

Álgebra Relacional

Bases de Datos 1 (CC3088) - 2026

Álgebra Relacional

Semestre 01, 2026

Contexto

- Las bases de datos almacenan información estructurada
- Necesitamos una forma formal de consultar los datos
- SQL se basa en un modelo matemático

Definición

- Lenguaje formal para manipular relaciones
- Creado por Edgar F. Codd
- Base teórica de SQL

Diferencia con SQL

- Álgebra relacional → declarativa matemática
- SQL → lenguaje práctico
- SQL implementa estos operadores

Recurden

- Relación → tabla
- Tupla → fila
- Atributo → columna
- Dominio → conjunto de valores posibles

Propiedades

- El resultado de cada operación es una relación
- Se pueden encadenar operaciones
- Es un sistema cerrado

Operadores básicos

Selección (σ)

- Filtra filas según una condición lógica
- Opera sobre tuplas
- No modifica columnas
- Reduce el número de filas

Forma general:

σ condición (Relación)

Ejemplo - Selección

Relación Estudiantes(id, nombre, carrera)

$\sigma \text{ carrera} = \text{"Ingeniería"} \text{ (Estudiantes)}$

Resultado:

Solo estudiantes cuya carrera sea Ingeniería.

seleccion

En álgebra:

$\sigma \text{ carrera} = \text{"Ingeniería"} \text{ (Estudiantes)}$

En SQL:

```
SELECT *  
FROM Estudiantes  
WHERE carrera = 'Ingeniería';
```

Proyección (π)

- Selecciona atributos específicos
- Reduce el número de columnas
- Puede eliminar duplicados
- No modifica filas directamente

Forma general:

$\pi \text{ atributos (Relación)}$

Ejemplo - Proyeccion

Relación Estudiantes(id, nombre, carrera)

$\pi \text{ nombre (Estudiantes)}$

Resultado:

Lista de nombres de estudiantes.

proyeccion

En álgebra:

π nombre (Estudiantes)

En SQL:

```
SELECT nombre  
FROM Estudiantes;
```

Unión (u)

- Combina tuplas de dos relaciones
- Requiere compatibilidad
 - Mismo número de atributos
 - Mismos dominios
- Elimina duplicados

Forma:

$R \cup S$

Ejemplo - Unión

Relación Ingeniería(id, nombre)

Relación Derecho(id, nombre)

Unvenieria U Derecho

Resultado:

Lista completa de estudiantes de Ingeniería y Derecho, sin duplicados.

union

En álgebra:

Ingeniería u Derecho

En SQL:

```
SELECT id, nombre FROM Ingenieria
UNION
SELECT id, nombre FROM Derecho;
```

Unión → UNION (elimina duplicados)

Diferencia (–)

- Devuelve tuplas que están en una relación
- Pero no en la otra
- Requiere compatibilidad

Forma:

$R - S$

Ejemplo 1

Relación Estudiantes(id, nombre, carrera)

Relación Ingeniería(id, nombre)

Estudiantes - Ingeniería

Resultado:

Lista de estudiantes que no pertenecen a la carrera de Ingeniería.

diferencia

En álgebra:

Estudiantes – Ingeniería

En SQL:

```
SELECT * FROM Estudiantes  
EXCEPT  
SELECT * FROM Ingenieria;
```

Alternativa con subconsulta:

```
SELECT id, nombre, carrera  
FROM Estudiantes  
WHERE id NOT IN (  
    SELECT id FROM Ingenieria  
);
```

Producto cartesiano (×)

- Combina cada tupla de R con cada tupla de S
- Multiplica el número de filas
- Base para joins (ya falta poco...)

Forma:

$R \times S$

Ejemplo - Producto cartesiano

Relación Estudiantes(id, nombre)

Relación Cursos(id_curso, nombre_curso)

Estudiantes×Cursos

Resultado:

Cada estudiante combinado con cada curso.

producto

En álgebra:

Estudiantes × Cursos

En SQL:

```
SELECT *  
FROM Estudiantes  
CROSS JOIN Cursos;
```

Producto cartesiano → CROSS JOIN

Operadores derivados

Intersección ($\cap$)

- Devuelve las tuplas que aparecen en ambas relaciones
- Requiere compatibilidad
- Es el equivalente lógico del AND entre conjuntos

Forma:

$R \cap S$

Ejemplo 1

Relación Estudiantes(id, nombre)

Relación Ingeniería(id, nombre)

Estudiantes n Ingeniería

Resultado:

Todos los registros que aparecen en ambas relaciones.

interseccion

En álgebra:

Estudiantes n Ingeniería

En SQL:

```
SELECT id, nombre
FROM Estudiantes
INTERSECT
SELECT id, nombre
FROM Ingenieria;
```

Join (⋈)

- Combina relaciones relacionadas
- Se basa en una condición de igualdad
- Es producto cartesiano + selección
- Es el operador más importante en bases de datos

Forma general:

$R \bowtie_{\text{condición}} S$

Ejemplo - Join

Relación Estudiantes(id, nombre)

Relación Inscripciones(id, id_curso)

Estudiantes $\bowtie$ Estudiantes.id = Inscripciones.id Inscripciones

Resultado:

Combina las tuplas de ambas relaciones cuando se cumple la condición de igualdad entre los atributos indicados.

join

En álgebra:

Estudiantes $\bowtie$ Estudiantes.id = Inscripciones.id Inscripciones

En SQL:

```
SELECT *  
FROM Estudiantes  
INNER JOIN Inscripciones  
ON Estudiantes.id = Inscripciones.id;
```

Join natural

- Es un caso especial de join
- Une automáticamente por atributos con el mismo nombre
- No requiere condición explícita
- Elimina columnas duplicadas

Forma:

$R \bowtie S$

Ejemplo - Join Natural

Relación Estudiantes(id, nombre) Relación Inscripciones(id, id_curso)

Estudiantes $\bowtie$ Inscripciones

Resultado:

El join natural combina automáticamente las tuplas de ambas relaciones utilizando los atributos que tienen el mismo nombre.

join

En álgebra:

Estudiantes $\bowtie$ Inscripciones

En SQL:

```
SELECT *  
FROM Estudiantes  
NATURAL JOIN Inscripciones;
```

División ($\div$)

- Responde consultas del tipo “para todos”
- Identifica elementos relacionados con todos los de otra relación
- Es uno de los operadores más abstractos

Forma:

$R \div S$

Ejemplo - División

Inscripciones(estudiante, curso)

CursosObligatorios(curso)

$\text{Inscripciones} \div \text{CursosObligatorios}$

Resultado:

Estudiantes que han tomado todos los cursos obligatorios.

division

En álgebra:

Inscripciones $\div$ CursosObligatorios

En SQL:

```
SELECT estudiante
FROM Incripciones i
GROUP BY estudiante
HAVING COUNT(DISTINCT curso) = (
    SELECT COUNT(*) FROM CursosObligatorios
);
```

Operadores auxiliares

Renombramiento (ρ)

- Cambia el nombre de una relación
- Puede cambiar nombres de atributos
- No modifica los datos
- Solo cambia el esquema

Forma general:

ρ NuevoNombre(Relación)

En álgebra:

ρ E(Estudiantes)

En SQL:

```
SELECT *  
FROM Estudiantes AS E;
```

Ejemplo 1 - Renombrar relaciones

Relación Estudiantes(id, nombre, carrera)

ρ E(Estudiantes)

Resultado:

La relación ahora se llama E, pero contiene exactamente los mismos datos.

Ejemplo 2 - Renombrar atributos

Relación Estudiantes(id, nombre, carrera)

ρ Est(id_est, nom, car)(Estudiantes)

Resultado:

Solo cambian los nombres, no las tuplas.

Composición de operaciones

- El álgebra relacional permite encadenar operadores
- Cada operación produce una relación
- Las expresiones pueden anidarse
- El orden de evaluación es importante

Precedencia

- Los operadores internos se evalúan primero
- Paréntesis determinan el orden
- Sin paréntesis, se aplica precedencia estándar
- Selección y proyección suelen aplicarse antes que unión o diferencia

Ejemplos

Ejemplo 1

ej1-enunciado

Resultado:

Nombres de estudiantes de Ingeniería.

Paso 1 - Estudiantes $\bowtie$ Inscripciones

Join por id.

id	nombre	carrera	id_curso		--	-----	-----	-----		
1	Ana	Ingeniería	10		1	Ana	Ingeniería	20		
2	Luis	Derecho	20		2	Luis	Derecho	20		
3	Marta	Ingeniería	10		3	Marta	Ingeniería	10		

Paso 2 - Join con Cursos

Join por id_curso.

id	nombre	carrera	id_curso	nombre_curso		--	-----	-----	-----		
1	Ana	Ingeniería	10	Bases		1	Ana	Ingeniería	20	Redes	
2	Luis	Derecho	20	Redes		3	Marta	Ingeniería	10	Bases	

Paso 3 - Selecccion nombre_curso = "Redes"

id	nombre	carrera	id_curso	nombre_curso
1	Ana	Ingeniería	20	Redes
2	Luis	Derecho	20	Redes

Paso 4 - Proyeccion nombre

nombre
Ana
Luis

ej1

```
SELECT DISTINCT e.nombre
FROM Estudiantes e
INNER JOIN Inscripciones i
    ON e.id = i.id
INNER JOIN Cursos c
    ON i.id_curso = c.id_curso
WHERE c.nombre_curso = 'Redes';
```

Ejemplo 2

ej2-enunciado

Resultado:

Todos los estudiantes que no están inscritos en el curso Redes.

Paso 1 - π nombre(Estudiantes)

Proyección de todos los estudiantes.

nombre
Ana
Luis
Marta

Paso 2 - Estudiantes $\bowtie$ Inscripciones

Join por id.

id nombre carrera id_curso	--	-----	-----	-----	1 Ana Ingeniería
10	1 Ana Ingeniería	20	2 Luis Derecho	20	3 Marta Ingeniería
10					

Paso 3 - Join con Cursos

Join por id_curso.

```
| id | nombre | carrera | id_curso | nombre_curso | | -- | ----- | ----- | ----- |
----- | | 1 | Ana | Ingeniería | 10 | Bases | | 1 | Ana | Ingeniería | 20 | Redes | | 2
| Luis | Derecho | 20 | Redes | | 3 | Marta | Ingeniería | 10 | Bases |
```

Paso 4 - Seleccin nombre_curso = "Redes"

```
| id | nombre | carrera | id_curso | nombre_curso | | -- | ----- | ----- | ----- |
----- | | 1 | Ana | Ingeniería | 20 | Redes | | 2 | Luis | Derecho | 20 | Redes |
```

Paso 5 - π nombre (estudiantes inscritos en Redes)

nombre	-----	Ana	Luis
--------	-------	-----	------

Paso 6 - Diferencia

Todos los estudiantes – Estudiantes inscritos en Redes

```
| nombre | | ----- | | Marta |
```

ej2

```
SELECT DISTINCT e.nombre
FROM Estudiantes e
WHERE e.id NOT IN (
    SELECT e2.id
    FROM Estudiantes e2
    INNER JOIN Inscripciones i
```

```

        ON e2.id = i.id
    INNER JOIN Cursos c
        ON i.id_curso = c.id_curso
    WHERE c.nombre_curso = 'Redes'
);

```

Ejemplo 3

ej3-enunciado

Resultado:

Nombres de estudiantes de Ingeniería que están inscritos en el curso Bases.

Paso 1 - Estudiantes $\bowtie$ Inscripciones

Join por id.

id	nombre	carrera	id_curso	--	-----	-----	-----
1	Ana	Ingeniería	10	1	Ana	Ingeniería	20
2	Luis	Derecho	20	3	Marta	Ingeniería	10

Paso 2 - Join con Cursos

Join por id_curso.

id	nombre	carrera	id_curso	nombre_curso	--	-----	-----	-----
1	Ana	Ingeniería	10	Bases	1	Ana	Ingeniería	20
2	Luis	Derecho	20	Redes	3	Marta	Ingeniería	10

Paso 3 - Selecccion carrera = "Ingeniería" $\wedge$ nombre_curso = "Bases"

id	nombre	carrera	id_curso	nombre_curso	--	-----	-----	-----	
1	Ana	Ingeniería	10	Bases	3	Marta	Ingeniería	10	Bases

Paso 4 - Proyeccion nombre

nombre	-----
Ana	
Marta	

ej3

```
SELECT DISTINCT e.nombre
FROM Estudiantes e
INNER JOIN Inscripciones i
    ON e.id = i.id
INNER JOIN Cursos c
    ON i.id_curso = c.id_curso
WHERE e.carrera = 'Ingeniería'
AND c.nombre_curso = 'Bases';
```