Bases de Datos I

Lenguaje de Consulta de Datos

Lic. Carlos León González
Dra.C. Lucina García Hernández

Facultad de Matemática y Computación
Universidad de La Habana

21 de noviembre de 2023

¿Qué es lo mínimo a implementar en una BD?

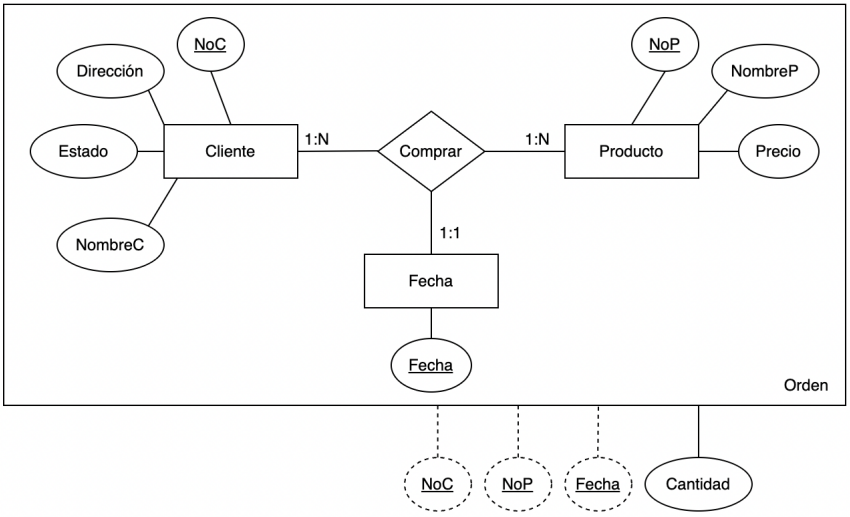
¿Qué es lo mínimo a implementar en una BD?

CRUD

¿Qué es lo mínimo a implementar en una BD?

C reate (Insert)
R ead (Select)
U pdate (Update)
D elete (Delete)

Definiendo las tablas



Iniciando con la sintaxis de la cláusula SELECT

```
01 |  SELECT * | <column_1>, <column_2>, ...
02 |  FROM <table_name>;
```

Iniciando con la sintaxis de la cláusula SELECT

```
01 |  SELECT * | <column_1>, <column_2>, ...
02 |  FROM <table_name>;
```

SELECT: Define columnas a tomar

FROM: Define la tabla a consultar

La primera consulta

Sintaxis:

```
01 |  SELECT * | <column_1>, <column_2>, ...
02 |  FROM <table_name>;
```

Construya una consulta para ver la información del **cliente**.

La primera consulta

Sintaxis:

```
01 |  SELECT * | <column_1>, <column_2>, ...
02 |  FROM <table_name>;
```

Construya una consulta para ver la información del **cliente**.

```
01 |  SELECT *
02 |  FROM cliente;
```

```
01 |  SELECT NoC, NombreC, Dirección,
       Estado
02 |  FROM cliente;
```

La primera consulta

Sintaxis:

```
01 |  SELECT * | <column_1>, <column_2>, ...
02 |  FROM <table_name>;
```

Construya una consulta para ver la información del **cliente**.

```
01 |  SELECT *
02 |  FROM cliente;
```

```
01 |  SELECT NoC, NombreC, Dirección,
       Estado
02 |  FROM cliente;
```

Dos consultas que generan el mismo resutado, pero una aprovecha el “azúcar” sintáctica de SQL.

La primera consulta

Sintaxis:

```
01 |  SELECT * | <column_1>, <column_2>, ...
02 |  FROM <table_name>;
```

Construya una consulta para ver la información del **cliente**.

```
01 |  SELECT *
02 |  FROM cliente;
```

```
01 |  SELECT NoC, NombreC, Dirección,
       Estado
02 |  FROM cliente;
```

Dos consultas que generan el mismo resutado, pero una aprovecha el “azúcar” sintáctica de SQL.

¿Cómo elegir las filas a mostrar?

Seleccionando filas

Sintaxis:

```
01 |  SELECT * | <column_1>, <column_2>, ...
02 |  FROM <table_name>
03 |  WHERE <constraint_expresion>;
```

Seleccionando filas

Sintaxis:

```
01 |  SELECT * | <column_1>, <column_2>, ...
02 |  FROM <table_name>
03 |  WHERE <constraint_expresion>;
```

Posibles operadores a usar:

- Lógicos: *AND*, *OR*, *NOT*
- Relacionales: $<$, $<=$, $>=$, $>$, $=$, $<>$

Seleccionando filas

Sintaxis:

```
01 |  SELECT * | <column_1>, <column_2>, ...
02 |  FROM <table_name>
03 |  WHERE <constraint_expresion>;
```

Posibles operadores a usar:

- Lógicos: *AND*, *OR*, *NOT*
- Relacionales: $<$, $<=$, $>=$, $>$, $=$, $<>$

Construya una consulta para obtener todos los productos cuyo precio sea menor o igual a 10, o mayor que 2000.

Seleccionando filas

Sintaxis:

```
01 |  SELECT * | <column_1>, <column_2>, ...
02 |  FROM <table_name>
03 |  WHERE <constraint_expresion>;
```

Posibles operadores a usar:

- Lógicos: *AND*, *OR*, *NOT*
- Relacionales: $<$, $<=$, $>=$, $>$, $=$, $<>$

Construya una consulta para obtener todos los productos cuyo precio sea menor o igual a 10, o mayor que 2000.

```
01 |  SELECT NombreP, Precio
02 |  FROM producto
03 |  WHERE Precio <= 10 OR Precio > 2000;
```

Seleccionando filas

Sintaxis:

```
01 |  SELECT expresion
02 |  FROM table_name
03 |  WHERE constraint_expresion;
```

Posibles operadores a usar:

- Especiales: *IN*, *LIKE*, *BETWEEN*, *IS NULL*

Seleccionando filas

Sintaxis:

```
01 |  SELECT expresion
02 |  FROM table_name
03 |  WHERE constraint_expresion;
```

Posibles operadores a usar:

- Especiales: *IN*, *LIKE*, *BETWEEN*, *IS NULL*

  Construya una consulta para obtener todos los clientes que pertenezcan a “Ohio”, “Nevada” o “Misisipi”.

Seleccionando filas

Sintaxis:

```
01 |  SELECT expresion
02 |  FROM table_name
03 |  WHERE constraint_expresion;
```

Posibles operadores a usar:

- Especiales: *IN*, *LIKE*, *BETWEEN*, *IS NULL*

Construya una consulta para obtener todos los clientes que pertenezcan a “Ohio”, “Nevada” o “Misisipi”.

```
01 |  SELECT NombreC
02 |  FROM cliente
03 |  WHERE Estado IN ('Ohio', 'Nevada',
          'Misisipi');
```

```
01 |  SELECT NombreC
02 |  FROM cliente
03 |  WHERE Estado = 'Ohio' OR  Estado =
          'Nevada' OR Estado =  '
         Misisipi';
```

Seleccionando filas

Sintaxis:

```
01 | SELECT <expresion>
02 | FROM <table_name>
03 | WHERE <constraint_expresion>;
```

Posibles operadores a usar:

- Especiales: *IN*, *LIKE*, *BETWEEN*, *IS NULL*

Seleccionando filas

Sintaxis:

```
01 |  SELECT <expresion>
02 |  FROM <table_name>
03 |  WHERE <constraint_expresion>;
```

Posibles operadores a usar:

- Especiales: *IN*, *LIKE*, *BETWEEN*, *IS NULL*

  Construya una consulta para obtener todos los clientes cuyo nombre comience con la letra "`I`" y culmine con al menos cuatro letras, donde las dos últimas tienen que ser "`er`". La columna de la tabla resultante tiene que llamarse "Cliente Especial I-er".

Seleccionando filas

Sintaxis:

```
01 |  SELECT <expresion>
02 |  FROM <table_name>
03 |  WHERE <constraint_expresion>;
```

Posibles operadores a usar:

- Especiales: *IN*, *LIKE*, *BETWEEN*, *IS NULL*

Construya una consulta para obtener todos los clientes cuyo nombre comience con la letra "I" y culmine con al menos cuatro letras, donde las dos últimas tienen que ser "er". La columna de la tabla resultante tiene que llamarse "Cliente Especial I-er".

```
01 |  SELECT NombreC AS `Cliente Especial I-er`
02 |  FROM cliente
03 |  WHERE NombreC LIKE 'I%_er';
```

Desglosando el comando LIKE

```
01 |  SELECT NombreC AS `Cliente Especial I-er`
02 |  FROM cliente
03 |  WHERE NombreC LIKE 'I%_er';
```

El comando AS brinda alias a columnas o tablas.

Desglosando el comando LIKE

```
01 |  SELECT NombreC AS `Cliente Especial I-er`
02 |  FROM cliente
03 |  WHERE NombreC LIKE 'I%_er';
```

El comando AS brinda alias a columnas o tablas.

Comodines de LIKE:

- % Representa cero, uno o múltiples caracteres
- _ Representa un solo caracter

Desglosando el comando LIKE

```
01 |  SELECT NombreC AS `Cliente Especial I-er`
02 |  FROM cliente
03 |  WHERE NombreC LIKE 'I%_er';
```

El comando AS brinda alias a columnas o tablas.

Comodines de LIKE:

- % Representa cero, uno o múltiples caracteres
- _ Representa un solo caracter

Existen otros comodines que permiten generalizar el comando y utilizarlo con expresiones regulares.

Desglosando el comando LIKE

```
01 |  SELECT NombreC AS `Cliente Especial I-er`
02 |  FROM cliente
03 |  WHERE NombreC LIKE 'I%_er';
```

El comando AS brinda alias a columnas o tablas.

Comodines de LIKE:

- % Representa cero, uno o múltiples caracteres
- _ Representa un solo caracter

Existen otros comodines que permiten generalizar el comando y utilizarlo con expresiones regulares.

¿Cómo ordenar la relación resultante?

Ordenando los resultados

Construya una consulta para obtener todos los clientes, su dirección y su estado, pero ordenados de forma descendente por la dirección y el estado al que pertenecen.

Ordenando los resultados

Construya una consulta para obtener todos los clientes, su dirección y su estado, pero ordenados de forma descendente por la dirección y el estado al que pertenecen.

Sintaxis:

```
01 |  SELECT <expresion>
02 |  FROM <table_name>
03 |  ORDER BY <field> [ASC|DESC], ...;
```

Ordenando los resultados

Construya una consulta para obtener todos los clientes, su dirección y su estado, pero ordenados de forma descendente por la dirección y el estado al que pertenecen.

Sintaxis:

```
01 |  SELECT <expresion>
02 |  FROM <table_name>
03 |  ORDER BY <field> [ASC|DESC], ...;
```

Solución:

```
01 |  SELECT NombreC, Dirección, Estado
02 |  FROM cliente
03 |  ORDER BY Estado ASC, Dirección
         DESC;
```

```
01 |  SELECT NombreC, Dirección, Estado
02 |  FROM cliente
03 |  ORDER BY Estado, Dirección DESC;
```

Ordenando los resultados

Construya una consulta para obtener todos los clientes, su dirección y su estado, pero ordenados de forma descendente por la dirección y el estado al que pertenecen.

Sintaxis:

```
01 |  SELECT <expresion>
02 |  FROM <table_name>
03 |  ORDER BY <field> [ASC|DESC], ...;
```

Solución:

```
01 |  SELECT NombreC, Dirección, Estado
02 |  FROM cliente
03 |  ORDER BY Estado ASC, Dirección
         DESC;
```

```
01 |  SELECT NombreC, Dirección, Estado
02 |  FROM cliente
03 |  ORDER BY Estado, Dirección DESC;
```

¿Cómo limitar el número de filas mostradas?

Limitando la cantidad de filas a mostrar

Sintaxis:

```
01 |  SELECT * | <column_1>, <column_2>, ...
02 |  FROM <table_name>
03 |  LIMIT <number>;
```

Limitando la cantidad de filas a mostrar

Sintaxis:

```
01 |  SELECT * | <column_1>, <column_2>, ...
02 |  FROM <table_name>
03 |  LIMIT <number>;
```

Con la siguiente consulta, se muestran solo 10 filas

```
01 |  SELECT *
02 |  FROM cliente
03 |  LIMIT 10;
```

Limitando la cantidad de filas a mostrar

Sintaxis:

```
01 |  SELECT * | <column_1>, <column_2>, ...
02 |  FROM <table_name>
03 |  LIMIT <number>;
```

Con la siguiente consulta, se muestran solo 10 filas

```
01 |  SELECT *
02 |  FROM cliente
03 |  LIMIT 10;
```

¿Qué pasa si se quiere resumir la información?

Agrupando los resultados

Construya una consulta que muestre la cantidad de clientes por estado.

Agrupando los resultados

Construya una consulta que muestre la cantidad de clientes por estado.

Sintaxis:

```
01 |  SELECT <expresion>
02 |  FROM <table_name>
03 |  GROUP BY <field_1> [, <field_2>, ...] ;
```

Agrupando los resultados

Construya una consulta que muestre la cantidad de clientes por estado.

Sintaxis:

```
01 |  SELECT <expresion>
02 |  FROM <table_name>
03 |  GROUP BY <field_1> [, <field_2>, ...] ;
```

Solución:

```
01 |  SELECT Estado, COUNT(NoC) AS Cantidad
02 |  FROM cliente
03 |  GROUP BY Estado;
```

Funciones útiles para resumir

- COUNT() : Cuenta la cantidad de registros
- MAX() : Determina el valor máximo para una atributo
- MIN() : Determina el valor mínimo para una atributo
- SUM() : Suma los valores de un atributo
- AVG() : Calcula el promedio de los valores de un atributo

Funciones útiles para resumir

- COUNT() : Cuenta la cantidad de registros
- MAX() : Determina el valor máximo para una atributo
- MIN() : Determina el valor mínimo para una atributo
- SUM() : Suma los valores de un atributo
- AVG() : Calcula el promedio de los valores de un atributo

Las funciones a utilizar por el uso de la cláusula GROUP BY requieren un conjunto de valores dado por el nombre del atributo. En el caso de **COUNT**, no es necesario la definición de un atributo, basta con el uso de “*”.

Funciones útiles para resumir

- COUNT() : Cuenta la cantidad de registros
- MAX() : Determina el valor máximo para una atributo
- MIN() : Determina el valor mínimo para una atributo
- SUM() : Suma los valores de un atributo
- AVG() : Calcula el promedio de los valores de un atributo

Las funciones a utilizar por el uso de la cláusula GROUP BY requieren un conjunto de valores dado por el nombre del atributo. En el caso de **COUNT**, no es necesario la definición de un atributo, basta con el uso de “*”.

¿Y si se quiere seleccionar algunas tuplas, luego de agrupar?

Añadiendo condición al agrupar

Sintaxis:

```
01 |  SELECT <expresion>
02 |  FROM <table_name>
03 |  GROUP BY <field_1> [, <field_2>, ...]
04 |  HAVING <constraint_expresion>;
```

Añadiendo condición al agrupar

Sintaxis:

```
01 |  SELECT <expresion>
02 |  FROM <table_name>
03 |  GROUP BY <field_1> [, <field_2>, ...]
04 |  HAVING <constraint_expresion>;
```

Construya una consulta que muestre la cantidad de clientes por estado tal que su total sea menor que 5.

Añadiendo condición al agrupar

Sintaxis:

```
01 |  SELECT <expresion>
02 |  FROM <table_name>
03 |  GROUP BY <field_1> [, <field_2>, ...]
04 |  HAVING <constraint_expresion>;
```

Construya una consulta que muestre la cantidad de clientes por estado tal que su total sea menor que 5.

```
01 |  al
```

Añadiendo condición al agrupar

Sintaxis:

```
01 |  SELECT <expresion>
02 |  FROM <table_name>
03 |  GROUP BY <field_1> [, <field_2>, ...]
04 |  HAVING <constraint_expresion>;
```

Construya una consulta que muestre la cantidad de clientes por estado tal que su total sea menor que 5.

```
01 |  al
```

Y si se quieren expresar alternativas, ¿qué cláusula me lo facilita?

Utilizando alternativas

La clásula CASE tiene un comportamiento similar al conocido *if-then-else*. En caso de no tener definido “else” y no cumplirse ninguna condición, el retorno será **NULL**.

Sintaxis:

```
01 |  CASE
02 |    WHEN condition_1 THEN result_1
03 |    [WHEN condition_2 THEN result_2,
04 |    ...
05 |    ELSE result]
06 |  END;
```

Utilizando alternativas

La clásula CASE tiene un comportamiento similar al conocido *if-then-else*. En caso de no tener definido “else” y no cumplirse ninguna condición, el retorno será **NULL**.

Sintaxis:

```
01 | CASE
02 |   WHEN condition_1 THEN result_1
03 |   [WHEN condition_2 THEN result_2,
04 |   ...
05 |   ELSE result]
06 | END;
```

Construya una consulta para conocer si el precio de un producto es barato, normal, caro o muy caro.

Etiqueta	Rango de precio
barato	[0, 100)
normal	[100, 1000)

Etiqueta	Rango de precio
caro	[1000, 2000)
muy caro	e.o.c

Utilizando *if-then-else*

Construya una consulta para conocer si el precio de un producto es barato, normal, caro o muy caro.

Etiqueta	Rango de precio
barato	[0, 100)
normal	[100, 1000)

Etiqueta	Rango de precio
caro	[1000, 2000)
muy caro	e.o.c

Utilizando *if-then-else*

Construya una consulta para conocer si el precio de un producto es barato, normal, caro o muy caro.

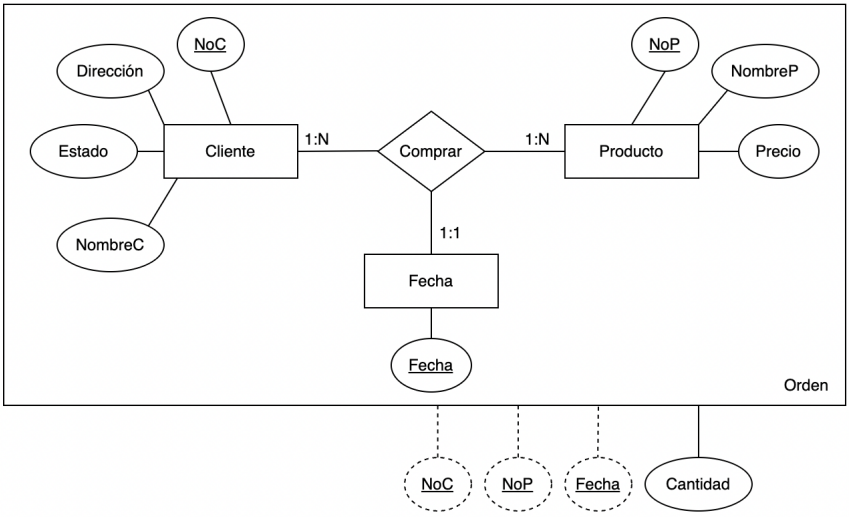
Etiqueta	Rango de precio
barato	[0, 100)
normal	[100, 1000)

Etiqueta	Rango de precio
caro	[1000, 2000)
muy caro	e.o.c

```
01 | SELECT
02 |   NombreP as Producto,
03 |   CASE
04 |     WHEN Precio >= 0 AND Precio < 100 THEN 'barato'
05 |     WHEN Precio >= 100 AND Precio < 1000 THEN 'normal'
06 |     WHEN Precio >= 1000 AND Precio < 2000 THEN 'caro'
07 |     ELSE 'muy caro'
08 |   END AS `Tipo de precio`
09 | FROM producto;
```


THE SQL
IS ON FIRE

Recordando las tablas



Vinculando tablas

La cláusula JOIN permite relacionar varias tablas, usando las *llaves foráneas* o bien columnas de referencias.

Vinculando tablas

La cláusula JOIN permite relacionar varias tablas, usando las *llaves foráneas* o bien columnas de referencias.

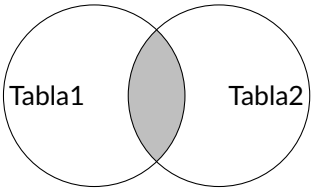
Sintaxis:

```
01 |  SELECT <expresion>
02 |  FROM <table_name_1>
03 |    JOIN <table_name_2>
04 |    ON <table_name_1>.<field_1> = <table_name_2>.<field_2>;
```

Tipos de vínculos

INNER JOIN

Obtiene todos los registros, siempre y cuando exista una coincidencia de los valores de las columnas definidas de ambas tablas.



Tipos de vínculos

INNER JOIN

Construya una consulta que muestre el cliente y la fecha en que efectuó alguna compra.

Tipos de vínculos

INNER JOIN

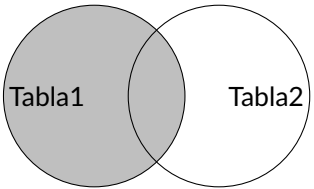
Construya una consulta que muestre el cliente y la fecha en que efectuó alguna compra.

```
01 |  SELECT
02 |    NombreC as Cliente,
03 |    Fecha
04 |  FROM orden
05 |    JOIN cliente
06 |      ON orden.NoC = cliente.NoC;
```

Tipos de vínculos

LEFT (OUTER) JOIN

Obtiene todos los registros de la tabla que aparece en el FROM, conocida como tabla izquierda (**Tabla1**). Los registros de la tabla posterior al JOIN, conocida como tabla derecha (**Tabla2**), se añaden si existe alguna coincidencia de los valores de las columnas definidas. En caso contrario, se mostrará NULL.



Tipos de vínculos

LEFT (OUTER) JOIN

Construya una consulta que muestre el cliente y la fecha en que efectuó alguna compra. Deben de aparecer también aquellos clientes que no hayan efectuado alguna compra.

Tipos de vínculos

LEFT (OUTER) JOIN

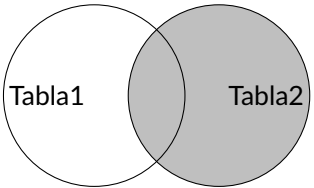
Construya una consulta que muestre el cliente y la fecha en que efectuó alguna compra. Deben de aparecer también aquellos clientes que no hayan efectuado alguna compra.

```
01 |  SELECT
02 |    NombreC AS Cliente,
03 |    Fecha
04 |  From cliente
05 |    LEFT JOIN orden
06 |      ON orden.NoC = cliente.NoC;
```

Tipos de vínculos

RIGHT (OUTER) JOIN

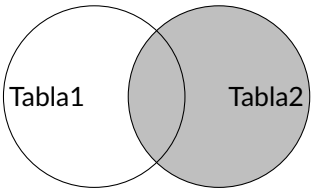
Obtiene todos los registros de la tabla posterior al JOIN, conocida como tabla derecha (**Tabla2**). Los registros de la tabla que aparece en el FROM, conocida como tabla izquierda (**Tabla1**), se añaden si existe alguna coincidencia de los valores de las columnas definidas. En caso contrario, se mostrará NULL.



Tipos de vínculos

RIGHT (OUTER) JOIN

Obtiene todos los registros de la tabla posterior al JOIN, conocida como tabla derecha (**Tabla2**). Los registros de la tabla que aparece en el FROM, conocida como tabla izquierda (**Tabla1**), se añaden si existe alguna coincidencia de los valores de las columnas definidas. En caso contrario, se mostrará NULL.

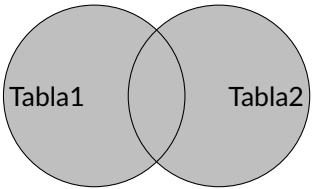


¿Qué devuelve la consulta anterior si se sustituye por el vínculo RIGHT JOIN?

Tipos de vínculos

FULL (OUTER) JOIN

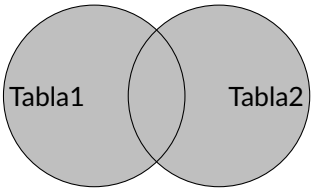
Combina los resultados de los *joins LEFT* y *RIGHT*. Alternativamente, aparecerá NULL cada registro de la tabla, cuando no haya coincidencia.



Tipos de vínculos

FULL (OUTER) JOIN

Combina los resultados de los *joins LEFT* y *RIGHT*. Alternativamente, aparecerá NULL cada registro de la tabla, cuando no haya coincidencia.



¿Qué devuelve la consulta anterior si se sustituye por el vínculo FULL JOIN?

¿Qué son las subconsultas?

- En ocasiones, se necesita definir consultas anidadas, o sea, consultas dentro de otra consulta. Esto se conoce como **subconsultas**.
- Las subconsultas pueden definirse en las cláusulas: SELECT, FROM, WHERE, HAVING.
- Existen reglas al utilizar subconsultas.

Trabajando con subconsultas

Construya una consulta para obtener los productos más baratos, teniendo en cuenta que:

Etiqueta	Rango de precio
barato	[0, 100)
normal	[100, 1000)

Etiqueta	Rango de precio
caro	[1000, 2000)
muy caro	e.o.c

Trabajando con subconsultas

Construya una consulta para obtener los productos más baratos, teniendo en cuenta que:

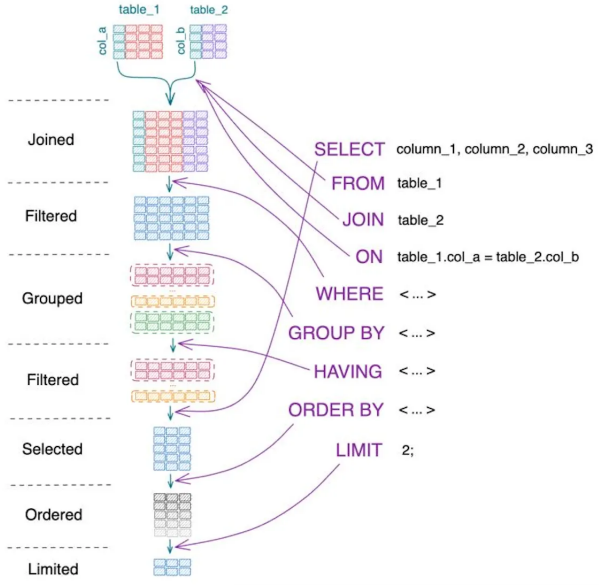
Etiqueta	Rango de precio
barato	[0, 100)
normal	[100, 1000)

Etiqueta	Rango de precio
caro	[1000, 2000)
muy caro	e.o.c

```
01 | SELECT Producto
02 | From (
03 |   SELECT
04 |     NombreP AS Producto,
05 |     CASE
06 |       WHEN Precio >= 0 AND Precio < 100 THEN 'barato'
07 |       WHEN Precio >= 100 AND Precio < 1000 THEN 'normal'
08 |       WHEN Precio >= 1000 AND Precio < 2000 THEN 'caro'
09 |       ELSE 'muy caro'
10 |     END AS `Tipo de precio`
11 |   FROM producto
12 |   ) AS table_tmp
13 | WHERE `Tipo de precio` = 'barato';
```


SQL
Queries
Sub Queries
JOIN clause

Orden de ejecución de las cláusulas



Resumen de cláusulas

Cláusula	¿Es obligatoria?	¿Qué hace?	¿Qué modifica?
Select	Sí	Identifica las columnas a mostrar en la relación	-
From	Sí	Determina la fuente de datos	-
Where	No	Restringe los registros de la fuente de datos a utilizar	From (y Join)
Join	No	Determina otras fuentes de datos y cómo relacionarlas con la establecida en el From	-
Group By	No	Agrupa los registros “similares”	-
Having	No	Restringe los grupos	Group By
Order by	No	Ordena los registros	-
Limit	No	Limita la cantidad de registros a mostrar	-

Dudas, preguntas, sugerencias ...