

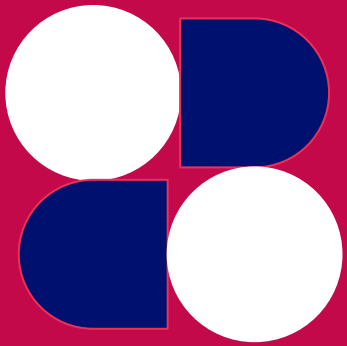


Programa PUCP DATA DRIVEN

Formación tecnológica

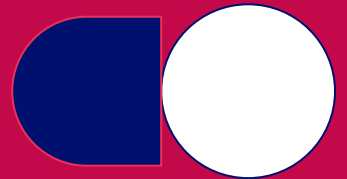
Dirección Académica de Planeamiento y Evaluación
Dirección de Gestión del Talento Humano





SESIÓN 7:

Subconsultas





Al finalizar la sesión, lograrás:

- Conocer y aplicar los diferentes tipos de subconsultas
- Implementar formas de organización de subconsultas optimas
- Implementar operadores avanzados como CASE y ROLLUP



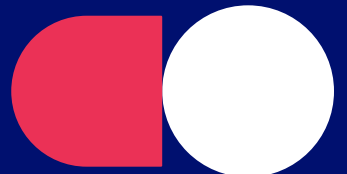
Contenido de la sesión:



- 1 Consultas Anidadas
- 2 Subconsulta en WHERE
- 3 Subconsulta Escalar
- 4 Subconsulta en FROM
- 5 Clausula WITH
- 6 Operador ROLLUP
- 7 Taller



1. Consultas anidadas (Subconsultas)





CONSULTAS ANIDADAS



Concepto

- SQL proporciona un mecanismo para el **anidamiento** de *subqueries*. Un *subquery* es una expresión select-from-where que está dentro de otra consulta.





CONSULTAS ANIDADAS



Concepto

- El anidamiento se puede realizar en la siguiente consulta SQL

```
SELECT A1,A2, ...,An  
FROM r1, r2, ..., rm  
WHERE P
```

donde:

- Clausula **FROM**: ri puede ser reemplazado por cualquier **subconsulta válida**
- Clausula **WHERE**: P se puede reemplazar con una expresión:
B <operation> (**subquery**)
- Clausula **SELECT**: Ai se puede reemplazar por una subconsulta que genera un único valor.

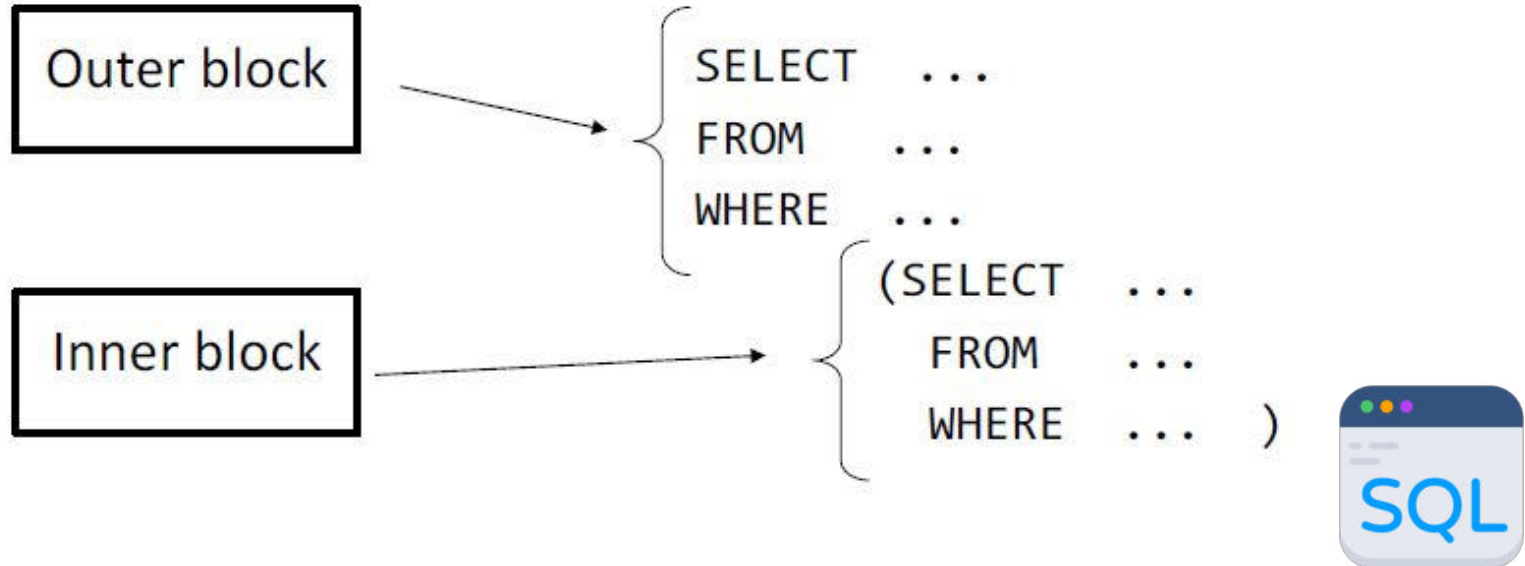




CONSULTAS ANIDADAS

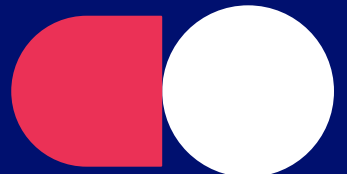


Concepto





2. Subconsultas en WHERE





SUBCONSULTAS EN WHERE



Concepto

- Una subconsulta en **WHERE** se coloca dentro de la cláusula WHERE de una consulta principal y devuelve un valor o un conjunto de valores que se usan para filtrar los resultados de la consulta principal.
- La subconsulta puede devolver:
 - Un solo valor (utilizado en una comparación).
 - Un conjunto de valores (utilizado con operadores como **IN**).
 - Puede estar relacionada con la consulta externa o independiente de ella.





SUBCONSULTAS EN WHERE



Operador “IN” / “NOT IN”

- El operador **IN** y su contraparte **NOT IN** se usa para comprobar si un valor existe o no dentro de un conjunto de valores, que pueden ser valores estáticos o el resultado de una subconsulta.
- Ejemplo: Encuentre los id de los horarios que se están ofreciendo para los cursos 10, 20 y 32

```
SELECT [id_horario]  
FROM [model].[faci-inc_tb_horarios]  
WHERE [id_curso] IN (10, 20, 32);
```





SUBCONSULTAS EN WHERE



Ejemplos

- Encuentre los nombres y los apellidos de los estudiantes que están inscritos en el curso con el mayor número de créditos.

```
SELECT E.nombres_estudiante  
      , E.apellidos_estudiante  
FROM [model].[faci-inc_tb_estudiantes] AS E  
     JOIN [model].[faci-inc_tb_inscripciones] AS I  
         ON E.id_estudiante = I.id_estudiante  
     JOIN [model].[faci-inc_tb_horarios] AS H  
         ON I.id_horario = H.id_horario  
     JOIN [model].[faci-inc_tb_cursos] AS C  
         ON H.id_curso = C.id_curso  
WHERE C.creditaje = (  
    SELECT MAX(creditaje)  
    FROM [model].[faci-inc_tb_cursos]);
```





SUBCONSULTAS EN WHERE



Ejemplos

- Encuentre los id de los docentes que imparten el curso con el mayor número de créditos.

```
SELECT DISTINCT H.id_docente
FROM [model].[faci-inc_tb_horarios] AS H
JOIN [model].[faci-inc_tb_cursos] AS C
ON H.id_curso = C.id_curso
WHERE C.creditaje = (
    SELECT MAX(creditaje)
FROM [model].[faci-inc_tb_cursos]);
```





SUBCONSULTAS EN WHERE



Ejemplos

- Encuentre los nombres y apellidos de los estudiantes que están inscritos en cursos que pertenecen al ciclo 2024-2

```
SELECT DISTINCT E.nombres_estudiante  
                , E.apellidos_estudiante  
FROM [model].[faci-inc_tb_estudiantes] AS E  
     JOIN [model].[faci-inc_tb_inscripciones] AS I  
         ON E.id_estudiante = I.id_estudiante  
     JOIN [model].[faci-inc_tb_horarios] AS H  
         ON I.id_horario = H.id_horario  
WHERE H.id_curso IN (  
    SELECT id_curso  
    FROM [model].[faci-inc_tb_cursos]  
    WHERE ciclo = '2024-2');
```





SUBCONSULTAS EN WHERE



Ejemplos

- Encuentre los id de los docentes que están impartiendo cursos que pertenecen al ciclo 2024-2

```
SELECT DISTINCT H.id_docente
FROM [model].[faci-inc_tb_horarios] AS H
JOIN [model].[faci-inc_tb_cursos] AS C
ON H.id_curso = C.id_curso
WHERE H.id_curso IN (
  SELECT id_curso
  FROM [model].[faci-inc_tb_cursos]
  WHERE ciclo = '2024-2');
```





SUBCONSULTAS EN WHERE



Ejemplos

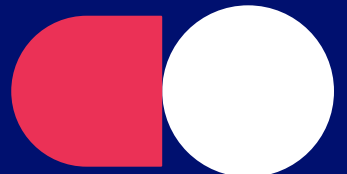
- Encuentre los nombres de los cursos cuyos docentes pertenecen al mismo departamento que el docente con código 10000025

```
SELECT DISTINCT C.nombre_curso
FROM [model].[faci-inc_tb_cursos] AS C
JOIN [model].[faci-inc_tb_horarios] AS H
  ON C.id_curso = H.id_curso
JOIN [model].[faci-inc_tb_docentes] AS D
  ON H.id_docente = D.id_docente
WHERE D.departamento_docente = (
  SELECT departamento_docente
FROM [model].[faci-inc_tb_docentes]
WHERE codigo_docente = 10000025);
```





3. Subconsulta en FROM





SUBCONSULTAS EN FROM



Concepto

- Una subconsulta en el **FROM**, también conocida como **consulta derivada**, es una consulta SQL que se inserta dentro de la cláusula **FROM** de otra consulta.
- Funciona como si fuera una **tabla temporal** y permite realizar consultas más complejas que requieren la combinación de datos procesados de manera intermedia.
- El concepto clave es que la subconsulta devuelve un conjunto de resultados **(una "tabla temporal")** que luego puede ser utilizada por la consulta externa para hacer selecciones, filtrados, o agregaciones adicionales.





SUBCONSULTAS EN FROM



Concepto

- Junto con las subconsultas del WHERE (y el operador IN), las subconsultas en el FROM son las mas comunes y poderosas de SQL, tanto así que de las subconsultas en el FROM se deriva una propiedad especial de los motores de BDD llamada “Vistas (VIEW)”





SUBCONSULTAS EN FROM



Ejemplo

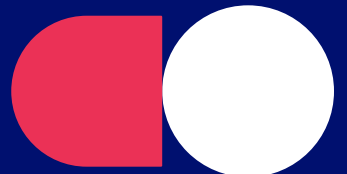
- Encuentre los nombres y apellidos de los estudiantes que están inscritos en más de un curso

```
SELECT sub.nombres_estudiante, sub.apellidos_estudiante, sub.total_cursos
FROM (
    SELECT E.nombres_estudiante
           , E.apellidos_estudiante
           , COUNT(I.id_inscripcion) AS total_cursos
    FROM [model].[faci-inc_tb_estudiantes] AS E
    JOIN [model].[faci-inc_tb_inscripciones] AS I
      ON E.id_estudiante = I.id_estudiante
    GROUP BY E.nombres_estudiante, E.apellidos_estudiante
) AS sub
WHERE sub.total_cursos > 1;
```

Observe que no
necesitamos usar la
cláusula having en
este caso ¿Por qué?



4. Subconsulta Escalar





Concepto

- La subconsulta escalar es aquella que se utiliza cuando se espera un solo valor y se aplica en el **SELECT**
- Ejemplo: Encontrar los nombres y apellidos de los estudiantes junto con el número total de cursos en los que están inscritos

```
SELECT E.nombres_estudiante  
      , E.apellidos_estudiante  
      , (SELECT COUNT(*)  
         FROM [model].[faci-inc_tb_inscripciones] AS I  
         WHERE I.id_estudiante = E.id_estudiante) AS total_cursos  
FROM [model].[faci-inc_tb_estudiantes] AS E;
```





SUBCONSULTAS ESCALAR



Concepto

- **OJO**: Error de tiempo de ejecución si la subconsulta devuelve más de una tupla de resultados
- La subconsulta escalar se utiliza cuando necesitamos agregar un valor calculado o derivado para cada fila de la consulta principal sin necesidad de realizar un JOIN complejo, especialmente cuando ese valor puede ser calculado directamente con una subconsulta que devuelve un único resultado para cada fila.





Expresión “CASE”

- El **CASE** es una expresión en SQL que permite realizar comparaciones condicionales dentro de una consulta y devolver resultados diferentes en función de esas condiciones.
- Se puede utilizar en las cláusulas SELECT, WHERE, ORDER BY, e incluso en GROUP BY y HAVING
- El 90% de las veces se encontrará en el SELECT y se aplica a los valores de una columna específica
- Es similar a una estructura condicional *IF-THEN-ELSE* en lenguajes de programación tradicionales.





Expresión “CASE”

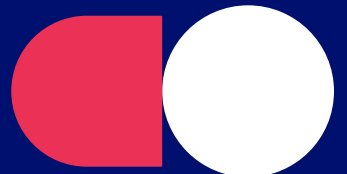
- Ejemplo: Encuentre una lista de estudiantes, y dependiendo de su género, mostrar "Masculino" o "Femenino" en lugar de los valores almacenados en la columna sexo ('M' para masculino, 'F' para femenino).

```
SELECT E.nombres_estudiante  
      , E.apellidos_estudiante  
      , CASE E.sexo  
        WHEN 'M' THEN 'Masculino'  
        WHEN 'F' THEN 'Femenino'  
        ELSE 'No especificado'  
      END AS genero  
FROM [model].[faci-inc_tb_estudiantes] AS E;
```





5. Clausula WITH





Concepto

- La cláusula **WITH**, también conocida como subconsulta de expresión común o CTE (Common Table Expression), proporciona una forma de definir una relación temporal cuya definición está disponible solo para la consulta en la que se produce la cláusula WITH.
- Funciona como una subconsulta nombrada que puede mejorar la legibilidad y organización de consultas complejas



CLAUSULA WITH



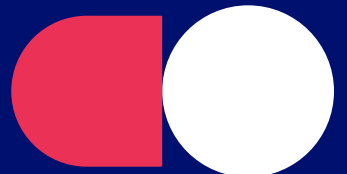
Ejemplo

- Encuentre los nombres y apellidos de los estudiantes que están inscritos en más de un curso

```
WITH sub AS (  
    SELECT E.nombres_estudiante  
           , E.apellidos_estudiante  
           , COUNT(l.id_inscripcion) AS total_cursos  
    FROM [model].[faci-inc_tb_estudiantes] AS E  
    JOIN [model].[faci-inc_tb_inscripciones] AS l  
      ON E.id_estudiante = l.id_estudiante  
    GROUP BY E.nombres_estudiante, E.apellidos_estudiante  
)  
  
SELECT sub.nombres_estudiante, sub.apellidos_estudiante, sub.total_cursos  
FROM sub  
WHERE sub.total_cursos > 1;
```



6. Operador ROLLUP





Concepto

- **ROLLUP** es una extensión de SQL que se utiliza para realizar agregaciones jerárquicas en consultas con **GROUP BY**.
- Es especialmente útil cuando se desean crear subtotales y totales generales en un conjunto de datos con múltiples niveles de agrupamiento.



Ejemplo

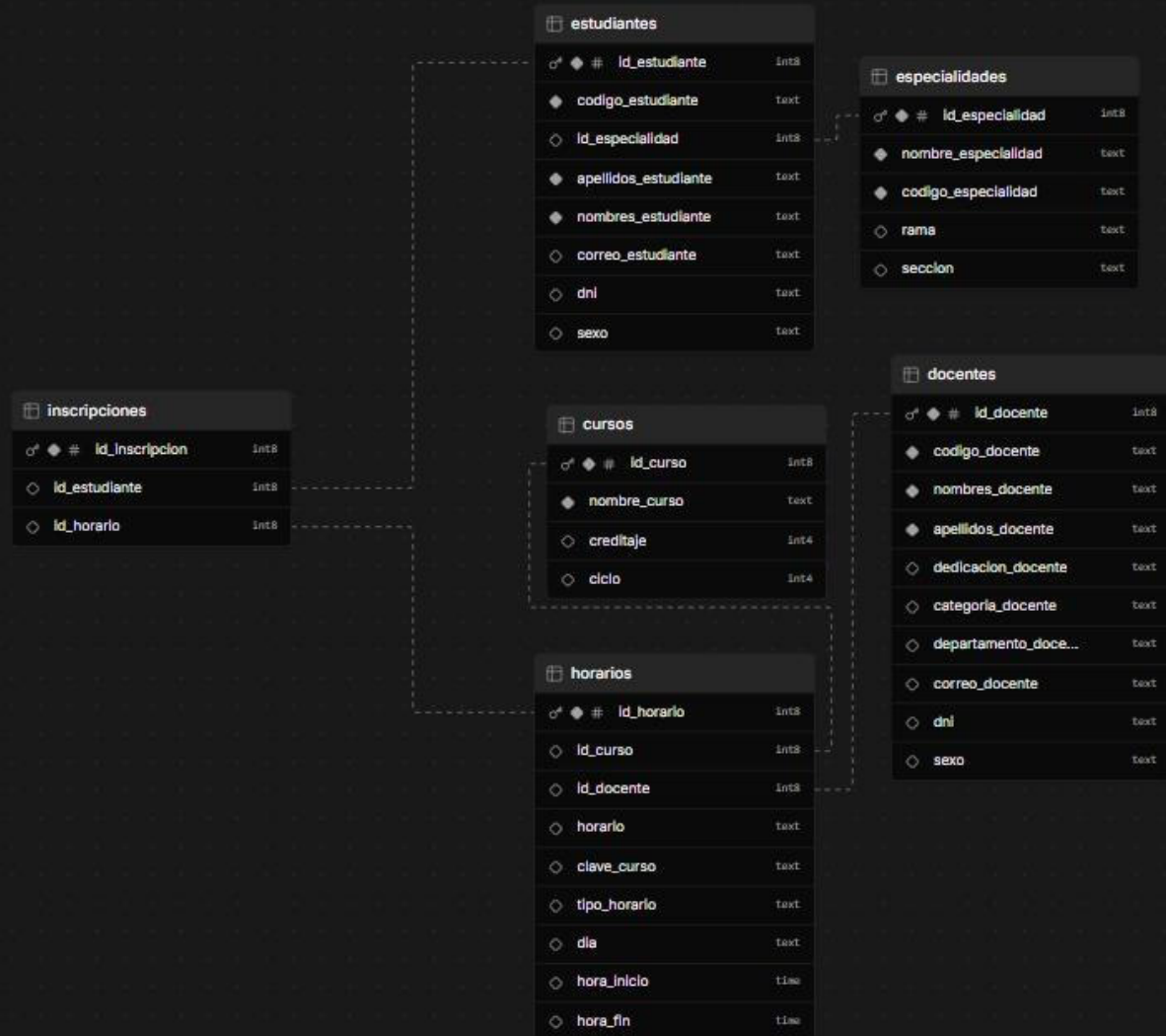
- Obtenga un resumen del número de estudiantes inscritos en los cursos por ciclo, y adicione el total por ciclo y total de totales general de estudiantes inscritos.

```
SELECT C.ciclo
      , C.nombre_curso
      , COUNT(I.id_estudiante) AS total_estudiantes
FROM [model].[faci-inc_tb_cursos] AS C
  JOIN [model].[faci-inc_tb_horarios] AS H
    ON C.id_curso = H.id_curso
  JOIN [model].[faci-inc_tb_inscripciones] AS I
    ON H.id_horario = I.id_horario
GROUP BY ROLLUP(C.ciclo, C.nombre_curso);
```



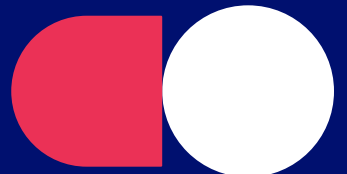
Diferencia entre **ROLLUP** y **GROUP BY** normal

- Un **GROUP BY** normal te devolverá agrupaciones por las columnas que se especifican, pero no agregará subtotales ni totales generales.
- El **ROLLUP** añade estas agregaciones automáticas.





7. Taller





TALLER



- Documento “Taller_SQL.pdf”
- NO calificable
- Preferiblemente de manera individual para retar su conocimiento
- Desarrollo esperado durante la monitoria
- Al final de la monitoria se comparten las respuestas a todos los asistentes

OJO: Todos los caminos llegan a Roma





BIBLIOGRAFÍA



Database system concepts / Abraham Silberschatz, Henry F. Korth and S. Sudarshan.
Seventh Edition. 2019. Cap 4



Gracias

