2020-Final_19-02-2020

**a)Con un constraint check, se puede definir un rango de fecha preestablecido**

V - CHECK CONSTRAINT → especifica condiciones al hacer un INSERT o UPDATE en una columna. Cada fila insertada o actualizada debe cumplir con dichas condiciones.

**b)Si un arbol binario de busqueda tiene N niveles, la cantidad maxima de lecturas sobre el mismo para encontrar una clave es N-1**

F - en la busqueda de un elemento el maximo de niveles a recorrer es hasta el nivel de la hoja mas baja en el nivel N. Por lo cual debo hacer comparaciones hasta N veces y encontrar el valor (no N-1).

---

2019-Final_26_de_Febrero

**a)En el algoritmo de Huffman, la cantidad de nodos es la siguiente: (total de hojas * 2)-1.**

VERDADERO

**b)Si una columna posee la constraint UNIQUE, entonces una sola fila como máximo puede contener NULL en dicha columna**

VERDADERO - cada valor debe ser distinto en la fila

---

2019-Final_25_de_Septiembre

**a)En una db Relacional las vistas pueden ser utilizadas para brindar consistencia de datos**

FALSO - Las transacciones son utilizadas para brindar consistencia de datos.

**b)En huffman si un caracter posee el codigo 0011, entonces con seguridad existe al menos otro cuyo codigo comienza con 001**

VERDADERO

---

2019-Final_12_de_Febrero

**a)Un constraint de tipo CHECK siempre puede ser reemplazado por un trigger**

VERDADERO

**b)Los Arboles B garantizan un numero de niveles menos que otro arboles**

FALSO - Los árboles-B no garantizan menos niveles. Los niveles dependen de cosas como la cantidad de datos, el orden, el grado de completitud de un nodo, …

---

2019-Final_04_de_Diciembre

**a)Una columna afectada por una constraint UNIQUE no puede almacenar valores repetidos ni NULL**

FALSO - Puede almacenar un unico NULL

**b) Hashing es mas performante que el Arbol B en la busqueda de una clave unica en particular existente.**

VERDADERO - En el caso más general, Hashing es más performante que Árbol B: Hashing, a partir de una clave, devuelve un índice de una tabla con la dirección de memoria del registro con dicha clave, con el riesgo de que ocurran colisiones, por eso es más performante que Árbol B en búsqueda por clave. De hecho, Árbol B es mejor para búsquedas por rango. Sin embargo, siempre hay que tener en cuenta la distribución de datos y como esté configurada la función de hash: si una búsqueda por hash produce muchas colisiones para una misma clave, es probable que un Árbol B sea más performante que Hashing.

TAMBIEN PUEDE SER FALSO - Si tenés que hacer rehashing 10000 veces para acceder directo a una clave cuando podías agarrar la dirección de la primer hoja del árbol B, bueno, ahí sí es más performante Árbol B

---

2018-Final_17_de_Julio

**a) Es posible implementar el concepto de ABB (Arbol Binario de Busqueda) en un Array**

VERDADERO

**b) Todo grafo de grado 2 es un arbol binario**

FALSO. De ser reflexivo ya el grafo no seria arbol.

---

2018-Final_12_de_Marzo

**a) Si se desea que no se puedan eliminar registros de una tabla de auditoria, una opcion es crear un trigger que lo impida.**

VERDADERO

**b) La cantidad de nodos de un arbol de expresion siempre es par**

FALSO

---

2018-Final_21_de_Febrero

**a) El Arbol B+ es un arbol Principal Derecho Balanceado**

VERDADERO - Es balanceado y el Arbol Principal Derecho todos los nodos no-principales tienen un unico padre y el nodo principal es el minimal del arbol (llamado raiz) y es unico.

**b) Para entornos transaccionales de alta concurrencia es conveniente setear el Isolation Level en Repeatable Read**

FALSO - Dado que tiene un mayor alcance de lockeo de los registros, se verá afectada la performance. Es recomendable un nivel mas permisivo como read commited o read uncommited

---

2017-Final_7_de_Febrero

**a) En caso de no querer eliminar registros de una tabla de audtoría, podría definirse un trigger para que lo impida.**

VERDADERO

**b) La cantidad de nodos de los árboles de expresión es siempre par.**
FALSO - Contraejemplo: {(x, 1), (y, 2)}.

---

2017-Final_12_de_Mayo

**b) El algoritmo de Huffman obtiene los códigos comprimidos parseando un árbol binario balanceado**
FALSO El árbol binario no es necesariamente balanceado. Tomen como ejemplo el árbol que surge de comprimir el texto "Tomemos como ejemplo esta frase. " con Huffman.

---

2017-Final_12_de_Diciembre

**a)En el arbol de Huffman la cantidad total de nodos es la siguiente: (total de hojas *2)-1**
VERDADERO

**b)La implementacion de la cantidad de entradas para claves en una tabla de hash es dinamica.**
FALSO - La implementación de las claves de hash pueden ser dinámicas o estáticas.

---

2017-Final_25_de_Julio

**a)Si un arbol N-ario tiene un total de 2 niveles, entonces la profundidad maxima que alcanza un nodo en la transformada de Knuth es N**
VERDADERO siempre que N>2

**b)El nivel de aislamiento repeatable read acepta lecturas fantasmas**
VERDADERO

---

2017-Final_11_de_Julio

**a)El algoritmo de Huffman solo es aplicable a archivos de texto por la forma en que trabajan las repeticiones.**
FALSO

**b)Las foreign key son la unica forma que tienen las bases de datos para implementar la integridad relacional entre las tablas de un modelo.**
FALSO

---

2016-Final_20_de_Diciembre

**a)Si una funcion de hash no posee una buena dispersion, se van a producir muchas colisiones**
VERDADERO

**b)Nunca es posible ejecutar la operacion de insert sobre un vista**
FALSO - Se puede hacer un INSERT sobre una vista

---

2016-Final_06_de_Diciembre

**a). El árbol de expresión siempre está balanceado en su raíz.**

FALSO - Contraejemplo: a + (b + c) * d.

**b)El Heapsort tiene peor rendimiento si los datos ya vienen ordenados.**

FALSO - El rendimiento del Heapsort es siempre constante y es O(n log n).

---

2016-Final_24_de_Mayo

**a)Un arbol binario de busqueda siempre es un arbol completo**

FALSO

**b)En un trigger de update las tablas inserted y deleted tienen siempre la misma cantidad de registros**

VERDADERO

---

2016-Final_01_de_Marzo

**a)El tiempo de ejecucion del Algoritmo de clasificacion HEAPSORT solo depende de la cantidad de elementos a ordenar.**

VERDADERO

**b)Un arbol de grado mayor a dos no puede ser representado mediante una representacion computacional estatica.**

FALSO

---

2018-Final_30_de_Julio

**a)Siempre es recomendable que toda la tabla indexada por ARBOL B+ tenga load factor.**

VERDADERO

Manejo del Load Factor

FILLFACTOR- Porcentaje de cada página del índice a ser dejado como espacio libre en su creación. Por ej. Si el FILLFACTOR=20, en la creación del índice se ocupará hasta el 80% de cada nodo.

```
CREATE UNIQUE INDEX ix1_ordenes ON ordenes(N_orden)  WITH FILLFACTOR = 20
```

**b)Un nivel de aislamiento SERIALIZABLE es recomendable para no leer datos sucios de una tabla.**

VERDADERO

---

2018-Final_14_de_Febrero

**a)Un sistema de Data Warehousing no permite la integracion de bases de datos heterogeneas (relacionales, documentales, geograficas, archivos, etc)**

FALSO

**b)Un trigger unicamente puede modificar datos de la tabla a la que pertenece**
FALSO

---

2017-Final_21_de_Febrero

**a)La reexpresion de caracteres al aplicar huffman implica la disminucion de 8bits para la expresion de todos los caracteres.**
FALSO

**b)La ejecucion sin filas de resultado de una query dentro de un trigger genera la cancelacion de la transaccion.**
FALSO

---

2017-Final_14_de_Febrero

**a)Los niveles de aislamiento permiten que no se generen deadlocks.**
FALSO - Repeatable read puede generar deadlocks.

**b)En SQL Server una tabla puede tener mas de un trigger INSTEAD OF INSERT**
FALSO

---

2016-Final_23_de_Febrero

**a) El algoritmo de Huffman siempre se basa en árboles completos**
FALSO. Puede ser que tengas un nodo a la derecha y 200 a la izquierda; todo depende de la frecuencia de los caracteres

**b) El algoritmo QuickSort es siempre más rápido que Heap Sort**
FALSO. QuickSort iba de O(n log n) a O(n^2) (creo; ya no me acuerdo), mientras que HeapSort siempre tenía O(n log n)

---

2016-Final_16_de_Febrero

**a)Si un arbol B tiene N claves entonces el grado es N+1**
FALSO
**b)Una sub consulta ubicada en el where siempre debe retornar al menos una fila**
FALSO

---

2015-Final_01_de_Noviembre

**a)Es posible asegurar Integridad Referencial entre dos tablas de Base de Datos Diferentes**
VERDADERO - Trigger

**b)La estructura de un ABB (Arbol Binario de Busqueda) es un arbol completo.**
FALSO

---

2015-Final_15_de_Diciembre

**a)Si una consulta posee la constraint de unique, entonces ninguna fila acepta nulos en dicha columna.**
FALSO. aceptara solo uno

**b)Al aplicar un barrido simetrico sobre un ABB se obtiene el conjunto de datos ordenado.**
VERDADERO

---

2015-Final_01_de_Diciembre

**a) Dada una tabla que tiene un trigger de INSERT; Si al insertar una fila sin ninguna transacción iniciada en la tabla, la acción disparada por el trigger falla, el insert de la fila no se inserta en la tabla. (Quedó medio chota, por eso en el final el profesor aclaró que la afirmación se refiere a que no se insertan datos en la tabla)**
FALSO

**b) El árbol lleno y el árbol completo son dos tipos de árboles binarios balanceados.**
FALSO

---

2015-Final_28_de_Julio

**a)Las claves alternas, son posibles claves foraneas que pertenecen al conjunto de las claves candidatas**
FALSO. Claves PRIMARIAS

**b)El metodo de Hashing resuelve mas eficientemente las busquedas con rangos de claves.**
FALSO. ARBOL B+

---

2014-Final_16_de_Diciembre

**a)La siguiente consulta retorna como maximo 1 fila**
**SELECT distinct 1 FROM tabla1 WHERE campo1 = 1 union all SELECT distinct 1 FROM tabla1 WHERE campo2=2**
FALSO

**b)La cantidad de nodos de un arbol de huffman siempre es impar.**
VERDADERO

---

2014-Final_02_de_Diciembre

**a)El QuickSort es mas performante que el HeapSort, sin importar como vengan los datos (ordenados o desordenados)**
FALSO - misma performance para ambos

**b)Sobre un arbol n-ario con n>2 se pueden realizar los siguientes barridos (recorridos), preorden, simetrico, posorden y por niveles**
FALSO - Solo por niveles

---

2014-Final_26_de_Mayo

**b)La implementacion de la cantidad de entradas para claves en una tabla de hash es dinamica**
FALSO

---

2014-Final_11_de_Febrero

**a)Toda consulta que utilice al menos una funcion de grupo debe ir acompañada de la clausula "group by"**
VERDADERO

**b)La ejecucion sin filas de resultado de una query dentro de un trigger siempre genera la cancelacion de la transaccion.**
FALSO

---

2013-Final_26_de_Febrero

**a)Si tengo un conjunto de datos tendiendo a ordenados, el algoritmo de QuickSort es el mas eficiente para su ordenamiento total.**
FALSO.

**b)La accion que ejecuta un trigger y el evento que lo dispara siempre se ejecutan de manera atomica**
VERDADERO

---

2012-Final_04_de_Diciembre

**a)El Orden de complejidad de un ABB siempre es mejor que el Orden de complejidad del QuickSort**
FALSO

**b)Para comprimir en el algoritmo de Huffman, se debe leer en un ciclo cada caracter del archivo a comprimir y acceder al arbol desde la raiz para llegar a la hoja que contiene el caracter. Si desciendo por un hijo**

**izquierdo agrego un 0 como bit del codigo comprimido, si desciendo por un hijo derecho agrego un 1.**
FALSO

---