



**UNIVERSIDAD AUTÓNOMA
DE AGUASCALIENTES**

**CENTRO DE CIENCIAS BÁSICA
DEPARTAMENTO DE SISTEMAS ELECTRÓNICOS
BASES DE DATOS DISTRIBUIDAS**

Esquemas de fragmentación usando vistas SQL

DOCENTE:

Luis Eduardo Bautista Villalpando

ALUMNO:

Gerardo Martínez Martínez

Brayan de Jesús Capetillo Macias

8º A

Fecha: 12 de abril de 2022

Contenido

Introducción	3
Contenido	4
Iniciando sesión.....	4
Seleccionando la base de datos	4
Mostrando las tablas contenidas en la base de datos	4
Mostrando el contenido de la tabla PROJ.....	5
Creando las primeras 2 vistas	5
Mostrando el contenido de las vistas	6
Insertando nuevos proyectos dentro de la tabla PROJ.....	6
Mostrando el contenido de la tabla PROJ.....	7
Mostrando el contenido de las vistas	7
Fragmentando la tabla EMP.....	8
Verificando el contenido de las vistas de EMP.....	9
Insertando nuevos datos en EMP	9
Mostrando el contenido de las vistas (Fragmentos) de EMP	10
Fragmentando la tabla ASG.....	10
Mostrando el contenido de las vistas de ASG.....	11
Insertando nuevos datos en ASG	12
Mostrando el contenido de las vistas (Fragmentos) de ASG	12
Fragmentando verticalmente la tabla PROJ.....	13
Verificando el contenido de las fragmentaciones verticales de PROJ	14
Comentado el problema	14
Sugerencia de cambios.....	14
Insertando nuevos datos en PROJ.....	16
Mostrando el contenido de la fragmentación vertical de PROJ	16
Conclusión	17

Introducción

En este documento se encuentra los pasos realizados para completar el séptimo laboratorio de bases de datos distribuidas, en la cual nos introducimos de forma practica a lo que son los esquemas de fragmentación.

Un aspecto importante por considerar es que no se utiliza la fragmentación como regularmente se aplica en los casos de producción, esto porque se requiere pagar licencias que nos permitan acceder a tales recursos del manejador de base de datos.

Por el motivo anterior vamos a simular fragmentación utilizando una herramienta útil cuando hablamos de bases de datos; nos referimos a las vistas. Las vistas son un objeto de bases de datos que no contiene valores, si no referencias de acceso rápido a consultas predefinidas. Estas contienen filas y columnas como la tabla real, por lo regular solo contienen algunas filas (tuplas) y columnas que más sean convenientes para acceder más rápido a la información u otro caso de uso es la combinación de varias tablas (utilizando joins con llaves primarias y foráneas).

Como ya lo habíamos visto en clase existen dos tipos de fragmentación: vertical y horizontal.

Comencemos hablando de la horizontal, a nuestro parecer es la más utilizada ya que agrupa todas las cabeceras de las de la tabla en diferentes secciones, ósea agrupa las tuplas completas sin perder ningún atributo, la división puede estar definida por alguna diferencia entre los valores de algún atributo de la tabla. Esto se entenderá un poco más con las actividades realizadas a continuación.



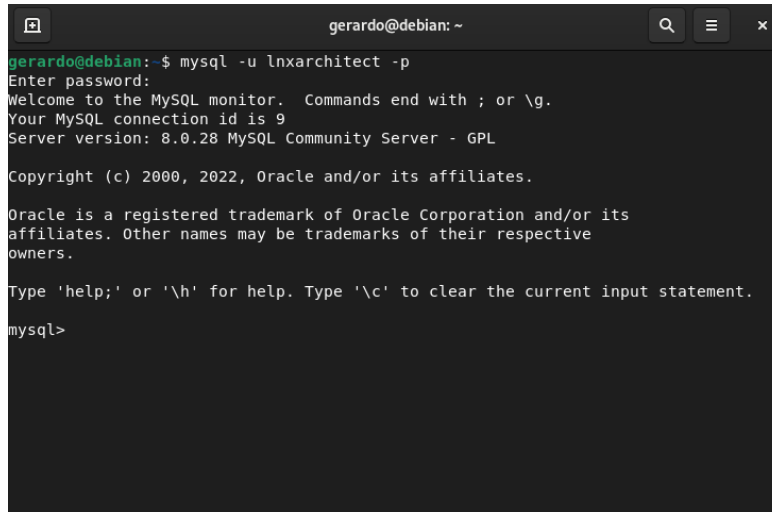
Pasando a la fragmentación vertical, entendemos que las tablas se dividen completamente separando una o algunas de los atributos, con lo que tenemos toda la información de un atributo de la tabla en una sola tabla.



Contenido

Iniciando sesión

Iniciamos sesión con el usuario arquitecto de Linux.

A terminal window titled 'gerardo@debian: ~' showing the MySQL login process. The user enters 'mysql -u lnxarchitect -p', followed by a password prompt. The MySQL monitor displays a welcome message, connection ID (9), server version (8.0.28), and copyright information. The prompt 'mysql>' is shown at the bottom.

```
gerardo@debian:~$ mysql -u lnxarchitect -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.28 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

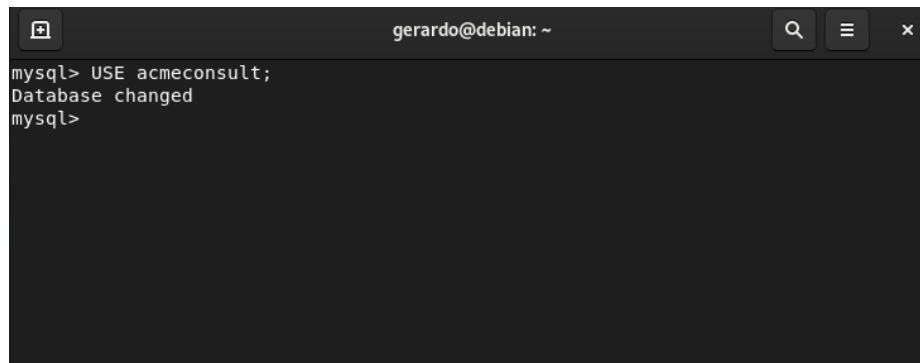
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Seleccionando la base de datos

Indicamos que vamos a utilizar la base de datos acmeconsult.

A terminal window titled 'gerardo@debian: ~' showing the MySQL prompt. The user enters 'USE acmeconsult;', and the MySQL monitor responds with 'Database changed'. The prompt 'mysql>' is shown at the bottom.

```
mysql> USE acmeconsult;
Database changed
mysql>
```

Mostrando las tablas contenidas en la base de datos

Mostramos cuales son las tablas que contiene.

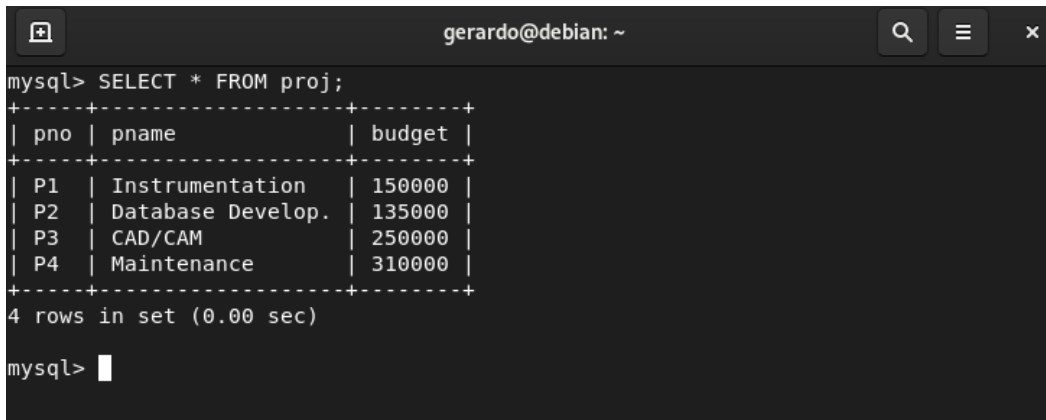
A terminal window titled 'gerardo@debian: ~' showing the MySQL prompt. The user enters 'SHOW TABLES;', and the MySQL monitor displays a list of tables in the 'acmeconsult' database: 'asg', 'emp', 'pay', 'pay2', and 'proj'. The output is formatted as a table with a header row and a footer indicating 5 rows in set (0.00 sec). The prompt 'mysql>' is shown at the bottom.

```
mysql> SHOW TABLES;
+-----+
| Tables_in_acmeconsult |
+-----+
| asg                    |
| emp                    |
| pay                    |
| pay2                   |
| proj                   |
+-----+
5 rows in set (0.00 sec)

mysql>
```

Mostrando el contenido de la tabla PROJ

Mostramos todo lo que contiene la tabla PROJ con una operación SELECT.



```
gerardo@debian: ~  
mysql> SELECT * FROM proj;  
+-----+-----+-----+  
| pno | pname           | budget |  
+-----+-----+-----+  
| P1  | Instrumentation | 150000 |  
| P2  | Database Develop. | 135000 |  
| P3  | CAD/CAM         | 250000 |  
| P4  | Maintenance     | 310000 |  
+-----+-----+-----+  
4 rows in set (0.00 sec)  
  
mysql>
```

Vemos que solamente tenemos 4 proyectos registrados.

Creando las primeras 2 vistas

Vamos a crear dos vistas que dividan los proyectos por su presupuesto, una vista contendrá los proyectos con un presupuesto menor a \$200,000 y otra vista los que sea mayor o igual a \$200,000.

La sintaxis para la creación es la siguiente:



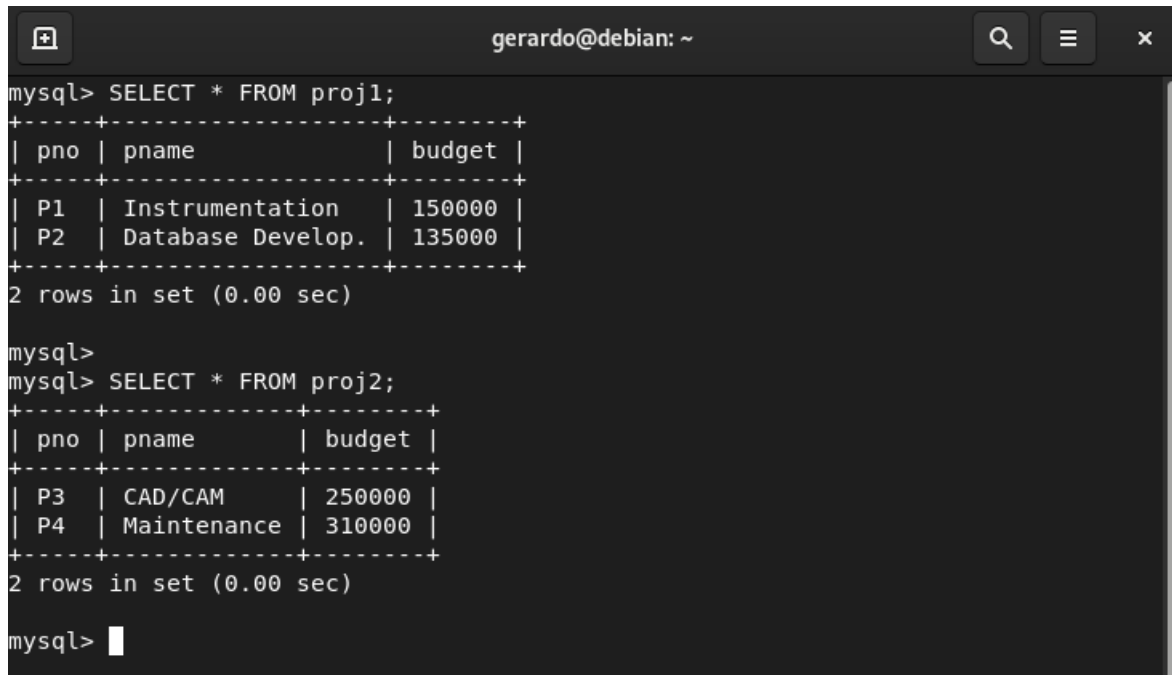
```
gerardo@debian: ~  
mysql> CREATE VIEW proj1 AS SELECT * FROM proj WHERE budget < 200000;  
Query OK, 0 rows affected (0.41 sec)  
  
mysql> CREATE VIEW proj2 AS SELECT * FROM proj WHERE budget >= 200000;  
Query OK, 0 rows affected (0.02 sec)  
  
mysql>
```

Las tablas están diseñadas para que la mitad de los valores se referencien dentro de una vista y la otra mitad dentro de la otra, por tal motivo cuando seleccionemos lo que contiene estas vistas tendremos que cada una contiene 2 registros.

Esto simula una fragmentación horizontal ya que nos estamos llevando todos los atributos de una tabla para crea una o varias tablas “nuevas”, esto es la fragmentación, pero por obvias razones cuando lo trabajamos como es, estos registros se pueden encontrar en dos o muchos más diferentes servidores físicos, en este caso se encuentra tanto en el mismo servidor físico, como en la misma manejador ya que estamos utilizando vistas.

Mostrando el contenido de las vistas

Ahora mostraremos los que contiene estas vistas.

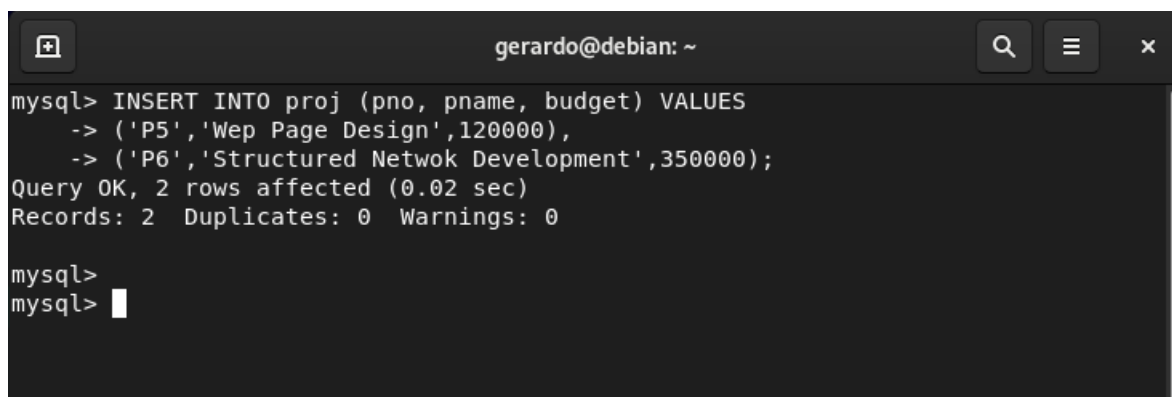


```
gerardo@debian: ~  
mysql> SELECT * FROM proj1;  
+-----+-----+-----+  
| pno | pname           | budget |  
+-----+-----+-----+  
| P1  | Instrumentation | 150000 |  
| P2  | Database Develop. | 135000 |  
+-----+-----+-----+  
2 rows in set (0.00 sec)  
  
mysql>  
mysql> SELECT * FROM proj2;  
+-----+-----+-----+  
| pno | pname           | budget |  
+-----+-----+-----+  
| P3  | CAD/CAM         | 250000 |  
| P4  | Maintenance     | 310000 |  
+-----+-----+-----+  
2 rows in set (0.00 sec)  
  
mysql> █
```

Como ya lo había comentado la tabla original se dividió en 2, esta es la ventaja de la fragmentación, en una implementación real estas dos tablas podrían estar en diferentes servidores físicos y aun así mantener la lógica como si manejáramos una sola tabla.

Insertando nuevos proyectos dentro de la tabla PROJ

Vamos a insertar dos nuevos valores a la tabla PROJ para demostrar que los datos se actualizan en las vistas.



```
gerardo@debian: ~  
mysql> INSERT INTO proj (pno, pname, budget) VALUES  
-> ('P5', 'Wep Page Design', 120000),  
-> ('P6', 'Structured Netwok Development', 350000);  
Query OK, 2 rows affected (0.02 sec)  
Records: 2 Duplicates: 0 Warnings: 0  
  
mysql>  
mysql> █
```

Veamos el valor del presupuesto, por ende, uno de estos registros pertenecerá a la vista PROJ1 y otro a la PROJ2

Mostrando el contenido de la tabla PROJ

Vemos que la tabla original sigue con los mismos valores iniciales y los nuevos que ingresamos.

```
gerardo@debian: ~  
mysql> SELECT * FROM proj;  
+-----+-----+-----+  
| pno | pname                | budget |  
+-----+-----+-----+  
| P1  | Instrumentation      | 150000 |  
| P2  | Database Develop.    | 135000 |  
| P3  | CAD/CAM              | 250000 |  
| P4  | Maintenance          | 310000 |  
| P5  | Wep Page Design      | 120000 |  
| P6  | Structured Netwok Development | 350000 |  
+-----+-----+-----+  
6 rows in set (0.00 sec)  
  
mysql>
```

Ahí podemos ver que realmente si se insertaron los datos en la tabla original.

Mostrando el contenido de las vistas

Pero ahora veamos que paso con las vistas.

```
gerardo@debian: ~  
mysql> SELECT * FROM proj1;  
+-----+-----+-----+  
| pno | pname                | budget |  
+-----+-----+-----+  
| P1  | Instrumentation      | 150000 |  
| P2  | Database Develop.    | 135000 |  
| P5  | Wep Page Design      | 120000 |  
+-----+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql>  
mysql> SELECT * FROM proj2;  
+-----+-----+-----+  
| pno | pname                | budget |  
+-----+-----+-----+  
| P3  | CAD/CAM              | 250000 |  
| P4  | Maintenance          | 310000 |  
| P6  | Structured Netwok Development | 350000 |  
+-----+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql>
```

Ya lo habíamos comentado, un proyecto entraría a una vista y otro a otra. Véase que el proyecto P5 entro a la vista PROJ1 y el proyecto P6 a la vista PROJ2.

Fragmentando la tabla EMP

Esta es la estructura y contenido de la tabla EMP

```
gerardo@debian: ~  
mysql> SELECT * FROM emp;  
+-----+-----+-----+  
| eno |  ename  | title      |  
+-----+-----+-----+  
| E1  | J. Doe  | Elect. Eng. |  
| E2  | M. Smith| Syst. Anal. |  
| E3  | A. Lee  | Mech. Eng.  |  
| E4  | J. Miller| Programmer  |  
| E5  | B. Casey| Syst. Anal. |  
| E6  | L. Chu  | Elect. Eng. |  
| E7  | R. Davis| Mech. Eng.  |  
| E8  | J. Jones| Syst. Anal. |  
+-----+-----+-----+  
8 rows in set (0.00 sec)  
  
mysql>  
mysql>
```

Lo que se busca es hacer 3 fragmentaciones, una que contenga todos aquellos empleados que tenga el título de Ingeniero electrónico ("Elect. Eng."), otra que contenga a los empleados con título de Ingeniero Mecánico ("Mech. Eng.") y una última que contenga a todos los demás.

Para ello hacemos lo siguiente.

```
gerardo@debian: ~  
mysql> CREATE VIEW electeng AS SELECT * FROM emp WHERE title = 'Elect. Eng.';  
Query OK, 0 rows affected (0.02 sec)  
  
mysql> CREATE VIEW mecheng AS SELECT * FROM emp WHERE title = 'Mech. Eng.';  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> CREATE VIEW other AS SELECT * FROM emp WHERE title != 'Mech. Eng.' and title != 'Elect. Eng.';  
Query OK, 0 rows affected (0.03 sec)  
  
mysql> █
```


Verificando el contenido de las vistas de EMP

```
gerardo@debian: ~  
mysql> SELECT * FROM electeng;  
+-----+-----+-----+  
| eno | ename | title |  
+-----+-----+-----+  
| E1  | J. Doe | Elect. Eng. |  
| E6  | L. Chu | Elect. Eng. |  
+-----+-----+-----+  
2 rows in set (0.00 sec)  
  
mysql> SELECT * FROM mecheng;  
+-----+-----+-----+  
| eno | ename | title |  
+-----+-----+-----+  
| E3  | A. Lee | Mech. Eng. |  
| E7  | R. Davis | Mech. Eng. |  
+-----+-----+-----+  
2 rows in set (0.00 sec)  
  
mysql> SELECT * FROM other;  
+-----+-----+-----+  
| eno | ename | title |  
+-----+-----+-----+  
| E2  | M. Smith | Syst. Anal. |  
| E4  | J. Miller | Programmer |  
| E5  | B. Casey | Syst. Anal. |  
| E8  | J. Jones | Syst. Anal. |  
+-----+-----+-----+  
4 rows in set (0.00 sec)  
  
mysql>
```

Podemos ver que realmente si se cumplen las restricciones de creación de vistas.

Insertando nuevos datos en EMP

Para hacer la misma prueba que con la tabla PROJ, vamos a insertar nuevos datos en la tabla original, cuidando que cada uno entre a una vista diferente.

```
gerardo@debian: ~  
mysql> INSERT INTO emp(eno, ename, title) values  
-> ('E9', 'Lalo', 'Elect. Eng.'),  
-> ('E10', 'Ruth', 'Mech. Eng.'),  
-> ('E11', 'Brayan', 'Programmer'),  
-> ('E12', 'Alex', 'Other');  
Query OK, 4 rows affected (0.01 sec)  
Records: 4 Duplicates: 0 Warnings: 0  
  
mysql> SELECT * FROM emp;  
+-----+-----+-----+  
| eno | ename | title |  
+-----+-----+-----+  
| E1  | J. Doe | Elect. Eng. |  
| E10 | Ruth  | Mech. Eng. |  
| E11 | Brayan | Programmer |  
| E12 | Alex  | Other |  
| E2  | M. Smith | Syst. Anal. |  
| E3  | A. Lee | Mech. Eng. |  
| E4  | J. Miller | Programmer |  
| E5  | B. Casey | Syst. Anal. |  
| E6  | L. Chu | Elect. Eng. |  
| E7  | R. Davis | Mech. Eng. |  
| E8  | J. Jones | Syst. Anal. |  
| E9  | Lalo  | Elect. Eng. |  
+-----+-----+-----+  
12 rows in set (0.00 sec)  
  
mysql>
```

Mostrando el contenido de las vistas (Fragmentos) de EMP

```
gerardo@debian: ~  
mysql> SELECT * FROM electeng;  
+-----+-----+-----+  
| eno | ename | title |  
+-----+-----+-----+  
| E1  | J. Doe | Elect. Eng. |  
| E6  | L. Chu | Elect. Eng. |  
| E9  | Lalo  | Elect. Eng. |  
+-----+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql> SELECT * FROM mecheng;  
+-----+-----+-----+  
| eno | ename | title |  
+-----+-----+-----+  
| E10 | Ruth  | Mech. Eng. |  
| E3  | A. Lee | Mech. Eng. |  
| E7  | R. Davis | Mech. Eng. |  
+-----+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql> SELECT * FROM other;  
+-----+-----+-----+  
| eno | ename | title |  
+-----+-----+-----+  
| E11 | Brayan | Programmer |  
| E12 | Alex   | Other      |  
| E2  | M. Smith | Syst. Anal. |  
| E4  | J. Miller | Programmer |  
| E5  | B. Casey | Syst. Anal. |  
| E8  | J. Jones | Syst. Anal. |  
+-----+-----+-----+  
6 rows in set (0.00 sec)  
  
mysql>
```

Con esto podemos comprobar que si se realizaron correctamente las validaciones y que cada empleado entra a la vista o fragmento al que pertenece.

Fragmentando la tabla ASG

Esta parte queda libre de fragmentar a consideración de nosotros los arquitectos, por este motivo, decidimos analizar primero la estructura:

```
gerardo@debian: ~  
mysql> SELECT * FROM asg;  
+-----+-----+-----+-----+  
| eno | pno | resp | dur |  
+-----+-----+-----+-----+  
| E1  | P1  | Manager | 12 |  
| E2  | P1  | Analyst | 24 |  
| E2  | P2  | Analyst | 6 |  
| E3  | P3  | Consultant | 10 |  
| E3  | P4  | Enginner | 48 |  
| E4  | P2  | Programmer | 18 |  
| E5  | P2  | Manager | 24 |  
| E6  | P4  | Manager | 48 |  
| E7  | P3  | Engineer | 36 |  
| E8  | P3  | Manager | 40 |  
+-----+-----+-----+-----+  
10 rows in set (0.00 sec)  
  
mysql>
```

Viendo sus atributos decidimos hacer una fragmentación horizontal separado por su duración. Dicha fragmentación creará tres vistas o fragmentos, el primero contendrá los proyectos de corta duración [0,15), el segundo de mediana [15, 25) y el tercer de larga duración [25, ∞).

Para ello establecemos las siguientes vistas.

```
gerardo@debian: ~  
mysql> CREATE VIEW min AS SELECT * FROM asg WHERE dur < 15;  
Query OK, 0 rows affected (0.02 sec)  
  
mysql> CREATE VIEW med AS SELECT * FROM asg WHERE dur >= 15 and dur < 25;  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> CREATE VIEW max AS SELECT * FROM asg WHERE dur >= 25;  
Query OK, 0 rows affected (0.03 sec)  
  
mysql>
```

Mostrando el contenido de las vistas de ASG

```
gerardo@debian: ~  
mysql> SELECT * FROM min;  
+-----+-----+-----+-----+  
| eno | pno | resp      | dur |  
+-----+-----+-----+-----+  
| E1  | P1  | Manager   | 12  |  
| E2  | P2  | Analyst   | 6   |  
| E3  | P3  | Consultant | 10  |  
+-----+-----+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql> SELECT * FROM med;  
+-----+-----+-----+-----+  
| eno | pno | resp      | dur |  
+-----+-----+-----+-----+  
| E2  | P1  | Analyst   | 24  |  
| E4  | P2  | Programmer | 18  |  
| E5  | P2  | Manager   | 24  |  
+-----+-----+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql> SELECT * FROM max;  
+-----+-----+-----+-----+  
| eno | pno | resp      | dur |  
+-----+-----+-----+-----+  
| E3  | P4  | Enginner  | 48  |  
| E6  | P4  | Manager   | 48  |  
| E7  | P3  | Engineer  | 36  |  
| E8  | P3  | Manager   | 40  |  
+-----+-----+-----+-----+  
4 rows in set (0.01 sec)  
  
mysql>
```

Podemos ver cada una de las salidas y todo está correcto, falta hacer las pruebas de inserción a la tabla original para comprobar que todo funcione a la perfección.

Insertando nuevos datos en ASG

```
gerardo@debian: ~  
mysql> INSERT INTO asg(eno, pno, resp, dur) values  
-> ('E9', 'P5', 'Engineer', 14),  
-> ('E10', 'P6', 'Enginner', 23),  
-> ('E11', 'P5', 'Programmer', 25),  
-> ('E12', 'P6', 'Programmer', 300);  
Query OK, 4 rows affected (0.34 sec)  
Records: 4 Duplicates: 0 Warnings: 0  
  
mysql> SELECT * FROM asg;  
+-----+-----+-----+-----+  
| eno | pno | resp      | dur |  
+-----+-----+-----+-----+  
| E1  | P1  | Manager   | 12  |  
| E2  | P1  | Analyst   | 24  |  
| E2  | P2  | Analyst   | 6   |  
| E3  | P3  | Consultant| 10  |  
| E3  | P4  | Enginner  | 48  |  
| E4  | P2  | Programmer| 18  |  
| E5  | P2  | Manager   | 24  |  
| E6  | P4  | Manager   | 48  |  
| E7  | P3  | Engineer  | 36  |  
| E8  | P3  | Manager   | 40  |  
| E9  | P5  | Engineer  | 14  |  
| E10 | P6  | Enginner  | 23  |  
| E11 | P5  | Programmer| 25  |  
| E12 | P6  | Programmer| 300 |  
+-----+-----+-----+-----+  
14 rows in set (0.00 sec)  
  
mysql>
```

Ingresamos 4 nuevos valores, de los cuales 1 va a ingresar a proyectos de corta duración, 1 a mediana y 2 a larga duración.

Mostrando el contenido de las vistas (Fragmentos) de ASG

```
gerardo@debian: ~  
mysql> SELECT * FROM min;  
+-----+-----+-----+-----+  
| eno | pno | resp      | dur |  
+-----+-----+-----+-----+  
| E1  | P1  | Manager   | 12  |  
| E2  | P2  | Analyst   | 6   |  
| E3  | P3  | Consultant| 10  |  
| E9  | P5  | Engineer  | 14  |  
+-----+-----+-----+-----+  
4 rows in set (0.00 sec)  
  
mysql> SELECT * FROM med;  
+-----+-----+-----+-----+  
| eno | pno | resp      | dur |  
+-----+-----+-----+-----+  
| E2  | P1  | Analyst   | 24  |  
| E4  | P2  | Programmer| 18  |  
| E5  | P2  | Manager   | 24  |  
| E10 | P6  | Enginner  | 23  |  
+-----+-----+-----+-----+  
4 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM max;
+-----+-----+-----+-----+
| eno | pno | resp      | dur |
+-----+-----+-----+-----+
| E3  | P4  | Enginner   | 48  |
| E6  | P4  | Manager    | 48  |
| E7  | P3  | Engineer   | 36  |
| E8  | P3  | Manager    | 40  |
| E11 | P5  | Programmer  | 25  |
| E12 | P6  | Programmer  | 300 |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Podemos ver que también funciona correctamente.

Fragmentando verticalmente la tabla PROJ

Ahora pasamos a la fragmentación vertical, que no es más que separar por columnas una tabla, pueden ser una o varias, por el momento solo vamos a crear dos fragmentaciones verticales o vistas que contengan el nombre del proyecto y el presupuesto.

```
gerardo@debian: ~
mysql> SELECT * FROM proj;
+-----+-----+-----+
| pno | pname                                | budget |
+-----+-----+-----+
| P1  | Instrumentation                     | 150000 |
| P2  | Database Develop.                   | 135000 |
| P3  | CAD/CAM                             | 250000 |
| P4  | Maintenance                         | 310000 |
| P5  | Wep Page Design                     | 120000 |
| P6  | Structured Netwok Development       | 350000 |
+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Para ello hacemos lo siguiente:

```
gerardo@debian: ~
mysql> CREATE VIEW pname AS SELECT pname FROM proj;
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE VIEW budget AS SELECT budget FROM proj;
Query OK, 0 rows affected (0.02 sec)

mysql>
```

Verificando el contenido de las fragmentaciones verticales de PROJ

```
gerardo@debian: ~  
mysql> SELECT * FROM pname;  
+-----+  
| pname |  
+-----+  
| Instrumentation |  
| Database Develop. |  
| CAD/CAM |  
| Maintenance |  
| Web Page Design |  
| Structured Network Development |  
+-----+  
6 rows in set (0.00 sec)  
  
mysql> SELECT * FROM budget;  
+-----+  
| budget |  
+-----+  
| 150000 |  
| 135000 |  
| 250000 |  
| 310000 |  
| 120000 |  
| 350000 |  
+-----+  
6 rows in set (0.00 sec)  
  
mysql>
```

Aquí esta como lo fragmentamos verticalmente, pero nos topamos con un pequeño problema.

Comentado el problema

Al hacer fragmentación vertical, podemos encontrar nos con un pequeño problema, como podemos observar la salida anterior donde verificamos que se realizara bien la fragmentación, la información se lista correctamente, pero ahora entramos a un problema a la hora de querer consultar la información, por ejemplo si quiero saber cual es el presupuesto del proyecto con nombre "CAD/CAM", entramos a un problema ya que no tenemos algo conciso de como referenciar información de una lado a otro, ya que puede que la salida en algunas consultas no me los muestre en el orden adecuado o que nos muestren menos salidas por lo que la posición no es una buena manera de referenciar o guiarse para buscar esa información.

Lo que podemos hacer para resolver este problema es arrastra la llave primaria a cada una de las tablas fragmentadas, para así poder saber a que proyecto pertenece que cosa. Con esa referencia ahora podemos ver que el proyecto con nombre "CAD/CAM" tiene la llave primaria "P3" y que el presupuesto con llave primaria "P3" corresponde a ese proyecto.

Sugerencia de cambios

Por el problema anterior nosotros sugerimos que se agregue un segundo campo en cada fragmentación donde se almacene la llave primaria a la cual pertenece, para ello ejecutamos lo siguiente:

```
gerardo@debian: ~  
mysql> CREATE OR REPLACE VIEW pname AS SELECT pno, pname FROM proj;  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> CREATE OR REPLACE VIEW budget AS SELECT pno, budget FROM proj;  
Query OK, 0 rows affected (0.02 sec)  
  
mysql> 
```

Por lo que ahora las salidas serían las siguientes:

```
gerardo@debian: ~  
mysql> SELECT * FROM pname;  
+-----+-----+  
| pno | pname |  
+-----+-----+  
| P1 | Instrumentation |  
| P2 | Database Develop. |  
| P3 | CAD/CAM |  
| P4 | Maintenance |  
| P5 | Web Page Design |  
| P6 | Structured Netwok Development |  
+-----+-----+  
6 rows in set (0.01 sec)  
  
mysql> SELECT * FROM budget;  
+-----+-----+  
| pno | budget |  
+-----+-----+  
| P1 | 150000 |  
| P2 | 135000 |  
| P3 | 250000 |  
| P4 | 310000 |  
| P5 | 120000 |  
| P6 | 350000 |  
+-----+-----+  
6 rows in set (0.00 sec)  
  
mysql> 
```

Así ya podemos realizar consultas de forma adecuada.

Insertando nuevos datos en PROJ

```
gerardo@debian: ~  
mysql> INSERT INTO proj(pno, pname, budget) values  
-> ('P7', 'Strong Legend', 1000000),  
-> ('P8', 'Btn SAN', 500000);  
Query OK, 2 rows affected (0.01 sec)  
Records: 2 Duplicates: 0 Warnings: 0  
  
mysql> SELECT * FROM proj;  
+-----+-----+-----+  
| pno | pname | budget |  
+-----+-----+-----+  
| P1 | Instrumentation | 150000 |  
| P2 | Database Develop. | 135000 |  
| P3 | CAD/CAM | 250000 |  
| P4 | Maintenance | 310000 |  
| P5 | Wep Page Design | 120000 |  
| P6 | Structured Netwok Development | 350000 |  
| P7 | Strong Legend | 1000000 |  
| P8 | Btn SAN | 500000 |  
+-----+-----+-----+  
8 rows in set (0.00 sec)  
  
mysql>
```

Mostrando el contenido de la fragmentación vertical de PROJ

```
gerardo@debian: ~  
mysql> SELECT * FROM pname;  
+-----+-----+  
| pno | pname |  
+-----+-----+  
| P1 | Instrumentation |  
| P2 | Database Develop. |  
| P3 | CAD/CAM |  
| P4 | Maintenance |  
| P5 | Wep Page Design |  
| P6 | Structured Netwok Development |  
| P7 | Strong Legend |  
| P8 | Btn SAN |  
+-----+-----+  
8 rows in set (0.00 sec)  
  
mysql> SELECT * FROM budget;  
+-----+-----+  
| pno | budget |  
+-----+-----+  
| P1 | 150000 |  
| P2 | 135000 |  
| P3 | 250000 |  
| P4 | 310000 |  
| P5 | 120000 |  
| P6 | 350000 |  
| P7 | 1000000 |  
| P8 | 500000 |  
+-----+-----+  
8 rows in set (0.01 sec)  
  
mysql>
```


Un cambio considerable es que ahora por cada fragmentación no incrementa solo alguno de las nuevas tuplas introducidas si no que muestra la cantidad de tuplas que se introdujeron tanto en la primera como en la segunda fragmentación, obviamente en cada una mostrando los datos preestablecidos.

Conclusión

La fragmentación es una herramienta útil si la vemos del lado del almacenamiento físico, ya que existen varios lugares donde se guardan fragmentos de una tabla mucho más grande y si se trata de hacer más eficientes las consultas, por su puesto que lo hace ya que podemos establecer parámetros dependiendo de las entradas, de donde buscar y ya no es necesario buscar en una enorme tabla con todas las tuplas. Esto cuando hablamos de la fragmentación horizontal.

De la fragmentación vertical ya vimos cual sería uno de sus mayores problemas y me parece que en una implementación real esto seria muy ineficiente ya que se esta duplicando una columna por cada fragmentación vertical, realmente desconozco si así lo maneja, pero por lo menos en este laboratorio nos pudimos dar cuenta de eso.

El utilizar vistas es una buena simulación ya que nos permite trabajar casi de una manera idéntica a como trabajaríamos con fragmentación real, ya que cuando cargamos un nuevo dato, no se tiene que indicar explícitamente a que tabla (fragmentada) o servidor de la base de datos distribuida va a pertenecer, pues recordemos que es una de las principales ventajas de las base de datos distribuidas, que aunque estén localizadas a miles de kilómetros de distancia e incluso no se utilice un manejador estándar, nos permite ver una base de datos como si la trabajáramos localmente.

En conclusión, la fragmentación puede ser muy útil en algunas implementaciones y en otras no tanto, por ineficiencia, costos y posiblemente para alguien que no conozca mucho de redes, sea algo difícil de comprender e implementar, ya que hace uso de una combinación de tecnologías de sistemas distribuidos, redes y bases de datos.