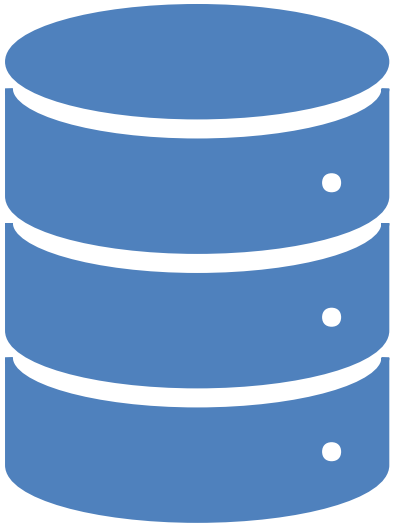
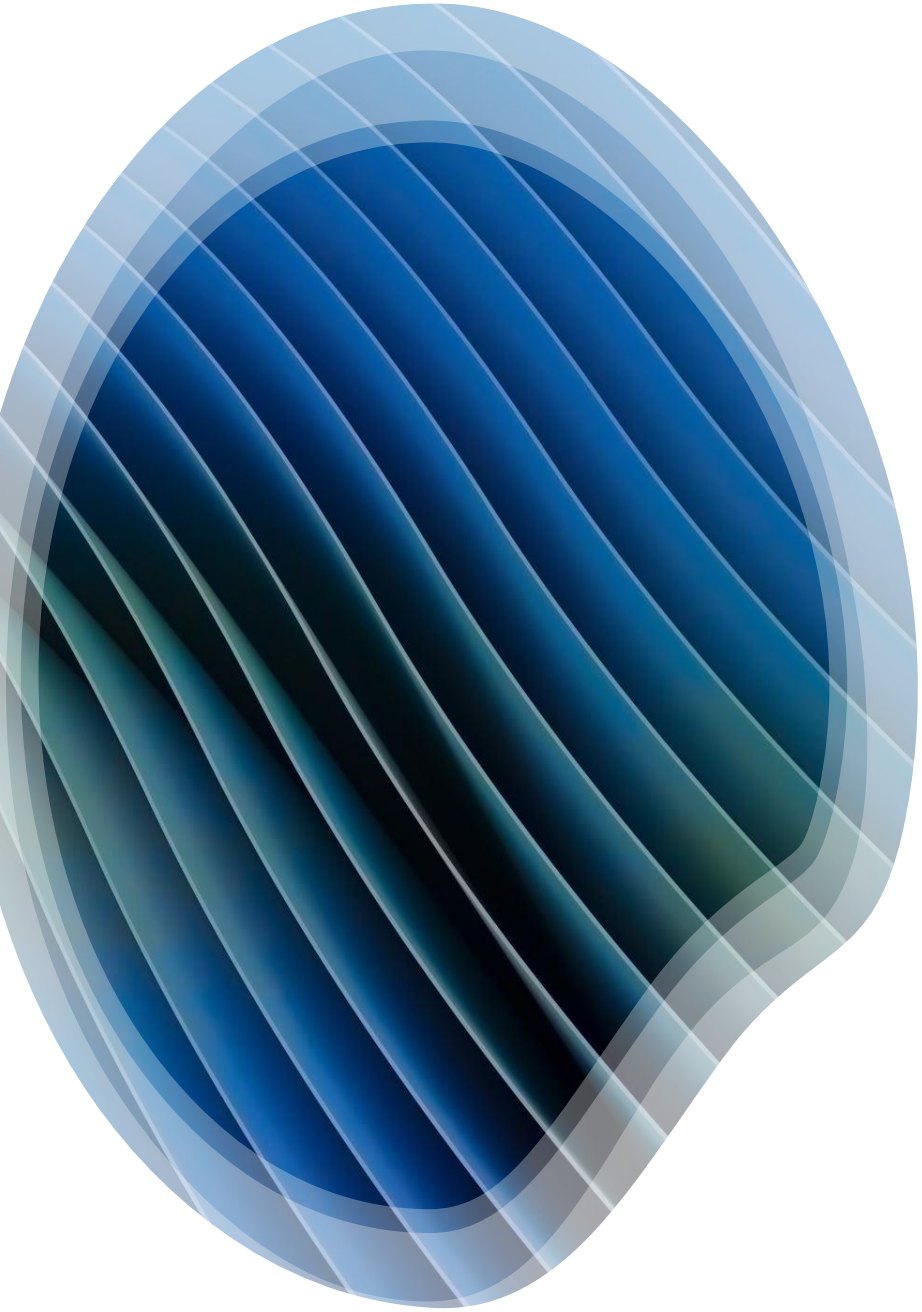


Query Language

Query Language

- A Language which is used to store and retrieve data from database is known as query language. For example – **SQL**
- There are two types of query language:
  1. Procedural Query language
  2. Non-procedural query language

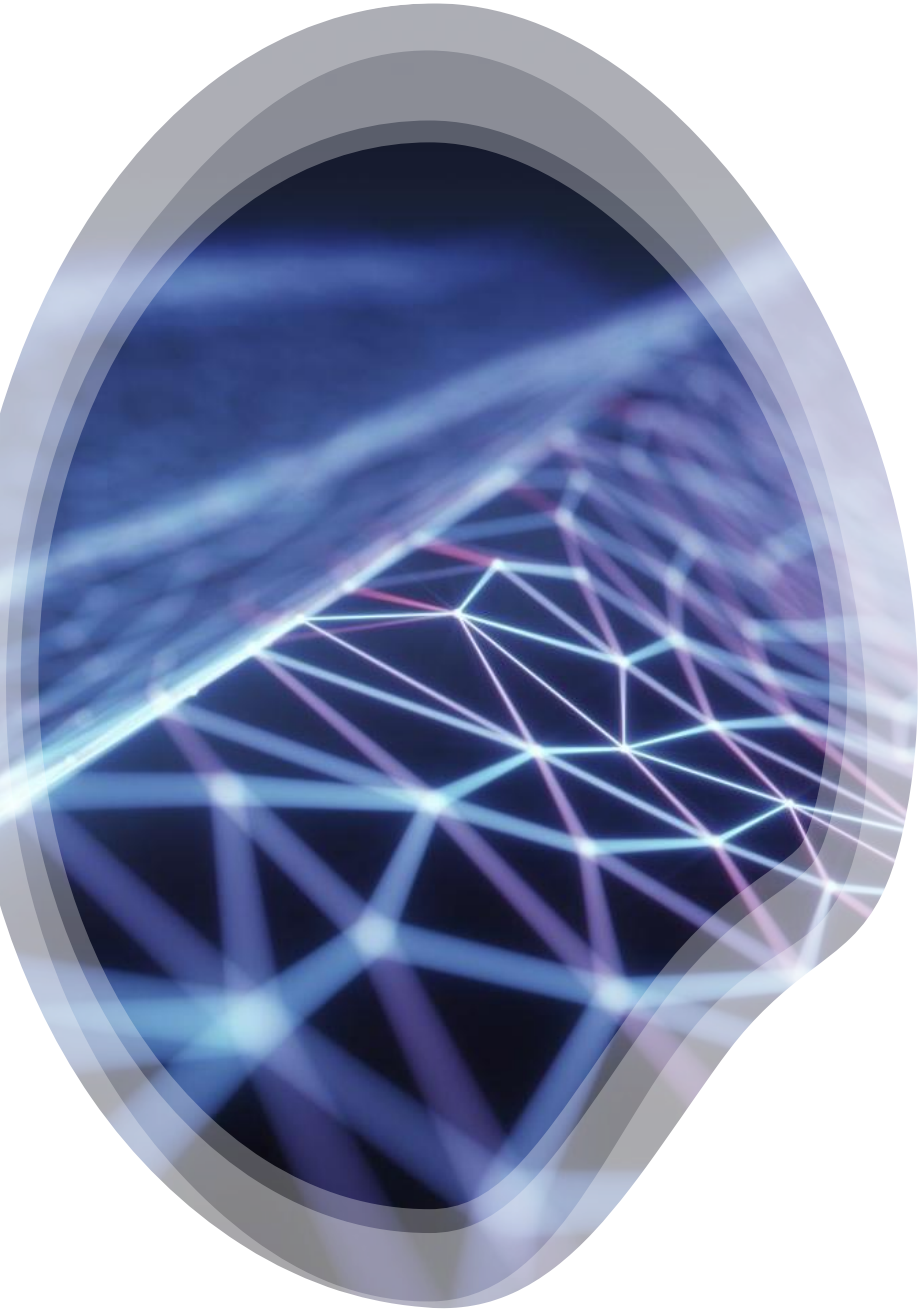


Procedural Query language

The user instructs the system to perform a series of operations to produce the desired results.

Here users tells what data to be retrieved from database and how to retrieve it.

For example – Making a cup of tea. If you are telling the step by step process like switch on the stove, boil the water, add milk etc. then it is a procedural language.

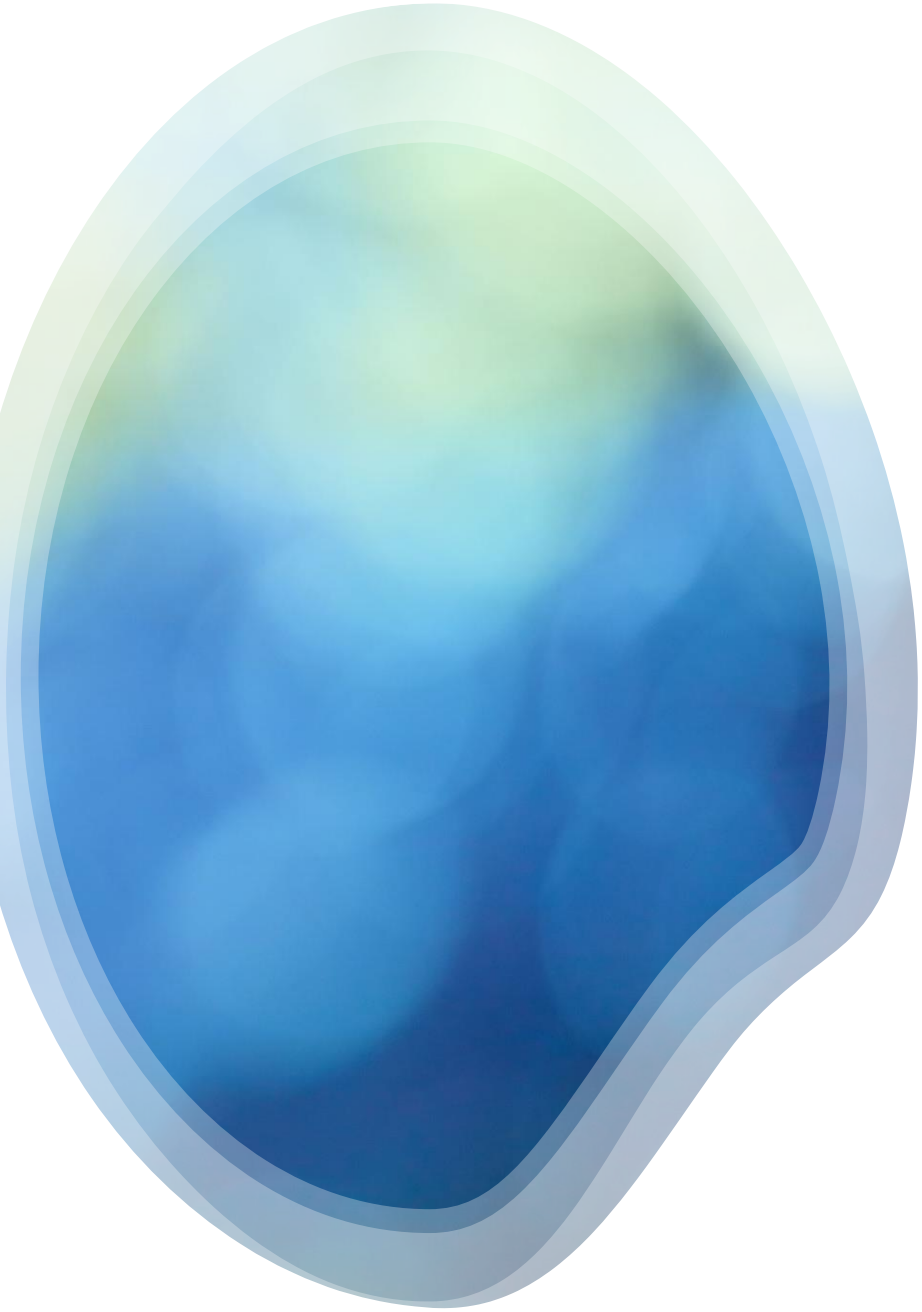


Non Procedural Query Language

The user instructs the system to produce the desired result without telling the step by step process.

Here users tells **what** data to be retrieved from database but doesn't tell **how** to retrieve it.

Example: if you are just telling someone to make a cup of tea and not telling the process



Conceptual Query Language

Theoretical mathematical system or query language, to store and retrieve data is called as Conceptual Query Language.

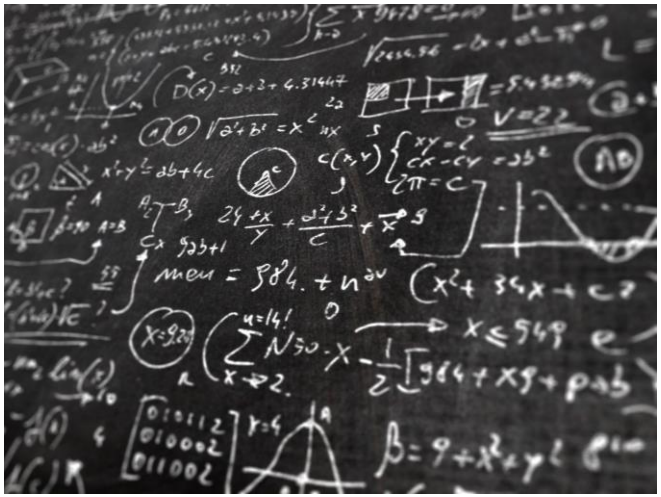
They are not the practical implementation.

Types:

- Conceptual Procedural Query Language. E.g., Relational Algebra
- Conceptual non-procedural query language. E.g., Relational calculus

Relational algebra and calculus are the theoretical concepts used on relational model.

SQL is a practical implementation of relational algebra and calculus.



Relational Algebra

Relational Algebra

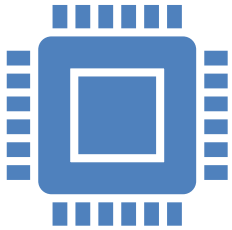
The relational algebra is a procedural query language or formal query language.

It consists of a set of operations that take one or two relations as input and produce a new relation as their result.

The select, project, and rename operations are called unary operations, because they operate on one relation.

The other operations operate on pairs of relations and are therefore called binary operations.

Operators



Basic Operators

Projection ($\prod$)

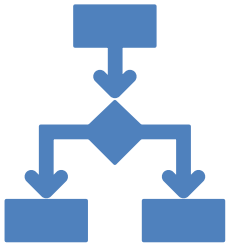
Selection (σ)

Cross Product (X)

Union (U)

Rename (ρ)

Set Difference (-)



Derived Operators

Join ($\bowtie$)

Intersection ($\cap$)

Division (/)

Project Operation – selection of columns (Attributes)

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}(r)$

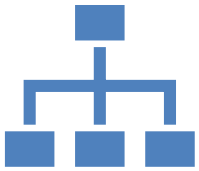
A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

No duplicate data

Project Operation – selection of columns (Attributes)



Roll	Name	Age
1	A	20
2	B	21
3	A	19

Relation *Student*:

Retrieve the roll no and name from table student.

$\prod_{roll,name}$ (*Student*)

Roll	Name
1	A
2	B
3	A

Select Operation – selection of rows (tuples)

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$\sigma_{(A=B) \wedge (D > 5)} (r)$

A	B	C	D
α	α	1	7
β	β	23	10

Select Operation – selection of rows (tuples)

- Relation *Student*:

Roll	Name	Age
1	A	20
2	B	21
3	A	19

- Select the name of the student whose Roll='2'.
- $\sigma_{roll='2'}$ (Student)

2	B	21

- Π_{name} ($\sigma_{roll='2'}$ (Student))

B

Cartesian-product / Cross Product

- Relations *r, s*:

A	B	C
1	2	3
2	1	4

C	D	E
3	4	5
2	1	2

- *r* x *s*: no of column =(no of column in r) + (no of column in s)

  = 3 + 3 = 6

  no of rows = (no of rows in r) * (no of rows in s)

  = 2 * 2 = 4

1st row r	1st row s
1st row r	2nd row s
2nd row r	1st row s
2nd row r	2nd row s

A	B	C	C	D	E
1	2	3	3	4	5
1	2	3	2	1	2
2	1	4	3	4	5
2	1	4	2	1	2

Cartesian-product

- Relations *r, s*:

A	*B*
α	1
β	2

r

C	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

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

Cartesian-product – naming issue

- Relations *r, s*:

A	*B*
α	1
β	2

r

B	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

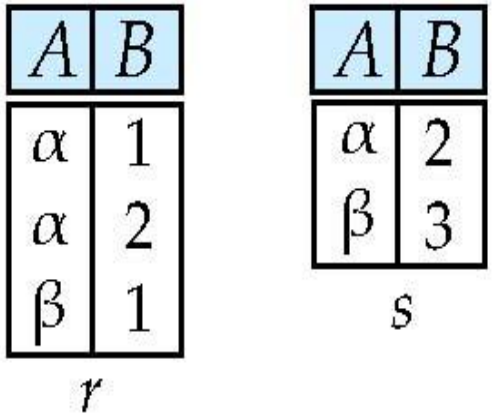
s

- *r* x *s*:

A	*r.B*	*s.B*	*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

Union of two relations

- Relations *r, s:*



A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r ∪ s:

A	B
α	1
α	2
β	1
β	3

No duplicate values

Union of two relations

- Relations *student, employee:*

Roll	Name
1	A
2	B
3	C

student

Emp.no	Name
7	E
1	A

Employee

- The no. of columns in both the tables should be same
- If Domain of attributes are same, then only union operation take place
- student ∪ employee:

Roll	Name
1	A
2	B
3	C
7	E

Column name should be similar to the 1st table (left hand side of the union operator)

Union of two relations

- Relations *student, employee:*

Roll	Name
1	A
2	B
3	C

student

Emp.no	Name
7	E
1	A

Employee

$\prod_{name}$ (student) $\cup$ $\prod_{name}$ (employee): The person who are student or employee or both

Name
A
B
C
E

Renaming a Table

- Allows us to refer to a relation, (say E) by more than one name.

$$\rho_x(E)$$

returns the expression *E* under the name *X*

- Relations *r*

A	B
α	1
β	2

r

- $r \times \rho_s(r)$

r.A	r.B	s.A	s.B
α	1	α	1
α	1	β	2
β	2	α	1
β	2	β	2

Product of similar tables (r,s) but different name

$r \times \rho_s(r) = (r \times s) = r \times r$

Set difference of two relations

- Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- *r* – *s*:

  In *r* set but not in *s* set

A	B
α	1
β	1

Set difference of two relations

- Relations *student, employee:*

Roll	Name
1	A
2	B
3	C

student

Emp.no	Name
7	E
1	A

Employee

- The no. of columns in both the tables should be same
- If Domain of attributes are same, then only difference operation take place
- student - employee:

Find the name of the person who are student but not employee

Roll	Name
2	B
3	C

Column name should be similar to the 1st table (left hand side of the difference operator)

Set difference of two relations

- Relations *student, employee:*

Roll	Name
1	A
2	B
3	C

student

Emp.no	Name
7	E
1	A

Employee

$\prod_{name}$ (student) - $\prod_{name}$ (employee): The person who are student but not employee

Name
B
C

Join

- Join is a special form of cross product of two tables.
- In Cartesian product we join a tuple of one table with the tuples of the second table. But in join there is a special requirement of relationship between tuples.
- For example, if there is a relation STUDENT and a relation BOOK then it may be required to know that how many books have been issued to any particular student. Now in this case, the primary key of STUDENT that is *stid* is a foreign key in BOOK table through which the join can be made.

Joining two relations – Natural Join

- Let r and s be relations on schemas R and S respectively.
  Then, the "natural join" of relations R and S is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from r and t_s from s.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result, where
    - t has the same value as t_r on r
    - t has the same value as t_s on s

Natural Join Example

- Relations r, s:

A	B	C
1	2	3
2	1	2
3	4	5

C	D
2	6
4	6

- Natural Join
  - r ⋈ s

$$\Pi_{A,\, r.B,\, C,\, r.D}(\sigma_{r.C = s.C}\ (r \times s))$$

Natural Join Example

- Relations r, s:

A	B	C
1	2	3
2	1	2
3	4	5

C	D
2	6
4	6

- Natural Join
  - r ⋈ s

$$\Pi_{A, r.B, C, r.D}(\sigma_{r.C = s.C}(r \times s))$$

A	B	R.C	S.C	D
1	2	3	2	6
1	2	3	4	6
2	1	2	2	6
2	1	2	4	6
3	4	5	2	6
3	4	5	4	6

Natural Join Example

- Relations r, s:

A	B	C
1	2	3
2	1	2
3	4	5

C	D
2	6
4	6

- Natural Join
  - r ⋈ s

$$\Pi_{A,\, r.B,\, C,\, r.D}(\sigma_{r.C = s.C}(r \times s))$$

A	B	R.C	S.C	D
1	2	3	2	6
1	2	3	4	6
2	1	2	2	6
2	1	2	4	6
3	4	5	2	6
3	4	5	4	6

Natural Join Example

- Relations r, s:

A	B	C
1	2	3
2	1	2
3	4	5

C	D
2	6
4	6

- Natural Join
  - r ⋈ s

A	B	R.C	D
2	1	2	6

$$\Pi_{A,\, r.B,\, C,\, r.D}(\sigma_{r.C = s.C}(r \times s))$$

Outer Join

- An outer join does not require each record in the two joined tables to have a matching record.
- The joined table retains each record – even if no other matching record exists.
- Outer joins subdivide further into left outer joins, right outer joins, and full outer joins, depending on which table (s) one retains the rows from (left, right, or both).

Left Outer Join

- The result of a left outer join or simply left join for table A and B always contains all records of left table A, even if the join-condition does not find any matching record in the right table B.
- This means that if the ON clause matches 0 records in B, the join will still return a row in the result, but with NULL in each column from B.
- If the right table returns one row and the left table returns more than one matching row for it, the values in the right table will be repeated for each distinct row on the left table.

LEFT JOIN Tables

cid	cname	cemail
1	A	A@mail.com
2	B	B@mail.com
3	C	C@mail.com
5	D	D@mail.com

customers

oid	orderdate	amount	cid
1	2020-04-04	100	1
2	2020-05-05	200	2
3	2020-06-06	300	1
4	2020-07-07	400	3
5	2020-08-08	500	4

select customers.cid, cname, amount from customers LEFT JOIN orders ON customers.cid = orders.cid;

customers  orders

customers.cid	cname	amount
1	A	100
2	B	200
1	A	300
3	C	400
5	D	NULL

Right Outer Join

- A right outer join or simply right join closely resembles a left outer join, except with the treatment of the tables reversed.
- Every row from the right table B will appear in the joined table at least once.
- If no matching row from the left table A exists, NULL will appear in columns from A for those records that have no matching in B.

RIGHT JOIN Tables

cid	cname	cemail
1	A	A@mail.com
2	B	B@mail.com
3	C	C@mail.com
5	D	D@mail.com

customers

oid	orderdate	amount	cid
1	2020-04-04	100	1
2	2020-05-05	200	2
3	2020-06-06	300	1
4	2020-07-07	400	3
5	2020-08-08	500	4

orders

select customers.cid, cname, amount from customers RIGHT JOIN orders ON customers.cid=orders.cid;

customers.cid	cname	amount
1	A	100
1	A	300
2	B	200
3	C	400
NULL	NULL	500

customers  orders

Full outer join

- Conceptually, a full outer join combines the effect of applying both left and right outer joins.
- Where records in the full outer joined tables do not match, the result set will have null values for every column of the table that lacks a matching row.

Full outer join

Name	EmpID	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales

Employee

DeptName	Mgr
Sales	Harriet
Production	Charles

Dept

Employee ⟗ Dept

Name	EmpID	DeptName	Mgr
Harry	3415	Finance	NULL
Sally	2241	Sales	Harriet
George	3401	Finance	NULL
Harriet	2202	Sales	Harriet
NULL	NULL	Production	Charles

Semi Join

- The left semijoin is joining similar to the natural join and written as R⋉ S where R and S are relation.
- The result of this semijoin is only the set of all tuples in R for which there is a tuple in S that is equal on their common attribute names.

Employee

Name	EmpID	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales

Dept

DeptName	Mgr
Sales	Harriet
Production	Charles

Name	EmpID	DeptName
Sally	2241	Sales
Harriet	2202	Sales

Anti Join

- The antijoin written as R ▷S where R and S are relations, is similar to the natural join, but the result of an antijoin is only those tuples in R for which there is no tuple in S that is equal on their common attribute names.

Employee

Name	EmpID	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales

Dept

DeptName	Mgr
Sales	Harriet
Production	Charles

DeptName	EmpId	DeptName
Harry	3415	Finance
George	3401	Finance

Self Join

Find student id from study table who is enrolled in at least two courses.

Study

S_id	C_id
S1	C1
S2	C2
S1	C2

1st step:
Study as T1

S_id	C_id
S1	C1
S2	C2
S1	C2

Study as T2

S_id	C_id
S1	C1
S2	C2
S1	C2

Step 2: T1 X T2

~~S1~~	~~C1~~	~~S1~~	~~C1~~
~~S1~~	~~C1~~	~~S2~~	~~C2~~
S1	C1	S1	C2
~~S2~~	~~C2~~	~~S1~~	~~C1~~
~~S2~~	~~C2~~	~~S2~~	~~C2~~
~~S2~~	~~C2~~	~~S1~~	~~C2~~
S1	C2	S1	C1
~~S1~~	~~C2~~	~~S2~~	~~C2~~
~~S1~~	~~C2~~	~~S1~~	~~C2~~

Step 3:
T1.S_id=T2.S_id AND T1.Cid <> T2.C_id

T1. S_id	T1. C_id	T2. S_id	T2. C_id
S1	C1	S1	C2
S1	C2	S1	C1

Step 4: Select T1.S_id

S_id
S1

Set intersection of two relations

- Relation *r, s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

Common in *r* and *s*

Set intersection of two relations

- Relation *r, s*:

A	*B*
α	1
α	2
β	1

r

A	*B*
α	2
β	3

s

- $r \cap s$

A	*B*
α	2

Note: $r \cap s = r - (r - s)$

Common in *r* and *s*

Set intersection of two relations

- Relations *student, employee:*

Roll	Name
1	A
2	B
3	C

student

Emp.no	Name
7	E
1	A

Employee

$\prod_{name}$ (student) $\cap$ $\prod_{name}$ (employee): The person who are student and also employee

Name
A

Set Division of two relations

- Retrieve Sid of students who enrolled in **every** course. (at all, for all-division is used)

Sid	Cid
S1	C1
S2	C1
S1	C2
S3	C2

Enrolled

Cid
C1
C2

Course

A(Sid,Cid)/B(Cid)
Enrolled(Sid,Cid)/Course(Cid)
=Sid

- A(x,y)/B(y)= it results *x* values: for that there should be a tuple <*x,y*> in *A* for every *y* value of relation *B.*
- *Step 1 :* $\prod_{Sid}$ (Enrolled) X $\prod_{cid}$ (Course)

  S1 *X* *C1*
  S2 *C2*
  S3

It results all students are enrolled in every courses

Set Division of two relations

- Retrieve Sid of students who enrolled in **every** course.

Sid	Cid
S1	C1
S2	C1
S1	C2
S3	C2

Enrolled

Cid
C1
C2

Course

- *Step 2:*

$(\Pi_{Sid}$ (Enrolled) X Π_{cid} (Course)) – Enrolled

S1	*C1*
S1	*C2*
S2	*C1*
S2	*C2*
S3	*C1*
S3	*C2*

–

S1	*C1*
S2	*C1*
S1	*C2*
S3	*C2*

Step 2: Result

$(\Pi_{Sid}$ (Enrolled) X Π_{cid} (Course)) – Enrolled

~~*S1*~~	~~*C1*~~			
~~*S1*~~	~~*C2*~~			
~~*S2*~~	~~*C1*~~		*S2*	*C2*
S2	*C2*	=	*S3*	*C1*
S3	*C1*			
~~*S3*~~	~~*C2*~~			

It provides d the set of students and their not enrolled courses

Step 3:

$\Pi_{Sid}((\Pi_{Sid}$ (Enrolled) X Π_{cid} (Course)) – Enrolled)

S2
S3

It provides Sid of disqualified students

Step 4:

Π_{Sid} (Enrolled)- $(\Pi_{Sid}((\Pi_{Sid}$ (Enrolled) X Π_{cid} (Course)) – Enrolled))

S1 S2
S2 - S3 = S1
S3

S1 enrolled in every course

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$

- $r \times s$

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

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

- $\sigma_{A=C}(r \times s)$

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

Summary of Relational Algebra Operators

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{salary >= 85000}(instructor)$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID, salary}(instructor)$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
x (Cartesian Product)	*instructor* x *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
∪ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the *two* input relations.
– (Set Difference)	$\Pi_{name}(instructor) - \Pi_{name}(student)$
	Output the set difference of tuples from the two input relations.
⋈ (Natural Join)	*instructor* ⋈ *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.

MCQs

A_____ is a query that retrieves rows from more than one table or view:

a) Start

b) End

c) Join

d) All of the mentioned

MCQs

A_____ is a query that retrieves rows from more than one table or view:

a) Start

b) End

c) Join

d) All of the mentioned

MCQs

Which product is returned in a join query have no join condition:

a) Equijoins

b) Cartesian

c) Both Equijoins and Cartesian

d) None of the mentioned

MCQs

Which product is returned in a join query have no join condition:

a) Equijoins

b) Cartesian

c) Both Equijoins and Cartesian

d) None of the mentioned

MCQs

Which join refers to join records from the right table that have no matching key in the left table are include in the result set:

a) Left outer join

b) Right outer join

c) Full outer join

d) Half outer join

MCQs

Which join refers to join records from the right table that have no matching key in the left table are include in the result set:

a) Left outer join

b) Right outer join

c) Full outer join

d) Half outer join

MCQs

The _____ operation, denoted by −, allows us to find tuples that are in one relation but are not in another.

a) Union

b) Set-difference

c) Difference

d) Intersection

MCQs

The _____ operation, denoted by −, allows us to find tuples that are in one relation but are not in another.

a) Union

b) Set-difference

c) Difference

d) Intersection

MCQs

The _____ operation, denoted by −, allows us to find tuples that are in one relation but are not in another.

a) Union

b) Set-difference

c) Difference

d) Intersection