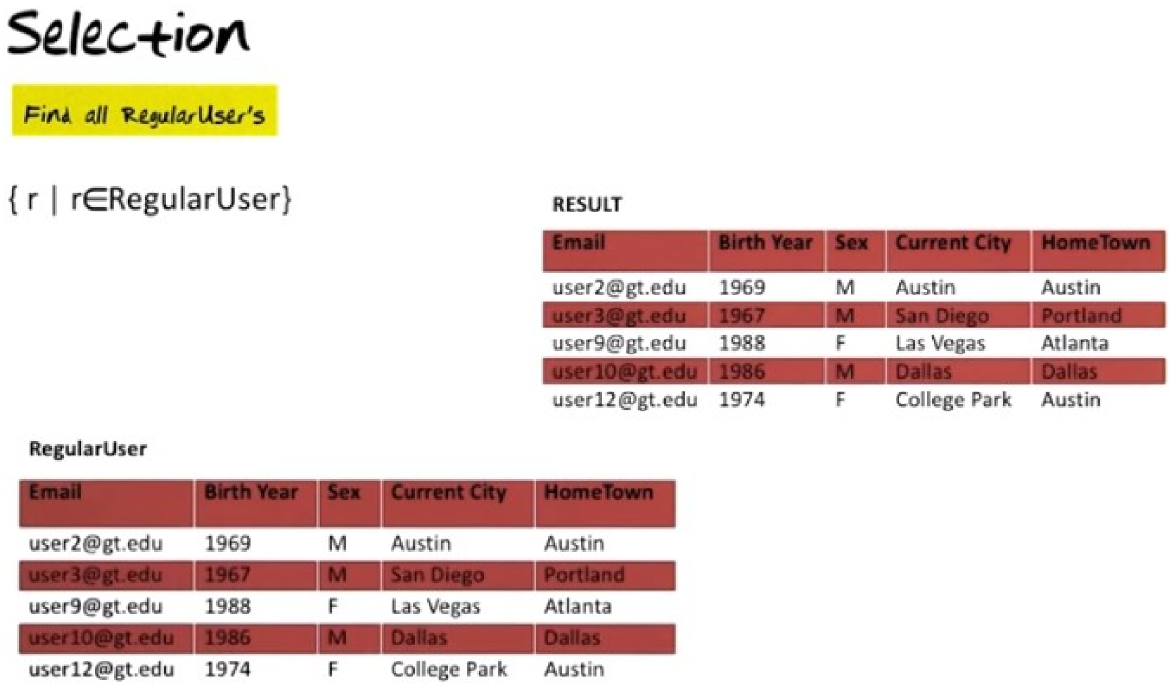
comparison operators θ: $=, \neq, \leq, \geq, <, >$

atoms: $t \in R$, $r.A\ \theta\ s.B$, or $r.A\ \theta\ c$

predicate: an atom is a predicate; if P_1 and P_2 are predicates, so are (P_1), **not**(P_1), P_1 **or** P_2, P_1 **and** P_2, $P_1 \Rightarrow P_2$

if P(t) is a predicate, t is a free variable in P, and R is a relation then $\exists(t \in R)(P(t))$ and $\forall(t \in R)(P(t))$ are predicates

上圖中的(P1)表示判斷 P1 是否為 true.



Selection

Find all RegularUser's

{ r | r∈RegularUser}

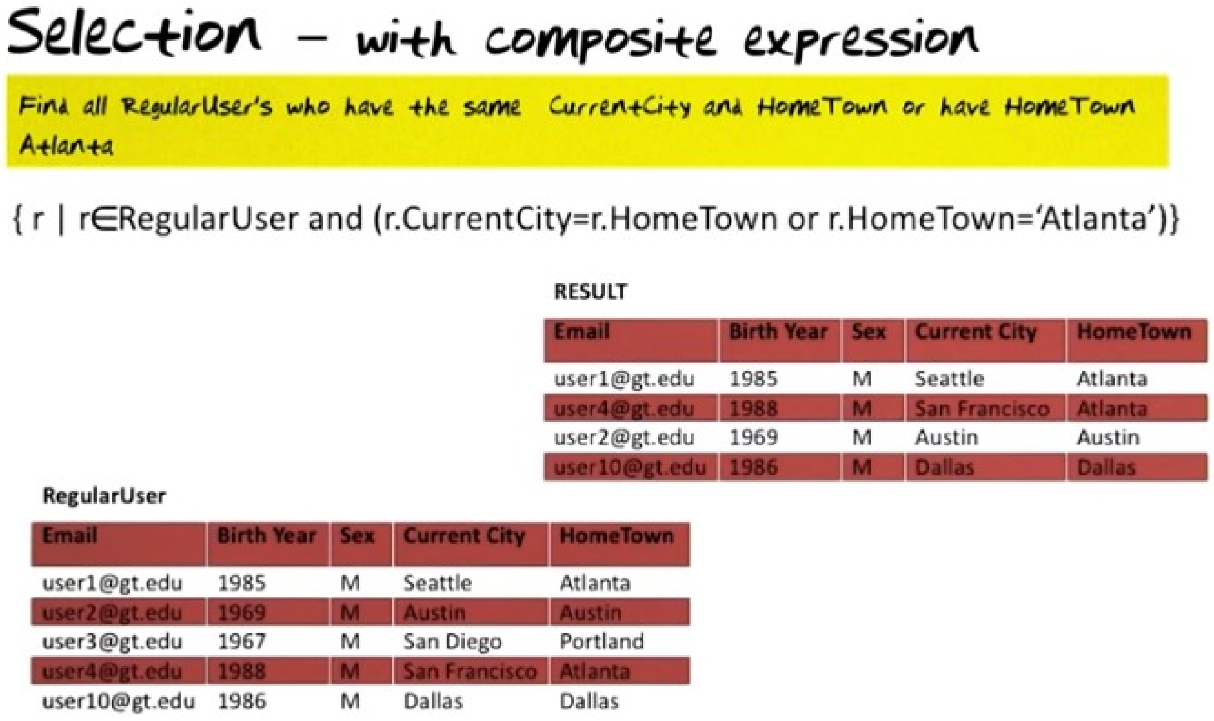
RESULT

Email	Birth Year	Sex	Current City	HomeTown
user2@gt.edu	1969	M	Austin	Austin
user3@gt.edu	1967	M	San Diego	Portland
user9@gt.edu	1988	F	Las Vegas	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas
user12@gt.edu	1974	F	College Park	Austin

RegularUser

Email	Birth Year	Sex	Current City	HomeTown
user2@gt.edu	1969	M	Austin	Austin
user3@gt.edu	1967	M	San Diego	Portland
user9@gt.edu	1988	F	Las Vegas	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas
user12@gt.edu	1974	F	College Park	Austin

The above figure: now let's take a look at the first very very simple operation expressed in tuple calculus, namely that of selection.



Selection – with composite expression

Find all RegularUser's who have the same CurrentCity and HomeTown or have HomeTown Atlanta

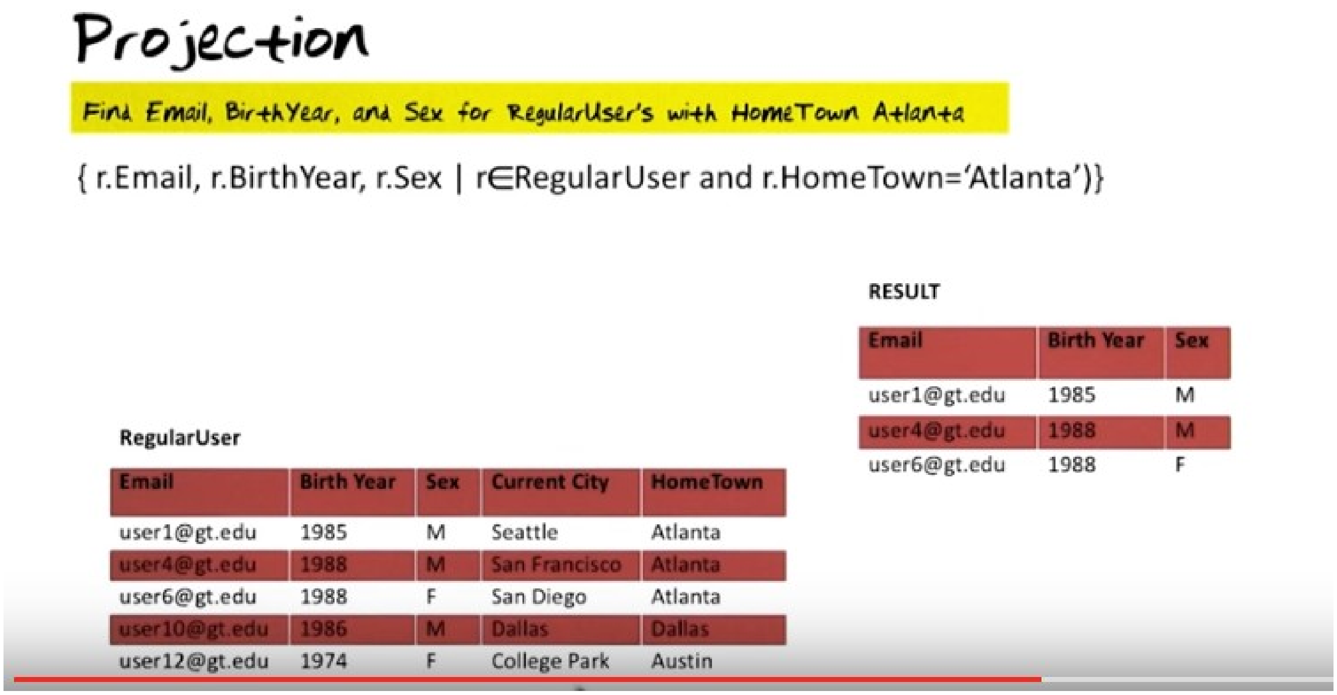
{ r | r∈RegularUser and (r.CurrentCity=r.HomeTown or r.HomeTown='Atlanta')}

RESULT

Email	Birth Year	Sex	Current City	HomeTown
user1@gt.edu	1985	M	Seattle	Atlanta
user4@gt.edu	1988	M	San Francisco	Atlanta
user2@gt.edu	1969	M	Austin	Austin
user10@gt.edu	1986	M	Dallas	Dallas

RegularUser

Email	Birth Year	Sex	Current City	HomeTown
user1@gt.edu	1985	M	Seattle	Atlanta
user2@gt.edu	1969	M	Austin	Austin
user3@gt.edu	1967	M	San Diego	Portland
user4@gt.edu	1988	M	San Francisco	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas



Projection

Find Email, BirthYear, and Sex for RegularUser's with HomeTown Atlanta

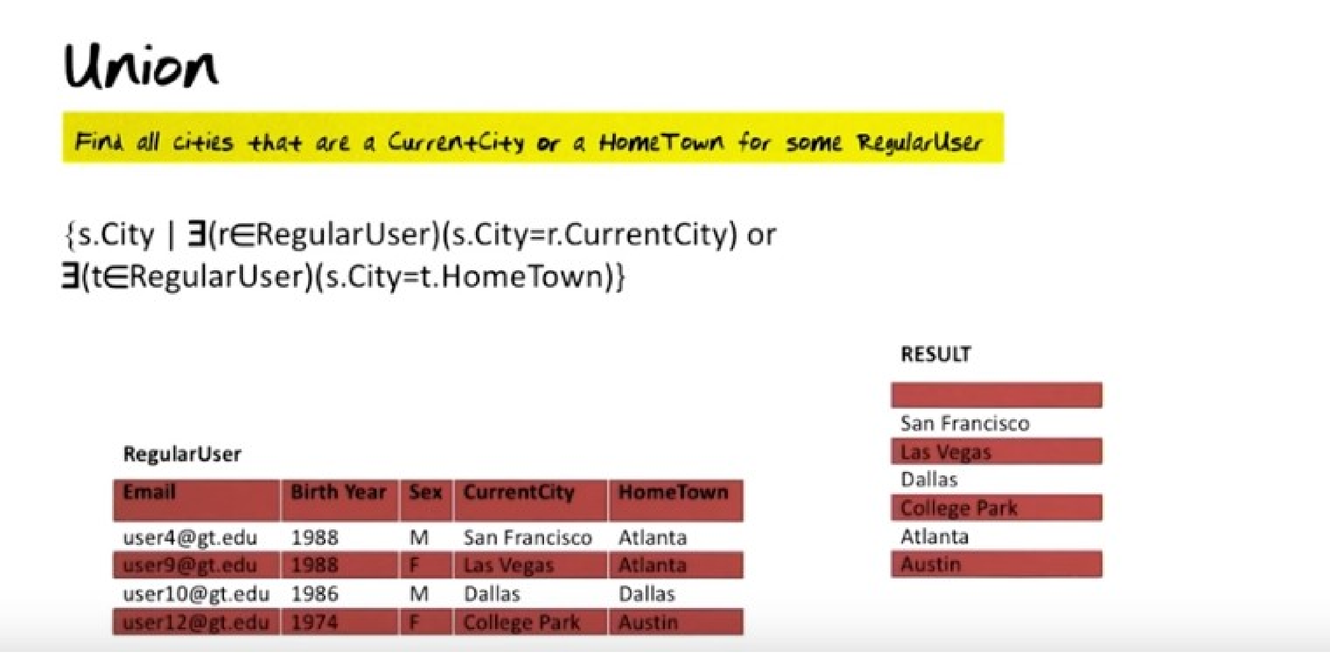
{ r.Email, r.BirthYear, r.Sex | r∈RegularUser and r.HomeTown='Atlanta')}

RegularUser

Email	Birth Year	Sex	Current City	HomeTown
user1@gt.edu	1985	M	Seattle	Atlanta
user4@gt.edu	1988	M	San Francisco	Atlanta
user6@gt.edu	1988	F	San Diego	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas
user12@gt.edu	1974	F	College Park	Austin

RESULT

Email	Birth Year	Sex
user1@gt.edu	1985	M
user4@gt.edu	1988	M
user6@gt.edu	1988	F



Union

Find all cities that are a CurrentCity or a HomeTown for some RegularUser

{s.City | ∃(r∈RegularUser)(s.City=r.CurrentCity) or
∃(t∈RegularUser)(s.City=t.HomeTown)}

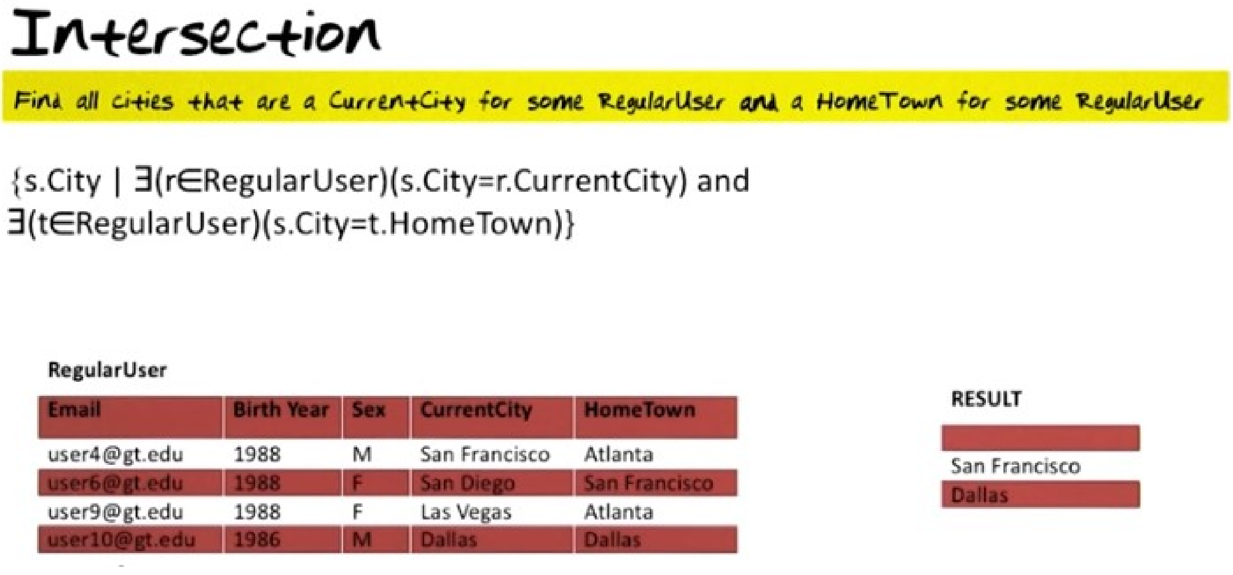
RegularUser

Email	Birth Year	Sex	CurrentCity	HomeTown
user4@gt.edu	1988	M	San Francisco	Atlanta
user9@gt.edu	1988	F	Las Vegas	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas
user12@gt.edu	1974	F	College Park	Austin

RESULT

San Francisco
Las Vegas
Dallas
College Park
Atlanta
Austin

上圖的 s.City 的那個表達式看起來複雜, 其實很簡單. 不用管 s 是甚麼, 把 s.City 理解為一個 city 就可以了, 這個 city 可不斷地變, 直到存在一個 r, 使得它的 r.CurrentCity 等於這個 city(即 s.City), 這個 city 就是一個 CurrentCity, 即就是我們想要的.



Intersection

Find all cities that are a CurrentCity for some RegularUser and a HomeTown for some RegularUser

{s.City | ∃(r∈RegularUser)(s.City=r.CurrentCity) and
∃(t∈RegularUser)(s.City=t.HomeTown)}

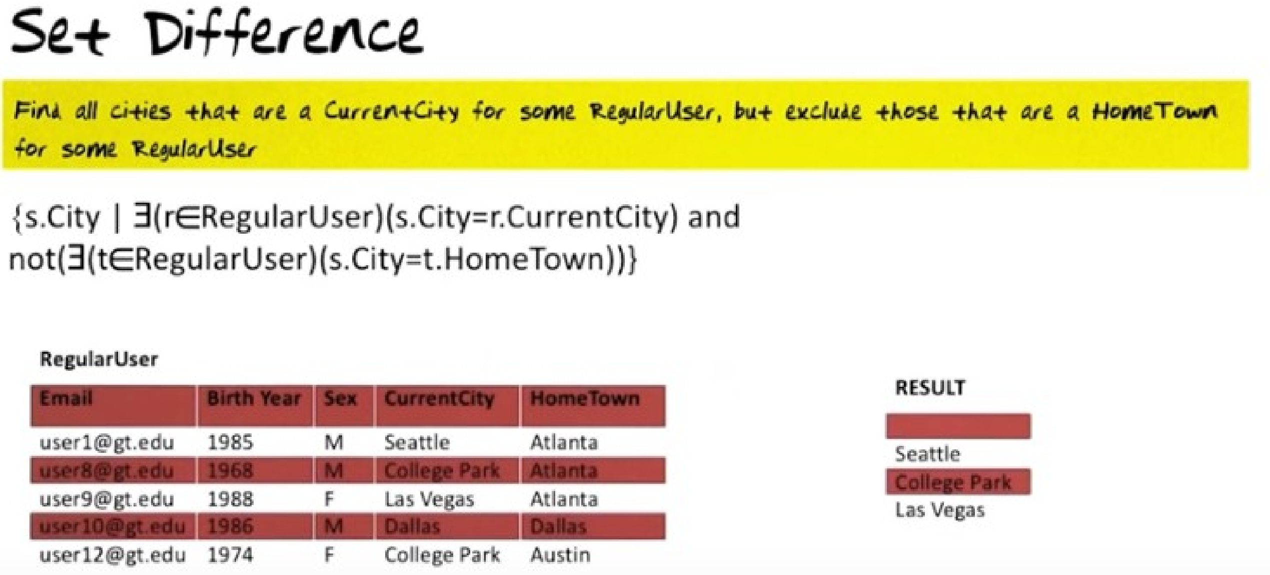
RegularUser

Email	Birth Year	Sex	CurrentCity	HomeTown
user4@gt.edu	1988	M	San Francisco	Atlanta
user6@gt.edu	1988	F	San Diego	San Francisco
user9@gt.edu	1988	F	Las Vegas	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas

RESULT

San Francisco
Dallas

上圖中的那句黃話: Find all cities that are a CurrentCity for some RegularUser and a HomeTown for some RegularUser. 其中的 some RegularUser 和 some RegularUser 可以是不同的 RegularUser. 上圖中的表達式跟上上圖(Union)中的表達式唯一的區別就是 or 換成了 and.



Set Difference

Find all cities that are a CurrentCity for some RegularUser, but exclude those that are a HomeTown for some RegularUser

{s.City | ∃(r∈RegularUser)(s.City=r.CurrentCity) and
not(∃(t∈RegularUser)(s.City=t.HomeTown))}

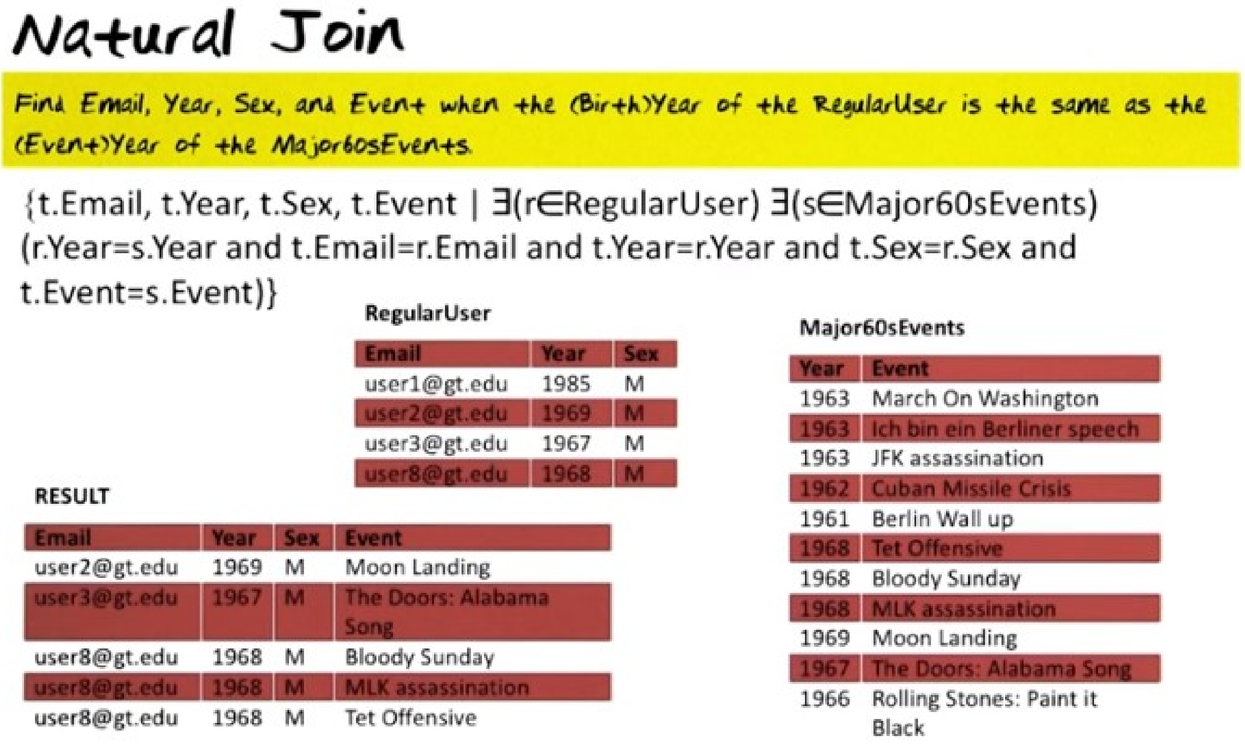
RegularUser

Email	Birth Year	Sex	CurrentCity	HomeTown
user1@gt.edu	1985	M	Seattle	Atlanta
user8@gt.edu	1968	M	College Park	Atlanta
user9@gt.edu	1988	F	Las Vegas	Atlanta
user10@gt.edu	1986	M	Dallas	Dallas
user12@gt.edu	1974	F	College Park	Austin

RESULT

Seattle
College Park
Las Vegas

上圖中的表達式跟上上圖(Intersection)中的表達式唯一的區別就是多了個 not.



Natural Join

Find Email, Year, Sex, and Event when the (Birth)Year of the RegularUser is the same as the (Event)Year of the Major60sEvents.

{t.Email, t.Year, t.Sex, t.Event | ∃(r∈RegularUser) ∃(s∈Major60sEvents) (r.Year=s.Year and t.Email=r.Email and t.Year=r.Year and t.Sex=r.Sex and t.Event=s.Event)}

RegularUser

Email	Year	Sex
user1@gt.edu	1985	M
user2@gt.edu	1969	M
user3@gt.edu	1967	M
user8@gt.edu	1968	M

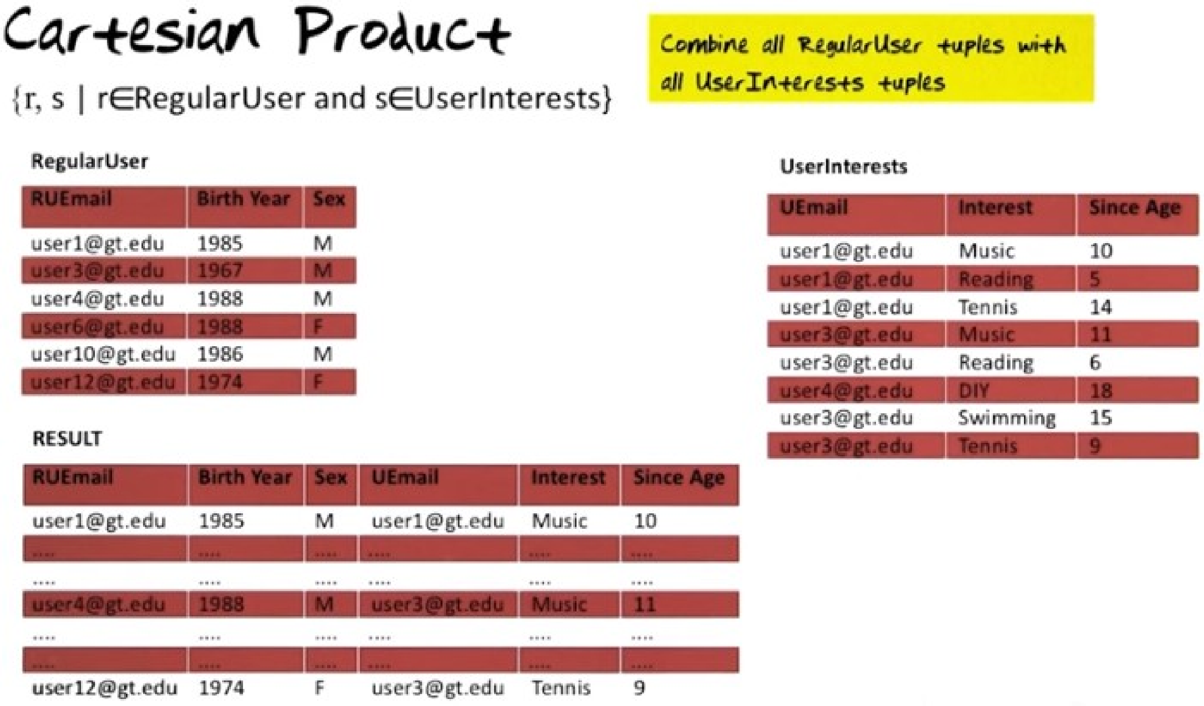
Major60sEvents

Year	Event
1963	March On Washington
1963	Ich bin ein Berliner speech
1963	JFK assassination
1962	Cuban Missile Crisis
1961	Berlin Wall up
1968	Tet Offensive
1968	Bloody Sunday
1968	MLK assassination
1969	Moon Landing
1967	The Doors: Alabama Song
1966	Rolling Stones: Paint it Black

RESULT

Email	Year	Sex	Event
user2@gt.edu	1969	M	Moon Landing
user3@gt.edu	1967	M	The Doors: Alabama Song
user8@gt.edu	1968	M	Bloody Sunday
user8@gt.edu	1968	M	MLK assassination
user8@gt.edu	1968	M	Tet Offensive

上圖的表達式看起來複雜, 其實也很簡單. 表達式第二中重要的是 r.Year = s.Year 這一項, 因為它是 join 的要求條件. 剩下的都是賦值, 如 t.Email = r.Email 表示 t.Email 取 r.Email 的值, t.Event = s.Event 表示 t.Event 最 s.Event 的值(用 s 是因為只有 s 中才有 Event, s 是 Major60sEvents 中的一行).



Cartesian Product

Combine all RegularUser tuples with all UserInterests tuples

{r, s | r∈RegularUser and s∈UserInterests}

RegularUser

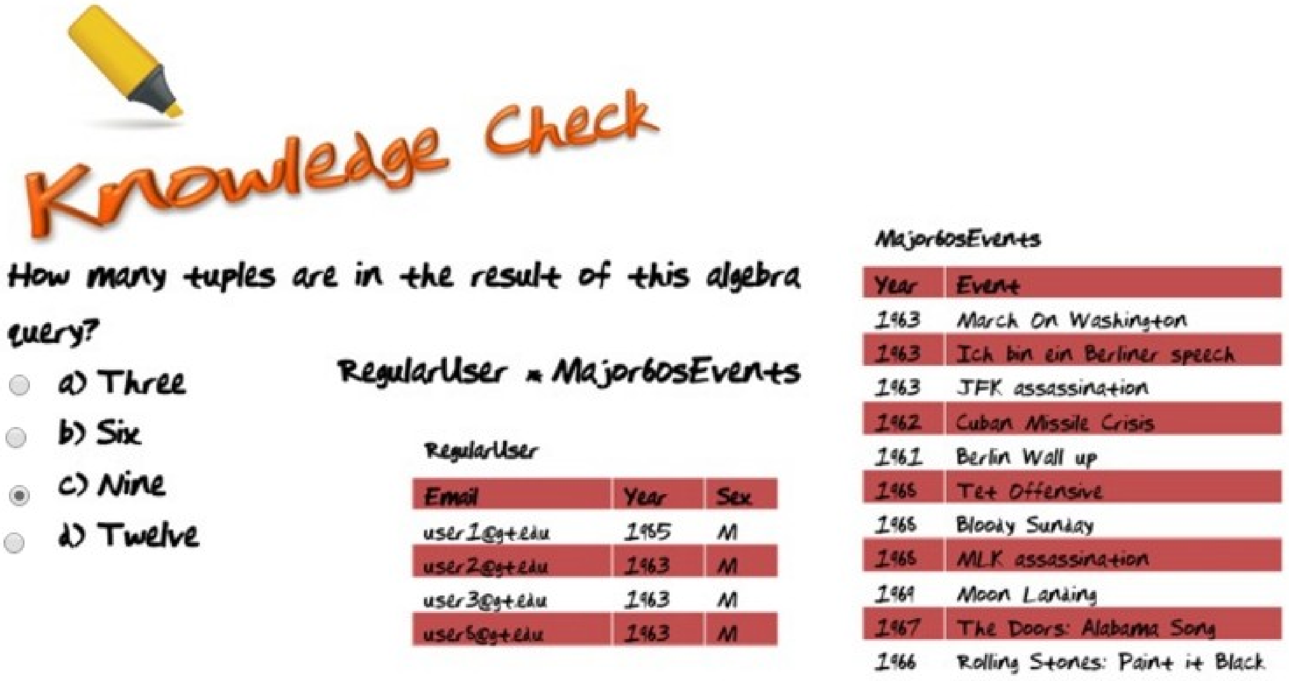
RUEmail	Birth Year	Sex
user1@gt.edu	1985	M
user3@gt.edu	1967	M
user4@gt.edu	1988	M
user6@gt.edu	1988	F
user10@gt.edu	1986	M
user12@gt.edu	1974	F

UserInterests

UEmail	Interest	Since Age
user1@gt.edu	Music	10
user1@gt.edu	Reading	5
user1@gt.edu	Tennis	14
user3@gt.edu	Music	11
user3@gt.edu	Reading	6
user4@gt.edu	DIY	18
user3@gt.edu	Swimming	15
user3@gt.edu	Tennis	9

RESULT

RUEmail	Birth Year	Sex	UEmail	Interest	Since Age
user1@gt.edu	1985	M	user1@gt.edu	Music	10
....	....	....	....	....	....
....	....	....	....	....	....
user4@gt.edu	1988	M	user3@gt.edu	Music	11
....	....	....	....	....	....
....	....	....	....	....	....
user12@gt.edu	1974	F	user3@gt.edu	Tennis	9



Knowledge Check

How many tuples are in the result of this algebra query?

○ a) Three

○ b) Six

● c) Nine

○ d) Twelve

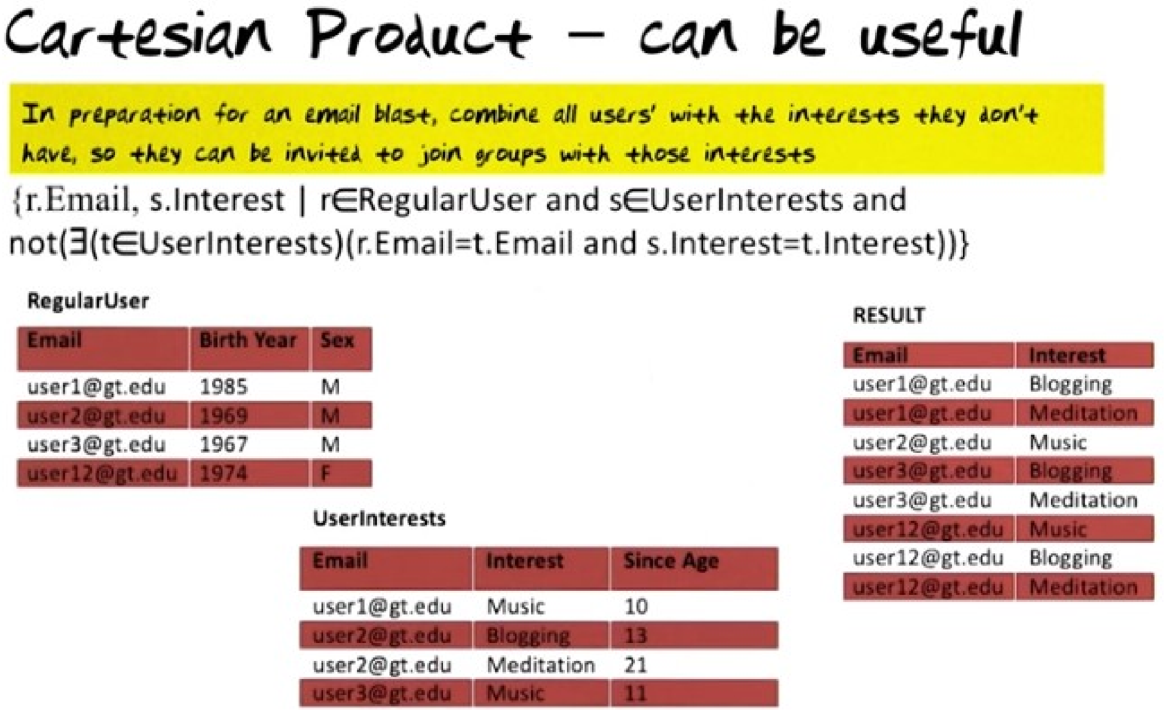
RegularUser × Major60sEvents

RegularUser

Email	Year	Sex
user1@gt.edu	1965	M
user2@gt.edu	1963	M
user3@gt.edu	1963	M
user6@gt.edu	1963	M

Major60sEvents

Year	Event
1963	March On Washington
1963	Ich bin ein Berliner speech
1963	JFK assassination
1962	Cuban Missile Crisis
1961	Berlin Wall up
1968	Tet Offensive
1968	Bloody Sunday
1968	MLK assassination
1969	Moon Landing
1967	The Doors: Alabama Song
1966	Rolling Stones: Paint it Black



Cartesian Product – can be useful

In preparation for an email blast, combine all users' with the interests they don't have, so they can be invited to join groups with those interests

{r.Email, s.Interest | r∈RegularUser and s∈UserInterests and not(∃(t∈UserInterests)(r.Email=t.Email and s.Interest=t.Interest))}

RegularUser

Email	Birth Year	Sex
user1@gt.edu	1985	M
user2@gt.edu	1969	M
user3@gt.edu	1967	M
user12@gt.edu	1974	F

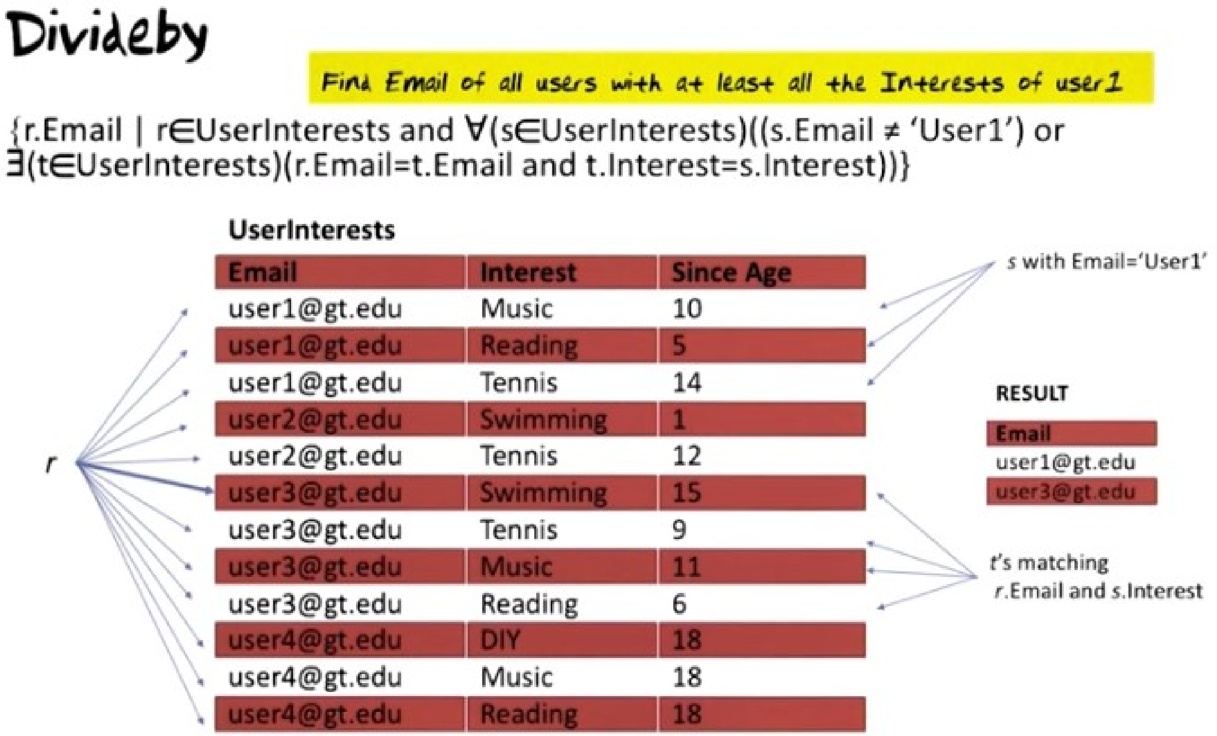
UserInterests

Email	Interest	Since Age
user1@gt.edu	Music	10
user2@gt.edu	Blogging	13
user2@gt.edu	Meditation	21
user3@gt.edu	Music	11

RESULT

Email	Interest
user1@gt.edu	Blogging
user1@gt.edu	Meditation
user2@gt.edu	Music
user3@gt.edu	Blogging
user3@gt.edu	Meditation
user12@gt.edu	Music
user12@gt.edu	Blogging
user12@gt.edu	Meditation

上圖表達式中的 and 之前的那段就是 cartesian product 的表達式, and 之後的也很好理解.



Divideby

Find Email of all users with at least all the Interests of user1

{r.Email | r∈UserInterests and ∀(s∈UserInterests)((s.Email ≠ 'User1') or ∃(t∈UserInterests)(r.Email=t.Email and t.Interest=s.Interest))}

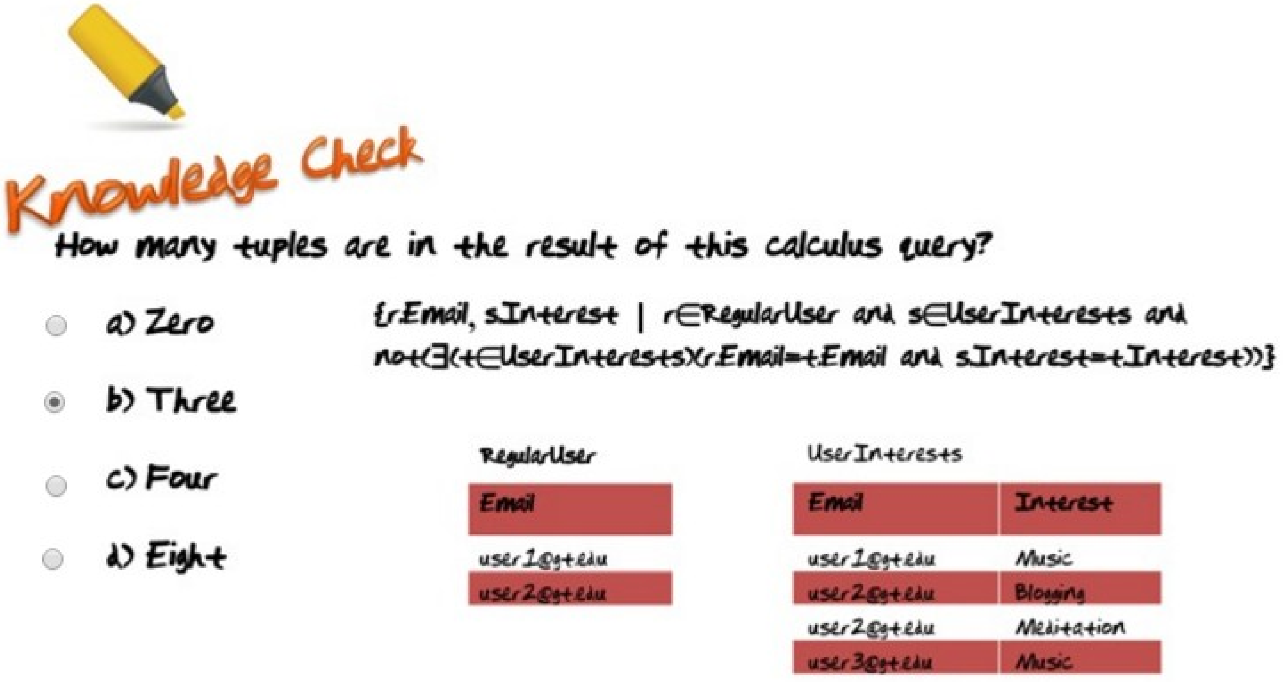
UserInterests

Email	Interest	Since Age
user1@gt.edu	Music	10
user1@gt.edu	Reading	5
user1@gt.edu	Tennis	14
user2@gt.edu	Swimming	1
user2@gt.edu	Tennis	12
user3@gt.edu	Swimming	15
user3@gt.edu	Tennis	9
user3@gt.edu	Music	11
user3@gt.edu	Reading	6
user4@gt.edu	DIY	18
user4@gt.edu	Music	18
user4@gt.edu	Reading	18

RESULT

Email
user1@gt.edu
user3@gt.edu

上圖表達式之理解: 主要是看 or 的兩個 operand. or 之後的那段其實暗含了 s.Email = 'User1'這個要求. 至於 or 之前那段 為何當 s.Email != 'User1'時沒通過 or, 還沒想清楚.



Knowledge Check

How many tuples are in the result of this calculus query?

○ a) Zero

● b) Three

○ c) Four

○ d) Eight

{r.Email, s.Interest | r∈RegularUser and s∈UserInterests and not(∃(t∈UserInterests)(r.Email=t.Email and s.Interest=t.Interest))}

RegularUser

Email
user1@gt.edu
user2@gt.edu

UserInterests

Email	Interest
user1@gt.edu	Music
user2@gt.edu	Blogging
user2@gt.edu	Meditation
user3@gt.edu	Music

上題中的表達式其實就是 Cartesian Product – can be useful 中的表達式, 即自己沒有的興趣. 那三個 tuple 為:

user1@gt.edu	Blogging
user1@gt.edu	Meditation

user2@gt.edu	Music

之所以無

user3@gt.edu	Blogging
user3@gt.edu	Meditation

是因為 user3 不在 RegularUser 中