Carrera: Técnico Superior en Programación
Materia: Laboratorio de Computación III
Tema: Consultas de selección - Parte 3

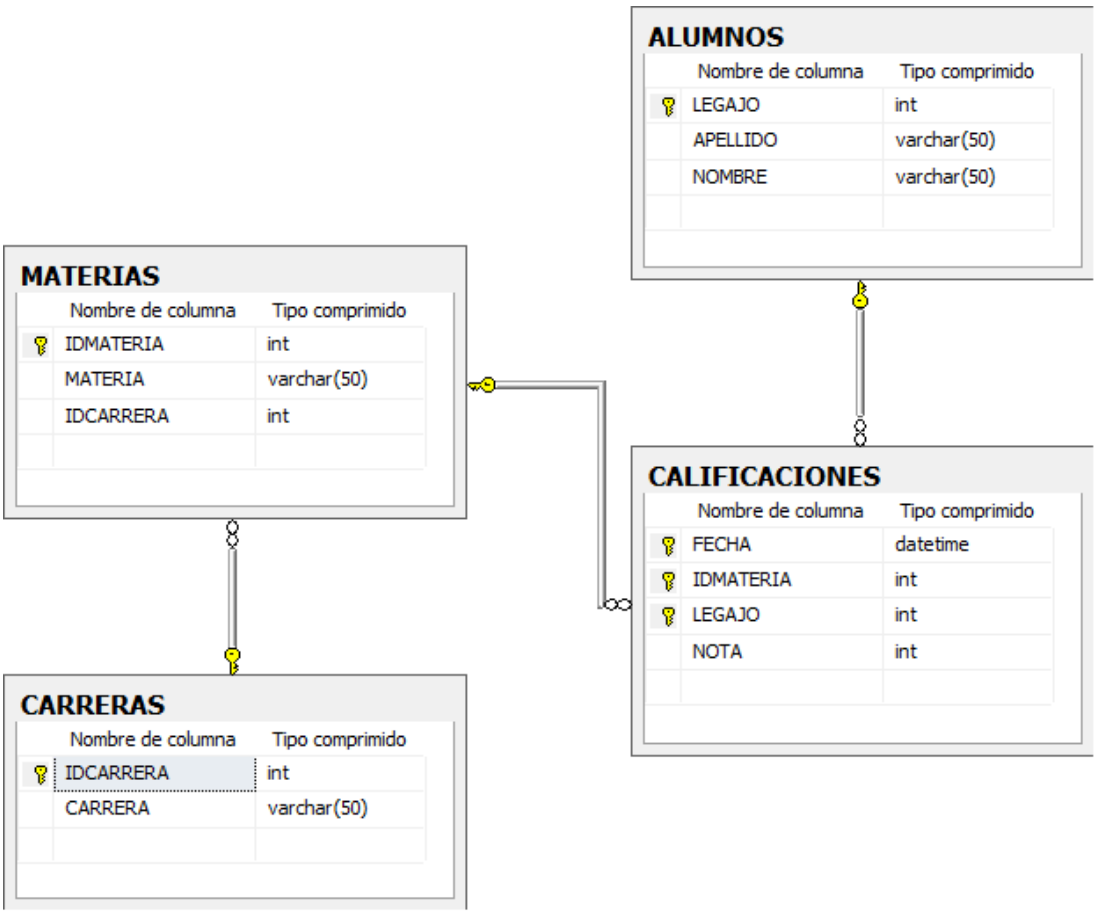
Consultas de selección - Parte 3

Consultas de resumen

Las consultas de resumen se pueden definir como el tipo de consultas cuyas filas resultantes son un resumen de las filas de la tabla de origen.
Las mismas pueden ser sobre la tabla en su totalidad o sobre un grupo de campos. Entre los resúmenes que se pueden realizar tenemos mínimo, máximo, suma, promedio y contabilización de registros.

Este tipo de consultas son muy útiles cuando queremos hacer reportes con datos sumarizados ya que no será necesario obtener todos los datos de las tablas correspondientes y luego realizar los cálculos con algún lenguaje de programación en particular (como podría ser C#, Java o PHP) sino que lo realiza la misma base de datos.

Para comenzar a trabajar con las consultas de resumen, utilizaremos una base de datos de ejemplo que nos ayudará a entender mejor el funcionamiento de este tipo de consultas.



Si realizamos un SELECT de todos los registros de la tabla de calificaciones obtendremos:

	FECHA	IDMATERIA	LEGAJO	NOTA
1	2011-05-09 00:00:00.000	12	4000	3
2	2011-05-10 00:00:00.000	9	4000	6
3	2011-05-11 00:00:00.000	9	5000	7
4	2011-05-11 00:00:00.000	10	4000	7
5	2011-05-11 00:00:00.000	10	5000	5
6	2011-05-12 00:00:00.000	11	4000	4
7	2012-04-17 15:30:19.353	1	1000	10
8	2012-04-17 15:30:19.353	1	3000	4
9	2012-04-17 15:30:19.353	2	1000	7
10	2012-04-17 15:30:19.353	2	2000	10
11	2012-04-17 15:30:19.353	2	3000	4
12	2012-04-17 15:30:19.353	3	1000	5
13	2012-04-17 15:30:19.353	3	2000	9
14	2012-04-17 15:30:19.353	3	3000	4
15	2012-04-17 15:30:19.353	4	3000	4
16	2012-04-17 15:30:19.353	5	3000	3
17	2012-04-17 15:30:19.370	6	3000	2

Se realiza una muestra de todos los registros de la tabla calificaciones para entender mejor el funcionamiento de las funciones de agregado.

Veamos algunos ejemplos de las funciones de agregado sin agrupamiento:

COUNT

La función de agregado COUNT es utilizada para contar una serie de registros

```
SELECT COUNT(*) AS CANTIDAD_APROBADOS FROM CALIFICACIONES WHERE NOTA >= 4
```

	CANTIDAD_APROBADOS
1	14

```
SELECT COUNT(*) AS CANTIDAD_TOTAL FROM CALIFICACIONES
```

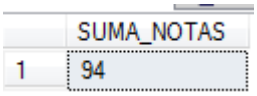
	CANTIDAD_APROBADOS
1	17

Como podemos observar la función de agregado COUNT es utilizada para contar registros. La misma puede ser utilizada en conjunto con la cláusula WHERE para poder filtrar la cantidad de registros a contar. En el ejemplo de la primer consulta, sólo se tienen en cuenta los registros cuya nota supere la calificación 4.

SUM

La función de agregado SUM es utilizada para sumar una columna entre una serie de registros.

```
SELECT SUM(NOTA) AS SUMA_NOTAS FROM CALIFICACIONES
```



	SUMA_NOTAS
1	94

La función de agregado SUM es utilizada para sumar el valor numérico de una columna dentro de un conjunto de registros. En el caso del ejemplo anterior, se suman todas los valores numéricos de las calificaciones sin aplican ningún tipo de filtro.

```
SELECT SUM(NOTA)/COUNT(*)  AS PROMEDIO_NOTAS FROM CALIFICACIONES
```

	PROMEDIO_NOTAS
1	5

En la consulta anterior, queda ejemplificado como la combinación del uso de la función SUM con la función COUNT permite obtener, en este caso, en promedio de calificaciones total.

AVG

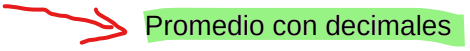
La función de agregado AVG es utilizada para calcular el promedio de una columna entre una serie de registros.

```
SELECT AVG(NOTA) AS PROMEDIO_NOTAS FROM CALIFICACIONES
```

	PROMEDIO_NOTAS
1	5

En el ejemplo anterior, se obtiene el promedio de las calificaciones mediante el uso de la función AVG sobre la columna NOTA. De ésta manera, obtenemos el mismo resultado que aplicando la suma de notas dividido por la cantidad de registros.

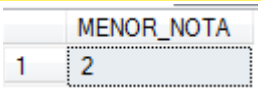
Hay que tener en cuenta que si la columna que se pretende calcular el promedio es del tipo INT en alguna de sus variantes, el resultado del promedio se mostrará sin sus valores decimales. Para que aparezca con valores decimales, una buena solución es multiplicar a la columna por 1.0. Por ejemplo, AVG(NOTA * 1.0)



MIN

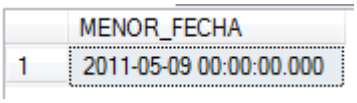
La función de agregado MIN es utilizada para obtener el valor mínimo de una columna entre una serie de registros.

```
SELECT MIN(NOTA) AS MENOR_NOTA FROM CALIFICACIONES
```



	MENOR_NOTA
1	2

```
SELECT MIN(FECHA) AS MENOR_FECHA FROM CALIFICACIONES
```



	MENOR_FECHA
1	2011-05-09 00:00:00.000

La función MIN se puede utilizar para obtener el valor mínimo de una columna. Como se puede

notar es utilizable en diversos tipos de columnas, funciona tanto para valores numéricos como la nota o valores de tipo fecha como la fecha de la calificación.

MAX

La función de agregado MAX es utilizada para obtener el valor máximo de una columna entre una serie de registros.

```
SELECT MAX(NOTA) AS MAYOR_NOTA FROM CALIFICACIONES
```

	MAYOR_NOTA
1	10

Hasta el momento, hemos visto que todos los ejemplos sumarizaron datos pero utilizando la totalidad de la tabla. Es decir, no hubo ningún tipo de agrupamiento para realizar el resúmen de los datos y es por eso también que siempre obtuvimos un registro y una columna en nuestro listado.
Generalmente, este tipo de consultas no nos será de gran utilidad. Ya que es preferible tener un agrupamiento en la sumarización. Es decir, en los ejemplos que hemos visto, probablemente necesitemos la cantidad de exámenes rendidos por alumno, el promedio de notas por materia, la cantidad de materias por carrera, etc.
Para aplicar un agrupamiento a una consulta de agregado se utiliza la cláusula GROUP BY.

La cláusula GROUP BY

En los ejemplos anteriores hemos hecho consultas en las que solamente obteníamos una columna (o un valor) al utilizar funciones de agregado sin agrupamiento (el SELECT sólo contiene la función).

Sin embargo es muy usual que, además del o los valores que obtenemos con la función de agregado, necesitemos otros valores provenientes de otras columnas. En este caso la naturaleza de las funciones de agregado es la misma, a diferencia que ahora debemos entenderlo en el contexto de un agrupamiento de información. A continuación veremos una serie de ejemplos para dejarlo más claro:

- La cantidad de exámenes rendidos por alumno (se necesita el nombre y el apellido).

Para realizar ésta consulta primero debemos comprender qué tipo de función de sumarización necesitamos y luego qué tipo de agrupamiento hay que aplicar. Está claro que vamos a querer obtener el nombre y apellido de cada alumno y a continuación otra columna para registrar la cantidad de materias que dicho alumno rindió.

Por lo tanto las columnas a mostrar podrían llamarse 'Apellido y nombre' y 'Cantidad de materias rendidas'. De modo que tendremos que utilizar la función COUNT para tal fin.

La consulta final quedaría:

```
SELECT A.APELLIDO + ', ' + A.NOMBRE AS Apenom, COUNT(*) AS 'Cantidad de
materias' FROM ALUMNOS A INNER JOIN CALIFICACIONES C ON A.LEGAJO = C.LEGAJO
GROUP BY A.APELLIDO + ', ' + A.NOMBRE
ORDER BY COUNT(*) DESC
```

	Apenom	Cantidad de materias
1	COSTANZA, GEORGE	6
2	KRAMER, COSMO	4
3	SEINFELD, JERRY	3
4	DAVOLA, JOE	2
5	BENES, ELAINE	2

Como resultado obtenemos un listado que cuenta la cantidad de materias rendidas por cada alumno y que se encuentra agrupado por apellido y nombre. Además se puede ver como el listado puede ser ordenado por la cantidad de registros que contó, en este caso de mayor a menor.
La clave aquí se encuentra en la cláusula GROUP BY, sin ella no podría llevarse a cabo el agrupamiento de registros y por lo tanto no se podría contar las filas aplicando dicho criterio. Esto quiere decir que en una consulta de agrupamiento, todas las columnas no calculables (que no hayan sido resumidas mediante COUNT, SUM, AVG, MIN o MAX) deberán aperecer en la cláusula GROUP BY.

Para completar este ejemplo en particular nos restaría preguntarnos, ¿Y si hay alumnos que no hayan rendido materias? ¿Por qué no aparecen aquellos alumnos que no hayan rendido materias?

Básicamente vamos a querer traer a todos los alumnos y si no figuran en la tabla de CALIFICACIONES es porque no rindieron exámenes y por lo tanto deberían contabilizar 0 materias rendidas.
Por lo que la respuesta a la pregunta empieza por utilizar un tipo de JOIN que me permita traer los registros de la izquierda sin importar su existencia en la tabla de la derecha: LEFT JOIN.
Pero como veremos en el siguiente ejemplo (que no soluciona completamente lo que deseamos), no es únicamente LEFT JOIN el cambio que debemos hacer:

```
SELECT A.APELLIDO + ', ' + A.NOMBRE AS Apenom, COUNT(*) AS 'Cantidad de
materias' FROM ALUMNOS A LEFT JOIN CALIFICACIONES C ON A.LEGAJO = C.LEGAJO
GROUP BY A.APELLIDO + ', ' + A.NOMBRE
ORDER BY COUNT(*) DESC
```

	Apenom	Cantidad de materias
1	COSTANZA, GEORGE	6
2	KRAMER, COSMO	4
3	SEINFELD, JERRY	3
4	DAVOLA, JOE	2
5	BENES, ELAINE	2
6	BANIA, KENNY	1

Al tener COUNT(*) con asterisco,
te cuenta todos los registros y los que están en NULL también lo toma como válido

NO

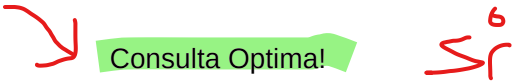
Sin caer en la misma explicación que figura en párrafos más arriba, simplemente reemplazamos el INNER JOIN por el LEFT JOIN y efectivamente obtenemos el sexto registro. El alumno Kenny Bania no rindió ningún examen, por lo que no aparece su legajo en la tabla de calificaciones pero si observamos la cantidad de materias del listado, le contabiliza uno.

¿Por qué? … Cómo habíamos mencionado anteriormente, LEFT JOIN no iba a ser suficiente. El problema de la consulta anterior es la utilización del COUNT(*). Al utilizar el asterisco con que

figure algún campo con algún valor en el registro este será contabilizado. Nosotros queremos que figure un registro de nota con algún valor y si no lo encuentra o encuentra NULL entonces sea contabilizado como cero.

```
SELECT A.APELLIDO + ', ' + A.NOMBRE AS Apenom, COUNT(C.NOTA) AS 'Cantidad de
materias' FROM ALUMNOS A LEFT JOIN CALIFICACIONES C ON A.LEGAJO = C.LEGAJO
GROUP BY A.APELLIDO + ', ' + A.NOMBRE
ORDER BY COUNT(C.NOTA) DESC
```

	Apenom	Cantidad de materias
1	COSTANZA, GEORGE	6
2	KRAMER, COSMO	4
3	SEINFELD, JERRY	3
4	DAVOLA, JOE	2
5	BENES, ELAINE	2
6	BANIA, KENNY	0



Veamos el ejemplo para obtener un promedio de notas por materia

- Criterio de sumarización: Promedio (AVG)
- Criterio de agrupamiento: Agrupados por nombre de materia

El listado que se pretende obtener debería mostrar el nombre de la materia y en la siguiente columna el cálculo del promedio. Sería importante que el promedio aparezca con sus valores decimales es por eso que como se explicó anteriormente será necesario multiplicar a la columna por 1.0 para que se procese como un campo decimal.

```
SELECT M.MATERIA, AVG(C.NOTA*1.0) AS 'Promedio notas' FROM MATERIAS M LEFT JOIN
CALIFICACIONES C ON C.IDMATERIA = M.IDMATERIA
GROUP BY M.MATERIA
```

	MATERIA	Promedio notas
1	ALGEBRA	NULL
2	AUDITORIA EN INFORMATICA	6.000000
3	FISICA I	NULL
4	INGENIERIA DEL CONOCIMIENTO	6.500000
5	LABORATORIO I	7.000000
6	LABORATORIO V	4.000000
7	MATEMATICA II	2.000000
8	METODOLOGIA DE SISTEMAS	6.000000
9	PROCESOS ESTOCASTICOS Y SIMULACION	4.000000
10	PROGRAMACION I	7.000000
11	REDES	3.000000
12	SISTEMAS DE SOPORTE DE DECISION	3.000000

Cláusula HAVING

Ahora que podemos obtener listados con valores sumarizados o calculados ya sea agrupando datos o utilizando toda la tabla, también sería interesante poder ponerle condiciones a dichos cálculos.

En el primero de los ejemplos habíamos contabilizado todas las calificaciones que sean mayores o iguales que 4. Ese tipo de restricción es simple ya que se realiza sobre uno de los campos no sumarizados y se lleva a cabo mediante la cláusula WHERE.

Pero distinto sería si quisieramos obtener un listado de materias y sus promedios de calificaciones. Pero sólo incluirlos al listado si el promedio calculado supera el valor 6,1.

Nótese que no se quiere calcular el promedio sólo de las notas que superen el valor 6,1 sino que se quiere calcular el promedio de todas las calificaciones agrupado por materia y luego determinar si debe incluirse al listado o no si éste supera el valor deseado. Ese tipo de restricción, que se realiza sobre un valor de una columna calculada por una función de agregado se realiza utilizando la cláusula HAVING.

Veamos el ejemplo en el que queremos obtener el promedio agrupado por nombre de materia sólo si este supera los 6,1 puntos.

```
SELECT M.MATERIA, AVG(C.NOTA*1.0) AS 'Promedio notas' FROM MATERIAS M LEFT JOIN
CALIFICACIONES C ON C.IDMATERIA = M.IDMATERIA
GROUP BY M.MATERIA
HAVING AVG(C.NOTA*1.0) > 6.1
```

	MATERIA	Promedio notas
1	INGENIERIA DEL CONOCIMIENTO	6.500000
2	LABORATORIO I	7.000000
3	PROGRAMACION I	7.000000

Por último, veamos un último ejemplo que puede resultar un tanto confuso:

```
SELECT A.LEGAJO, A.APELLIDO + ', ' + A.NOMBRE AS APENOM, MAX(C.NOTA) AS 'NOTA
MAXIMA' FROM ALUMNOS A
INNER JOIN CALIFICACIONES C ON C.LEGAJO = A.LEGAJO
GROUP BY A.LEGAJO, A.APELLIDO + ', ' + A.NOMBRE
```

En principio cuando vemos la siguiente consulta pensamos que vamos a obtener el legajo, apellido y nombre y nota del alumno que mayor calificación haya obtenido.
Sin embargo, sólo obteníamos la nota máxima cuando no aplicábamos ningún tipo de agrupamiento. En ese caso, obteníamos la calificación máxima de la tabla.
En cambio y como podemos observar en el resultado de la consulta que figura abajo, lo que realmente estamos obteniendo con dicha consulta es la nota máxima agrupado por legajo y apellido y nombre. Esto quiere decir que obtenemos la nota máxima que haya obtenido cada uno de los alumnos.

	LEGAJO	APENOM	NOTA MAXIMA
1	1000	SEINFELD, JERRY	10
2	2000	BENES, ELAINE	10
3	3000	COSTANZA, GEORGE	4
4	4000	KRAMER, COSMO	7
5	5000	DAVOLA, JOE	7