Database Management Systems

Mathematical Preliminaries

Malay Bhattacharyya

Assistant Professor

Machine Intelligence Unit
and
Centre for Artificial Intelligence and Machine Learning
Indian Statistical Institute, Kolkata

April, 2021

Preliminaries

Relation: Given the sets $X_1, X_2, \ldots, X_n \subseteq \mathbb{R}$ (the real plane), a relation $\mathcal{R}$ can be defined on $X_1, X_2, \ldots, X_n$ as
$\mathcal{R} = \{(x_1, x_2, \ldots, x_n) : (x_1, x_2, \ldots, x_n) \in X_1 \times X_2 \times \cdots \times X_n\}$.

If the sets denote different attributes in a database then a table represents nothing but a relation (subset of the Cartesian product of attributes) between the attributes.

Based on this, we can assume:
A **relation** is a table
The **attributes** are the headers of the table
A **tuple** is a row.

Preliminaries

Example of a relation:

Table: MATH_OLYMPIC

Year	Gold	Silver
2008	0	0
2009	0	3
2010	0	2
2011	1	1
2012	2	3
2013	0	2
2014	0	1
2015	0	1
2016	0	1
2017	0	0
2018	0	3
2019	1	4

Preliminaries

Example of another relation:

Table: MATH_OLYMPIC_GOLDEN

Year	Gold	Silver
2011	1	1
2012	2	3
2019	1	4

Preliminaries of relational algebra

Query language: A language for manipulation and retrieval of data from a database.

* Query languages can be – procedural (user provides requirements along with instructions) or non-procedural/declarative (user provides requirements only).

The relational algebra is a procedural query language

The relational algebra works on relations.

Note: Tuple relational calculus and domain relational calculus are non-procedural.

Preliminaries of relational algebra

Relational algebra is *closed* because every operation in relational algebra returns a relation.

Relational algebra is not "Turing complete". This is inevitably favourable because it manifests that relational algebra is subject to algorithmic analysis (to be precise for query optimization).

Union

Notation: $R_1 \cup R_2$, where R_1, R_2 are relational algebra expressions.

Description: Returns tuples that appear in either or both of the two relations, thereby producing a relation with at most $\mathcal{T}(R_1) + \mathcal{T}(R_2)$ tuples.

Note: Union operation is valid iff the attributes of R_1 and R_2 are the same, i.e. $\mathcal{A}(R_1) = \mathcal{A}(R_2)$.

Union

Example: MATH_OLYMPIC $\cup$ MATH_OLYMPIC_GOLDEN

Year	Gold	Silver
2008	0	0
2009	0	3
2010	0	2
2011	1	1
2012	2	3
2013	0	2
2014	0	1
2015	0	1
2016	0	1
2017	0	0
2018	0	3
2019	1	4

Intersection

Notation: $R_1 \cap R_2$, where R_1, R_2 are relational algebra expressions.

Description: Returns tuples that appear in both the relations, thereby producing a relation with at most $\min(\mathcal{T}(R_1), \mathcal{T}(R_2))$ tuples.

Note: Intersection operation is valid iff the attributes of R_1 and R_2 are the same, i.e. $\mathcal{A}(R_1) = \mathcal{A}(R_2)$.

Intersection

Example: MATH_OLYMPIC $\cap$ MATH_OLYMPIC_GOLDEN

Year	Gold	Silver
2011	1	1
2012	2	3
2019	1	4

Difference

Notation: $R_1 - R_2$, where R_1, R_2 are relational algebra expressions.

Description: Returns the tuples that appear in one relation (first one) but not in the other (second one), thereby producing a relation with at most $\mathcal{T}(R_1)$ tuples.

Note: Difference operation is valid iff the attributes of R_1 and R_2 are the same, i.e. $\mathcal{A}(R_1) = \mathcal{A}(R_2)$.

Difference

Example: MATH_OLYMPIC − MATH_OLYMPIC_GOLDEN

Year	Gold	Silver
2008	0	0
2009	0	3
2010	0	2
2013	0	2
2014	0	1
2015	0	1
2016	0	1
2017	0	0
2018	0	3

Difference

Lemma

Given a pair of relations R_1 and R_2, the set difference operation $R_1 - R_2$ monotonically increases with respect to R_1 but monotonically decreases with respect to R_2.

Difference

Lemma

Given a pair of relations R_1 and R_2, the set difference operation $R_1 - R_2$ monotonically increases with respect to R_1 but monotonically decreases with respect to R_2.

Proof: Suppose a new tuple t is added to R_1, without affecting R_2. Then the number of tuples in $R_1 - R_2$ will either remain the same or increase based on whether t was already there in R_2 or not, respectively. On the other hand, suppose a new tuple t is added to R_2, without affecting R_1. Then the number of tuples in $R_1 - R_2$ will either decrease or remain the same based on whether t was already there in R_1 or not, respectively. Hence, the lemma.

Cartesian product / Cross join

Notation: $R_1 \times R_2$, where R_1, R_2 are relational algebra expressions.

Description: Returns the Cartesian product of two relations, thereby producing a relation with attributes $\mathcal{A}(R_1) \cup \mathcal{A}(R_2)$ and $\mathcal{T}(R_1) * \mathcal{T}(R_2)$ number of tuples.

Note: No validity constraint.

Cartesian product / Cross join

Example: MATH_OLYMPIC $\times$ MATH_OLYMPIC_GOLDEN

M.Year	M.Gold	M.Silver	M_G.Year	M_G.Gold	M_G.Silver
2008	0	0	2011	1	1
2008	0	0	2012	2	3
2009	0	3	2011	1	1
2009	0	3	2012	2	3
2010	0	2	2011	1	1
2010	0	2	2012	2	3
2011	1	1	2011	1	1
2011	1	1	2012	2	3
2012	2	3	2011	1	1
2013	2	3	2012	2	3
...	...	...	...	...	...
2019	1	4	2011	1	1
2019	1	4	2012	2	3
2019	1	4	2019	1	4

Cartesian product / Cross join

Lemma

Cartesian product monotonically increases with respect to the relations on which they are applied in relational algebra.

Cartesian product / Cross join

Lemma

Cartesian product monotonically increases with respect to the relations on which they are applied in relational algebra.

Proof: Let there be four relations R_1, R_2, R_3 and R_4 such that $R_1 \subseteq R_2$ and $R3 \subseteq R4$. Consider any arbitrary element $(x, y) \in R_1 \times R3$. Given $R_1 \subseteq R_2$ and $R3 \subseteq R4$, we can show $(x, y) \in R_1 \times R3 \subseteq R_2 \times R3 \subseteq R_2 \times R4$.
Hence, for any arbotrary quadruplet of relations R_1, R_2, R_3 and R_4, we can write

$$R_1 \subseteq R_2 \wedge R_3 \subseteq R_4 \Rightarrow R_1 \times R_3 \subseteq R_2 \times R_4.$$

This in turn proves the monotonic increase of Cartesian product in relational algebra.

Let us brainstorm!!!

Suppose there exists a pair of relations $R_1(X, Y)$ and $R_2(X, Y)$ having $t_1 > 0$ and $t_2 > 0$ tuples, respectively. Consider that X and Y take integer values only. Without making any further assumptions, find out the minimum and maximum possible number of tuples that may appear in the resulting relations provided by the following operations.

i) $R_1 \cup R_2$

ii) $R_1 \cap R_2$

iii) $R_1 - R_2$

iv) $R_1 \times R_2$

Let us brainstorm!!!

For arbitrary relations, without assumption on keys, tighter bounds are as follows.

Expression	Minimum tuples	Maximum tuples
$R_1 \cup R_2$	$\max(t_1, t_2)$	$t_1 + t_2$
$R_1 \cap R_2$	0	$\min(t_1, t_2)$
$R_1 - R_2$	0	t_1
$R_1 \times R_2$	$t_1 t_2$	$t_1 t_2$

Selection

Notation: $\sigma_P(R)$, where P is a predicate on the attributes of the relation R.

Description: Returns the tuples that satisfy a given predicate (extracts a subset of tuples).

Selection

Example: $\sigma_{\text{Gold}\neq 0}(\text{MATH\_OLYMPIC})$

Year	Gold	Silver
2011	1	1
2012	2	3
2019	1	4

Example: $\sigma_{\text{Gold}\neq 0 \wedge \text{Silver}>1}(\text{MATH\_OLYMPIC})$

Year	Gold	Silver
2012	2	3
2019	1	4

Projection

Notation: $\pi_S(R)$, where S is a subset of the attributes in the relation R.

Description: Returns all tuples with the given attributes only (extracts a subset of attributes).

Note: A projection returns the distinct tuples (after removing duplicates) only.

Projection

Example: $\pi_{\text{Gold,Silver}}(\text{MATH\_OLYMPIC})$

Gold	Silver
0	0
0	3
0	2
1	1
2	3
0	1
1	4

Example: $\pi_{\text{Year,Silver}}(\sigma_{\text{Gold}>1}(\text{MATH\_OLYMPIC}))$

Year	Silver
2012	3

Rename

Notation: $\rho_N(R)$, where N is the new name for the result of R.

Description: Renames a relation in relational algebra.

Rename – A caution

Rename

Example: $\rho_{IMO}(\text{MATH\_OLYMPIC})$

Table: IMO

Year	Gold	Silver
2008	0	0
2009	0	3
2010	0	2
2011	1	1
2012	2	3
2013	0	2
2014	0	1
2015	0	1
2016	0	1
2017	0	0
2018	0	3
2019	1	4

Natural join

Notation: $R_1 \bowtie R_2$, where R_1, R_2 are relations obtained from relational algebra operations.

Description: Cartesian product of two relations followed by the removal of duplicate attributes.

Note: If we consider the pair of relations R_1 and R_2, then the natural join between them $(R_1 \bowtie R_2)$ is a relation on schema $\mathcal{A}(R_1) \cup \mathcal{A}(R_2)$ such that

$$R_1 \bowtie R_2 = \pi_{\mathcal{A}(R_1) \cup \mathcal{A}(R_2))}(\sigma_{\mathcal{A}_1(R_1)=\mathcal{A}_1(R_2) \wedge \ldots \wedge \mathcal{A}_n(R_1)=\mathcal{A}_n(R_2)}(R_1 \times R_2)).$$

The selection is defined on the common set of attributes between R_1 and R_2, i.e., $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n \in \mathcal{A}(R_1) \cap \mathcal{A}(R_2)$. Hence, natural join reduces to Cartesian product if no attribute is common.

Natural join

Example: $\pi_{\text{Year}}(\text{IMO} \bowtie \text{MATH\_OLYMPIC\_GOLD})$

Year
2011
2012
2019

Natural join – A deeper look

Table: SYM

A1	A2
1	pi
2	e

Table: VAL

A2	A3
pi	22/7
pi	333/106

Table: $\sigma_{SYM.A2=VAL.A2}(SYM \times \text{VAL})$

SYM.A1	SYM.A2	VAL.A2	VAL.A3
1	pi	pi	22/7
1	pi	pi	333/106
~~2~~	~~e~~	~~pi~~	~~22/7~~
~~2~~	~~e~~	~~pi~~	~~333/106~~

Table: SYM $\bowtie$ VAL

A1	A2	A3
1	pi	22/7
1	pi	333/106

Theta join

Notation: $R_1 \bowtie_\theta R_2$, where R_1, R_2 are relations obtained from relational algebra operations.

Description: Cartesian product of two relations followed by selection operation. We can write

$$R_1 \bowtie_\theta R_2 = \sigma_\theta(R_1 \times R_2).$$

Note: The result of theta join is defined only if the attributes of the relations are disjoint.

EQUI join

Notation: $R_1 \bowtie_{=} R_2$, where R_1, R_2 are relations obtained from relational algebra operations.

Description: Cartesian product of two relations followed by selection operation with respect to equity. EQUI join is a special case of theta join where $\theta = `` ='' $.

Division

Notation: $R_1 \div R_2$, where R_1, R_2 are relations obtained from relational algebra operations.

Description: Satisfies universal specification.

Note: Division operation is valid iff the attributes of R_2 is a proper subset of R_1, i.e. $\mathcal{A}(R_2) \subset \mathcal{A}(R_1)$. A tuple is said to be in $R_1 \div R_2$ iff the tuple is in $\pi_{\mathcal{A}(R_1)-\mathcal{A}(R_2)}(R_1)$ and its Cartesian product with any arbitrary tuple in R_2 produces a tuple that belongs to R_1. Interestingly, we can represent the division operation as follows

$$\begin{aligned} R_1 \div R_2 \quad = \quad & \pi_{\mathcal{A}(R_1)-\mathcal{A}(R_2)}(R_1) - \\ & \pi_{\mathcal{A}(R_1)-\mathcal{A}(R_2)}((\pi_{\mathcal{A}(R_1)-\mathcal{A}(R_2)}(R_1) \times R_2) - R_1). \end{aligned}$$

Division

Example: Let us consider the following pair of relations.

Table: CODE

Roll	Coding	Feature
1	Python	Programming
2	C	Programming
2	R	Programming
3	Python	Programming
3	Python	Visualization
4	C++	Programming
5	R	Visualization

Table: SKILL

Feature
Programming
Visualization

CODE $\div$ SKILL

Roll	Coding
3	Python

Division – A deeper look

Table: $\pi_{\mathcal{A}(CODE)-\mathcal{A}(SKILL)}(CODE) \times SKILL$

Roll	Coding	Feature
1	Python	Programming
1	Python	Visualization
2	C	Programming
2	C	Visualization
2	R	Programming
2	R	Visualization
3	Python	Programming
3	Python	Visualization
4	C++	Programming
4	C++	Visualization
5	R	Programming
5	R	Visualization

Note: $\mathcal{A}(CODE) - \mathcal{A}(SKILL)$ includes the attributes {Roll, Coding}.

Division – A deeper look

Table: $(\pi_{\mathcal{A}(CODE)-\mathcal{A}(SKILL)}(CODE) \times SKILL) - CODE$

Roll	Coding	Feature
1	Python	Visualization
2	C	Visualization
2	R	Visualization
4	C++	Visualization
5	R	Programming

Table:
$\pi_{\mathcal{A}(CODE)-\mathcal{A}(SKILL)}((\pi_{\mathcal{A}(CODE)-\mathcal{A}(SKILL)}(CODE) \times SKILL) - CODE)$

Roll	Coding
1	Python
2	C
2	R
4	C++
5	R

Division – A deeper look

Table: $\pi_{\mathcal{A}(CODE)-\mathcal{A}(SKILL)}(CODE)$

Roll	Coding
1	Python
2	C
2	R
3	Python
4	C++
5	R

Table: $\pi_{\mathcal{A}(CODE)-\mathcal{A}(SKILL)}(CODE) - \pi_{\mathcal{A}(CODE)-\mathcal{A}(SKILL)}((\pi_{\mathcal{A}(CODE)-\mathcal{A}(SKILL)}(CODE) \times SKILL) - CODE)$

Roll	Coding
3	Python

Assignment

Notation: $var \leftarrow R$, where var is a variable and R is a relation obtained from relational algebra operations

Description: Assigns a relational algebra expression to a relational variable

Example: Gold $\leftarrow E$

Inner join – Basics

Inner join is a generalized representation of natural join operation. The following pair of relational algebra expressions are the same.

$$R_1 \bowtie R_2$$

* Implicitly uses the common attributes to join

$$\sigma_P(R_1 \times R_2)$$

* The common attributes are to be mentioned in P

Outer join – Basics

Outer join has been extended from the natural join operation for avoiding information loss. Let us consider the following pair of relations.

Table: FAC

Name	**Unit**	**Centre**
Malay	MIU	Kolkata
Mandar	CVPRU	Kolkata
Ansuman	ACMU	Kolkata
Sandip	ACMU	Kolkata

Table: RES

Name	**Area**	**Level**
Malay	CB	Junior
Mandar	IR	Senior
Sasthi	WSN	Senior
Sandip	DM	Senior

Outer join – Motivation

Example: FAC $\bowtie$ RES

Name	Unit	Centre	Area	Level
Malay	MIU	Kolkata	CB	Junior
Mandar	CVPRU	Kolkata	IR	Senior
Sandip	ACMU	Kolkata	DM	Senior

The information about *Ansuman* and *Sasthi* are lost.

Outer join – Left outer join / Left join

Notation: $R_1 ⟕ R_2$, where R_1, R_2 are relations obtained from relational algebra operations

Description: Makes a natural join but adds extra tuples to the result padded with NULL values to deal with missing information in the first relation

Outer join – Left outer join / Left join

Example: FAC ⟕ RES

Name	Unit	Centre	Area	Level
Malay	MIU	Kolkata	CB	Junior
Mandar	CVPRU	Kolkata	IR	Senior
Sandip	ACMU	Kolkata	DM	Senior
Ansuman	ACMU	Kolkata	NULL	NULL

Outer join – Right outer join / Right join

Notation: $R_1 ⟖ R_2$, where R_1, R_2 are relations obtained from relational algebra operations

Description: Makes a natural join but adds extra tuples to the result padded with NULL values to deal with missing information in the second relation

Outer join – Right outer join / Right join

Example: FAC ⟖ RES

Name	Unit	Centre	Area	Level
Malay	MIU	Kolkata	CB	Junior
Mandar	CVPRU	Kolkata	IR	Senior
Sandip	ACMU	Kolkata	DM	Senior
Sasthi	NULL	NULL	WSN	Senior

Outer join – Full outer join / Full join

Notation: $R_1 ⟗ R_2$, where R_1, R_2 are relations obtained from relational algebra operations

Description: Makes a natural join but adds extra tuples to the result padded with NULL values to deal with missing information in both the relations

Outer join – Full outer join / Full join

Example: FAC $⟗$ RES

Name	Unit	Centre	Area	Level
Malay	MIU	Kolkata	CB	Junior
Mandar	CVPRU	Kolkata	IR	Senior
Sandip	ACMU	Kolkata	DM	Senior
Ansuman	ACMU	Kolkata	NULL	NULL
Sasthi	NULL	NULL	WSN	Senior

Outer join – A deeper look

Let us show that the intersection of left and outer joins reduces to natural join.

Note that, we can write $R_1 ⟕ R_2$ as follows

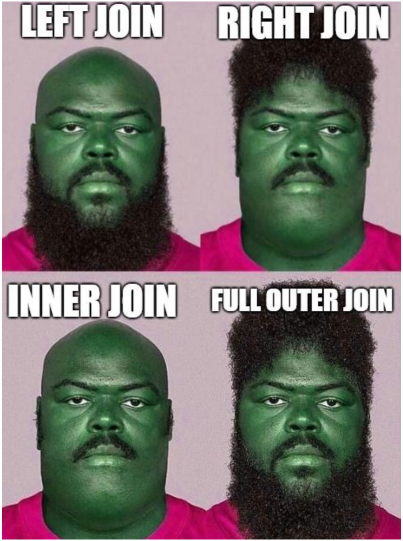
$$\pi_{\mathcal{A}(R_1)\cup\mathcal{A}(R_2)}(\sigma_{\mathcal{A}_1(R_1)=\mathcal{A}_1(R_2)\wedge\ldots\wedge\mathcal{A}_n(R_1)=\mathcal{A}_n(R_2)}(R_1 \times R_2))$$
$$\cup\ \pi_{\mathcal{A}(R_1)\cup\mathcal{A}(R_2)}(\sigma_{\mathcal{A}_1(R_2)=NULL\wedge\ldots\wedge\mathcal{A}_n(R_2)=NULL}(R_1 \times R_2)).$$

Similarly, we can write $R_1 ⟖ R_2$ as follows

$$\pi_{\mathcal{A}(R_1)\cup\mathcal{A}(R_2)}(\sigma_{\mathcal{A}_1(R_1)=\mathcal{A}_1(R_2)\wedge\ldots\wedge\mathcal{A}_n(R_1)=\mathcal{A}_n(R_2)}(R_1 \times R_2))$$
$$\cup\ \pi_{\mathcal{A}(R_1)\cup\mathcal{A}(R_2)}(\sigma_{\mathcal{A}_1(R_1)=NULL\wedge\ldots\wedge\mathcal{A}_n(R_1)=NULL}(R_1 \times R_2)).$$

Hence, by intersecting the two expressions stated above, we obtain the result.

Join operations – The interpretation



Join operations – The interpretation

Let us brainstorm!!!

Suppose there exists a pair of relations $R_1(X, Y)$ and $R_2(X, Y)$ having $t_1 > 0$ and $t_2 > 0$ tuples, respectively. Consider that X and Y take integer values only. Without making any further assumptions, find out the minimum and maximum possible number of tuples that may appear in the resulting relations provided by the following operations.

i) $\pi_Y(R_2)$

ii) $R_1 \div \pi_Y(R_2)$

iii) $(R_1 \bowtie R_2) \cup (R_2 \bowtie R_1)$

iv) $(R_1 - R_2) \cup (R_2 - R_1)$

v) $R_1 \bowtie (R_1 - R_2)$

Let us brainstorm!!!

For arbitrary relations, tighter bounds are as follows.

Expression	**Minimum tuples**	**Maximum tuples**
$\pi_Y(R_2)$	1	t_2
$R_1 \div \pi_Y(R_2)$	0	t_1/t_2
$(R_1 \bowtie R_2) \cup (R_2 \bowtie R_1)$	0	$\min(t_1, t_2)$
$(R_1 - R_2) \cup (R_2 - R_1)$	0	$t_1 + t_2$
$R_1 \bowtie (R_1 - R_2)$	0	t_1

Note: The natural join of a relation R with itself will return the original relation R.

Let us brainstorm!!!

Suppose there exists a pair of relations $R_1(X, Y)$ and $R_2(Y, Z)$ having $t_1 > 0$ and $t_2 > 0$ tuples, respectively. Consider that X and Y take integer values only. Without making any further assumptions, find out the minimum and maximum possible number of tuples that may appear in the resulting relations provided by the following operations.

i) $\pi_Y(\sigma_{X=0}(R_1)) - \pi_Y R_2$

ii) $\pi_Y R_1 - (\pi_Y R_1 - \pi_Y R_2)$

iii) $R_1 \cup \rho_{R_2(X,Y)} R_2$

iv) $\pi_{X,Z}(R_1 \bowtie R_2)$

v) $R_1 \bowtie (R_1 \bowtie R_1)$

vi) $\sigma_{X>Y} R_1 \cup \sigma_{X<Y} R_1$

Let us brainstorm!!!

For arbitrary relations, tighter bounds are as follows.

Expression	Minimum tuples	Maximum tuples
$\pi_Y(\sigma_{X=0}(R_1)) - \pi_Y R_2$	0	t_1
$\pi_Y R_1 - (\pi_Y R_1 - \pi_Y R_2)$	0	$\min(t_1, t_2)$
$R_1 \cup \rho_{R_2(X,Y)} R_2$	$\max(t_1, t_2)$	$t_1 + t_2$
$\pi_{X,Z}(R_1 \bowtie R_2)$	0	$\min(t_1, t_2)$
$R_1 \bowtie (R_1 \bowtie R_1)$	t_1	t_1
$\sigma_{X>Y} R_1 \cup \sigma_{X<Y} R_1$	0	t_1

Semijoin and antijoin

Semijoin and antijoin are two convenient but infrequently used join operations in relational algebra.

Semijoin and antijoin

Semijoin and antijoin are two convenient but infrequently used join operations in relational algebra.

Semijoin (demarcated with $\ltimes$) is alike natural join with the only exception that attributes in the first relation are returned in the result.

Semijoin and antijoin

Semijoin and antijoin are two convenient but infrequently used join operations in relational algebra.

Semijoin (demarcated with $\ltimes$) is alike natural join with the only exception that attributes in the first relation are returned in the result.

On the contrary, antijoin (demarcated with $\triangleright$) returns all tuples in the first relation such that there are no tuples in the second relation with matching values for the shared attributes.

Semijoin and antijoin

The equivalent relational algebra expressions used for semijoin and antijoin are as follows:

$$R_1 \ltimes R_2 = \pi_{A(R_1)}(R_1 \bowtie R_2)$$

$$R_1 \triangleright R_2 = R_1 - (R_1 \ltimes R_2) = R_1 - \pi_{A(R_1)}(R_1 \bowtie R_2)$$

Understanding the concepts in a better way

Try this out!!!

RelaX – relational algebra calculator:
`https://dbis-uibk.github.io/relax`

Completeness

A complete set comprises a subset of relational algebra operations that can express any other relational algebra operations.
E.g., the set $\{\sigma, \pi, \cup, -, \times\}$ is complete.

Completeness – An example

Let us show that the set of operations $\{\sigma, \pi, \rho, \cup, -, \times\}$ is complete.

As the given set already contains selection, projection and rename, it is sufficient to establish that the operations like set intersection, set division, and natural join can be performed from the rest.

Notably, $R_1 \cap R_2 = R_1 - (R_1 - R_2)$.

Further note that, $R_1 \div R_2 =$
$\pi_{\mathcal{A}(R_1)-\mathcal{A}(R_2)}(R_1) - \pi_{\mathcal{A}(R_1)-\mathcal{A}(R_2)}((\pi_{\mathcal{A}(R_1)-\mathcal{A}(R_2)}(R_1) \times R_2) - R_1)$.

Finally, $R_1 \bowtie R_2 =$
$\pi_{\mathcal{A}(R_1)\cup\mathcal{A}(R_2))}(\sigma_{\mathcal{A}_1(R_1)=\mathcal{A}_1(R_2)\wedge\ldots\wedge\mathcal{A}_n(R_1)=\mathcal{A}_n(R_2)}(R_1 \times R_2))$.

Hence, the set $\{\sigma, \pi, \rho, \cup, -, \times\}$ is complete.