Databases: The Relational Model and Relational Algebra

http://simshadows.com/
contact@simshadows.com

Last Updated: December 30, 2023

Relation Schemas

Relation schemas are finite sets of *attributes*.

Notation for some relation R with attributes A, B, C, D and primary key B, D:

$$R(A, \underline{B}, C, \underline{D})$$

The *domain* of a schema is the set of all possible tuples that can be associated with the schema.

For relation R, this is denoted:

$$\mathrm{Dom}\,(R)$$

Keys

A *superkey* of a relation schema is any subset of that schema that can uniquely identify a tuple.

In other words, all tuples have unique values in the superkey.

Or more formally, for $r(R)$ and any two distinct tuples $t_1, t_2 \in r$, if $t_1[K] \neq t_2[K]$ for some $K \subseteq R$, then K is a superkey.

Note that for $r(R)$, R is always a superkey.

A *candidate key* is a "minimal superkey".

More formally, superkey K is a candidate key if no subset $K' \subset K$ exists that is also a superkey.

A *primary key* is the specific candidate key designated to a relation schema.

All relations must have exactly one primary key.

A *foreign key* of a relation schema is a subset of attributes that reference a particular primary key.

The referenced primary key is often a different relation, but may also be of the same relation.

Attributes

Attributes are labels for parts of a schema.

The *domain* of an attribute is the set of values that can be associated with the attribute.

For attribute A, this is denoted:

$$\mathrm{Dom}\,(A)$$

Relation Instances

Relation instances are finite sets of *tuples*.

To show that r is an instance of schema R, we write:

$$r(R)$$

Example

relation, table

attributes, columns, fields

`Customers`

customerid	name	address
1774504	Amy	Surry Hills
4389167	Chris	Richmond
4622780	Josh	North Ryde
5691729	Sam	Kensington
9527291	Vanessa	Richmond

tuples, rows, records

Tuples

Tuples are mappings from schema to schema domain.

Concretely, a tuple can be thought of as a set of attribute-value pairs.

We will use the following notation in this cheatsheet:

$t[X]$ extracts attribute set X.
$t[X]$ is itself a tuple.

$t[A]$ extracts a single attribute A.
$t[A]$ is a value. *(Yes, I know this is confusing.)*

$(t_r : t_s)$ combines tuples t_r and t_s.
$(t_r : t_s)$ is itself a tuple.

The Null Value (Extended RA)

The null value is a special value that may be part of an attribute domain.

We will use the symbol $\perp$ to represent null in this cheatsheet.

While the null value is an important part of the relational model, it is not a core part of relational algebra.

Selection

Produces the subset of tuples that satisfy a condition.

For $r(R)$ and selection condition c:

$$\sigma_c\,(r) = \{t \in r \mid c(t)\}$$

Example

r

A	B	C
a_1	2	x
a_2	8	x
a_3	7	x
a_4	9	y

$\sigma_{(B>5)\wedge(C=x)}\,(r)$

A	B	C
a_2	8	x
a_3	7	x

Projection

Produces a relation with a subset of attributes.

For $r(R)$ and attribute set X:

$$\pi_X\,(r) = \{t[X] \mid t \in r\}$$

Example

r

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2
a_3	b_3	c_3	d_3

$\pi_{B,D}\,(r)$

B	D
b_1	d_1
b_2	d_2
b_3	d_3

Rename

Casts a relation to a different schema.

For $r(R)$ and a compatible schema S:

$$\rho_S\,(r)$$

Example

r

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

$\rho_{S(D,E,F)}\,(r)$

D	E	F
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

Union, Intersection, and Difference

Set theoretic operations between *union-compatible* relations (i.e. same schema).

For $r_1(R)$ and $r_2(R)$:

$$r_1 \cup r_2 = \{t \mid (t \in r_1) \vee (t \in r_2)\}$$

$$r_1 \cap r_2 = \{t \mid (t \in r_1) \wedge (t \in r_2)\}$$

$$r_1 - r_2 = \{t \mid (t \in r_1) \wedge (t \notin r_2)\}$$

Example

r_1

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3
a_4	b_4	c_4
a_5	b_5	c_5

r_2

A	B	C
a_4	b_4	c_4
a_5	b_5	c_5
a_6	b_6	c_6
a_7	b_7	c_7

$r_1 \cup r_2$

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3
a_4	b_4	c_4
a_5	b_5	c_5
a_6	b_6	c_6
a_7	b_7	c_7

$r_1 - r_2$

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

$r_1 \cap r_2$

A	B	C
a_4	b_4	c_4
a_5	b_5	c_5

$r_2 - r_1$

A	B	C
a_6	b_6	c_6
a_7	b_7	c_7

Cartesian Product

Produces every possible combination of tuples from two relations.

For $r(R)$ and $s(S)$:

$$r \times s = \{(t_r : t_s) \mid (t_r \in r) \wedge (t_s \in s)\}$$

Example

s

M	N
m_1	n_1
m_2	n_2
m_3	n_3

r

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$r \times s$

A	B	C	M	N
a_1	b_1	c_1	m_1	n_1
a_1	b_1	c_1	m_2	n_2
a_1	b_1	c_1	m_3	n_3
a_2	b_2	c_2	m_1	n_1
a_2	b_2	c_2	m_2	n_2
a_2	b_2	c_2	m_3	n_3
a_3	b_3	c_3	m_1	n_1
a_3	b_3	c_3	m_2	n_2
a_3	b_3	c_3	m_3	n_3

Theta Join (or Inner Join) and Equijoin

Theta join combines the tuples of two relations using a matching criterion.

For $r(R)$ and $s(S)$, and some arbitrary matching criterion c:

$$r \bowtie_c s = \{(t_r : t_s) \mid (t_r \in r) \wedge (t_s \in s) \wedge c(t_r : t_s)\}$$

Theta join can also be defined as:

$$r \bowtie_c s = \sigma_{c(r,s)} (r \times s)$$

Duplicate attributes are not removed.

Equijoin is a special case of theta join that only tests for equality between attributes.

Example

r

A	B	C
a_1	b_1	0
a_2	b_2	3
a_3	b_3	2

s

M	N
m_1	3
m_2	4
m_3	1
m_4	2

$r \bowtie_{N \leq C} s$

A	B	C	M	N
a_2	b_2	3	m_1	3
a_2	b_2	3	m_3	1
a_2	b_2	3	m_4	2
a_3	b_3	2	m_3	1
a_3	b_3	2	m_4	2

Natural Join

A special case of *equijoin* that matches tuples on all their common attributes.

For $r(R)$ and $s(S)$:

$$r \bowtie s = \{(t_r : t_s[S - R]) \mid (t_r \in r) \wedge (t_s \in s) \wedge m(t_r, t_s)\}$$

where m is "all common attributes must match":

$$m(t_r, t_s) = \Big(t_r[R \cap S] = t_s[R \cap S]\Big)$$

Natural join can also be defined as:

$$r \bowtie s = \pi_{R \cup S} \left(\sigma_{m(r,s)} (r \times s)\right)$$

where m is:

$$m(r, s) = \Big(r[R \cap S] = s[R \cap S]\Big)$$

and $\pi_{R \cup S}$ assumes removal of duplicate attributes.

***Natural join* is dangerous in real applications. Use *equijoin* to explicitly match tuples instead.**

Example

r

A	B	C	D
a_1	b_1	x	x
a_2	b_2	$\boldsymbol{x}$	$\boldsymbol{y}$
a_3	b_3	$\boldsymbol{x}$	$\boldsymbol{y}$
a_4	b_4	$\boldsymbol{y}$	$\boldsymbol{x}$

s

C	D	E
y	y	e_1
$\boldsymbol{x}$	$\boldsymbol{y}$	e_2
$\boldsymbol{y}$	$\boldsymbol{x}$	e_3

$r \bowtie s$

A	B	C	D	E
a_2	b_2	$\boldsymbol{x}$	$\boldsymbol{y}$	e_2
a_3	b_3	$\boldsymbol{x}$	$\boldsymbol{y}$	e_2
a_4	b_4	$\boldsymbol{y}$	$\boldsymbol{x}$	e_3

Division (Extended RA)

For $r(R)$ and $s(S)$, with $S \subseteq R$:

$$r \div s = \{t \in \pi_{R-S}(r) \mid \Phi(t)\}$$

where:

$$\Phi(t) = \underbrace{\forall t_s \in s}_{(1)} \underbrace{\exists t_r \in r}_{(2)} \underbrace{(t_r[S] = t_s}_{(3)} \underbrace{\wedge\, t = t_r[R-S])}_{(4)}$$

In plain English: (1) For all tuples in s, (2) there is at least one relation in r (3) whose common attributes match, (4) and whose non-common attributes are the same.

Division can also be defined as:

$$r \div s = r' - \underbrace{\pi_{(R-S)}\Big([r' \times s] - r\Big)}_{\text{"disqualifier" term}}$$

where r' is "all possible result tuples":

$$r' = \pi_{(R-S)}(r)$$

Example

r

A	B	C	D
α	β	x	x
α	β	x	z
α	β	w	z
γ	δ	x	x
γ	δ	x	y
γ	δ	x	z
γ	δ	w	x
γ	δ	w	y
γ	δ	w	z
γ	δ	w	w
ϵ	ζ	x	y
ϵ	ζ	x	z
ϵ	ζ	w	z
η	θ	x	y

s

C	D
x	y
x	z
w	z

$r \div s$

A	B
γ	δ
ϵ	ζ

In r, values (γ, δ) and (ϵ, ζ) are the only values that occur with all values (x, y), (x, z), and (w, z).

Outer Join (Extended RA)

Similar to *theta join*, except in the result relation, we also include tuples that don't have matches in the join.

These unmatched tuples get padded with null values in the result relation.

Full outer join includes all tuples of both operands:

$$\begin{aligned} r ⟗_c s &= (r \bowtie_c s) \cup \Big((r - \pi_R(r \bowtie_c s)) \times \{(\perp, \dots)\}\Big) \\ &\quad \cup \Big((r - \pi_R(r \bowtie_c s)) \times \{(\perp, \dots)\}\Big) \\ &= (r ⟕_c s) \cup (r ⟖_c s) \end{aligned}$$

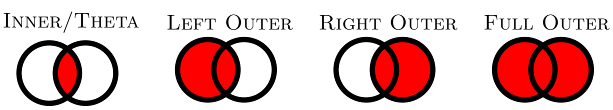
Left outer join includes all tuples of the left:

$$\begin{aligned} r ⟕_c s &= (r \bowtie_c s) \cup \Big((r - \pi_R(r \bowtie_c s)) \times \{(\perp, \dots)\}\Big) \\ &= s ⟖_c r \end{aligned}$$

Right outer join includes all tuples of the right:

$$\begin{aligned} r ⟖_c s &= (r \bowtie_c s) \cup \Big((s - \pi_S(r \bowtie_c s)) \times \{(\perp, \dots)\}\Big) \\ &= s ⟕_c r \end{aligned}$$

To summarize the differences between the joins:



Example

r

A	B	C	D
a_1	b_1	x	x
a_2	b_2	x	y
a_3	b_3	y	x
a_4	b_4	z	w

s

C	D	E
y	y	e_1
x	y	e_2
y	x	e_3

$r ⟗ s$

A	B	C	D	E
a_1	b_1	x	x	$\perp$
a_2	b_2	x	y	e_2
a_3	b_3	y	x	e_3
a_4	b_4	z	w	$\perp$
$\perp$	$\perp$	y	y	e_1

$r ⟕ s$

A	B	C	D	E
a_1	b_1	x	x	$\perp$
a_2	b_2	x	y	e_2
a_3	b_3	y	x	e_3
a_4	b_4	z	w	$\perp$

Grouping Operator (Extended RA)

Performs calculations over groups of tuples within a relation. Produces a relation containing the results.

For $r(R)$ and operator subscript G:

$$\gamma_G\,(R)$$

where G is a list containing:

- one or more attributes from R to be taken as *grouping attributes*, and
- one or more *aggregate functions*, written in the form $\theta(A,\dots)$, where θ is an aggregate function to be applied to attributes $A,\dots$.

Formally, aggregate functions can be considered to take a multiset (i.e. a set with duplicates) of values.

Aggregate functions defined in SQL-92:

AVG, MAX, MIN, SUM, COUNT

Most major DBMS implementations offer many more aggregate functions and allow user-defined functions.

Example

r

A	B	C	D
x	α	5	7
x	α	7	3
x	β	1	9
x	β	5	6
y	α	3	1
y	α	2	1
y	β	3	5
z	α	9	4

$\gamma_{A,\mathrm{COUNT}(A),\mathrm{MAX}(C),\mathrm{MAX}(D)}\,(r)$

A	COUNT(A)	MAX(C)	MAX(D)
x	4	7	9
y	3	3	5
z	1	9	4

$\gamma_{A,B,\mathrm{COUNT}(A),\mathrm{SUM}(C)}\,(r)$

A	B	COUNT(A)	SUM(C)
x	α	2	12
x	β	2	6
y	α	2	5
y	β	1	3
z	α	1	9

The COUNT() function is interesting in that it doesn't really matter which column you use.

Generalized Projection (Extended RA)

We can extend the projection operator π to also contain expressions for computation.

Example

r

A	B	C	D
a_1	2	6	d_1
a_2	8	9	d_2
a_3	3	2	d_3

$\pi_{A,B+C}\,(r)$

A	$B+C$
a_1	8
a_2	17
a_3	5

Functional Dependency