INSTITUTO TECNOLOGICO DE ESTUDIOS SUPERIORES DE LOS CABOS

MATERIA:

Administración de base de datos

TRABAJO:

Arquitectura del SGBD

DOCENTE:

MSC. Lilia Ureña Lugo.

GRUPO:

6ISC01V

ESTUDIANTE(S):

Sanchez Arroyo Edgar

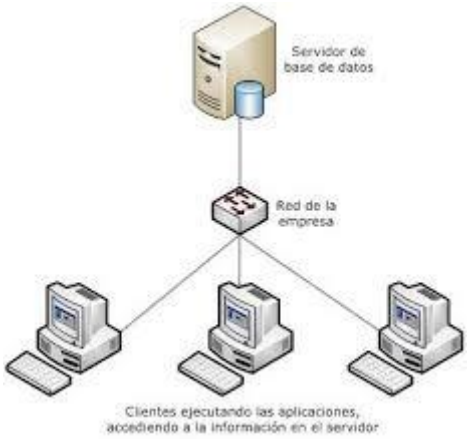
Fecha: 31/05/24

La arquitectura de SQL Server:

SQL Server es un sistema gestor de bases de datos desarrollado por la compañía de Microsoft, este sistema gestor sigue la arquitectura de cliente-servidor, lo que facilita la gestión eficiente de bases de datos esto al poder permitir múltiples conexiones de manera simultánea y proporcionar servicios centralizados de gestión de datos.

El servidor de SQL Server manera la gestión de las bases de datos y se comunica con los clientes por medio del protocolo TDS (Tabular Data Stream). Dentro de esto pueden existir múltiples clientes conectándose al servidor de manera simultánea, aparte tenemos que SQL Server puede ejecutarse en clúster lo que permite una alta disponibilidad.

Para que se quede más clara cómo es la arquitectura cliente servidor, podemos observar la siguiente imagen que expresa lo que es cliente-servidor, que básicamente es una comunicación entre el servidor principal y los clientes de la empresa dueña del servidor:



El núcleo de la arquitectura de SQL server consta de diversos componentes clave:

- **Motor de Base de Datos:**

El servidor de SQL Server incluye el Motor de Base de Datos, el cual es fundamental para procesar las consultas y gestionar todas las interacciones con las bases de

datos. Este motor de base de datos es altamente eficiente, proporcionando una administración robusta de transacciones, recuperación ante desastres y mantenimiento de la integridad de los datos. Además, SQL Server admite modos de instancia única o en clúster, lo que permite configurar el sistema para garantizar alta disponibilidad y escalabilidad. Esto significa que el servidor puede manejar grandes volúmenes de datos y altas cargas de trabajo sin comprometer el rendimiento. Las capacidades de clúster también permiten minimizar el tiempo de inactividad y asegurar la continuidad del negocio, lo cual es crucial para aplicaciones empresariales críticas.

- **Protocolo TDS (Tabular Data Stream):**

Como bien se mencionó anteriormente la comunicación entre clientes y el servidor se realiza a través del protocolo TDS, el cual facilita la transferencia eficiente de datos entre aplicaciones y la base de datos.

SQL Server como se mencionó con anterioridad permite la conexión de múltiples clientes al mismo tiempo, Cada conexión es de manera autentica y aparte se gestiona de manera segura, lo cual garantiza la integridad y seguridad de los datos.

Escalabilidad:

Otro punto muy importante a destacar es que la arquitectura de SQL Server es altamente escalable, lo que significa que puede adaptarse a entornos de bases de datos en crecimiento. Se pueden agregar recursos y configurar clústeres para satisfacer las demandas de cargas de trabajo cada vez mayores.

En si la arquitectura de SQL Server de cliente-servidor proporciona una plataforma robusta y escalable para una gestión eficiente de bases de datos. Con la capacidad para gestionar múltiples conexiones, su modo de instancia y sus protocolos

eficientes, logrando que SQL Server se logre destacar como una opción líder en sistemas de gestión de bases de datos relacionales.

La estructura lógica de SQL Server:

La estructura lógica de SQL Server es la que define cómo se organizan y gestionan los datos dentro del sistema de gestión de bases de datos. Comprender esta estructura es fundamental para diseñar y administrar eficazmente las bases de datos en entornos de SQL Server.

Entonces pasaremos a los componentes principales que la conforman:

Base de Datos:

La base de datos es la unidad fundamental de organización de datos en SQL Server.

Contiene tablas, vistas, procedimientos almacenados y otros objetos relacionados con los datos.

Puede haber múltiples bases de datos alojadas en un servidor de SQL Server, cada una independiente y aislada de las demás.

Una base de datos se puede comprender como discos donde hay información almacenada que se visualiza en forma de tablas.

Tablas:

Las tablas son estructuras que almacenan datos en filas y columnas.

Cada tabla está compuesta por columnas que representan atributos y filas que contienen los datos reales.

Las relaciones entre tablas se definen mediante claves primarias y foráneas para garantizar la integridad referencial.

Results Messages

	CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
1	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	NULL	12209	Germany	030-0074321	030-0
2	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.	NULL	05021	Mexico	(5) 555-4729	(5) 555
3	ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	NULL	05023	Mexico	(5) 555-3932	NULL
4	AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	NULL	WA1 1DP	UK	(171) 555-7788	(171) 5
5	BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8	Luleå	NULL	S-958 22	Sweden	0921-12 34 65	0921-
6	BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim	NULL	68306	Germany	0621-08460	0621-(
7	BLONP	Blondesddsl père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg	NULL	67000	France	88.60.15.31	88.60.
8	BOLID	Bólido Comidas preparadas	Martín Sommer	Owner	C/ Araquil, 67	Madrid	NULL	28023	Spain	(91) 555 22 82	(91) 55
9	BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Marseille	NULL	13008	France	91.24.45.40	91.24.
10	BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.	Tsawassen	BC	T2F 8M4	Canada	(604) 555-4729	(604) 5

Ejemplo de tablas en SQL server.

Vistas:

Las vistas son consultas SQL almacenadas que actúan como tablas virtuales. Las vistas tienen la misma apariencia de una tabla a la hora de visualizarlas

Proporcionan una capa adicional de abstracción sobre los datos, lo que permite simplificar consultas complejas y restringir el acceso a datos sensibles.

Procedimientos Almacenados:

Los procedimientos almacenados son bloques de código SQL que se almacenan en el servidor de base de datos.

Pueden aceptar parámetros de entrada y devolver resultados o realizar acciones en la base de datos.

Por último, estos proporcionan un mecanismo para encapsular lógica de negocio compleja dentro del servidor de base de datos con el que se esté trabajando o en donde se encuentre ubicado nuestro procedimiento almacenado.

Con todo esto podemos entender que la estructura lógica no solo para este sistema gestor en específico, es el que proporciona un marco organizado para los datos dentro del sistema gestor, que para este caso es SQL Server, lo que permite comprender y trabajar de manera mas eficiente con los elementos como los que vimos que son la base de datos como tal, las tablas, las vistas y los procedimientos almacenados, que son esenciales para el desarrollo de aplicaciones robustas, y poder mantener la integridad de los datos en entornos SQL Server.

Esta información es la que esta involucrada a la jora de crear bases de datos y realizar acciones correspondientes, divididas en definición y manipulación. En el caso de manipulación, se incluyen las siguientes acciones:

Creación de una tabla: Para almacenar datos en SQL Server, se utiliza la siguiente sintaxis básica con la instrucción CREATE TABLE:

Recordemos que las bases de datos como vimos anteriormente están echas de tablas, entonces en la creación de base de datos lo que se hace principalmente es crear tablas:

```
CREATE TABLE nombre_tabla (
    columna1 tipo_dato,
    columna2 tipo_dato,
    ...
);
```

Ejemplo:

```
CREATE TABLE usuarios (
    id INT PRIMARY KEY,
    nombre NVARCHAR(50),
    edad INT
);
```

A continuación, se especifica el significado de cada identificador y cláusula en la sentencia CREATE TABLE:

- nombre_tabla: Nombre de la nueva tabla.
- columna: Nombre de una columna de la tabla.
- tipo_dato: Tipo de datos de la columna.
- DEFAULT expr: Asigna un valor por defecto a la columna cuando no se especifica otro valor en una inserción.
- CONSTRAINT nombre_restricción: Nombre opcional para las restricciones sobre columnas y tablas.
- NOT NULL: Indica que la columna no puede contener valores nulos.
- NULL: Indica que la columna puede contener valores nulos si no se especifica NOT NULL.
- UNIQUE: Restricción que asegura valores únicos en una columna o grupo de columnas.
- PRIMARY KEY: Especifica la clave primaria de la tabla, asegurando valores únicos y no nulos.
- CHECK (expr): Permite incluir reglas de integridad específicas para cada fila insertada o actualizada, basadas en expresiones booleanas.

Inserción de datos: Para agregar datos a una tabla en SQL Server, se utiliza la instrucción INSERT INTO:

```
INSERT INTO nombre_tabla (columna1, columna2, ...)
VALUES (valor1, valor2, ...);
```

Ejemplo:

```
INSERT INTO usuarios (id, nombre, edad)
VALUES (1, 'Juan', 25);
```

Consulta de datos: Para recuperar información de una tabla en SQL Server, se utiliza la instrucción SELECT:

SELECT columna1, columna2, ...

FROM nombre_tabla

WHERE condicion;

En esta consulta, SELECT indica la acción de recuperar datos. Puedes usar * para seleccionar todas las columnas de la tabla especificada después de FROM. La cláusula WHERE filtra los resultados según la condición especificada.

Ejemplo:

SELECT nombre, edad

FROM usuarios

WHERE edad > 21;

Actualización de datos: Para modificar registros existentes, se utiliza la instrucción UPDATE:

UPDATE nombre_tabla

SET columna1 = nuevo_valor1, columna2 = nuevo_valor2, ...

Ejemplo:

WHERE condicion;

UPDATE usuarios

SET edad = 26

WHERE nombre = 'Juan';

Eliminación de datos: Para eliminar registros de una tabla, se utiliza la instrucción DELETE:

DELETE FROM nombre_tabla

WHERE condicion;

Ejemplo:

DELETE FROM usuarios

WHERE id = 1;

La estructura física de SQL Server:

La estructura física de SQL Server se centra en cómo los datos que se almacenan y gestionan a nivel del sistema operativo y del almacenamiento en disco. Este conocimiento es crucial para optimizar el rendimiento y garantizar la eficiencia en el acceso y manipulación de datos. Como bien veremos cuatro ejemplos de lo que hacemos referencia para poder saber a qué se refiere la estructura física de SQL Server.

Archivos de Datos y Log:

SQL Server utiliza archivos de datos (.mdf) para almacenar datos y archivos de registro (.ldf) para almacenar el registro de transacciones.

Estos archivos pueden distribuirse en unidades de almacenamiento físicas separadas para mejorar el rendimiento y la disponibilidad.

Almacenamiento en Páginas:

La unidad básica de almacenamiento físico en SQL Server es la página, que tiene un tamaño estándar de 8 KB.

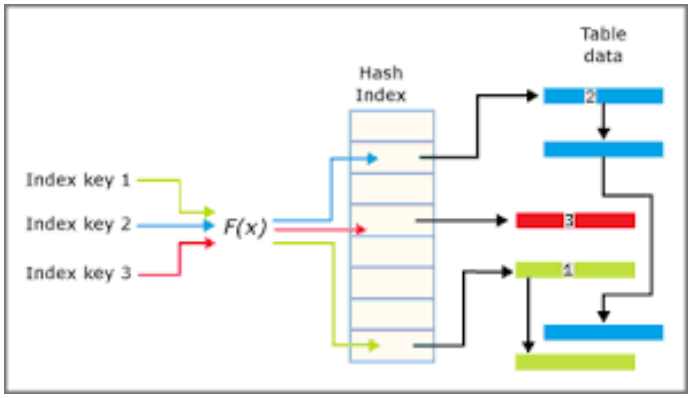
Los datos de la tabla se almacenan en estas páginas, y las páginas se leen y escriben en disco según sea necesario para satisfacer las operaciones de consulta y modificación.

Índices:

SQL Server utiliza índices para acelerar la recuperación de datos.

Los índices pueden ser almacenados en la misma unidad de almacenamiento que la tabla o en un grupo de archivos diferente.

Tipos de índices incluyen índices clustered, no clustered y columnstore, cada uno con características específicas para diferentes escenarios de consulta.



Lo anterior es una imagen que explica como es que funciona un índice, es un intermediario el cual funciona cuando realizamos consultas.

Gestión de Espacio:

SQL Server implementa un sistema de gestión de espacio que asigna y libera automáticamente espacio en disco para los datos y registros de transacciones.

El DBCC (Database Console Commands) CHECKDB se utiliza para mantener la integridad de la base de datos y gestionar la consistencia de los datos almacenados.

Por lo que la estructura física de SQL Server aborda lo que es la implementación concreta de los datos en el sistema operativo y en el almacenamiento en disco, la gestión eficiente de los archivos, las páginas, los índices y lo que seria el espacio

es algo esencial para garantizar un rendimiento optimo y una administración eficaz de bases de datos para entornos de SQL Server.

Arquitectura de MYSQL:

La arquitectura del Sistema de Gestión de Base de Datos (SGBD) MySQL se basa en el lenguaje de consulta SQL (Structured Query Language), donde los datos se organizan y almacenan en tablas dentro de un servidor. Cada tabla está compuesta por columnas y filas, permitiendo que la información sea consultada y gestionada mediante aplicaciones web, software de escritorio, o entornos de desarrollo integrado (IDE) como MySQL Workbench o dbForge. MySQL está diseñado para usuarios finales y ofrece robustez y flexibilidad en la manipulación de datos.

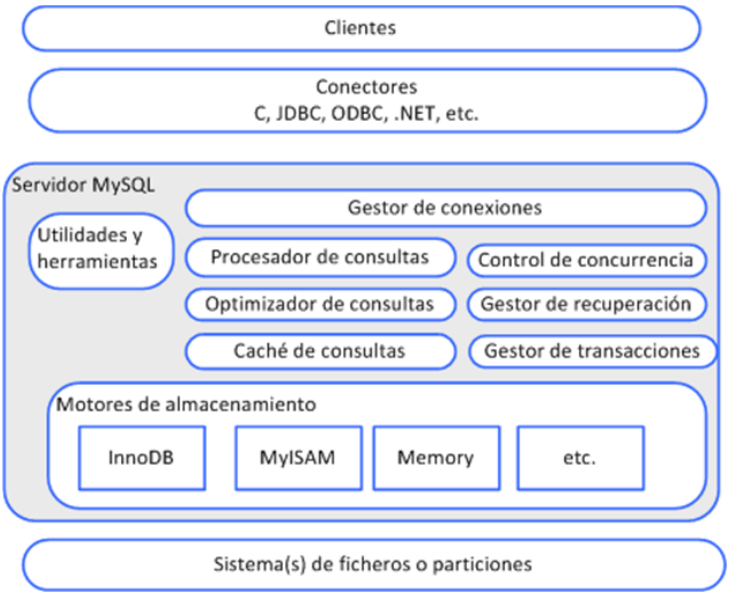
En términos generales, un sistema de base de datos relacional como MySQL consta de tres capas fundamentales:

Capa de Aplicación o Cliente: Interfaz a través de la cual los usuarios interactúan con la base de datos para realizar consultas, actualizaciones y gestión de datos.

Capa de Lógica o Servidor: Encargada de procesar las solicitudes de los clientes, ejecutar consultas SQL, aplicar reglas de negocio y gestionar la seguridad y la integridad de los datos. MySQL utiliza un motor de almacenamiento específico para manejar estas operaciones de manera eficiente.

Capa Física o de Almacenamiento: Donde se almacenan físicamente los datos en disco, organizados en estructuras optimizadas para la recuperación rápida y eficiente. Esta capa incluye índices, archivos de datos y estructuras de almacenamiento que optimizan el rendimiento de las consultas.

MySQL no solo proporciona una plataforma robusta para la gestión de datos, sino que también es altamente escalable y adaptable a diferentes necesidades de aplicación y carga de trabajo.



Iremos viendo capa por capa de la arquitectura del gestor para cubrir en orden los temas:

MySQL utiliza el lenguaje de consulta estructurado (SQL) para interactuar con la base de datos, permitiendo realizar operaciones como inserción, consulta, actualización y eliminación de datos. Este sistema gestor se fundamenta en el modelo relacional, donde los datos y sus relaciones se representan mediante conjuntos de tablas. Cada tabla está compuesta por un conjunto de columnas con nombres únicos que representan los atributos de los datos.

Dentro del contexto de las tablas en MySQL, los elementos principales son:

Atributos: Corresponden a las columnas que describen las características específicas de los datos.

Tuplas: Son conjuntos de filas que representan los registros individuales de la tabla. Cada tupla se visualiza horizontalmente en la tabla.

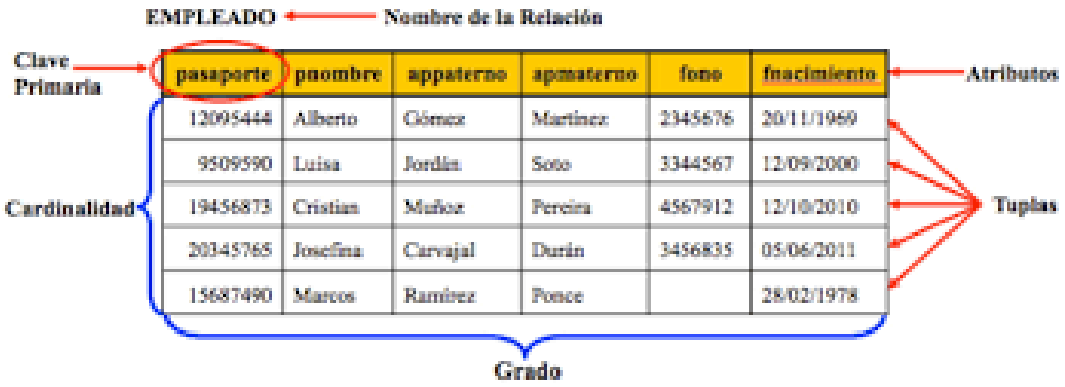
Relaciones: Representan las uniones entre tablas, identificadas por nombres que definen estas conexiones.

Claves: Son valores únicos utilizados para identificar de manera única cada registro dentro de una tabla.

Además, algunos conceptos clave en el contexto de las tablas en MySQL incluyen:

Cardinalidad: Se refiere al número de registros o tuplas que existen en una tabla específica.

Grado: Indica el número de atributos o columnas dentro de una tabla determinada.



pasaporte	pnombre	appaterno	apmaterno	fono	fnacimiento
12095444	Alberto	Gómez	Martínez	2345676	20/11/1969
9509590	Luisa	Jordán	Soto	3344567	12/09/2000
19456873	Cristian	Muñoz	Pereira	4567912	12/10/2010
20345765	Josefina	Carvajal	Durán	3456835	05/06/2011
15687490	Marcos	Ramírez	Ponce		28/02/1978

Con toda la información anterior es con la que se crean las bases de datos y se realizan acciones correspondientes, divididas en definición y manipulación. En el caso de manipulación, se incluyen las siguientes acciones:

Creación de una tabla: Para almacenar datos en MySQL, primero se debe crear una tabla utilizando la siguiente sintaxis básica:

```
CREATE TABLE nombre_tabla (
    columna1 tipo_dato,
    columna2 tipo_dato,
    ...
);
```

Ejemplo con datos:

```
CREATE TABLE usuarios (
    id INT PRIMARY KEY,
    nombre VARCHAR(50),
    edad INT
);
```

Luego, se especifica el significado de cada identificador y cláusula en la sentencia CREATE TABLE:

- nombre_tabla: Nombre de la nueva tabla.
- nombre_columna: Nombre de una columna de la tabla.
- tipo_dato: Tipo de datos de la columna.
- DEFAULT expr: Asigna un valor por defecto a la columna cuando no se especifica otro valor en una inserción.
- CONSTRAINT nombre_restricción: Nombre opcional para las restricciones sobre columnas y tablas.
- NOT NULL: Indica que la columna no puede contener valores nulos.
- NULL: Indica que la columna puede contener valores nulos si no se especifica NOT NULL.
- UNIQUE: Restricción que asegura valores únicos en una columna o grupo de columnas.
- PRIMARY KEY: Especifica la clave primaria de la tabla, asegurando valores únicos y no nulos.
- CHECK (expr): Permite incluir reglas de integridad específicas para cada fila insertada o actualizada, basadas en expresiones booleanas.

Inserción de datos: Para agregar datos a una tabla, se utiliza la instrucción INSERT INTO:

```
INSERT INTO nombre_tabla (columna1, columna2, ...)
VALUES (valor1, valor2, ...);
```

Ejemplo:

```
INSERT INTO usuarios (id, nombre, edad)
```

VALUES (1, 'Juan', 25);

Consulta de datos: Para recuperar información de una tabla en SQL, se utiliza SELECT:

```
SELECT columna1, columna2, ...
FROM nombre_tabla
WHERE condicion;
```

En esta consulta, SELECT indica la acción de recuperar datos. Puedes usar * para seleccionar todas las columnas de la tabla especificada después de FROM. La cláusula WHERE filtra los resultados según la condición especificada.

Actualización de datos: Para modificar registros existentes, se utiliza UPDATE:

```
UPDATE nombre_tabla
SET columna1 = nuevo_valor1, columna2 = nuevo_valor2, ...
WHERE condicion;
```

Eliminación de datos: Para eliminar registros de una tabla, se usa DELETE:

```
DELETE FROM nombre_tabla
WHERE condicion;
```

Capa cliente:

En esta capa, el cliente realiza solicitudes a través de la interfaz gráfica del sistema gestor. Si la sintaxis es correcta, los servicios involucrados incluyen:

Manejo de la conexión: El cliente inicia una solicitud de conexión al servidor, que luego acepta y establece la conexión. Una vez conectado, el cliente obtiene un hilo para realizar consultas. Por ejemplo, una aplicación Java se conecta al servidor MySQL utilizando JDBC, lo que implica cargar el controlador JDBC, establecer la URL de conexión y proporcionar nombre de usuario y contraseña.

Autenticación de usuario y contraseña: Antes de establecer la conexión, el cliente debe proporcionar credenciales (nombre de usuario y contraseña). Esto es crucial para garantizar que solo los usuarios autorizados accedan a la base de datos.

Seguridad: En este nivel, la seguridad se enfoca en verificar la identidad del usuario y garantizar que solo aquellos con las credenciales correctas puedan acceder y realizar operaciones en la base de datos.

Este enfoque asegura que las conexiones sean seguras y que solo los usuarios autenticados puedan interactuar con la base de datos MySQL, manteniendo así la integridad y la seguridad de los datos.

Gestor de conexiones:

La gestión de conexiones es fundamental para manejar múltiples conexiones de clientes de manera eficiente. Un gestor de conexiones eficaz no solo permite la conexión de clientes, sino que también gestiona el uso de recursos evitando la creación y destrucción frecuente de conexiones, que son procesos costosos. Se puede configurar un límite de conexiones concurrentes para evitar el uso excesivo de recursos. Además, se puede implementar un pool de conexiones que administre un conjunto predefinido de conexiones, permitiendo su reutilización y controlando su tiempo de vida.

Cache de consultas

MySQL implementa un caché de consultas para mejorar el rendimiento al almacenar consultas y sus resultados. Cuando un cliente envía una consulta, el servidor verifica si existe en la caché antes de realizar el análisis y la optimización:

Si la consulta está en el caché, se omite el procesamiento completo y se devuelve el resultado almacenado en la caché, mejorando significativamente el tiempo de respuesta.

Este enfoque ayuda a reducir la carga del servidor al evitar la reevaluación de consultas frecuentemente solicitadas, mejorando así la eficiencia del sistema globalmente.

Motores de almacenamiento:

Los motores de almacenamiento de la arquitectura MYSQL la hace única y la más preferible para los desarrolladores. Debido a esta capa, la capa MYSQL se cuenta como el RDBMS más utilizado y se usa ampliamente. En el servidor MYSQL, para diferentes situaciones y requisitos, se utilizan diferentes tipos de motores de almacenamiento, que son InnoDB, MYiSAM, NDB, Memory, etc. Estos motores de almacenamiento se utilizan como motores de almacenamiento conectables donde las tablas creadas por el usuario se conectan con ellos.

Son elementos muy notables debido a que estos motores son reemplazables (pluggable storage engine architecture).El objetivo de esa arquitectura es hacer una interfaz abstracta con funciones comunes de gestión de datos en el nivel físico. De ese modo, el gestor de almacenamiento puede intercambiarse, e incluso un mismo servidor MySQL puede utilizar diferentes motores de almacenamiento para diferentes bases de datos o para diferentes tablas en la misma base de datos. Otorgando la ventaja de utilizar el motor que mejor se adapte a nuestras necesidades.

También permite que terceros puedan implementar motores de almacenamiento nuevos para necesidades específicas, o adaptar el código de los existentes a ciertos requisitos de almacenamiento. Así, las interfaces definidas por MySQL aíslan el resto de los componentes de la arquitectura de las complejidades de la gestión física de datos, facilitando el mantenimiento de los motores de almacenamiento. También esto permite que ciertos motores de almacenamiento no implementen parte de los servicios, lo cual les hace inapropiados para algunas aplicaciones, pero más eficientes para otros.

Por ejemplo, un motor de almacenamiento que no implementa bloqueos en la base de datos no debe utilizarse en aplicaciones multi-usuario, pero la ausencia de sobrecarga de procesamiento en la gestión de los bloqueos para el acceso concurrente lo hará mucho más eficiente para una aplicación monousuario. Por esto, una primera tarea de diseño físico en MySQL es la de decidir el motor de almacenamiento más apropiado.

Parte física de MySQL:

Junto con la parte lógica, la parte física de MySQL se compone del registro en memoria, que incluye varios componentes esenciales:

Archivos de datos: Contienen los datos internos de la base de datos, organizados en páginas de 8KB, que son la unidad mínima de almacenamiento. Entre los tipos de páginas se encuentran:

Páginas de datos: Almacenan los registros de datos principales.

Páginas de espacio libre (PFS - Page Free Space): Mantienen información sobre la ubicación y tamaño del espacio libre disponible.

Páginas GAM y SGAM: Utilizadas para gestionar extensiones de archivos.

Páginas de Mapa de Asignación de Índices (IAM - Index Allocation Map): Contienen detalles sobre la ubicación de páginas de tablas o índices específicos.

Páginas de índices: Almacenan registros de índices para optimizar la búsqueda y acceso a datos.

Archivo de Registro de Transacciones: Este archivo tiene como objetivo facilitar la recuperación de datos hasta un momento específico en el tiempo o complementar una restauración completa desde una copia de seguridad.

Fuentes:

rwestMSFT. (s/f). *Guía de arquitectura de páginas y extensiones*. Microsoft.com. Recuperado el 24 de febrero de 2024, de https://learn.microsoft.com/eses/sql/relational-databases/pages-and-extents-architectureguide?view=sql-server-ver16

MashaMSFT. (s/f). *Documentación técnica de SQL Server*. Microsoft.com. Recuperado el 24 de febrero de 2024, de https://learn.microsoft.com/eses/sql/sql-server/?view=sql-server-ver16

De dbamemories, V. T. las E. (2011, julio 12). *Arquitectura de Bases de Datos SQL Server*. Memorias de un DBA. https://dbamemories.wordpress.com/2011/07/11/arquitectura-de-basesde-datos-sql-server/

.1.2 Estructuras físicas de la base de datos. (2017, marzo 23). *Blogspot.com*. http://blogjazz21.blogspot.com/2017/03/212-estructuras-fisicas-de-la-base-de.html

Arquitectura de MySQL. (2019). Studocu.com.

https://www.studocu.com/latam/document/universidad-nacionalagraria/administracion-de-empresas/pdf-arquitectura-de-mysql-compress/35554719