1. ¿Qué son los archivos ISAM? ¿Describa brevemente su funcionamiento? ¿Es una estructura fácil de migrar? ¿por qué?

**ISAM** son siglas de *Indexed Sequential Access Method* (Método de Acceso Secuencial Indexado),

* Método para almacenar información a la que se pueda acceder rápidamente.
* La información se organiza en [registros](http://es.wikipedia.org/wiki/Registro) compuestos por [campos](http://es.wikipedia.org/wiki/Campo_%28base_de_datos%29) de tamaño fijo.
* Los registros se almacenan secuencialmente, inicialmente para acelerar el acceso en sistemas de cinta.
* Un conjunto secundario de [ficheros dispersos](http://es.wikipedia.org/wiki/Hash) (tablas "hash") conocidos como *índices* contienen «punteros» a los registros que permiten acceder a los registros individuales sin tener que buscar en todo el fichero.
* Este es el punto de partida para todos los modernos sistemas de bases de datos navegacionales, en los cuales los punteros que dirigen hacia otra información fueron almacenados dentro de los propios registros.
* El avance clave que posee ISAM es que los índices son pequeños y pueden ser buscados rápidamente, permitiendo a la base de datos acceder sólo a los registros que necesita.
* Modificaciones adicionales a la información no requieren cambios a otra información, sólo a la tabla y los índices.

Las [bases de datos](http://es.wikipedia.org/wiki/Base_de_datos) relacionales pueden fácilmente ser construidas en una red ISAM con la adición de lógica para mantener la validez de los enlaces entre las tablas. Típicamente el campo usado como enlace, será indexado para su lectura rápida. Si bien es cierto que esto es más lento que simplemente almacenar el puntero relacionado a la información directamente en los registros, esto también significa que los cambios al orden físico de la información no requiere ninguna actualización de punteros, entonces éstos siguen siendo válidos.

ISAM es muy fácil de entender e implementar, puesto que básicamente consiste en el acceso directo y secuencial a un archivo en una base de datos. También es muy barato. El truco está en que cada máquina cliente debe administrar su propia conexión a cada archivo que accede. Esto, a su tiempo, presenta la posibilidad de inserciones conflictivas a esos archivos, que a su vez causa una base de datos inconsistente. Esto es típicamente solucionado con la adición de una red cliente-servidor, que supervisa las solicitudes del cliente y se mantiene ordenando.

1. ¿Cuáles son los utilitarios de la sección frontal? Describa sus principales características. De ejemplos. ( No lo encontré)
2. ¿Qué son los archivos de montículo? Describa brevemente sus características.

Hay varias organizaciones primarias de archivos que determinan la forma en que los registros de un archivo *se colocan físicamente en el disco.* Los *archivos de montículo* (o *archivos no ordenados)* colocan los registros en disco sin un orden específico, en tanto que los *archivos ordenados* (o *archivos secuenciales)* mantienen los registros ordenados según el valor de un cierto campo. Los *archivos dispersos* utilizan una función de dispersión para determinar la colocación de los registros en el disco. Otras organizaciones primarias de los archivos, como los *árboles* B, se valen de estructuras de árbol.

En el tipo más simple y básico de organización (montículo) , los registros se colocan en el archivo en el orden en que se insertan, y los registros nuevos se insertan al final del archivo. Un archivo con este tipo de organización se denomina archivo de montículo o de montón. Esta organización suele utilizarse con caminos de acceso adicionales, como los índices secundarios.

La inserción de un registro nuevo es muy eficiente: el último bloque del archivo en disco se copia en un buffer, se añade el nuevo registro, y se reescribe el bloque en el disco.

Para eliminar un registro, el programa encargado de ello primero tendrá que hallarlo, copiar el bloque en un buffer, eliminar el registro del buffer y finalmente reescribir el bloque en el disco. Esto deja espacio desocupado adicional en el bloque, así que la eliminación de un gran número de registros de esta forma origina una gran cantidad de espacio desperdiciado.

1. Dentro de las técnicas de dispersión se encuentra la “Dispersión Interna”. Explique el concepto.

AGREGADO PARA ENTENDER MAS EL TEMA

Otro tipo de organización primaria de archivos se basa en la dispersión *(hashing),* que proporciona un acceso muy rápido a los registros con ciertas condiciones de búsqueda. Esta organización suele recibir el nombre de archivo **disperso** o **directo.** La condición de búsqueda debe ser una condición de igualdad sobre un solo campo, el **campo de dispersión** del archivo. Es común que el campo de dispersión sea también un campo clave del archivo, en cuyo caso se habla de la **clave de dispersión.** La dispersión se basa en establecer una función h, llamada **función de dispersión** o **función de aleatorización,** que se aplica al valor del campo de dispersión de un archivo y produce la *dirección* del bloque de disco en el que está almacenado el registro. La búsqueda del registro dentro del bloque puede realizarse en un *buffer* de la memoria principal. Para la mayoría de los registros, basta un solo acceso a bloque para obtener un registro. La dispersión también sirve como estructura interna de datos en un programa siempre que se tenga acceso a un archivo temporal pequeño empleando el valor de un campo.

Por lo regular, la dispersión en los archivos internos se implementa con un arreglo de registros. Supongamos que el intervalo del índice del arreglo va de 0 a M – 1 ; entonces, tendremos M **casillas** cuyas direcciones corresponderán a los índices del arreglo. Elegiremos una función de dispersión que transforme el valor del campo de dispersión en un entero entre 0 y M - 1. Una función de dispersión común es la función *h(K) = K* **modM,** que devuelve el residuo, o resto, de dividir un valor entero *K* del campo de dispersión entre M; este residuo se utiliza como dirección del registro. Los valores no enteros del campo de dispersión se pueden convertir en enteros antes de aplicar la función mod. En el caso de cadenas de caracteres, en la transformación se pueden

usar los códigos numéricos asociados a los caracteres; por ejemplo, multiplicando los valores de dichos códigos.

1. ¿Cómo se define la navegación entre tipos de registro de un modelo de red? ( ya esta hecha)
2. ¿Que son los tipos de vínculos padre/hijo virtuales? ¿Cómo mejoran la capacidad de modelado del modelo jerárquico?

( ya esta hecha)

1. De todas las propiedades que poseen las relaciones, cuales tienen que ver con el cuerpo de la relación. Explique cada una de ellas justificando su respuesta.

Propiedades de las relaciones

Las relaciones poseen ciertas propiedades, todas ellas consecuencias inmediatas de la definición de *relación* dada en la subsección anterior, y todas ellas muy importantes. Primero enunciare­mos brevemente las propiedades y luego las explicaremos en detalle. Las propiedades son como sigue. Dentro de cualquier relación dada:

1. No existen tupias duplicadas;
2. Las tupias están en desorden, de arriba hacia abajo;
3. Los atributos están en desorden, de izquierda a derecha;

■ Cada tupla contiene exactamente un valor para cada atributo.

1. No existen tupias duplicadas

Esta propiedad surge del hecho de que el cuerpo de la relación es un conjunto matemático (de tuplas); los conjuntos en matemáticas no incluyen elementos duplicados.

Nota: De hecho, debe ser obvio que el concepto de "tuplas duplicadas" no tiene sentido. Considere una relación con los atributos V# y CIUDAD (que significan que "el proveedor V# está ubicado en la ciudad CIUDAD"). Suponga que la relación contiene una tupia que muestra que es un "hecho verdadero" que el proveedor V1 está ubicado en Londres. Entonces, si la relación contuviera un duplicado de esa tupia (si eso fuese posible), simplemente nos estaría di- ciendo, por segunda vez, el mismo "hecho verdadero". Pero si algo es cierto, ¡decirlo dos veces no lo hace más cierto!

A propósito, esta primera propiedad sirve para ilustrar el punto de que (en general) una relación y una tabla no son lo mismo; ya que (en general) una tabla puede contener filas dupli­cadas —en ausencia de cualquier disciplina que evite tal cosa—, mientras que una relación no puede contener una tupia duplicada. (Ya que si una "relación" contiene tupias duplicadas, en-tonces, por definición, ¡no es una relación!) Es muy desafortunado el caso de que SQL permita que las tablas contengan filas duplicadas. Éste no es el lugar apropiado para abordar todas las razones por las cuales tales duplicidades son un error (para una explicación amplia, vea las referencias [5.3] y [5.6]); para los fines actuales, es suficiente señalar que el modelo relacionala) reconoce duplicados, y por lo tanto, en este libro nos aseguraremos que nunca ocurran. (Esta ob-servación se aplica principalmente a nuestras explicaciones de SQL. Por supuesto, en lo que res- pecta al modelo relacional en sí mismo, no es necesario tener un cuidado especial.)

2. Las tupias están en desorden, de arriba hacia abajo

Esta propiedad también surge del hecho de que el cuerpo de la relación es un conjunto ma-temático; en matemáticas, los conjuntos no están ordenados. Por ejemplo, en la figura 5.1 las tu-pias también podrían haber sido mostradas en secuencia inversa, y seguiría siendo la misma relación. Por lo tanto, no hay algo que se llame "la quinta tupia" o "la 97a tupia" o "la primen tupia" de una relación y tampoco hay algo como "la siguiente tupia"; en otras palabras, no exis­te el concepto de direccionamiento posicional y no existe el concepto de "la que sigue". La re­ferencia [5.6], mencionada anteriormente con relacion a la propiedad de que "no hay tupias duplicadas", muestra por qué también es tan importante la propiedad de "no ordenamiento de tupias" (de hecho, ambas propiedades están interrelacionadas).

Nota: Es cierto que se requiere algún concepto como el de "la que sigue" en una interíaz entre la base de datos y un lenguaje anfitrión como C o COBOL (vea la explicación de cursores de SQL y la cláusula ORDER BY en el capítulo 4). Pero es el lenguaje anfitrión y no el modelo relacional el que impone ese requerimiento. En efecto, el lenguaje anfitrión requiere que conjuntos sin orden se conviertan en listas o arreglos (de tupias) ordenados, de modo que las operaciones como "ac-ceder a la siguiente tupia" puedan tener significado. Observe además que dichas características sólo forman parte de la interfaz del programa de aplicación; no están expuestas a los usuarios finales.

Como mencioné anteriormente en esta sección, esta segunda propiedad sirve también para ilustrar la idea de que una relación y una tabla no son lo mismo, ya que las filas de una tabla ob­viamente tienen un ordenamiento de arriba hacia abajo, mientras que las tupias de una relación no lo tienen.

3. Los atributos están en desorden, de izquierda a derecha

Esta propiedad surge del hecho de que el encabezado de una relación también es un conjunto (de atributos). Por ejemplo, en la figura 5.1 los atributos también podrían haber sido mostrados en el orden (por decir) PROVEEDOR, CIUDAD, STATUS, V#, y seguiría siendo la misma relación (por lo menos en lo que respecta al modelo relational).\* Por lo tanto, no existe algo como "el primer atributo" o "el segundo atributo" (etcétera) y tampoco hay "el siguiente atri­buto" (de nuevo, no existe el concepto de "el siguiente"); en otras palabras, siempre se hace re­ferencia a los atributos por nombre, nunca por posición. Como resultado, se reduce el alcance de los errores y la programación confusa. Por ejemplo, no hay (o no debe haber) una forma de hacer que el sistema "se extienda" de algún modo de un atributo a otro. Esta situación contrasta con la que se encuentra en muchos sistemas de programación, donde a menudo es posible ex­plotar de muy diversas formas la adyacencia física de elementos lógicamente discretos, de ma­nera deliberada o no.

Observe que esta cuestión del ordenamiento de los atributos es otra área más en donde la representación concreta de una relación como una tabla, sugiere algo que no es cierto: las colum­nas de una tabla tienen obviamente un ordenamiento de izquierda a derecha, pero los atributos de una relación no lo tienen.

4. Cada tupia contiene exactamente un valor para cada atributo

Esta propiedad surge inmediatamente de la definición de una tupia: una tupia es un conjunto de n componentes o pares ordenados de la forma Ai:vi (i= 1,2,..., n). Se dice que una relación que satisface esta cuarta propiedad está normalizada o, lo que es equivalente, está en la primera forma normal.

Nota: Esta propiedad en particular podría parecer muy obvia y de hecho lo es; en especial debido a que (como tal vez haya notado) todas las relaciones están normalizadas de acuerdo con la definición. Sin embargo, esta propiedad sí tiene algunas consecuencias importantes. Vea (a) la nota a la referencia [5.10], (b) el capítulo 18 sobre información faltante y (c) la subsección siguiente.

1. Defina las operaciones del algebra relacional correspondientes a los operadores tradicionales.
2. ¿Que son las técnicas de dispersión? Explique el concepto de “Dispersión Externa”. Enumere las ventajas y desventajas.
3. Describa los siguientes tipo de índice: primario, de agrupamiento y secundario ¿Qué diferencia hay entre ellos?
4. Defina la arquitectura de una base de datos. ¿Que representa cada nivel? ¿Qué son las transformaciones?
5. ¿Que son las reglas de inferencia? ¿Para qué sirven? Enumérelas.
6. ¿Qué significan las relaciones? ¿Cómo se componen? Defina los conceptos de “Predicado” “proposición verdadera” ¿Qué representan cada uno de ellos?
7. Defina el concepto de dominio ¿Qué consideraciones se deben tener sobre las operaciones?
8. ¿Cuáles son las propiedades de las relaciones?

1. Defina el concepto de dependencia funcional. ¿Que representa el determinante? ¿Es posible que exista una dependencia funcional establecida entre dos subconjuntos de atributos de la cabecera de una relación que puedan ser a la vez determinantes y dependientes (obviamente en dos Dependencias Funcionales distintas)? Ejemplifique.