Base de datos de proveedores, partes y proyectos

Presentación tipo INSTANCIA

- **tabla S:** proveedores. Cada proveedor tiene: un número de proveedor S# **único**, un nombre de proveedor SNOMBRE, un valor de calificación SITUACION y está situado en una sola localidad CIUDAD.

S#	SNOMBRE	SITUACIÓN	CIUDAD
S1	Salazar	20	Londres
S2	Jaimes	10	París
S3	Bernal	30	París
S4	Corona	20	Londres
S5	Aldana	30	Atenas

- **tabla P:** partes. Cada tipo de parte tiene: un número de parte P# **único**, un nombre de parte PNOMBRE, un color COLOR, un peso PESO y una localidad donde se almacenan las partes de ese tipo CIUDAD. Cada tipo de parte tiene un solo color y se almacena en una bodega de una sola ciudad.

P#	PNOMBRE	COLOR	PESO	CIUDAD
P1	Tuerca	Rojo	12	Londres
P2	Perno	Verde	17	París
P3	Burlete	Azul	17	Roma
P4	Burlete	Rojo	14	Londres
PS	Leva	Azul	12	París
P6	Engranaje	Rojo	19	Londres

- **tabla J:** proyectos. Cada proyecto tiene: un número de proyecto J# **único**, un nombre de proyecto JNOMBRE y una localidad CIUDAD.

J#	JNOMBRE	CIUDAD
J1	Clasificador	París
J2	Perforadora	Roma
J3	Lectora	Atenas
J4	Consola	Atenas
J5	Compaginador	Londres
J6	Terminal	Oslo
J7	Cinta	Londres

- **tabla SPJ:** envíos. Significado: un proveedor suministra una parte a un proyecto en la cantidad especificada (la combinación S#-P#-J#' identifica de manera única cada registro).

S#	P#	J#	CANT
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P5	J2	100
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S5	P5	J7	100
S5	P1	J4	100
S5	P3	J4	200
S5	P4	J4	800
S5	P5	J4	400
S5	P6	J4	500

Práctica SQL (DML) Parte 1- RESUELTA

(6 de abril de 2020)

Sea la base de datos de proveedores, partes y proyectos, cuyo **esquema conceptual** es:

S (S#, SNOMBRE, SITUACIÓN, CIUDAD)
P (P#, PNOMBRE, COLOR, PESO, CIUDAD)
J (J#, JNOMBRE, CIUDAD)
SPJ (S#, P#, J#, CANT)

En los ejercicios siguientes escribir una proposición o conjunto de proposiciones en SQL para la operación indicada.

Consultas sencillas

1. **Obtener los detalles completos de todos los proyectos.**

```
SELECT    J#, JNOMBRE, CIUDAD    FROM    J;
```

Consulta alternativa: SELECT * FROM J porque se quiere visualizar todos los campos (atributos) de la tabla J.

2. **Obtener los detalles completos de todos los proyectos de Londres.**

```
SELECT    J#, JNOMBRE, CIUDAD   FROM   J
          WHERE CIUDAD = 'Londres';
```

Se asigna una condición con el WHERE para ver sólo las filas (tuplas) que la cumplen.

3. **Obtener los números de los proveedores que suministran (o envían) partes al proyecto J1, ordenados por número de proveedor.**

```
SELECT    DISTINCT S#          FROM   SPJ
     WHERE  J# = 'J1'
     ORDER  BY S# ;
```

4. **Obtener todos los envíos en los cuales la cantidad está en el intervalo de 300 a 750 inclusive.**

```
SELECT S#, P#, J#, CANT            FROM   SPJ
    WHERE  CANT >= 300    AND        CANT <= 750;
```

Alternativa: Usar WHERE Between (300, 750)
Recordar: Ver si lo admite el DBMS y ver cómo maneja los límites del intervalo

5. **Obtener una lista de todas las combinaciones parte-color/parte-ciudad, eliminando todas las parejas color/ciudad repetidas.**

```
SELECT DISTINCT COLOR, CIUDAD  FROM   P ;
```

El uso de DISTINCT hace que no se muestren las filas repetidas (filas con los mismos valores)

Reuniones (JOIN)

6. **Obtener todas las 3-uplas número de proveedor/número de parte/ número de proyecto tales que el proveedor, la parte y el proyecto indicados estén todos en la misma ciudad (cosituados).**

Se debe analizar en la expresión de la consulta que se quiere hacer dónde se encuentran los datos a ver (que se listan en SELECT) y dónde se encuentran los datos a los cuales se les pondrán condiciones (indicadas en el WHERE). Estas serán las tablas que van en FROM.

```
SELECT S#, P#, J#              FROM S, P, J
      WHERE  S.CIUDAD = P.CIUDAD    AND      P.CIUDAD = J.CIUDAD;
```

Se piden 2 (dos) condiciones que deben cumplirse al mismo tiempo, por eso se usa AND que indica que se cumpla la condición primera y la condición segunda para brindar las filas de la respuesta.

7. **Obtener todas las 3-uplas número de proveedor/número de parte/ número de proyecto tales que el proveedor, la parte y el proyecto indicados no estén todos cosituados.**

```
SELECT S#, P#, J#    FROM S, P, J
       WHERE  NOT   (S.CIUDAD = P.CIUDAD    AND   P.CIUDAD=J.CIUDAD);
```

Consulta alternativa:

```
SELECT S#, P#, J#      FROM  S, P, J
WHERE  S.CIUDAD <> P.CIUDAD   OR  P.CIUDAD <> J.CIUDAD OR S.CIUDAD <> J.CIUDAD;
```

8. **Obtener todas las 3-uplas número de proveedor/número de parte/ número de proyecto tales que el proveedor, la parte y el proyecto indicados estén todos en diferente ciudad.**

```
SELECT S#, P#, J#    FROM S, P, J
       WHERE         S.CIUDAD <> P.CIUDAD
       AND     P.CIUDAD <> J.CIUDAD
       AND    J.CIUDAD <> S.CIUDAD
```

9. **Obtener los números de las partes suministradas por algún proveedor de Londres.**

```
SELECT DISTINCT P#         FROM SPJ, S
      WHERE    SPJ.S# =     S.S#    AND   CIUDAD = 'Londres';
```

10. **Obtener los números de las partes suministradas por un proveedor de Londres a un proyecto en Londres.**

```
SELECT DISTINCT P#         FROM SPJ, S, J
       WHERE   SPJ.S# = S.S#    AND   SPJ.J# = J.J#
       AND   S.CIUDAD = 'Londres'   AND   J.CIUDAD = ' Londres';
```

11. **Obtener todas las parejas de nombres de ciudad tales que un proveedor de la primera ciudad suministre partes a un proyecto en la segunda ciudad.**

```
SELECT  DISTINCT  S.CIUDAD,  J.CIUDAD       FROM  S, SPJ, J
       WHERE     S.S# = SPJ.S#     AND    SPJ.J# = J.J#;
```

12. Obtener los números de las partes suministradas a un proyecto por un proveedor situado en la misma ciudad que el proyecto.

```
SELECT DISTINCT P#        FROM SPJ, S, J
        WHERE        SPJ.S# = S.S#   AND   SPJ.J# = J.J#
        AND   S.CIUDAD = J.CIUDAD;
```

13. Obtener los números de los proyectos a los cuales suministra partes por lo menos un proveedor situado en una ciudad distinta.

```
SELECT DISTINCT J.J#      FROM SPJ, S, J
        WHERE        SPJ.S# = S.S#   AND SPJ.J# = J.J#
        AND   S.CIUDAD <> J.CIUDAD;
```

14. Obtener todas las parejas de números de parte tales que algún proveedor suministre las dos partes indicadas.

```
SELECT  SPJX.P#  , SPJY.P#          FROM  SPJ SPJX, SPJ  SPJY
        WHERE      SPJX.S# = SPJY.S#     AND      SPJX.P# > SPJY.P#;
```

La condición resaltada en amarillo hace que las filas del tipo

~~**P1 P1**~~
~~**P1 P2**~~

no se muestren y de esta forma queda sólo la fila

P2 P1

Aunque esta condición no afecta la correcta ejecución de la consulta, su inclusión hace que el resultado mostrado al usuario humano sea más entendible por él (sacamos "ruido").

Funciones de agregados

Estas funciones se pueden utilizar solamente en dos lugares de la sentencia SELECT:

- como un dato a visualizar (columna) en la lista de columnas del SELECT
- como una condición en el HAVING que indique qué grupos se desea mostrar.

NO se pueden **utilizar** como condición en el **WHERE**.

15. Obtener el número total de proyectos a los cuales suministra partes el proveedor S1.

```
SELECT COUNT (DISTINCT J#)    FROM    SPJ
      WHERE    S#   = 'S1';
```

16. Obtener la cantidad total de la parte P1 suministrada por el proveedor S1.

```
SELECT SUM (CANT)        FROM SPJ
      WHERE  P#  = 'P1'    AND      S#       = 'S1';
```

17. Para cada parte suministrada a un proyecto, obtener el número de parte, el número de proyecto y la cantidad total correspondiente.

```
SELECT P#, J#, SUM (CANT)        FROM     SPJ
        GROUP    BY P#, J#;
```

18. Obtener los números de las partes suministradas a algún proyecto tales que la cantidad promedio suministrada sea mayor que 320.

```
SELECT DISTINCT P#        FROM SPJ
        GROUP BY P#, J#
        HAVING        AVG (CANT) > 320;
```

Operativamente, lo que hace el motor (gestor) de base de datos (en inglés DBMS, en español SGBD) utilizado es generar una tabla temporaria para cada grupo y aplicar la condición dada en el HAVING a cada una de ellas para mostrar o no ese grupo. Estas tablas temporarias tienen como "nombre" de tabla los valores que se pide agrupar.
Para la instancia que tenemos, se generan los grupos:

P1J1

P#	J#	CANT
P1	J1	200

P1J4

P#	J#	CANT
P1	J4	700
P1	J4	100

P2J2

P#	J#	CANT
P2	J2	200

P2J4, P3J1, P3J2, P3J3, P3J4, P3J5, P3J6, P3J7, P4J2, P4J4, P5J5, P5J7, P6J3, P6J4 y P6J7.

Luego se aplica la condición dada en el HAVING y se mantienen únicamente los grupos que a cumplen. Por ejemplo, el grupo P1J1 tiene una sola fila, con un valor de CANT de 200, con lo cual el promedio (AVG) es 200. Así, la condición del HAVING es FALSA y el grupo P1J1 se descarta. En cambio, el grupo P1J4, que tiene dos filas, da un promedio de cantidad de 400, en este caso, el grupo se mantiene.
Cuando termina de evaluar todos los grupos, aplica el pedido de visualización de columnas dado en la parte SELECT de la sentencia.

La parte de HAVING de la sentencia SELECT es una condición que se aplica a los grupos, esto es DEBE haber un GROUP BY para utilizar el HAVING.

En el caso de haber agregado una condición de WHERE en la consulta, **ANTES** de armar los grupos aplica esa condición quedando menos filas.
Por ejemplo: si pedimos en la consulta anterior que sean las partes provistas por el proveedor S1, la consulta queda:

```
SELECT DISTINCT P#        FROM SPJ
        WHERE  S# = 'S1'
        GROUP BY P#, J#
        HAVING        AVG (CANT) > 320;
```

y la tabla sobre la cual aplica el procedimiento descrito para el agrupado (GROUP BY) y su condición (HAVING) es:

S#	P#	J#	CANT
S1	P1	J1	200
S1	P1	J4	700

Diversas

19. Obtener todos los envíos para los cuales la cantidad no sea nula.

```
SELECT S#, P#, J#, CANT    FROM     SPJ
        WHERE CANT IS NOT NULL
```

La anterior es la respuesta "oficial". Sin embargo, ésta también funciona:

```
SELECT  S#, P#, J#, CANT   FROM SPJ
        WHERE  CANT = CANT;
```

20. Obtener números de proyecto y ciudades en los cuales la segunda letra del nombre de la ciudad sea una "o".

```
SELECT        J#, CIUDAD            FROM    J
        WHERE CIUDAD LIKE   '_o%';
```

Subconsultas

Todas estas soluciones expresadas con subconsultas también pueden resolverse con reuniones (join). Sugerencia: resolver los ejercicios siguientes utilizando reuniones.

Pero, ¿cuáles son las **ventajas de utilizar subconsultas**???

- Permite dividir el problema en **subproblemas**, dado que se pueden ir resolviendo partes de la consulta en una subconsulta.
- Mejora la **performance** dado que el resultado de una subconsulta es un valor o una lista de valores. Al hacer una reunión (join) se generan tablas más grandes a las que luego se aplican las condiciones del where y/o los group by, esto implica utilizar mayor espacio de memoria y es posible que aumente la cantidad de accesos disco-memoria y memoria-disco para resolver la consulta. Esto se verá mejor en el tema de Optimización.

21. Obtener los nombres de los proyectos a los cuales suministra partes el proveedor S1.

```
SELECT JNOMBRE            FROM     J
      WHERE J#   IN      (SELECT J#        FROM   SPJ    WHERE S# = 'S1' ) ;
```

Para evaluar condiciones en las subconsultar, los operadores son IN, NOT IN, = (si aseguro que el resultado es UN SOLO valor), < y > (si el resultado de la subconsulta es un número y lo utilizo para evaluar una condición contra un campo numérico).

22. Obtener los colores de las partes suministradas por el proveedor S1.

```
SELECT DISTINCT COLOR          FROM     P
      WHERE P# IN (SELECT P#   FROM     SPJ  WHERE S# = 'S1'  )  ;
```

23. Obtener los números de las partes suministradas a cualquier proyecto en Londres.

```
SELECT DISTINCT P#        FROM     SPJ
      WHERE     J#  IN      (SELECT J#   FROM     J    WHERE   CIUDAD  =  'Londres' ) ;
```

24. Obtener los números de los proyectos donde se utilice al menos una de las partes suministradas por el proveedor S1.

```
SELECT DISTINCT J#        FROM      SPJ
       WHERE   P#   IN      (SELECT P#     FROM SPJ    WHERE S# = 'S1'  )  ;
```

25. Obtener los números de los proveedores que suministren por lo menos una de las partes suministradas por al menos uno de los proveedores que suministran por lo menos una parte roja.

```
SELECT DISTINCT S#        FROM      SPJ   WHERE  P# IN
       (SELECT P#    FROM SPJ    WHERE  S#  IN
              ( SELECT S#    FROM SPJ  WHERE P# IN
                     (SELECT P#  FROM    P  WHERE COLOR  =  'Rojo'  ) ) ) ;
```

Las subconsultas se pueden anidar.

26. Obtener los números de proveedores cuya situación sea inferior a la del proveedor S1.

```
SELECT S#    FROM S
      WHERE  SITUACIÓN <   (SELECT SITUACION   FROM    S   WHERE S# = 'S1' ) ;
```

SITUACION es un campo numérico y lo contrasto contra un valor numérico que obtengo en la subconsulta.

27. Obtener los números de los proyectos cuya ciudad sea la primera en la lista alfabética de las ciudades donde hay proyectos.

```
SELECT J#     FROM  J
       WHERE  CIUDAD  =  (SELECT MIN (CIUDAD)  FROM      J ) ;
```

Se utiliza un igual = porque estoy segura que el resultado de la consulta es UN valor.