



PRÁCTICA 6 - OPTIMIZACIÓN

Adrián Alonso Ridao



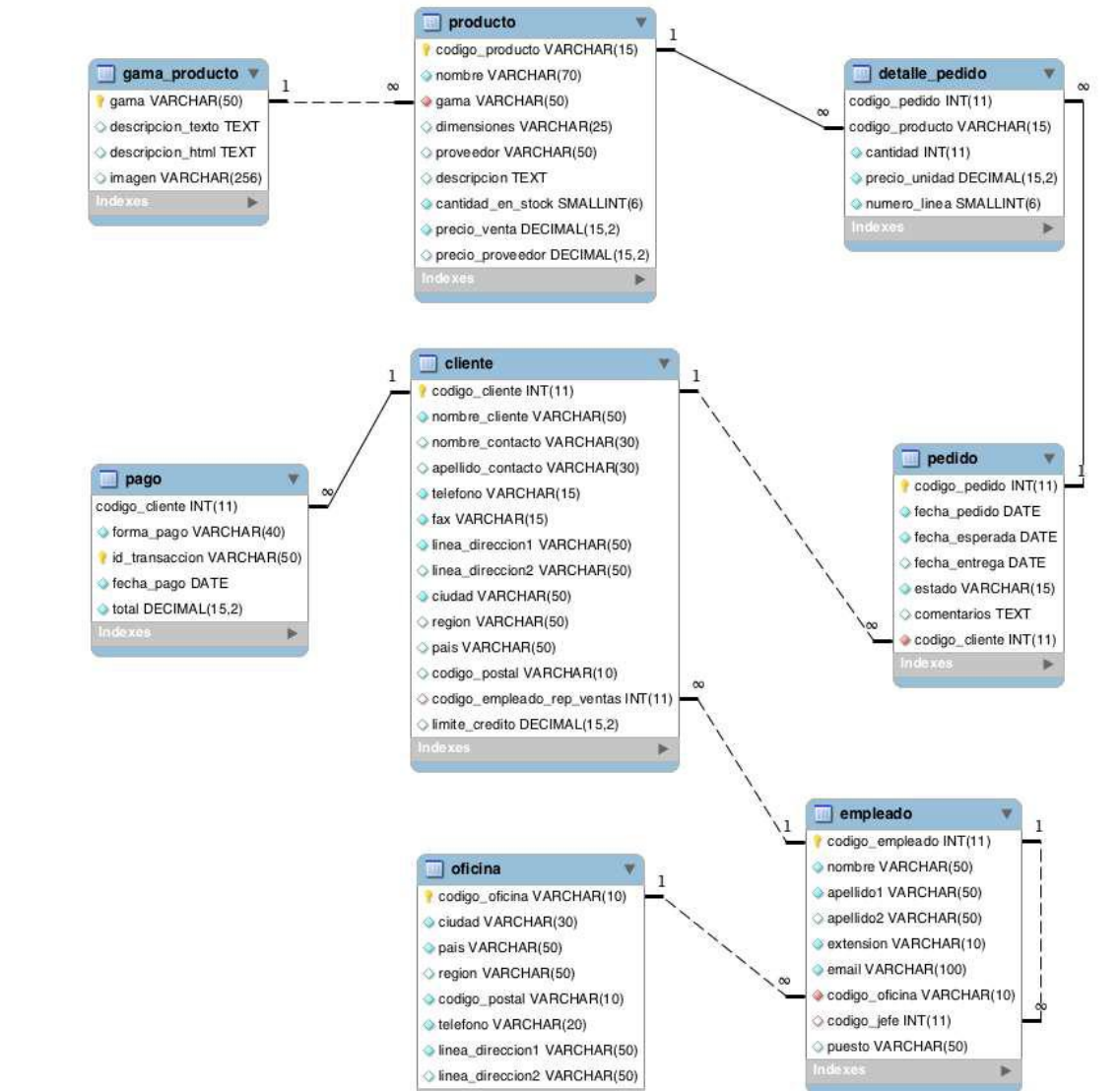
2º ASIR
1ª EVALUACIÓN
ADMINISTRACIÓN DE BASE DE DATOS

Contenido

| | |
|-------------------------------|---|
| Diagrama ER Clientes.sql..... | 2 |
| Ejercicios a resolver | 3 |

Diagrama ER Clientes.sql

Usaremos la BBDD de Clientes.sql:



Ejercicios a resolver

1. Consulte cuáles son los índices que hay en la tabla **producto** utilizando las instrucciones SQL que nos permiten obtener esta información de la tabla.

```
mysql> SHOW INDEX FROM producto;
```

| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
|----------|------------|----------|--------------|-----------------|-----------|-------------|----------|--------|------|------------|---------|---------------|---------|------------|
| producto | 0 | PRIMARY | 1 | codigo_producto | A | 276 | | | | BTREE | | | YES | NULL |
| producto | 1 | gama | 1 | gama | A | | | | | BTREE | | | YES | NULL |

2 rows in set (0.00 sec)

2. Haga uso de **EXPLAIN** para obtener información sobre cómo se están realizando las consultas y diga cuál de las dos consultas realizará menos comparaciones para encontrar el producto que estamos buscando. ¿Cuántas comparaciones se realizan en cada caso? ¿Por qué?

```
SELECT *
FROM producto
WHERE codigo_producto = 'OR-114';
```

```
SELECT *
FROM producto
WHERE nombre = 'Evonimus Pulchellus';
```

```
mysql> EXPLAIN SELECT * FROM producto
-> WHERE codigo_producto = 'OR-114';
```

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|----|-------------|-------|------------|------|---------------|------|---------|------|------|----------|--------------------------------|
| 1 | SIMPLE | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | no matching row in const table |

1 row in set, 1 warning (0.01 sec)

```
mysql> EXPLAIN SELECT * FROM producto
-> WHERE nombre = 'Evonimus Pulchellus';
```

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|----|-------------|----------|------------|------|---------------|------|---------|------|------|----------|-------------|
| 1 | SIMPLE | producto | NULL | ALL | NULL | NULL | NULL | NULL | 276 | 10.00 | Using where |

1 row in set, 1 warning (0.00 sec)

La primera consulta realiza menos comparaciones que la segunda, en la primera consulta se realizan **0** comparaciones y en la segunda consulta se hacen **276** comparaciones. La primera consulta usamos **código_producto** que usa un **índice**, lo cual es mucho más rápido a la hora de buscar un valor. Sin embargo, en la segunda consulta, al usar **nombre** este no es un índice, lo cual la consulta tiene que ir por todas las filas de la base de datos que en total son 276, comparando **'Evonimus Pulchellus'** con los nombres de cada fila.

3. Suponga que estamos trabajando con la base de datos clientes y queremos saber optimizar las siguientes consultas. ¿Cuál de las dos sería más eficiente? Se recomienda hacer uso de EXPLAIN para obtener información sobre cómo se están realizando las consultas.

```
SELECT AVG(total)
FROM pago
WHERE YEAR(fecha_pago) = 2008;
```

```
SELECT AVG(total)
FROM pago
WHERE fecha_pago >= '2008-01-01' AND fecha_pago
<= '2008-12-31';
```

```
mysql> EXPLAIN SELECT AVG(total) FROM pago
-> WHERE YEAR(FECHA_PAGO) = 2008;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | pago | NULL | ALL | NULL | NULL | NULL | NULL | 26 | 100.00 | Using where |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.04 sec)

mysql> EXPLAIN SELECT AVG(total) FROM pago
-> WHERE fecha_pago >= '2008-01-01'
-> AND fecha_pago <= '2008-12-31';
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | pago | NULL | ALL | NULL | NULL | NULL | NULL | 26 | 11.11 | Using where |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.03 sec)
```

Si nos fijamos en la columna **filtered** podemos ver que en la primera consulta es **100,00**, eso significa que la primera consulta tendrá que revisar todos los datos para dar con lo que busca nuestra consulta. En la segunda consulta, **filtered** tiene **11,11**, eso significa que la base de datos sabe que el equivalente a nuestra consulta en ella es un 11%, por lo tanto, el rango de búsqueda es menor. En conclusión, la **segunda consulta** será más eficiente.

4. Optimiza la siguiente consulta creando índices cuando sea necesario. Se recomienda hacer uso de EXPLAIN para obtener información sobre cómo se están realizando las consultas.

```
SELECT *
FROM cliente INNER JOIN pedido
ON cliente.codigo_cliente = pedido.codigo_cliente
WHERE cliente.nombre_cliente LIKE 'A%';
```

```
mysql> EXPLAIN SELECT * FROM cliente INNER JOIN pedido
-> ON cliente.codigo_cliente = pedido.codigo_cliente
-> WHERE cliente.nombre_cliente LIKE 'A%';
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | cliente | NULL | ALL | PRIMARY | NULL | NULL | NULL | 36 | 11.11 | Using where |
| 1 | SIMPLE | pedido | NULL | ref | codigo_cliente | codigo_cliente | 4 | clientes.cliente.codigo_cliente | 6 | 100.00 | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.03 sec)
```

El problema que encontramos es que en la consulta al mirar la tabla **cliente** tiene de tipo **ALL** eso significa que para encontrar clientes que empiecen por 'A' MySQL está escaneando toda la tabla. Al fijarnos en la columna **key** vemos que pone **NULL** lo que nos indica que no está usando ningún índice.

Vamos a optimizar esa parte creando un índice en la columna (nombre_cliente):

```
mysql> CREATE INDEX idx_nombre_cliente ON cliente (nombre_cliente);
Query OK, 0 rows affected (0.09 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Vamos a Ejecutar el EXPLAIN una vez mas y comparar resultados:

```
mysql> EXPLAIN SELECT * FROM cliente INNER JOIN pedido
-> ON cliente.codigo_cliente = pedido.codigo_cliente
-> WHERE cliente.nombre_cliente LIKE 'AA';
```

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|----|-------------|---------|------------|-------|-----------------------------|--------------------|---------|---------------------------------|------|----------|-----------------------|
| 1 | SIMPLE | cliente | NULL | range | PRIMARY, idx_nombre_cliente | idx_nombre_cliente | 282 | NULL | 3 | 100.00 | Using index condition |
| 1 | SIMPLE | pedido | NULL | ref | codigo_cliente | codigo_cliente | 4 | clientes.cliente.codigo_cliente | 6 | 100.00 | NULL |

2 rows in set, 1 warning (0.04 sec)

Como podemos observar, la consulta ha pasado de leer 36 líneas para buscar la consulta en cliente a leer tan solo 3 líneas.

5. Crea un índice de tipo INDEX compuesto por las columnas apellido_contacto y nombre contacto de la tabla cliente. Una vez creado el índice del ejercicio anterior realice las siguientes consultas haciendo uso de EXPLAIN:

```
mysql> CREATE INDEX idx_ap_nom_contacto ON cliente (apellido_contacto, nombre_contacto);
Query OK, 0 rows affected (0.05 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

- a. Busca al cliente Javier Villar. ¿Cuántas filas se han examinado hasta encontrar el resultado?

```
mysql> SELECT * FROM cliente
-> WHERE nombre_contacto = 'Javier'
-> AND apellido_contacto = 'Villar';
```

| codigo_cliente | nombre_cliente | nombre_contacto | apellido_contacto | telefono | fax | linea_direccion1 | linea_direccion2 | ciudad | region | pais | codigo_postal | codigo_empleado_rep_ventas |
|----------------|------------------|-----------------|-------------------|-----------|-----------|------------------|------------------|--------|--------|-------|---------------|----------------------------|
| 15 | Jardin de Flores | Javier | Villar | 654865643 | 914538776 | C/ Olla 34 | NULL | Madrid | Madrid | Spain | 28950 | 39 |

1 row in set (0.00 sec)

```
mysql> EXPLAIN SELECT * FROM cliente
-> WHERE nombre_contacto = 'Javier'
-> AND apellido_contacto = 'Villar';
```

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|----|-------------|---------|------------|------|---------------------|---------------------|---------|-------------|------|----------|-------|
| 1 | SIMPLE | cliente | NULL | ref | idx_ap_nom_contacto | idx_ap_nom_contacto | 246 | const,const | 1 | 100.00 | NULL |

1 row in set, 1 warning (0.00 sec)

Se ha examinado una sola fila.

- b. Busca el cliente anterior utilizando solamente el apellido Villar. ¿Cuántas filas se han examinado hasta encontrar el resultado?

```
mysql> SELECT * FROM cliente
-> WHERE apellido_contacto = 'Villar';
```

| codigo_cliente | nombre_cliente | nombre_contacto | apellido_contacto | telefono | fax | linea_direccion1 | linea_direccion2 | ciudad | region | pais | codigo_postal | codigo_empleado_rep_ventas |
|----------------|------------------|-----------------|-------------------|-----------|-----------|------------------|------------------|--------|--------|-------|---------------|----------------------------|
| 15 | Jardin de Flores | Javier | Villar | 654865643 | 914538776 | C/ Olla 34 | NULL | Madrid | Madrid | Spain | 28950 | 39 |

1 row in set (0.00 sec)

```
mysql> EXPLAIN SELECT * FROM cliente
-> WHERE apellido_contacto = 'Villar';
```

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|----|-------------|---------|------------|------|---------------------|---------------------|---------|-------|------|----------|-------|
| 1 | SIMPLE | cliente | NULL | ref | idx_ap_nom_contacto | idx_ap_nom_contacto | 123 | const | 1 | 100.00 | NULL |

1 row in set, 1 warning (0.00 sec)

Se ha examinado una sola fila.

- c. Busca el cliente anterior utilizando solamente el nombre Javier. ¿Cuántas filas se han examinado hasta encontrar el resultado? ¿Qué ha ocurrido en este caso?

```
mysql> SELECT * FROM cliente
-> WHERE nombre_contacto = 'Javier';
```

| codigo_cliente | nombre_cliente | nombre_contacto | apellido_contacto | telefono | fax | linea_direccion1 | linea_direccion2 | ciudad | region | pais | codigo_postal | codigo_empleado_rep_ventas |
|----------------|------------------|-----------------|-------------------|-----------|-----------|------------------|------------------|--------|--------|-------|---------------|----------------------------|
| 15 | Jardin de Flores | Javier | Villar | 654865643 | 914538776 | C/ Olla 34 | NULL | Madrid | Madrid | Spain | 28950 | 30 |

```
1 row in set (0.00 sec)
```

```
mysql> EXPLAIN SELECT * FROM cliente
-> WHERE nombre_contacto = 'Javier';
```

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|----|-------------|---------|------------|------|---------------|------|---------|------|------|----------|-------------|
| 1 | SIMPLE | cliente | NULL | ALL | NULL | NULL | NULL | NULL | 36 | 10.00 | Using where |

```
1 row in set, 1 warning (0.00 sec)
```

Se han examinado **36** filas. Ha examinado todas las filas, esto se debe a que a la hora de crear nuestro índice hemos indicado que apellido_contacto sea el primero en ser buscado y luego nombre_contacto, esto hace que si al hacer una búsqueda solo de nombre_contacto inutilice el índice creado ya que nuestro índice empezaría a buscar primero por apellido_contacto y luego por nombre_contacto.