# Índices

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

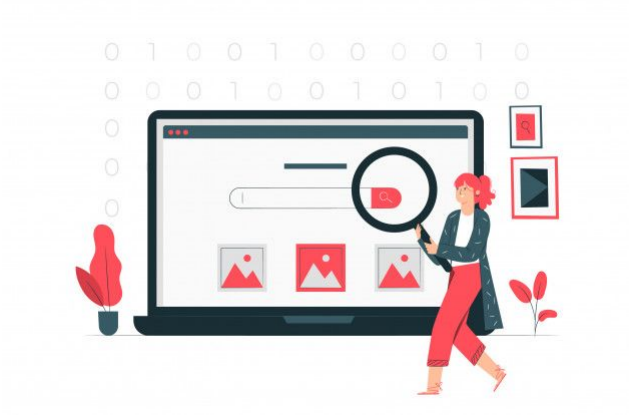
# Índice

# 1 Introducción a índices

# Introducción a índices

Un índice dentro de una base de datos es una **estructura de datos** que mejora la velocidad de las consultas, por medio de un **identificador único** de cada fila de una tabla, permitiendo un rápido acceso a los registros de una tabla en una base de datos.

Pero... **¿qué significa esto?**



Veamos con un ejemplo sobre una tabla de alumnos:

| Id_Alumno (PK) | Nombre | Apellido | Email | Fecha_Nacimiento |
|---|---|---|---|---|
| **1** | Jose | Mentos | jmentos@gmail.com | 29/5/2002 |
| **2** | Laura | Gomez | LauG93@gmail.com | 02/3/1993 |
| **3** | Lucas | Estevanez | EsteLu@gmail.com | 31/3/2000 |

Si buscamos un alumno por su ID, sería fácil porque este es un elemento único, ordenado y estructurado.

## ¿Qué sucede si una aplicación necesita buscar los alumnos por email?

Cada vez que tenga que encontrar un alumno, deberá buscar celda por celda hasta encontrar el email solicitado. Con 3 filas es fácil, pero ¿si tenemos una base de 1 millón de alumnos? ¿Y si la búsqueda se realiza repetidas veces durante el día?

| Id_Alumno (PK) | Nombre | Apellido | Email | Fecha_Nacimiento |
|---|---|---|---|---|
| **1** | Jose | Mentos | jmentos@gmail.com | 29/5/2002 |
| **2** | Laura | Gomez | LauG93@gmail.com | 02/3/1993 |
| **3** | Lucas | Estevanez | EsteLu@gmail.com | 31/3/2000 |

Vamos a tener problemas de performance. Donde cada consulta que realicemos demorará tiempo en responder.

**¡Para solucionar esto se utilizan índices!**

Un índice es una **estructura adicional**, donde elegimos 1 o más columnas que formarán parte del índice. Esto permitirá localizar de forma rápida las filas de la tabla en base a su contenido en la/las columna/s indexada/s.

# Ventajas

- La utilización de índices puede mejorar el rendimiento de las consultas ya que los datos necesarios para satisfacer las necesidades de la consulta existen en el propio índice.
- Pueden mejorar de forma significativa el rendimiento si la consulta contiene agregaciones (GROUP BY), combinaciones (JOINS) de tabla o una mezcla de agregaciones y combinaciones.

# Desventajas

Aunque parece muy bueno el funcionamiento de un índice también tiene sus desventajas.

- Las tablas utilizadas para almacenar los índices ocupan espacio.
- Los índices **consumen recursos** ya que cada vez que se realiza una **operación de actualización, inserción o borrado** en la tabla indexada, se tienen que actualizar todas las tablas de índice definidas sobre ella —en la actualización solo es necesaria la actualización de los índices definidos sobre las columnas que se actualizan—. Por estos motivos no es buena idea definir índices indiscriminadamente.

# Consideraciones

Que tener en cuenta:

- Hay que **evitar crear demasiados índices** en tablas que se actualizan con mucha frecuencia y procurar definirlos con el menor número de columnas posible.
- Es conveniente utilizar un número mayor de índices para mejorar el rendimiento de consultas en tablas con pocas necesidades de actualización, pero con grandes volúmenes de datos.
- Se recomienda utilizar una longitud corta en la clave de los índices agrupados. Los índices agrupados también mejoran si se crean en columnas únicas o que no admiten valores NULL.

# 2 Tipos de índices

Tenemos los tipos de índices: simple, compuesto, agrupado y no agrupado. Veamos un ejemplo de cada uno:

- **Simple:** está definido sobre una sola columna:

```sql
CREATE INDEX "I_LIBROS_AUTOR"
ON "LIBROS" (AUTOR);
```

- **Compuesto:** está formado por varias columnas de la misma tabla.

```sql
CREATE INDEX "I_LIBROS_AUTOREDITORIAL"
ON "LIBROS" (AUTOR,EDITORIAL);
```

- **Único:** los valores deben ser únicos y diferentes. Si intentamos agregar un registro con un valor ya existente, aparece un mensaje de error. A su vez, puede ser simple o compuesto.

```sql
CREATE UNIQUE INDEX "I_LIBROS_AUTOR"
ON "LIBROS" (AUTOR);
```

```sql
CREATE UNIQUE INDEX "I_LIBROS_AUTOREDITORIAL"
ON "LIBROS" (AUTOR,EDITORIAL);
```

# En algunos otros motores SQL

- **Índice agrupado (CLUSTERED):** almacena los datos de las filas en orden. Solo se puede crear un único índice agrupado en una tabla de base de datos. Esto funciona de manera eficiente únicamente si los datos se ordenan en orden creciente o decreciente.

SQL

```sql
CREATE CLUSTERED INDEX "I_LIBROS_AUTOR"
ON "LIBROS" (AUTOR);
```

# En algunos otros motores SQL

- **Índice no agrupado:** organiza los datos de forma aleatoria, pero el índice específica internamente un orden lógico. El orden del índice no es el mismo que el ordenamiento físico de los datos. Los índices no agrupados funcionan bien con tablas donde los datos se modifican con frecuencia y el índice se crea en las columnas utilizadas en orden por las declaraciones WHERE y JOIN.

```sql
CREATE NONCLUSTERED INDEX "I_LIBROS_AUTOR"
ON "LIBROS" (AUTOR);
```

# 3 | Sintaxis

# Sintaxis CREATE INDEX

Con **CREATE INDEX** podemos crear un índice indicando las columnas involucradas:

```sql
CREATE INDEX "NOMBRE_ÍNDICE"
ON "NOMBRE_TABLA" (NOMBRE_COLUMNA);
```

```sql
CREATE INDEX "I_LIBROS_AUTOREDITORIAL"
ON "LIBROS" (AUTOR,EDITORIAL);
```

# Sintaxis DROP INDEX

Con **DROP INDEX** podemos eliminar un índice de una determinada tabla:

```sql
ALTER TABLE "NOMBRE_TABLA"
DROP INDEX "NOMBRE_ÍNDICE";
```

# Sintaxis ANALYZE TABLE

Con **ANALYZE TABLE** analizamos y almacenamos la distribución de claves para una tabla:

```sql
ANALYZE TABLE "NOMBRE_TABLA";
```


DigitalHouse>
Coding School