



# 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 2024**



# Contenido

01

Integridad de  
datos


02

Índices



## Integridad de datos



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

# Tipos de integridad

La integridad de los datos hace referencia a:

- La consistencia de los mismos
- La precisión de los mismos

Tipos de integridad:

- Integridad de dominio (o columna)
- Integridad de entidad (o tabla)
- Integridad referencial

# Tipos de integridad

## **Integridad de dominio (o columna):**

- Especifica los valores válidos para una columna, y si la misma admite o no valores nulos
- Se fuerza con:
  - Reglas de validez
  - Restricciones
  - Formatos y rangos de valores admitidos

# Tipos de integridad

## **Integridad de entidad (o tabla):**

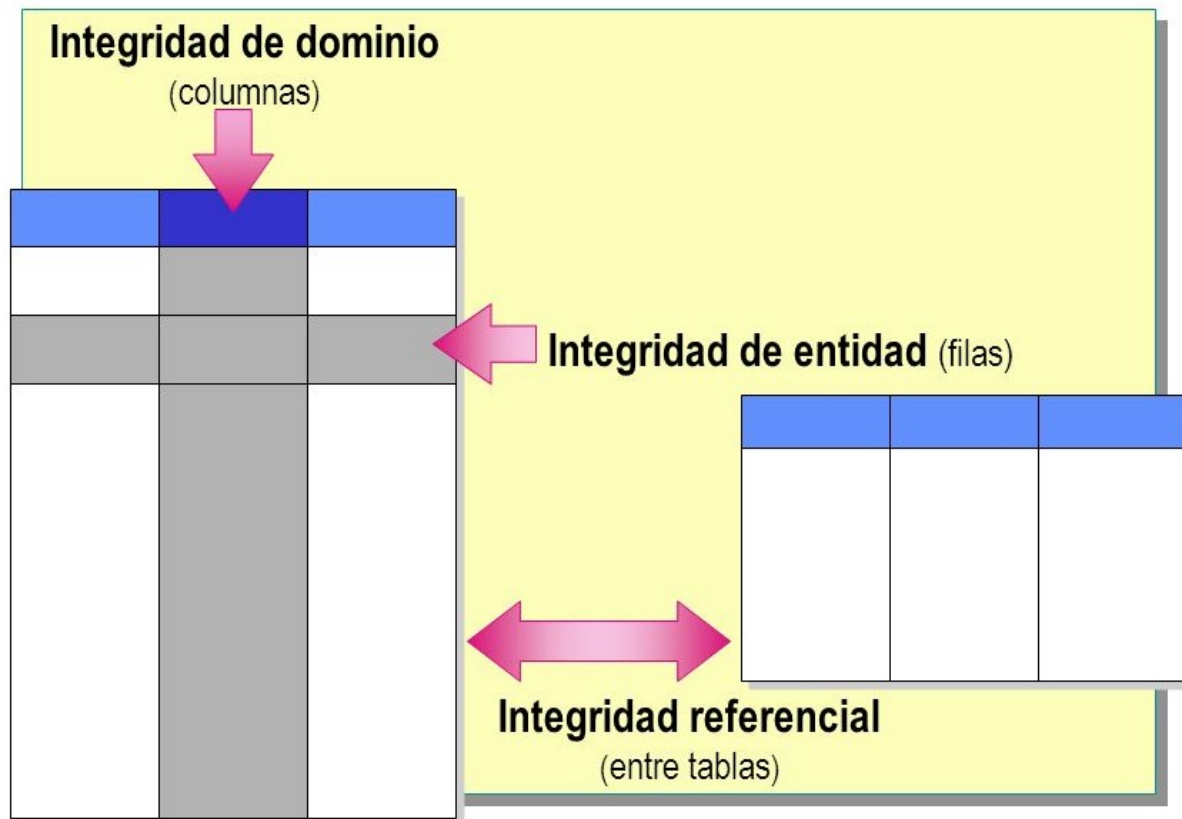
- Requiere que todas las filas de una tabla tengan un identificador único llamado clave primaria (PK)
- Que la clave pueda ser cambiada o borrada dependerá del nivel de integridad requerido

# Tipos de integridad

## 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

# Tipos de integridad





# Tipos de integridad

La integridad se datos se puede forzar de forma:

- **Declarativa**
  - En la definición de los objetos de la BD
  - Automáticamente por el SGBDR
  - Mediante restricciones
- **Procedural**
  - En un script, en el cliente y/o en el servidor
  - Desencadenadores y/o procedimientos almacenados

# Uso de restricciones

Método estándar ANSI para forzar integridad:

- **De dominio**
  - DEFAULT
  - CHECK
  - FOREIGN KEY
- **De entidad**
  - PRIMARY KEY
  - UNIQUE

# Uso de restricciones

- Referencial
  - FOREIGN KEY
  - CHECK

# Uso de restricciones

## 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

## 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

## 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

## Consideraciones:

- Se crean con las sentencias CREATE o ALTER TABLE
- Se pueden agregar restricciones a tablas con datos
- Se pueden aplicar restricciones a una o múltiples columnas
- Se pueden crear, modificar y borrar restricciones sin tener que borrar y recrear la tabla

# Uso de restricciones

Consideraciones (continuación):

- 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
- La BD information\_schema tiene información sobre las restricciones



# Uso de restricciones

Consideraciones (continuación):

- Generalmente se deben especificar nombres para las restricciones, y deben ser únicos en la BD

# Uso de restricciones

## PRIMARY KEY (PK)

- Solo se permite una por tabla (puede ser compuesta)
- Su valor **debe** ser único (si es compuesta, la combinación de los valores **debe** ser única)
- Las columnas que la componen **no pueden** aceptar valores nulos

# Uso de restricciones

## PRIMARY KEY (PK)

- Crea un índice UNIQUE con las columnas que la componen
- El índice no se puede borrar, a menos que se borre la PK

# Uso de restricciones

## PRIMARY KEY (PK)

- Usar una PK cuando:
  - Una o más columnas deban identificar unívocamente cada fila en la tabla
  - Una de las columnas soporte la generación automática de valores

# Uso de restricciones

## FOREIGN KEY (FK)

- Debe referenciar una PK o una restricción UNIQUE en la misma u otra tabla
- Puede proveer referencias a una o varias columnas
- MySQL crea automáticamente índices de las columnas que la componen

# Uso de restricciones

## FOREIGN KEY (FK)

- 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 DEFAULT

# Uso de restricciones

## FOREIGN KEY (FK)

- CASCADE:
  - Al modificar o borrar la fila en la tabla “padre” se modifican o borran las filas correspondientes en la tabla “hija”
- 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 FK

# Uso de restricciones

## FOREIGN KEY (FK)

- RESTRICT:
  - Rechaza la modificación o borrado de la fila en la tabla “padre”
  - Se puede especificar este valor, o bien NO ACTION, u omitir las cláusulas ON DELETE / ON UPDATE



# Uso de restricciones

## FOREIGN KEY (FK)

- SET DEFAULT:
  - Si bien MySQL reconoce esta acción, el tipo de tabla InnoDB rechaza la definición de tablas con esta acción

# Uso de restricciones

## FOREIGN KEY (FK)

- Usar una FK 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

## 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

# Uso de restricciones

## DEFAULT

- 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

## CHECK (desde MySQL 8.0.16)

- A nivel tabla:
  - No aparece en la definición de una columna
  - Puede referenciar cualquier cantidad de columnas
  - Se permiten referencias a columnas no definidas
- A nivel columna:
  - Aparece en la definición de una columna

# Uso de restricciones

## CHECK (desde MySQL 8.0.16)

- Forma genérica:

`[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

## CHECK (desde MySQL 8.0.16)

- 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

## CHECK (desde MySQL 8.0.16)

- Se puede usar en columnas no generadas y generadas, salvo aquellas con el atributo `AUTO_INCREMENT` y columnas en otras tablas
- Se permiten literales, funciones determinísticas y operadores
- No se permiten funciones no determinísticas, procedimientos almacenados, variables ni subconsultas



# Uso de restricciones

## **CHECK (desde MySQL 8.0.16)**

- 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
- Una columna puede tener múltiples restricciones de este tipo

# Uso de restricciones

## CHECK (desde MySQL 8.0.16)

- 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

## UNIQUE

- Se implementa con un índice tipo UNIQUE
- Permite valores nulos (uno solo)
- Una tabla puede tener varias restricciones UNIQUE
- Se puede definir en una o más columnas

# Uso de restricciones

## UNIQUE

- Usar una restricción UNIQUE cuando:
  - La tabla contenga columnas que no sean parte de la PK 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

Restricciones en tablas con datos:

- Cuando se definen restricciones CHECK y FK en una tabla con datos, si los mismos 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 temporalmente este control al crearla

# Uso de restricciones

Restricciones en tablas con datos:

- Si los datos existentes no cumplieran con las condiciones de las restricciones, los mismos se comprobarán al ejecutar una sentencia UPDATE

# Uso de restricciones

Restricciones en tablas con datos:

- También se pueden deshabilitar al insertar datos:
  - Cuando se debe importar una gran cantidad de datos y se sabe que los mismos satisfacen las restricciones
  - Cuando 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

# Elección del método de integridad

Se debe considerar funcionalidad y 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

## Restricciones

- Funcionalidad: media
- Costo: bajo
- Momento: antes de la transacción

## Desencadenadores

- Funcionalidad: alta
- Costo: alto
- Momento: antes y/o después de la transacción

# Índices

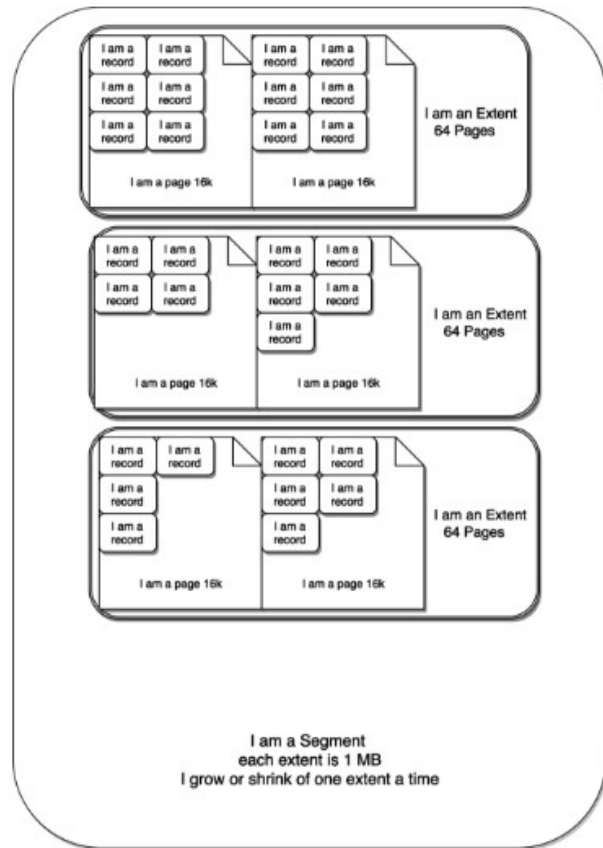


- Arquitectura de índices
- Creación de índices
- Opciones
- Consideraciones

# Arquitectura de índices

## Almacenamiento en MySQL

- Segmentos
- Extents:
  - 1MB de tamaño
  - 64 páginas
- Páginas:
  - 16KB de tamaño



# Arquitectura de índices

## Almacenamiento en MySQL

- 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 se tiene un límite aproximado de 8000 bytes por fila

# Arquitectura de índices

- Ejemplo:

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

# Arquitectura de índices

- 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

Alexander, Mary  
344-555-0133

Kurtz, Jeffrey  
452-555-0179

Vessa, Robert  
560-555-0171

Thames, Judy  
799-555-0198

Martinez, Frank  
171-555-0147

Haines, Betty  
867-555-0114

Burnett, Linda  
121-555-0121

Harris, Keith  
170-555-0127

Kitt, Sandra  
303-555-0117

Brewer, Alan  
494-555-0134

Campbell, Frank  
491-555-0132

Logan, Todd  
783-555-0110

...

Clayton, Jane  
206-555-0195

Johnson, Brian  
320-555-0134

Lis, David  
440-555-0132

Diaz, Brenda  
147-555-0192

# Arquitectura de índices

- Para buscar el teléfono de una persona:

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

- Se recorren **todas** las filas (escaneo de tabla)

# Arquitectura de índices

## 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

En la guía de teléfono:

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

# Arquitectura de índices

## Índice:

- 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

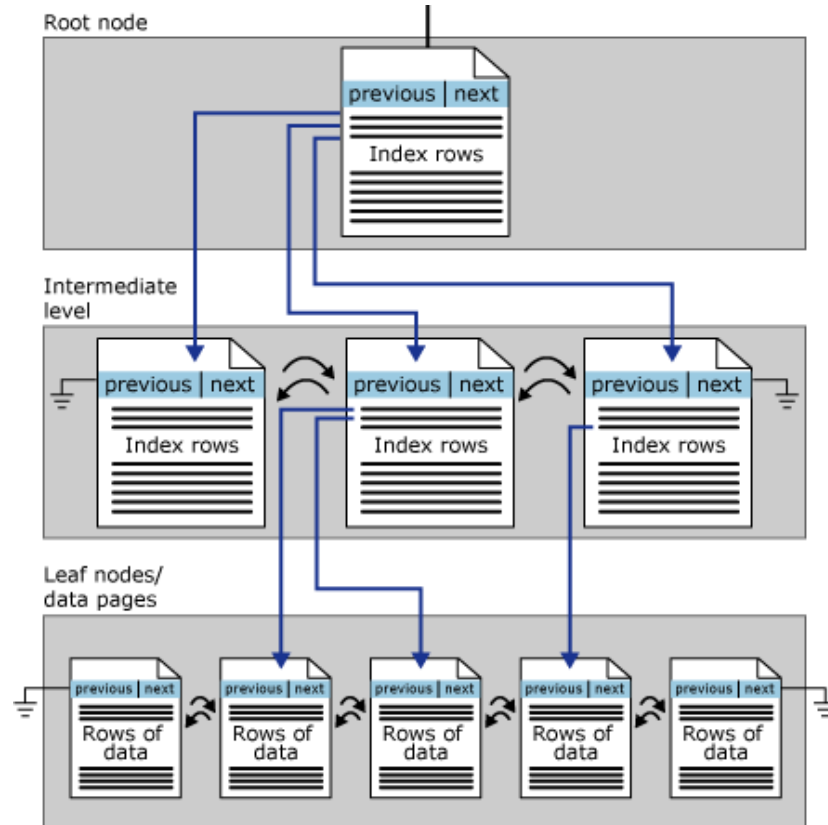
Cuando hay un índice (Búsqueda por Índice o Index Seek):

- 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 la consulta
- Costo:  $O(\log_{2,3} n)$

# Arquitectura de índices

- 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

# Arquitectura de índices



# Arquitectura de índices

- 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

## 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

## **Cuándo no crear índices**

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



# Arquitectura de índices

## Tipos de índices

- Agrupados (clustered) o primarios
- No agrupados (non clustered) o secundarios

# Arquitectura de índices

## Índices agrupados

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

Alexander, Mary  
344-555-0133

Brewer, Alan  
494-555-0134

Burnett, Linda  
121-555-0121

Campbell, Frank  
491-555-0132

Clayton, Jane  
206-555-0195

Diaz, Brenda  
147-555-0192

Haines, Betty  
867-555-0114

Harris, Keith  
170-555-0127

Johnson, Brian  
320-555-0134

Kitt, Sandra  
303-555-0117

Kurtz, Jeffrey  
452-555-0179

Lis, David  
440-555-0132

...

Logan, Todd  
783-555-0110

Martinez, Frank  
171-555-0147

Thames, Judy  
799-555-0198

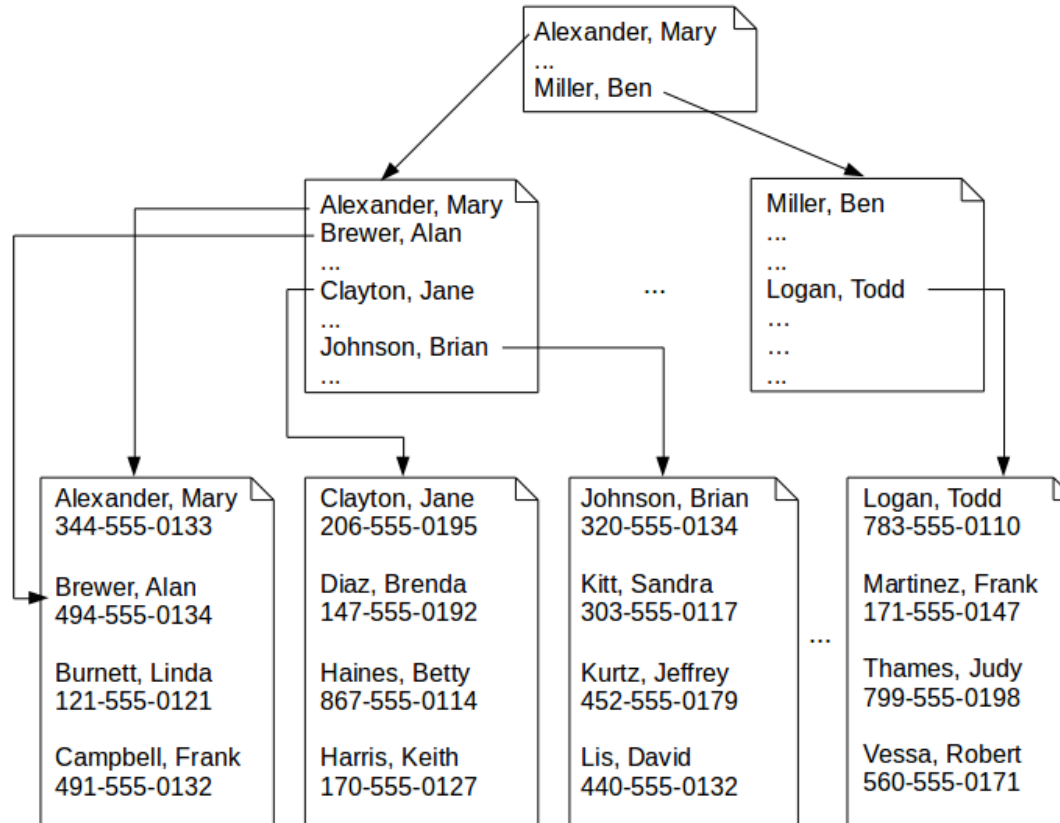
Vessa, Robert  
560-555-0171

# Arquitectura de índices

## Índices agrupados

- 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



# Arquitectura de índices

## Indices agrupados

- A toda la estructura anterior se la llama Índice Agrupado
- La raíz y nodos intermedios sólo guardan los valores de la clave del índice (ordenados ascendentemente)
- Las hojas guardan los datos (ordenados según la clave)

# Arquitectura de índices

## Índices agrupados

- 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

# Arquitectura de índices

## Indices agrupados

- Cada tabla InnoDB tiene un índice agrupado, cuya clave generalmente es la PK
- Si no hay una PK, MySQL localiza el primer índice UNIQUE donde todas las columnas de la clave NO sean NULL y lo usa como índice agrupado

# Arquitectura de índices

## Índices agrupados

- Si no hay una PK o un índice UNIQUE adecuado, se genera internamente un índice agrupado oculto llamado GEN\_CLUST\_INDEX en una columna que contiene valores con identificadores de filas
- **Conclusión:** en MySQL el usuario no puede crear explícitamente un índice agrupado (lo hace MySQL)



# Arquitectura de índices

## Índices no agrupados

- Tienen la misma estructura árbol B que los agrupados
- La raíz y nodos intermedios sólo guardan los valores de la clave del índice no agrupado (ordenados ascendentemente)
- Las hojas guardan las claves del índice agrupado

# Arquitectura de índices

## Índices no agrupados

- La búsqueda comienza recorriendo la estructura del índice no agrupado 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 (se recorren los 2 árboles)

# Arquitectura de índices

## Índices no agrupados

- Los datos y el índice se guardan separados
- Puede haber varios por tabla

# Arquitectura de índices

## Usar un índice agrupado

- **Para recuperar un rango de datos o recuperar datos preordenados**
  - 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

## Usar un índice no agrupado

- Cuando se quieran 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

# Arquitectura de índices

## **No usar un índice agrupado**

- En columnas que se modifican frecuentemente
- Con muchas inserciones secuenciales y concurrentes
- Con claves grandes

# Arquitectura de índices

## No usar un índice no agrupado

- Cuando se recuperan muchas filas

# Arquitectura de índices

## 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

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

# Creación de índices

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

## Índices UNIQUE:

- Aseguran que los datos en columnas indexadas sean únicos
- Se crean automáticamente al crear una restricción UNIQUE o PK
- Si la tabla tiene datos, se controlan valores duplicados
- Al ejecutar un INSERT o un UPDATE, se controlan los valores duplicados y se cancela la sentencia
- En MySQL, 2 o más filas pueden valer NULL. Si se desea evitar esto, no se deben permitir valores nulos

# Creación de índices

- Detección de valores duplicados antes de crear un índice:

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

# Creación de índices

## Índices compuestos:

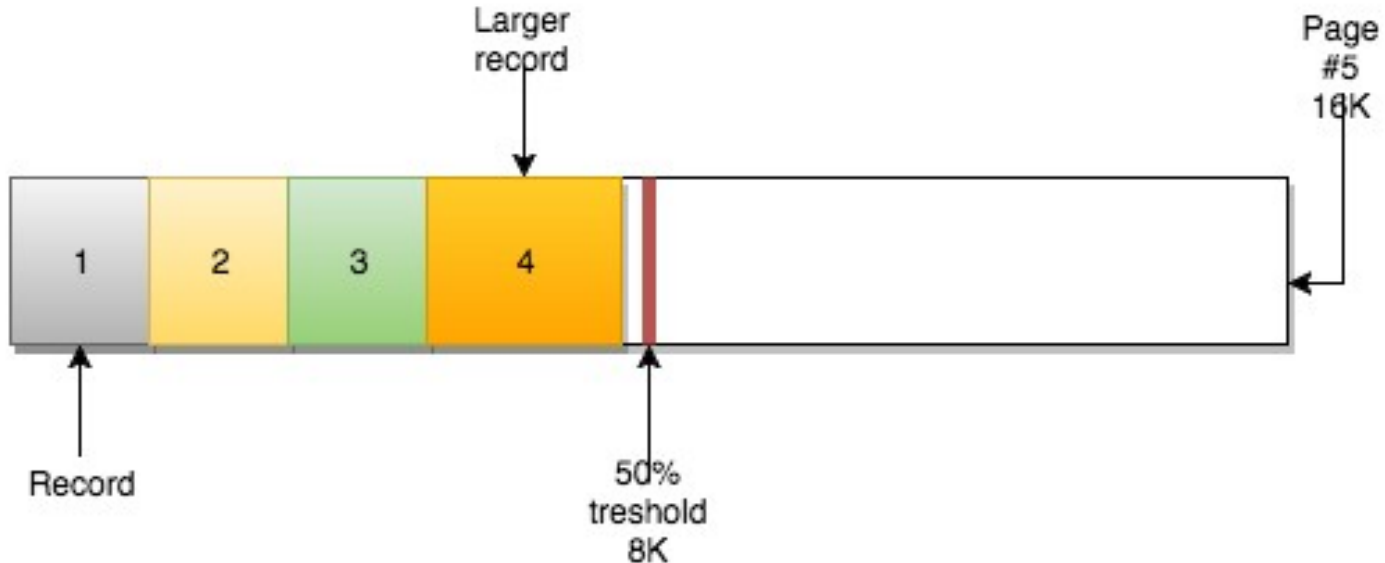
- Crear índices compuestos cuando se solicitan 2 o más columnas para la búsqueda u orden
- Crear índices compuestos cuando las consultas referencien sólo las columnas del índice
- Hasta 16 columnas (MySQL)
- Todas las columnas deben ser de la misma tabla
- El orden debe ser de mayor unicidad a menor unicidad
- Un índice (c1,c2) es diferente a otro (c2,c1)

# Creación de índices

- Información sobre índices:
  - Se emplea la sentencia SHOW INDEX

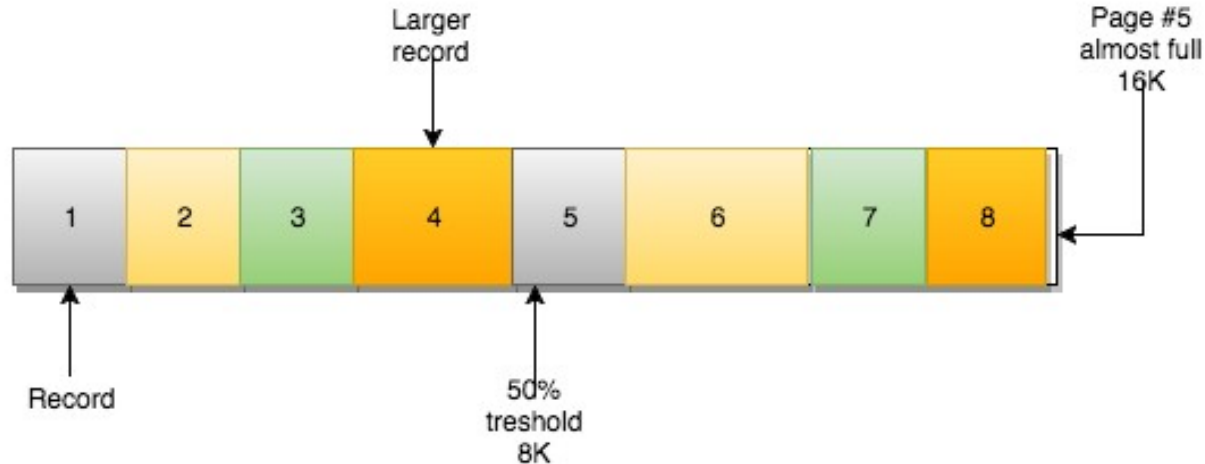
# Opciones

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



# Opciones

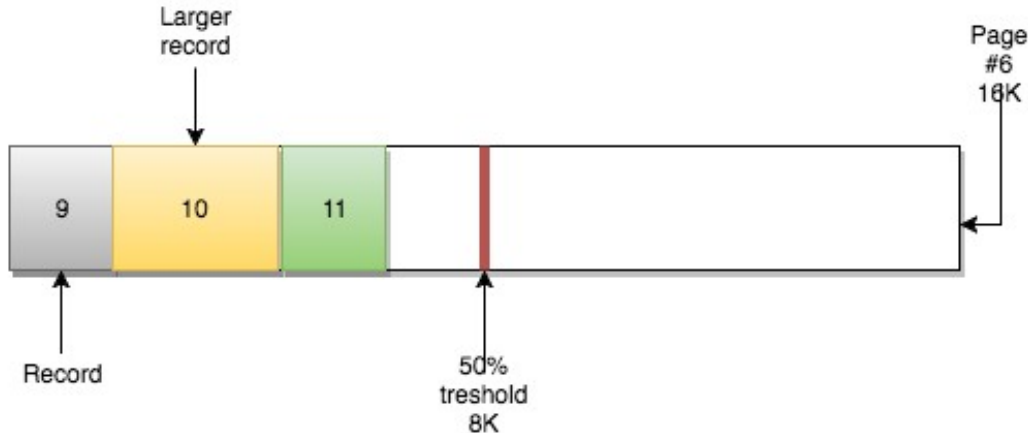
- Una página puede estar vacía o llena (100%)
- Cada página tiene un atributo llamado MERGE\_THRESHOLD





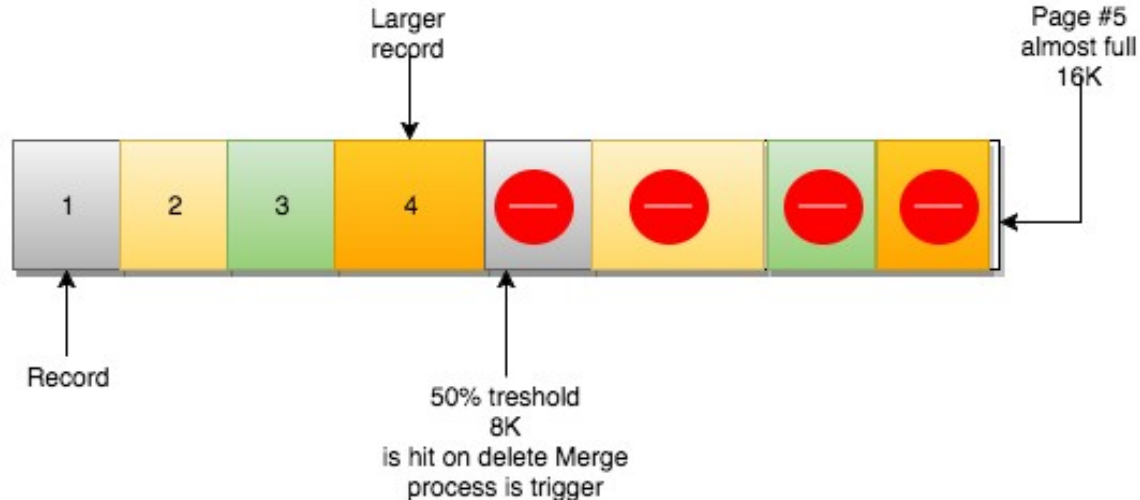
# Opciones

- 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



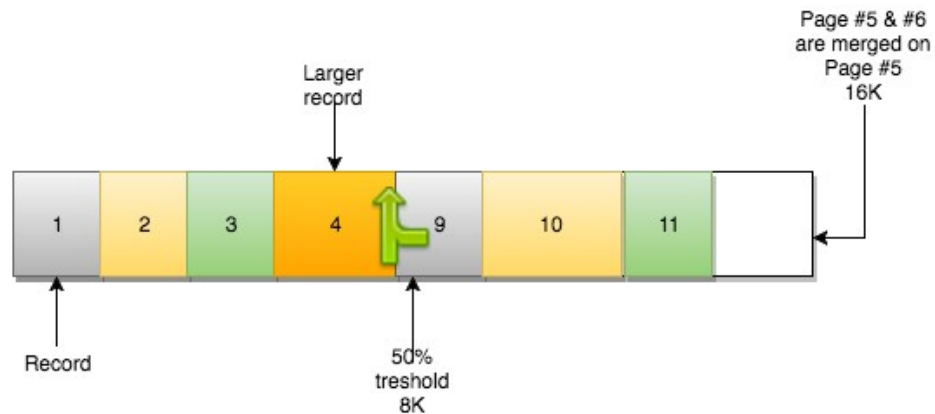
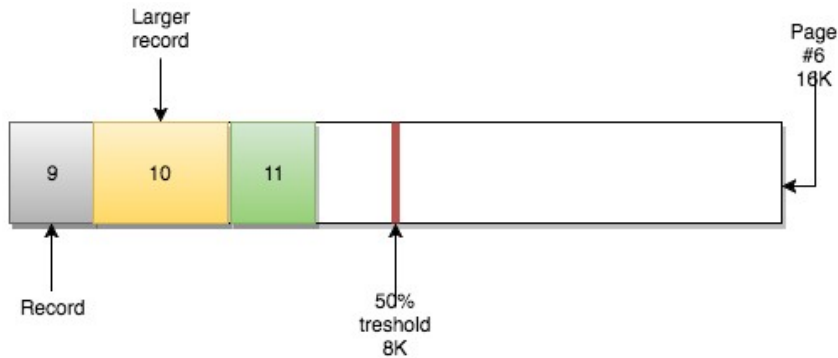
# Opciones

- 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

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

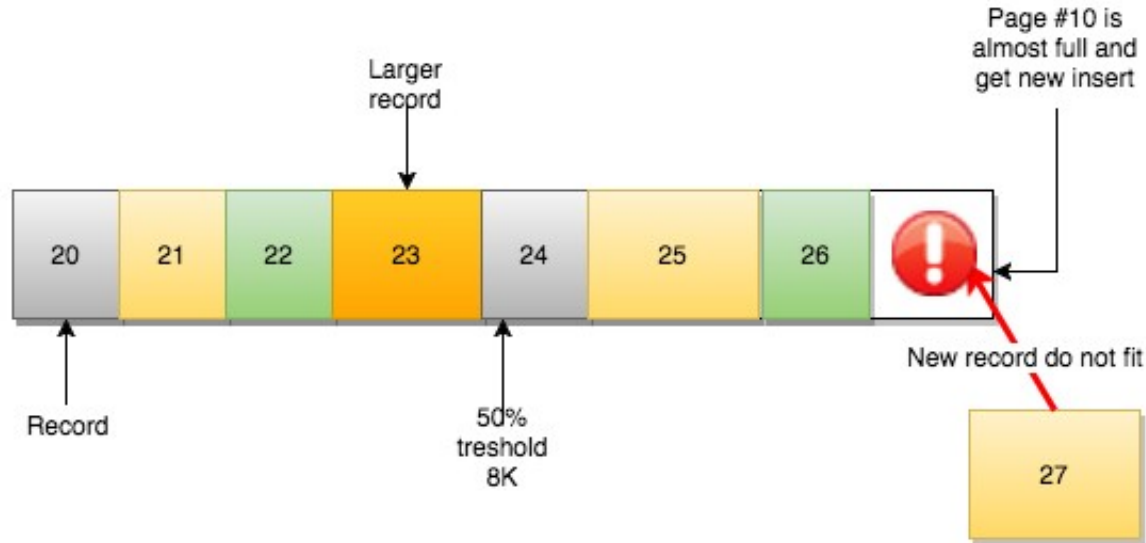


# Opciones

- 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**

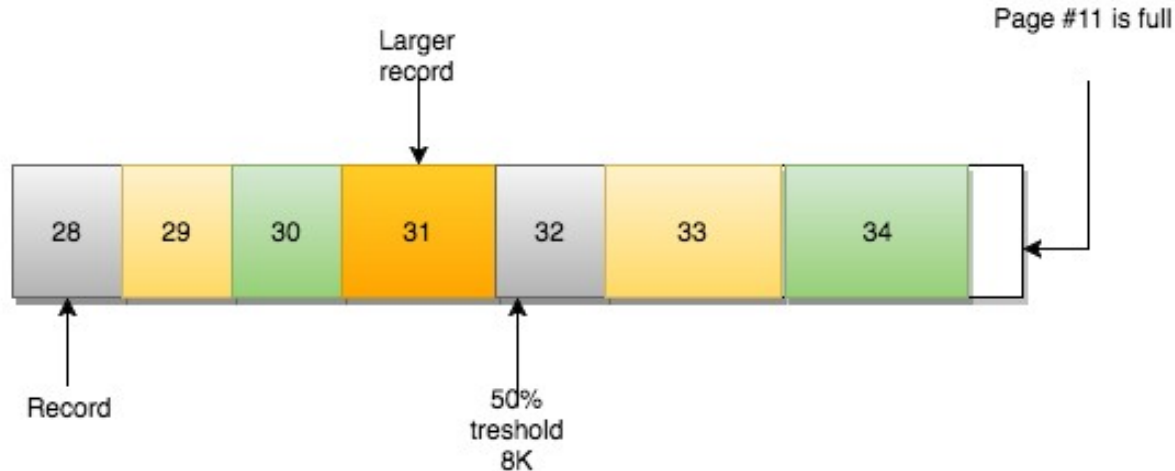
# Opciones

- 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

- 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:

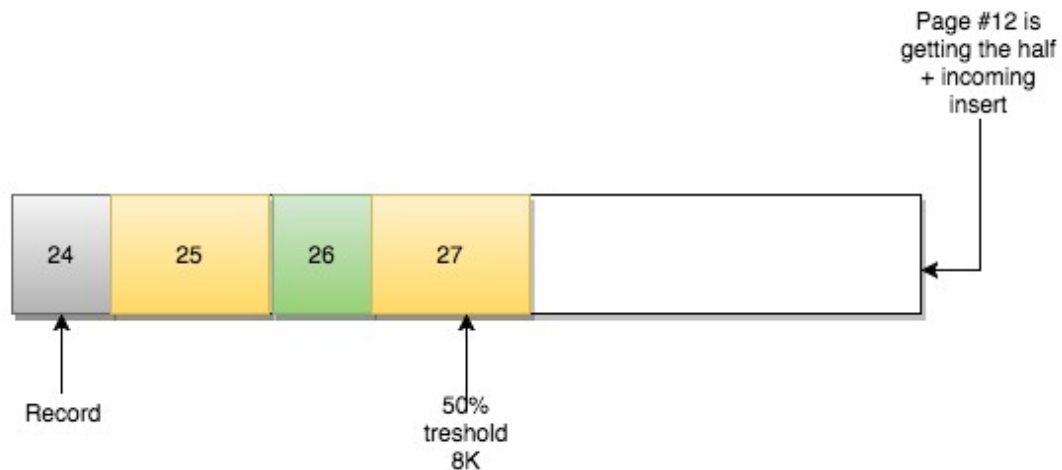
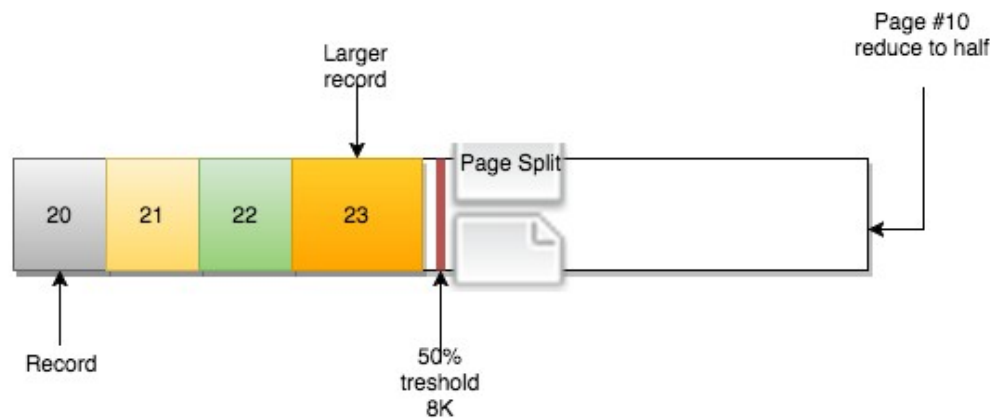


# Opciones

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 de las páginas

# Opciones





# Opciones

- Al hacer la división de páginas, el árbol B mantiene su organización lógica, pero físicamente 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)

# Opciones

- 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

## Tipos de fragmentación:

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

# Opciones

- Según el ambiente, 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

# Opciones

## 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)

# Opciones

- Cuando el porcentaje de llenado cae por debajo del 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

- 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)
  - Para un índice en particular (prevalece sobre el valor a nivel tabla)

# Opciones

- 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`



# Consideraciones

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

# Consideraciones

MySQL utiliza índices:

- Para encontrar filas que satisfagan la cláusula WHERE
- Si hay varios índices, usa el más selectivo
- Para recuperar filas de otras tablas en un JOIN

# Consideraciones

Sobre los índices:

- Para comparar columnas CHAR/VARCHAR, las mismas deben usar el mismo juego de caracteres
- La comparación de columnas diferentes (una cadena y la otra numérica) puede evitar el uso de un índice si los valores no se pueden comparar sin hacer una conversión
- A veces se puede optimizar una consulta para recuperar valores sin consultar las filas de datos (índice cobertor)

# Consideraciones

Sobre los índices:

- Si el índice es compuesto, la columna más a la izquierda es la que se usa para buscar las filas
- Los índices son poco importantes en consultas sobre tablas chicas, o tablas grandes donde la consulta procesa todas, o casi todas las filas

# Consideraciones

Verificar si las consultas utilizan los índices definidos mediante la sentencia EXPLAIN:

- Brinda información sobre cómo se ejecutan las consultas
- Funciona con las sentencias SELECT, INSERT, DELETE, REPLACE y UPDATE
- Con la ayuda de EXPLAIN se puede ver dónde hacen falta índices

# Consideraciones

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

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

# Otros recursos

## **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/>



# Otros recursos

## **Configuración del valor de umbral:**

- <https://dev.mysql.com/doc/refman/8.0/en/index-page-merge-threshold.html>

## **Configuración del factor de llenado de página:**

- <https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html>