



# PROYECTO DE SQL

DIEGO CANTALAPIEDRA  
ALEXANDER RICART  
REINER FUENTES





# ÍNDICE

01

Contexto

02

Diagrama  
Lógico

03

Diagrama  
Entidad  
Relación

04

Querys

# CONTEXTO



01

No sólo se pueden añadir alumnos y profesores.

También **se pueden añadirse nuevas promociones, campuses, modalidades, verticales, roles, proyectos, e incluso calificaciones.**

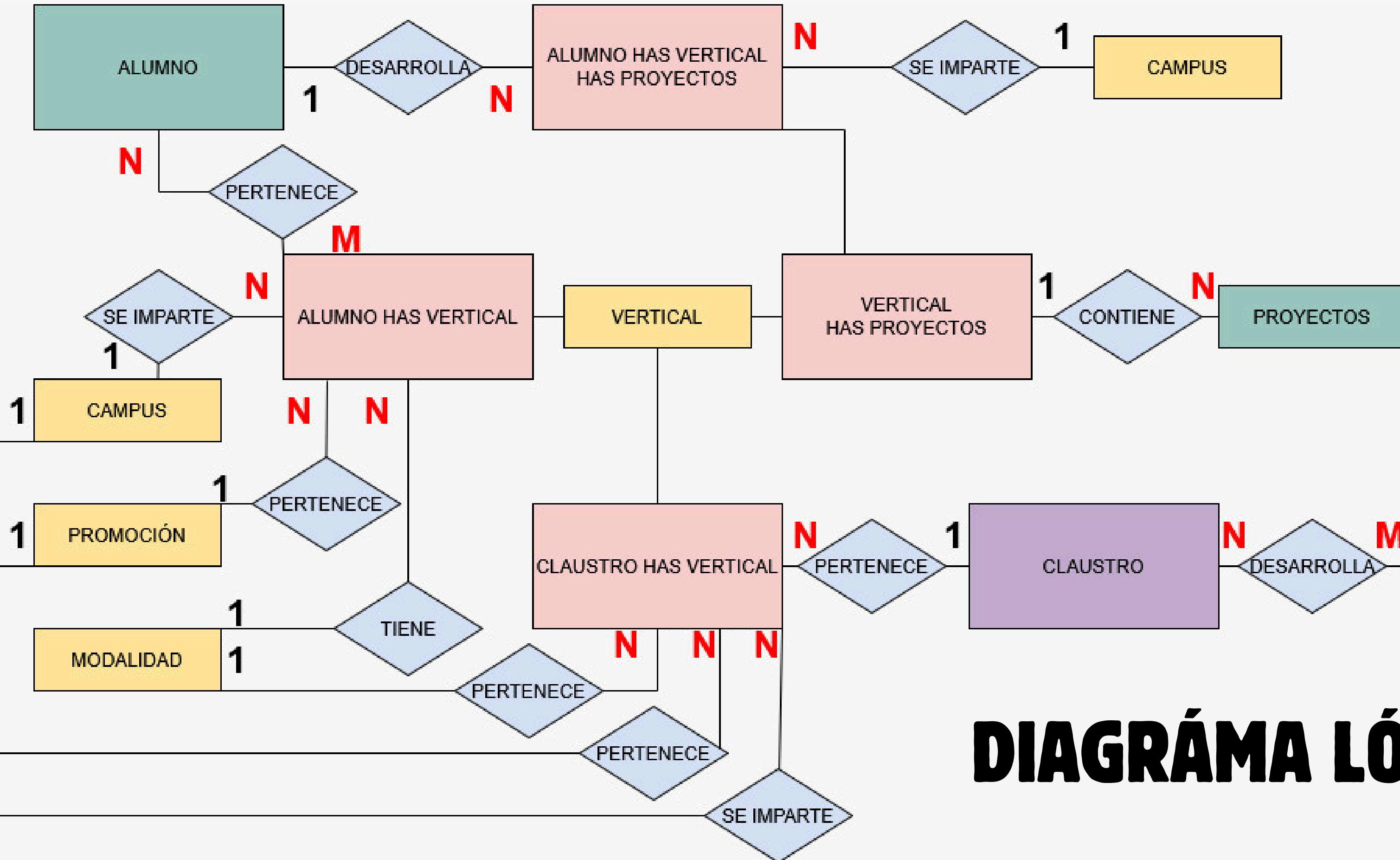
02

**Una vertical puede tener n proyectos, y estos pueden ordenar como se desee.**

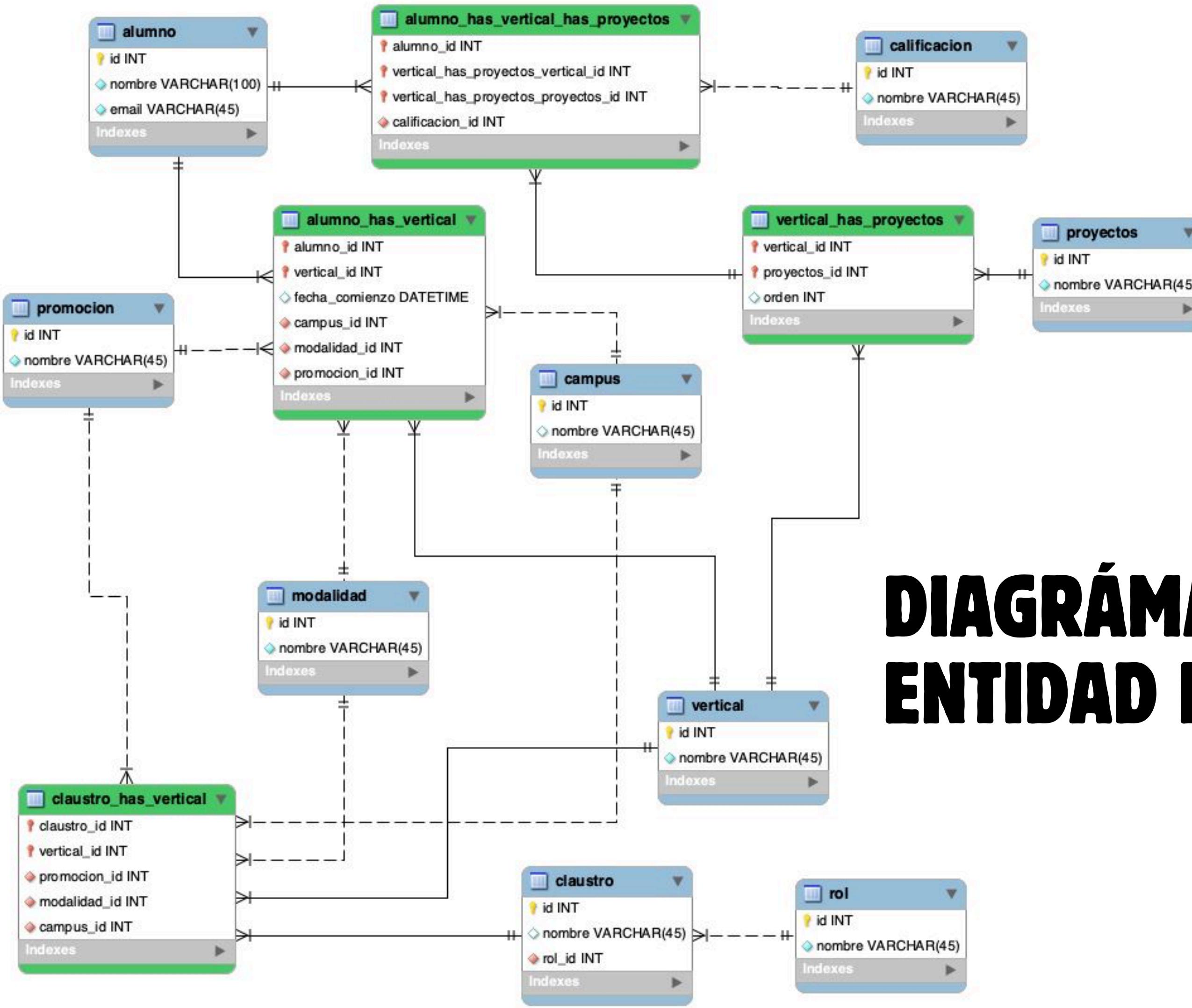
**Varias verticales pueden compartir uno o más proyectos que puedan ser comunes (desafío de tripulaciones).**

03

- Un mismo alumno puede asociarse a una o varias verticales.**
- Un mismo profesor puede asociarse a una o varias verticales.**



# DIAGRÁMA LÓGICO



# DIAGRÁMA ENTIDAD RELACIÓN

# TABLAS

```
SELECT table_name  
FROM information_schema.tables  
WHERE table_schema = 'the_bridge'  
ORDER BY table_name;
```

	table_name	name
1	alumno	
2	alumno_has_vertical	
3	alumno_has_vertical_has_proyecto	
4	calificacion	
5	campus	
6	claustro	
7	claustro_has_vertical	
8	modalidad	
9	promocion	
10	proyecto	
11	rol	
12	vertical	
13	vertical_has_proyecto	

# QUERYS

## Conteo de alumnos por vertical

```
SELECT
    v.nombre AS vertical,
    COUNT(DISTINCT av.alumno_id) AS total_alumnos
FROM the_bridge.alumno_has_vertical av
JOIN the_bridge.vertical v ON v.id = av.vertical_id
GROUP BY v.nombre
ORDER BY total_alumnos DESC;
```

¿Cuántos estudiantes por vertical?

1. Usa DISTINCT para no contar dos veces al mismo alumno si aparece más de una vez.
2. Ordena de mayor a menor cantidad de alumnos.

## Relación Docentes-Verticales

```
SELECT cl.nombre AS docente, v.nombre AS vertical
FROM the_bridge.claustro cl
JOIN the_bridge.claustro_has_vertical cv ON cl.id = cv.claustro_id
JOIN the_bridge.vertical v ON cv.vertical_id = v.id;
```

¿Qué docentes están asociados a cada vertical?

- JOIN the\_bridge.claustro\_has\_vertical cv ON cl.id = cv.claustro\_id
- Relaciona cada docente con sus verticales usando la tabla intermedia.

# QUERYS

**Información de los alumnos por campus, modalidad y promoción**

```
SELECT
```

```
    a.nombre AS alumno,  
    c.nombre AS campus,  
    m.nombre AS modalidad,  
    p.nombre AS promocion  
FROM the_bridge.alumno a  
JOIN the_bridge.alumno_has_vertical av ON a.id = av.alumno_id  
JOIN the_bridge.campus c ON c.id = av.campus_id  
JOIN the_bridge.modalidad m ON m.id = av.modalidad_id  
JOIN the_bridge.promocion p ON p.id = av.promocion_id  
ORDER BY a.nombre;
```

- Muestra los datos principales de los alumnos: nombre, campus, modalidad y promoción.
- Ordena la lista por nombre de alumno.

**¿Cuál es la información básica de cada alumno y en qué campus/modalidad/promoción está?**

# QUERYS

## Rendimiento de alumnos por proyectos

```
SELECT
    a.nombre AS alumno,
    c.nombre AS campus,
    m.nombre AS modalidad,
    pr.nombre AS promocion,
    COUNT(*) FILTER (WHERE cal.nombre = 'Apto') AS total_apto,
    COUNT(*) AS total_proyectos,
    ROUND(100.0 * COUNT(*) FILTER (WHERE cal.nombre = 'Apto') / COUNT(*), 2) AS porcentaje_aprobacion
FROM the_bridge.alumno_has_vertical_has_proyecto avp
JOIN the_bridge.alumno a ON avp.alumno_id = a.id
JOIN the_bridge.calificacion cal ON avp.calificacion_id = cal.id
JOIN the_bridge.alumno_has_vertical av
    ON a.id = av.alumno_id
    AND av.vertical_id = avp.vertical_has_proyecto_vertical_id
JOIN the_bridge.campus c ON av.campus_id = c.id
JOIN the_bridge.modalidad m ON av.modalidad_id = m.id
JOIN the_bridge.promocion pr ON av.promocion_id = pr.id
GROUP BY a.nombre, c.nombre, m.nombre, pr.nombre
ORDER BY porcentaje_aprobacion DESC, a.nombre;
```

- Calcula cuántos proyectos de cada alumno fueron “Apto” y el porcentaje de aprobación.
- Incluye campus, modalidad y promoción para contextualizar.
- Ordena primero por el porcentaje de aprobación (de mayor a menor) y luego por nombre.

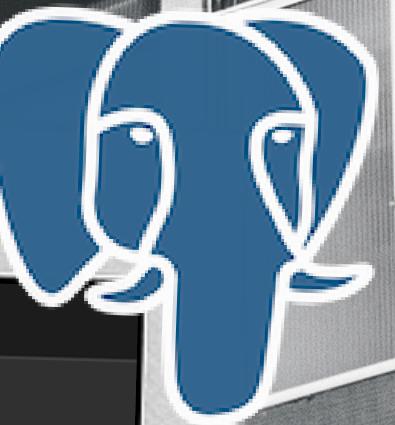
**¿Cómo ha rendido cada alumno en sus proyectos y cuál es su porcentaje de aprobación?**

# QUERYS

## Rendimiento por Vertical

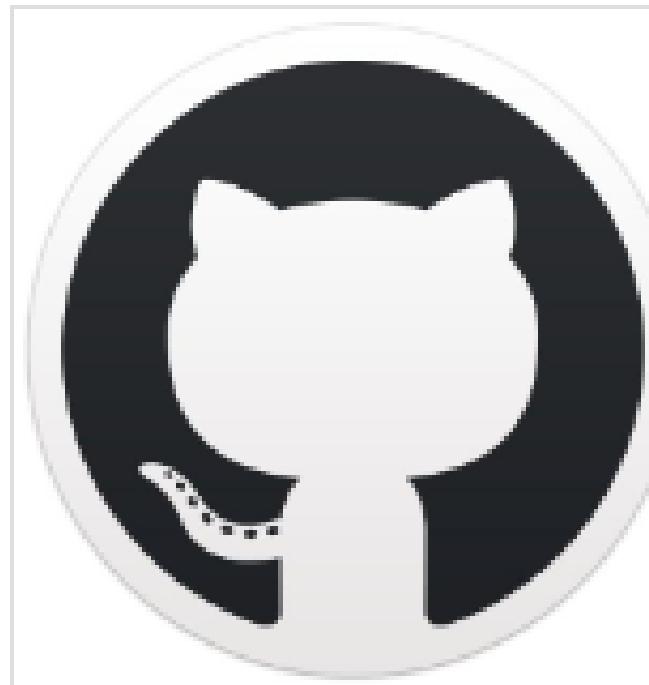
```
SELECT
    v.nombre AS vertical,
    COUNT(*) FILTER (WHERE cal.nombre = 'Apto') AS total_apto,
    COUNT(*) AS total_proyectos,
    ROUND(100.0 * COUNT(*) FILTER (WHERE cal.nombre = 'Apto') / COUNT(*),2) AS porcentaje_aprobacion
FROM the_bridge.alumno_has_vertical_has_proyecto avp
JOIN the_bridge.vertical v ON avp.vertical_has_proyecto_vertical_id = v.id
JOIN the_bridge.calificacion cal ON avp.calificacion_id = cal.id
GROUP BY v.nombre
ORDER BY porcentaje_aprobacion DESC;
```

- Calcula el total de proyectos “Apto” y porcentaje de aprobación, pero agregado por vertical, no por alumno.
- Ordena las verticales de mayor a menor porcentaje de aprobación.



vamos a ver si es verdad

# GRACIAS



**cantalaweb/SQL\_project**

Contribute to [cantalaweb/SQL\\_project](#) development  
by creating an account on GitHub.

 GitHub