



Invoice

Invoice_id
Customer_id
Order_id
Product_id
Date_time
Status
Total
Remark

UT7. OPTIMIZACIÓN DE CONSULTAS E ÍNDICES.

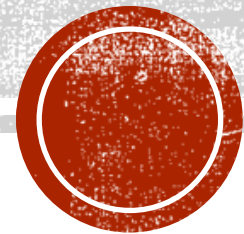
Módulo: BASES DE DATOS

Curso 2022/2023. 1º DAM

Ruth Lospitao Ruiz



INTRODUCCIÓN A LOS ÍNDICES



CONCEPTO

Un índice es un puntero a una fila de una determinada tabla de nuestra base de datos.

Esto, ¿qué significa?

- Un puntero no es más que una referencia que asocia el valor de una determinada columna (o el conjunto de valores de una serie de columnas) con las filas que contienen ese valor (o valores) en las columnas que componen el puntero.



CONSIDERACIONES

- Si quisiéramos buscar un valor específico en la columna de una tabla y la columna sobre la que queremos buscar no tuviese un índice, tendríamos que recorrer toda la tabla comparando fila a fila hasta encontrar el valor que coincide con el valor buscado.
- Para tablas con pocas filas puede que esto no sea un problema, pero imagina las operaciones de comparación que tendría que realizar sobre una tabla con millones de filas.

Animales			
Código	Nombre	Tipo	Propietario
1	Cloncho	Gato	51993482Y
2	Yoda	Gato	51993482Y
3	Sprocket	perro	37276317Z

Duration			(mm)			(mm)	(%)
10 min	1	Cuevas de Nerja, Málaga	41.6	21 Set 2007	N/A	N/A	N/A
20 min	1	Cuevas de Nerja, Málaga	74.2	21 Set 2007	Romania	206	36.0
30 min	2	Sineu, Islas Baleares	87.8	12 Oct 2012	China	280	31.4
60 min	3	Santa Cruz de Tenerife	129.9	31 Mar 2002	China	401	32.4
2 hours	4	San Sebastian, Guipuzcoa	193.0	1 Jun 1997	China	489	39.5
3 hours	4	San Sebastian, Guipuzcoa	204.7	1 Jun 1997	USA	724	28.3
4 hours	5	Huercal-Overa, Almeria	216.3	28 Sep 2012	N/A	N/A	N/A
5 hours	5	Huercal-Overa, Almeria	248.3	28 Sep 2012	N/A	N/A	N/A
6 hours	5	Huercal-Overa, Almeria	275.0	28 Sep 2012	China	840	32.7
9 hours	6	Oliva, Valencia	306.4*	3 Nov 1987	La Reunion	1087	28.2
12 hours	6	Oliva, Valencia	408.5*	3 Nov 1987	N/A	N/A	N/A
18 hours	6	Oliva, Valencia	612.8*	3 Nov 1987	La Reunion	1589	38.6
1 day	6	Oliva, Valencia	817.0	3 Nov 1987	La Reunion	1825	44.8
2 days	7	Javea, Alicante	878.0	1-2 Oct 1957	India	2493	35.2
3 days	7	Javea, Alicante	978.0	1-3 Oct 1957	La Reunion	3929	24.9
4 days	7	Javea, Alicante	978.0	1-3 Oct 1957	La Reunion	4869	20.1
5 days	7	Javea, Alicante	978.0	1-3 Oct 1957	La Reunion	4979	19.6
6 days	8	Sauces, Santa Cruz de Tenerife	984.8	24-29 Feb 1988	La Reunion	5075	19.4
7 days	9	Grazalema, Cadiz	1023.2	14-20 Dec 1958	La Reunion	5400	18.9
8 days	9	Grazalema, Cadiz	1099.2	14-21 Dec 1958	La Reunion	5510	19.9
9 days	9	Grazalema, Cadiz	1226.2	14-22 Dec 1958	La Reunion	5512	22.2
10 days	9	Grazalema, Cadiz	1273.6	13-22 Dec 1958	La Reunion	5678	22.4
11 days	9	Grazalema, Cadiz	1277.2	12-22 Dec 1958	La Reunion	5949	21.5
12 days	9	Grazalema, Cadiz	1280.0	12-23 Dec 1958	La Reunion	5949	21.5
13 days	9	Grazalema, Cadiz	1282.2	11-23 Dec 1958	La Reunion	6072	21.1
14 days	9	Grazalema, Cadiz	1282.2	11-23 Dec 1958	La Reunion	6082	21.1
15 days	9	Grazalema, Cadiz	1284.8	9-23 Dec 1958	La Reunion	6083	21.1
20 days	9	Grazalema, Cadiz	1454.1	3-23 Dec 1958	N/A	NA	N/A
31 days	10	Cortes de la Frontera, Málaga	1674.0	18 Nov - 18 Dec 1989	N/A	N/A	N/A
1 natural month		Caldera Taburiente, Santa Cruz de Tenerife	1626.1	1-31 Jan 1979	India	9300	17.5
2 months	10	Cortes de la Frontera, Málaga	2420.0	Dec 1995 - Jan 96	India	12767	19.0
3 months	11	Casteloais, Ourense	2866.8	Nov 1959 - Jan 60	India	16369	17.5
4 months	11	Casteloais, Ourense	3269.9	Nov 1959 - Feb 60	India	18738	17.5
5 months	12	Casas do Porto, A Coruña	3835.8	Nov 2000 - Mar 01	India	20412	18.8
6 months	12	Casas do Porto, A Coruña	4176.1	Oct 2000 - Mar 01	India	22434	18.6
9 months	12	Casas do Porto, A Coruña	4680.1	Aug 2000 - Apr 01	N/A	N/A	N/A
12 months	12	Casas do Porto, A Coruña	5503.4	Apr 2000 - Mar 01	India	26481	20.8
18 months	13	Dondro, A Coruña	7573.6	Oct 1984 - Mar 86	N/A	N/A	N/A

CONSIDERACIONES

- La mejor forma de optimizar el rendimiento de una consulta es creando índices sobre las columnas que se utilizan en la cláusula WHERE.
- Los índices se comportan como punteros sobre las filas de la tabla y nos permiten determinar rápidamente cuáles son las filas que cumplen la condición de la cláusula WHERE.
- Todos los tipos de datos de MySQL pueden ser indexados
- Ten en cuenta que no es conveniente crear un índice para cada una de las columnas de una tabla
 - El **exceso de índices innecesarios** pueden provocar un incremento del espacio de almacenamiento y un aumento del tiempo para MySQL a la hora de decidir qué índices necesita utilizar.
 - Los índices además añaden una **sobrecarga a las operaciones de inserción, actualización y borrado**, porque cada índice tiene que ser actualizado después de realizar cada una de estas operaciones.
- Debe tratar de **buscar un equilibrio entre el número de índices y el tiempo de respuesta de su consulta**, de modo que pueda reducir el tiempo de respuesta de su consulta utilizando el menor número de índices posible.



¿PARA QUÉ SE USAN?

- MySQL emplea los índices para encontrar las filas que contienen los valores específicos de las columnas empleadas en la consulta de una forma más rápida.
- Si no existiesen índices, MySQL empezaría buscando por la primera fila de la tabla hasta la última buscando aquellas filas que cumplen los valores establecidos para las columnas empleadas en la consulta. Esto implica que, cuanto más filas tenga la tabla, más tiempo tardará en realizar la consulta. En cambio, **si la tabla contiene índices en las columnas empleadas en la consulta**, MySQL tendría una referencia directa hacia los datos sin necesidad de recorrer secuencialmente todos ellos.



¿PARA QUÉ SE USAN?: ACCIONES

- Encontrar las filas que cumplen la condición WHERE de la consulta cuyas columnas estén indexadas.
- Para recuperar las filas de otras tablas cuando se emplean operaciones de tipo JOIN. Para ello, es importante que los índices sean del mismo tipo y tamaño ya que aumentará la eficiencia de la búsqueda.
 - Por ejemplo: una operación de tipo JOIN sobre dos columnas que tengan un índice del tipo INT(10).
- Disminuir el tiempo de ejecución de las consultas con ordenación (ORDER BY) o agrupamiento (GROUP BY) si todas las columnas presentes en los criterios forman parte de un índice.
- Si la consulta emplea una condición simple cuya columna de la condición está indexada, las filas serán recuperadas directamente a partir del índice, sin pasar a consultar la tabla.



TIPOS DE ÍNDICES



TIPOS DE ÍNDICES

- INDEX (NON-UNIQUE)
- UNIQUE
- PRIMARY
- FULLTEXT
- SPATIAL



TIPOS DE ÍNDICES: INDEX (NON-UNIQUE)

- Este tipo de índice se refiere a un índice normal, no único.
- Esto implica que admite valores duplicados para la columna (o columnas) que componen el índice.
- No aplica ninguna restricción especial a los datos de la columna (o columnas) que componen el índice sino que se emplea simplemente para mejorar el tiempo de ejecución de las consultas.



TIPOS DE ÍNDICES: UNIQUE

- Este tipo de índice se refiere a un índice en el que todas las columnas deben tener un valor único.
- Esto implica que no admite valores duplicados para la columna (o columnas) que componen el índice.
- Aplica la restricción de que los datos de la columna (o columnas) deben tener un valor único.



TIPOS DE ÍNDICES: PRIMARY

- Este tipo de índice se refiere a un índice en el que todas las columnas deben tener un valor único (al igual que en el caso del índice UNIQUE) pero con la limitación de que sólo puede existir un índice PRIMARY en cada una de las tablas.
- Aplica la restricción de que los datos de la columna (o columnas) deben tener un valor único.



TIPOS DE ÍNDICES: FULLTEXT

- Estos índices se emplean para realizar búsquedas sobre texto (CHAR, VARCHAR y TEXT).
- Estos índices se componen por todas las palabras que están contenidas en la columna (o columnas) que contienen el índice.
- **No aplica ninguna restricción** especial a los datos de la columna (o columnas) que componen el índice sino que se emplea simplemente para mejorar el tiempo de ejecución de las consultas.
 - Este tipo de índices sólo están soportados por InnoDB y MyISAM en MySQL 5.7.



TIPOS DE ÍNDICES: SPATIAL

- Estos índices se emplean para realizar búsquedas sobre datos que componen formas geométricas representadas en el espacio.
- Este tipo de índices sólo están soportados por InnoDB y MyISAM en MySQL 5.7.



SINTAXIS CREAR ÍNDICE

«CREATE [UNIQUE | FULLTEXT | SPATIAL] INDEX index_name
ON table_name (index_col_name...) index_type;»

- Donde:
 - index_name: es el nombre del índice.
 - table_name: es el nombre de la tabla donde se va a crear el índice.
 - index_col_name: nombre de la columna (o columnas) que formarán el índice.
 - index_type: es el tipo del índice. Se emplea con USING [BTREE | HASH].

Es importante destacar que todos estos índices pueden construirse empleando una o más columnas. Del mismo modo, el orden de las columnas que se especifique al construir el orden es relevante para todos los índices menos para el FULLTEXT (ya que este índice mira en TODAS las columnas que componen el índice).



ÍNDICES DE CLAVE PRIMARIA (PRIMARY)


- Una clave primaria, como ya sabemos, es un índice sobre uno o más campos donde cada valor es único y ninguno de los valores son NULL.
- Para crear un índice de clave primaria tenemos básicamente dos opciones:
 1. Crear el índice de clave primaria al momento de crear la tabla. En este caso se usa, como ya sabemos la opción PRIMARY KEY al final de la definición de los campos, con una lista de los campos que serán parte del índice.
 2. Crear una clave primaria en una tabla existente con el uso del comando ALTER

La sentencia DESC es útil para identificar las columnas que son clave primaria en la tabla

```
mysql> DESC usuarios;
```

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI	0	
nombre	varchar(50)	YES		NULL	
apellidos	varchar(70)	YES		NULL	

3 rows in set (0.00 sec)



ÍNDICES ORDINARIOS (NON-UNIQUE)

- Son índices que no son primarios y permiten valores duplicados (a menos que los campos hayan sido especificados como UNIQUE).
- Para crear un índice ordinario tenemos básicamente tres opciones:

1) Podemos crear un índice ordinario al mismo tiempo que creamos la tabla con el uso de la opción INDEX.

```
CREATE TABLE nombreTabla (campo1 tipoDato, campo2 tipoDato, ...  
INDEX [nombreIndice] (campo1 [, campo2 ...]));
```

2) De igual manera, podemos crear el índice con el uso de la sentencia ALTER TABLE si es que la tabla ya existe.

```
ALTER TABLE nombreTabla ADD INDEX [nombreIndice] (campo1 [, campo2 ...]);
```

3) También es posible usar la sentencia CREATE INDEX para crear un índice en una tabla existente.

```
CREATE INDEX nombreIndice ON nombreTabla (campo1 [, campo2 ...]);
```



ÍNDICES ORDINARIOS (NON-UNIQUE)

- De todas las formas se pide el nombre del índice, sin embargo con la sentencia `CREATE INDEX` el nombre es obligatorio.
- Por ejemplo, para la siguiente definición de tabla:

```
CREATE TABLE usuarios(id int, nombre varchar(50), apellidos varchar(70));
```

- Se puede crear un índice en la columna `apellidos` con una sentencia `ALTER TABLE`:

```
ALTER TABLE usuarios ADD INDEX idx_apellidos (apellidos);
```

- O bien, con una sentencia `CREATE INDEX`:

```
CREATE INDEX idx_apellidos ON usuarios(apellidos);
```



ÍNDICES DE TEXTO COMPLETO (FULL TEXT)

- Los índices de texto completo son del tipo FULLTEXT y pueden contener uno o más campos del tipo CHAR, VARCHAR y TEXT. Un índice de texto completo está diseñado para facilitar y optimizar la búsqueda de palabras clave en tablas que tienen grandes cantidades de información en campos de texto.
- Para crear un índice de texto completo tenemos básicamente las mismas tres opciones:

```
1. CREATE TABLE nombreTabla( campo1 TIPO, campo2 TIPO, FULLTEXT [nombreIndice] (campo1 [campo2, ...]) );
```

```
2. ALTER TABLE nombreTabla ADD FULLTEXT [nombreIndice] (campo1[, campo2, ...]);
```

```
3. CREATE FULLTEXT INDEX nombreIndice ON nombreTabla(campo1[, campo2, ...]);
```



ÍNDICES DE TEXTO COMPLETO

- Como ejemplo consideremos la siguiente definición de tabla:

```
CREATE TABLE usuarios(id int, nombre varchar(50), apellidos varchar(70));
```

- Podríamos crear un índice FULLTEXT en la columna nombre, en la columna apellidos, o bien, un índice que ocupe ambos campos. A continuación se muestran los tres casos.

```
CREATE FULLTEXT INDEX idx_nombre ON usuarios(nombre);
```

```
CREATE FULLTEXT INDEX idx_apellidos ON usuarios(apellidos);
```

```
CREATE FULLTEXT INDEX idx_nombre_apellidos ON  
usuarios(nombre,apellidos);
```



ÍNDICES ÚNICOS (UNIQUE)

- Los índices únicos son básicamente como los índices ordinarios, excepto que los valores duplicados no son permitidos.
- Para crear un índice UNIQUE se tienen de nuevo las mismas tres opciones:

```
1. CREATE TABLE nombreTabla(campo1 tipoDato, campo2 tipoDato,..UNI QUE [nombreIndice]
(campo1 [,campo2...]));
```

```
2. ALTER TABLE nombreTabla ADD UNIQUE [nombreIndice] (campo1,campo2) ...
```

```
3. CREATE UNIQUE INDEX nombreIndice ON nombreTabla(campo1[,campo2...]);
```



ÍNDICES ÚNICOS (UNIQUE)

- Siguiendo con el ejemplo, consideremos de nuevo la siguiente definición de tabla:

```
CREATE TABLE usuarios(id int, nombre varchar(50), apellidos varchar(70));
```

- Podríamos crear un índice UNIQUE en la columna nombre, y un índice UNIQUE en la columna apellidos.

```
ALTER TABLE usuarios ADD UNIQUE idx_nombre (nombre);
```

```
CREATE UNIQUE INDEX idx_apellidos ON usuarios(apellidos);
```

- En el primer caso hacemos uso del comando ALTER TABLE, y en el segundo caso creamos el índice con la sentencia CREATE INDEX.



**ELIMINAR O
MODIFICAR UN
ÍNDICE**



INTRODUCCIÓN

- Algunas veces tendremos la necesidad de cambiar o eliminar un índice. Cuando hagamos algún cambio en el índice, necesitamos eliminar primero el índice y entonces reconstruirlo con la nueva definición
- Para eliminar un índice de clave primaria podemos usar la siguiente sintaxis:

```
ALTER TABLE nombreTabla DROP PRIMARY KEY;
```

- Para eliminar un índice ordinario, único, o de texto completo, necesitamos especificar el nombre del índice y usar esta sintaxis:

```
ALTER TABLE nombreTabla DROP INDEX nombreIndice;
```

- También es válida esta otra sintaxis:

```
DROP INDEX nombreIndice ON nombreTabla;
```

- Si no estamos seguros de cuál es el nombre del índice que deseamos eliminar, podemos hacer uso de la sentencia SHOW KEYS:

```
SHOW KEYS FROM nombreTabla;
```



EJEMPLO

```
CREATE TABLE usuarios
(
id INT NOT,
nombre VARCHAR(50) NOT NULL,
apellidos VARCHAR(70) NOT NULL,
PRIMARY KEY (id),
INDEX (nombre, apellidos)
);
```

```
mysql> SHOW KEYS FROM usuarios;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name
usuarios	0	PRIMARY	1	id
usuarios	1	nombre	1	nombre
usuarios	1	nombre	2	apellidos

```
3 rows in set (0.00 sec)
```

La tercera columna es la que nos proporciona los nombres de los índices. Podemos observar que al no especificar un nombre al índice ordinario en (nombre, apellidos), se le ha asignado el nombre de la primera columna que forma el índice.



EJEMPLO

- A continuación vamos a eliminar los dos índices que existen en esta tabla:

```
mysql> ALTER TABLE usuarios DROP PRIMARY KEY;
```

```
Query OK, 0 rows affected (0.01 sec)  
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> ALTER TABLE usuarios DROP INDEX nombre;  
Query OK, 0 rows affected (0.00 sec)  
Records: 0  Duplicates: 0  Warnings: 0
```

Por último, podemos verificar que estos índices ya no existen:

```
mysql>| SHOW KEYS FROM usuarios;  
Empty set (0.00 sec)
```





Invoice

Invoice_id
Customer_id
Order_id
Product_id
Date_time
Status
Total
Remark

UT7. OPTIMIZACIÓN DE CONSULTAS E ÍNDICES.

Módulo: BASES DE DATOS

Curso 2022/2023. 1º DAM

Ruth Lospitao Ruiz

