Relational Algebra:

- We can perform queries on a set of relations to get information from them. A **query** is a request for data or information from a relation. The input is a relation and the output is a new relation.
- An algebra is a mathematical system consisting of the following:
    1. **Operands:** Variables or values from which new values can be constructed.
    2. **Operators:** Symbols denoting procedures that construct new values from given values.
- **Relational algebra** is a widely used procedural query language. It collects instances of relations as input and gives occurrences of relations as output. It uses various operations to perform this action. It is an algebra whose operands are relations or variables that represent relations. Operators are designed to do the most common things that we need to do with relations in a database.
- Relational algebra operations are performed recursively on a relation. The output of these operations is a new relation, which might be formed from one or more input relations. Relational algebra operations do not modify the input relation in any way.

SELECT (σ):

- The SELECT operation is used for selecting a subset of the tuples according to a given selection condition. It is denoted by $\sigma_p(x)$. It is used as an expression to choose tuples which meet the selection condition. The select operation selects tuples that satisfy a given predicate.
- Notation: $\sigma_p(x)$
    - σ is the selection predicate.
    - x is the name of the relation.
    - p is the propositional logic. It is a boolean formula of terms and connectives. These connectives are: **^(and), V(or), ~(not)**.
      The operators are: **<, >, ≤, ≥, =, ≠**
      These terms are: **attribute operator attribute** and **attribute operator constant**.
- E.g. Consider the below relation.

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Instructor Relation

If we do $\sigma_{SALARY >= 85000}$(instructor), we get all the tuples with attribute salary at least 85000

from the instructor relation.
I.e. We would get this as the output:

ID	name	dept_name	salary
12121	Wu	Finance	90000
22222	Einstein	Physics	95000
33456	Gold	Physics	87000
83821	Brandt	Comp. Sci.	92000

PROJECTION (π):

- The projection operation gets the specified attributes from a relation.
- Notation: $\pi_{A1,\ A2,\ An}(r)$
    - π denotes the project operation.
    - A1, A2, An are the attributes in the relation, r.
    - r is the name of the relation.
- E.g. Consider the relation below:

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Instructor Relation

If we do: $\pi_{ID,salary}$(instructor), it would get the attributes ID and salary from the instructor relation.
I.e. This would be the output:

ID	salary
10101	65000
12121	90000
15151	40000
22222	95000
32343	60000
33456	87000
45565	75000
58583	62000
76543	80000
76766	72000
83821	92000
98345	80000

<u>NATURAL JOIN (⋈):</u>

- Combines two relations into a single relation.
- Can only be performed if there is a common attribute between the relations. The name and domain of the attribute must be the same. Note that if there is an entry in only one relation, it will be omitted from the result relation.
- Also called inner join.
- Notation: **r ⋈ s**
- E.g. Consider the two relations below:

Instructor Relation

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

and

Department Relation

dept_name	building	budget
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Since they both have the attribute dept_name and since the domain of both dept_name is string, if we do instructor ⋈ department, we get the following output:

ID	name	salary	dept_name	building	budget
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
12121	Wu	90000	Finance	Painter	120000
15151	Mozart	40000	Music	Packard	80000
22222	Einstein	95000	Physics	Watson	70000
32343	El Said	60000	History	Painter	50000
33456	Gold	87000	Physics	Watson	70000
45565	Katz	75000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
76543	Singh	80000	Finance	Painter	120000
76766	Crick	72000	Biology	Watson	90000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000

Theta-Join($\bowtie_C$):

- Theta join combines tuples from different relations provided they satisfy the theta condition.
- Notation: $r \bowtie_C s$
- R3 = R1 $\bowtie_C$ R2
  - Take the product R1 X R2.
  - Then apply σ_C to the result. As for σ, C can be any boolean-valued condition.
- E.g.

Sells(

bar,	beer,	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

)

Bars(

name,	addr
Joe's	Maple St.
Sue's	River Rd.

)

BarInfo := Sells $\bowtie_{\text{Sells.bar = Bars.name}}$ Bars

BarInfo(

bar,	beer,	price,	name,	addr
Joe's	Bud	2.50	Joe's	Maple St.
Joe's	Miller	2.75	Joe's	Maple St.
Sue's	Bud	2.50	Sue's	River Rd.
Sue's	Coors	3.00	Sue's	River Rd.

)

14

Left Outer Join ($\bowtie_L$):

- The left outer join operation allows keeping all tuple in the left relation. However, if there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.
- Notation: $R \bowtie_L S$

Right Outer Join ($\bowtie_R$):

- The right outer join operation allows keeping all tuple in the right relation. However, if there is no matching tuple is found in the left relation, then the attributes of the left relation in the join result are filled with null values.
- Notation: $A \bowtie_R B$

Full Outer Join ($\bowtie_O$):

- In a full outer join, all tuples from both relations are included in the result, irrespective of the matching condition.
- Notation: $A \bowtie_O B$

CARTESIAN PRODUCT (x):

- The cross product of 2 relations. The cross product produces all possible pairs of rows of the two relations.
- This is used to merge columns from two relations. Generally, a cartesian product is never a meaningful operation when it performs alone. However, it becomes meaningful when it is followed by other operations.
- Notation: **r x s**
- E.g. The cross product of {a, b} and {c, d} is {a,c}, {a,d}, {b,c} and {b,d}.

- E.g. Consider the 2 relations below:

A	B
α	1
β	2

r

and

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

If we do r x s, we get the following output:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

- A problem arises when the 2 relations share a same attribute name. How would we differentiate between the 2 attributes? We can rename the attributes of the relations.

RENAME (ρ):

- Notation: **$\rho_x(E)$**
    - E is the relation name.
    - x is what will be prepended to all the attribute names in relation E.
    - The rename operation renames all attributes in relation E by prepending them with x.
- E.g. Suppose the below table is the relation r.

A	B
a	1
b	2

If I do $P_r(r)$ x $P_s(r)$, we get the following relation:

r.A	r.B	s.A	s.B
a	1	a	1
a	1	b	2
b	2	a	1
b	2	b	2

UNION (U):

- Union operator when applied on two relations R1 and R2 will give a relation with tuples which are either in R1 or in R2. Furthermore, it eliminates all duplicate tuples.
  I.e. The tuples that are in both R1 and R2 will appear only once in the result relation.
- For a union operation to be valid, the following conditions must hold:
  1. The 2 relations must have the same **arity** (same number of attributes).
  2. The attribute domains must be compatible.
     I.e. The ith column of relation 1 must be of the same domain as the ith column of relation 2.
  3. Duplicate tuples are automatically eliminated.
- Notation: **r U s**
- E.g. Consider the 2 relations below:

A	B
α	1
α	2
β	1

r

and

A	B
α	2
β	3

s

If we do r U s, we will get the output:

A	B
α	1
α	2
β	1
β	3

- E.g. Consider the 2 relations below:

Table A		Table B	
column 1	column 2	column 1	column 2
1	1	1	1
1	2	1	3

If we do A U B, we get the following output:

Table A U B	
column 1	column 2
1	1
1	2
1	3

DIFFERENCE (-):

- Returns a relation consisting of all the tuples which are present in the first relation but are not in the second relation.
- Notation: **r - s**
- E.g. Consider the 2 relations below:

A	B
α	1
α	2
β	1

r

and

A	B
α	2
β	3

s

If we do r - s, we will get the following output:

A	B
α	1
β	1

- E.g. Consider the 2 relations below:

Table A		Table B	
column 1	column 2	column 1	column 2
1	1	1	1
1	2	1	3

If we do r - s, we will get the following output:

Table A - B	
column 1	column 2
1	2

INTERSECTION (∩):

- Defines a relation consisting of a set of all tuple that are in both A and B, where A and B are 2 relations.
- Notation: **A ∩ B**
- E.g. Consider the 2 relations below:

  r:

A	B
α	1
α	2
β	1

  and

  s:

A	B
α	2
β	3

  If we do r ∩ s, we will get the output:

A	B
α	2

- E.g. Consider the 2 relations below:

Table A		Table B	
column 1	column 2	column 1	column 2
1	1	1	1
1	2	1	3

  If we do A ∩ B, we will get the output:

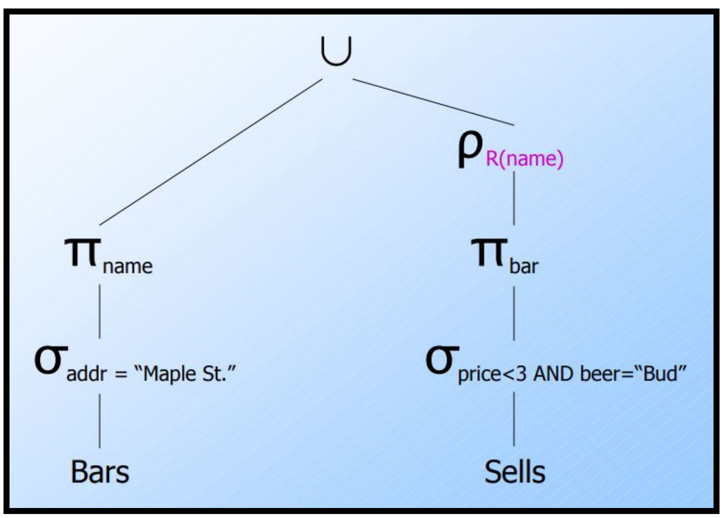
Table A ∩ B	
column 1	column 2
1	1

- **Note:** r ∩ s = r–(r–s)

Building Complex Expressions:

- Combine operators with parentheses and precedence rules.
- Three notations:
    1. **Sequences of assignment statements:**
        a. Create temporary relation names.
        b. Renaming can be implied by giving relations a list of attributes.
        c. E.g. R3 = R1 $\bowtie_C$ R2 can be written as:
        R4 = R1 X R2
        R3 = σ_C (R4)
    2. **Expressions with several operators:**
        a. E.g. R3 = R1 $\bowtie_C$ R2 can be written as R3 = σ_C (R1 X R2).
        b. Precedence of relational operators:
            1. [σ, π, ρ] (Highest)
            2. [X, ⋈]
            3. ∩
            4. [∪ , —] (Lowest)
    3. **Expression trees:**
        - Leaves are operands. Variables stand for relations.
        - Interior nodes are operators, applied to their child or children.
        - E.g. Using the relations Bars(name, addr) and Sells(bar, beer, price), find the names of all the bars that are either on Maple St. or sell Bud for less than $3.

        Solution:



Summary of Relational Algebra Operations:

Operation	**Purpose**
Select(σ)	The select operation is used for selecting a subset of the tuples according to a given selection condition
Projection(π)	The projection eliminates all attributes of the input relation but those mentioned in the projection list.
Union Operation(∪)	It includes all tuples that are in tables A or in B.
Difference(-)	The result of A - B, is a relation which includes all tuples that are in A but not in B.
Intersection(∩)	Intersection defines a relation consisting of a set of all tuple that are in both A and B.
Cartesian Product(X)	Cartesian product merges columns from two relations.
Theta Join($\bowtie_C$)	The general case of JOIN operation is called a Theta join.
Natural Join(⋈)	Natural join can only be performed if there is a common attribute (column) between the relations. Same as inner join.
Left Outer Join($\bowtie_L$)	In left outer join, the operation allows keeping all tuple in the left relation.
Right Outer join($\bowtie_R$)	In right outer join, the operation allows keeping all tuple in the right relation.
Full Outer Join ($\bowtie_O$)	In a full outer join, all tuples from both relations are included in the result, irrespective of the matching condition.
Rename(ρ)	Renames the attributes of the relation.