Relational Algebra and Calculus

Relational Algebra and Calculus

- Relational Algebra and Relational Calculus are the formal query languages for a relational model.
- Query languages are specialized languages for asking questions (or queries) that involve the data in a database.
- Both form the base for the SQL language which is used in most of the relational DBMSs.

Relational Algebra vs Relational Calculus

Relational Algebra

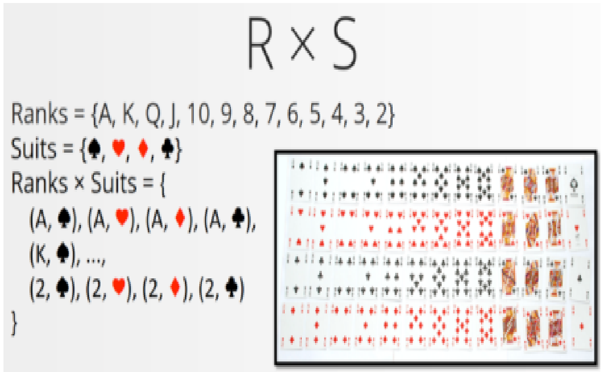
- ***Procedural language*** that describes the procedure to obtain the result.
- It describes the ***order of operations in the query*** that specifies how to retrieve the result.

Relational Algebra vs Relational Calculus

Relational Calculus

- ***Declarative language*** that defines *what* result is to be obtained.
- It ***does not specify the sequence of operations*** in which query will be evaluated.

Relation Algebra Introduction

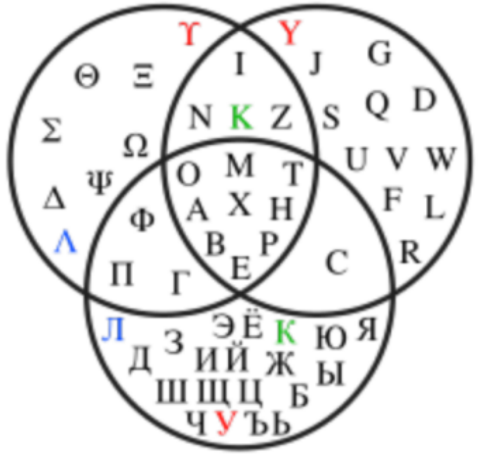


Relation data model is based on Relation algebra, and Relation algebra based on Set theory.

A **relation** is a subset of the Cartesian product of a list of **domains** characterized by a name. The steps below outline the logic between a relation and its domains.

- Domain, D - the set of values that a data item di can take.
- The Cartesian product of domains is the set of all possible combinations of domain values:
- Given n domains are denoted by D1, D2, … Dn; then (D1×D2×... ×Dn = {(d11 , ..., d1i , ..., dki, ..., dni)}, where dki ∊ Dk)
- And r is a relation defined on these domains. Then r ⊆ D1×D2×…×Dn
- Example 1. D1=(1,2), D2=(a,b,c). D1×D2={(1,a),(1,b),(1,c),(2,a),(2,b),(2,c)}; r1={(1,a),(1,c),(2,b)}; r2={(1,a),(1,b),(1,c),(2,a),(2,b),(2,c)}.
- Example 2. Let A=B=R, where R is the set of all real numbers. Then R×R is the set of all Cartesian coordinates of the points of the plane.

Set Theory - Definitions



Elements of a set (or points) are the objects of which the set consists. Set denoted with capital letters of the Latin alphabet, its elements are lowercase. Each element of the set is unique.

Venn diagrams are a schematic representation of all possible intersections of several sets.

Special sets

- **An empty set ø** is a set that does not contain a single element.
- **The universe U** is a multitude containing all conceivable objects.

The cardinality of a set |A| is a characteristic of a set that generalizes the concept of the number of elements. The power of an empty set is zero; for finite sets, the power coincides with the number of elements

Relationships between sets

- **Inclusion**: $A \subseteq B \Leftrightarrow \forall a \in A: a \in B$
- **Not intersection** when A and B do not have common elements: $\Leftrightarrow \forall a \in A: a \notin B$

Set Theory - Operations

Unary operations on sets:

complement: $\overline{A}:=U\setminus A=\{x|x\notin A\}$ - is the difference of the set A with the universe U.

Binary operations on sets:

intersection: $A\cap B:=\{x|x\in A\wedge x\in B\}$. If the sets A and B are not intersect, then $A\cap B = \emptyset$.

union: $A\cup B:=\{x|x\in A\vee x\in B\}$.

difference: $A\setminus B:=A\cap\overline{B}=\{x|x\in A\wedge x\notin B\}$.

symmetric difference: $A\triangle B:=(A\cup B)\setminus(A\cap B)=A\cap\overline{B}\cup\overline{A}\cap B=\{x|\ (x\in A\wedge x\notin B)\vee(x\notin A\wedge x\in B)\}$

Cartesian or direct product: $A\times B=\{(a, b)|a\in A, b\in B\}$.

Priority operations.

First, unary operations (addition) are performed, then intersections, then unions and differences, which have the same priority. The sequence of operations can be changed in brackets.

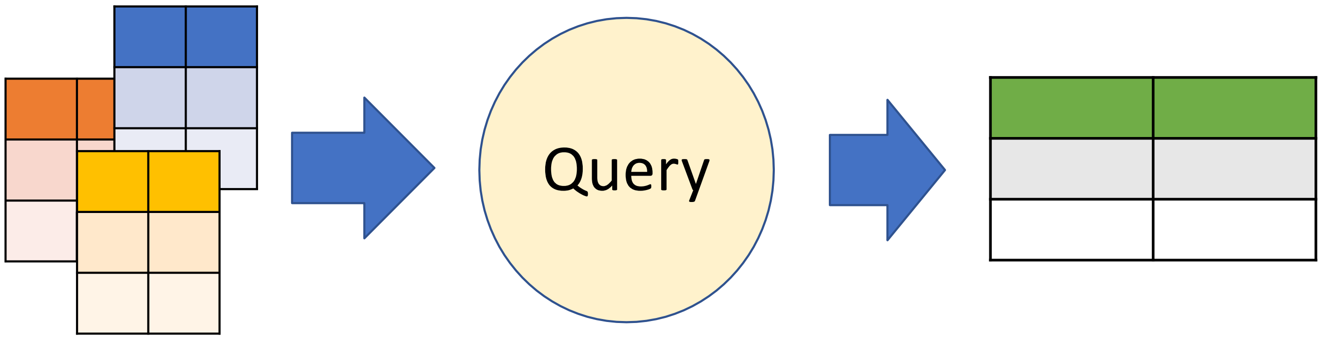
Set Operations Properties

1. Commutativity: $A\cup B=B\cup A$; $A\cap B=B\cap A$
2. Associativity: $(A\cup B)\cup C=A\cup(B\cup C)$; $(A\cap B)\cap C=A\cap(B\cap C)$
3. Distributivity: $(A\cup B)\cap C=(A\cap C)\cup(B\cap C)$; $(A\cap B)\cup C=(A\cup C)\cap(B\cup C)$
4. Selfie: $A\cup A=A$; $A\cap A=A$
5. Nullity: $A\cup\emptyset=\emptyset$; $A\cap\emptyset=\emptyset$
6. Duality (de Morgan laws): $\overline{A\cup B}=\overline{A}\cap\overline{B}$; $\overline{A\cap B}=\overline{A}\cup\overline{B}$

Relational Queries

- Before we start, we need to clarify important points about the relational queries:

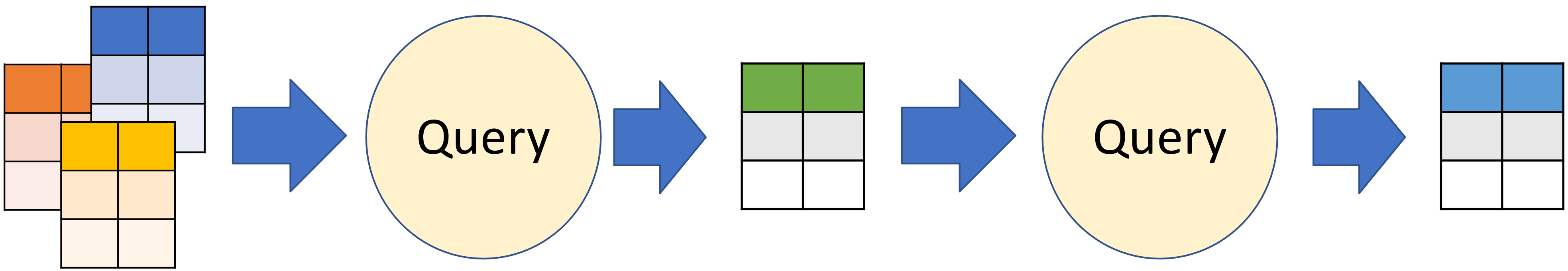
The inputs and output of a query are relations



Relational Queries

- Before we start, we need to clarify important points about the relational queries:

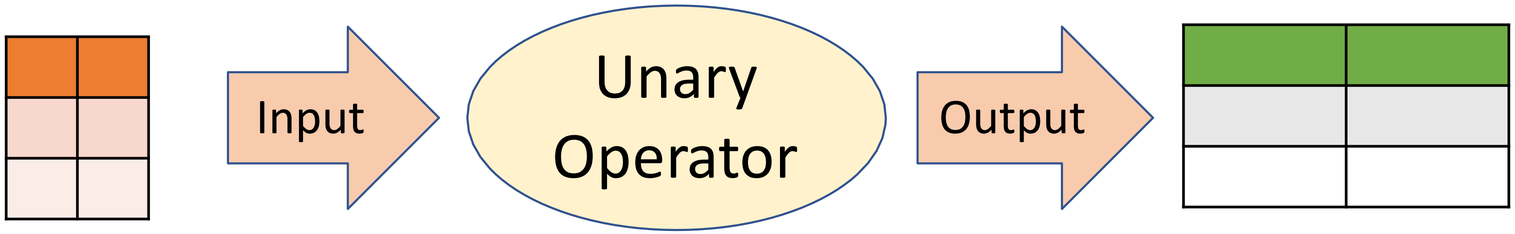
Queries involve the computation of intermediate results which are themselves ***relation instances***



Relational Algebra

- The operators in any expression are either unary or binary operators.

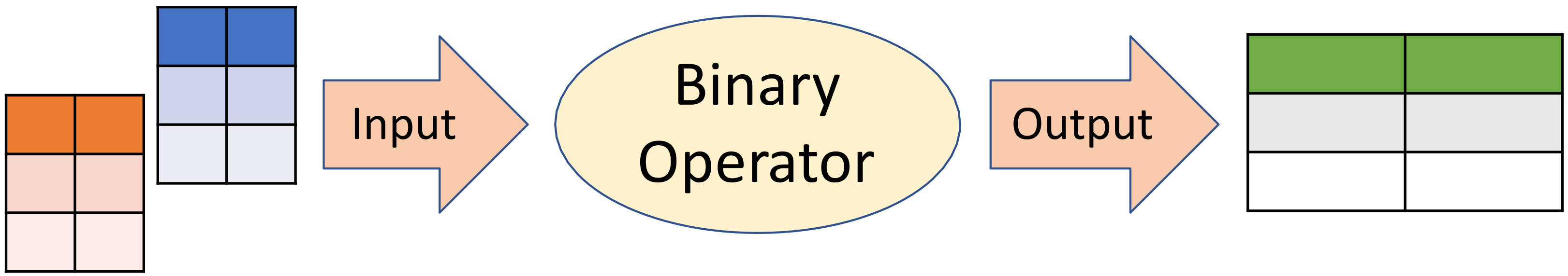
- The ***unary operator*** accepts one relation as an input and produces a new relation as a result.



Relational Algebra

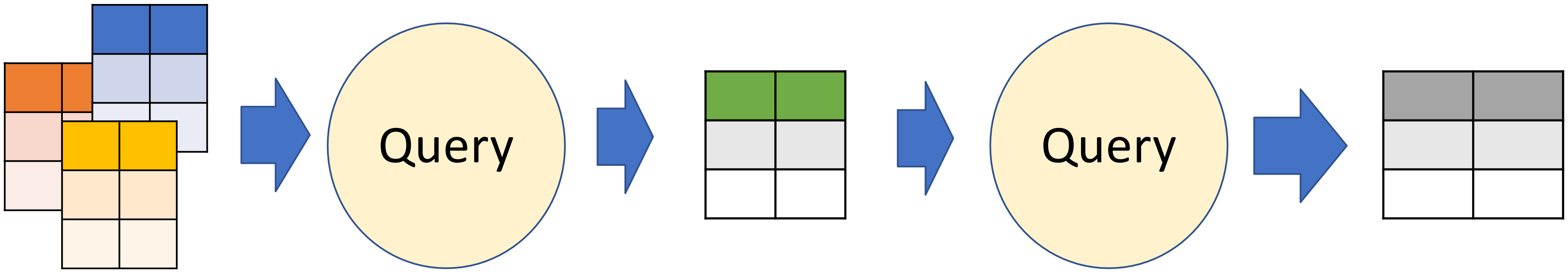
- The operators in any expression are either unary or binary operators.

- The ***binary operator*** accepts two relations as input and produces a new relation as a result.

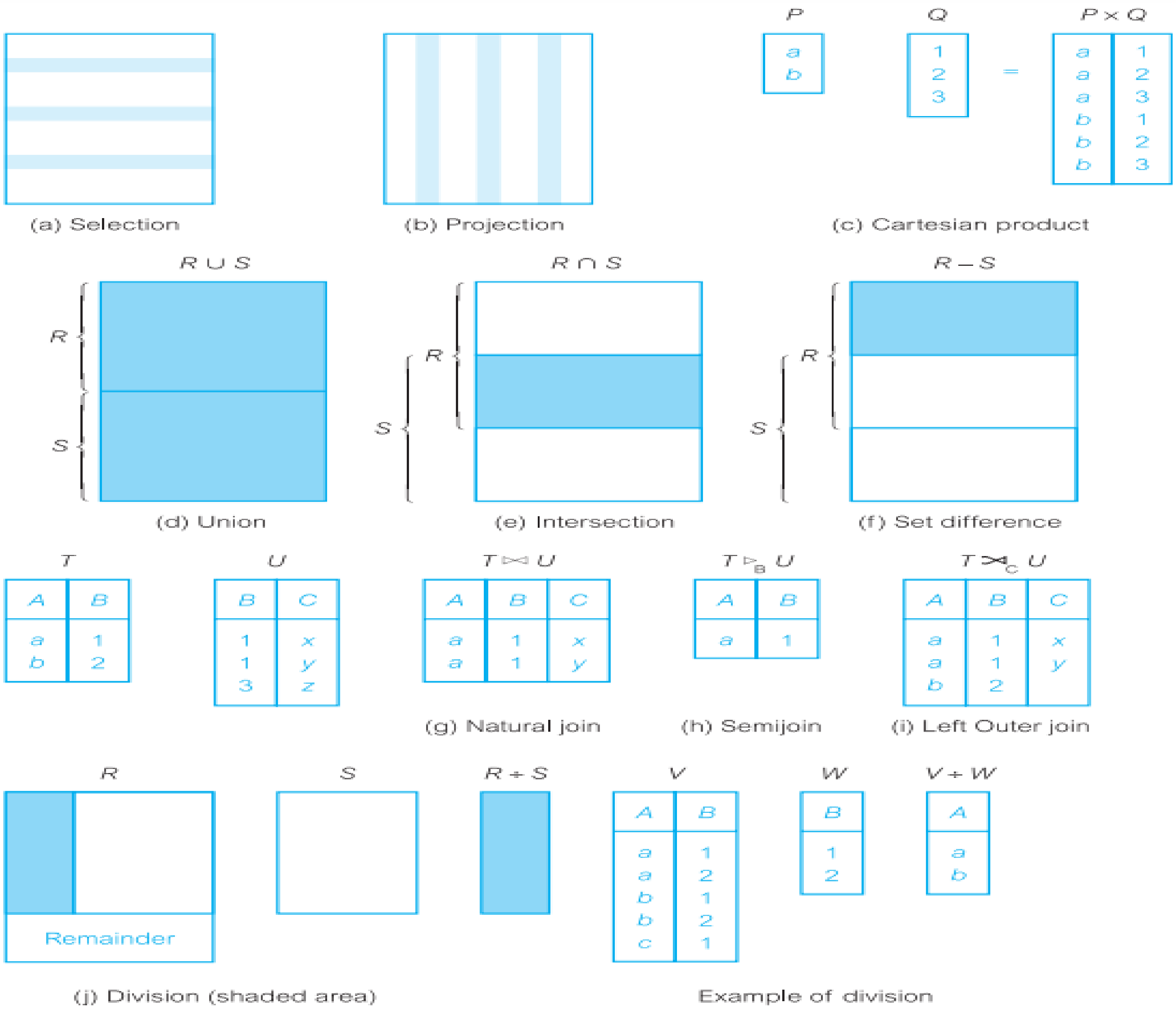


Relational Algebra

- The result relation obtained from the expression can be further composed to other expression whose result will again be ***a new relation***.



- This property allows the ***composition of operators to form complex queries***



(a) Selection
(b) Projection
P
a
b
Q
1
2
3
P × Q
a a a b b b
1 2 3 1 2 3
(c) Cartesian product
R ∪ S
R
S
(d) Union
R ∩ S
R
S
(e) Intersection
R − S
R
S
(f) Set difference
T
A B
a 1
b 2
U
B C
1 x
1 y
3 z
T ⋈ U
A B C
a 1 x
a 1 y
(g) Natural join
T ▷B U
A B
a 1
(h) Semijoin
T ⟕C U
A B C
a 1 x
a 1 y
b 2
(i) Left Outer join
R
Remainder
S
R ÷ S
(j) Division (shaded area)
V
A B
a 1
a 2
b 1
b 2
c 1
W
B
1
2
V ÷ W
A
a
b
Example of division

Selection operator - σ

- The selection operator (unary operator) returns a ***subset of tuples*** from a relation that ***satisfies certain condition***.

$$\sigma_{\text{<selection condition>}}(\text{Relation})$$

- Think of the selection condition as the if statement in programming languages.

Selection operator - σ

$$\sigma_{\text{<selection condition>}}\ (\text{Relation})$$

- The selection condition is a ***Boolean combination of terms*** with the form of:

< Attribute > < Comparison operator > < Constant value >

< Attribute 1 > < Comparison operator > < Attribute 2>

- The comparison operators can be: $>, <, =, >=, <=, \neq$

Selection operator - σ

$$\sigma_{\text{<selection condition>}}(\text{Relation})$$

- The selection operator is applied independently to each ***individual tuple*** of the operand (Relation), and the tuple is selected if and only if the condition evaluates to ***TRUE***.

$$\sigma_{Age = 18}(\text{Student})$$

ID	Short name	Age	GPA
344	A J	20	3.8
342	B K	18	3.6
767	C E	20	3.2
345	D P	18	3.5
234	E U	19	3.7

- The schema of the result is the schema of the input relation instance (all the fields exist in the result, we are selecting rows/tuples)

$$\sigma_{\text{Age} = 18}\ (\text{Student})$$

ID	Short name	Age	GPA
344	**A J**	**20**	**3.8**
342	**B K**	**18**	**3.6**
767	**C E**	**20**	**3.2**
345	**D P**	**18**	**3.5**
234	**E U**	**19**	**3.7**

ID	Short name	Age	GPA
342	B K	18	3.6
345	D P	18	3.5

$$\sigma_{GPA <= 3.6} (Student)$$

ID	Short name	Age	GPA
344	A J	20	3.8
342	B K	18	3.6
767	C E	20	3.2
345	D P	18	3.5
234	E U	19	3.7

ID	Short name	Age	GPA
342	B K	18	3.6
767	C E	20	3.2
345	D P	18	3.5

Selection operator - σ

$$\sigma_{\text{<selection condition>}}\ (\text{Relation})$$

- We can have one or more selection condition linked through Boolean operators (e.g., AND, OR, NOT)

$$\sigma_{\text{< condition> Boolean operator < condition>}}\ (\text{R})$$

$$\sigma_{\text{GPA} <= 3.6 \text{ AND Age} = 20} \text{(Student)}$$

ID	Short name	Age	GPA
344	A J	20	3.8
342	B K	18	3.6
767	C E	20	3.2
345	D P	18	3.5
234	E U	19	3.7

ID	Short name	Age	GPA
767	C E	20	3.2

Selection operator - σ - equivalence

$$\sigma_{<C2>}(\sigma_{<C1>}(R))$$

$$=$$

$$\sigma_{<C1>}(R)$$ Step 1

$$\downarrow$$

$$\sigma_{<C2>}(\quad S \quad)$$ Step 2

Selection operator - σ - equivalence

$$\sigma_{<C2>} (\sigma_{<C1>} (R))$$

$$=$$

$$\sigma_{<C1>} (\sigma_{<C2>} (R))$$

$$=$$

$$\sigma_{<C1> \text{ AND } <C2>} (R)$$

(a) Selection

Selection [$\sigma_{<selection\_condition>}(R)$]

This is a unary operation, the result is a subset of the original relation corresponding to the **conditions** on the values of some attributes.

Relation R

A	B	C
a	b	c
c	a	d
c	b	d

Selection $\sigma_{C=d}(R)$

A	B	C
c	a	d
c	b	d

- The number of rows returned by a selection is <= of rows in the original table.
    - Minimum Cardinality = 0
    - Maximum Cardinality = |R|
- Simple condition: AoperationB or AoperationConst, where A,B - attributes.
- We may use relational operators like =, ≠ ,> ,< ,<=, >=; logical operators like ∧ , ∨ , ! with the selection condition.
- Degree of the relation from a selection is same as input relation degree.
- Selection operator is commutative in nature:
  $\sigma_{A \wedge B}(R) \equiv \sigma_{B \wedge A}(R)$ **OR** $\sigma_B(\sigma_A(R)) \equiv \sigma_A(\sigma_B(R))$.

- Selection always selects the entire tuple. It can not select a section a tuple.
- Selection operator only selects the required tuples according to the selection condition, it does not display the selected tuples. To display the selected tuples, projection operator is used.

Selection (or Restriction) The Selection operation works on a single relation R and defines a relation that contains only those tuples of R that satisfy the specified condition (predicate)

Projection operator - π

- The projection operator (unary operator) returns a ***subset of fields (attributes/columns)*** from a relation.

$$\pi_{\text{<attribute 1, attribute 2 ... attribute n>}} \text{(Relation)}$$

$$\pi_{\text{ID, Short name}}(\text{Relation})$$

ID	Short name	Age	GPA
344	A J	20	3.8
342	B K	18	3.6
767	C E	20	3.2
345	D P	18	3.5
234	E U	19	3.7

ID	Short name
344	A J
342	B K
767	C E
345	D P
234	E U

$$\pi_{\text{ID}}(\pi_{\text{ID, Short name}}(\text{Relation}))$$

ID	Short name	Age	GPA
344	A J	20	3.8
342	B K	18	3.6
767	C E	20	3.2
345	D P	18	3.5
234	E U	19	3.7

ID	Short name
344	A J
342	B K
767	C E
345	D P
234	E U

ID
344
342
767
345
234

π_{Age} (Relation)

ID	Short name	Age	GPA
344	A J	20	3.8
342	B K	18	3.6
767	C E	20	3.2
345	D P	18	3.5
234	E U	19	3.7

Age
20
18
20
18
19

- The relational model is a set-based (***no duplicate tuples allowed***)

$$\pi_{\text{Age}}(\text{Relation})$$

ID	Short name	Age	GPA
344	A J	20	3.8
342	B K	18	3.6
767	C E	20	3.2
345	D P	18	3.5
234	E U	19	3.7

Age
20
18
20
18
19

Age
20
18
19

- The relational model is a set-based (***no duplicate tuples allowed***)

$$\pi_{<A1>}(\sigma_{<C1>}(R))$$

$$\sigma_{<C1>}(R)$$

$$\pi_{<A1>}(S)$$

$$(\sigma_{Age<20} (\text{Student})$$

ID	Short name	Age	GPA
344	**A J**	**20**	**3.8**
342	**B K**	**18**	**3.6**
767	**C E**	**20**	**3.2**
345	**D P**	**18**	**3.5**
234	**E U**	**19**	**3.7**

ID	Short name	Age	GPA
342	B K	18	3.6
345	D P	18	3.5
234	E U	19	3.7

- The relational model is a set-based (***no duplicate tuples allowed***)

$$\pi_{\text{Short name}} (\sigma_{\text{Age<20}} (\text{Student}))$$

ID	**Short name**	**Age**	**GPA**
344	**A J**	**20**	**3.8**
342	**B K**	**18**	**3.6**
767	**C E**	**20**	**3.2**
345	**D P**	**18**	**3.5**
234	**E U**	**19**	**3.7**

ID	**Short name**	**Age**	**GPA**
342	B K	18	3.6
345	D P	18	3.5
234	E U	19	3.7

Short name
B K
D P
E U

- The relational model is a set-based (***no duplicate tuples allowed***)

(b) Projection

Projection [$\pi_{<\text{attribute list}>}(R)$]

This is a unary operation (performed on one relation) that serves to select a subset of attributes from the relation R.

Relation R

A	B	C
a	b	c
c	a	d
c	b	d

Projection $\pi_{A,C}(R)$

A	C
a	c
c	d

- The degree of output relation (number of columns present) is equal to the number of attributes mentioned in the attribute list.
- Projection operator automatically **removes all the duplicates** while projecting the output relation.
- If there are no duplicates in the original relation, then the cardinality will remain same otherwise it will surely reduce.
- If attribute list is a super key on R, then we will always get the same number of tuples in output relation, because no duplicates to filter.

- Projection operator does not obey commutative property: $\pi_{<\text{list2}>}(\pi_{<\text{list1}>}(R)) \neq \pi_{<\text{list1}>}(\pi_{<\text{list2}>}(R))$
- Following expressions are equivalent because both finally projects columns of list1: $\pi_{<\text{list1}>}(\pi_{<\text{list2}>}(R)) \equiv \pi_{<\text{list1}>}(R)$
- Projection does not allow duplicates while SELECT operation allows; to avoid duplicates in SQL, we use “SELECT distinct”.

The Projection operation works on a single relation R and defines a relation that contains a vertical subset of R, extracting the values of specified attributes and eliminating duplicates.

Renaming operator - ρ

- The results of relational algebra are relations ***without names***.
- The rename operation allows us to ***rename the output relation***.

Renaming operator - ρ

- Sometimes it is necessary to use the same relation or the same attribute several times in a query, so you can use the renaming operator (unary operator).

$$\rho_{S}(R)$$

rename relation R into relation S

Renaming operator - ρ

- Sometimes it is necessary to use the same relation or the same attribute several times in a query, so you can use the renaming operator (unary operator).

$$\rho_{\text{<attribute1 new name} \leftarrow \text{attribute1 old name>}} (R)$$

Rename attribute1 of R from old name to new name

$$\rho_{\text{Student id} \leftarrow \text{ID}}\ \text{Student}$$

ID	Short name	Age	GPA
344	A J	20	3.8
342	B K	18	3.6
767	C E	20	3.2
345	D P	18	3.5
234	E U	19	3.7

Student id	Short name	Age	GPA
344	A J	20	3.8
342	B K	18	3.6
767	C E	20	3.2
345	D P	18	3.5
234	E U	19	3.7

- The relational model is a set-based (***no duplicate tuples allowed***)

$$\rho_{\text{FirstYear\_Students}} (\sigma_{\text{Age} = 18} (\text{Student}))$$

ID	Short name	Age	GPA
344	**A J**	**20**	**3.8**
342	**B K**	**18**	**3.6**
767	**C E**	**20**	**3.2**
345	**D P**	**18**	**3.5**
234	**E U**	**19**	**3.7**

Result: "FirstYear_Students"

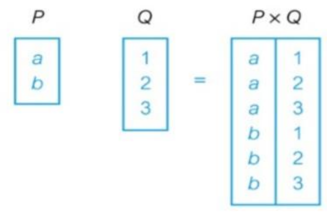
Student id	Short name	Age	GPA
342	B K	18	3.6
345	D P	18	3.5

- The relational model is a set-based (***no duplicate tuples allowed***)

$$\rho_{\text{FirstYear\_Students}} \, (\sigma_{\text{Age} = 18} \, (\text{Student}))$$

$$(\sigma_{\text{GPA} > 3.5} \, (\text{FirstYear\_Students}))$$

Cross-product
(Cartesian Product)



(c) Cartesian product

Cross-product (cartesian product)

- R x S returns a relation instance whose schema contains all the fields of R followed by all the fields of S – ***forming all possible combinations*** (fields of the same name are unnamed).

R

Rid	name
22	D W
31	L M
58	R S

X

S

Sid	Bid
20	109
39	102

=

Rid	name	Sid	Bid
22	D W	20	109
22	D W	39	102
31	L M	20	109
31	L M	39	102
58	R S	20	109
58	R S	39	102

- Cartesian product operation is both commutative and associative on relation algebra: D1XD2=D2xD1; D1x(D2xD3)=(D1xD2)xD3
- Cartesian product cardinality (tuples) =|D1xD2|=|D1|x|D2|;
- Cartesian product degree (attributes) =D1degree+D2degree

Cross-product (cartesian product)

Employee

Name	SSN
John	9999
Tony	7777

X

Dependent

ESSN	DName
9999	Emily
7777	Joe

=

Name	SSN	ESSN	DName
John	9999	9999	Emily
John	9999	7777	Joe
Tony	7777	9999	Emily
Tony	7777	7777	Joe

Assume the following relations:

```
BOOKS(DocId, Title, Publisher, Year)
STUDENTS(StId, StName, Major, Age)
AUTHORS(AName, Address)
borrows(DocId, StId, Date)
has-written(DocId, AName)
describes(DocId, Keyword)
```

- *List the year and title of each book.*

  $\pi_{\text{Year, Title}}(\text{BOOKS})$

- *List all information about students whose major is CS.*

  $\sigma_{\text{Major} = \text{'CS'}}(\text{STUDENTS})$

- *List all students with the books they can borrow.*

  $\text{STUDENTS} \times \text{BOOKS}$

- *List all books published by McGraw-Hill before 1990.*

  $\sigma_{\text{Publisher} = \text{'McGraw-Hill'} \wedge \text{Year} < 1990}(\text{BOOKS})$

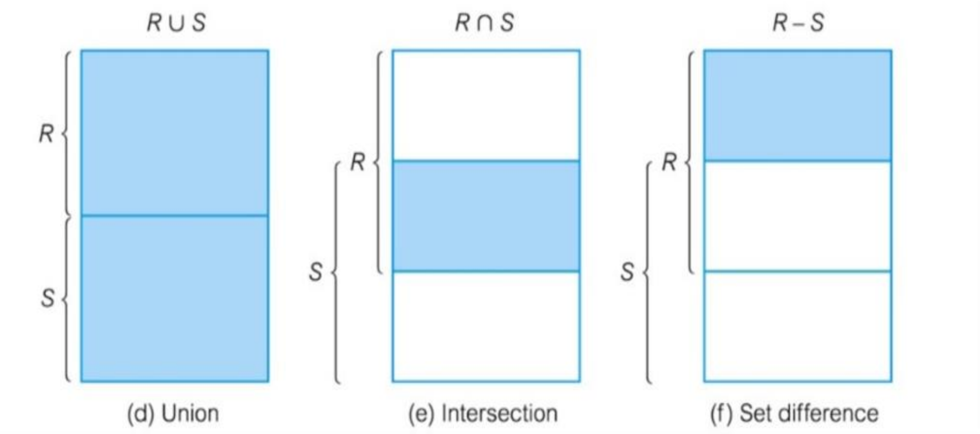
Relational Algebra operators from the Set theory

Relational Algebra operators from the Set theory

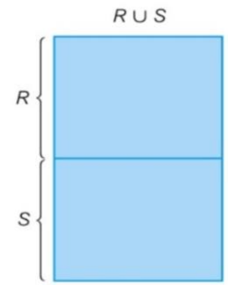
Union $\cup$

Intersection $\cap$

Difference $-$

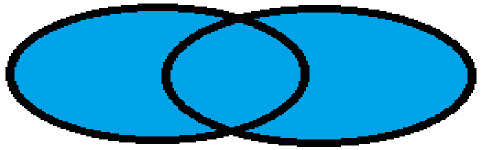


- The input relations ***must be compatible*** (must have the same number and names of attributes – same schema)
- The result will follows the input schema
- Duplicate tuples are eliminated.



(d) Union

The union binary operator:



- R ∪ S returns a relation instance containing all tuples that occur in either relation instance R **or** relation instance S (or both)

A	B
1	2
1	3
2	2

$\cup$

A	B
1	2
3	3
2	2

=

A	B
1	2
1	3
2	2
3	3

- $(R \cup S) \; = \; (S \cup R)$

- Union operation is both commutative and associative.
- In RUS, **duplicates** are automatically **removed**.

Professors	<u>**Pid**</u>	**name**	**room**	**rank**
Students	<u>**Sid**</u>	**name**	**semester**	**gpa**

Find the names of all teachers and students

$$\pi_{\text{name}}\,(\text{Professors}) \cup \pi_{\text{name}}\,(\text{Students})$$

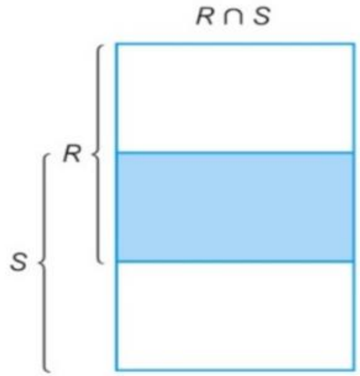
To union different schemas, rename fields

Professors	<u>**Pid**</u>	**name**	**room**	**rank**

Students	<u>**Sid**</u>	**name**	**semester**	**gpa**

Find the names and ids of all teachers and students

$$\rho_{\,id \leftarrow Pid}(\pi_{\,Pid,\ name}(\text{Prof})) \cup \rho_{\,id \leftarrow Sid}(\pi_{\,Sid,\ name}(\text{Stud}))$$



(e) Intersection

The intersection binary operator:

- R ∩ S returns a relation instance containing all tuples that occur in both relation instance R **and** relation instance S.
- (R-(R-S))

A	B
1	2
1	3
2	2

$\cap$

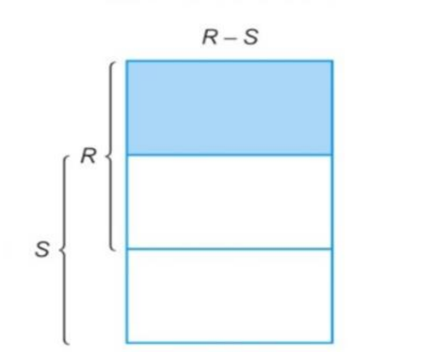
A	B
1	2
3	3
2	2

$=$

A	B
1	2
2	2

- (R ∩ S) = (S ∩ R)

The Difference binary operator:



(f) Set difference

- R — S returns a relation instance containing all tuples that occur in relation instance R **but not** in relation instance S

R

A	B
1	2
1	3
2	2

—

S

A	B
1	2
3	3
2	2

=

R — S

A	B
1	3

S — R

A	B
3	3

- $(R - S) \neq (S - R)$

The Symmetrical Difference binary operator:

$$(R \Delta S) = (R - S) \cup (S - R)$$

- **Determine all students who so far have not taken any exam**

Professors

<u>Pid</u>	name	room	rank

Students

<u>Sid</u>	name	semester	gpa

tests

<u>Sid</u>	<u>Lid</u>	<u>Pid</u>	grade

$$\pi_{\text{Sid}}(\text{students}) \quad - \quad \pi_{\text{Sid}}(\text{tests})$$

Assume the following relations:

```
BOOKS(DocId, Title, Publisher, Year)

STUDENTS(StId, StName, Major, Age)

AUTHORS(AName, Address)

borrows(DocId, StId, Date)

has-written(DocId, AName)

describes(DocId, Keyword)
```

- *List the name of students who are older than 30 and who are not studying CS.*

$$\pi_{\mathsf{StName}}(\sigma_{\mathsf{Age}>30}(\mathsf{STUDENTS})) - \pi_{\mathsf{StName}}(\sigma_{\mathsf{Major}='\mathsf{CS}'}(\mathsf{STUDENTS}))$$

Division

- $R \div S$
  - Defines a relation over the attributes C that consists of a set of tuples from R that match the combination of *every* tuple in S.

- Expressed using basic operations:

$$T_1 \leftarrow \Pi_{R-S}(R)$$

$$T_2 \leftarrow \Pi_{R-S}((S \ X \ T_1) - R)$$

$$T \leftarrow T_1 - T_2$$

Division – An Example

- Identify all clients who have viewed all properties with three rooms.

$$(\Pi_{clientNo,\ propertyNo}(Viewing)) \div$$
$$(\Pi_{propertyNo}(\sigma_{rooms\ =\ 3}\ (PropertyForRent)))$$

$\Pi_{clientNo,propertyNo}(Viewing)$

clientNo	propertyNo
CR56	PA14
CR76	PG4
CR56	PG4
CR62	PA14
CR56	PG36

$\Pi_{propertyNo}(\sigma_{rooms=3}(PropertyForRent))$

propertyNo
PG4
PG36

RESULT

clientNo
CR56

Join Operator

The Join Operator

- The most used operator in the relational algebra.

- The join operator allows us to establish ***connections among data*** in different relations.

- Three main versions of the join:
  1. Natural Join
  2. Theta Join
  3. Equi Join

Natural Join Operator: ⋈

- Assume relation R has attributes A_1, … A_m, **B_1, …, B_k**
- Assume relation S has attributes **B_1, … B_k**, C_1, …, C_n

$$R \bowtie S$$

$$\pi_{A1,..,Am,\mathbf{R.B1,..,R.Bk},C1,..,Cn}\left(\sigma_{R.B1=S.B1 \wedge \ldots \wedge R.Bk=S.Bk}\left(R \times S\right)\right)$$

Natural Join Operator: $\bowtie$

First step: R x S

R

A	B	C
A1	B1	C1
A2	B2	C2

X

S

C	D	E
C1	D1	E1
C3	D3	E3

=

R			S		
A	B	R.C	S.C	D	E
A1	B1	C1	C1	D1	E1
A1	B1	C1	C3	D3	E3
A2	B2	C2	C1	D1	E1
A2	B2	C2	C3	D3	E3

Natural Join Operator: ⋈

Second step: $\sigma_{R.C=S.C}(R x S)$

R			S		
A	B	R.C	S.C	D	E
A1	**B1**	**C1**	**C1**	**D1**	**E1**
A1	B1	C1	C3	D3	E3
A2	B2	C2	C1	D1	E1
A2	B2	C2	C3	D3	E3

R			S		
A	B	R.C	S.C	D	E
A1	**B1**	**C1**	**C1**	**D1**	**E1**

Natural Join Operator: ⋈

Third step: $\pi_{A,\ B,\ R.C,\ D,\ E}\ (\ \sigma_{R.C=S.C}\ (\ RxS\ )\ )$

R S

A	B	R.C	S.C	D	E
A1	**B1**	**C1**	**C1**	**D1**	**E1**

R S

A	B	C	D	E
A1	**B1**	**C1**	**D1**	**E1**

Natural Join Operator: ⋈

$$R \bowtie S = \pi_{A,\ B,\ R.C,\ D,\ E}\left(\ \sigma_{\mathbf{R.C=S.C}}\left(\ R \mathrm{x} S\ \right)\ \right)$$

R

A	B	C
A1	B1	C1
A2	B2	C2

⋈

S

C	D	E
C1	D1	E1
C3	D3	E3

R ⋈ S

A	B	C	D	E
A1	**B1**	**C1**	**D1**	**E1**

Natural Join Operator: ⋈

$$R \bowtie S = \pi_{A,\ R.B,\ C}\left(\sigma_{R.B=S.B} \left(R x S \right) \right)$$

R

A	B
X	Y
X	Z
Y	Z
Z	A

⋈

S

B	C
Z	U
A	B
Z	M

Natural Join Operator: ⋈

$$R \bowtie S = \pi_{A,\, R.B,\, C}\,(\,\sigma_{R.B=S.B}\,(\,R x S\,)\,)$$

R

A	B
X	Y
X	Z
Y	Z
Z	A

⋈

S

B	C
Z	U
A	B
Z	M

=

A	B	C
X	Z	U
X	Z	M
Y	Z	U
Y	Z	M
Z	A	B

Which lectures are held by which professors?

Professors	Pid	name	room	rank
Lectures	Lid	title	credits	Pid

Professors ⋈ Lectures

Result	Pid	name	room	rank	Lid	title	credits

Which lectures are held by which professors, in terms of the lecture title and professor name?

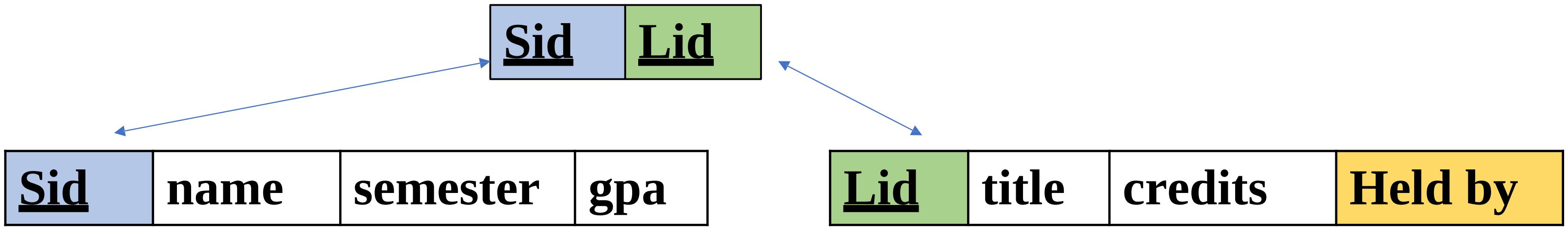
Professors	**Pid**	name	room	rank
Lectures	**Lid**	title	credits	Pid

$$\pi_{\text{name,title}} (\text{Professors} \bowtie \text{Lectures})$$

Which students attend which lectures?

Students	**<u>Sid</u>**	**name**	**semester**	**gpa**
Lectures	**<u>Lid</u>**	**title**	**credits**	**Held by**
Attends		**<u>Sid</u>**	**<u>Lid</u>**	

Which students attend which lectures?



Students ⋈ Attends ⋈ Lectures

Result

Sid	name	semester	gpa	Lid	title	credits	Held by

Theta Join Operator: $\bowtie_{condition}$

- Assume relation R has attributes $A_1, \ldots A_m$
- Assume relation S has attributes $B_1, \ldots B_k$

- The result schema has m + k attributes

$$R \bowtie_{condition} S \quad = \quad \sigma_{condition} (R \times S)$$

Theta Join Operator: $\bowtie_{condition}$

- Assume relation R has attributes $A_1, \ldots A_m$
- Assume relation S has attributes $B_1, \ldots B_k$

- The result schema has m + k attributes

$$R \bowtie_{R.Ai \,<\, S.Bj} S \quad = \quad \sigma_{R.Ai \,<\, S.Bj} (R \times S)$$

Theta Join Operator: $\bowtie_{condition}$

- Assume relation R has attributes $A_1, \ldots A_m$
- Assume relation S has attributes $B_1, \ldots B_k$

- The result schema has m + k attributes

$$R \bowtie_{R.Ai \;=\; S.Bj} S \quad = \quad \sigma_{R.Ai \;=\; S.Bj} \left(R \times S \right)$$

- When the condition involves equality check between certain attributes, the theta join is donated as **equi-join**

Inner Join	Inner join, includes only those tuples that satisfy the matching criteria.
Theta Join(θ)	The general case of JOIN operation is called a Theta join. It is denoted by symbol θ
EQUI Join	When a theta join uses only equivalence condition, it becomes a equi join.
Natural Join(⋈)	Natural join can only be performed if there is a common attribute (column) between the relations.
Outer Join	In an outer join, along with tuples that satisfy the matching criteria.
Left Outer Join(⟕)	In the left outer join, operation allows keeping all tuple in the left relation.
Right Outer join(⟖)	In the right outer join, operation allows keeping all tuple in the right relation.
Full Outer Join(⟗)	In a full outer join, all tuples from both relations are included in the result irrespective of the matching condition.

Which lectures are held by which professors?

Professors	**Pid**	name	room	rank
Lectures	**Lid**	title	credits	Held by

$$\textbf{Professors} \bowtie_{\text{Pid} = \text{Held\_by}} \textbf{Lectures}$$

OR

$$\textbf{Professors} \bowtie (\rho_{\text{Pid} \leftarrow \text{Held\_by}} \textbf{Lectures})$$

Popup quiz

Sailor	**<u>Sid</u>**	**Sname**	**Rating**	**age**

Reserve	**<u>Sid</u>**	**<u>Bid</u>**	**day**	**month**

Boat	**<u>Bid</u>**	**Bname**	**Color**	**Size**

Q1: Find names of sailors who have reserved boat with id 103?

$$\pi_{\text{Sname}} (\sigma_{\text{Bid} = 103} (\text{Reserve} \bowtie \text{Sailors}))$$

OR

$$\pi_{\text{Sname}} ((\sigma_{\text{Bid} = 103} \text{Reserve}) \bowtie \text{Sailors})$$

Popup quiz

Sailor	**<u>Sid</u>**	**Sname**	**Rating**	**age**
Reserve	**<u>Sid</u>**	**<u>Bid</u>**	**day**	**month**
Boat	**<u>Bid</u>**	**Bname**	**Color**	**Size**

Q2: Find names of sailors who have reserved red boat?

$$\pi_{\text{Sname}} \left(\sigma_{\text{Color} = \text{'red'}} \left(\textbf{Reserve} \bowtie \textbf{Sailors} \bowtie \textbf{Boat} \right) \right)$$

OR

$$\pi_{\text{Sname}} \left(\left(\sigma_{\text{color} = \text{'red'}} \textbf{Boat} \right) \bowtie \textbf{Sailors} \bowtie \textbf{Reserve} \right)$$

Popup quiz

Sailor	Sid	Sname	Rating	age

Reserve	Sid	Bid	day	month

Boat	Bid	Bname	Color	Size

Q3: Find the colors of boats reserved by John?

$$\pi_{\text{Color}} (\sigma_{\text{Sname} = \text{'john'}} (\textbf{Sailor} \bowtie \textbf{Reserve} \bowtie \textbf{Boat}))$$

OR

$$\pi_{\text{Color}} ((\sigma_{\text{Sname} = \text{'john'}} \textbf{Sailor}) \bowtie \textbf{Reserve} \bowtie \textbf{Boat})$$

Relation algebra and SQL

	Name	Relational Algebra	SQL
Basic / Complete Set	**select**	$\sigma_{age>18}(\text{patients})$	`select * from patients where age > 18`
	project	$\pi_{patient\_id,name}(\text{patients})$	`select patient_id, name from patients`
	product	$\text{patients} \times \text{medical\_records}$	`select * from patients,` `medical_records`

	union
	(note: union tables must have the same number of columns and same data types)
		$\pi_{name}(\text{patients}) \cup \pi_{name}(\text{doctors})$	`select name from patients`
			`union`
			`select name from doctors`

	difference
	(second table is not necessarily a subset of the first)
		$\pi_{name}(\text{patients}) - \pi_{name}(\text{doctors})$	`select name from patients`
			`minus`
			`select name from doctors`

Derived	**rename**	$\rho_{staff}(\pi_{patient\_id,name}(\text{patients}))$	`select * from (`
			`  select patient_id, name`
			`  from patients`
			`) as staff`

	intersection
	note: A intersect B = A-(A-B)
		$\pi_{name}(\text{patients}) \cap \pi_{name}(\text{doctors})$	`select name from patients`
			`intersect`
			`select name from doctors`

	natural join note: lecture notes use star, but every other source seems to use bowtie	$\text{patients} \bowtie \text{medical\_records}$ or $\text{patients} * \text{medical\_records}$	`select * from patients` `natural join medical_records`

	theta join
	note: you may sometimes see the '=' replaced with 'θ'
		$\text{patients} \bowtie_{patients.id=doctors.patient\_id} (\text{doctors})$	`select * from patients`
			`join doctors`
			`on patients.id=doctors.patient_id`