Laboratorio de Bases de Datos (EBB)

Unidad II – Integridad e Índices

Departamento de Electricidad, Electrónica y Computación
Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán

Primer Cuatrimestre 2020

Introducción [1 | 2]

- Integridad de datos
  - Tipos de integridad
  - Uso de restricciones
  - Elección del método de integridad

Introducción [2 | 2]

- Índices
  - Arquitectura de índices
  - Creación de índices
  - Opciones
  - Consideraciones

- Código: https://github.com/luisenieto/LBD2020.git

Tipos de Integridad [1 | 7]

- La integridad de los datos hace referencia a:
  - La consistencia de los mismos
  - La precisión de los mismos

Tipos de Integridad [2 | 7]

- Tipos de integridad:
  - Integridad de Dominio (o columna)
  - Integridad de Entidad (o tabla)
  - Integridad Referencial

Tipos de Integridad [3 | 7]

- **Integridad de Dominio (de columna):**
  - Especifica los valores de datos válidos para una columna, y si la misma admite o no valores nulos
  - Se fuerza con reglas de validez, restricción de tipos de datos, formato y rango de posibles valores admitidos en una columna

Tipos de Integridad [4 | 7]

- **Integridad de Entidad (de tabla):**
  - Requiere que todas las filas de una tabla tengan un identificador único llamado **clave primaria**
  - Si la clave puede ser cambiada o borrada dependerá del nivel de integridad requerido

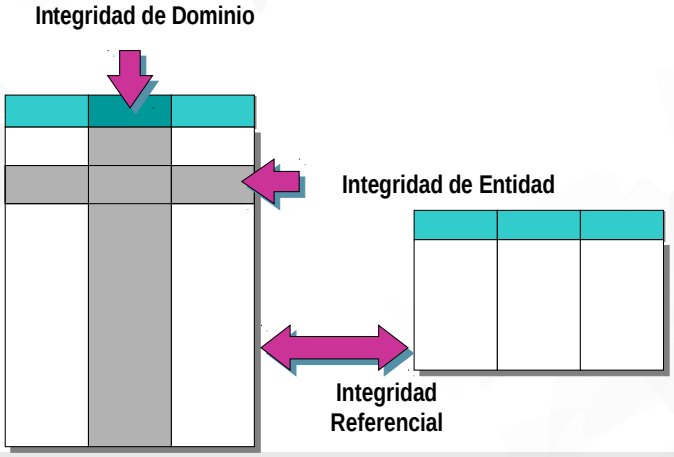
Tipos de Integridad [5 | 7]

- **Integridad Referencial:**
  - Asegura que las relaciones entre las claves primarias (en la tabla de referencia) y las claves propagadas (en la tabla referenciada) siempre se mantengan
  - Se pueden establecer relaciones de integridad referencial en la misma tabla o entre tablas separadas

- No se puede borrar una fila en una tabla de referencia si existe una clave propagada en una tabla referenciada.

Tipos de Integridad [6 | 7]

- Tipos de integridad:



Tipos de Integridad [7 | 7]

- La integridad se datos se puede forzar de forma:
  - **Declarativa**
    - En la definición de los objetos de la BD
    - Forzada automáticamente por el SGBDR
    - Implementada por el uso de restricciones
  - **Procedural**
    - En un *script*, en el cliente y/o en el servidor
    - Implementada por desencadenadores y/o procedimientos almacenados

Uso de restricciones [1 | 32]

- Las restricciones son un método estándar ANSI para forzar la integridad referencial:

Tipo de integridad	Tipo de restricción
Dominio	DEFAULT
	CHECK
	FOREIGN KEY
Entidad	PRIMARY KEY
	UNIQUE
Referencial	FOREIGN KEY
	CHECK

Uso de restricciones [2 | 32]

- **Integridad de Dominio**
  - **DEFAULT**: especifica el valor por defecto de una columna cuando la sentencia `INSERT` no provee uno
  - **CHECK**: especifica los valores aceptados en una columna
  - **FOREIGN KEY**: especifica los valores aceptados en una columna basados en los valores de columnas de otra tabla

Uso de restricciones [3 | 32]

- **Integridad de Entidad**
  - **PRIMARY KEY**: identifica una fila unívocamente en una tabla. Se implementa por medio de índices y no admite valores nulos
  - **UNIQUE**: evita que una columna tenga valores duplicados. Se implementa por medio de índices y sí admite valores nulos

Uso de restricciones [4 | 32]

- **Integridad Referencial**
  - **FOREIGN KEY**: define una columna o combinación de columnas cuyos valores concuerdan con las claves primarias de la misma u otra tabla
  - **CHECK**: especifica los valores aceptados en una columna basados en valores de otras columnas de la misma tabla

Uso de restricciones [5 | 32]

- **Consideraciones:**
  - Se pueden crear restricciones con las sentencias `CREATE TABLE` o `ALTER TABLE`
  - Se pueden agregar restricciones a tablas con datos
  - Se pueden aplicar restricciones a una (restricción a nivel columna) o múltiples columnas (restricción a nivel tabla)

Uso de restricciones [6 | 32]

- **Consideraciones (continuación):**
  - Se pueden crear, modificar y borrar restricciones sin tener que borrar y recrear la tabla
  - Se debe incluir lógica de control de errores en la aplicación para ver si se violó una restricción
  - Los SGBDR verifican los datos existentes cuando se agregan restricciones a una tabla

Uso de restricciones [7 | 32]

- **Consideraciones (continuación):**
  - En MySQL se puede consultar la BD `information_schema` para obtener información sobre las restricciones
  - Generalmente, se deben especificar nombres para las restricciones, y deben ser únicos en la BD

Uso de restricciones [8 | 32]

- **Restricción PRIMARY KEY - Consideraciones:**
  - Solo se permite una clave primaria por tabla (puede ser compuesta)
  - Sus valores deben ser únicos (si la clave es compuesta, la combinación de los valores debe ser única)
  - Las columnas que forman parte de la clave primaria no pueden aceptar valores nulos

Uso de restricciones [9 | 32]

- **Restricción PRIMARY KEY - Consideraciones:**
  - Una clave primaria crea un índice `UNIQUE` con las columnas afectadas como clave del mismo
  - Este índice no se puede borrar, a menos que se borre la clave primaria

Creación/borrado de PK en MySQL e información sobre restricciones

Uso de restricciones [10 | 32]

- **Usar una restricción PRIMARY KEY cuando:**
  - Una o más columnas en una tabla deban identificar unívocamente cada fila (entidad) en la tabla
  - Una de las columnas en la tabla soporte la generación automática de valores

Uso de restricciones [11 | 32]

- **Restricción FOREIGN KEY - Consideraciones:**
  - Debe referenciar una clave primaria o una restricción del tipo `UNIQUE` en la misma u otra tabla
  - Puede proveer referencias a una o varias columnas
  - MySQL crea automáticamente índices de las columnas que forman parte de la clave propagada

Creación/borrado de FK en MySQL e información sobre restricciones

Uso de restricciones [12 | 32]

- **Restricción FOREIGN KEY - Consideraciones (cont):**
  - Cuando una modificación o borrado afecte una clave en la tabla padre, que tenga sus filas correspondientes en la tabla hija, el resultado dependerá de la acción especificada para la acción referencial:
    - `CASCADE`
    - `SET NULL`
    - `RESTRICT` (o `NO ACTION`)
    - `SET DEFAULTción_referencial]`

- Estas acciones referenciales no activan los desencadenadores que hubiera definidos.

Uso de restricciones [13 | 32]

- **Restricción FOREIGN KEY - Consideraciones (cont):**
  - Acción `CASCADE`:
    - Al modificar o borrar la fila en la tabla padre se modifican o borran las filas correspondientes en la tabla hija
    - Se admite esta acción para la modificación y el borrado

- Si entre 2 tablas hay definidas 2 restricciones del tipo **FOREIGN KEY**, de forma tal que ambas sean padre e hija, si se define una cláusula `ON UPDATE CASCADE` u `ON DELETE CASCADE` para una de las claves propagadas, se debe definir otra para la otra clave, ya que de lo contrario se produce un error.

Uso de restricciones [14 | 32]

- **Restricción FOREIGN KEY - Consideraciones (cont):**
  - Acción `SET NULL`:
    - Al modificar o borrar la fila en la tabla padre, las filas correspondientes en la tabla hija toman el valor `NULL` en las columnas de la clave propagada
    - Se admite esta acción para la modificación y el borrado

- Si se especifica esta acción, las columnas de la clave propagada en la tabla hija deben soportar valores nulos.

Uso de restricciones [15 | 32]

- **Restricción FOREIGN KEY - Consideraciones (cont):**
  - Acción `RESTRICT`:
    - Rechaza la modificación o borrado de la fila en la tabla padre
    - Se puede especificar este valor, o bien `NO ACTION`, o bien omitir la cláusula `ON DELETE` u `ON UPDATE`

Uso de restricciones [16 | 32]

- **Restricción FOREIGN KEY - Consideraciones (cont):**
  - Acción `SET DEFAULT`:
    - El analizador (*parser*) de MySQL reconoce esta acción, pero los tipos de tablas InnoDB y NDB rechazan las definiciones de tablas con estas acciones

FK con acción referencial en MySQL

Uso de restricciones [17 | 32]

- **Usar una restricción FOREIGN KEY cuando:**
  - Los datos en una o más columnas puedan contener únicamente valores contenidos en ciertas columnas en la misma u otra tabla
  - Las filas en una tabla no se puedan borrar porque las filas en otra tabla dependan de ellas

Uso de restricciones [18 | 32]

- **Restricción DEFAULT:**
  - Sólo se permite una restricción `DEFAULT` por columna
  - Se aplica sólo a la sentencia `INSERT`
  - No se puede usar con generación automática de valores o con determinados tipos de datos
  - Permite valores provistos por el sistema

Creación/borrado de DEFAULT en MySQL

- Si se intenta agregar más de una restricción DEFAULT a una columna, el último valor que se especifique es el que queda.
- Algunos tipos de datos, como `date`, no permiten usar esta restricción.
- En MySQL hay valores provistos por el sistema. Por ejemplo la función `CURRENT_USER()`.

Uso de restricciones [19 | 32]

- **Usar una restricción DEFAULT cuando:**
  - Los datos guardados en una columna tengan un valor por defecto que resulte obvio
  - La columna no acepte valores nulos
  - La columna no requiera valores únicos

Uso de restricciones [20 | 32]

- **Restricciones CHECK:**
  - Hasta la versión 8.0.15 de MySQL, la sentencia `CREATE TABLE` sólo permite la siguiente versión limitada para una restricción tipo **CHECK** a nivel tabla:

```
CHECK (expr)
```

  - Estas restricciones se pueden definir en una consulta DDL por cuestiones de compatibilidad, pero se ignoran

- Para implementar estas restricciones también se pueden usar vistas o desencadenadores:

- http://www.mysqltutorial.org/mysql-check-constraint/
- https://stackoverflow.com/questions/2115497/check-constraint-in-mysql-is-not-working

Uso de restricciones [21 | 32]

- **Restricciones CHECK (continuación):**
  - Desde la versión 8.0.16 se pueden definir restricciones del tipo **CHECK** a:
    - Nivel tabla:
      - No aparecen en la definición de una columna
      - Pueden referenciar cualquier cantidad de columnas
      - Se permiten referencias a columnas no definidas
    - Nivel columna:
      - Aparecen en la definición de una columna

Uso de restricciones [22 | 32]

- **Restricciones CHECK (continuación):**
  - Ejemplo:

```
CREATE TABLE t1 (
   CHECK (c1 <> c2),
   c1 INT CHECK (c1 > 10),
   c2 INT CONSTRAINT c2_positivo CHECK (c2 > 0),
   c3 INT CHECK (c3 < 100),
   CONSTRAINT c1_nocero CHECK (c1 <> 0),
   CHECK (c1 > c3)
);
```


- La primer restricción es a nivel tabla: ocurre fuera de la definición de una columna, referencia varias columnas (en este caso) y referencia a columnas todavía no definidas (`c1` y `c2`).

- Las siguientes 3 restricciones son a nivel columna: de estas 3 la segunda tiene asignado explícitamente un nombre. Para el caso de las otras 2 MySQL se los genera a partir del nombre de la tabla, luego el literal `_chk_` y luego un número (1, 2, 3, ...).

- Las últimas 2 restricciones son a nivel tabla: la primera tiene asignado explícitamente un nombre y la segunda no.

Uso de restricciones [23 | 32]

- **Restricciones CHECK (continuación):**
  - Forma genérica para definir una restricción tipo **CHECK**:

```
[CONSTRAINT [nombre]] CHECK (expr) [[NOT] ENFORCED]
```

  - `expr`: expresión booleana que debe evaluarse a `TRUE` o `UNKNOWN` (para valores `NULL`) para cada fila de la tabla. Si se evalúa a `FALSE` ocurre una violación de la restricción

Uso de restricciones [24 | 32]

- **Restricciones CHECK (continuación):**
  - Se puede indicar si la restricción es obligatoria o no:
    - `ENFORCED`: se crea la restricción y es obligatoria
    - `NOT ENFORCED`: se crea la restricción, pero no es obligatoria

Uso de restricciones [25 | 32]

- **Restricción CHECK - Consideraciones:**
  - Se pueden usar columnas no generadas y generadas, salvo columnas con el atributo `AUTO_INCREMENT` y columnas en otras tablas
  - Se permiten literales, funciones determinísticas y operadores
  - No se permiten funciones, procedimientos almacenados, variables ni subconsultas

- Para más información sobre columnas generadas, se puede consultar:

- http://www.mysqltutorial.org/mysql-generated-columns/

- Una función es determinística si dado el mismo conjunto de datos múltiples llamadas producen los mismos resultados. Ejemplos de funciones que no son deterministas: `CONNECTION_ID()`, `CURRENT_USER()`, `NOW()`.

Uso de restricciones [26 | 32]

- **Restricción CHECK – Consideraciones (cont):**
  - Se aplica para operaciones `INSERT`, `UPDATE`, `REPLACE`, `LOAD DATA` y `LOAD XML`
  - En columnas con restricciones **CHECK** no se pueden emplear las acciones `ON UPDATE` y `ON DELETE` de las restricciones **FOREIGN KEY**, y viceversa
  - Una columna puede tener múltiples restricciones de este tipo

Creación/borrado de CHECK en MySQL e información sobre restricciones

Uso de restricciones [27 | 32]

- **Usar una restricción CHECK cuando:**
  - La lógica de negocio determine que los datos en una columna deban tener un cierto rango o formato
  - Los datos en una columna tengan ciertos límites en sus posibles valores
  - Existan relaciones entre las columnas de una tabla que restrinjan los valores que puede tener una columna

Uso de restricciones [28 | 32]

- **Restricción UNIQUE:**
  - Se implementa con índices tipo `UNIQUE`
  - Permite valores nulos (uno solo)
  - Una tabla puede tener varias restricciones `UNIQUE`
  - Se puede definir en una o en un conjunto de columnas

Creación/borrado de UNIQUE en MySQL e información sobre restricciones

Uso de restricciones [29 | 32]

- **Usar una restricción UNIQUE cuando:**
  - La tabla contenga columnas que no sean parte de la clave primaria y las mismas deban tener valores únicos
  - Las reglas de negocio determinen que los datos guardados en una columna deban ser únicos

Uso de restricciones [30 | 32]

- **Restricciones en tablas con datos:**
  - Cuando se definen restricciones del tipo **CHECK** y **FOREIGN KEY** en una tabla con datos, se verifican los mismos: si no satisfacen las condiciones de las restricciones no se las puede crear
  - Si se sabe que los datos de una tabla cumplen con las condiciones de una restricción, por cuestiones de rendimiento se puede deshabilitar este control al crearla

Uso de restricciones [31 | 32]

- **Restricciones en tablas con datos (continuación):**
  - Si los datos existentes no cumplieran con las condiciones de las restricciones, los mismos se comprobarán al ejecutar una sentencia `UPDATE` (es conveniente cambiarlos antes)

Uso de restricciones [32 | 32]

- **Restricciones en tablas con datos (continuación):**
  - También se pueden deshabilitar al insertar datos:
    - Se debe importar una gran cantidad de datos y sabe que los mismos satisfacen las restricciones
    - Los datos masivos a importar no satisfacen las restricciones (se los importa y luego se los pone en regla antes de volver a habilitar las restricciones)

Restricciones en tablas con datos en MySQL

Elección del método de integridad [1 | 2]

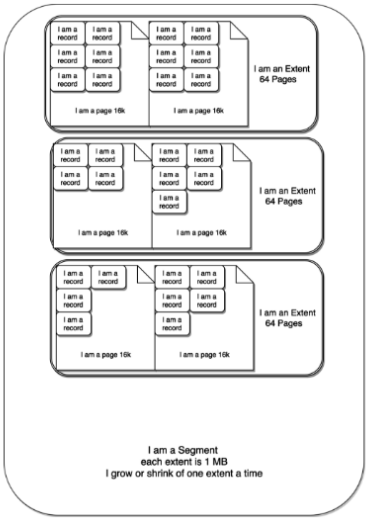
- Se debe considerar la funcionalidad y el costo en tiempo y recursos:
  - Es mejor usar integridad declarativa (restricciones) siempre y cuando el dominio lo permita
  - Si se necesitan operaciones en cascada, usar desencadenadores y procedimientos almacenados

Elección del método de integridad [2 | 2]

Método de integridad	Funcionalidad	Costo	Antes o después de la transacción
Restricciones	Media	Bajo	Antes
Desencadenadores	Alta	Alto	Después

Arquitectura de índices [1 | 33]

- **Almacenamiento en MySQL**
  - Segmentos
  - *Extents*:
    - 1MB de tamaño
    - 64 páginas
  - Páginas:
    - 16KB de tamaño



- Debido al valor 1 del parámetro `innodb_file_per_table`, por cada tabla InnoDB que se crea se tiene un archivo (o varios si la tabla está particionada) dentro de la carpeta correspondiente a la BD.

- Cada uno de estos archivos internamente se divide en segmentos. A su vez, cada segmento se divide internamente en *extents*, y cada *extent* en páginas.

- Las páginas tienen un tamaño predeterminado de 16 KB (pueden tener otro). Si las páginas son del tamaño predeterminado, los *extents* tienen un tamaño fijo de 1 MB (así, un *extent* puede tener hasta 1024/16=64 páginas).

- La cantidad de filas que puede contener una página depende del tamaño de la fila.

Arquitectura de índices [2 | 33]

- **Almacenamiento en MySQL (cont.)**
  - Mediante la opción `innodb_page_size` se puede definir el tamaño de una página. Los tamaños admitidos son 64KB, 32KB, 16KB (predeterminado), 8KB y 4KB
  - En InnoDB, como mínimo deben entrar 2 filas en una página, por lo que se tiene un límite aproximado de 8000 bytes por fila

- En InnoDB se puede definir el tamaño de página para una instancia de MySQL a través de la opción `innodb_page_size` antes de inicializar la instancia de MySQL. Una vez que se define el tamaño de página para una instancia, no se puede cambiarlo sin reinicializar la instancia. Los tamaños admitidos son 64KB, 32KB, 16KB (predeterminado), 8KB y 4KB.

- Una instancia de MySQL que use un tamaño de página InnoDB en particular no puede usar archivos de datos o archivos de registro de una instancia que use un tamaño de página diferente.

Arquitectura de índices [3 | 33]

- Ejemplo:

```
CREATE TABLE Agenda (
  Apellido VARCHAR(50) NOT NULL,
  Nombre VARCHAR(50) NOT NULL,
  Telefono VARCHAR(50) NOT NULL
)
```

Arquitectura de índices [4 | 33]

- Sin un índice, a medida que se ingresan los datos, los mismos se agregan a las páginas donde haya lugar, sin un orden en particular

Arquitectura de índices [5 | 33]

- Para buscar el teléfono de una persona:

```
SELECT Telefono
FROM Agenda
WHERE Apellido = 'Logan' AND Nombre = 'Todd'
```

- Se debe buscar por todas las filas (escaneo de tabla)

- Cuando no hay un índice, y se debe buscar una fila en particular, se deben recorrer todas las filas, no sólo porque no haya un orden, sino también porque incluso si se encuentra una fila que cumpla con el criterio especificado, no se sabe si habrá otras después.

- Cuando para buscar una fila se deben recorrer todas las páginas, se habla de un **escaneo de tabla**.

Arquitectura de índices [6 | 33]

- **Escaneo de tabla (*Table Scan*)**
  - Se recorren desde el comienzo, y secuencialmente, todas las páginas de datos que componen la tabla
  - Se extraen las filas que satisfacen algún criterio
  - Costo: O(n)

Arquitectura de índices [7 | 33]

- En la guía de teléfono:
  - Las entradas están ordenadas por apellido, de A-Z
  - Si hay 2 apellidos iguales se ordenan por nombre, de A-Z
- En terminología de BDs, apellido y nombre serían la clave del índice

Arquitectura de índices [8 | 33]

- **Indice**:
  - Estructura que **a veces** puede acelerar el acceso a los datos, con el costo de escrituras y espacio de almacenamiento adicional para su mantenimiento

Arquitectura de índices [9 | 33]

- Al recorrer una tabla cuando hay un índice:
  - Primero se determina si existe o no un índice
  - Luego se determina la mejor forma para acceder a los datos: **recorrer la tabla o emplear un índice**
  - Si se usa un índice, se recorre su estructura para encontrar las filas que satisfacen el criterio de la consulta

- La presencia de un índice no garantiza su uso.

Arquitectura de índices [10 | 33]

- Al recorrer una tabla cuando hay un índice (cont.):
  - Se extraen las filas que satisfacen el criterio de la consulta
  - Costo: $O(\log_{2,3} n)$
  - A la operación de navegar por un índice hasta llegar a los datos se la llama **Búsqueda por Indice** (*Index Seek*)

Arquitectura de índices [11 | 33]

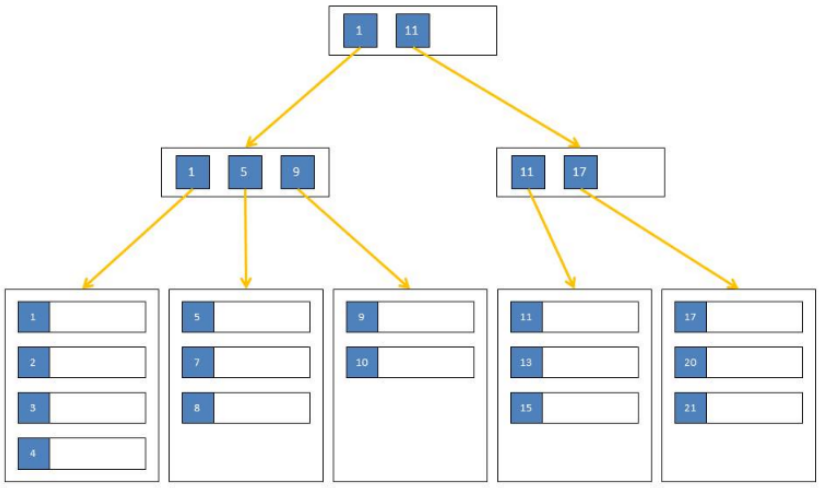
- MySQL organiza los índices en una estructura tipo árbol B
- Tipos de nodos:
  - Los nodos en el nivel más bajo, llamados hojas, tienen los datos
  - Todos los otros nodos, incluyendo a la raíz, sólo tienen las claves del índice y punteros a los nodos con datos

- Un árbol B no es lo mismo que un árbol binario: un árbol binario es un árbol en el cual cada nodo tiene a lo sumo 2 hijos.

- Un árbol binario permite realizar búsquedas en forma muy eficiente. Por otro lado, hacer una modificación suele requerir reorganizar toda la estructura (aún cuando la modificación sea de un único valor).

- Un árbol B es mucho más eficiente que un árbol binario cuando se tienen que hacer modificaciones, pero algunas operaciones siguen resultando costosas dependiendo de la posición del nodo que tendrá el valor nuevo o modificado dentro del árbol.

Arquitectura de índices [12 | 33]



Arquitectura de índices [13 | 33]

- Las tablas y los índices se guardan en páginas (páginas de datos y páginas de índice)
- El índice correcto en la columna (o columnas) correcta, es la base para empezar a optimizar las consultas
- Un índice faltante, o un índice en una columna incorrecta, puede ser el origen de los problemas de rendimiento

Arquitectura de índices [14 | 33]

- **Cuándo crear índices**
  - Cuando se quiera acelerar el acceso a los datos
  - Cuando se necesite forzar la unicidad de las filas (índices `UNIQUE`)
  - Cuando las columnas tengan muchos valores únicos (selectividad alta)

Arquitectura de índices [15 | 33]

- **Cuándo no crear índices**
  - Cuando no se los usa frecuentemente
  - Cuando las columnas tienen muchos valores repetidos (selectividad baja)

Arquitectura de índices [16 | 33]

- **Tipos de índices**
  - Agrupados (*clustered*) o primarios
  - Secundarios

Arquitectura de índices [17 | 33]

- **Indices agrupados**
  - Siguiendo con la agenda, se podrían ordenar físicamente los datos según apellido y nombre:

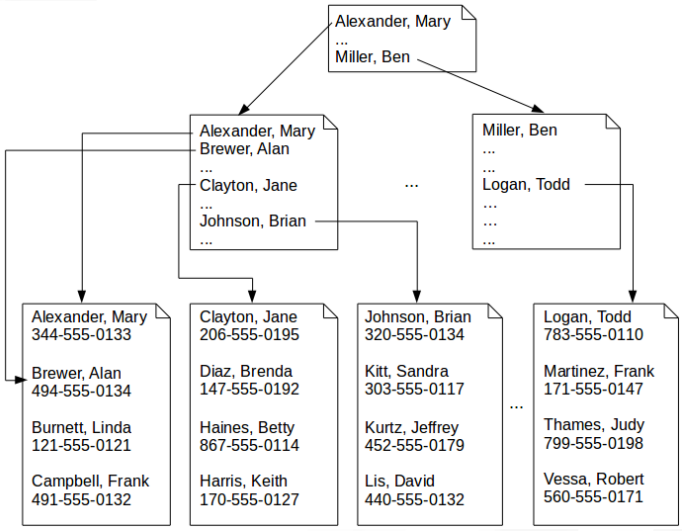
- Esta alternativa solucionaría en parte el problema del escaneo de tabla: ahora se sabe cuando parar de leer las páginas ya que los datos están ordenados.

Arquitectura de índices [18 | 33]

- **Indices agrupados (cont.)**
  - Después de ordenar físicamente los datos se puede armar un conjunto de páginas (de índice) que permita navegar directamente a los mismos

Arquitectura de índices [19 | 33]

Indices agrupados



Arquitectura de índices [20 | 33]

- **Indices agrupados (cont.)**
  - A toda la estructura anterior se la llama **Indice Agrupado** (o *Clustered*)
  - La raíz y nodos intermedios sólo guardan los valores de la clave del índice (ordenados ascendentemente)
  - Las hojas guardan todos los datos (ordenados según la clave del índice)

- En el dibujo anterior sólo se muestra un nivel intermedio, pero este número depende del tamaño de la tabla: un índice grande tendrá más de un nivel intermedio, mientras que un índice chico podría no tener directamente (sólo la raíz y las hojas).
- La raíz sirve para saber qué página de nivel intermedio se debe leer, y la página intermedia sirve para saber qué página de datos leer (suponiendo que haya un único nivel intermedio).
- En MySQL, los índices agrupados también se llaman primarios.

Arquitectura de índices [21 | 33]

- **Indices agrupados (cont.)**
  - Cada fila en las páginas de índice (nodo raíz e intermedios) tiene la clave del índice y un puntero ya sea a una página intermedia en el árbol o a una fila de datos en el nivel hoja
  - Como el índice determina el orden en el que se guardan las filas de la tabla, cada tabla puede tener un único índice agrupado

- En un índice agrupado, los términos “nodo hoja” o “página de datos” se pueden usar de forma indistinta, ya que son los mismos (el nivel hoja de un índice agrupado es donde se guardan los datos, ordenados según la clave del índice).

- En las hojas y nodos intermedios también hay un puntero al nodo anterior y al siguiente, formando una lista doblemente enlazada.

- Como las páginas a nivel hoja y las páginas de datos de la tabla son las mismas, esto implica que las filas de la tabla están ordenadas físicamente según la clave del índice. Como solo puede haber una única forma de ordenar físicamente los datos, una tabla sólo puede tener un único índice agrupado.

Arquitectura de índices [22 | 33]

- **Indices agrupados (cont.)**
  - En MySQL cada tabla InnoDB tiene un índice agrupado, cuya clave **generalmente** es la PK de la tabla
  - Si no hay una PK para la tabla, MySQL localiza el primer índice `UNIQUE` donde todas las columnas de la clave NO sean `NULL` e InnoDB lo usa como índice agrupado

Arquitectura de índices [23 | 33]

- **Indices agrupados (cont.)**
  - Si la tabla no tiene una PK o un índice `UNIQUE` adecuado, InnoDB genera internamente un índice agrupado oculto llamado `GEN_CLUST_INDEX` en una columna que contiene valores con identificadores de filas (las filas se ordenan por los identificadores que InnoDB asigna a las filas)

- El identificador de fila es un campo de 6 bytes que aumenta monótonamente a medida que se insertan nuevas filas. Por lo tanto, las filas ordenadas por este identificador de fila están físicamente en orden de inserción.

Arquitectura de índices [24 | 33]

- **Indices agrupados (cont.)**
  - Conclusión: en MySQL no se puede crear explícitamente un índice agrupado (lo hace MySQL)

Arquitectura de índices [25 | 33]

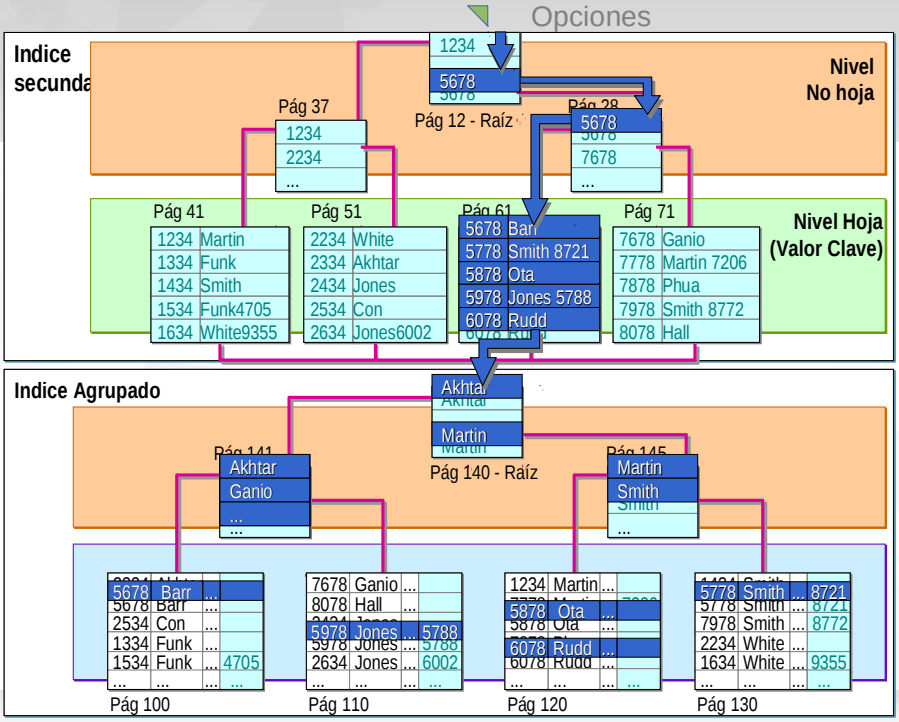
- **Indices secundarios**
  - Tienen la misma estructura de árbol B que los índices agrupados
  - La raíz y nodos intermedios sólo guardan los valores de la clave del índice secundario (ordenados ascendentemente)
  - Las hojas guardan las claves del índice agrupado

Arquitectura de índices [26 | 33]

- **Indices secundarios (cont.)**
  - La búsqueda comienza recorriendo la estructura del índice secundario hasta llegar al nivel hoja
  - Como al llegar a las hojas se recupera la clave del índice agrupado, la búsqueda sigue ahora por el árbol del índice agrupado

Arquitectura de índices [27 | 33]

- **Indices secundarios (cont.)**
  - Los datos y el índice se guardan separados
  - Puede haber varios por tabla


Indice secunda
Nivel No hoja
1234
5678
Pág 12 - Raíz
Pág 37
1234
2234
...
5678
7678
...
Pág 41
Pág 51
Pág 61
Pág 71
Nivel Hoja (Valor Clave)
1234 Martin
1334 Funk
1434 Smith
1534 Funk4705
1634 White9355
2234 White
2334 Akhtar
2434 Jones
2534 Con
2634 Jones6002
5678 Barr
5778 Smith 8721
5878 Ota
5978 Jones 5788
6078 Rudd
7678 Ganio
7778 Martin 7206
7878 Phua
7978 Smith 8772
8078 Hall
Indice Agrupado
Akhtar
Martin
Pág 140 - Raíz
Akhtar
Ganio
...
Martin
Smith
...
5678 Barr ...
2534 Con ...
1334 Funk ...
1534 Funk ... 4705
7678 Ganio ...
8078 Hall ...
5978 Jones ... 5788
2634 Jones ... 6002
1234 Martin ...
5878 Ota ...
6078 Rudd ...
5778 Smith ... 8721
7978 Smith ... 8772
2234 White ...
1634 White ... 9355
Pág 100
Pág 110
Pág 120
Pág 130

Arquitectura de índices [29 | 33]

- **Usar un índice agrupado**
  - **Para recuperar un rango de datos**
    - Como las páginas a nivel hoja son los datos de la tabla, las páginas de índice y las de datos están ordenadas físicamente según la clave del índice
    - Si el ordenamiento físico de las filas de datos coincide con el del resultado de una consulta, se pueden leer todas las filas secuencialmente

Arquitectura de índices [30 | 33]

- **Usar un índice agrupado (cont.)**
  - **Para recuperar datos preordenados**

Arquitectura de índices [31 | 33]

- **Usar un índice secundario**
  - **Cuando se quieren recuperar pocas filas de una tabla grande**
  - A medida que aumenta la cantidad de filas, aumenta proporcionalmente el costo debido al recorrido de las 2 estructuras de índice

- Para recuperar pocas filas, la columna indexada debería tener una selectividad muy alta (muchos valores únicos).

- Un gran costo asociado a los índices secundarios se debe a las excesivas búsquedas de claves, el cual es el mecanismo para navegar desde las filas del índice secundario a la fila correspondiente del índice agrupado.

- Estos saltos desde una estructura de índice a otra son la razón por la cual grandes cantidades de filas se sirven mejor con un índice agrupado.

Arquitectura de índices [32 | 33]

- **NO usar un índice agrupado**
  - En columnas que se modifican frecuentemente
  - Con muchas inserciones secuenciales y concurrentes
  - Con claves grandes
- **NO usar un índice secundario**
  - Cuando se recuperan muchas filas

- Si la PK es grande, los índices secundarios usan más espacio, por lo que es ventajoso tener una PK chica corta.

Arquitectura de índices [33 | 33]

- **Consideraciones de rendimiento:**
  - Crear índices en las claves propagadas
  - Crear índices compuestos ya que el rendimiento de las consultas aumenta especialmente cuando el usuario accede a los datos de varias maneras
  - Crear múltiples índices en una tabla que sea leída con mucha frecuencia de diferentes maneras

Creación de índices [1 | 8]

- **Creación**
  - Se emplea la sentencia `CREATE INDEX`
- **Borrado**
  - Se emplea la sentencia `DROP INDEX`

Creación/borrado de índices en MySQL

Creación de índices [2 | 8]

- Consideraciones al borrar un índice:
  - Se libera el espacio en disco
  - No se pueden borrar índices de PKs o restricciones UNIQUE
  - Cuando se borra una tabla se borran todos sus índices

Creación de índices [3 | 8]

- **Índices UNIQUE:**
  - Aseguran que los datos en las columnas indexadas sean únicos
  - Se crean automáticamente al crear una restricción UNIQUE o PK
  - Si la tabla tiene datos, se chequean valores duplicados
  - Cuando se ejecutan las sentencias `INSERT` y `UPDATE`, se chequean los valores duplicados y se cancela la sentencia

Creación de índices [4 | 8]

- Sintaxis para detectar valores duplicados antes de crear un índice UNIQUE:

```
SELECT <col>, COUNT(<col>)
FROM <tabla>
GROUP BY <col> HAVING COUNT(<col>) > 1
```


- MySQL da un error si se deja un espacio después de `COUNT`.

Creación de índices [5 | 8]

- **Índices UNIQUE (cont.):**
  - En MySQL, 2 o más filas pueden contener el valor NULL (MySQL considera que NULL no es igual a sí mismo). Si se desea evitar esto, no se deben permitir valores nulos en estas columnas

Creación de índices UNIQUE en MySQL

Creación de índices [6 | 8]

- **Índices compuestos:**
  - Crear índices compuestos cuando se solicitan 2 o más columnas para la búsqueda u orden
  - Crear índices compuestos cuando las consultas referencien sólo las columnas del índice

- Cuando una consulta referencia sólo las columnas del índice, se dice que el mismo abarca la consulta.

Creación de índices [7 | 8]

- **Índices compuestos - Consideraciones:**
  - Hasta 16 columnas (MySQL)
  - Todas las columnas deben ser de la misma tabla
  - El orden debe ser de mayor unicidad a menor unicidad
  - Un índice (c1,c2) es diferente a otro (c2,c1)

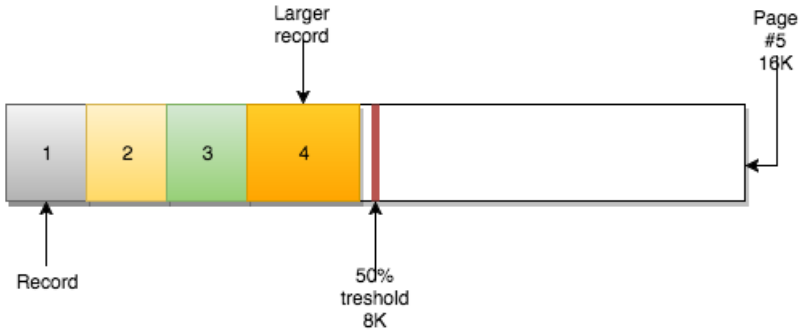
Creación de índices [8 | 8]

- **Índices – Información:**
  - Se emplea la sentencia `SHOW INDEX`

Información de índices en MySQL

Opciones [1 | 18]

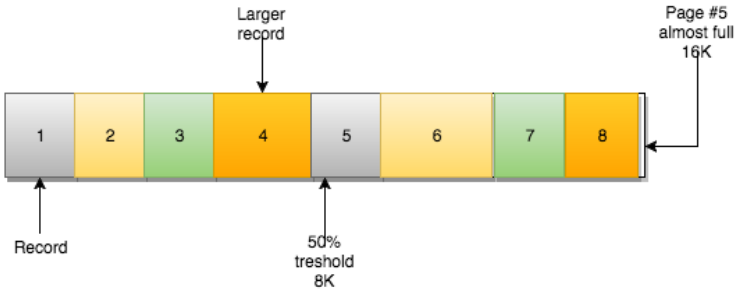
- En MySQL, a medida que se ingresan datos, los mismos se agregan a las páginas del índice agrupado ordenados según su clave



- En el ejemplo, si la tabla usa un auto incremental como PK, se tendrá una secuencia de 1, 2, 3, 4, etc.

Opciones [2 | 18]

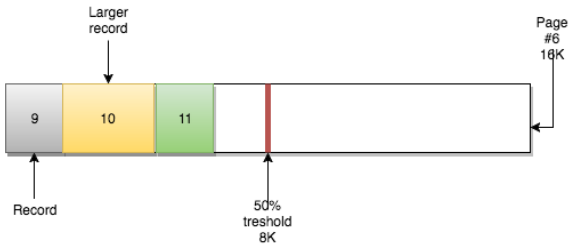
- Una página puede estar vacía o llena (100%)
- Cada página tiene un atributo llamado MERGE_THRESHOLD



- El valor predeterminado del atributo MERGE_THRESHOLD es el 50% del tamaño de la página.

Opciones [3 | 18]

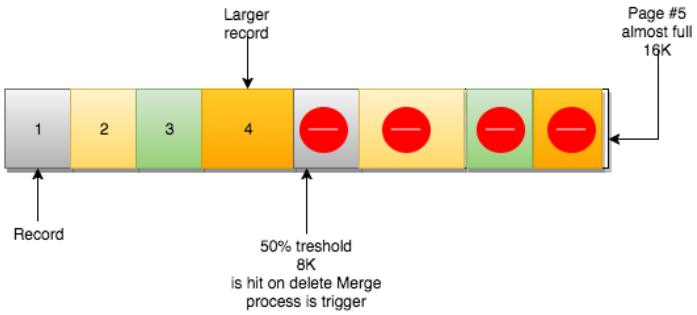
- A medida que se insertan datos, la página se va llenando secuencialmente
- Cuando se llena una página, el siguiente registro se inserta en la siguiente



- Dada la naturaleza de los árboles B, la estructura se puede navegar no sólo desde arriba hacia abajo, sino también horizontalmente a través de las hojas, ya que cada una forma una lista doblemente enlazada permitiendo ir a la página siguiente y a la anterior.

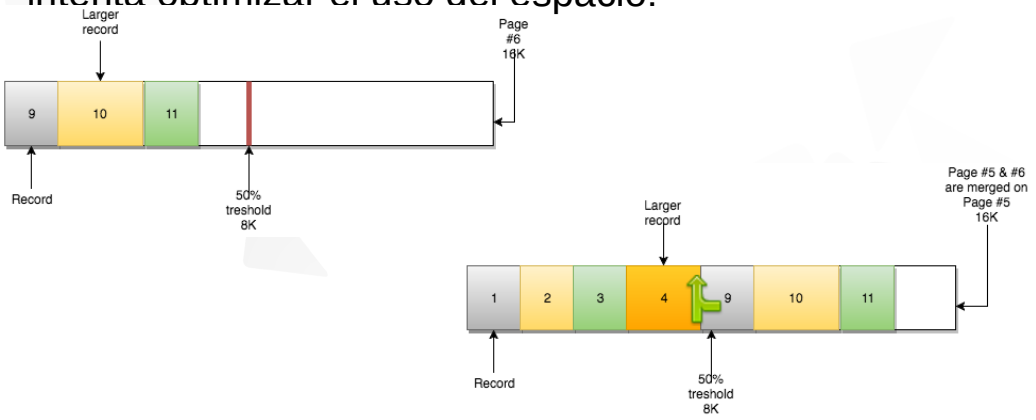
Opciones [4 | 18]

- Cuando se borra un registro, el mismo no se borra físicamente, sino que se marca como borrado y el espacio que ocupaba puede ser usado por otro registro



Opciones [5 | 18]

- Cuando una página tiene muchos borrados, InnoDB intenta optimizar el uso del espacio:



- Cuando una página tiene tantos borrados que coincidan con el valor del MERGE_THRESHOLD (50% del tamaño de la página de forma predeterminada), InnoDB busca en las páginas previa y siguiente para ver si puede optimizar el uso del espacio combinando 2 páginas.

- En este ejemplo, la página #6 está utilizando menos de la mitad de su espacio, y la página #5 recibió muchos borrados y también tiene menos de la mitad de su espacio usado, y por lo tanto se pueden combinar. La combinación resulta en la página #5 con sus datos más los de la página #6, y la página #6 queda vacía, usable para nuevos datos.

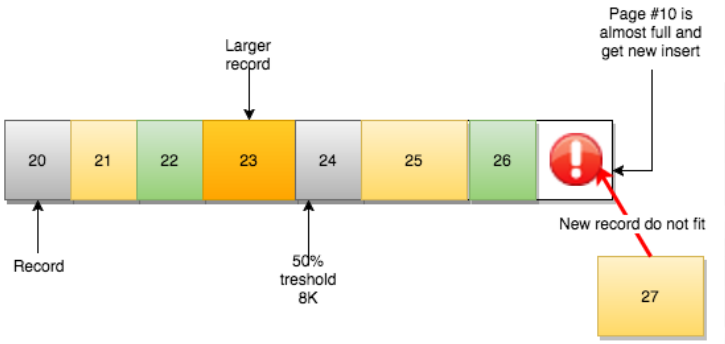
Opciones [6 | 18]

- El mismo proceso se produce cuando se actualiza un registro y su tamaño hace que la página quede debajo del umbral
- Se realiza una combinación cuando se producen borrados y modificaciones que tengan que ver con **páginas enlazadas**

- Si la operación de combinación resulta exitosa, la métrica `index_page_merge_successful` en la BD `INFORMATION_SCHEMA.INNODB_METRICS` se incrementa. El valor de `index_page_merge_successful` se puede consultar en la columna `Name` de la tabla `InnoDB_Metrics`.

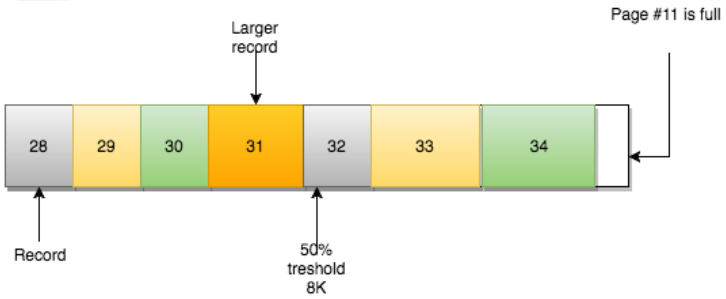
Opciones [7 | 18]

- Si una página está totalmente llena, la siguiente página guarda los nuevos registros. ¿Qué sucede cuando se hace una inserción en una página casi llena?



Opciones [8 | 18]

- En el ejemplo, la página #10 no tiene el suficiente espacio para insertar un registro nuevo (o uno modificado), por lo que el mismo debería ir a la #11, pero ésta también está llena:

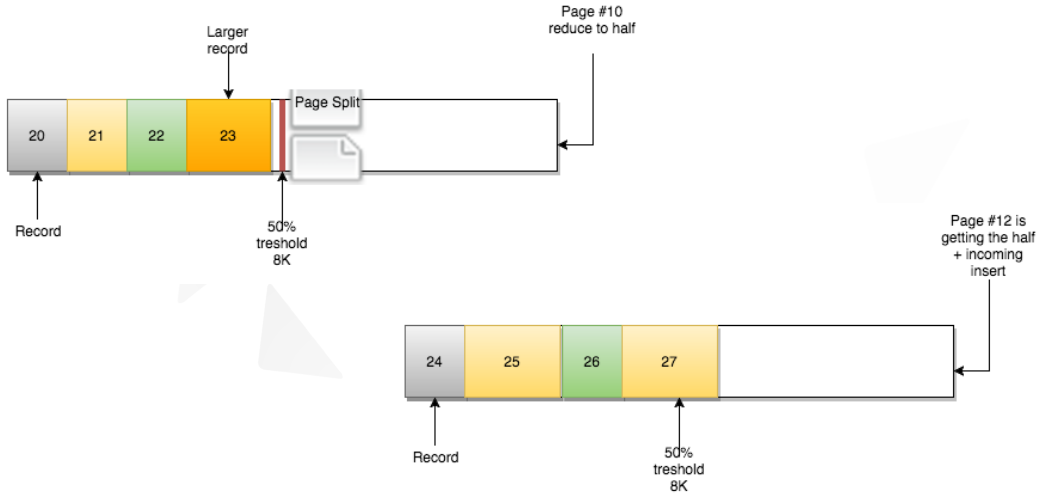


Opciones [9 | 18]

- Como la página #10 tiene un puntero a la #9 y a la #11:
  - InnoDB crea una página nueva (#12)
  - Busca en la página original (#10) dónde se puede dividir (a nivel registro)
  - Mueve los registros
  - Actualiza los punteros siguiente y anterior de las páginas

- Al actualizar los punteros siguiente y anterior de las páginas:
- La página #11 queda sin tocar
- La página #10 tiene como anterior a la página #9 y como siguiente a la página #12
- La página #12 tiene como anterior a la página #10 y como siguiente a la página #11
- La página #11 tiene como anterior a la página #12 y como siguiente a la página #13

Opciones [10 | 18]



- La división de páginas aumenta el tamaño de la tabla y el tiempo necesario para responder las consultas.

Opciones [11 | 18]

- Al hacer la división de páginas, el árbol B mantiene su organización lógica, pero físicamante las páginas quedan desordenadas
- Se realiza una división de páginas cuando se producen inserciones o modificaciones. Provocan desorganización de páginas (en muchos casos en *extents* diferentes)

- InnoDB lleva el control de las divisiones de página en `INFORMATION_SCHEMA.INNODB_METRICS`.

Opciones [12 | 18]

- Una vez que se divide una página, para volver atrás hay que lograr que la nueva página quede debajo del valor de umbral de combinación
- Otra forma es reorganizar físicamente los datos y el índice asociado mediante el comando `OPTIMIZE`, lo cual puede resultar en un proceso pesado y largo, pero suele ser la única forma de corregir la situación donde muchas páginas se localizan en *extents* dispersos

Opciones [13 | 18]

- Tipos de fragmentación:
  - **Interna**: debido a páginas con mucho espacio libre
  - **Externa**: debido a páginas que están desordenadas
- ¿La fragmentación es buena o mala?

- **Fragmentación interna**: suponer que al comienzo del día se tiene una tabla con 40 páginas que están totalmente llenas, pero al final del día se tienen 50 páginas llenas en un 80% debido a los distintos borrados e inserciones. Ahora, cuando se tenga que leer la tabla entera, en lugar de 40 se deberán leer 50 páginas, disminuyendo el rendimiento.

- **Fragmentación externa**: suponer que al comienzo del día se tiene una tabla perfectamente ordenada, pero durante el día se realizan cientos de modificaciones, dejando posiblemente espacio en algunas páginas. Ahora, para leer la tabla entera, se deberá saltar de página en página, en lugar de leer en una única dirección.

Opciones [14 | 18]

- Dependiendo del ambiente, la fragmentación puede ser buena o mala:
  - **OLTP**: la fragmentación puede ser buena: debido al gran número de filas procesadas, éstas siempre encuentran lugar en las páginas para guardarse, reduciendo mucho los tiempos
  - **OLAP**: la fragmentación es mala debido al gran número de páginas con un bajo porcentaje de ocupación

- **OLTP (On Line Transactional Processing)**: son bases de datos orientadas al procesamiento de transacciones. El acceso a los datos está optimizado para tareas frecuentes de lectura y escritura.

- **OLAP (On Line Analytical Processing)**: son bases de datos orientas al procesamiento analítico, lo cual suele implicar la lectura de grandes volúmenes de datos con el fin de extraer algún tipo de información útil. El acceso a los datos suele ser de sólo lectura.

Opciones [15 | 18]

- Opción `innodb_fill_factor`:
  - Define el porcentaje de espacio que se llena en cada página
    - Ejemplo: un valor de 80 llena una página hasta un 80%, por lo que reserva un 20% de espacio libre
  - Valor predeterminado: 100 (llena hasta 15/16 de la página)

- Si el porcentaje de llenado es 100, InnoDB en realidad deja 1/16 de espacio libre en la página. El valor mínimo que se puede especificar es 10.

- El porcentaje de llenado se aplica para todas las páginas del árbol (hojas, intermedias y raíz).

- Esta opción se configura como parámetro de inicio del servicio de MySQL.

Opciones [16 | 18]

- Cuando el porcentaje de llenado de una página cae por debajo del valor de umbral, InnoDB trata de combinar las páginas vecinas:
  - Si ambas están llenas casi en el valor del umbral, se puede producir una división de páginas después de realizada la combinación
- Si este proceso de combinación y división ocurre frecuentemente, puede afectar el rendimiento de las consultas

Opciones [17 | 18]

- Para disminuir la combinación y división, se puede disminuir el valor del umbral, lo cual deja más espacio libre en las páginas
- El valor de umbral se puede especificar a nivel tabla (todos sus índices) o para un índice en particular (prevalece sobre el valor a nivel tabla)

Especificación del valor de umbral en MySQL

- Si no se especifica el valor de umbral, vale 50.
- El valor de umbral no se puede modificar para el índice `GEN_CLUST_INDEX` (este valor se puede cambiar si se modifica el umbral para la tabla).

Opciones [18 | 18]

- Para ver el valor de umbral para un índice se puede:
  - Emplear la sentencia `SHOW INDEX`
- Para ver el valor de umbral para una tabla se puede:
  - Emplear la sentencia `SHOW CREATE TABLE`

Consulta del valor de umbral en MySQL

- Si no se especifica el valor de umbral, vale 50.
- El valor de umbral no se puede modificar para el índice `GEN_CLUST_INDEX` (este valor se puede cambiar si se modifica el umbral para la tabla).

Consideraciones [1 | 7]

- El rendimiento de un motor de BD depende en gran medida de contar con los índices adecuados en las tablas de la BD
- A medida que se modifican la carga y datos con el tiempo, los índices existentes pueden no resultar del todo adecuados, requiriéndose nuevos índices

Consideraciones [2 | 7]

- MySQL utiliza índices:
  - Para encontrar rápidamente las filas que satisfagan la cláusula `WHERE`
  - Si hay varios índices que se puedan usar, MySQL generalmente usa el índice más selectivo
  - Si el índice es compuesto, la columna más a la izquierda es la que se usa para buscar las filas

- Debido a que la columna más a la izquierda en un índice compuesto es la que se usa para buscar las filas, al definir el índice se lo debe hacer desde la columna más selectiva a la menos.

Consideraciones [3 | 7]

- MySQL utiliza índices (cont.):
  - Para recuperar filas de otras tablas cuando se las combina (JOIN)
  - Para comparar columnas de cadenas no binarias, ambas columnas deben usar el mismo juego de caracteres

- En una combinación de tablas, MySQL puede usar más eficientemente los índices si se declaran del mismo tipo y tamaño. En este contexto, `VARCHAR` y `CHAR` se consideran iguales si se declaran del mismo tamaño. Por ejemplo, `VARCHAR(10)` y `CHAR(10)` son del mismo tamaño, pero `VARCHAR(10)` y `CHAR(15)` no.

- Comparar una columna `utf8` con una `latin1` descarta el uso de un índice.

Consideraciones [4 | 7]

- MySQL utiliza índices (cont.):
  - La comparación de columnas diferentes (por ejemplo, una tipo cadena y la otra numérica) puede evitar el uso de un índice si los valores no se pueden comparar directamente sin hacer una conversión
  - En algunos casos, una consulta se puede optimizar para recuperar valores sin consultar las filas de datos (índice cobertor)

- Si todas las columnas que forman la consulta forman parte de la clave del índice, no hay necesidad de llegar hasta la página de datos.

Consideraciones [5 | 7]

- MySQL utiliza índices (cont.):
  - Los índices son poco importantes en consultas sobre tablas chicas, o tablas grandes donde la consulta procesa todas, o casi todas las filas

- Cuando una consulta necesita acceder a casi la totalidad de las filas, leer secuencialmente resulta más rápido que recorrer el árbol del índice. Las lecturas secuenciales minimizan las búsquedas en disco, aún cuando no todas las filas se necesiten para una consulta.

Consideraciones [6 | 7]

- Verificar si las consultas utilizan los índices definidos mediante la sentencia `EXPLAIN`:
  - Brinda información sobre cómo ejecuta las consultas MySQL
  - Funciona con las sentencias `SELECT`, `INSERT`, `DELETE`, `REPLACE` y `UPDATE`
  - Con la ayuda de `EXPLAIN` se puede ver dónde hacen falta índices

- Cuando se usa EXPLAIN con las sentencias mencionadas, MySQL muestra información sobre el plan de ejecución: cómo procesa la sentencia, incluyendo información sobre cómo se combinan las tablas y en qué orden

Consideraciones [7 | 7]

- También se puede ejecutar la sentencia `ANALIZE TABLE` para actualizar las estadísticas de la tabla, lo cual afecta las decisiones que toma el optimizador

Resumen [1 | 1]

- Tipos de integridad
- Restricciones: PK, FK, Default, Check, Unique
- Arquitectura de índices
- Tipos de índices: agrupados y secundarios
- Creación y borrado de índices
- Índices Unique
- Índices compuestos
- Consideraciones

Otro Material [1 | 2]

- **Tipos de índices en MySQL:**
  - https://dev.mysql.com/doc/refman/5.7/en/innodb-index-types.html
- **Combinación y división de páginas:**
  - https://www.percona.com/blog/2017/04/10/innodb-page-merging-and-page-splitting/
- **Configuración del factor de llenado de página:**
  - https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html

Otro Material [2 | 2]

- **Configuración del valor de umbral:**
  - https://dev.mysql.com/doc/refman/8.0/en/index-page-merge-threshold.html