IN3020&4020 – Database Systems (2026)
Part 1: Query Processing
Lectures 7.2, 8: Relational Algebra for SQL

9, 11 February

Egor V. Kostylev
IFI, University of Oslo
egork@ifi.uio.no



Materials to read about Relational Algebra (same)

1. *Fundamentals of Database Systems* by R. Elmasri and S.B. Navathe (7th edition): Chapter 8 (maybe, not the best)
2. *Foundation of Databases* by S. Abiteboul, R. Hull & V. Vianu: Part B (more details)
   available at `http://webdam.inria.fr/Alice/`
3. *Database Systems: the Complete Book* by H. Garcia-Molina, J. Ullman & J. Widom (2nd edition): Section 2.4
4. IN2090 – Databaser og datamodellering (Leif Harald Karlsen, Dimitru Roman): Lectures 3 (basics)

   `https://www.uio.no/studier/emner/matnat/ifi/IN2090/h21/timeplan/`
5. (English) Wikipedia has a good introduction
6. etc.

Materials to Play With (same)

- Many good places with interactive exercises:
  - w3resource (`https://www.w3resource.com/`)
  - w3schools (`https://www.w3schools.com/sql/`)
  - Web pages of systems (MySQL, PostgreSQL, Oracle, etc.)

- They use SQL rather than RA
- Should not be a problem: for simple examples
  - syntactic translation is direct
  - semantic differences are immaterial

Lecture Plan

1. What is missing in the classic set RA to cover SQL?
2. More functionality for sets
3. Bag Relational Algebra intro
4. Extending and re-defining known operations
5. Bag-specific operations
6. Algebraic laws and integrity constraints

1. What is missing in the classic set RA to cover SQL?

Variants of Relational Algebra

- There are many RAs. Two RA may be
  - Equivalent: some operations are expressible via each other (e.g., $\times$ is expressible via $\bowtie$ and back in the presence of σ and ρ)
  - One strictly more expressive than another: one has operations non-expressible in the other
  - Have different domains (e.g., named/unnamed, sets/bags)
- We considered basic RA for sets under named perspective (as suggested by Codd)
- Full SQL requires a very expressive variant of RA for ordered bags (Turing complete!)
- We will look at a version of RA for (unordered) bags:
  - enough to capture most of SQL
  - but not all

Set Relational Algebra we know

- Domain: (Set) Relations
- Core operations:
  - (set) union $\cup$
  - (set) difference $\setminus$
  - selection σ_C
  - projection π_{Atts}
  - Cartesian product $\times$ (a.k.a. cross product, cross join)
  - renaming ϱ_N (technical, often not mentioned)
- Some unary, some binary, usually not always applicable
- Some parametrised: families of operations rather than single
- Expressible operations:
  - (set) intersection $\cap$
  - other joins: natural $\bowtie$, theta- $\bowtie_\theta$, semi- $\ltimes$ & $\rtimes$, etc.
  - division $\div$
  - ...

Algebra for Full SQL

What is missing in our core RA to cover all SQL functionality?

Algebra for Full SQL

What is missing in our core RA to cover all SQL functionality?

1. Value invention (`SELECT R.A+5 FROM R`)
2. NULLs (in particular, for `OUTER JOIN`s):
3. Recursion (`WITH RECURSIVE`)
4. Bag relations and aggregates (`SELECT SUM(R.A) FROM R`)
5. Ordered relations (`ORDER BY`, `LIMIT`)

... ...

Algebra for Full SQL

What is missing in our core RA to cover all SQL functionality?

1. Value invention (`SELECT R.A+5 FROM R`)
2. NULLs (in particular, for `OUTER JOIN`s):
3. Recursion (`WITH RECURSIVE`) not covered
4. Bag relations and aggregates (`SELECT SUM(R.A) FROM R`)
5. Ordered relations (`ORDER BY`, `LIMIT`) not covered

... ...

2. More functionality for sets

Generalised Projection: unary, one parameter (example)

Projection + renaming + value invention using expressions

$\pi_{site,\ topic \to area,\ ID+5 \to IDplus}$ **Tutorials**:

ID	*site*	*tutorial*	*topic*
1	w3schools	SQL_2003STD	Databases
2	w3schools	HTML_5	WebDev
3	w3schools	CSS_3	WebDev
4	w3resource	SQL_2003STD	Databases
5	w3resource	MySQL	Databases

$=$

$\pi_{site,\ topic \to area,\ ID+5 \to IDplus}$**(Tutorials)**:

IDplus	*site*	*area*
6	w3schools	Databases
7	w3schools	WebDev
8	w3schools	WebDev
9	w3resource	Databases
10	w3resource	Databases

Generalised Projection: unary, one parameter (definition)

Selects and modifies attributes, as specified in the parameter

Generalised Projection (formal definition)

- Let R be a relation
- Let N be a list of attributes A and maps $Exp \to A'$, where Exp is an expression over attributes and values, and A, A' are attributes
- Generalised Projection $\pi_N(R)$ is the relation that
  - is defined when all A' are different and not attributes of R, all other mentioned attributes (incl., A) are in the schema of R, and all types are matched
  - with schema consisting of all A and A'
  - contains all tuples constructed from tuples in R as specified by N

- Repetitions are removed
- Just renaming when Exp is an attribute
- Exp may be a constant (not mentioning attributes), including NULL
- Can create bigger table than input

Left Outer Join: binary, no parameters (example)

Combines tuples from two relations in all possible ways that agree on common attributes, and extend left dangling tuples with NULLs

Tutorials:

ID	*site*	*tutorial*	*topicID*
1	w3schools	SQL_2003STD	1
2	w3schools	HTML_5	2
3	w3schools	CSS_3	3
4	w3resource	SQL_2003STD	1

⟕

Topics:

topicID	*topic*
1	Databases
2	WebDev
25	Asronomy

=

Tutorials ⟕ **Topics**:

ID	*site*	*tutorial*	*topicID*	*topic*
1	w3schools	SQL_2003STD	1	Databases
2	w3schools	HTML_5	2	WebDev
3	w3schools	CSS_3	3	NULL
4	w3resource	SQL_2003STD	1	Databases

Outer Joins: binary, no parameters (definition)

Natural Join + dangling tuples extended with NULLs

Outer Joins (native formal definition)

- Let R and S be two relations
- Left Outer Join $R \ltimes\!\!\!\bowtie S$ is the relation that
  - all common attributes of R and S have the same domains
  - has the union of schemas of R and S as schema
  - contains all possible combinations of a tuple in R with a tuple in S that agree on common attributes and all dangling tuples in R extended with NULLs
  - a dangling tuple in R is a tuple that does not agree with any tuple in R on common attributes
- Right Outer Join $R \bowtie\!\!\!\rtimes S := S \ltimes\!\!\!\bowtie R$
- Full Outer Join $R \ltimes\!\!\!\bowtie\!\!\!\rtimes S := (R \ltimes\!\!\!\bowtie S) \cup (R \bowtie\!\!\!\rtimes S)$

Outer Joins: binary, no parameters (definition)

Natural Join + dangling tuples extended with NULLs

Outer Joins (native formal definition)

- Let R and S be two relations
- Left Outer Join $R \ltimes S$ is the relation that
  - all common attributes of R and S have the same domains
  - has the union of schemas of R and S as schema
  - contains all possible combinations of a tuple in R with a tuple in S that agree on common attributes and all dangling tuples in R extended with NULLs
  - a dangling tuple in R is a tuple that does not agree with any tuple in R on common attributes
- Right Outer Join $R \rtimes S := S \ltimes R$
- Full Outer Join $R \ltimes\!\rtimes S := (R \ltimes S) \cup (R \rtimes S)$

Expressible via the core RA + Generalised Projection

(see the first mandatory)

Algebraic Laws

Generalised Projection and outer joins have their laws. For example:

Generalised Projection can be pushed through union and Cartesian product:

$$\pi_N(R \cup S) = \pi_N(R) \cup \pi_N(S)$$

$$\pi_{N_R, N_S}(R \times S) = \pi_{N_R}(R) \times \pi_{N_S}(S)$$

if N_R mentions only attributes of R
and N_S mentions only attributes of S

Left outer join is idempotent: $R \rtimes R = R \quad (= R \ltimes R)$

But not commutative: $R \rtimes S \neq S \rtimes R$ (unless $R = S$)

And not associative: $(R \rtimes S) \rtimes T \neq R \rtimes (S \rtimes T)$

There are many more

3. Bag Relational Algebra intro

Bag Relational Algebra domain

- Domain: Bag relations (i.e., bag tables)
- Bag relation components (reminder):
  - Relation name: just a string (abstract away)
  - Schema: set of attribute names with domains (i.e., datatypes)
  - Bag (possible repetitions!) of tuples:
    assignments of values to attribute names conforming schema

- Example:

Tutorials:

ID	*site*	*tutorial*	*topic*
1	w3schools	SQL_2003STD	Databases
2	w3schools	HTML_5	WebDev
1	w3schools	SQL_2003STD	Databases
4	w3resource	SQL_2003STD	Databases
5	w3resource	MySQL	Databases

Bag Relational Algebra operations

- Domain: Bag Relations
- Every Set Relation is a Bag Relation
- All set RA operations generalise to bags:
  - some are just extensions to bags that are not sets—that is, if input relations are sets then the result is as before
  - others need re-definition even for sets

Bag Relational Algebra operations

- Domain: Bag Relations
- Every Set Relation is a Bag Relation
- All set RA operations generalise to bags:
  - some are just extensions to bags that are not sets—that is, if input relations are sets then the result is as before:
    difference $\setminus$, intersection $\cap$, selection σ_C, Cart. product $\times$, division $\div$, renaming ϱ_N, joins ($\bowtie$, $\bowtie_\theta$, $\ltimes$, $\rtimes$, $\lhd$, $\rhd$, ⟗, ⟕, ⟖)
  - others need re-definition even for sets (marked by sup-script b):
    union $\cup^b$, (generalised) projection π_N^b
      (we also keep the set versions $\cup$ and π_N,
        which are undefined on bags that are not sets)

Bag Relational Algebra operations

- Domain: Bag Relations
- Every Set Relation is a Bag Relation
- All set RA operations generalise to bags:
  - some are just extensions to bags that are not sets—that is, if input relations are sets then the result is as before:
    difference $\setminus$, intersection $\cap$, selection σ_C, Cart. product $\times$, division $\div$, renaming ϱ_N, joins ($\bowtie, \bowtie_\theta, \ltimes, \rtimes, \triangleleft, \triangleright, ⟕, ⟖, ⟗$)
  - others need re-definition even for sets (marked by sup-script b):
    union $\cup^b$, (generalised) projection π^b_N
    (we also keep the set versions $\cup$ and π_N, which are undefined on bags that are not sets)
- Several new operations:
  - duplicate elimination δ, grouping+aggregation γ_L, ...

4. Extending and re-defining the known operations

Bag Union: binary, no parameters (example only)

Bag-theoretic union of tuples (multiplicities are added):

Tutorials1:

ID	*site*	*tutorial*	*topic*
1	w3schools	SQL_2003STD	Databases
1	w3schools	SQL_2003STD	Databases
3	w3schools	CSS_3	WebDev

$\cup^b$

Tutorials2:

ID	*site*	*tutorial*	*topic*
3	w3schools	CSS_3	WebDev
3	w3schools	CSS_3	WebDev
5	w3resource	MySQL	Databases

$=$

Tutorials1 $\cup^b$ **Tutorials2**:

ID	*site*	*tutorial*	*topic*
1	w3schools	SQL_2003STD	Databases
1	w3schools	SQL_2003STD	Databases
3	w3schools	CSS_3	WebDev
3	w3schools	CSS_3	WebDev
3	w3schools	CSS_3	WebDev
5	w3resource	MySQL	Databases

Difference for bags: binary, no parameters (example only)

Bag-theoretic difference of tuples in same-structured relations (multiplicities are subtracted; if negative, overwritten by 0)

Tutorials1:

ID	*site*	*tutorial*	*topic*
1	w3schools	SQL_2003STD	Databases
1	w3schools	SQL_2003STD	Databases
2	w3schools	HTML_5	WebDev

\

Tutorials2:

ID	*site*	*tutorial*	*topic*
1	w3schools	SQL_2003STD	Databases
2	w3schools	HTML_5	WebDev
2	w3schools	HTML_5	WebDev
3	w3schools	CSS_3	WebDev

=

Tutorials1 \ Tutorials2:

ID	*site*	*tutorial*	*topic*
1	w3schools	SQL_2003STD	Databases

Selection for bags: unary, one parameter (example only)

Selects tuples as in the set case:

$\sigma_{ID>2 \ \& \ topic=\text{'Databases'}}$ (**Tutorials**:)

ID	*site*	*tutorial*	*topic*
1	w3schools	SQL_2003STD	Databases
2	w3schools	HTML_5	WebDev
3	w3schools	CSS_3	WebDev
4	w3resource	SQL_2003STD	Databases
4	w3resource	SQL_2003STD	Databases
5	w3resource	MySQL	Databases

$=$

$\sigma_{ID>2 \ \& \ topic=\text{'Databases'}}$**(Tutorials)**:

ID	*site*	*tutorial*	*topic*
4	w3resource	SQL_2003STD	Databases
4	w3resource	SQL_2003STD	Databases
5	w3resource	MySQL	Databases

Generalised Projection for bags (including renaming): unary, one parameter (example only)

Also as in set case (outputs the same number of tuples as in input):

$\pi^{b}_{site,\ topic \to area,\ ID+5 \to IDplus}$ (**Tutorials**:)

ID	*site*	*tutorial*	*topic*
1	w3schools	SQL_2003STD	Databases
2	w3schools	HTML_5	WebDev
3	w3schools	CSS_3	WebDev
4	w3resource	SQL_2003STD	Databases
4	w3resource	SQL_2003STD	Databases

$=$

$\pi^{b}_{site,\ topic \to area,\ ID+5 \to IDplus}$**(Tutorials)**:

IDplus	*site*	*area*
6	w3schools	Databases
7	w3schools	WebDev
8	w3schools	WebDev
9	w3resource	Databases
9	w3resource	Databases

Natural Join for bags (including Cartesian product): binary, no parameters (example only)

Combines tuples from two relations in all possible ways that agree on common attributes (multiplicities are multiplied):

Tutorials:

ID	*site*	*tutorial*	*topicID*
1	w3schools	SQL_2003STD	1
1	w3schools	SQL_2003STD	1
2	w3schools	HTML_5	2
2	w3schools	HTML_5	2

$\bowtie$

Topics:

topicID	*topic*
1	Databases
1	Databases
2	WebDev
25	Astronomy

$=$

Tutorials $\bowtie$ **Topics**:

ID	*site*	*tutorial*	*topicID*	*topic*
1	w3schools	SQL_2003STD	1	Databases
1	w3schools	SQL_2003STD	1	Databases
1	w3schools	SQL_2003STD	1	Databases
1	w3schools	SQL_2003STD	1	Databases
2	w3schools	HTML_5	2	WebDev
2	w3schools	HTML_5	2	WebDev

Expressible operations for bags

Many operations are expressed exactly as in the set RA:

Theta Join:	$R \bowtie_\theta S$	$:=$	$\sigma_\theta(R \bowtie S)$
Intersection:	$R \cap S$	$:=$	$R \setminus (R \setminus S)$

Others require more elaborate expressions to take care of multiplicity (not used so much for bags, so do not give them here)

Left Semi-Join:	$R \ltimes S$	$:=$	$\ldots$
Right Semi-Join:	$R \rtimes S$	$:=$	$\ldots$
Left Anti-Join:	$R \triangleright S$	$:=$	$\ldots$
Right Anti-Join:	$R \triangleleft S$	$:=$	$\ldots$
Left Outer Join:	$R \mathbin{⟕} S$	$:=$	$\ldots$
Right Outer Join:	$R \mathbin{⟖} S$	$:=$	$\ldots$
Full Outer Join:	$R \mathbin{⟗} S$	$:=$	$\ldots$
Division:	$R \div S$	$:=$	$\ldots$

5. Bag-specific operations

Duplicate elimination: unary, no parameters (example)

Removes duplicates:

δ (**Tutorials**:

ID	*site*	*tutorial*	*topic*
1	w3schools	SQL_2003STD	Databases
2	w3schools	HTML_5	WebDev
3	w3schools	CSS_3	WebDev
4	w3resource	SQL_2003STD	Databases
4	w3resource	SQL_2003STD	Databases
5	w3resource	MySQL	Databases

)

=

δ(**Tutorials**):

ID	*site*	*tutorial*	*topic*
1	w3schools	SQL_2003STD	Databases
2	w3schools	HTML_5	WebDev
3	w3schools	CSS_3	WebDev
4	w3resource	SQL_2003STD	Databases
5	w3resource	MySQL	Databases

Duplicate elimination: unary, no parameters (definition)

Removes duplicates

Duplicate Elimination (formal definition)

- Let R be a relation
- Duplicate elimination $\delta(R)$ is the relation that
  - is always defined
  - has the same schema as R
  - contains all tuples in R but without duplicates (i.e., with all multiplicities 1)

Set operations can be simulated by the bag versions and δ

For example, $R \cup S = \delta(R \cup^b S)$ for every set relations R and S

More laws below

Grouping+aggregation: unary, one parameter (example)

Groups and aggregates according to the parameter

$$\gamma_{\textit{site},\ \mathrm{SUM}(\textit{ID})\to\textit{sID},\ \mathrm{COUNT}(\textit{topic})\to\textit{ctopic}}$$

Tutorials:

ID	*site*	*tutorial*	*topic*
1	w3schools	SQL_2003STD	Databases
2	w3schools	HTML_5	WebDev
3	w3schools	CSS_3	WebDev
4	w3resource	SQL_2003STD	Databases
5	w3resource	MySQL	*NULL*

$=$

$\gamma_{\textit{site},\ \mathrm{SUM}(\textit{ID})\to\textit{sID},\ \mathrm{COUNT}(\textit{topic})\to\textit{ctopic}}$**(Tutorials)**:

site	*sID*	*ctopic*
w3schools	6	3
w3resource	9	1

Question: Why 3 and 1 in the last column?

Grouping+aggregation: unary, one parameter (example)

Groups and aggregates according to the parameter

$$\gamma_{\textit{site},\ \mathrm{SUM}(\textit{ID})\to\textit{sID},\ \mathrm{COUNT}(\textit{topic})\to\textit{ctopic}}$$

Tutorials:

ID	*site*	*tutorial*	*topic*
1	w3schools	SQL_2003STD	Databases
2	w3schools	HTML_5	WebDev
3	w3schools	CSS_3	WebDev
4	w3resource	SQL_2003STD	Databases
5	w3resource	MySQL	*NULL*

$=$

$\gamma_{\textit{site},\ \mathrm{SUM}(\textit{ID})\to\textit{sID},\ \mathrm{COUNT}(\textit{topic})\to\textit{ctopic}}$**(Tutorials)**:

site	*sID*	*ctopic*
w3schools	6	3
w3resource	9	1

Question: Why 3 and 1 in the last column?

Answer: Duplicates count, *NULL* does not

Aggregate functions

An aggregate function takes a bag of values and returns a value

May be also undefined (e.g., due to type mismatch)

There are several aggregate functions relevant to SQL

We consider some examples:

- COUNT(Ω) counts the number of elements in Ω (with multiplicities) different from *NULL*:
  $$\text{COUNT}(\{\{a, b, b, b, \mathit{NULL}\}\}) = 4$$
- COUNT_DISTINCT(Ω) counts the number of distinct elements in Ω (without multiplicities) different from *NULL*:
  $$\text{COUNT_DISTINCT}(\{\{a, b, b, b, \mathit{NULL}\}\}) = 2$$

Aggregate functions

An aggregate function takes a bag of values and returns a value

May be also undefined (e.g., due to type mismatch)

There are several aggregate functions relevant to SQL

We consider some examples:

- MIN(Ω) computes the minimum of non-*NULL* elements in Ω:
  $\text{MIN}(\{\{1, 2, 3, 3, \mathit{NULL}\}\}) = 1$
- MAX(Ω) computes the maximum of non-*NULL* elements in Ω:
  $\text{MAX}(\{\{1, 2, 3, 3, \mathit{NULL}\}\}) = 3$
- SUM(Ω) computes the sum of non-*NULL* elements in Ω:
  $\text{SUM}(\{\{1, 2, 3, 3, \mathit{NULL}\}\}) = 9$
- AVG(Ω) computes the average of non-*NULL* elements in Ω:
  $\text{AVG}(\{\{1, 2, 3, 3, \mathit{NULL}\}\}) = 2.25$

All undefined if Ω has anything different from numbers and *NULL*, evaluate to *NULL* if Ω has only *NULL*s (or empty)

Grouping+aggregation: unary, one parameter (definition)

Groups and aggregates the values of attributes, as specified in the parameter

Grouping and Aggregation (formal definition)

- Let R be a relation and N a list of attributes A_g and maps $Agg(A) \rightarrow A'$, where Agg is an aggregate function, and A_g, A, A' are attributes
- Grouping with Aggregation $\gamma_N(R)$ is the relation that
  - is defined when all A' are different and not attributes of R, all other mentioned attributes (incl., A_g and A) are in signature of R, and all types (incl., in aggregates) are matched
  - with signature consisting of all A_g and A'
  - contains all tuples constructed from R as follows:
    1. tuples in R are grouped by distinct tuples of values of A_g
    2. for each group, aggregate $Agg(\Omega)$ is computed for every $Agg(A) \rightarrow A'$, where Ω is the bag of values of A in the group
    3. one tuple is constructed for each group from all A_g and all A' as computed values of the aggregates $Agg(A) \rightarrow A'$

Becomes projection + duplicate elimination when no aggregates mentioned

6. Algebraic Laws and Integrity Constraints

Algebraic Laws for bags

Almost all the laws mentioned before for sets extend to bags

For example, natural join is commutative and associative:

$R \bowtie S = S \bowtie R \quad (R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

Selection and projection can be pushed through join (not always!):

$\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S$ if C mentions only attributes of R

However, there are exceptions (e.g., $R \bowtie R \neq R$ for bags)

Algebraic Laws for bags

Almost all the laws mentioned before for sets extend to bags

For example, natural join is commutative and associative:

$R \bowtie S = S \bowtie R \quad (R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

Selection and projection can be pushed through join (not always!):

$\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S$ if C mentions only attributes of R

However, there are exceptions (e.g., $R \bowtie R \neq R$ for bags)

Bag union $\cup^b$ and generalised projection π^b_N have their own laws, often different from their set versions $\cup$ and π_N (e.g., $R \cup^b R \neq R$)

Algebraic Laws for bags

Almost all the laws mentioned before for sets extend to bags

For example, natural join is commutative and associative:

$R \bowtie S = S \bowtie R \quad (R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

Selection and projection can be pushed through join (not always!):

$\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S$ if C mentions only attributes of R

However, there are exceptions (e.g., $R \bowtie R \neq R$ for bags)

Bag union $\cup^b$ and generalised projection π^b_N have their own laws, often different from their set versions $\cup$ and π_N (e.g., $R \cup^b R \neq R$)

Dupl. elimination δ, and grouping+aggregation γ_N also have laws:

- δ can be pushed through join, selection, etc.:

  $\delta(R \bowtie S) = \delta(R) \bowtie \delta(S) \qquad \delta(\sigma_C(R)) = \sigma_C(\delta(R))$

- γ_N has very few useful laws

  (i.e., allow for only very limited optimisation)

Integrity Constraints in RA

We can express integrity constraints (primary and foreign keys, etc.) using RA

Integrity Constraints in RA

We can express integrity constraints (primary and foreign keys, etc.) using RA

Primary key: attributes $A_1, \ldots, A_k$ of relation R

Relation R satisfies this primary key if for every tuple t_1 in R there is no other tuple t_2 in R such that $t_1[A_1, \ldots, A_k] = t_2[A_1, \ldots, A_k]$

Integrity Constraints in RA

We can express integrity constraints (primary and foreign keys, etc.) using RA

Primary key: attributes $A_1, \ldots, A_k$ of relation R

Relation R satisfies this primary key if for every tuple t_1 in R there is no other tuple t_2 in R such that $t_1[A_1, \ldots, A_k] = t_2[A_1, \ldots, A_k]$

Assuming that R is a set relation, can be written as

$\sigma_C(R \bowtie \rho_N(R)) = \emptyset$, where

- N is $B_1 \to B_1', \ldots, B_m \to B_m'$ for $B_1, \ldots, B_m$ are all the attributes of R different from $A_1, \ldots, A_k$
- C is $B_1 \neq B_1' \vee \cdots \vee B_m \neq B_m'$

For arbitrary bag relations also possible, but requires using aggregates (Homework: try it!)

Integrity Constraints in RA

We can express integrity constraints (primary and foreign keys, etc.) using RA

Foreign key: attributes $B_1, \ldots, B_k$ of relation S and attributes $A_1, \ldots, A_k$ of relation R with matching domains

Relations S and R satisfy this foreign key if every tuple t_1 in S has a tuple t_2 in R such that $t_1[B_1, \ldots, B_k] = t_2[A_1, \ldots, A_k]$

Integrity Constraints in RA

We can express integrity constraints (primary and foreign keys, etc.) using RA

Foreign key: attributes $B_1, \ldots, B_k$ of relation S and attributes $A_1, \ldots, A_k$ of relation R with matching domains

Relations S and R satisfy this foreign key if every tuple t_1 in S has a tuple t_2 in R such that $t_1[B_1, \ldots, B_k] = t_2[A_1, \ldots, A_k]$

Can be written as $\delta(\pi_{B_1,\ldots,B_k}(S)) \subseteq \delta(\pi_{A_1,\ldots,A_k}(R))$

What have we learned?

What is missing in the classic set RA to cover (most of) SQL