

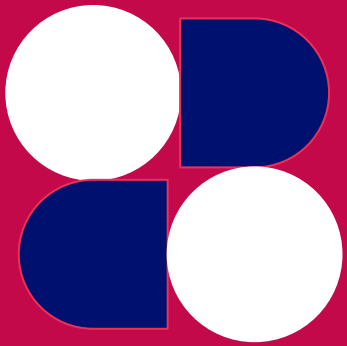


Programa PUCP DATA DRIVEN

Formación tecnológica

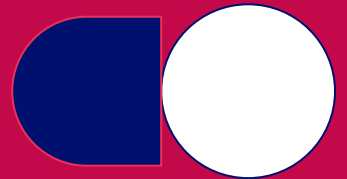
Dirección Académica de Planeamiento y Evaluación
Dirección de Gestión del Talento Humano





SESIÓN 6:

Agregaciones, Agrupamientos y JOINS





Al finalizar la sesión, lograrás:

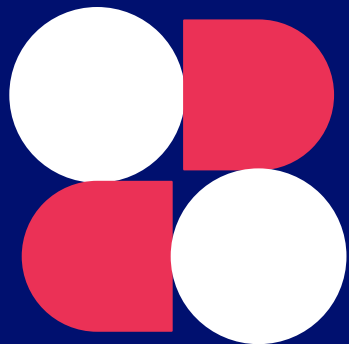
- Conocer y aplicar las principales funciones de agregación en SQL
- Conocer y aplicar los diferentes tipos de JOINS en sql



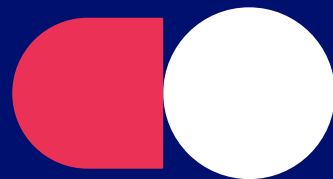
Contenido de la sesión:



- 1 Funciones de Agregación
- 2 Relaciones Conjuntas (JOIN)



1. Funciones de Agregación





FUNCIONES DE AGREGACIÓN



Concepto

- Estas funciones operan en el *multiconjunto* de valores de una columna de una relación y **devuelven un valor**. No es posible anidarlas*

avg: average value

min: minimum value

max: maximum value

sum: sum of values

count: number of values

- **OJO**: Las funciones de operaciones matemáticas aplican para valores numéricos, por lo que el CAST se vuelve fundamental. La función COUNT no tiene ese problema





FUNCIONES DE AGREGACIÓN



Ejemplos

- Encuentre el creditaje promedio de los del ciclo 2024-2

```
SELECT AVG(CAST(creditaje AS INT))  
FROM [model].[faci-inc_tb_cursos]  
WHERE [ciclo] = '2024-2';
```

- ¿Cuántos registros hay en la tabla de estudiantes?

```
SELECT COUNT(*)  
FROM [model].[faci-inc_tb_estudiantes]
```





Ejemplos

- Encuentre el la cantidad de horarios que dictó el profesor con id 1084 los días Lunes

```
SELECT COUNT(*)  
FROM [model].[faci-inc_tb_horarios]  
WHERE [dia] = 'Lunes' and [id_docente] = 1084;
```





Clausula “GROUP BY”

- Aunque no es una función de agregación, **GROUP BY** se usa junto con las funciones de agregación para agrupar los resultados por una o varias columnas.
- Suele ubicarse después del WHERE (si lo hay) y antes de ORDER BY (si lo hay)
- Cuando se desean mezclar atributos agregados con atributos no agregados en la clausula SELECT, los atributos de la cláusula SELECT fuera de las funciones agregadas deben aparecer en el **GROUP BY**
TODAS SIN EXCEPCIÓN Y DE MANERA OBLIGATORIA





Clausula “GROUP BY”

- Ejemplo: Encuentre el creditaje promedio de cada ciclo. Ordene el resultado por ciclo de manera descendente

```
SELECT AVG(CAST(creditaje AS INT)) AS PROMEDIO_CREDITAJE  
      , [ciclo]  
FROM [model].[faci-inc_tb_cursos]  
GROUP BY [ciclo]  
ORDER BY [ciclo] DESC;
```





Clausula “HAVING”

- **ES EL WHERE DE LAS AGREGACIONES**
- La cláusula **HAVING** se utiliza para filtrar los resultados después de que se han aplicado las funciones de agregación. Es similar a la cláusula WHERE, pero mientras que WHERE filtra filas antes de aplicar la agregación, HAVING filtra los resultados después de que los datos han sido agrupados y agregados.
- Si se va a utilizar **SIEMPRE** se ubica después del **GROUP BY** y antes del ORDER BY (si lo hay)





Clausula “HAVING”

- Ejemplo: Encuentre el creditaje promedio de cada ciclo, cuyo promedio sea mayor o igual a 3. Ordene el resultado por ciclo de manera descendente

```
SELECT AVG(CAST(creditaje AS INT)) AS PROMEDIO_CREDITAJE  
      ,[ciclo]  
FROM [model].[faci-inc_tb_cursos]  
GROUP BY [ciclo]  
HAVING AVG(CAST(creditaje AS INT)) >= 3  
ORDER BY [ciclo] DESC;
```





EJERCICIO

- Encuentre el promedio, la suma, el máximo y el mínimo del creditaje de los cursos por ciclo

```
SELECT AVG(CAST([creditaje] AS INT))  
      , SUM(CAST([creditaje] AS INT))  
      , MAX(CAST([creditaje] AS INT))  
      , MIN(CAST([creditaje] AS INT))  
      , [ciclo]  
FROM [model].[faci-inc_tb_cursos]  
GROUP BY [ciclo]  
ORDER BY [ciclo]
```





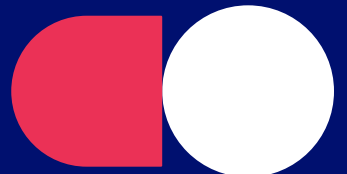
Advertencia

- Muchas veces parece intuitivo extraer la información agregada de una tabla. Sin embargo, es necesario entender cuando se desea realmente extraer la información y cuando se desea aplicar realmente como filtro para otra consulta.
- El problema está en los **GROUP BY** ya que, si la información consultada se agrupa por una columna que es de identificador único (Ej: Una PK), entonces la consulta regresará la tabla sin ningún cambio pues formalmente no se podrán crear grupos por ese atributo.
- **SOLUCIÓN:** Muchas veces se soluciona trabajando con subconsultas (Tema de la próxima clase)





2. Relaciones Conjuntas (JOIN)





JOINS



Concepto

- Las operaciones de **JOIN** toman *dos tablas* y devuelven como resultado otra tabla.
- Una operación de **JOIN** es un producto cartesiano que requiere que las tuplas en las dos tablas coincidan (bajo alguna condición). También especifica los atributos que estarán presentes en el resultado del JOIN.
- Las operaciones de **JOIN** se utilizan normalmente como expresiones de “subconsulta” en la cláusula FROM



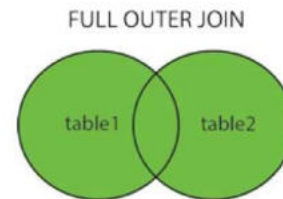
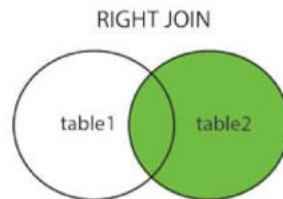
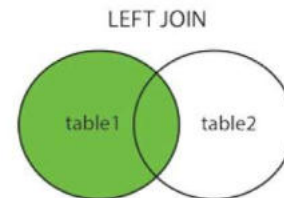
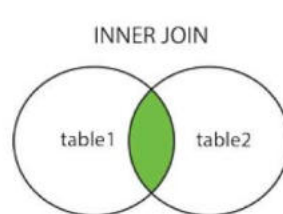


JOINS



Concepto

- Existen tres tipos de **JOIN** :
 - Natural Join
 - Inner Join
 - Outer Join



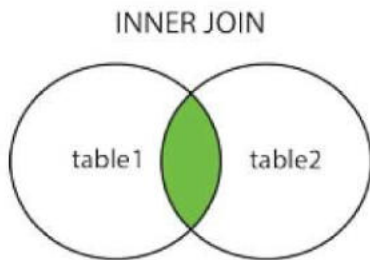


JOINS



INNER JOIN

- Un **INNER JOIN** devuelve las filas de dos tablas solo si hay una coincidencia entre las columnas especificadas en ambas tablas. Si no hay coincidencia, esas filas no aparecerán en el conjunto de resultados.
- La palabra **INNER** puede omitirse ya que, por defecto, un **JOIN** es INNER a menos que se especifique lo contrario.



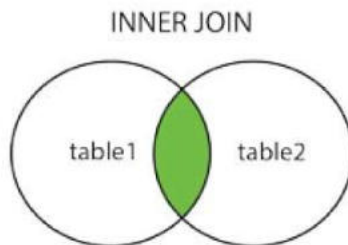


JOINS



INNER JOIN

- En cualquier tipo de **JOIN** (menos en el natural) es fundamental y recomendable especificar explícitamente las columnas de unión de entre las dos tablas. Dichas columnas suelen ser la PK y la FK respectivas.
- Para especificar las columnas de anclaje, se utiliza la clausula de condición **ON**, especialmente diseñada para los JOIN





JOINS



INNER JOIN

- La condición **ON** permite un predicado general sobre las relaciones que se unen
- Este predicado se escribe similar a un predicado de cláusula **WHERE** excepto por que se adiciona la palabra reservada **ON**





JOINS



INNER JOIN

- Ejemplo: Encuentre el nombre de los estudiantes y el nombre de la especialidad a la cual están adscritos

```
SELECT [model].[faci-inc_tb_estudiantes].[nombres_estudiante]  
    , [model].[faci-inc_tb_especialidades].[nombre_especialidad]  
FROM [model].[faci-inc_tb_estudiantes] JOIN [model].[faci-inc_tb_especialidades]  
    ON [model].[faci-inc_tb_estudiantes].[id_especialidad] = [model].[faci-inc_tb_especialidades].[id_especialidad]
```

En realidad, esto se ve feo. Clausula **AS** al rescate





JOINS



INNER JOIN

- Ejemplo: Encuentre el nombre de los estudiantes y el nombre de la especialidad a la cual están adscritos

```
SELECT est.[nombres_estudiante]  
      , esp.[nombre_especialidad]  
FROM [model].[faci-inc_tb_estudiantes] AS est  
     JOIN [model].[faci-inc_tb_especialidades] AS esp  
     ON est.[id_especialidad] = esp.[id_especialidad]
```

El 99% de las veces
se utiliza una sintaxis
similar a esta





JOINS



INNER JOIN

```
SELECT est.[nombres_estudiante]  
      , esp.[nombre_especialidad]  
FROM [model].[faci-inc_tb_estudiantes] AS est  
     JOIN [model].[faci-inc_tb_especialidades] AS esp  
     ON est.[id_especialidad] = esp.[id_especialidad]
```

	ABC nombres_estudiante	ABC nombre_especialidad
1	Kimberly	INGENIERÍA BIOMÉDICA
2	Brandy	INGENIERÍA BIOMÉDICA
3	Olivia	INGENIERÍA BIOMÉDICA
4	Nicole	INGENIERÍA BIOMÉDICA
5	Mary	INGENIERÍA BIOMÉDICA
6	Jacqueline	INGENIERÍA BIOMÉDICA
7	Monica	INGENIERÍA BIOMÉDICA
8	Joyce	INGENIERÍA BIOMÉDICA
9	Catherine	INGENIERÍA BIOMÉDICA
10	Julia	INGENIERÍA BIOMÉDICA
11	Brittany	INGENIERÍA BIOMÉDICA
12	Maria	INGENIERÍA BIOMÉDICA
13	Renee	INGENIERÍA BIOMÉDICA
14	Melinda	INGENIERÍA BIOMÉDICA
15	Shannon	INGENIERÍA BIOMÉDICA



JOINS

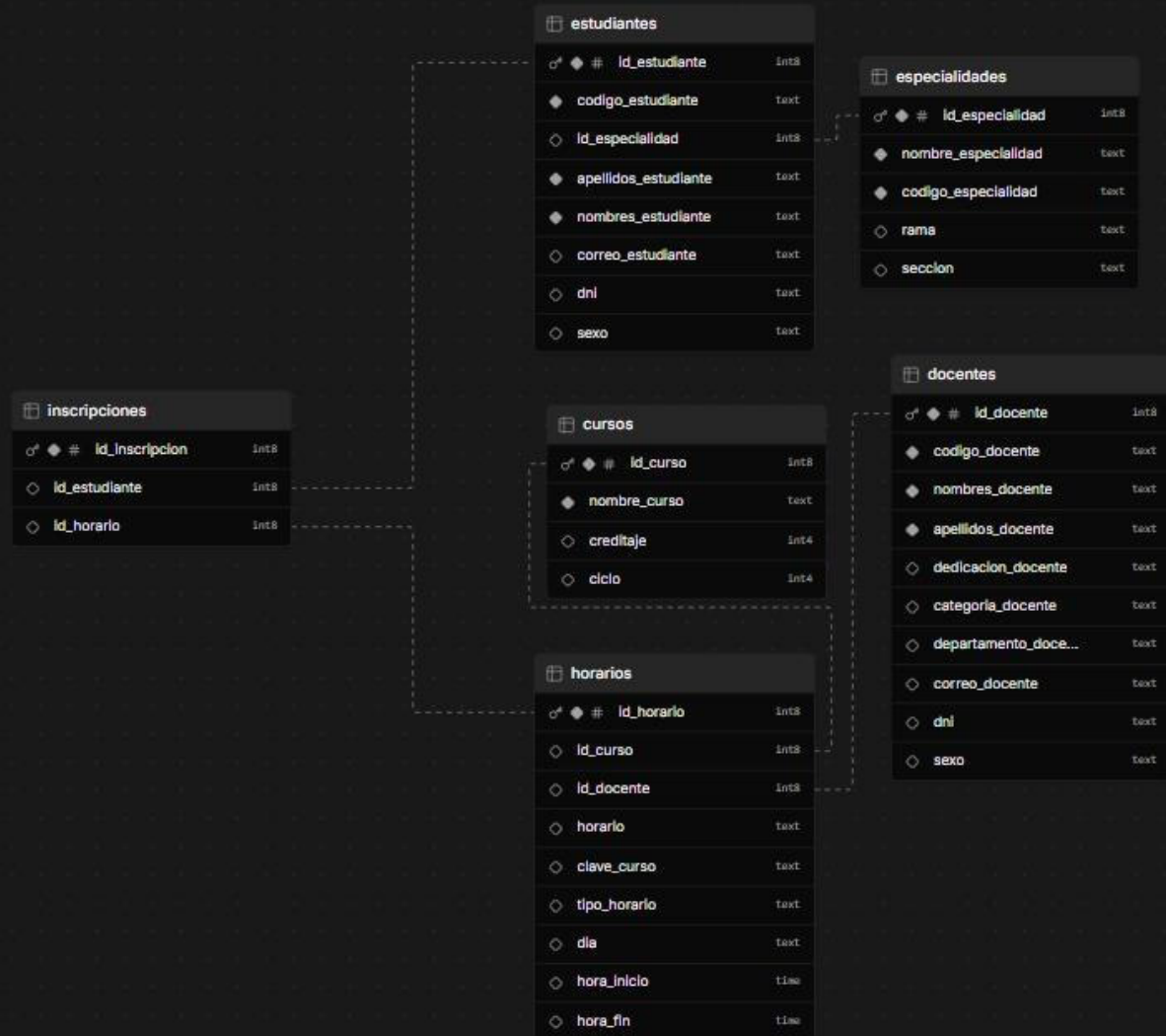


INNER JOIN

- ¿Recuerdan que el WHERE podía comparar atributos, y dijimos que tendría sentido más adelante?
- **OK!!**, una forma no tan común de hacer un JOIN es usando las clausulas simples:
- Ejemplo: Encuentre el nombre de los estudiantes y el nombre de la especialidad a la cual están adscritos

```
SELECT est.[nombres_estudiante]  
       , esp.[nombre_especialidad]  
FROM [model].[faci-inc_tb_estudiantes] AS est  
     , [model].[faci-inc_tb_especialidades] AS esp  
WHERE est.[id_especialidad] = esp.[id_especialidad]
```







JOINS



OUTER JOIN

- Los **OUTER JOIN** son una extensión de la operación de unión que evita la pérdida de información
- Los **OUTER JOIN** calculan la unión y, a continuación, agrega tuplas de una relación que no coincide con las tuplas de la otra relación en el resultado de la unión.
- La falta de punto de anclaje usualmente se debe a valores NULL o por problemas de calidad de los datos.





JOINS

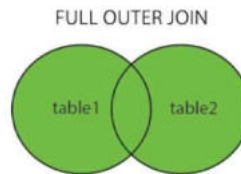
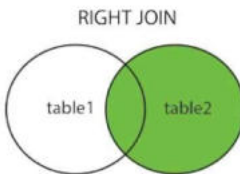
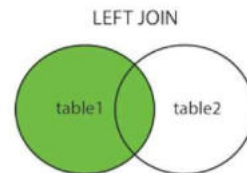


OUTER JOIN

- Existen tres formas de **OUTER JOIN**:

- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

Este es el nombre formal. Sin embargo, la palabra OUTER se puede omitir y el 99% de las veces se omite



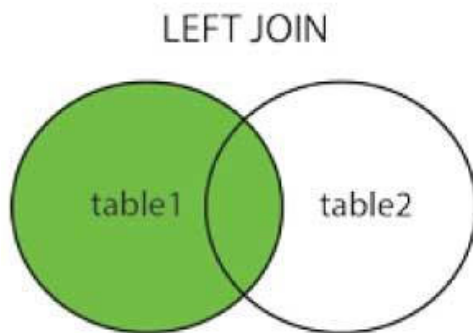


JOINS



LEFT OUTER JOIN

- Un **LEFT JOIN** garantiza que todas las filas de la tabla de la izquierda estén presentes en los resultados, incluso si no hay una coincidencia en la tabla de la derecha. Si una fila de la tabla izquierda no encuentra coincidencias en la tabla derecha, las columnas correspondientes a la tabla derecha se rellenan con NULL.





JOINS

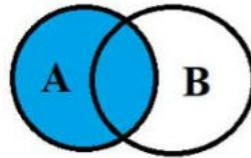


LEFT OUTER JOIN

A

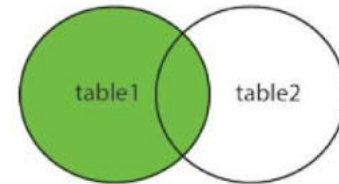
Student ID	Name
1001	A
1002	B
1003	C
1004	D

+



Student ID	Name	Department
1001	A	NULL
1002	B	NULL
1003	C	NULL
1004	D	Mathematics

LEFT JOIN



B

Student ID	Department
1004	Mathematics
1005	Mathematics
1006	History
1007	Physics
1008	Computer Science



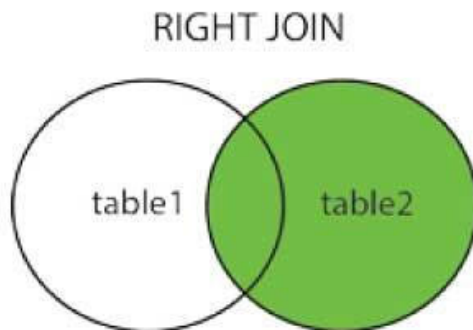


JOINS



RIGHT OUTER JOIN

- Un **RIGHT JOIN** garantiza que todas las filas de la tabla de la derecha aparezcan en el conjunto de resultados, incluso si no hay una coincidencia en la tabla de la izquierda. Si una fila de la tabla derecha no tiene coincidencias en la tabla izquierda, las columnas correspondientes a la tabla izquierda se rellenan con NULL.



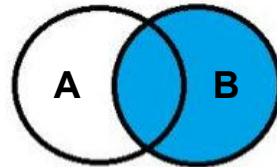


JOINS



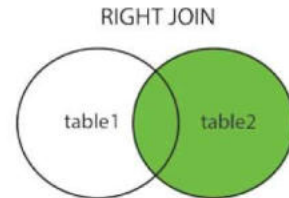
RIGHT OUTER JOIN

A	Student ID	Name
	1001	A
	1002	B
	1003	C
	1004	D



B	Student ID	Department
	1004	Mathematics
	1005	Mathematics
	1006	History
	1007	Physics
	1008	Computer Science

Student ID	Name	Department
1004	D	Mathematics
1005	NULL	Mathematics
1006	NULL	History
1007	NULL	Physics
1008	NULL	Computer Science



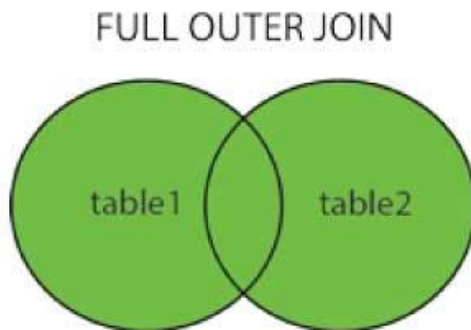


JOINS



FULL OUTER JOIN

- El **FULL OUTER JOIN** es una combinación de los resultados de un **LEFT JOIN** y un **RIGHT JOIN**, incluyendo todas las filas de ambas tablas, independientemente de si hay coincidencia o no.

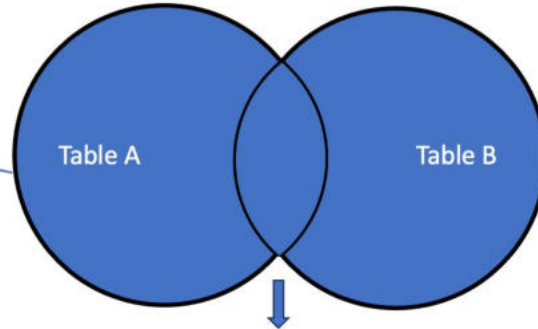




JOINS

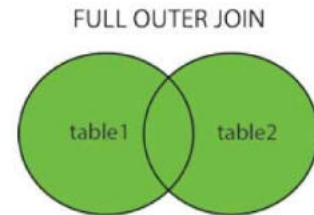
FULL OUTER JOIN

Order ID	Customer ID	Cust. Name
1	A1	Le
2	B4	Timon
3	C6	Pumbaa
1	D2	Tom
5	A8	Jerry



Order ID	Product
1	Rice
2	Tomato
3	Cooking oil
4	Sugar

Order ID	Customer ID	Cust. Name	Product
1	A1	Le	Rice
2	B4	Timon	Tomato
3	C6	Pumbaa	Cooking oil
1	D2	Tom	Rice
5	A8	Jerry	
4			Sugar



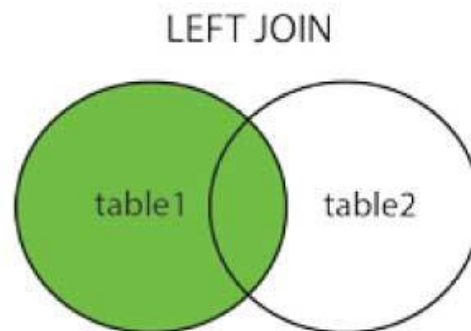
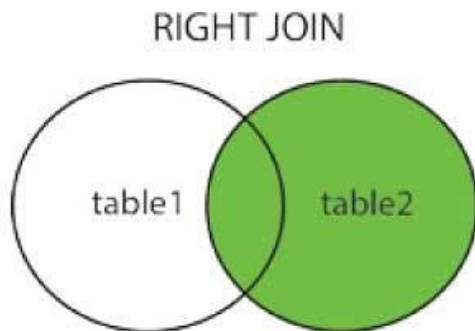


JOINS



LEFT JOIN Y RIGHT JOIN

- ¿Quién determina cual tabla es DERECHA y cual es IZQUIERDA?
- RTA: El orden en que se especifiquen en el JOIN





JOINS



Ejemplo LEFT JOIN

- Encuentre los nombres y apellidos de los estudiantes con su respectivo id de inscripción e id de horario. Incluya estudiantes que no tengan asociadas inscripciones.

```
SELECT est.nombres_estudiante  
      , est.apellidos_estudiante  
      , ins.id_inscripcion  
      , ins.id_horario  
FROM [model].[faci-inc_tb_estudiantes] AS est  
  LEFT JOIN [model].[faci-inc_tb_inscripciones] AS ins  
    ON est.id_estudiante = ins.id_estudiante;
```





JOINS



Ejercicio LEFT JOIN

- Encuentre los id y días de los horarios disponibles con la información de los nombres y apellidos de los docentes que los dictan. Incluya los horarios que no tienen docentes asignados.

```
SELECT hor.id_horario
       , hor.dia
       , doc.nombres_docente
       , doc.apellidos_docente
FROM [model].[faci-inc_tb_horarios] AS hor
LEFT JOIN [model].[faci-inc_tb_docentes] AS doc
ON hor.id_docente = doc.id_docente;
```





JOINS



Ejemplo RIGHT JOIN

- Encuentre los nombres y apellidos de los docentes con el día, la hora de inicio y la hora final de los horarios que dictan. Incluya docentes que no tienen horarios asignados.

```
SELECT doc.nombres_docente  
      , doc.apellidos_docente  
      , hor.dia  
      , hor.hora_inicio  
      , hor.hora_fin  
FROM [model].[faci-inc_tb_horarios] AS hor  
  RIGHT JOIN [model].[faci-inc_tb_docentes] AS doc  
  ON hor.id_docente = doc.id_docente;
```





JOINS



Ejercicio RIGHT JOIN

- Encuentre los nombres de todos los cursos y su día, hora de inicio y hora de fin en que se dicta el curso. Incluya los cursos que no tienen inscripciones en ningún horario

```
SELECT cur.nombre_curso  
      , hor.dia  
      , hor.hora_inicio  
      , hor.hora_fin  
FROM [model].[faci-inc_tb_horarios] AS hor  
  RIGHT JOIN [model].[faci-inc_tb_cursos] AS cur  
    ON hor.id_curso = cur.id_curso;
```





JOINS



Ejemplo FULL OUTER JOIN

- Encuentre todos los nombres y apellidos de los estudiantes y sus respectivos id de inscripción y de horario, incluyendo estudiantes que no tengan inscripciones y también inscripciones sin estudiantes asociados.

```
SELECT est.nombres_estudiante  
       , est.apellidos_estudiante  
       , ins.id_inscripcion  
       , ins.id_horario  
FROM [model].[faci-inc_tb_estudiantes] AS est  
   FULL OUTER JOIN [model].[faci-inc_tb_inscripciones] AS ins  
   ON est.id_estudiante = ins.id_estudiante;
```





JOINS



Ejercicio FULL OUTER JOIN

- Encuentre todos los id de los horarios, sus días y los nombres y apellidos de los docentes asociados, incluyendo horarios sin docentes asignados y docentes que no tienen ningún horario asignado

```
SELECT hor.id_horario
       , hor.dia
       , doc.nombres_docente
       , doc.apellidos_docente
FROM [model].[faci-inc_tb_horarios] AS hor
FULL OUTER JOIN [model].[faci-inc_tb_docentes] AS doc
ON hor.id_docente = doc.id_docente;
```





JOINS

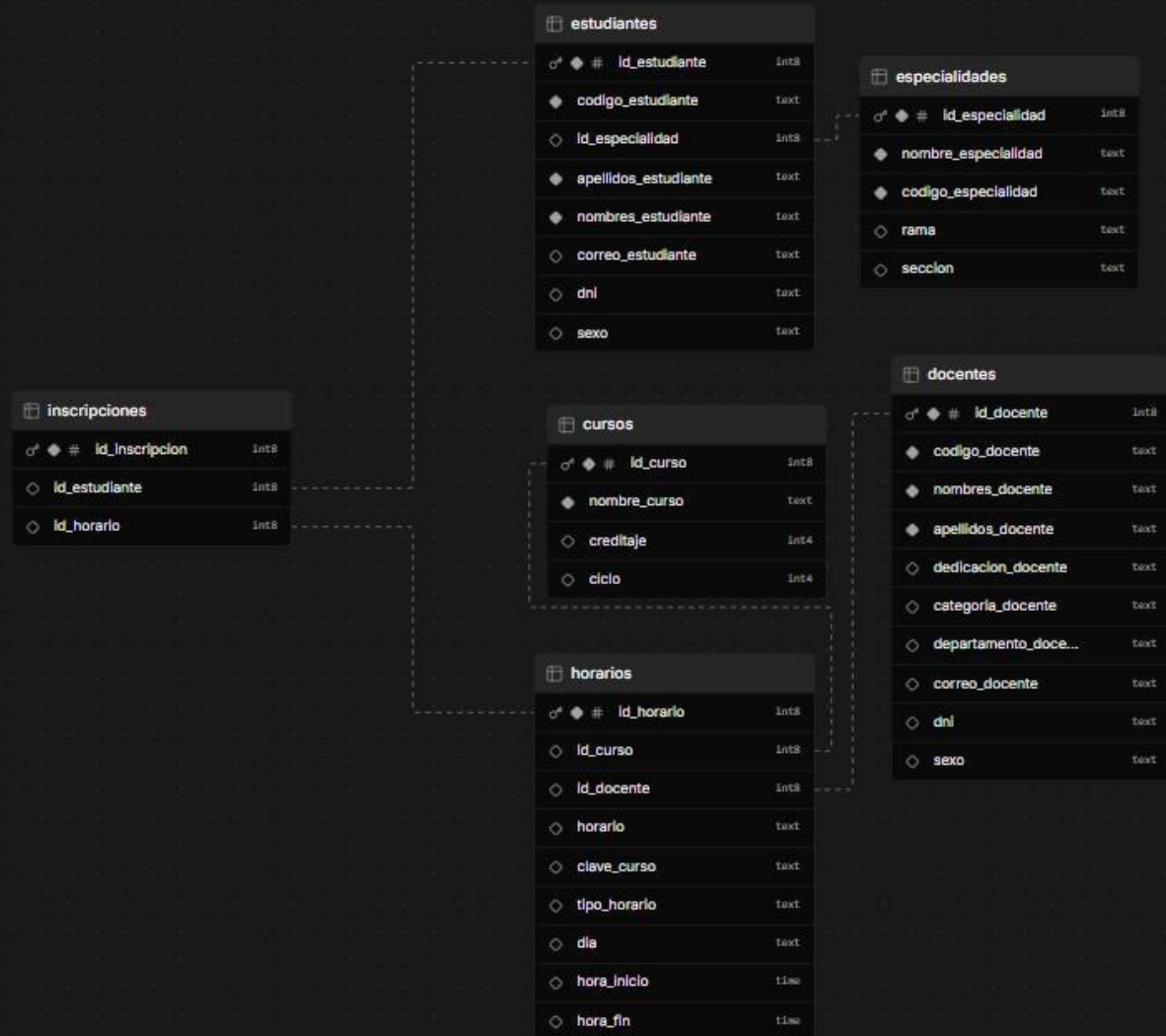


Ejercicio

- Obtenga una lista con los nombres y apellidos de los estudiantes, los nombres de los cursos en los que están inscritos, el día y las horas de inicio y fin de sus cursos. Solo tenga en cuenta los estudiantes con horario.

```
SELECT est.nombres_estudiante, est.apellidos_estudiante, cur.nombre_curso, hor.dia
       , hor.hora_inicio, hor.hora_fin
FROM [model].[faci-inc_tb_inscripciones] AS ins
  JOIN [model].[faci-inc_tb_estudiantes] AS est
    ON ins.id_estudiante = est.id_estudiante
  JOIN [model].[faci-inc_tb_horarios] AS hor
    ON ins.id_horario = hor.id_horario
  JOIN [model].[faci-inc_tb_cursos] AS cur
    ON hor.id_curso = cur.id_curso;
```







BIBLIOGRAFÍA



Database system concepts / Abraham Silberschatz, Henry F. Korth and S. Sudarshan.
Seventh Edition. 2019. Cap 3 y 4



Gracias

